

Department of Computing Sciences

Sketch-based Digital Storyboards and Floor Plans for Authoring Computer-Generated Film Pre-visualisations

Timothy Matthews

Supervisors: Dr. D. Vogts and Mr. K.A. Naudé

December 2012

Submitted in fulfilment of the requirements for the degree of
Magister Scientiae in the Faculty of Science
at the Nelson Mandela Metropolitan University

Declaration

I, Timothy Matthews, hereby declare that the dissertation for the degree Magister Scientiae is my own work and that it has not previously been submitted for assessment or completion of any postgraduate qualification to another University or for another qualification.

Timothy Matthews

T. Matthews

Acknowledgements

I would like to thank my supervisors, Dr. Dieter Vogts and Mr. Kevin Naudé, for supporting, advising and encouraging me throughout my research project. I would also like to thank my supervisors for reading the various versions of my dissertation and providing valuable feedback.

I would also like to thank the staff of the Department of Computing Sciences for providing me with valuable feedback during my research and the staff of Triggerfish Animations Studios for making the interview possible.

I wish to thank Dr. Dieter Vogts for providing me with the equipment necessary to complete this research. Finally, I would like to thank the National Research Foundation for supporting this research financially.

Summary

Pre-visualisation is an important tool for planning films during the pre-production phase of filmmaking. Existing pre-visualisation authoring tools do not effectively support the user in authoring pre-visualisations without impairing software usability. These tools require the user to either have programming skills, be experienced in modelling and animation, or use drag-and-drop style interfaces. These interaction methods do not intuitively fit with pre-production activities such as floor planning and storyboarding, and existing tools that apply a storyboarding metaphor do not automatically interpret user sketches.

The goal of this research was to investigate how sketch-based user interfaces and methods from computer vision could be used for supporting pre-visualisation authoring using a storyboarding approach. The requirements for such a sketch-based storyboarding tool were determined from literature and an interview with Triggerfish Animation Studios.

A framework was developed to support sketch-based pre-visualisation authoring using a storyboarding approach. Algorithms for describing user sketches, recognising objects and performing pose estimation were designed to automatically interpret user sketches. A proof of concept prototype implementation of this framework was evaluated in order to assess its usability benefit.

It was found that the participants could author pre-visualisations effectively, efficiently and easily. The results of the usability evaluation also showed that the participants were satisfied with the overall design and usability of the prototype tool. The positive and negative findings of the evaluation were interpreted and combined with existing heuristics in order to create a set of guidelines for designing similar sketch-based pre-visualisation authoring tools that apply the storyboarding approach.

The successful implementation of the proof of concept prototype tool provides practical evidence of the feasibility of sketch-based pre-visualisation authoring. The positive results from the usability evaluation established that sketch-based interfacing techniques can be used effectively with a storyboarding approach for authoring pre-visualisations without impairing software usability.

Keywords: Pre-visualisation, Computer Graphics, Computer Vision, Sketch-based Interfaces, Multi-touch Interfaces

Table of Contents

- Declarationi**
- Acknowledgements ii**
- Summary iii**
- Table of Contents.....iv**
- List of Figuresx**
- List of Tablesxv**
- List of Abbreviationsxvi**
- Mathematical Notation.....xviii**
- Chapter 1: Introduction 1**
 - 1.1 Background 1
 - 1.2 Situation of Concern.....3
 - 1.3 Thesis Statement4
 - 1.4 Research Objectives4
 - 1.5 Research Questions4
 - 1.6 Research Methodology5
 - 1.7 Scope and Constraints6
 - 1.8 Significance.....7
 - 1.9 Dissertation Outline.....8
- Chapter 2: Pre-production 10**
 - 2.1 Introduction..... 10
 - 2.2 Filmmaking Process 11
 - 2.3 Analysing the Script 13
 - 2.4 Staging 14
 - 2.5 Storyboarding..... 17
 - 2.5.1 How Storyboards Are Used 18
 - 2.5.2 The Storyboarding Process 18
 - 2.5.3 Storyboard Detail 20
 - 2.5.4 Annotations and Graphical Devices Used in Storyboarding 22

2.5.5	Storyboarding Software	25
2.5.6	Discussion	28
2.6	Pre-visualisation	29
2.6.1	Textual Authoring Approach	30
2.6.1.1	Game-engine Code	31
2.6.1.2	Control Languages	31
2.6.2	Graphical Authoring Approach	33
2.6.2.1	Director Notation	33
2.6.2.2	Modelling and Animation Tools	35
2.6.2.3	Simplified Animation Tools	36
2.6.2.4	Sketch-based Storyboarding Tools	39
2.6.3	Discussion	41
2.7	Case study: Pre-visualisation for animated films	43
2.7.1	Background of Triggerfish Animation Studios	43
2.7.2	Interview Methodology	44
2.7.3	Outcomes of the interview	44
2.7.3.1	The Filmmaking Process Used at Triggerfish	45
2.7.3.2	Storyboarding at Triggerfish	46
2.7.3.3	Pre-visualisation at Triggerfish	47
2.7.3.4	Problems and Areas open to Improvement	48
2.7.3.5	Requirements of a Sketch-based Pre-visualisation Authoring Tool	49
2.8	Summary of Sketch-based Pre-visualisation Requirements	50
2.9	Conclusions	52
Chapter 3: Understanding Sketches		55
3.1	Introduction	55
3.2	The Object Recognition Process	56
3.3	Describing Images	57
3.3.1	Local Features and Global Features	58
3.3.2	Local Feature Detectors	58
3.3.2.1	Corner Detectors	59
3.3.2.2	Region Detectors	60
3.3.2.3	Edge Detectors	62
3.3.2.4	Discussion	64
3.3.3	Local Feature Descriptors	65

3.3.3.1	SIFT-based Descriptors	66
3.3.3.2	Grid-based Descriptors	68
3.3.3.3	Shape-based Descriptors	69
3.3.3.4	Summary of Local Region Descriptors for Sketches	71
3.3.4	Global Feature Descriptors	72
3.3.5	Discussion	73
3.4	Comparing Images	74
3.4.1	Feature Matching	74
3.4.2	Classification Approaches	75
3.4.2.1	Naïve Bayesian	77
3.4.2.2	Artificial Neural Networks	77
3.4.2.3	Support Vector Machines	78
3.4.2.4	Decision Trees	79
3.4.2.5	<i>k</i> -Nearest Neighbours	79
3.4.2.6	Discussion	79
3.5	Rigid Body Pose Estimation	80
3.5.1	Recognising Poses	81
3.5.2	Analytical Pose Estimation	82
3.5.3	Numerical Pose Estimation	83
3.5.3.1	POSIT	84
3.5.3.2	Iterative Closest Point (ICP) Registration	86
3.5.3.3	Discussion	87
3.6	Articulated Body Pose Estimation	88
3.6.1	Vision-based Posing Methods	89
3.6.2	Sketch-based Posing Methods	90
3.7	Conclusions	92
Chapter 4: Design and Implementation		94
4.1	Introduction	94
4.2	A Generalised Framework for Authoring Pre-visualisations	95
4.3	A Framework for a Sketch-based Storyboarding GUI	97
4.4	Data Collection	98
4.5	Graphical User Interface Design	100
4.5.1	Look and Feel	101
4.5.2	Interface Layout and Navigation	102

4.5.3	Story Viewer and Script Viewer	104
4.5.4	Storyboard Editor	104
4.5.5	Floor Plan Editor	105
4.5.6	Sketch Editor.....	108
4.5.6.1	Sketching New Shots.....	108
4.5.6.2	Adding Characters.....	110
4.5.6.3	Adjusting Shots	112
4.5.7	Discussion.....	113
4.6	Algorithm Design.....	114
4.6.1	Requirements for Interpreting Sketches	114
4.6.2	Raster Representations and Vector Representations	115
4.6.2.1	Sampling Edgels from a Raster Image	115
4.6.2.2	Sampling a Raster Image from Edgels	117
4.6.3	Approaches for Describing Sketches	118
4.6.3.1	Contour-based Approach.....	118
4.6.3.2	Part-based Approach	120
4.6.3.3	Grid-based Approaches	122
4.6.3.4	Sampling Multiple Disks.....	123
4.6.3.5	Combined Approach	125
4.6.3.6	Discussion.....	129
4.6.4	Recognising Objects from Sketches.....	129
4.6.4.1	Training Phase	130
4.6.4.2	Recognition Phase.....	131
4.6.4.3	Discussion.....	132
4.6.5	Pose Estimation.....	133
4.6.5.1	Estimating the Pose of Rigid Bodies with an Unknown Camera	133
4.6.5.2	Estimating the Pose of Articulated Figures and Rigid Bodies with a Known Camera	135
4.6.5.3	Discussion.....	138
4.6.6	Camera Estimation	138
4.7	Implementation	140
4.7.1	Implementation Tools.....	141
4.7.2	Third Party Libraries Used	142
4.7.3	Component Implementation	142
4.7.3.1	Computer Graphics Module	142

4.7.3.2	Component Model.....	143
4.7.3.3	Multi-touch Module	144
4.7.3.4	Computer Vision Module	144
4.8	Quantitative Performance Evaluation.....	145
4.8.1	Methodology.....	145
4.8.2	Performance Results.....	146
4.8.3	Discussion.....	147
4.9	Conclusions.....	148
Chapter 5: Usability Evaluation.....		149
5.1	Introduction.....	149
5.2	Existing Evaluation Methods.....	150
5.3	Selection of Evaluation Methods	151
5.4	Analytical Evaluation Design	151
5.5	Analytical Evaluation Results	152
5.5.1	Nielsen’s Heuristics for the GUI Design.....	152
5.5.2	Heuristic Evaluation for Touch-based Interaction with a 2D/3D Interface.....	154
5.5.3	Heuristic Evaluation for Sketch-based Recognition	155
5.5.4	Usability Issues	156
5.6	Empirical Evaluation Design	156
5.6.1	Participants.....	157
5.6.2	Evaluation Metrics	158
5.6.3	Instruments.....	159
5.6.4	Tasks.....	159
5.6.5	Experimental Setup and Procedure	160
5.7	Empirical Evaluation Results.....	161
5.7.1	Performance Results.....	161
5.7.1.1	Task Success Rate	161
5.7.1.2	Errors	162
5.7.1.3	Time on Task	163
5.7.1.4	Learnability.....	164
5.7.1.5	Efficiency.....	165
5.7.2	Satisfaction Results	169
5.7.2.1	Workload	169
5.7.2.2	Overall Satisfaction	170

5.7.2.3	Usability Results	171
5.7.2.4	Qualitative Feedback.....	171
5.7.3	Observations	173
5.8	Design Guidelines	176
5.9	Conclusions.....	179
Chapter 6: Conclusions.....		180
6.1	Introduction.....	180
6.2	Achievement of Research Objectives.....	180
6.2.1	Achievements.....	181
6.2.2	Summary.....	185
6.3	Research Contribution	186
6.3.1	Theoretical Contributions	186
6.3.2	Practical Contributions	188
6.4	Limitations and Problems Encountered.....	188
6.5	Future Research.....	188
References		190
Appendix A : An Interview with Triggerfish Animation Studios		204
Appendix B : XML Data Format Examples		221
Appendix C : GUI Touch Gestures		222
Appendix D : Ethics Approval (Application).....		224
Appendix E : Ethics Approval (Letter).....		229
Appendix F : Heuristics Checklist		230
Appendix G : Informed Consent Form		231
Appendix H : Task List.....		233
Appendix I : Pre-Task Questionnaire.....		241
Appendix J : Performance Sheet.....		242
Appendix K : Post-Task Questionnaire (Staging).....		243
Appendix L : Post-Task Questionnaire (Storyboarding)		245

List of Figures

Figure 1.1 The flow of the major phases of the filmmaking process.	1
Figure 1.2: An example of a partial storyboard.	2
Figure 1.3: The outline of the dissertation showing chapters, research questions and research methods.	8
Figure 2.1: An overview of Chapter 2.	10
Figure 2.2: The filmmaking process, tools and aspects that need planning.	11
Figure 2.3: The script specifying character action and dialogue	13
Figure 2.4: A floor plan of the set	15
Figure 2.5: A floor plan for a dramatic block showing character blocking	16
Figure 2.6: A floor plan for a dramatic block showing character blocking and shots.	17
Figure 2.7: A flow diagram illustrating the storyboarding process.	19
Figure 2.8: Examples of increasingly detailed character sketches	20
Figure 2.9: Expressing character emotion by sketching faces.	21
Figure 2.10: The three-point lighting approach	21
Figure 2.11: Examples of increasing detailed storyboard sketches	22
Figure 2.12: Graphical devices that illustrate object state.	23
Figure 2.13 Graphical devices and annotations for (a-c) movement and (d) motion paths.	23
Figure 2.14: Graphical devices for illustrating character feelings and emotions.	24
Figure 2.15 Annotations for illustrating camera behaviour.	24
Figure 2.16 Annotations for illustrating transitions between shots.	25
Figure 2.17: A screenshot of Storyboard Quick.	27
Figure 2.18: A screenshot of Storyboard Pro.	28
Figure 2.19: Pre-visualisation authoring approaches.	30
Figure 2.20: A diagram depicting the SAIBA framework.	31

Figure 2.21: Illustrating a scene using (a) a storyboard frame, and (b) Director Notion.	34
Figure 2.22: Rigging and posing a character	36
Figure 2.23: A screenshot of iClone 5.....	38
Figure 2.24: The basic activities involved in each part of the pre-production phase.....	39
Figure 2.25: A screenshot of Storyboard Pro 3D	40
Figure 2.26: How authoring approaches improve.....	41
Figure 2.27: A scene from Zambezia	43
Figure 2.28: The interviewing methodology used.	44
Figure 2.29: Comparing the traditional film making process with Triggerfish’s process.....	45
Figure 3.1: Bottom-up and top-down computer vision approaches	56
Figure 3.2: The general process followed by object recognition algorithms	57
Figure 3.3: Harris corners for a sketch of a character holding a book	60
Figure 3.4: An illustration of the scale space of an image.....	61
Figure 3.5: Local features with (a) blob structures and (b) arbitrary shapes for a sketch of a character holding a book	62
Figure 3.6: Indistinguishable edge responses.	63
Figure 3.7: Edge features for a sketch of a character holding a book	63
Figure 3.8: Sketching anomalies	64
Figure 3.9: An example of a textured image and an outlined image.....	65
Figure 3.10: SIFT-key detection process	66
Figure 3.11: The SIFT-key descriptor	67
Figure 3.12: Grid-based descriptors configured in rectangular and elliptical arrangements....	68
Figure 3.13: Describing shape contours using a of distance function histogram a Delaunay triangulation angle histogram.....	70
Figure 3.14: Chamfer matching for comparing shapes.	71
Figure 3.15: Using PCA to estimate the transformation parameters of a point set.....	73
Figure 3.16: The probabilistic image classification problem.....	76

Figure 3.17: The pose estimation problem	80
Figure 3.18: Indexing a model from sixteen reference views using a shape descriptor	81
Figure 3.19: A diagram showing the variables involved in solving a P3P problem using a pinhole camera model.....	82
Figure 3.20: The perspective projection and scaled orthographic projection of object points to image points.....	84
Figure 3.21: An illustration of the 2D case of the point registration problem.....	86
Figure 3.22: Pseudocode for the ICP registration algorithm	87
Figure 3.23: The process of sketching a character	88
Figure 3.24: Vision-based pose estimation using (a) a body plan and (b,c) part assembly	89
Figure 3.25: A direct sketch-based rigid body posing method.....	91
Figure 4.1: A generalised overview of frameworks for generating pre-visualisations	95
Figure 4.2: SISPA: A framework for sketch-based pre-visualisation authoring using a storyboarding approach.....	97
Figure 4.3: The Environmental Data Model.....	99
Figure 4.4: Example assets that form part of the environment.....	99
Figure 4.5: The Project Data Model.....	100
Figure 4.6: The Asus EEE Slate EP121 used for prototyping	101
Figure 4.7: Touch gestures for performing a (a) Tap (b) Horizontal flick/scroll (c) Vertical flick/scroll (d) Drag (e) Spread/enlarge (f) Pinch/shrink (g) rotate.....	102
Figure 4.8: GUI layout and navigation.....	103
Figure 4.9: A screenshot of the storyboard editor.....	105
Figure 4.10: A screenshot of the floor plan editor.....	106
Figure 4.11: Sketches with their matching objects.....	107
Figure 4.12: A screenshot of the Sketch Editor for sketching a new shot.....	109
Figure 4.13: A screenshot of the Sketch Editor for adding a character.....	111
Figure 4.14: Example facial expressions with their matching emotion symbols.....	112

Figure 4.15: A screenshot of the Sketch Editor for adjusting an existing shot.....	112
Figure 4.16: Sampling edgels from a raster image.....	116
Figure 4.17: Sampling a raster from edgels.	117
Figure 4.18: Overview of image descriptors investigated during the algorithm design process.	119
Figure 4.19: Part-based image description.	122
Figure 4.20: Describing a user sketch (a) using a grid-based approach (b-c).....	122
Figure 4.21: Approaches for sampling multiple disks from an image.	124
Figure 4.22: Pseudocode for describing objects in sketches.....	126
Figure 4.23: Pseudocode for measuring the dissimilarity of objects in sketches.....	127
Figure 4.24: Comparing two disks with four SIFT feature bags.....	128
Figure 4.25: The general process followed by object recognition algorithms	129
Figure 4.26: The training images used for recognising symbols.	130
Figure 4.27: The 3D model sampling method.	131
Figure 4.28: Pseudocode for the k-NN classifier used to classify query images.	132
Figure 4.29: The pose estimation problem as viewed from (a) top-down, and (b) the shot sketched by the user.....	133
Figure 4.30: The process of estimating the pose of a rigid body sketched by the user.	134
Figure 4.31: The coordinate systems involved for un-projecting an image point.	136
Figure 4.32: Posing props and characters with a known camera.	137
Figure 4.33: The camera estimation problem.	139
Figure 4.34: A comparison between two component models: Windows default and custom.	143
Figure 4.35: (a) Prop recognition scalability and (b) camera estimation scalability (n=10). .	146
Figure 4.36: Camera estimation accuracy (n=10).	147
Figure 5.1: Participant (a) background (b) storyboarding experience and (c) modelling and animation experience (n=10).	157

Figure 5.2: Participant experience with filmmaking activities.	158
Figure 5.3: Task success rates for (a) staging, and (b) storyboarding (n=10).	162
Figure 5.4: Errors made during (a) staging, and (b) storyboarding (n=10).	163
Figure 5.5: Time taken to complete non-trivial tasks (n=10).	164
Figure 5.6: Learnability results (n=10).....	165
Figure 5.7: Success per minute efficiency results for non-trivial tasks (n=10).	166
Figure 5.8: Physical effort efficiency results (n=10).....	168
Figure 5.9: The cognitive load using a 5-point semantic differential scale (n=10).	169
Figure 5.10: Overall satisfaction results using a 5-point Likert scale (n=10).....	170
Figure 5.11: Usability results using a 5-point Likert scale (n=10).....	171
Figure 6.1: The SISPA framework.....	187

List of Tables

Table 2.1: A comparative summary of Storyboard Quick and Storyboard Pro.	26
Table 2.2: A list of control languages.....	32
Table 2.3: A comparison of features provided by 3D modelling and animation software.....	35
Table 2.4: A comparison of features provided by simplified animation software.	37
Table 2.5: Elements used in floor plans.....	50
Table 2.6: Elements used in storyboards.	51
Table 2.7: A summary of the requirements for a sketch-based pre-visualisation authoring tool using a storyboarding approach.....	52
Table 3.1: A comparison of several corner detectors.....	59
Table 3.2: A comparison of several region detectors.....	60
Table 3.3: A summary of several region descriptors.....	72
Table 3.4: An overview of several classification approaches.....	76
Table 4.1: Important specifications of the Asus EEE EP121 tablet.	100
Table 4.2: Objects that are required be interpreted by the Computer Vision Module.	115
Table 4.3: The sketches used for quantitative performance evaluation.	145
Table 4.4: Quantitative performance evaluation hardware specifications.....	146
Table 5.1: Task grouping for measuring learnability.	165
Table 5.2: Calculation of the average efficiency for non-trivial tasks.	166
Table 5.3: Most positive comments for the staging interface.	172
Table 5.4: Most negative comments for the staging interface.	172
Table 5.5: Most positive comments for the storyboarding interface.....	173
Table 5.6: Most negative comments for the storyboarding interface.....	173
Table 5.7: Observed usability issues (n=10).....	174

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ANN	Artificial Neural Network
APML	Affective Presentation Markup Language
BEAT	Behaviour Expression Animation Toolkit
BML	Behaviour Markup Language
CG	Computer Graphics
CPT	Conditional Probability Tables
CV	Computer Vision
DFT	Discrete Fourier Transform
DN	Director Notation
DoG	Difference of Gaussian
EBR	Edge Based Regions
EBSR	Entropy Based Salient Regions
FML	Functional Markup Language
GLOH	Gradient Location-Orientation Histogram
GUI	Graphical User Interface
IBR	Intensity Based Regions
ICP	Iterative Closest Point
IK	Inverse Kinematics
KKT	Karush-Kuhn-Tucker
LoD	Level of Detail
MMH	Maximum Marginal Hyperplane
MSER	Maximally Stable Extremal Regions
MSML	Movie Script Mark-up Language
MURML	Multimodal Utterance Representation Markup Language
NURBS	Non-Uniform Rational Basis Spline
k -NN	k Nearest Neighbours
OCD	Orientated Chamfer Distance
OI	Orthogonal Iteration
POSIT	Pose from Orthography and Scaling with Iterations
PC	Personal Computer
PCA	Principal Component Analysis
PCA-SIFT	Principal Component Analysis-SIFT
PDF	Portable Document Format
PML	Player Markup Language
PnP	Perspective-n-Point
POS	Pose from Orthography and Scaling
RRL	Rich Representation Language

SAIBA	Situation, Agent, Intention, Behaviour, Animation
SIFT	Scale Invariant Feature Transform
SISPA	Storyboarding Interface for Sketch-based Pre-visualisation Authoring
SOP	Scaled Orthographic Projection
SSE	Sum of Squared Error
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol/Internet Protocol
UI	User Interface
WIMP	Windows Icons Menus Pointer
XML	Extensible Markup Language

Mathematical Notation

$X = \{x_0, x_1, \dots, x_n\}$	Set of elements x_0
$\sum_{i=0}^n x_i$	Summation of the elements x_i from $i = 0$ to $i = n$
$M_{n \times m}$	An n by m matrix
$\mathbf{v}_i = \bar{\mathbf{v}}_i = v_i$	The i^{th} vector
$\mathbf{P}_i = \bar{\mathbf{P}}_i = P_i$	The i^{th} point
$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$	Matrix
$R = [a \quad b]$	Row matrix
$C = \begin{bmatrix} a \\ b \end{bmatrix} = [a \quad b]^T$	Column matrix and transpose
$\hat{\mathbf{k}}$	Basis vector
$\overline{\mathbf{M}_0 \mathbf{M}_1} = \mathbf{M}_1 - \mathbf{M}_0$	Vector from point \mathbf{M}_0 and \mathbf{M}_1
$\bar{\mathbf{v}}_{i,j} = \bar{\mathbf{v}}(i,j)$	Vector \mathbf{v} is a function of $i, j \in \mathbb{Z}$
$M_{i,j} = M(i,j)$	Matrix \mathbf{M} is a function of $i, j \in \mathbb{Z}$
$\ \bar{\mathbf{v}}\ $	Norm of $\bar{\mathbf{v}}$ (e.g. Euclidian)
$O(mn)$	Big O notation
$x \leftarrow x + 1$	Variable assignment: value of variable x is replaced by $x + 1$

Chapter 1:

Introduction

1.1 Background

The filmmaking process has five major phases as shown in Figure 1.1 namely development, pre-production, production, postproduction and distribution. The development phase involves writing the script for the film and the pre-production phase involves preparation for film production. The film receives the necessary funding if the goals of the development and pre-production phases are met. If the film is approved for production then the filmmaking team begins shooting each scene. The postproduction phase involves editing and finishing the final film, which is then advertised and distributed during the distribution phase.



Figure 1.1 The flow of the major phases of the filmmaking process (Mamer 2008).

In particular, the pre-production phase is important because it involves planning the film and preparing for the production phase. Every person, prop and shot must be planned in advance so that the director can focus on actor performances (Tomaric 2010).

Being able to visualise the film during the pre-production phase is very important (Moviestorm 2011). This is achieved by using *pre-visualisation* tools. Pre-visualisation is useful for exploring various scene settings, actor behaviours and camera placements without incurring the cost of production (Jung *et al.* 2010). Traditional storyboarding is one of the most widely used pre-visualisation tools. Oxford (2010) defines a storyboard as “a sequence

of drawings, typically with some directions and dialogue, representing the shots planned for a movie or television production.”

An example of a storyboard is shown in Figure 1.2. It conveys information about what is happening in the current scene. Each storyboard panel illustrates background scenery, actor and prop placement, shot type and many other aspects that cannot be illustrated easily in a textual script. Traditional storyboards are static in the sense that they cannot be used to illustrate change continuously as time progresses. They only provide discrete snapshots of the events in a scene. Directors want pre-visualisation tools that provide support for dynamic content (Jung *et al.* 2010).



Figure 1.2: An example of a partial storyboard (Glebas 2008).

The simplest form of dynamic pre-visualisation is achieved by creating *animated storyboards*. Animated storyboards display storyboard sketches in succession with synchronised dialogue and audio. Camera movement is simulated by panning, rotating and zooming storyboard panels in two dimensional (2D) space. 2D animation is problematic if the scene requires accurate three dimensional (3D) planning.

3D computer generated animation is another approach to pre-visualisation (Chakravarthy *et al.* 2010; Ye and Baldwin 2008). 3D animations are becoming more popular for film pre-visualisation because they communicate a greater wealth of dynamic content more naturally when compared with traditional storyboards and animated storyboards.

Several commercial pre-visualisation authoring solutions are available, such as Autodesk Maya and 3D Studio Max (Autodesk 2011b; Autodesk 2011a). These tools require the author to work at a low and very technical level. Simplified animation tools such as FrameForge3D

are available to provide the user with an easy to use authoring environment (Innoventive Software 2009). Both 3D animation approaches employ drag-and-drop style user interfaces. The author needs to place individual characters and props on the virtual set and direct the behaviour of virtual actors and cameras using mouse-based interaction. Authoring pre-visualisations using drag-and-drop style user interfaces is a tedious, manual and non-intuitive process. Traditional storyboarding techniques can be used to quickly communicate what is happening in each scene.

One of the main limitations of the drag-and-drop style authoring approach is the chosen user interaction technique. Traditional storyboards are typically sketched by hand, whereas software alternatives require the user to use a mouse and keyboard. The storyboards themselves are rich in scene information, but little attempt has been made to create pre-visualisation tools that can extract this information automatically. Computer vision literature demonstrates that information can be extracted from sketches automatically. It has been shown how objects can be recognised from images and posed in 3D space (Shin and Igarashi 2007; Lee and Funkhouser 2008). It has also been shown how characters can be posed using sketches (Dementhon and Davis 1995; Chaudhuri, Kalra and Banerjee 2004).

Sketch-based storyboarding is a novel approach to authoring pre-visualisations. Limited research has been conducted on how pre-visualisation tools can be designed and implemented using methods from computer vision to automatically interpret the user's sketches (Jhala *et al.* 2008; Skorupski 2009). There is also a need to evaluate how effectively, efficiently and easily users can author pre-visualisations using this approach.

1.2 Situation of Concern

The following problem statement articulates the situation of concern:

Existing authoring software does not effectively support the authoring of pre-visualisations without impairing software usability.

Pre-visualisations are authored using software that requires users to specify scene and filming information using drag-and-drop style user interfaces (Jung *et al.* 2010). The authoring process is tedious and requires the user to manually provide information that could be conveyed easily using simple storyboards. Authoring systems that provide interfaces based on a storyboarding metaphor do not automate information extraction from user sketches sufficiently (Jhala *et al.* 2008). This research addresses the situation of concern by

investigating how a sketch/touch-based storyboarding approach can be applied in order to allow a user to author pre-visualisations effectively and easily.

1.3 Thesis Statement

The following thesis statement will be used to guide this research:

Storyboarding with sketch-based interfacing techniques can be used effectively for authoring pre-visualisations without impairing software usability.

Storyboarding is used to support decision-making during the pre-production phase of filmmaking (Mamer 2008). This is because each frame in a storyboard includes information about the camera in 3D space, the set, shot style, character identities, character action and behaviour, camera action, etc. Directors can use existing storyboarding skills to author pre-visualisations easily and quickly using a sketch-based interface that can interpret sketches automatically. This simplifies the pre-visualisation authoring process by automating the extraction of storyboard information instead of requiring the user to manually specify it using traditional authoring software.

1.4 Research Objectives

The objectives of this research are:

- O₁: To investigate how pre-visualisations are currently created by focusing on the methods and theories already developed.
- O₂: To investigate what methods are available for extracting information from sketches.
- O₃: To design and implement a sketch/touch-based storyboarding tool for authoring pre-visualisations.
- O₄: To measure the extent to which a sketch-based storyboarding interface supports the user in authoring pre-visualisations in terms of software usability.

1.5 Research Questions

The main research question addressed by this research is:

How can the authoring of pre-visualisations be supported using sketch-based digital storyboarding without impairing software usability?

The following sub-questions will be answered in order to answer this research question:

- Q₁: What methods and theories have been developed for authoring pre-visualisations?
- Q₂: What methods are available for extracting information from sketches?
- Q₃: How can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?
- Q₄: To what extent does a sketch-based storyboarding interface support the user in authoring pre-visualisations in terms of software usability?

1.6 Research Methodology

The positivism research philosophy with a deductive approach will be used for this research. The findings will be observed and described objectively without the interference of the researcher (Oates 2006; Levin 1988; Saunders, Lewis and Thornhill 2009). Several theories and practices have been developed in processing sketch-based input. This research will involve investigating the most applicable methods for authoring pre-visualisations using a storyboarding metaphor. The research methods used for answering the research questions are summarised below:

Q₁ What methods and theories have been developed for authoring pre-visualisations?

Literature studies provide critical reviews of specific research areas (Hofstee 2009). This research question will be answered by conducting a literature study in order to identify what methods and theories have been developed for authoring pre-visualisations. This will involve reviewing existing film-making processes and approaches for authoring pre-visualisations. An *interview* with experts from the computer animation industry will also be conducted in order to investigate how film-making and pre-visualisation is done in practice (Triggerfish Animation Studios 2012a).

Q₂ What methods are available for extracting information from sketches?

This research question will be answered by conducting a literature study in order to investigate methods for extracting information from sketches automatically. This involves reviewing methods for consistently identifying points or regions of interest known as features. Methods for describing, comparing and matching features will also be investigated in order to perform object recognition. The methods will be critically reviewed to determine the feasibility of each for automatically interpreting user

sketches (with reasonable processing times). Methods for posing rigid bodies and articulated bodies in 3D space will also be reviewed.

Q₃ How can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?

This research question will be answered by applying the design and implementation research strategy (Oates 2006). A framework will be created for relating the various components required for implementing a storyboarding sketch/touch-based Graphical User Interface (GUI) capable of automatically interpreting user sketches. A proof of concept prototype tool will be designed and implemented for demonstrating the feasibility of the methods and ideas presented in this research. The prototype will also be required for conducting an experiment in order to answer the next research question.

Q₄ To what extent does a sketch-based storyboarding interface support the user in authoring pre-visualisations in terms of software usability?

The experimental research strategy will be applied in order to answer this research question (Oates 2006). The usability of the prototype will be evaluated analytically and empirically. The analytical evaluation will measure the design of the GUI against several usability heuristics (Sharp, Rogers and Preece 2007). The empirical evaluation will involve conducting usability tests in order to collect quantitative and qualitative data from observations and questionnaires. The design and results of the usability experiment will be discussed in detail in Chapter 5.

1.7 Scope and Constraints

There are many components involved in authoring, planning and rendering pre-visualisations; however, the scope of the research will be limited to the design and implementation of the following:

- A framework for sketch-based pre-visualisation authoring using a storyboarding approach,
- Algorithms for recognising 2D objects and 3D objects,
- Algorithms for estimating the pose of rigid bodies and articulated bodies in 3D space,
- An algorithm for estimating the camera from a user sketch,
- A sketch/touch-based interface for authoring pre-visualisations using floor plans,

- A sketch/touch-based interface for sketching storyboard panels using the algorithms and ideas presented in this research, and
- A touch-based interface for creating and managing storyboards containing static pre-visualisation images.

There are two main approaches to creating pre-visualisations (Moviestorm 2011):

1. Minimalistic pre-visualisations make use of simple virtual environments and omit unnecessary animation.
2. Detailed pre-visualisations attempt to visualise each shot as closely to the final film as possible using detailed virtual environments and animations.

The focus of this research will be on authoring *minimalistic* pre-visualisations using a sketch/touch-based storyboarding interface. Animation will therefore be excluded from the scope of this research. The design and implementation of the planning and rendering components are not included. Furthermore, programmatically authoring pre-visualisations and performing natural language processing on film scripts are not included.

1.8 Significance

There are several problems with using traditional tools for authoring pre-visualisations. Existing pre-visualisation tools require programming skills or modelling and animation expertise. Simplified tools are available that use drag and drop approaches; however, the interaction method does not intuitively fit with pre-production activities such as writing and sketching. Existing tools that apply storyboarding approaches do not automatically interpret user sketches.

This research proposes methods for designing and implementing sketch-based pre-visualisation authoring tools that apply a storyboarding approach for interpreting user sketches automatically. The findings of the research are significant because it simplifies the transition from storyboarding to pre-visualisation. It also accelerates the creative process by allowing pre-visualisation artists to capitalise on existing storyboarding skills. The research findings are significant to modellers and animators because it allows them to begin working with 3D content earlier in the filmmaking process. The methods proposed by this research can be used in other research areas such as sketch-based modelling, animation and technical drawing.

1.9 Dissertation Outline

This chapter provided a background to the research problem. It discussed the situation of concern and articulated the research design by making a thesis statement, asking research questions, stating research objectives and discussing the research methodology, scope and significance. An outline of the dissertation is given in Figure 1.3.

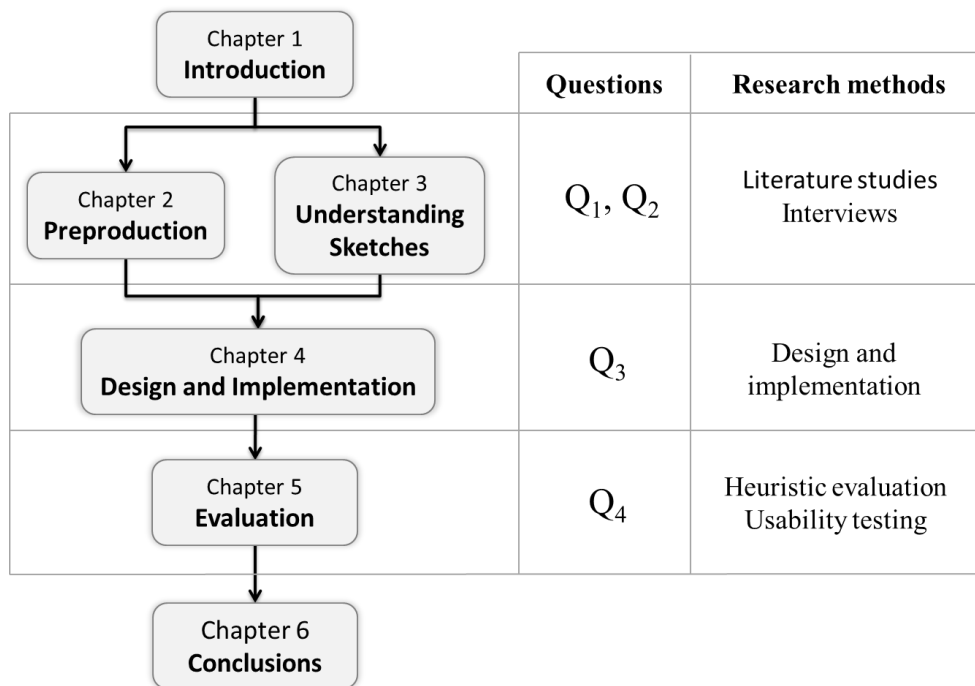


Figure 1.3: The outline of the dissertation showing chapters, research questions and research methods.

Chapter 2 and Chapter 3 cover the literature study component of this research. Chapter 2 provides an overview of the filmmaking process as well as several pre-production activities. These include analysing the script, staging, storyboarding and pre-visualisation. It provides the necessary background for designing a pre-visualisation authoring tool that aligns with existing filmmaking processes, activities and deliverables. A detailed review on storyboarding is provided which discusses the storyboarding process, what details are illustrated, what annotations are used and what software is available for storyboarding. Approaches for authoring pre-visualisations are discussed in order to determine their strengths and weaknesses. Chapter 2 concludes with a practical view of storyboarding, pre-visualisation and the filmmaking process within the context of a case study (explored through an interview) in order to establish pre-visualisation requirements.

The literature study continues to Chapter 3 by reviewing approaches from computer vision for extracting information from images in order to automatically interpret user sketches. Methods

for detecting, describing, comparing and matching features are investigated. The methods are critically reviewed to establish how feasible each would be for processing images sketched by the user. Methods are also investigated for posing rigid bodies and articulated bodies in 3D space from user sketches. The review of computer vision literature is used to answer the second research question and provide groundwork for Chapter 4.

The main contributions of this research are presented in Chapter 4. It begins with a brief overview of pre-visualisation authoring frameworks and proposes the SISPA framework for the user interface layer. The chapter discusses the various components of the SISPA framework, including the data model, the GUI and the Computer Vision Module. The GUI is designed in order to address the shortcomings of existing drag-and-drop, mouse-based authoring environments. The chapter includes a detailed discussion on various approaches for describing user sketches so that the descriptors can be used for image recognition. The strengths and weaknesses of each method are investigated in order to identify an approach which is best suited for describing user sketches. The methods used for estimating the pose of rigid bodies and articulated figures are discussed and an algorithm for estimating the camera from user sketches is provided. The chapter concludes by discussing how the proof of concept prototype tool was implemented so that it could be used for usability testing.

The evaluation of the prototype tool is discussed in Chapter 5. The chapter begins by investigating the methods available for evaluating the usability of a user interface design or prototype. It discusses the design of the evaluation by outlining its outcomes, selecting evaluation methods and describing the details of the evaluation. This includes the sampling of participants, the evaluation metrics used and the evaluation procedure (instruments, tasks, and setup). The analytical and empirical results of the usability evaluation are reported as well as observations made during the usability study. The lessons learned from the positive and negative findings of the evaluation are compiled into a set of guidelines that can be used for creating future sketch-based pre-visualisation authoring tools that use a storyboarding approach.

The dissertation is concluded in Chapter 6, which shows how each research objective was achieved and how the dissertation answered the corresponding research questions. It discusses the theoretical and practical contributions that were made during this research as well as the limitations and problems encountered. The chapter concludes by recommending future research opportunities.

Chapter 2:

Pre-production

2.1 Introduction

The pre-production phase involves planning every aspect of the film and preparing for production. This chapter provides background on the research problem by discussing the filmmaking process and focusing on the activities performed during the pre-production phase (see Figure 2.1). The pre-production phase involves analysing the script, staging, storyboarding and pre-visualisation. Insight into traditional storyboarding is provided by discussing how storyboards are used and created. The details, graphical devices and annotations found in storyboards are discussed.

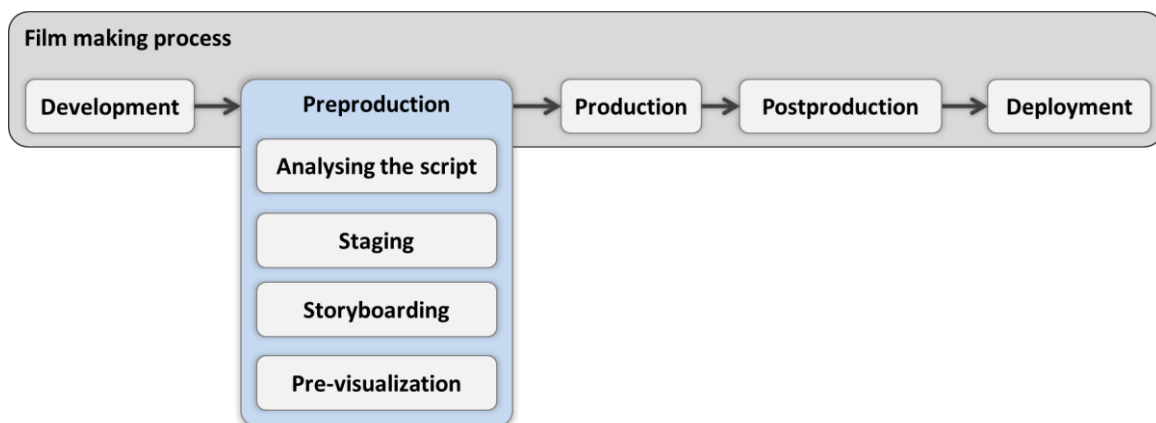


Figure 2.1: An overview of Chapter 2.

Pre-visualisation is a valuable communication tool that is used during the pre-production phase for planning and collaboration before production begins (Nitsche 2008; Mamer 2008). It is useful for exploring various options for scene settings, actor behaviour and camera placement without incurring the cost of production (Jung *et al.* 2010). This chapter discusses the various textual and graphical approaches to authoring pre-visualisations. It concludes with

a case study that reviews the topics discussed from a practical point of view. The case study is based on an interview conducted with a producer from an animation studio. The findings of the interview are used to determine the requirements for sketch-based pre-visualisation authoring (see Appendix A).

The chapter answers the first research question identified in Chapter 1, namely Q_1 : *what methods and theories have been developed for authoring pre-visualisations?* This is achieved by answered the following sub-questions:

- Q_{1.1}: What phases and activities are involved in the film making process?
- Q_{1.2}: How are the activities of the pre-production phase performed?
- Q_{1.3}: What approaches exist for authoring pre-visualisations?
- Q_{1.4}: What are the requirements of a sketch-based storyboarding tool for authoring pre-visualisations?

2.2 Filmmaking Process

The filmmaking process has five major phases: development, pre-production, production, postproduction and distribution. Figure 2.2 provides an outline of these phases.

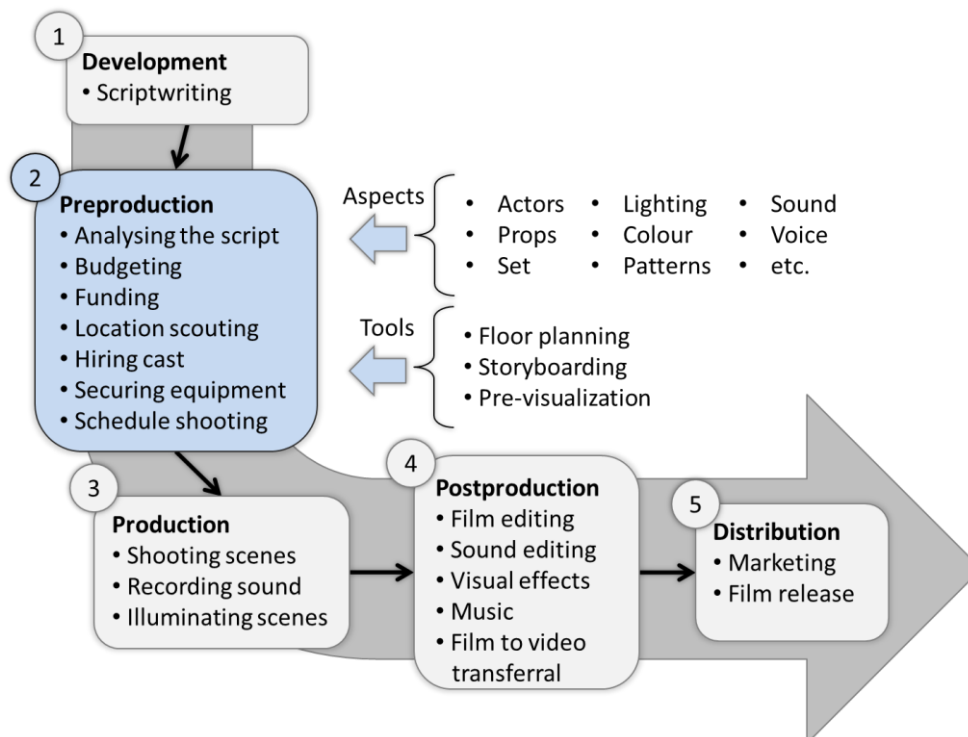


Figure 2.2: The filmmaking process, tools and aspects that need planning.

The filmmaking process:

1. Development involves the creation of a well-written script. The script is used to plan the dialogue, character actions and scene descriptions. Script writing is the most important aspect of filmmaking and it is also the least costly (Tomaric 2010).
2. Pre-production involves thinking through every aspect of the film and doing all the necessary preparations before the production phase begins. It includes analysing the script, floor planning, storyboarding, pre-visualisation, budgeting, obtaining funding, scouting locations, finding and hiring the cast, securing equipment and working out a shooting schedule.
3. Production involves shooting all the scenes. The director manages producers, cinematographers, sound and lighting technicians and other staff in order to bring their vision to life. This process involves shooting, recording sound, illuminating the scene and many other tasks. The production phase ends when the last scene has been shot (Mamer 2008).
4. Postproduction involves editing and finishing the final film image; editing the soundtrack; adding visual special effects; adding sound effects and music; and transferring the film to video.
5. Distribution involves advertising the film on posters and other media. The film is then duplicated for distribution and released to selected cinemas or other media outlets.

Filmmaking is a complicated process that requires the director to consider many aspects of the film, including where props should be placed, how shots should be taken and how actors should perform. This research focuses on some of the tools used during the pre-production phase including floor plans, storyboards and pre-visualisations.

Floor planning is the process of creating top-down drawings of the set where the artist indicates character behaviour and camera behaviour. Floor plans are used as a planning tool for the storyboarding process. Floor plans are discussed in more detail in Section 2.4.

Storyboarding is the process of creating comic book like representations of a film. Storyboards are used to convey information that may be difficult to describe in a textual script. Shots, camera motion, character behaviours and emotions are examples of details that are difficult or tedious to narrate. Section 2.5 provides a detailed discussion on storyboarding.

Pre-visualisation is the process of creating the initial shots for a scene using virtual environments (The Previsualisation Society 2012). Pre-visualisation authoring will be discussed in Section 2.6.

The storyboarding team and the pre-visualisation team are both responsible for presenting the film's story visually. The script tells the story of the film and it is therefore their primary information source. It narrates the film in terms of actions and dialogue that cannot be visualised immediately. The script has to be restructured in terms of smaller units. This process involves analysing the script.

2.3 Analysing the Script

The development phase begins with an original or adapted script. A *script* is a document containing the actions, expressions and dialogue of the characters within the film. Figure 2.3 shows an example scene from a script in a typical format. The format distinguishes dialogue from action by having the dialogue indented with respect to the rest of the text.

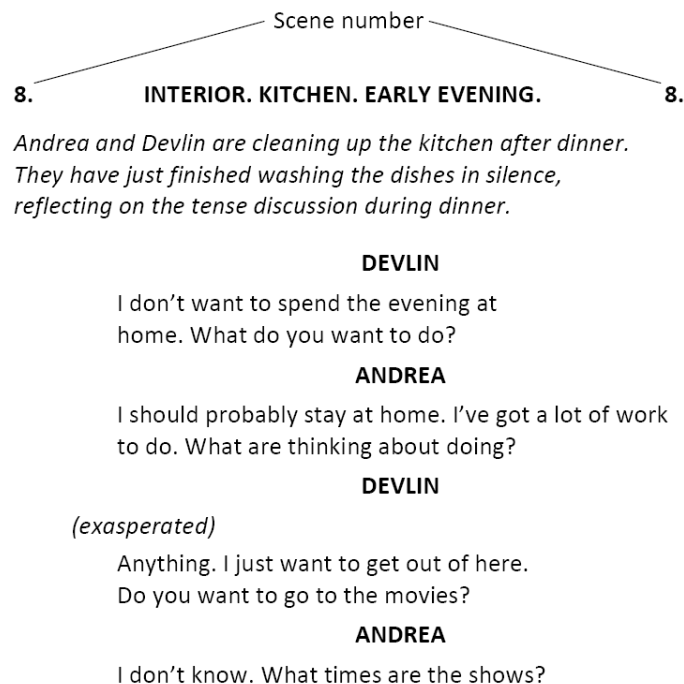


Figure 2.3: The script specifying character action and dialogue (Mamer 2008).

During the pre-production phase, the director begins with the original script and analyses it. This involves annotating the script with information in order to produce what is called a *shooting script*. The shooting script is used as a guide during production and is constantly refined by the director as well as other team members (Pramaggiore and Wallis 2008).

The process of analysing the script begins by identifying scenes, dramatic blocks, acting beats and narrative beats (Proferes 2005). *Scenes* are the smallest units of conflict in the story (Glebas 2008). They contain conflict between characters who are trying to achieve their own goals and overcome obstacles. Each scene in the shooting script is numbered. Scenes are divided into smaller sections called *dramatic blocks*. A dramatic block contains one dramatic idea. New dramatic ideas are often introduced to the audience by beginning each event in a different location on the set. Dramatic blocks are indicated in the shooting script using dramatic block headings. Each dramatic block contains several acting beats and narrative beats. An *acting beat* is a unit of action carried out by a character. It occurs every time the action of the character changes. Acting beats are used to differentiate the dialogue and action elements of a scene. The director also identifies *narrative beats* which are acting beats that progress the plot of the story. An additional column is added to indicate beats in the shooting script. Narrative beats are indicated using capitalization or formatting.

The next step in analysing the script involves developing a shooting strategy. Rabiger (2003) outlines two important aspects that have to be decided upon when creating such a strategy. The first aspect is character placement and movement. The director is responsible for placing the characters on the set and planning their movement and orientation. This task is called *character blocking*. The second aspect of creating a shooting strategy is camera placement and movement. It involves planning the placement and movement of the camera in order to display the environment sufficiently and obtain the desired shot type. The process of placing a character in a shot is called *character framing*. The director has to be careful when taking single shots of characters partaking in a conversation because there are lines connecting the eyes of the characters, called *eyelines*. The director has to place the camera on the correct side of the eyeline so that it appears to the audience that the correct characters are talking to each other. The director must also consider the movement speed of the camera as well as the type of lenses being used in order to achieve the desired dramatic effect. The shooting script provides descriptive detail about character blocking and character framing but the exact location and movement of each character and camera is specified during the staging process (Proferes 2005).

2.4 Staging

Staging refers to the process of designing the performance space of the film. It involves blocking characters and planning performances, as well as positioning and moving the

camera (Glebas 2008). Proferes (2005) and Rizzo (2005) identify several of the functions staging provides in terms of character blocking and camera behaviour.

The director performs character blocking by rendering the action of each character. This is achieved by indicating how characters move and turn, what gesture's they make, what poses they assume and any other physical action they perform. Character blocking is useful for indicating the relationships between characters. For example, depending on where two characters are sitting around an office table, the audience can assume which character is the subordinate. Character blocking is also useful for associating characters with each other. This can be achieved by blocking the characters closer together or by moving the camera.

The camera's position and movement are indicated during the staging process. Staging helps the director to create a frame for the camera. The director uses the camera to familiarise the audience with the location of the scene and emphasise important props in order to orientate the viewer. The viewer's attention can be guided by redirecting and moving the camera. For example, the camera can move along a kitchen table while pausing at particular dishes while a chef discusses them. This forces the audience to pay attention to the dishes and listen to what the chef is saying instead of focusing on the chef and missing the message.

The director visualises the staging of a scene of each dramatic block (see Section 2.3) by drawing an overhead view called a *floor plan*. Floor plans are used to illustrate the contents of the set and they illustrate the placement and movement of characters and cameras. The director begins by drawing a blank floor plan as shown in Figure 2.4. The blank floor plan only shows the walls, doors, windows and props of the set.

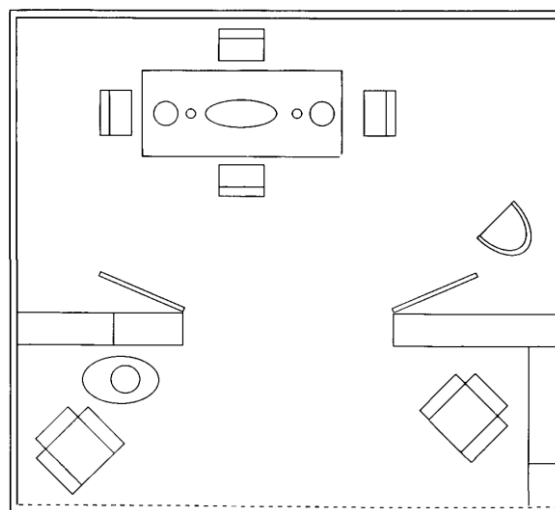


Figure 2.4: A floor plan of the set (Proferes 2005).

Character blocking for a particular dramatic block is indicated by drawing symbols that represent the characters. For example, in Figure 2.5, Andrea and Devlin are characters performing in the scene and they are denoted by an *A* and *D* respectively. The director draws a symbol for each character on the set. Character movement is illustrated by drawing solid paths with arrows. In-between character locations are also used to indicated pauses and reorientation.

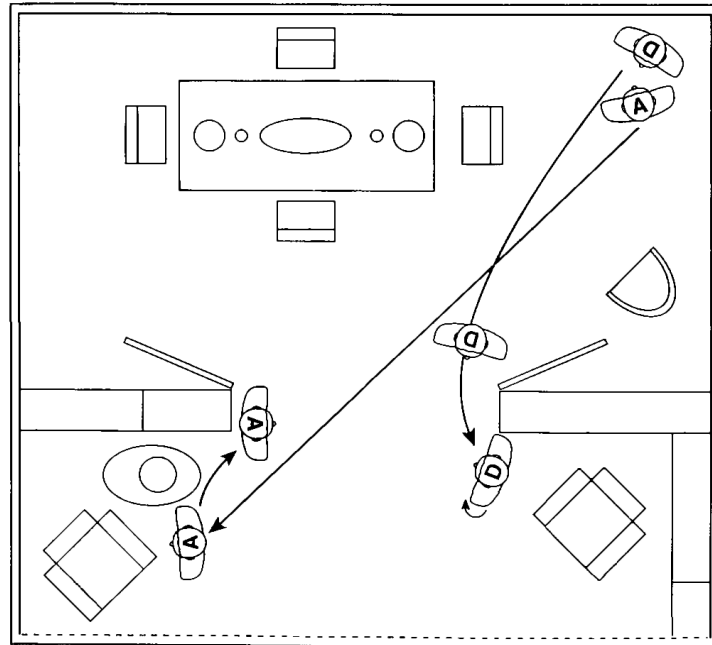


Figure 2.5: A floor plan for a dramatic block showing character blocking (Proferes 2005).

The director also uses the floor plan to document the position and movement of the camera (Mamer 2008; Proferes 2005). Floor plans are therefore used to plan and document the scene's shooting strategy. A shooting strategy consists of several shots. A shot is a series of frames that starts when the camera begins recording and stops when recording ends. There may be several cameras on the set, but when the camera is referred to as being singular, it is assumed to be the camera the audience is viewing from.

Figure 2.6 shows how the floor plan can be annotated to illustrate the shooting strategy. In this example, symbols that look like eyes are used to indicate the position and orientation of the camera for each shot. Camera movement is illustrated by drawing dotted paths with arrows so that it can be differentiated from character movement. The orientation of a moving camera can also be illustrated by drawing orientation symbols such as the darkened cameras shown in this example.

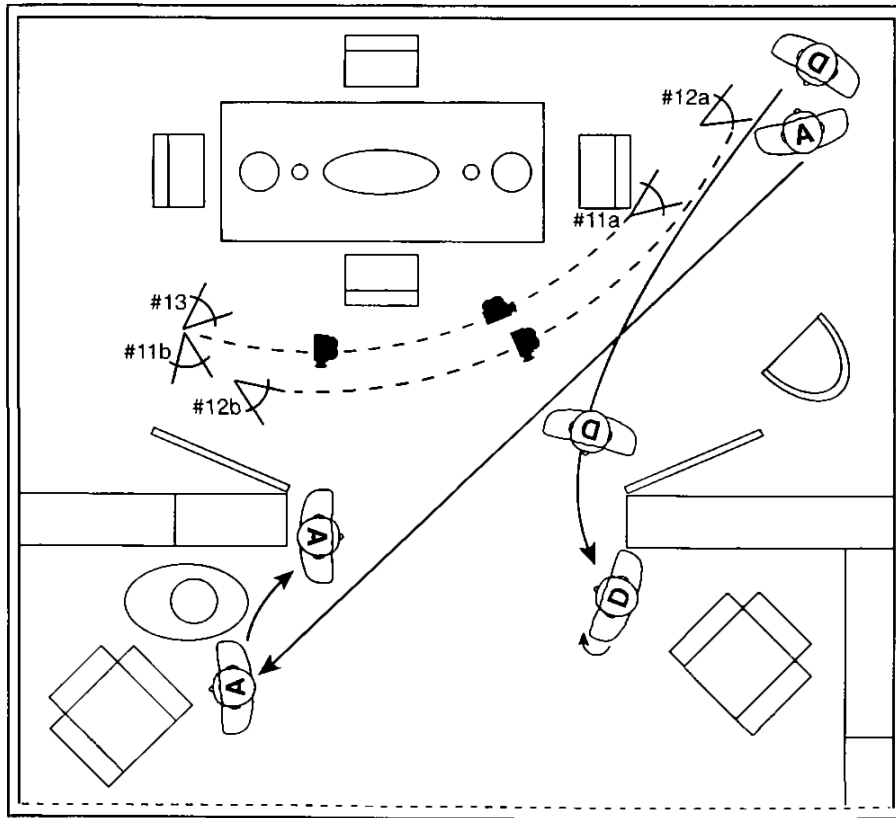


Figure 2.6: A floor plan for a dramatic block showing character blocking and shots (Proferes 2005).

Numbering shots is important when creating a shooting strategy. If the shots are numbered then the director can associate them with sections in the shooting script. The floor plan shown in Figure 2.6 indicates shot numbers using a hash followed by the shot number and a letter indicating whether the shot is beginning or ending (an “a” or “b” respectively). For example, shot twelve follows Devlin as he moves into the next room. The shot begins at #12a and ends at #12b while remaining focused on Devlin as he and the camera move.

The staging process produces information that is necessary for the next task performed during pre-production, which is storyboarding. Storyboarding begins once the shooting strategy has been documented on the shooting script and visualised on floor plans (Glebas 2008). Sometimes the storyboard artist is only provided with an annotated script. In this case, the storyboard artist creates rough floor plans in order to aid in the storyboarding process. The next section provides a detailed discussion on storyboarding.

2.5 Storyboarding

A *storyboard* is a comic book like representation of a film. Storyboards should convey the shooting strategy and approximate cast individuals (Mamer 2008). They are used in the early

stages of production (or during the transition from pre-production to production) to present the contents of the script visually. The type of storyboard and the amount of detail it contains depends on the scene that is being created, the requirements of the director and the financial limitations of the project (Glebas 2008; Long and Schenk 2002). This section discusses how storyboards are used and what the storyboarding process entails. It provides an overview of the details included in storyboard sketches, and it discusses the annotations and graphical devices storyboard artists use. The section is then concluded with a comparison of two popular software solutions that facilitate storyboarding.

2.5.1 How Storyboards Are Used

Storyboards are used differently depending on the needs of the director and the project. The main function of storyboarding is to help the director think about aspects that might have been missed during script development. Storyboarding requires the director and the storyboard artist to plan the visual details required before production begins. These details may include how to frame and shoot a scene, how the set appears, where props are and how characters perform (Long and Schenk 2002). Storyboards need to clearly depict every important detail of each character, including character action, emotion and appearance. The storyboard artist has to follow a process in order to create the required storyboards. The next section will provide a brief overview of the storyboarding process and the activities it involves.

2.5.2 The Storyboarding Process

Storyboarding is a process of preparation and iterative refinement (Glebas 2008). It consists of at least six steps as shown in Figure 2.7. This subsection briefly discusses the activities involved when preparing for storyboarding. It also discusses the activities involved when creating storyboards.

Preparation for storyboarding requires the script to be formally analysed in order to identify acting beats and narrative beats (see Section 2.3). The storyboard artist needs to know when and where the action starts and ends. Narrative beats indicate to the storyboard artist when action should be emphasised in order to capture the audience's interest. In general, the storyboard artist creates one or more storyboard drawings for each acting beat or narrative beat.

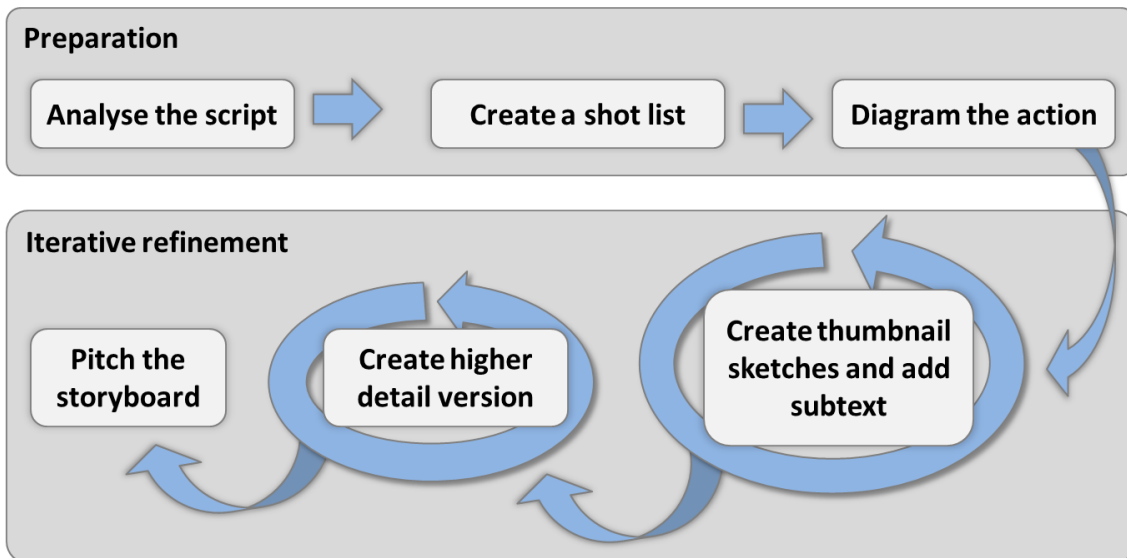


Figure 2.7: A flow diagram illustrating the storyboarding process.

The storyboard artist uses the annotated script to create a *shot list*. The shot list provides a list of camera setups for each scene. Several storyboard drawings may be made for each shot (Proferes 2005). The final preparation activity involves diagramming the action of each scene. The storyboard artist uses existing floor plans if they are available (see Section 2.4). If floor plans are not available then the artist roughly sketches out floor plans in order to establish the scene’s environment and plan the positioning and movement of the camera and characters. The storyboard artist begins sketching once the preparation for the storyboard is complete.

The first sketches the storyboard artist makes are quick, low quality sketches called *thumbnail sketches*. Thumbnail sketches are generally stamp-sized, or postcard-sized, and they are made in the margin of the script. They allow the artist to quickly sketch variations of the scene in order to iteratively explore and refine alternative ways of expressing the story visually. The storyboard artist also makes use of *subtext* on the script while thumbnail sketching. Subtext is additional text that is added to the script in order to add subtle ideas. The goal of subtexting is to enhance the story beyond what was originally contained in the script. It is used to annotate the script in order to answer questions related to the theme and story. The storyboard artist creates a higher detail version of the storyboard once thumbnail sketching is completed. The final sketching process is also iterative. The sketches are then placed in panels and laid out so that the final storyboard can be presented.

The storyboard artist *itches* the storyboard to the director and the team. Pitching a storyboard involves showing the team the storyboard while performing dialogue, discussing action and

showing how the story progresses by pointing at the storyboard panels. Storyboarding software can also be used to automate the pitching activity (see Section 2.5.5). Dialogue performances are recorded and combined with animation to create animated storyboards (Rizzo 2005).

2.5.3 Storyboard Detail

A traditional storyboard consists of a rectangular arrangement of panels that are laid out on each page. Each panel contains a single storyboard sketch and below it is a combination of dialogue, action text and camera instructions. Above each panel is the scene number as well as the panel number for that scene (Glebas 2008). This subsection discusses how the characters, environment and special effects are illustrated in each storyboard panel.

Characters are the most important elements in a storyboard, irrespective of the level of detail used. Sketches are used to convey character poses and body language. Figure 2.8 (a) shows an example storyboard sketch that illustrates the pose of two characters by only using stick figures. Artists use stick figures as a planning mechanism before adding body to their characters. Figure 2.8 (b) shows how an artist can expand a stick figure so that it has shape and body. It is impossible to identify each character in a sketch unless there is a way of differentiating characters from one another.

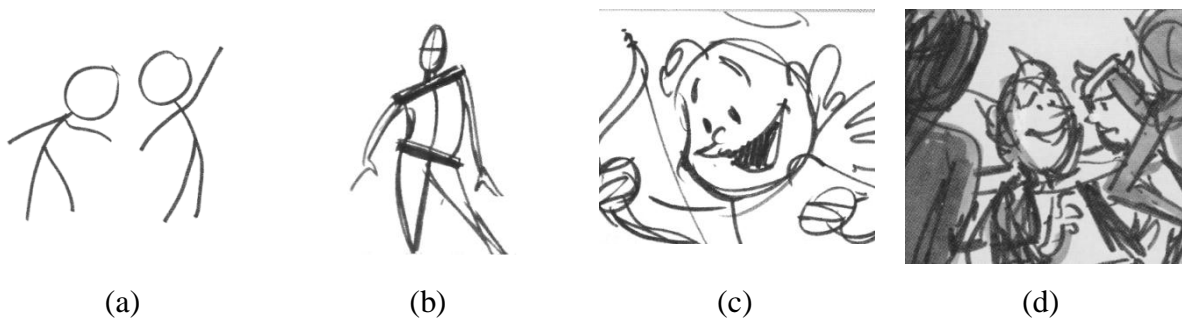


Figure 2.8: Examples of increasingly detailed character sketches (Glebas 2008).

Characters can be differentiated by designing and sketching their appearance. A character's external features, such as its proportions, face, hair and costume, are used to differentiate it from other characters. Figure 2.8 (c) and Figure 2.8 (d) show examples of how the appearance of characters can be sketched. Storyboard artists typically sketch the characters' faces and facial expressions as well. This allows them to express character emotion visually.

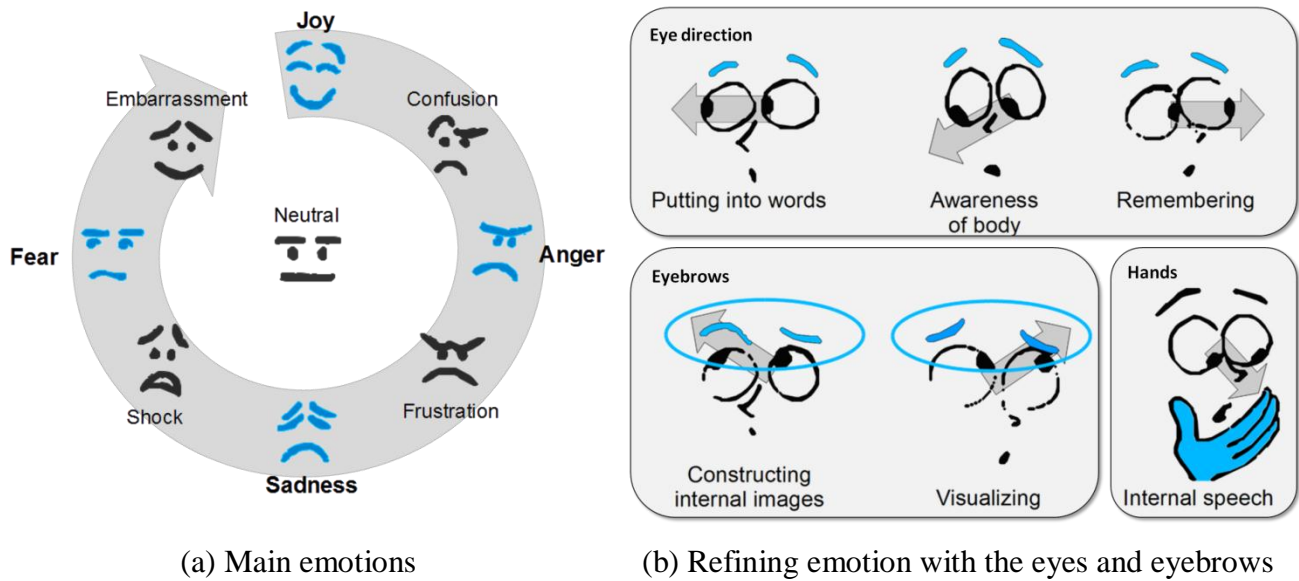


Figure 2.9: Expressing character emotion by sketching faces (Glebas 2008).

There are four main emotions as shown in Figure 2.9 (a): joy, anger, sadness and fear. Intermediate emotions include confusion, frustration, shock and embarrassment. The key to expressing emotion lies in the mouth and eyebrows of the character's face (Glebas 2008). The eyes themselves are used to refine the emotional state of the character. Figure 2.9 (b) shows an example of how the direction of the eyes, the shape of the eyebrows and the character's hands can be used to express various emotions. Storyboard artists can express characters sufficiently by combining emotion with character pose and appearance.

Shading is used to illustrate scene lighting and shadows. The three-point lighting approach is standard in filmmaking. It involves using three light sources. Consider Figure 2.10 (a) – (c). The *key light* is the primary light source. The *fill light* is used to illuminate the shadows being cast by the key light. The *back light* is used to visually separate the character from the background (Pramaggiore and Wallis 2008). Changing lighting attributes such as intensity, placement and contrast allows the lighting team to change the mood of the scene. A skilled storyboard artist can illustrate lighting setups if required.

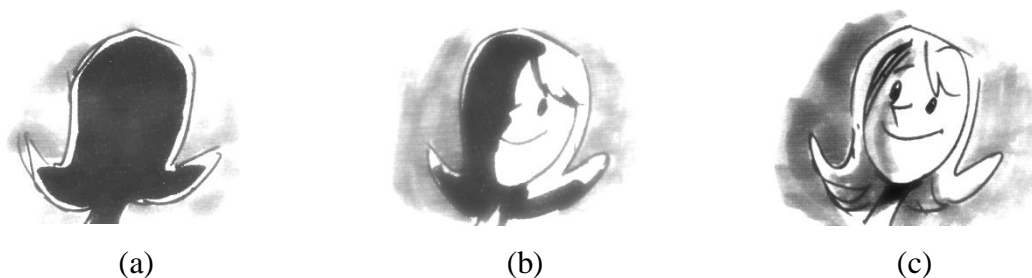


Figure 2.10: The three-point lighting approach (Glebas 2008).

Live-action films and animated films may require the storyboard artist to sketch the environment of the scene in order to illustrate the camera setup and the set. The outdoor scene shown in Figure 2.11 (a) illustrates the location and appearance of the building and characters. The storyboard artist has to consider the three-dimensional accuracy of the sketch when sketching the environment. This requires the artist to consider depth and perspective in order to accurately illustrate a scene in three-dimensional space. Figure (b) shows another example where depth, perspective and scale play an important role in the quality of the storyboard sketch. Environmental effects are also illustrated when creating high quality storyboard sketches such as the one shown in Figure 2.11 (c). The sketch shows the pose, emotion and appearance of the involved characters. It also illustrates the environment, the lighting setup and special effects, such as the slime and foam.

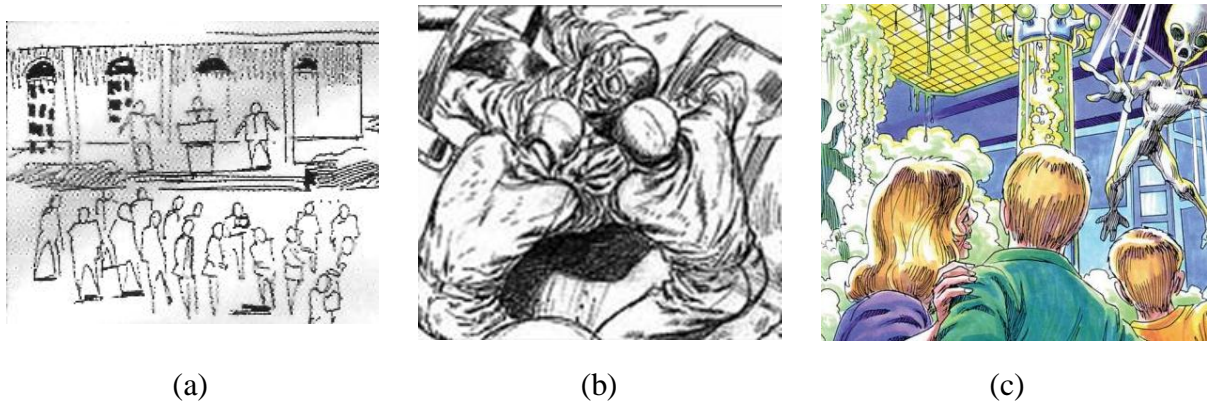


Figure 2.11: Examples of increasing detailed storyboard sketches (Glebas 2008; Forton 2011; Simon 2006).

Sketches can be used to effectively illustrate characters, emotions, lighting, environments and effects. However, artists often augment storyboards with annotations and graphical devices in order to illustrate character actions, shooting directions, emotions and effects that are difficult to illustrate with a plain sketch. The following section provides a brief overview of the annotations and graphical devices used in storyboarding.

2.5.4 Annotations and Graphical Devices Used in Storyboarding

Graphical devices are used to indicate action, emotion, feeling, and effects in storyboards (Simon 2006). The use of graphical devices depends heavily on the style of the storyboard artist. Attempts at recording and classifying graphical devices have been made, such as the *The Lexicon of Comicana* (Walker 2000). This is an informal encyclopaedia that attempts to describe, name and categorise most of the graphical devices found in cartoons. There is, however, no formalised standard taxonomy or vocabulary that comprehensively

covers the description of moving content in images (Luckow 2010). This subsection discusses and categorises the annotations and graphical devices that are found in storyboards. It includes several examples that highlight the annotations and graphical devices in colour. These examples include illustrations for indicating the state of objects, the movement of characters and objects, the feelings and emotions of a character, the behaviour of the camera and the type of transitions between shots.

The state of an object can be indicated by emphasising an effect which is unique to that state. For example, reflective objects like the glass ball shown in Figure 2.12 (a) can be indicated by drawing a shiny spot on the object. Objects that produce light can be indicated by drawing rays of light, as shown in Figure 2.12 (b). Hot objects can be sketched with steam like the cup of hot tea in Figure 2.12 (c).

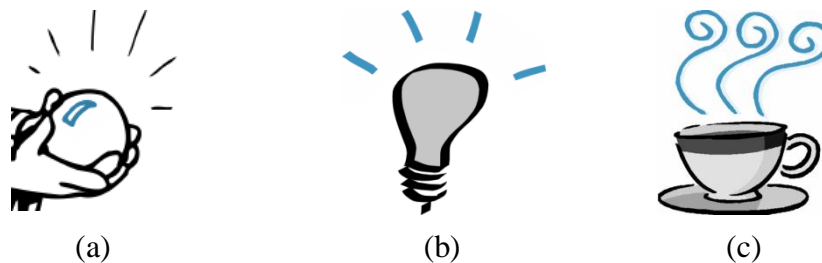


Figure 2.12: Graphical devices that illustrate object state (Clipartoday 2011; Clker 2011; SeagullsCalling 2011).

The movement of characters can be illustrated using graphical devices and annotations. Figure 2.13 (a) and Figure 2.13 (b) show how curved lines and straight lines can be used to indicate moving limbs and characters. Clouds that remain at a location can be used to indicate that an object or character moves quickly, as shown with the cow in Figure 2.13 (c). The movement of an object or character can also be illustrated using arrows and paths, as shown in Figure 2.13 (d).

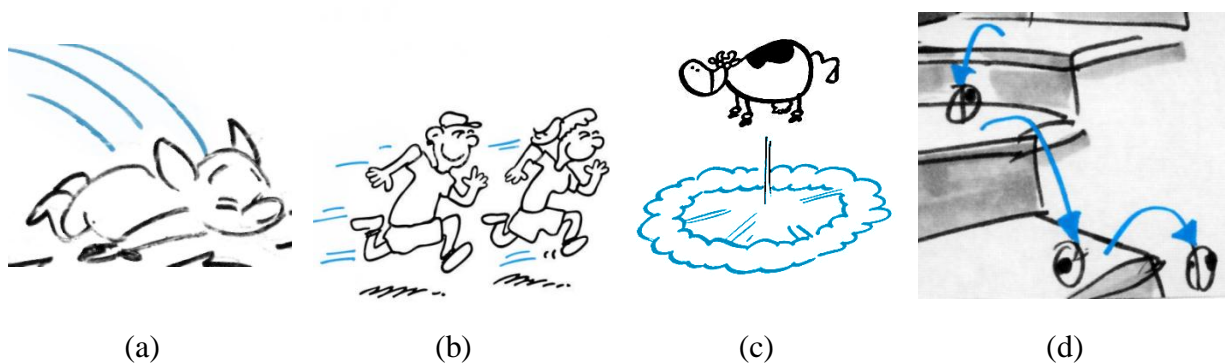


Figure 2.13 Graphical devices and annotations for (a-c) movement and (d) motion paths (a,d: Glebas 2008; b: StockphotoPro 2011; c: Greenhead 2011).

Walker's (2000) work documents many graphical devices that can be used for illustrating the emotions and feelings of characters. Figure 2.14 provides examples showing how graphical devices can be used for expressing stress, surprise and dizziness.



Figure 2.14: Graphical devices for illustrating character feelings and emotions (Shelly 2011; Glebas 2008; PicturesOf 2011).

The behaviour of the camera can be illustrated by annotating storyboards. The storyboard artist can indicate how the focal length (zoom) of the camera changes, how the camera moves on the set (dolly) and how the camera can be panned (Glebas 2008).

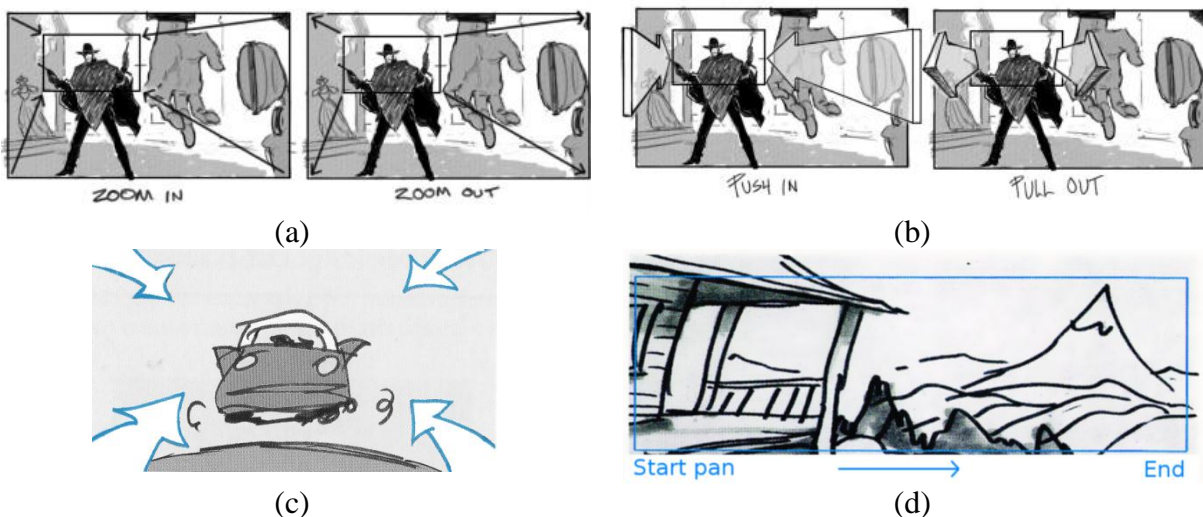


Figure 2.15 Annotations for illustrating camera behaviour (a,b: Goldman et al. 2006; c, d: Glebas 2008).

Changes in the camera's focal length are indicated using two rectangles of the same aspect ratio, where one rectangle is inside the other. The smaller rectangle contains the visible area of the shot at a closer zoom. The larger rectangle contains the visible area of the shot at zoom that is further away. The corners of both rectangles are also connected by arrows to indicate whether the camera is zooming inwards or outwards. Figure 2.15 (a) shows an example of the camera zooming towards and away from a character posing in a western scene. The zoomed in view focuses on the subject and the zoomed out view shows the context of the scene.

Camera movement is indicated using rectangles and arrows similarly to when the camera's focal length is changed. The internal rectangle, however, is not always shown and the arrows are normally thicker. The arrows are also drawn in perspective so that they appear in context with the scene (Goldman *et al.* 2006). Figure 2.15 (b) shows an example of the western scene but with a moving camera. Figure 2.15 (c) shows a variation of the annotation that does not show the interior rectangle.

Camera panning is indicated by drawing a rectangle that covers the area being panned. The storyboard artist indicates the starting and ending positions of the pan and considers the production's aspect ratio. Figure 2.15 (d) shows an example pan that starts with a view of a porch and ends with a view of a mountain.

Transitions between shots are illustrated on storyboards by making annotations between the storyboard panels (Glebas 2008). A transition that involves dissolving from one shot into another shot is indicated by drawing a cross between the involved storyboard panels. Figure 2.16 (a) shows an example of how a storyboard artist indicates dissolving shots. Shot two becomes more visible as shot one becomes less visible. A transition that involves fading from one shot to another is indicated using a triangular left or right symbol that directs away from the shot that is fading out. Figure 2.16 (b) shows an example of shot one fading out until it is black, or some other colour, and then shot two appears instantly.

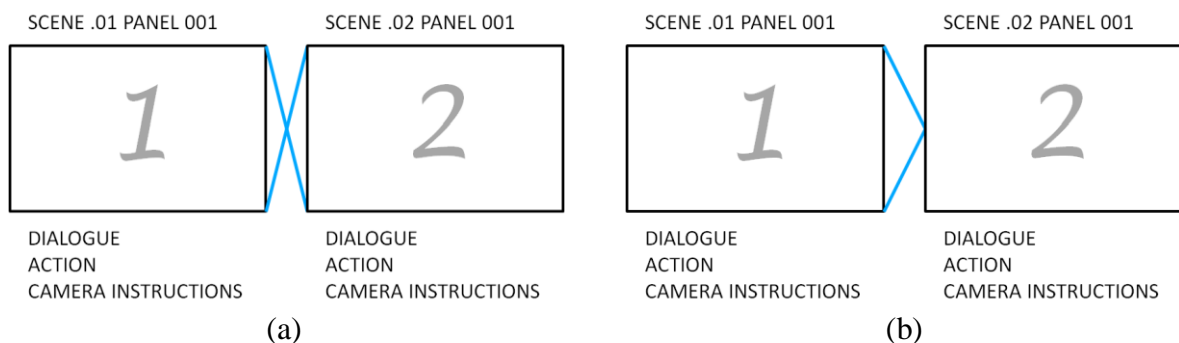


Figure 2.16 Annotations for illustrating transitions between shots.

2.5.5 Storyboarding Software

In the past, the final storyboard was sketched by hand on paper. Today, storyboarding software can be used to create storyboards quickly and easily. Computer generated storyboards have several advantages over paper-based storyboards. Higher quality storyboards can be made using tools that provide graphical user interfaces for digital sketching or content placement. The resulting storyboard sketches can be manipulated after

they have been created. This allows the storyboard artist to make changes to the storyboard quickly and easily, unlike paper-based storyboards that require the panels to be re-sketched by hand. This subsection provides a comparative discussion of the features provided by two popular storyboarding software packages, namely Storyboard Quick and Storyboard Pro. Table 2.1 provides a summary of the features compared in this discussion.

Table 2.1: A comparative summary of Storyboard Quick and Storyboard Pro.

	Storyboard Quick	Storyboard Pro
Developer	PowerProduction Software	Toon Boom Animation
3D virtual environment	✗	✓
Actor posing	✓	✗
Animation	✓	✓
Content library	✓	✓
Artistic skill required	✗	✓
Audio support	✗	✓
Tool output	Storyboards: Printed, HTML and Flash format	Storyboards: PDF; Video format
Support for storyboarding conventions and annotations	✗	✓

Storyboard Quick is a software package that storyboard artists can use for creating and editing storyboards using a drag-and-drop interface (Power Production Software 2011). It allows the user to create storyboards using 3D models from an object library. Figure 2.17 provides a screenshot of Storyboard Quick showing how the storyboarding tool can be used to place 3D props and characters on a 2D background image. The user has limited control over the interaction of virtual actors and the environment, for example a virtual actor cannot sit on a chair that's in the scene background. There is also limited support for animation, and the storyboarding tool does not support audio recording or replay for narrative voice-overs. Storyboard Quick has several features that typical storyboarding tools provide. It can import scripts from professional screenwriting applications or text documents. It allows the director to reorder, edit and add storyboard frames and shots. It also produces storyboards with professional layouts that can be printed out.

Storyboard Quick allows for rapid storyboarding without the need for sketching, but it compromises on the functionality that sketching interfaces provide. This includes sketching on multiple overlapping layers and using various brushes and manipulation tools. Sketching

interfaces provide the storyboard artist with more freedom and better tools for sketching storyboards, but they require sketching skills to use them.



Figure 2.17: A screenshot of Storyboard Quick (Power Production Software 2011).

Storyboard Pro is a storyboarding tool that provides a sketching interface for creating 2D storyboards (Toon Boom Animation Inc. 2011). The tool does not provide 3D graphics like Storyboard Quick, but it provides a library of images for scene locations, props and actors. Directors can use the drag-and-drop interface to create storyboards quickly and easily.

Figure 2.18 shows a screenshot of Storyboard Pro's sketching interface. A sketching device, such as a graphics tablet or a tablet PC, can be used to make sketching on a computer easier. The sketching interface allows the artist to sketch each storyboard panel using multiple layers. This is similar to photo editing software such as Photoshop (Adobe 2012). Layers allow for the separation of different parts of an image. The content of each layer is represented using vector graphics. Layers can be resized, rotated and transformed without loss of quality. The sketching interface allows the user to define and reuse different stroke types called *brushes*, for example pencils, pens, paint brushes, etc. Captions can also be added to storyboard panels for dialogue and action. Storyboard Pro does not interpret the contents of the storyboard, and each sketch remains a collection of 2D strokes. Storyboard Quick represents the contents of each panel with 3D models, but the user is required to place them manually and no sketching

is involved. It is important to note that neither of the interfaces can interpret the storyboard sketches in order to extract information about the scene.

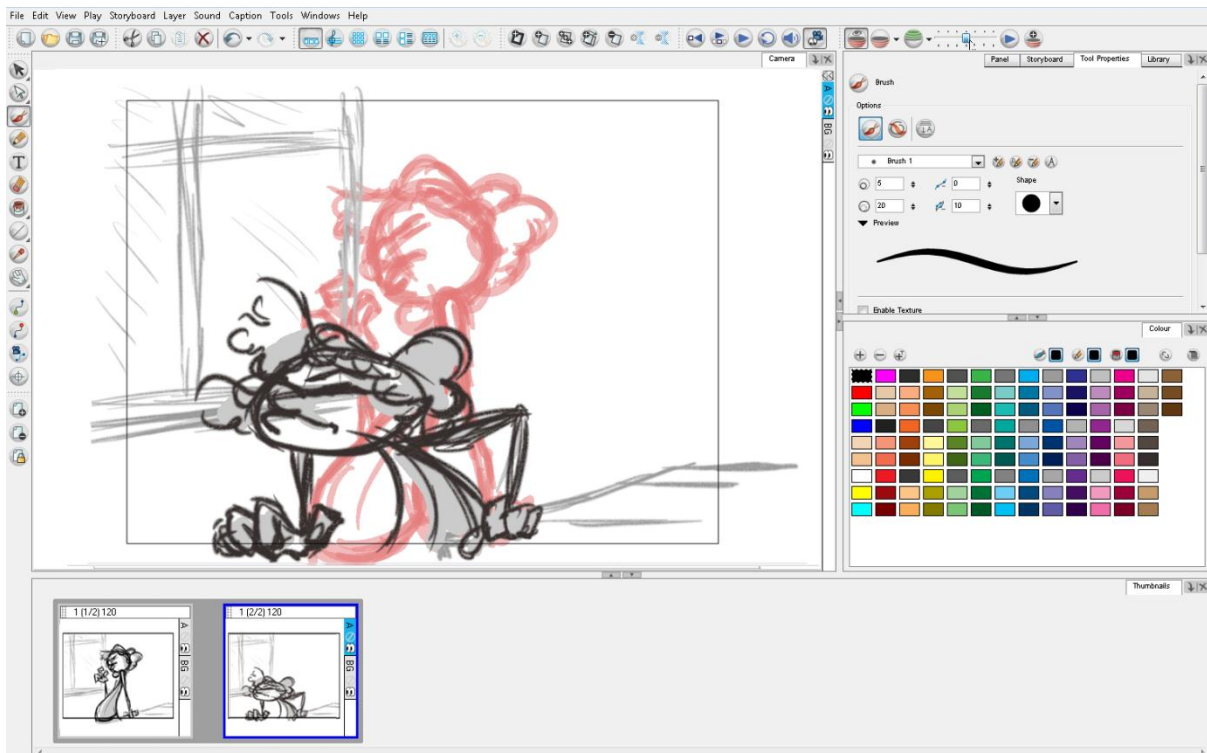


Figure 2.18: A screenshot of Storyboard Pro (Toon Boom Animation Inc. 2011).

Storyboard Pro can also be used to create animated storyboards with audio. The tool supports 2D camera motion such as translation, zooming and rolling. Each layer can be animated independently. For example, a young girl can be sketched on one layer, reaching out to something. A bee can be sketched in the distance on another layer within in same storyboard frame. The tool's animation functionality can then be used to translate and rotate the bee so that it appears to land on the girl's hand. The starting configuration shows the bee in the distance and the final configuration shows the bee on the girl's hand. The final storyboard or animation can then be stored as a PDF file or video.

2.5.6 Discussion

Storyboarding provides a way of visualising the contents of the script by showing static snapshots of the envisioned film. Each snapshot provides insight about the characters involved and their behaviour. It also illustrates the environment and how the audience views the scene. The accuracy of the storyboard sketch depends on the storyboard artist's skill and understanding of the scene's performance space. This is a problem when the storyboard artist

sketches a scene in a manner that cannot be recreated on the set. It is one of the reasons why directors require more accurate visualisation tools such as pre-visualisation. Pre-visualisation tools allow the director to create computer generated visualizations of a scene within a 3D virtual environment. The following section will discuss how pre-visualisations can be authored using textual and graphical authoring approaches.

2.6 Pre-visualisation

Pre-visualisation is the process of creating the initial shots for a scene using animation tools and virtual environments (The Previsualisation Society 2012). A pre-visualisation is generally taken to be a rough version of the film that replaces or enhances the traditional storyboard (Moviestorm 2011). Pre-visualisation is used to experiment with different character setups and camera setups. It is also used to explore various ideas for how the camera moves and how the characters perform (Bull and Kajder 2004; Robin 2008; Sadik 2008). A pre-visualisation can be a static collection of images or it can be an animation.

An animated pre-visualisation is often called an *animatic*. Creating a pre-visualisation involves creating an environment that contains the virtual versions of the set, props and characters (Ye and Baldwin 2008). It also involves defining the interactions and behaviour between all the characters, props and the environment according to the action defined in the film's script. There are two approaches used for creating a pre-visualisation, namely the minimalistic approach and the detailed approach (Moviestorm 2011).

The minimalistic approach specifies as little detail as possible. It involves creating simple virtual environments and omitting unnecessary animation. Background images are often used to represent the set. There is no need for detailed lighting, visual effects or detailed set dressings. The goal of minimalistic pre-visualisation is to get an idea of how the film flows and how long each scene is. Directors or artists who are proficient with pre-visualisation authoring tools create minimalistic pre-visualisations before asking a storyboard artist to sketch a storyboard. The pre-visualisation is then used to create a computer generated storyboard.

The detailed approach attempts to visualise each shot as close to the final film as possible. This requires a detailed virtual environment containing all the assets that will be seen on screen. Detailed actor performances are required with sound, music and visual effects. Detailed pre-visualisation is usually utilised in large budget productions. The goal of this

approach is to reduce the risk of expensive mistakes at a later point in the film making process. This goal is achieved by involving every senior member of the production team in creating the animatic. The pre-visualisation process becomes a collaborative effort where each expert performs as if the film was in production, except they are using a practice version of the film.

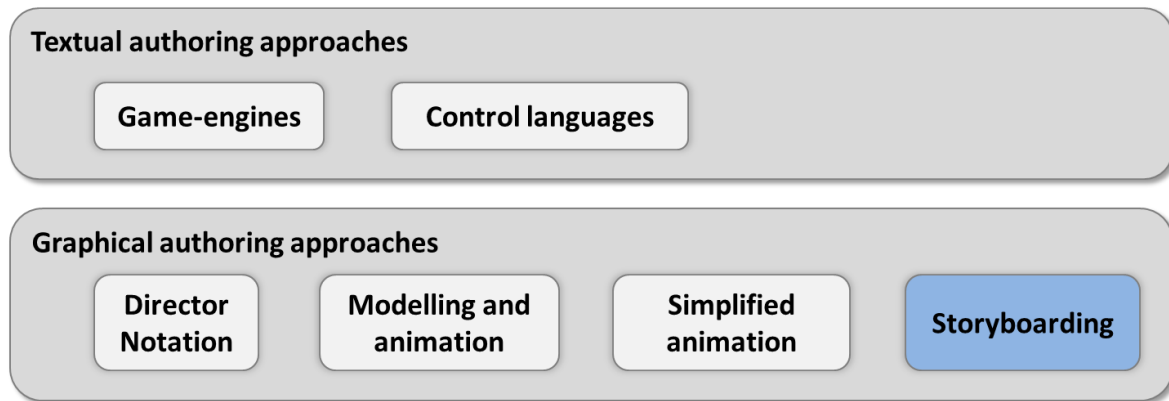


Figure 2.19: Pre-visualisation authoring approaches.

There are several authoring approaches for creating minimalistic pre-visualisations and detailed pre-visualisations. Figure 2.19 provides an overview of the textual and graphical approaches for authoring pre-visualisations. This section begins by briefly discussing how game-engine code and text-based control languages can be used to programmatically author pre-visualisations. It continues by discussing several graphical pre-visualisation authoring approaches. These include the Director Notation, modelling and animation tools, simplified animation tools and storyboard-based authoring tools. In particular, a critical review of two storyboard-based approaches that align closely with this research is provided.

2.6.1 Textual Authoring Approach

Textual authoring approaches can be used to author pre-visualisations programmatically. This can be achieved by writing code in a programming language. The lowest level of pre-visualisation programming is done using a software development language such as C, C++, Java, Visual Basic or C#. Game-engine code bases are often used to aid the development of the pre-visualisation. Higher level pre-visualisation programming can be done using control languages. A behaviour realisation engine parses and compiles the control language code in order to produce the final pre-visualisation. Control languages can also be used to carry information from a graphical authoring tool to a behaviour realisation engine. This subsection reviews methods for authoring pre-visualisations using game-engine code bases and control languages.

2.6.1.1 Game-engine Code

Machinima is a new field for creating animated films using a real-time virtual 3D environment (Marino 2004). Game-engines, such as the Unreal™ engine, are often used for rendering pre-visualisations in real time (Jung *et al.* 2010). These pre-visualisations are typically low fidelity, but they are sufficient for illustrating a film. The main problem with this authoring approach is that it is a complex and difficult task to define the behaviour of each character and camera using game-engine code (Skorupski 2009). Game programmers and modelling artists are required if this authoring method is used. Game-engines generally do not allow the user to directly control cameras, props and characters to the extent which animation packages allow. Another problem with game-engine pre-visualisation is that it is often difficult or impossible to customise character and camera behaviour because of inaccessible underlying game-engine code (Nitsche 2008). It is desirable to be able to author pre-visualisations using a higher level authoring approach. Control languages can be used to provide such an authoring environment.

2.6.1.2 Control Languages

A *control language* is a language used to control the behaviour of entities within a virtual environment (Jung 2009). Control languages are often used as intermediate tools for authoring animated pre-visualisations. A graphical user interface is typically used to create pre-visualisations, and the authoring system generates a digital script in a control language. The script is then sent to a rendering component to be realised as a viewable animation (Vilhjálmsón *et al.* 2007). This process is reflected in the application independent SAIBA framework (Situation, Agent, Intention, Behavior, Animation) (Kopp *et al.* 2006). Figure 2.20 illustrates the framework. It consists of three stages which share data using control languages.

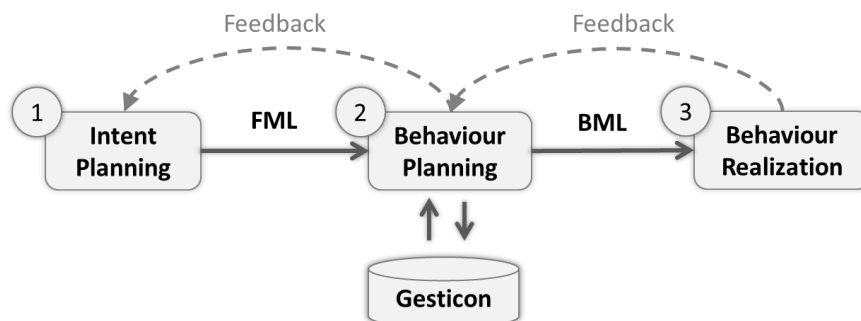


Figure 2.20: A diagram depicting the SAIBA framework (Kopp *et al.* 2006).

The first stage involves planning the communicative intents of the characters in the scene. The second stage involves planning for the behaviour realisation of each character. The final

stage involves parsing and rendering the planned behaviours of each character in order to create the final pre-visualisation. Bi-directional communication between these stages is supported using the Functional Markup Language (FML) and the Behaviour Markup Language (BML).

FML is used to share data between the first two stages by describing character intent without referring to physical behaviour. BML shares data between the second and third stages by describing the desired physical behaviour. This framework offers an advantage over machinima pre-visualisation because control languages are independent of the realisation engine, and the realisation engine may be implemented using any graphics or sound model.

Many other control languages have been proposed by the research community for describing the behaviour of characters within the scene (Vilhjálmsón *et al.* 2007). Table 2.2 provides a list of several control languages and where they have been published. The languages are based on the Extensible Markup Language (XML) which can be read easily using XML parsers. The languages allow the synchronisation and timing of actor performances. Control languages also support the scripting of individual limb movements in order to create gestures. They support facial animation, which is useful for expressing emotion and phonemes for character dialogue.

Table 2.2: A list of control languages.

Abbreviation	Full name	Published work
MURML	Multimodal Utterance Representation Markup Language	(Kopp <i>et al.</i> 2003; Kopp and Wachsmuth 2004)
MSML	Movie Script Mark-up Language	(Rijsselbergen <i>et al.</i> 2009)
PML	Player Markup Language	(Jung 2009)
APML	Affective Presentation Markup Language	(Decarolis <i>et al.</i> 2004)
RRL	Rich Representation Language	(Piwek <i>et al.</i> 2004)
BEAT	Behavior Expression Animation Toolkit	(Cassell, Vilhjálmsón and Bickmore 2001; Vilhjálmsón 2004; Vilhjálmsón 2005)
FML	Functional Markup Language	(Kopp <i>et al.</i> 2006)
BML	Behaviour Markup Language	

The behaviour of the camera is authored by specifying the viewing direction, field of view, target object and shot type. The camera can also be instructed to follow characters or objects in order to keep them in frame.

The main advantage of using a control language is that it is simpler to author behaviour descriptions for characters compared to using a software developing environment with game-engine code bases. Another advantage is that behaviour scripts are independent of the realisation engine and they can be reused for other applications, unlike machinima code, which can only be used for a specific game-engine and application.

The most apparent disadvantage of authoring pre-visualisations using a control language is that it is a textual authoring approach. Directors and storyboard artists desire pre-visualisation tools that provide them with immediate visual feedback instead of a technical translation of the film's script into a control language. If no graphical user interface is used to produce the required control language scripts then it is another form of pre-visualisation programming.

2.6.2 Graphical Authoring Approach

Graphical pre-visualisation authoring approaches allow trained animators, directors and storyboard artists to author pre-visualisations without using programming languages or control languages in text-based authoring environments. This subsection provides an overview of several graphical authoring approaches. It begins by discussing a symbolic approach to authoring pre-visualisations using Director Notation. The subsection continues by discussing modelling and animation packages that trained animators use for creating animated pre-visualisations. Simplified pre-visualisation authoring environments are also discussed that allow directors and storyboard artists to author pre-visualisations without requiring the technical skills and knowledge of trained animators. The subsection concludes by reviewing several storyboard-based pre-visualisation authoring approaches that align closely with this research.

2.6.2.1 Director Notation

A symbolic notation system called Director Notation (DN) has been developed for directing films (Chakravarthy *et al.* 2010). DN is similar to the notation used by music composers in the sense that both describe content formally. DN diagrams are created in order to author pre-visualisations. A DN diagram represents a semantic model where each symbol is mapped to a concept, relation or rule from film ontology (Chakravarthy *et al.* 2009; Chakravarthy *et al.*

2010). A Graphical User Interface (GUI) called the Notation Editor is used to display a preview of pre-visualisations authored using DN.

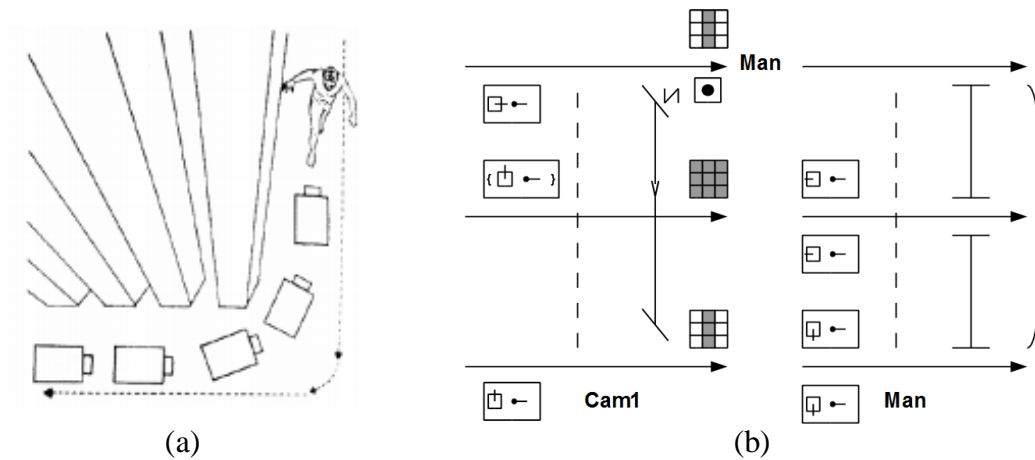


Figure 2.21: Illustrating a scene using (a) a storyboard frame, and (b) Director Notation (Chakravarthy *et al.* 2010).

Figure 2.21 (a) shows an example of a storyboard frame showing camera motion and character action. Figure 2.21 (b) shows how the same scene can be described using DN. The notation uses two columns called *staves*. The first staff is for acting and the second staff is for camerawork. Time is represented vertically and it progresses from the bottom up. Each symbol that is crossed by a horizontal line is in effect at that instant in time. Elements are delimited by relation symbols to define when, and for how long, the elements are in effect. Symbols used in DN, such as symbols for shot type, camera tracking and character movement are related to concepts in film ontology (Chakravarthy *et al.* 2010).

The main advantage of using the Notation Editor for creating DN diagrams is that no low level programming is required. Another attractive aspect of using this authoring approach is that the Notation Editor allows the director to adapt the DN diagram iteratively and preview the results in order to address shot requirements.

The main disadvantage of the approach is that the author is required to possess knowledge about the syntax of DN in order to create DN diagrams for each scene. The approach adds to the author's work load instead of capitalising on existing knowledge and skills. A storyboard such as the one shown in Figure 2.21 (a), can be used to convey what is happening in each scene without requiring the storyboard artist to learn a special authoring syntax. Another disadvantage of the authoring approach is that detailed aspects are often inferred implicitly when using DN because it is a high-level notation. For example, the director can specify that

an actor moves in a series of straight lines and turns, but it's not possible to specify what pose the actor assumes when turning.

Graphical user interfaces with additional functions are required in order to control the elements within each shot accurately. Modelling and animation software provide a means for creating all the required assets of a scene as well as controlling the behaviour of the camera and the actors. The next section discusses how modelling and animation software can be used for the purposes of pre-visualisation authoring.

2.6.2.2 Modelling and Animation Tools

There are several 3D modelling and animation tools available for creating animated films, visual effects and content for interactive 3D applications. The majority of the modelling and animation tools that are available focus on the highly technical process of creating and animating content. Table 2.3 lists and compares several popular high-end modelling and animation tools (Autodesk 2011b; Autodesk 2011a; Autodesk 2011d; Autodesk 2011c; Side Effects Software 2012; LightWave 2012; Blender Foundation 2012).

Table 2.3: A comparison of features provided by 3D modelling and animation software.

	Autodesk Maya	3D Studio Max	Softimage XSI	Motion Builder	Houdini	Light Wave	Blender
Developer	Autodesk	Autodesk	Autodesk	Autodesk	Side Effects Software	NewTek	Blender Foundation
Modelling	✓	✓	✓	✓	✓	✓	✓
Character animation	✓	✓	✓	✓	✓	✓	✓
Real time rendering	✓	✓	✓	✓	✓	✓	✓
Asset library	✗	✓	✓	✓	✗	✗	✗
Motion capture support	✓	✓	✓	✓	✓	✓	✓
Video editing	✓	✓	✗	✓	✗	✓	✓
Programmatic Scripting	✓	✓	✓	✓	✓	✓	✓

3D modellers create content by using modelling techniques to build 3D models. The 3D models represent objects, characters and environments. Most high-end modelling GUIs are complicated to use and require a steep learning curve. The technical process of creating a model involves using mathematical structures such as polygons, bézier surfaces and NURBS using a model editing interface.

Animators are tasked with adding motion to the assets once they have been created. This involves creating an *armature* for each character. An armature is a skeleton-like structure that is attached to the model so that the character can be manipulated. Figure 2.22 (a-c) demonstrates this process. Given a model such as the one in Figure 2.22 (a), the animator attaches an armature like the one highlighted in Figure 2.22 (b), and then poses the armature as shown in Figure 2.22 (c). The character's model is deformed in order to match the pose of the armature. Animation is achieved by key framing these poses in order to create an animated sequence. It is a complicated, tedious and lengthy task to create animations using this approach.

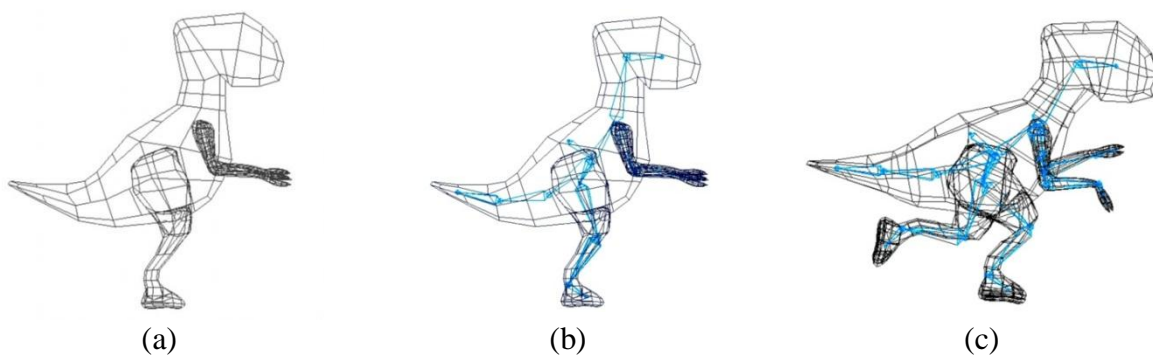


Figure 2.22: Rigging and posing a character (Blender Foundation 2012).

The 3D modelling and animation packages described in this subsection are not designed to allow novice animators to create low-fidelity animated pre-visualisations because the expertise and experience of skilled modellers and animators is required (Labschütz and Krösl 2011; Chakravarthy *et al.* 2010). Modelling and animation packages are designed to be used by modellers and animators that intend to create their own, specialised content.

Animation tools are available that provide a simplified approach to content management and animation. They do not require the director or the storyboard artist to possess animation and modelling expertise. The following subsection discusses the next graphical pre-visualisation authoring approach, which is to utilise simplified pre-visualisation authoring environments.

2.6.2.3 Simplified Animation Tools

Simplified animation tools allow filmmakers of any scale to create pre-visualisations without requiring a team of skilled modellers and animators with high-end tools, like 3D Studio Max or Maya. Simplified animation tools are designed from the director's perspective. The approach focuses on drag-and-drop interaction using terminology with which the director is familiar instead of technical jargon (Innoventive Software 2009).

Table 2.4: A comparison of features provided by simplified animation software.

	FrameForge 3D Studio	Moviestorm	iClone
Developer	Innoventive Software	Moviestorm	Reallusion
Set design	2D topdown view	3D perspective view	3D perspective view
3D objects and actors	✓	✓	✓
Scene construction	✓	✓	✓
Actor posing	✓	✓	✓
Equipment constraints	✓	✗	✗
Artistic skill required	✗	✗	✗
Authoring approach	Shot by shot	Game-style & Timeline	Controller-based & Timeline
Audio support	✓	✓	✓
Output	Annotated script with shots	Animatic	Animatic

The approach requires no artistic skills because the content is usually obtainable from a built in content library or from external sources such as online asset libraries. Table 2.4 lists and compares several examples of simplified animation tools that can be used for creating pre-visualisations.

The environment of each scene is created using a GUI with an interactive 3D virtual environment. The virtual environment of each scene contains models for the set and each prop and character. It is populated by dragging the assets from an asset window or asset menu and dropping it on a view of the virtual environment. The view can be a 3D perspective view such as one used by iClone and Moviestorm (see Figure 2.23) (Reallusion 2012; Moviestorm 2011). A 2D top-down representation of the set can also be used build the set, to place props and block characters (see Section 2.4). Walls, door and windows can be added for indoor scenes and outdoor models can be added for outdoor scenes (Innoventive Software 2009).

Content libraries often contain predefined animations and facial expressions that can be used across a variety of characters. The animation process is further simplified by providing the facility to automatically generate animations using motions paths and object relationships. The director does not have to specify the individual body part movements required. Some

animation tools, like Moviestorm, provide a context sensitive popup menu that is similar to the popup menus used in the popular Sims™ strategic life-simulation video game (Electronic Arts 2009).

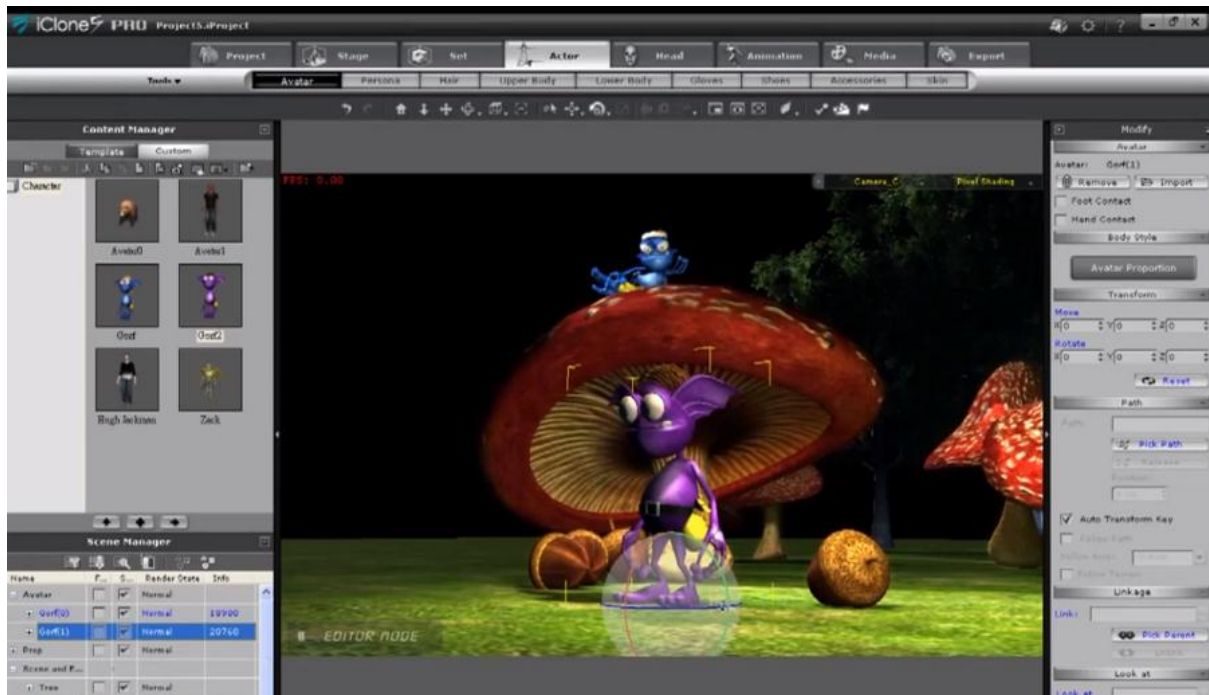


Figure 2.23: A screenshot of iClone 5 (Reallusion 2012).

Simplified animation tools are useful for planning the shots for a scene as well. Cameras can be added and manipulated using controls. Each camera shot can be specified using the environment view and camera controls (Innoventive Software 2009).

The final step in producing a pre-visualisation using this approach is to use the tool to generate the final pre-visualisation. Minimalistic pre-visualisation tools, such as FrameForge 3D Studio, are capable of generating a series of still images which are combined with the script in order to produce an annotated script with storyboard panels and camera instructions. Detailed pre-visualisation tools, such as Moviestorm and iClone, can generate animatics with sound and music.

The main advantage of using simplified animation tools is that they provide the director with an easy to use environment for graphically authoring pre-visualisations without requiring artistic skill or modelling and animation expertise. Automatic character animation and drag-and-drop user interfaces simplify and quicken the pre-visualisation authoring process.

The main disadvantage of using this approach is that it is based on traditional Windows Icons Menus Pointer (WIMP) interfaces (Sharp *et al.* 2007). The interaction method does not

intuitively fit with the basic activities involved in the pre-production phase. Figure 2.24 illustrates the flow of these activities. The director writes annotations onto the script and diagrams the action. Then the storyboard artist sketches the storyboard. Finally, a pre-visualisation is created using a WIMP interface by clicking on controls and dragging objects until the desired outcome is achieved (Machado, Gomes and Walter 2009). A more intuitive approach would be to use a storyboarding metaphor with a sketch-based user interface to author the final pre-visualisation. This makes the transition from storyboarding to pre-visualisation authoring smoother in the sense that sketching a storyboard and sketching a pre-visualisation is almost the same thing. The next subsection critically reviews previous work that closely aligns with this idea.

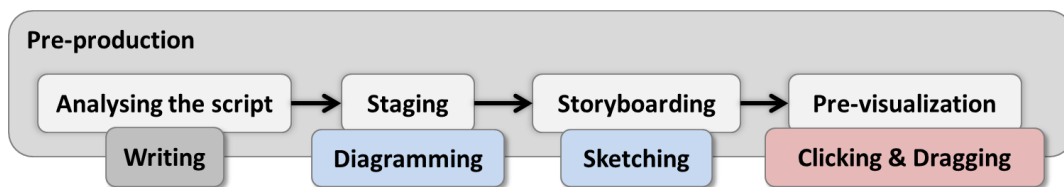


Figure 2.24: The basic activities involved in each part of the pre-production phase.

2.6.2.4 Sketch-based Storyboarding Tools

The storyboarding approach is similar to simplified animation except that the author is provided with a sketch-based GUI that supports existing storyboarding idioms and iconic conventions (see Section 2.5). There are two approaches for using sketch-based storyboarding interfaces for authoring pre-visualisations.

The first approach involves sketching the storyboard in a 2D environment. The storyboard artist sketches the set, the props and the characters in each storyboard panel. Actors are added to the storyboard by manually associating their sketches with their models in the 3D environment. This is done by using the click-and-drag approach used by simplified animation tools. The mapping between each sketched object and its corresponding model is therefore made manually. There is an implementation of this approach called Longboard which provides a shot planning algorithm (Jhala *et al.* 2008). It uses pre-defined character location markers as well as character bounding rectangles from the sketch in order to estimate the virtual camera. The props sketched in the background are not interpreted in order to establish the position and orientation of the virtual camera. The approach does not automatically recognise which model a prop in the sketch represents and it does not estimate the location and orientation of the model in 3D space.

The second approach involves sketching in a 3D environment (Toon Boom Animation Inc. 2012). Each sketch in the storyboard is represented by a collection of 2D layers that are contained inside a 3D space. The individual 2D layers can be moved, rotated and scaled in 3D space. The storyboard artist can add models from an asset library using the same click-and-drag approach used by simplified animation tools. The 2D layers can interact with the 3D models as they are manipulated or sketched. For example, Figure 2.25 shows a screenshot of Storyboard Pro 3D showing the storyboard artist sketching a 2D figure of a character on a pirate ship. The figure is manually placed on the ship's deck by moving the 2D layer in 3D space or by using a top-down view. The individual sketches are not automatically interpreted by the storyboarding tool to determine what each layer represents.

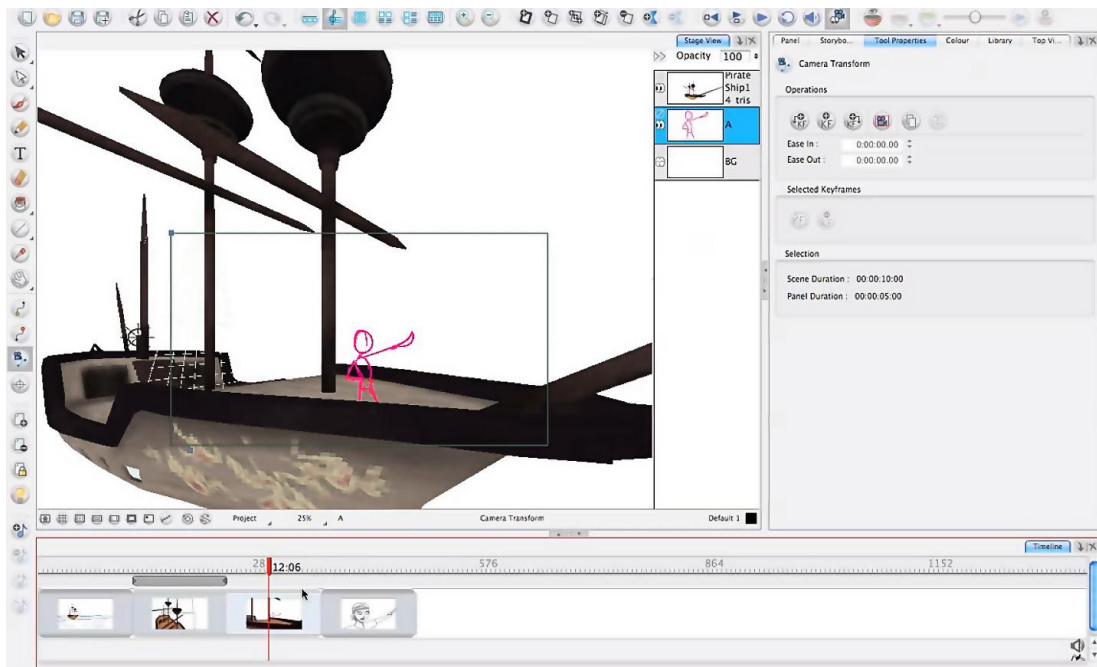


Figure 2.25: A screenshot of Storyboard Pro 3D (Toon Boom Animation Inc. 2012).

Existing tools, such as Storyboard Pro 3D do not perform object recognition or pose estimation to automatically convert the 2D layers into 3D models. The user is required to manually specify and position each model using click-and-drag interaction. The virtual camera has to be specified in the same manner. An animatic can be generated once the storyboard panels have been sketched and the 3D models have been added.

The main advantage of the sketch-based storyboarding approach is that it provides a sketching interface for authoring pre-visualisations. This makes the transition from storyboarding to pre-visualisation authoring smoother. It does not require the storyboard artist to have programming knowledge or modelling and animation expertise. Instead, the storyboard artist

can capitalise on existing storyboarding skills and use asset libraries to create pre-visualisations quickly and easily.

Existing sketch-based storyboarding tools for authoring pre-visualisations have the disadvantage of not automatically interpreting the contents of each storyboard sketch. The storyboard artist is required to manually add 3D objects and associate them to their corresponding elements in the storyboard sketch.

2.6.3 Discussion

Several text-based and graphical pre-visualisation authoring approaches were discussed in this section, and each approach introduced advantages over the approach preceding it. Figure 2.26 outlines the flow of authoring approaches, showing how each approach improves on the previous one.

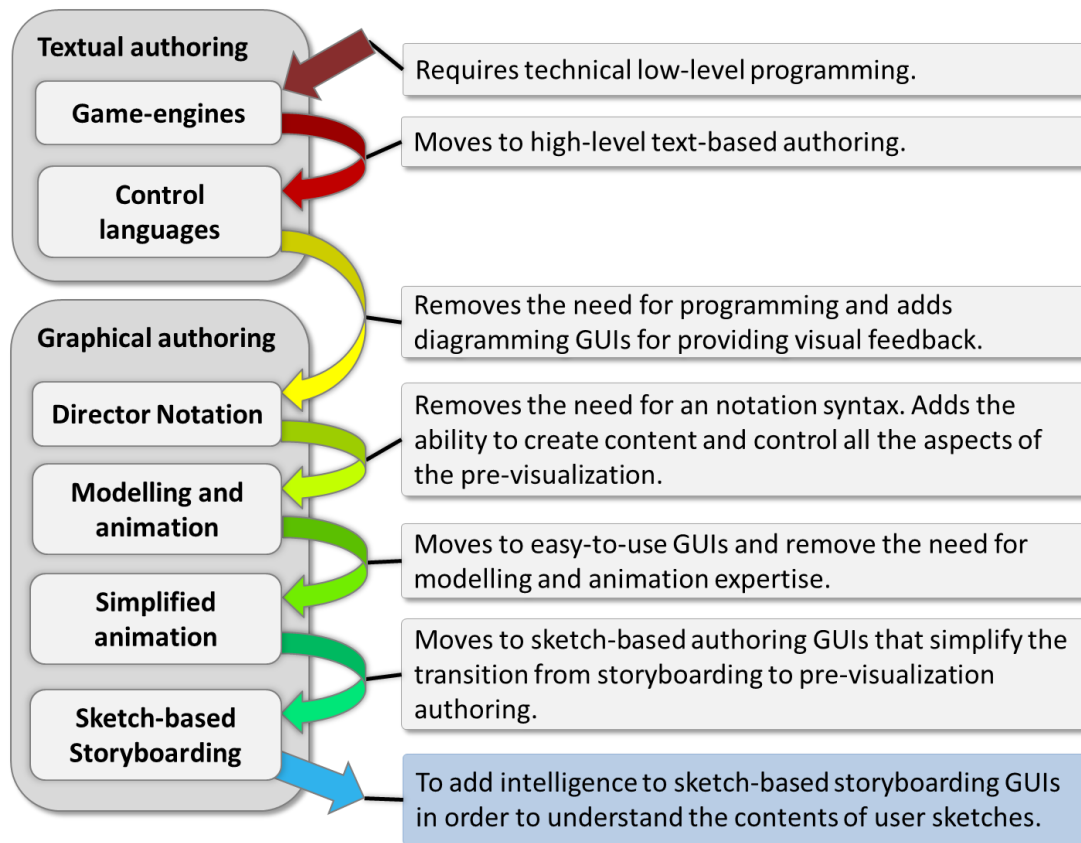


Figure 2.26: How authoring approaches improve.

The first approach that was discussed involves using game-engine code bases to programmatically author pre-visualisations. The problem with this approach is that it requires the author to possess strong programming skills. It was also pointed out that pre-visualisation programming is at a low and technical level and that game-engines are specifically designed

for game play. This introduces problems for controlling the camera and characters. Control languages address these problems by providing a high level text-based pre-visualisation authoring environment that makes it easier to program character performances and camera behaviour. Control languages are independent of the game-engine used to render the final pre-visualisation so each control language script can be re-used for different applications. Directors and storyboard artists prefer graphical authoring approaches over text-based authoring approaches, because they provide immediate visual feedback throughout the authoring process.

The Director Notation (DN) allows the director to graphically diagram the action of each scene using a GUI and an authoring syntax. The director does not have to author the pre-visualisation using game-engine code or control languages. The disadvantage is that it requires the director to learn the syntax of the DN. This is a high level authoring notation and many aspects of the pre-visualisation are often inferred implicitly, e.g. character poses. Modelling and animation tools address this problem by providing the director with full control of every aspect of the pre-visualisation.

Modelling and animation tools are used to create and animate the props, characters and environments. Modelling and animation software have complicated GUIs that require the expertise and experience of skilled modellers and animators. Simplified animation software addresses this problem by providing simplified easy-to-use GUIs that are designed from the director's perspective instead of that of the modeller or animator. The approach focusses on drag-and-drop interaction and terminology that the director is familiar with instead of technical jargon. Asset libraries make it possible to populate and animate the contents of virtual environments easily and quickly in order to author pre-visualisations. The main problem with simplified animation tools is that the interaction technique is based on clicking and dragging on WIMP based GUIs (Machado *et al.* 2009). This does not intuitively fit into the flow of basic activities performed during the pre-production phase.

Current graphical pre-visualisation authoring approaches involve clicking and dragging. A more intuitive interaction method would be to sketch. This makes the transition from storyboarding to pre-visualisation authoring smoother in the sense that sketching a storyboard and sketching a pre-visualisation are almost the same thing.

Previous work has investigated sketch-based storyboarding for authoring pre-visualisations; however, the sketches made by the storyboard artist are not automatically interpreted. The

storyboard artist is required to manually add 3D objects and associate them with their corresponding elements in the storyboard sketch (Jhala *et al.* 2008).

The next section discusses a case study which investigates how pre-production and pre-visualisation works in practice. It also provides a set of requirements for a pre-visualisation authoring tool that uses the sketch-based storyboarding approach.

2.7 Case study: Pre-visualisation for animated films

This section provides a practical view of storyboarding, pre-visualisation and the filmmaking process within the context of a case study. The case study is based on an interview conducted with a producer from a South African animation studio called Triggerfish Animation Studios (2012a). A brief review of Triggerfish's background is provided as well as the methodology and outcomes of the interview.

2.7.1 Background of Triggerfish Animation Studios

Triggerfish was established in 1996 as a stop-frame animation studio. The studio produced animations for a South African animated series called Takalani Sesame Street (Takalani 2011). It also created several South African commercials in order to grow and generate revenue. In 2008, Triggerfish moved from stop-frame animation to Computer Graphics (CG) animation technology and gained positive international exposure after it produced a 30-minute animated CG episode of the US series, *Life at the Pond* (2012). In 2012, the studio produced its first full length feature film called *Zambezia* (see Figure 2.27). The film was awarded the Best South African Feature Film award at the Durban Film Fest (Screen Africa 2012).

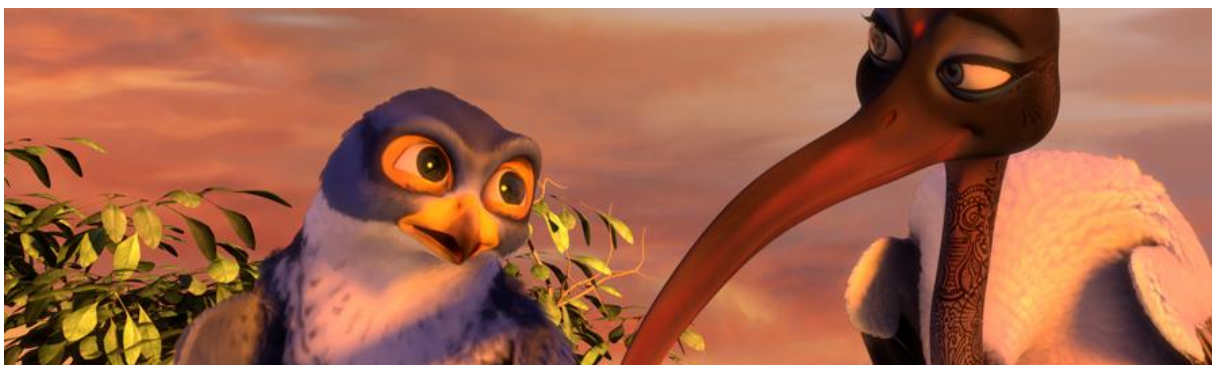


Figure 2.27: A scene from *Zambezia* (Triggerfish Animation Studios 2012b).

2.7.2 Interview Methodology

The overall interviewing process was conducted in three stages as shown in Figure 2.28 (see Appendix D and Appendix E). The preparation stage involved gathering information about the studio's background. A list of questions, topics and themes was prepared. These were addressed in order to achieve the following four goals:

1. To investigate how Triggerfish's filmmaking process compares with the traditional filmmaking process (see Section 2.2).
2. To investigate how storyboarding and pre-visualisation works at the studio.
3. To identify problems and areas open to improvement.
4. To determine the requirements of a sketch-based pre-visualisation authoring tool that uses a storyboarding approach.

The final preparation steps included sending an agenda to the interviewee and scheduling a time and venue. The next phase was to conduct the interview. The interview with Triggerfish was semi-structured. The order of the questions could change depending on the flow of the conversation. The discussion was captured (with permission) using field notes and audio recordings. If the interviewee introduced relevant themes that were not on the agenda then they were also discussed. The semi-structured approach was chosen, because the goals of the interview were exploratory in nature. The final phase of the interview was to transcribe the audio recording so that the data could be checked with the interviewee for accuracy and then analysed.

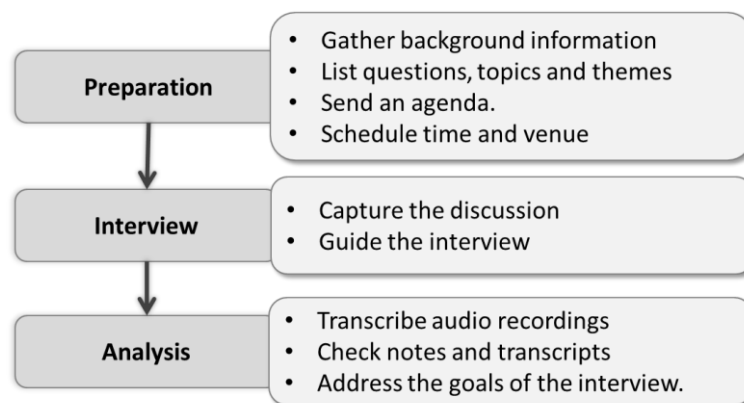


Figure 2.28: The interviewing methodology used (Oates 2006).

2.7.3 Outcomes of the interview

The main objective of the interview with Triggerfish was to investigate how the company creates storyboards and performs pre-visualisation. This subsection begins by comparing

Triggerfish's filmmaking process with the traditional filmmaking process (see Section 2.2). It continues by discussing how storyboarding is done at the studio and what the benefits and problems are. A short discussion on Triggerfish's pre-visualisation process is also provided. Several problems and areas open to improvement are also identified. The final outcome was to determine the requirements of a sketch-based pre-visualisation authoring tool that uses a storyboarding approach.

2.7.3.1 The Filmmaking Process Used at Triggerfish

The filmmaking process used at Triggerfish is similar to the traditional filmmaking process discussed in Section 2.2. The process starts with the development phase, which involves writing the script. This is where most of the exploration is done because it allows for the least expensive and fastest changes.

The pre-production phase begins by analysing the script to identify scenes, narrative beats and shots (see Section 2.3). The analysed script is then given to storyboard artists who are responsible for illustrating the scenes visually (see Section 2.5). Staging is performed by the storyboard artist as well (see Section 2.4). In traditional films, pre-visualisation can begin as soon as the storyboards are available. However, the filmmaking process at Triggerfish requires low fidelity assets to be created first using Softimage (see Section 2.6.2.2). The process of building content for the film is also part of the production phase. This means that the Triggerfish crew perform pre-visualisation activities at the end of the pre-production phase and at the start of the production phase as shown in Figure 2.29.

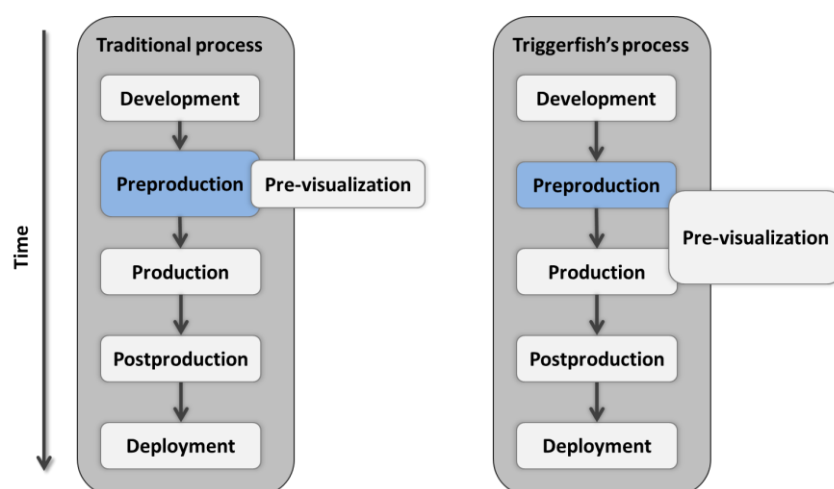


Figure 2.29: Comparing the traditional film making process with Triggerfish's process.

The production of each scene begins when the scene's pre-visualisation has been approved and the required assets are ready. The pre-visualised scene and high-resolution assets are then

given to the animators who do the performance animation. They also optimise each scene by removing unseen content in order to reduce its rendering time. The scene is given to the lighting team which is responsible for lighting and rendering each scene. Rendering is normally done in single passes. The passes are then composed together afterwards.

The next phase is postproduction. All the scenes are graded during this phase. Grading involves checking for consistent lighting and colouring across all the shots. The film is checked on a cinema display once the animation has been graded. Music and audio effects are added when the film is edited. The final phase is the distribution phase. The film is marketed and released to selected cinemas or other media outlets.

2.7.3.2 Storyboarding at Triggerfish

The Triggerfish storyboarding team is responsible for sketching the storyboards required during pre-production. The team uses storyboarding software developed by Toonboom called Storyboard Pro in order to create the storyboards (see Section 2.5.5). The sketching tools used by the individual storyboard artists depend on their individual preferences. Some artists scan in paper based storyboards and others use Photoshop or Storyboard Pro. The choice of the storyboarding tool is based on its usability and how quickly the artist can work with it.

Storyboard sketches are very rough estimations of the envisioned film. They are used to quickly develop, refine and document ideas. Triggerfish's storyboards show character outlines, basic poses and emotion. They contain limited detail on the environment for each scene. The storyboards rarely illustrate shading and colour, unless it is used to indicate the principle character. The characters are represented by simple contour sketches. Shading is also used occasionally to indicate special lighting conditions that are important for the story. Each character and prop is on a separate layer so that it can be animated independently from the other elements of the sketch. 2D animation is used to illustrate character movement using translation, rotation and scaling operations.

Annotations are also made on the storyboard. The storyboard artist can draw arrows to indicate the direction of movement. Arrows are also used to indicate when the camera moves in closer or further away. Storyboard Pro uses these annotations to create 2D animation for simple camera motion such as panning and zooming. The director instructs the storyboard artist on the appropriate shots for each scene.

The interviewee identified several reasons why storyboarding is beneficial to Triggerfish's filmmaking process. The main advantage of storyboarding is that it allows the crew to quickly

visualise each shot and see how the action unfolds. Triggerfish also adds audio dialogue to the storyboards in order to preview character performances. The storyboards document the graphical decisions made during the pre-production phase. A record of each storyboard sketch is made for later reference. Another advantage of storyboarding is that it informs the pre-visualisation team about character blocking, action and camera setups. Storyboards are therefore required as input for pre-visualising films at Triggerfish.

The interview also identified several problems that storyboarding poses during pre-production at Triggerfish. One of the problems of storyboarding the entire film is that it becomes a tedious and expensive task to track the shot number of each storyboard panel as the storyboard changes. Another problem is that the sketches used at Triggerfish are 2D illustrations which provide limited or no detail of the environment. The interviewee emphasised this problem by making the following statement:

“...we are finding that there’s a huge jump between drawing something in 2D where your artists can do a lot of cheats where the artists doesn’t really take into account perspective in real 3D space. So often, when it gets to the pre-vis guys upstairs they find that this can’t really work.”

The interviewee pointed out with this statement that the storyboards seldom provide sufficient information to the pre-visualisation team when the 3D space is important. It may be time-consuming or impossible to physically implement the storyboard artists’ illustrations because of the assumptions they made about the 3D space of the scene. Storyboards lack the 3D context provided by tools such as modelling and animation packages (see Section 2.6.2). This is where Triggerfish uses pre-visualisation to address this limitation.

2.7.3.3 Pre-visualisation at Triggerfish

The pre-visualisation stage is where the Triggerfish crew first gains an understanding of the technical requirements of the film. The pre-visualisation activity is used to identify what assets are required and at what level of detail. Triggerfish begins creating pre-visualisations at the end of the pre-production phase after the storyboards have been created. The required assets are created before beginning pre-visualisation. The resolution of each asset is reduced and the model is configured with low quality hierarchical structures for simple posing. The pre-visualisation team is responsible for manually translating the storyboards into animated pre-visualisations using an animation package called Softimage (see Section 2.6.2.2). This

manual process is a lengthy and costly task, but it is a crucial step to take before performance animation can begin. The pre-visualisation team uses the storyboard as a guide for blocking the characters in each scene and roughly animating their performances. Camera setups and shots are also inferred from the storyboard and implemented in the pre-visualisation.

2.7.3.4 Problems and Areas open to Improvement

The interview identified the following four problem areas of the Triggerfish filmmaking process that are open to improvement:

1. Shot management
2. Pre-visualisation authoring
3. Transferring from pre-visualisation to performance animation
4. Time requirement to reach the pre-visualisation stage

Managing shot information is problematic throughout the filmmaking process. Each shot is associated with a scene that is described by a block of script, an image and audio dialogue. The management of shot information is currently a manual process that requires the use of shot numbers. These shot numbers change as shots are added, removed or reordered. The interviewee proposed automating the management of shot information using an information system. It was proposed that shot information be stored as self-contained units which can be identified without referring to their shot numbers. All the shots of the film can then be stored in a database for efficient management.

The pre-visualisation authoring approach used by Triggerfish can also be improved. The pre-visualisation team manually places each character and prop and positions the camera so that the pre-visualised shots match the corresponding storyboard panels as close as possible. This involves using Softimage's 3D manipulation tools to manually pose the individual parts of each character and prop in the shot. It is a very time consuming and tedious process.

The translation from pre-visualisation content to performance animation is also a manual and time consuming process because blocking and posing data is not digitally transferred into the performance animation workspace. Instead, performance animators are required to manually pose characters from pre-visualisations. A more intuitive approach would be to perform the initial setup of character blockings and performances from pre-visualisation data automatically. Performance animators can then start animating instead of doing this setup manually.

The time required to reach the pre-visualisation stage is also problematic for the studio. Animators need to be able to represent 3D content at the storyboarding phase so that pre-visualisation can start earlier. A tool is therefore required that can be used during the fast-paced storytelling and the pre-visualisation phase.

2.7.3.5 Requirements of a Sketch-based Pre-visualisation Authoring Tool

In this research, it is proposed that a sketch-based interface is used for authoring pre-visualisations in a storyboarding context. The interviewee believed that sketching would be a faster and more intuitive approach to author pre-visualisations compared to using 3D manipulation controls to pose characters manually and set up the camera. There is therefore a need for a tool that can support sketch-based pre-visualisation authoring. The tool should support the activities performed during the pre-production phase. This includes analysing the script, developing a shooting strategy, creating storyboards and authoring pre-visualisations. It should be possible to import an existing script that has been structured in terms of scenes and dramatic blocks (see Section 2.3).

A floor plan editor is required in order to place props on the set and plan the shooting strategy (see Section 2.4). This includes specifying the movement of each character and indicating the shots involved in each scene. The floor plan editor should have basic manipulation functionality for moving and rotating objects on the set. A sketch-based interface would be desirable for staging because floor plans are traditionally made using paper-based sketches.

The tool should also provide a GUI for sketching storyboard panels. Various types of pens for drafting, shading and finalising sketches are required. It would be an advantage if the storyboard artist could sketch on a tablet computer because a stylus makes sketching easier and more natural for the storyboard artist. The GUI should be designed to be touch-friendly so that the sketching area can be manipulated using touch input and gestures. The sketching environment should also support the layering concept used by existing sketching software (see Section 2.5.5). Each character and prop should be sketched on a separate layer so that erasing and sketching is specific to a particular layer. The tool should automatically generate the required storyboard from the storyboard sketches and the floor plan. It should also automatically number the shots throughout the film.

An intelligent method of automatically converting 2D illustrations into 3D content is required in addition to the basic 2D sketching functionality. It should be able to interpret the symbols

sketched on the floor plan editor in order to automatically place and orientate the props, characters and shots for each scene. It should also be able to interpret the 2D illustrations of each storyboard panel sketched by the storyboard artist automatically in order to place the camera in the 3D scene and create the pre-visualisation with props and characters. The characters should be in the pose sketched by the storyboard artist and show the correct facial expression.

In general, the floor plan and the storyboard should be linked and automatically updated as each scene is authored. Changes in the floor plan must reflect on the storyboard and vice versa. The process of creating a pre-visualisation should be as simple and fast as possible. It is therefore important that the tool is easy to use and follows standard guidelines for touch interaction.

2.8 Summary of Sketch-based Pre-visualisation Requirements

This section provides a summary of the objects, symbols, annotations, graphical devices and other elements that are used in floor plans and storyboarding during the pre-production phase. It also provides a summary of the pre-visualisation requirements that have been gathered throughout this chapter. The requirements are summarised from the literature review and the interview conducted with Triggerfish Animation Studios.

Table 2.5¹ provides a summary of elements used in floor plans. It provides an element identified, a description of each element as well as the input method used by existing tools to create the element on the floor plan. Table 2.6 provides a similar summary for storyboards. Table 2.7 provides a summary of the pre-visualisation requirements in terms of functionality, usability, information, semantics and technical requirements.

Table 2.5: Elements used in floor plans.

ID	Element	Description	Input Method
F ₁	Prop	Indicates type, location and orientation of a prop.	Click-and-drag
F ₂	Blocking symbol	Indicates the location, orientation and name of a blocking for a character.	Click-and-drag
F ₃	Shot symbol	Indicates the location and orientation of a camera	Click-and-drag
F ₄	Character/camera motion path	Indicates the movement of a character/camera from one blocking/shot to another	Sketched, interpreted
F ₅	Set details	Illustrates the environment of the set.	Click-and-drag

¹ Elements that are highlighted in blue will be addressed in this research.

Table 2.6: Elements used in storyboards.

ID	Element	Description	Input method
S ₁	Character dialogue and action	Indicates what characters are saying and doing for a particular shot.	Click-and-drag / Typed
S ₂	Scene number	Identifies which scene a shot belongs to.	n/a
S ₃	Panel number	Identifies the shot for a storyboard panel.	n/a
S ₄	Stick figures	Illustrates character poses and body language	Sketched, not interpreted
S ₅	Shading	Highlight principle characters and illustrates lighting	Sketched, not interpreted
S ₆	Character faces	Expresses character emotion visually	Click-and-drag
S ₇	Character appearance	Illustrates how a character appears	Click-and-drag / Sketched, not interpreted
S ₈	Thumbnail sketches	Used for roughly planning storyboard panels in the margin of the script	Sketched, not interpreted
S ₉	Environment and props	Illustrates the camera setup for a shot and the environment	Click-and-drag / Sketched, not interpreted
S ₁₀	Special effects	Illustrates environmental effects	Click-and-drag / Sketched, not interpreted
S ₁₁	Objects state devices	Indicates if objects are glowing, shiny, hot, etc.	Sketched, not interpreted
S ₁₂	Movement lines, paths and clouds	Indicates how objects are moving	Sketched, interpreted
S ₁₃	Emotion devices	Illustrates the emotional expressions of characters	Sketched, not interpreted
S ₁₄	Camera annotations	Illustrates how the camera is zoomed, panned and dollied	Sketched, interpreted
S ₁₅	Shot transitions	Illustrates transitions between shots	Click-and-drag
S ₁₆	Layers	Used to separate individual props and characters when sketching.	Sketched, not interpreted
S ₁₇	Brushes	Used to create various types of digital strokes, e.g. pencil, paint brush, spray, etc.	Sketched, not interpreted
S ₁₈	Audio	Used for pitching actor and narrative performances	Recorded

Table 2.7: A summary of the requirements for a sketch-based pre-visualisation authoring tool using a storyboarding approach.

Requirement	Description
Function requirements	Support staging by creating floor plans and automatically generate storyboards if the shots are specified.
	Support for automatically placing props, characters and cameras on the set using symbols on the floor plan editor.
	Support storyboarding using sketch-based GUI for sketching 2D illustrations.
	Allow the emotional expressions of each character and their body postures to be indicated in each shot.
	Support for automatically interpreting the user's 2D illustration in each storyboard panel in order to place the camera and characters on the set.
	Allow content to be added to the pre-visualisation using a 3D view of the scene.
Usability	Usability and working quickly is very important.
	Seamless switching between pre-visualisation and storyboarding is important.
	The setup of pre-visualisations should be as fast as possible in terms of user effort and time.
	The simpler content is to draw, the better. Minimise sketching time. For example, drawing a symbol.
	The user interaction method should be a hybrid between sketch-based input and touch-based input.
Information and semantics	Sketch fidelity is at contours, basic poses, stick figures level of detail.
	Support for annotations that have no semantics for the system, such as character shading and strokes, is required. These may be useful for the director but they are not important for pre-visualisation generation.
Technical requirements	Pre-made content is available. No content generation is required.
	A digital link between storyboards, pre-visualisation and scripts is required.
	Packaging each shot as a unit of information.
	Synchronization between the floor plan and the storyboard is required.

2.9 Conclusions

This chapter addressed the first research question listed in Chapter 1, namely *what methods and theories have been developed for authoring pre-visualisations?* The chapter answered this question by answering the sub-questions provided in Section 2.1.

Q_{1.1} was answered by discussing the various activities involved in development, pre-production, production, postproduction and distribution phases of filmmaking. Q_{1.2} was answered by focusing on the pre-production activities of the filmmaking process including script analysis, staging, storyboarding and pre-visualisation. It was found that floor plans are useful for documenting the shooting strategy for each individual dramatic block (see Section 2.4 and Table 2.5). It was shown how storyboard artists illustrate characters, emotions, environments and dynamics using sketches, annotations and graphical devices (see Section 2.5 and Table 2.6). A review of storyboarding software provided insight into what storyboard artists would expect from a storyboarding tool with a sketching interface.

Q_{1.3} was answered by discussing pre-visualisation and several textual and graphical pre-visualisation authoring approaches, including game-engine code, control languages, DN, modelling and animation tools, simplified animation tools and sketch-based animation tools. It was found that each approach provided an advantage over the previous approach and that the most intuitive and user-friendly approach was the sketch-based storyboarding approach.

This research proposes using a sketch-based interface for authoring pre-visualisations using a storyboarding approach. The approach makes the transition from storyboarding to pre-visualisation authoring smoother in the sense that sketching a storyboard and sketching a pre-visualisation are almost the same thing. The need for such an authoring tool was confirmed at an interview conducted with a producer from Triggerfish Animation Studios (2012a). A set of requirements for such a tool was gathered in order to answer Q_{1.4}. The requirements were identified from the literature review and the interview and summarised in Section 2.8. In particular, it was determined that the tool should be able to interpret the sketches made by the artist during staging activities and storyboarding activities.

Existing pre-visualisation tools focus on authoring detailed animated pre-visualisations using a click-and-drag authoring approach (Jhala *et al.* 2008; Jung *et al.* 2010). Limited research has been conducted on how user sketched floor plans and storyboards can be automatically interpreted for authoring pre-visualisations. The focus of this research is to investigate how a sketch/touch-based storyboarding tool can be designed and implemented in order to author *minimalistic* pre-visualisations within a 3D virtual environment. The following floor plan and storyboard elements (highlighted in blue) will therefore be addressed from Table 2.5 and Table 2.6:

- F_1 , F_2 , F_3 and F_4 will be included. F_5 will not be included because it involves modelling virtual environments. Pre-designed virtual environments will be used.
- S_1 , S_2 , S_3 , S_{16} and S_{17} will be included. These were requirements identified from the interview with Triggerfish Animation Studios.
- S_6 and S_9 will be included for automatic interpretation because existing tools do not interpret the user's sketches of these elements. S_7 will also be included but the appearance of each character will not be interpreted for recognition purposes. The character's pose will be extracted from S_4 and the user will specify which character is represented using touch-based interaction.
- The remaining storyboard elements will not be addressed by this research because they are required for authoring detailed, animated pre-visualisations. Existing storyboarding tools such as Storyboard Pro 3D is capable of interpreting user sketched annotations for character and camera behaviours (Toon Boom Animation Inc. 2012). This research project will focus specifically on interpreting the user's sketches of the props and characters in each storyboard panel.

The next chapter investigates several methods from computer vision for interpreting the user's sketches of the floor plan and storyboard elements selected above. It critically reviews methods for recognising objects from 2D images and placing them within a 3D context. This involves performing object recognition and pose estimation. The chapter provides a broad overview of the various methods available, but also focuses on the methods that are useful for analysing sketch-based input in order to answer the second research question, namely *what methods are available for extracting information from sketches?*

Chapter 3:

Understanding Sketches

3.1 Introduction

Authoring pre-visualisations from storyboards requires information from each storyboard sketch to be extracted in order to understand its contents. This chapter answers second research question identified in Chapter 1, namely *Q₂: what methods are available for extracting information from sketches?* This research question is answered by answering the following sub-questions:

- Q_{2.1}: How can sketches be described qualitatively?
- Q_{2.2}: How can sketches be compared with images in order to recognise objects?
- Q_{2.3}: How can the pose of a rigid body be estimated from a sketch?
- Q_{2.4}: How can the pose of an articulated body be estimated from a sketch?

The problem of understanding the contents of an image falls within the field of computer vision (CV) and it has two main sub-problems, namely object recognition and pose estimation (Pellegrini 2007):

1. *Object recognition* seeks to identify objects within an image and label each with the appropriate class name or instance name.
2. *Pose estimation* involves estimating the position, orientation and posture (if applicable) of all the objects identified within the image. The position and orientation of each rigid body, such as a prop, are estimated with a single coordinate frame. The posture of each articulated body, such as a character, is estimated using a set of nested coordinate frames where each coordinate frame represents the location and orientation of a body part.

Human perception inspired the use of models in computer vision. A *model* is a collection of information that is known for an object. It has been theorised that human perception integrates two perception approaches, namely the bottom-up approach and the top-down approach (Marr 1982). The *bottom-up* approach involves extracting information from the image and comparing it to the information available from the model. The *top-down* approach involves instantiating the model and using it to learn more about the object in the image.

Understanding the image (the user's sketch of a particular object) involves recognising the object. This is achieved by describing the image, classifying it and then associating it with an instance of the class model. The object recognition step is therefore a bottom-up approach (see Figure 3.1). The transformation parameters required to estimate the pose of the instantiated model is calculated from the image using pose estimation methods. The pose estimation step is therefore a top-down approach.

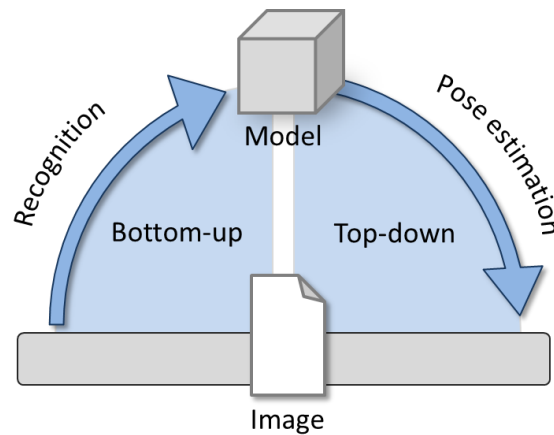


Figure 3.1: Bottom-up and top-down computer vision approaches (Marr 1982).

This chapter reviews the object recognition process as well as several methods for describing and comparing images for the purposes of recognising symbols and props from query images that represent user sketches. Methods for estimating the pose of rigid bodies and articulated bodies are also reviewed in order to estimate the location and posture of characters sketched by the user.

3.2 The Object Recognition Process

The overall object recognition process occurs in two phases: the offline training phase (see Figure 3.2 A) and the online recognition phase (see Figure 3.2 B). The training phase begins by sampling an object in order to create training images. Feature detectors are used to

determine points or regions of interest (see Section 3.3.2). The features are described and recorded in a codebook for the particular object (see Section 3.3.3).

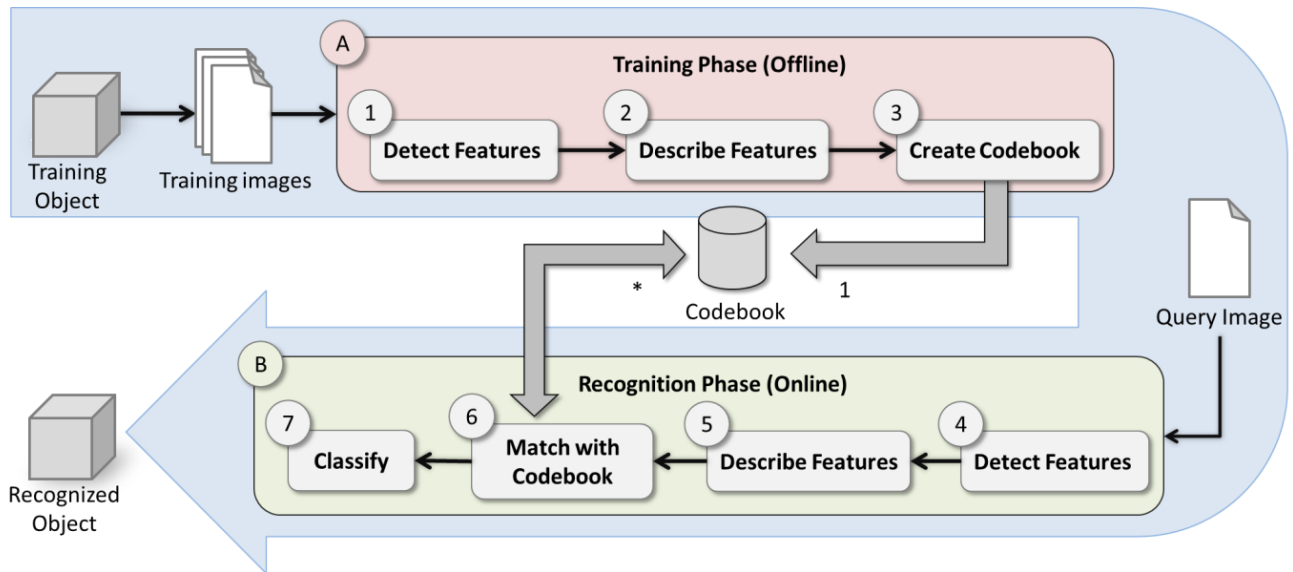


Figure 3.2: The general process followed by object recognition algorithms (Pellegrini 2007).

The object recognition phase accepts a query image as input. Similarly to the training phase, the object recognition phase involves detecting and describing the features of the query image. The codebook created in the training phase is now used to determine what object is most likely presented in the query image. This is achieved by first matching the features contained in each entry of the codebook with the features in the query image by determining feature correspondences (see Section 3.4.1). The query image is now associated with several candidate objects. The next step is to classify the query's feature set in order to determine which candidate is the most probable match (see Section 3.4.2). The most probable candidate is then considered to represent the object which best matches the query image.

The training phase and the recognition phase of the object recognition process involve describing training images and query images respectively. Images are often described in terms of their features. The next section discusses several approaches from computer vision literature for detecting and describing the features of an image.

3.3 Describing Images

A *feature* of an image is an image property that can be used to characterise the image or part of the image. This section defines two types of features, namely local features and global features. It also provides a detailed review of several local feature detectors (see

Section 3.3.2) and local feature descriptors (see Section 3.3.3). The section concludes with a brief overview of two global feature descriptors.

3.3.1 Local Features and Global Features

A *local feature* is an image property located on a point or small region of an image. It may be characterised by the colour, intensity and gradient values of the pixels in its local neighbourhood (Roth and Winter 2008). A local feature is interesting when it provides sufficient information to distinguish it from other local features. It is important that local features are invariant to the location, rotation, scale and illumination changes in the image. Local features should also be invariant to changes in the viewing direction of the observer. Selecting local features that support a high degree of invariance allows for more robust object recognition (Pellegrini 2007). Section 3.3.3 reviews how local features can be described using various local feature descriptors.

A *global feature* is an image property that quantitatively describes the entire image or sub-region of the image. Jolliffe (2002) illustrated how Principal Component Analysis (PCA) can be used to determine the principal components of a collection of points. The affine² parameters of the image can be determined from the principal components. Section 3.3.4 reviews how images can be described using PCA.

3.3.2 Local Feature Detectors

A *feature detector* is an algorithm that finds local features in an image. Local feature detection involves determining the location, orientation, scale and shape of each local feature in a robust manner. If the local feature detector provides insufficient local feature information, or the feature detector is not invariant to changes in the image, then the information encoded by the feature descriptor will be inaccurate and may misguide the recognition process (Battiatto *et al.* 2007).

This section discusses three types of local feature detectors. Corner detectors are used for identifying points of interest. Region detectors include rotation, scale and shape information; and edge detectors identify edges within the image (edges are typically found in areas where there are large changes in pixel intensity). The strengths and weaknesses of each local feature

² The affine parameters of an object represent scaling, rotation, translation and shearing transformation parameters.

detector are reviewed. The section concludes with a discussion on applying these approaches for detecting local features from user sketches reliably.

3.3.2.1 Corner Detectors

One of the earliest and most well know feature detectors is the Harris detector (Harris and Stephens 1988). The Harris detector, like many other feature detectors, considers the image as a function $I(x, y)$ where (x, y) is the location of each pixel and $I(x, y)$ is the greyscale value of the pixel. The Harris detector evaluates the “corner-ness” of each pixel by evaluating the eigenvalues of the second moment matrix of $I(x, y)$. If both eigenvalues³ are small in magnitude then the local region around the pixel is approximately constant in intensity. If one eigenvalue is large in magnitude and the other eigenvalue is small in magnitude then the pixel lies on an edge. If both eigenvalues are large in magnitude then the pixel lies on a corner. The main advantage of the Harris detector is that it can detect many local features quickly. The main disadvantage is that the Harris detector is not invariant to changes in the image. If the image is affected by affine transformations then the corners may not be detected correctly. In addition, the detector does not provide scale and shape information about local features.

Table 3.1: A comparison of several corner detectors (Roth and Winter 2008).

Detector	Transformation Invariance	Runtime	Number of detections
Harris	None	Very short	High
Hessian	Rotation	Very short	High
Harris-Laplace	Scale	Medium	Medium
Hessian-Laplace	Scale, Rotation	Medium	Medium

Several other corner-based feature detectors have been developed since the introduction of the Harris detector. A literature study conducted by Roth and Winter (2008) provides a review of several corner detectors (see Table 3.1). The Hessian matrix of the image function can be used instead of the second moment matrix to provide rotational invariance. Furthermore, the Harris-Laplace and Hessian-Laplace detectors provide additional scale invariance at the cost of performance by applying the normalised Laplacian in scale space (Mikolajczyk and Schmid 2001).

³ The second moment matrix is always a 2×2 matrix.

Figure 3.3 shows how corners can be detected from a user sketch using the Harris feature detector. The corners are detected around the turns of each stroke and they do not provide any scale or orientation information. This makes it difficult to describe each corner quantitatively because it is not known which region should be described. Instead, the image is described by the locations of all the corners relative to each other or a set of reference corners.

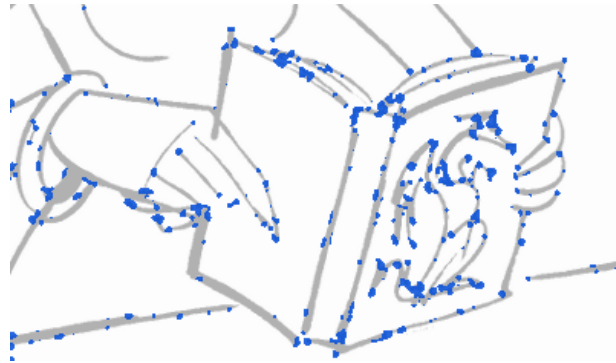


Figure 3.3: Harris corners for a sketch of a character holding a book (Glebas 2008).

3.3.2.2 Region Detectors

An alternative approach to edge detectors is to search for regions. Table 3.2 provides a comparative summary of several region detectors reviewed by Roth and Winter (2008). Region detectors are more desirable because they provide additional information regarding the shape, scale and orientation of each local feature. There are essentially two types of regions, namely free-form shapes and blobs. *Free-form shapes* define regions with arbitrary boundaries (usually pixels). Regions that lack clearly refined boundaries are known as *blobs* (Mikolajczyk and Schmid 2005).

Table 3.2: A comparison of several region detectors (Roth and Winter 2008).

Detector	Transformation Invariance	Runtime	Number of detections	Shape
Difference of Gaussian (DoG)	Scale and Rotation	Short	Medium	Blob
Harris-Affine	Affine	Medium	Medium	Blob
Hessian-Affine	Affine	Medium	Medium	Blob
Edge Based Regions (EBR)	Affine	Very long	Medium	Blob
Intensity Based Regions (IBR)	Projective	Long	Low	Blob
Maximally Stable Extremal Regions (MSER)	Projective	Short	Low	Free-form
Entropy Based Salient Regions (EBSR)	Affine	Very long	Low	Free-form

Blob detection involves determining the affine transformation parameters of each blob. Several region detectors are suitable for blob detection. One of the simplest blob detectors is the Difference of Gaussian (DoG) detector (Lowe 2004a). The DoG detector generates several images in a pyramid structure that represent the *scale space* (see Figure 3.4). Images within the pyramid are ordered from a high resolution at the base to a low resolution at the apex. The DoG detector generates each image by calculating the differences of several Gaussian blurred images at scales S_i and S_{i+1} where $S_{i+1} > S_i$. The variance of the Gaussian filters increase as the scale level of the image increases. The base of the pyramid contains a sharp image at a high resolution, and the apex of the pyramid contains a blurred image at a low resolution.

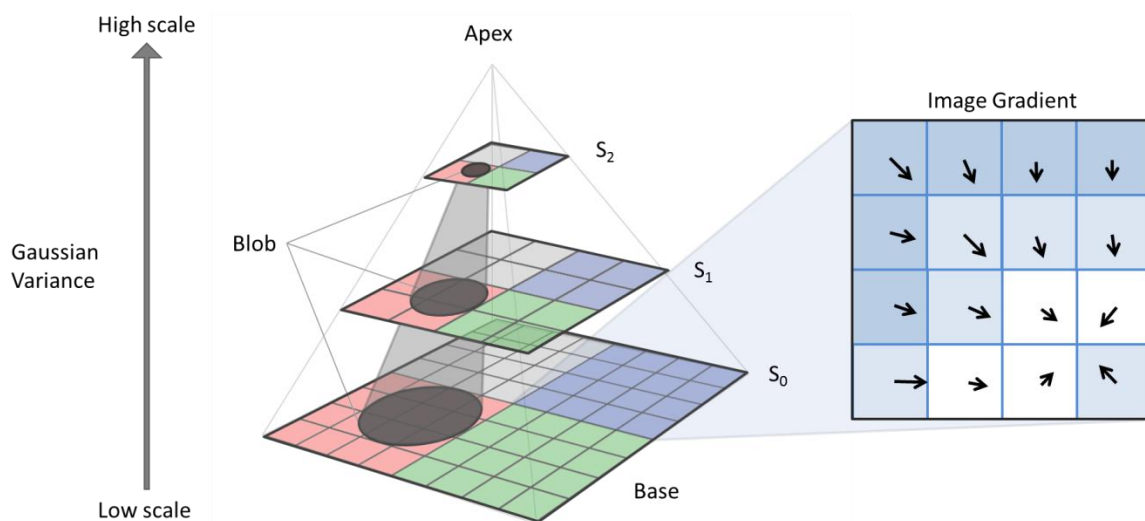


Figure 3.4: An illustration of the scale space of an image.

The location of each blob is obtained using a corner detector. Their scale is selected from the scale space in such a way that each blob appears similar across the scale space (see Figure 3.4). Their orientation is determined from the image gradient. The image gradient is calculated from the first order partial derivatives of the image's intensity function. The DoG detector is fast but it only provides scale and orientation information (see Figure 3.5 (a)). Affine transformation information is desirable for robust feature description.

The Harris-Affine detector and the Hessian-Affine detector provides location, shape and orientation information by making iterative estimations of the affine transformation of each local feature (Mikolajczyk and Schmid 2002). The disadvantage of the iterative approach is that it increases the runtime of the detector. Alternatively, the Edge Based Regions (EBR) detector can determine the affine transformation by examining local edge structures of local the features (Tuytelaars and Gool 1999). The latter method has a very long runtime. Similarly, the Intensity Based Regions (IBR) detector evaluates the intensity of pixels around each

interest point to determine their affine frames (Tuytelaars and Gool 2004). The IBR detector is slightly faster than the EBR detector but it is slower than the Harris-Affine detector and the Hessian-Affine detector.

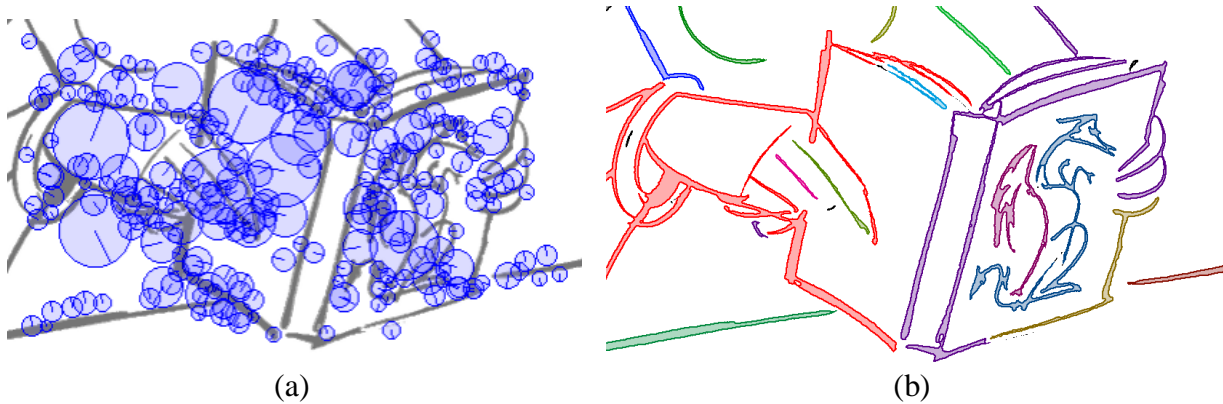


Figure 3.5: Local features with (a) blob structures and (b) arbitrary shapes for a sketch of a character holding a book (Glebas 2008).

Region detectors are available for identifying regions with clearly defined boundaries with arbitrary shapes. The boundary of each region is defined by the border pixels enclosing it. One approach for determining these boundaries is to consider the images obtained from applying all the possible binary thresholds. An area that does not change shape between consecutive thresholds is known as a Maximally Stable Extremal Region (MSER) (see Figure 3.5 (b)). The MSER detector is fast, robust and invariant to projective transformations and monotonic intensity changes (Matas *et al.* 2004). The number of features detected by the MSER detector is considerably less than the number of features detected by corner detectors. Similarly, the Entropy Based Salient Region (EBSR) detector uses a thresholding approach to determine regions of interest (Kadir and Brady 2003). Instead of applying the threshold on the intensity of each pixel, the EBSR detector uses a saliency measure of the pixel. The EBSR detector is robust and scale invariant but it has a very long runtime. Affine adaptations of the EBSR detector have been proposed but they have even longer runtimes (Mikolajczyk *et al.* 2005; Kadir, Zisserman and Brady 2004).

3.3.2.3 Edge Detectors

Corner-based feature detectors and region-based feature detectors analyse the texture of an image in order to identify regions of interest. In some cases, these approaches may not be able to detect invariant and distinguishable features in a robust and repeatable manner. Corner-based detectors often detect interest points that exist on edges. These interest points are called *edge responses* (Lowe 2004b). Figure 3.6 provides an example showing four blob regions for

an image containing edges. Blobs A, B and C are edge responses. The orientation of each blob is determined from the gradient of the image and the size is determined from the scale space of the image. If the blobs are compared irrespectively of their location, size and orientation then the edge responses A, B and C are nearly indistinguishable. Local feature detection is therefore optimised to reduce edge responses.

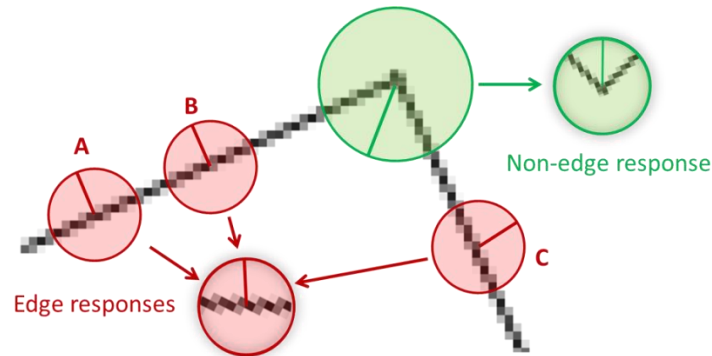


Figure 3.6: Indistinguishable edge responses.

Edges provide valuable information about the contents of an image. This is because the edges can be used to distinguish objects from one another and determine the shape of each object. Edge detectors are used for extracting edges from images. The majority of edge detection algorithms generate edge maps (see Figure 3.7). An *edge map* is a monochrome image where each pixel provides a binary indication of whether the corresponding pixel in the original image exists on an edge or not (Nadernejad and Sharifzadeh 2008).

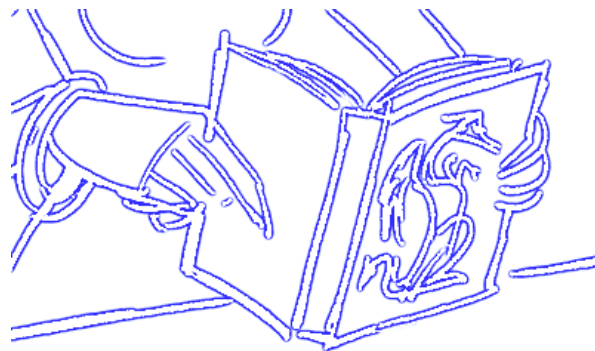


Figure 3.7: Edge features for a sketch of a character holding a book (Glebas 2008).

One of the earliest edge detectors is the Marr-Hildreth edge detector (Marr and Hildreth 1980). The Marr-Hildreth edge detector makes use of the first and second order derivatives of the smoothed image intensity function by applying a two dimensional Laplacian filter to the image. Pixels that undergo a sign change are then marked as edge pixels. The remaining pixels are marked as non-edge pixels. The Marr-Hildreth edge detector perceives many edges but they are often thick, disjoint or spotty.

The Canny edge detector was proposed for improved edge detection. Instead of applying a threshold on the pixels, the Canny edge detector compares the gradient of each pixel with gradients of its neighbours. Hysteresis thresholding is used to determine whether pixels are part of an edge or not (Canny 1986). The Marr-Hildreth edge detector and the Canny edge detector consider the gradient of the image, making it possible to determine the direction of edges (Nadernejad and Sharifzadeh 2008). This is useful when tracing edges for the purposes of extracting polylines and polygons from the image.

3.3.2.4 Discussion

There are two main challenges to consider when selecting a local feature detector for simple hand-drawn sketches. Firstly, no texturing information is available. This is because the user's sketches are not shaded and they primarily consist of minimalistic strokes. Secondly, the sketches may contain several anomalies caused when the user sketches in an arbitrary fashion.

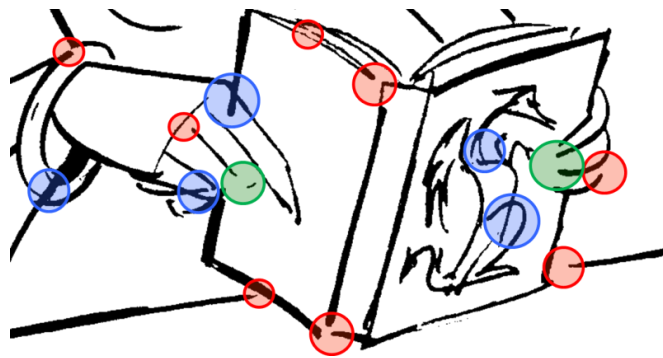


Figure 3.8: Sketching anomalies (Glebas 2008).

Figure 3.8 highlights three types of anomalies that can exist in a user's sketch:

1. Strokes may not be connected consistently (shown in red)
2. Strokes may be omitted (shown in green)
3. Unnecessary strokes may be included (shown in blue)

Corner detectors can identify corners reliably from user sketches but they do not provide scale and orientation information. Edge detectors and free-form region detectors (such as the MSER detector) detect edges and regions along the boundary pixels of each stroke. The problem is that edge detectors and free-form region detectors cannot be used reliably because the user's sketches are prone to the anomalies identified above. If any one of these anomalies is present then the detected edges or free-form regions may be significantly different to the local features detected during the training phase. This is problematic for reliable object recognition. These problems can be addressed by using blob-based region detectors.

Blob-based region detectors provide scale and orientation information for each local feature. The advantage of using blob-based region detectors is that they can be used to identify many local features. If some of the blobs are not consistently detected, or they are edge responses, then they can be omitted without significantly influencing the accuracy of the overall object recognition process (Roth and Winter 2008).

3.3.3 Local Feature Descriptors

A *local feature descriptor* quantitatively describes the appearance of the region around a local feature by measuring properties that are invariant to transformations and illumination changes in the image (Roth and Winter 2008). Local feature descriptors can be compared based on their rotational invariance, dimensionality, performance and the image type they are most suited for:

- The *rotational invariance* of a feature description determines if the description of the local region is dependent on its orientation in the image.
- The *dimensionality* of a feature description is the number of data values it contains. Feature descriptions with high dimensionalities are more expensive to compare.
- The *performance* of a local feature descriptor measures the extent to which local features of an image can be described and matched repeatedly with local features of another image (Mikolajczyk and Schmid 2005). The performance of a local feature detector is therefore directly proportional to the repeatability rate of achieving the desired matching results.
- The *image type* best suited for an image descriptor can be textured images or outlined images (see Figure 3.9).



Textured



Outlined

Figure 3.9: An example of a textured image and an outlined image (Ke and Sukthankar 2004; Glebas 2008).

If the local features are detected incorrectly then the local feature descriptions will be changed dramatically. This will lead to fewer matches during the feature matching step and the overall accuracy of object recognition will degrade. It is therefore clear that blob-based region detectors are better suited to identifying local features from user sketches because they are less prone to being severely affected by the anomalies mentioned in Section 3.3.2.4 (Mikolajczyk *et al.* 2005).

This section focuses on the description of blob-based local features. The local feature descriptors can be classified into three descriptor groups:

1. SIFT-based descriptors,
2. Grid-based descriptors, and
3. Shape-based descriptors

3.3.3.1 SIFT-based Descriptors

One of the most popular blob-based local feature descriptors is the Scale Invariant Feature Transform (SIFT) (Lowe 2004b; Lowe 1999). The SIFT is a combination of a detector and a descriptor for blob-based local features. Lowe called the blob-based local features *SIFT-keys*.

The SIFT-key detector is based on the DoG detector (see Section 3.3.2.2). The scale space pyramid and the Difference of Gaussian (DoG) pyramid are constructed for the image as a pre-processing step. The SIFT-keys are detected by scanning all the pixels of the image at different levels in the DoG pyramid. The resolution of the DoG pyramid is fixed for the SIFT descriptor and only the variance of the Gaussian function changes between scale levels (see Figure 3.10).

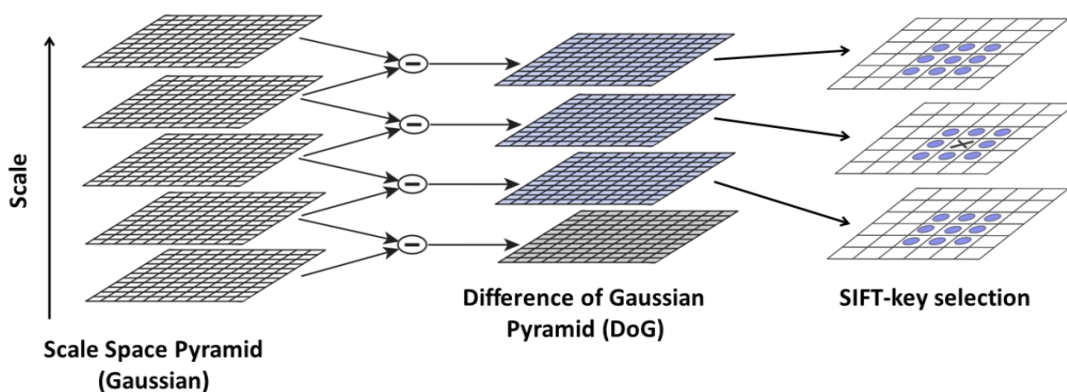


Figure 3.10: SIFT-key detection process (Lowe 2004a).

The neighbouring pixels at the current scale, the scale above and the scale below are considered for each candidate pixel (marked as “X” with 26 surrounding pixels). If the

intensity of the candidate pixel is greater than, or smaller than, all the 26 neighbouring pixels then it is detected as a SIFT-key with the current scale. A strict inequality $x < y$ can be used or a relaxed inequality $x < y + \beta$ for some threshold value β . The orientation of the SIFT-key is based on the gradient of the image at the current scale (see Figure 3.4).

Each SIFT-key is described as follows (see Figure 3.11). As a preparation step, the local region of the SIFT-key is normalised using its scale and orientation (Step 1). The region around the SIFT-key is divided into a 4×4 grid of non-overlapping patches (Step 2). A Gaussian weighted window with a standard deviation of 1.5 times the scale of the SIFT-key is used to weigh the contents of the SIFT-key. This allows the local feature to be transformed slightly without significantly affecting the resulting feature description.

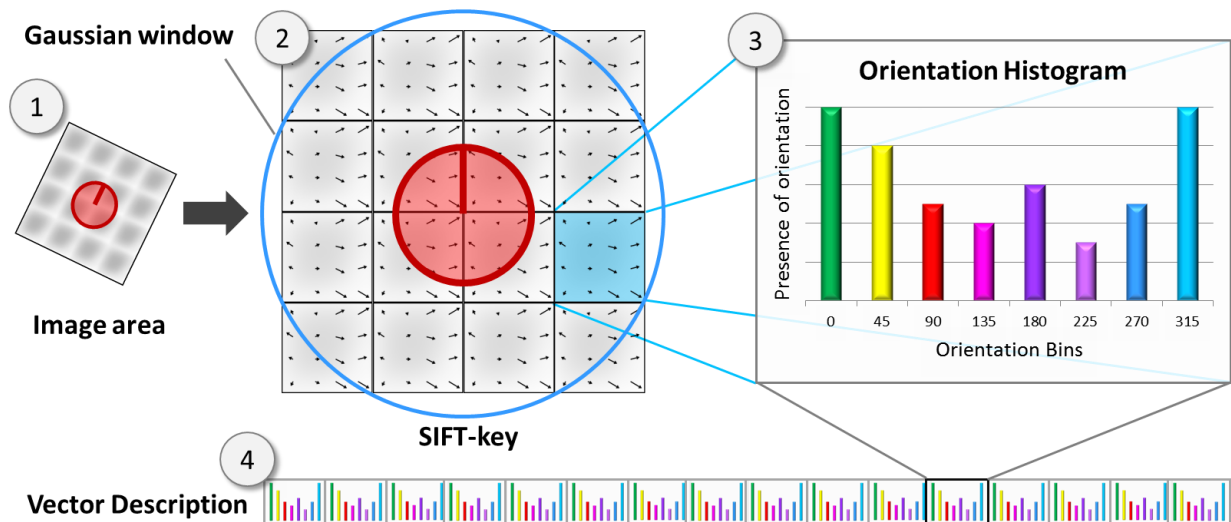


Figure 3.11: The SIFT-key descriptor (Lowe 2004a).

The SIFT-key descriptor is based on the orientation of the image's gradient vectors. The gradient vectors in each patch are weighted with the value of the Gaussian window at the location of the corresponding pixels. The orientations of the weighed gradient vectors for each patch are quantised into orientation histograms (Step 3). Lowe recommends using eight orientation bins for each patch. The values of the histograms are smoothed and concatenated into a single vector which is then normalised (Step 4). SIFT-key descriptions can be compared by measuring the Euclidean distance between their corresponding vectors. SIFT-key descriptor vectors therefore have a dimensionality of $4 \times 4 \times 8 = 128$.

In order to reduce the dimensionality for shorter runtimes, the Principal Component Analysis-SIFT (PCA-SIFT) descriptor uses PCA to determine the most significant eigenvectors instead of using all the gradient vectors. The PCA-SIFT descriptor provides shorter runtimes but at a

slight cost of performance (Ke and Sukthankar 2004). Mikolajczyk and Schmid (2005) proposed the Gradient Location-Orientation Histogram (GLOH), which utilises a circular arrangement of patches instead of a rectangular arrangement. They showed that it is more accurate for outlined images than rectangular SIFT-key descriptors.

SIFT-based local feature descriptors rely heavily on the availability of image gradient information. This poses a problem because the images sketched by the user mainly consist of simple strokes. No texturing information is available for the regions between the strokes. Change in the gradient of the image is therefore only observable near the strokes themselves. Removing edge responses (see Section 3.3.2.3) further decreases the number of SIFT-keys that can be detected by the SIFT-key detector. It can therefore be concluded that SIFT-based local feature descriptors should only be used on textured images that provide sufficient gradient information. Sketches can be described by combining SIFT-key descriptors with grid-based descriptors.

3.3.3.2 Grid-based Descriptors

One of the simplest methods for describing parts of an image uses grid-based descriptors (Sajjanhar and Lu 1997). A grid is placed over the image using a rectangular or elliptical arrangement of cells (see Figure 3.12). Grid-based descriptors require information about the scale, orientation and location of the image part. The normalisation step is therefore essential to the performance of grid-based descriptors.

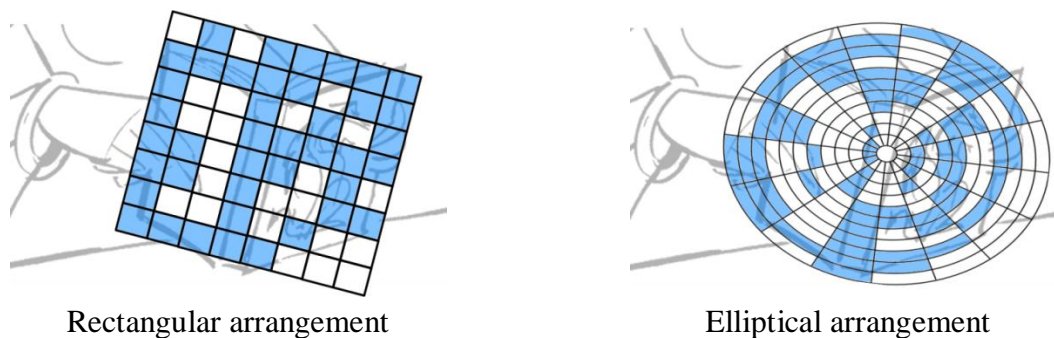


Figure 3.12: Grid-based descriptors configured in rectangular and elliptical arrangements.

The contents of each cell can be indicated using a binary representation or a real representation. *Binary grid-based descriptors* represent each region with a binary string. Each cell contains a 1 if and only if the cell intersects with an outline in the region. The dissimilarity between two binary grid-based descriptions is computed using the Hamming distance: summing the ones obtained by taking the exclusive-or of both strings (Sajjanhar and Lu 1997). *Real grid-based descriptors* represent each cell in terms of the mean intensity of the

outlines they contain. The intensities of the cells are transformed using the two dimensional Discrete Fourier Transform (DFT). This has the effect of removing high-frequency noise from the description and improving the performance of the grid-based descriptor (Zhang and Lu 2002).

A special case of real elliptical grid-based descriptors is when the grid is configured in a circular arrangement. Real, circular grid-based descriptors can be represented using spin images. A *spin image* is a 2D histogram of the intensity values and pixel distances to the centre of the circle. The 2D histogram representation allows for planar rotational invariance (Lazebnik, Schmid and Ponce 2003).

Grid-based descriptors provide a simple method for describing the contents of a user's sketch. There are, however, several complications when using grid-based descriptors to describe user sketches. It is crucial that the area being described is correctly normalised because grid-based descriptors do not provide translation, rotation and scaling invariance. Another issue is selecting the correct resolution for the grid descriptor. If the resolution is too low then the grid-based descriptor does not provide sufficient discriminative information. If the resolution is too high then the grid-based descriptor becomes too sensitive to noise in the image (such as the anomalies identified in the introduction of Section 3.3.2.4) (Shahabi and Safar 2007). An alternative to representing an image in terms of a grid is to describe the shapes of the objects contained in the image.

3.3.3.3 Shape-based Descriptors

Shape-based descriptors consider the edge structure of the local region being described. This makes shape-based descriptors well suited to describing outlined images like sketches because they do not require gradient information to be available (Zhang and Lu 2002). The local edge structure of each region is determined by detecting corners and tracing the edge map of the region. The edges and corners are then analysed in order to construct shape-based feature descriptions.

One of the simplest approaches to constructing a shape-based feature is to base it on the shape of the contour that surrounds the edges. This is achieved by sampling the distances along predefined radial lines from the centroid of the shape to its boundary (Ferilli *et al.* 2011). The distances are then quantised into a histogram where each bin represents the angle(s) of the radial lines sampled (see Figure 3.13 (a)). This distance histogram can be compared to other distance histograms using the Euclidean distance. A Discrete Fourier Transform can also be

applied to the distance function in order to remove noise (Zhang and Lu 2002). This approach is not invariant to scale and rotation and therefore requires the region to be normalised. The descriptions can, however, be compared irrespectively of the orientations of the corresponding regions by rotating one of the histograms with a finite number of angles in order to minimise the resulting Euclidean distance. The main limiting factor of the distance-based shape description approach is that it does not consider the internal details of the shape.

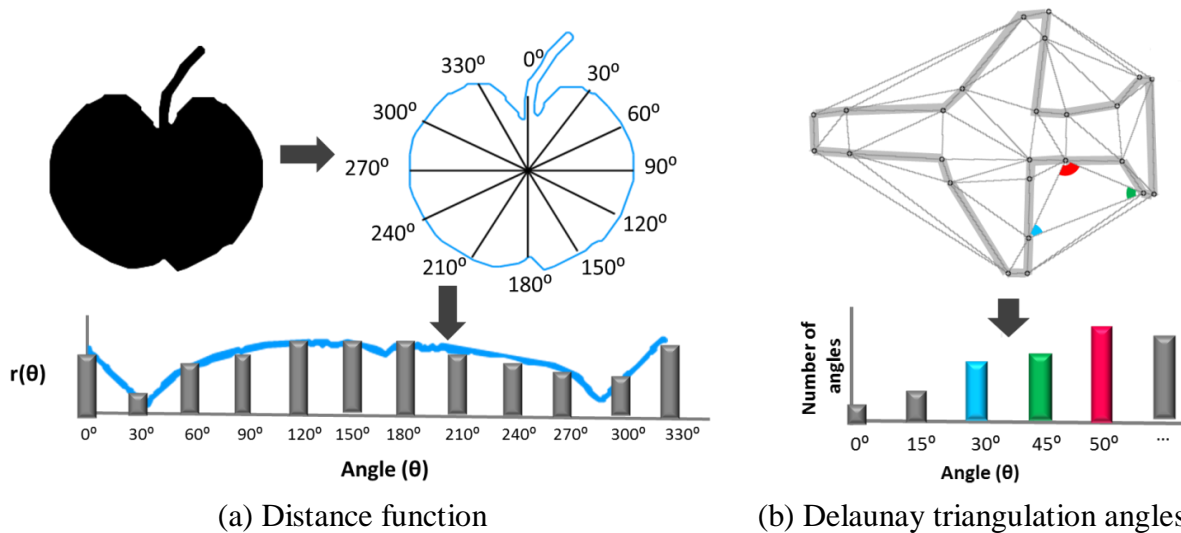


Figure 3.13: Describing shape contours using a distance function histogram a Delaunay triangulation angle histogram (Ferilli et al. 2011; Tao and Grosky 1999).

An improvement over the distance-based shape description approach was proposed by Tao and Grosky (1999). A shape can be described invariantly to the location, scale and orientation of the region by generating a Delaunay triangulation from the corners of the shape. The internal angles of all the Delaunay triangles are quantised into a histogram which then serves as a shape descriptor, as shown in Figure 3.13 (b). The Delaunay triangulation approach provides greater transformation invariance over the distance-based shape descriptor. However, the triangulation is dependent on the corners identified by the corner detector. Introducing new corners or removing existing corners from the Delaunay triangulation can affect the resulting angle histogram. The approach therefore relies heavily on robustness of the corner detector used. The sketching anomalies identified in Section 3.3.2.4 also effect the resulting Delaunay triangulation.

The advantages of boundary-based shape representation approaches are that they are highly descriptive, robust to noise and relatively simple to implement. The main disadvantage is that these methods only consider the boundary of the shape and therefore provide no means of describing the content of the shape's interior. This problem can be addressed by using SIFT-

based or grid-based descriptors. It is also possible to describe the individual edge structures of the feature.

The edges contained in a feature can be compared to the edges contained in another feature using the Chamfer Matching dissimilarity measure (Barrow *et al.* 1977). The Chamfer Matching dissimilarity measure estimates the integral distance or area between the edges of two shapes. The Chamfer distance between shape A and shape B is calculated as the sum of the minimum distances between the vertices of A and B as shown in Figure 3.14. Shotton *et al.* (2008) extended the Chamfer Matching dissimilarity measure to include line segment orientation as well. The advantage of considering the edges of the shape instead of its boundary is that the internal contents of the shape can be described. The approach is not invariant to scaling, translation or rotation. The shapes have to be normalised before they can be compared reliably.

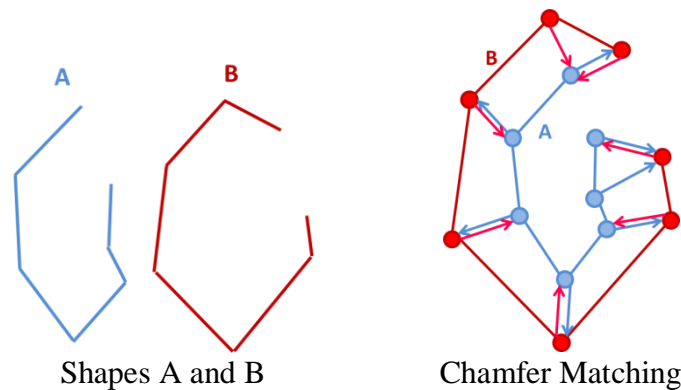


Figure 3.14: Chamfer matching for comparing shapes.

3.3.3.4 Summary of Local Region Descriptors for Sketches

Describing the local features of the user's sketch involves selecting or combining appropriate region descriptors based on their strengths and weaknesses. SIFT-based descriptors provide a robust and transformation invariant method of describing blob-based features. The problem is that the user's sketches do not contain sufficient gradient information for SIFT-based feature description. The grid-based description approach can be combined with the SIFT-based feature description approach in order to address this problem. This can be achieved by describing the sketch using a grid-based approach and then using the gradient information of the grid to detect and describe SIFT-based local features.

An alternative approach to describing user sketches is to describe the shapes of the local regions. This can be achieved by describing the boundary of each feature by measuring the distance function or Delaunay triangle angles. The limitation of this approach is that it does

not consider the internal details of each feature. The internal edge structures can be compared using the Chamfer matching approach in order to address this limitation; however, it would require the features to be normalised reliably first.

Table 3.3: A summary of several region descriptors (Roth and Winter 2008).

Descriptor Classification	Descriptor	Rotational invariance	Dimensionality	Performance	Image type
SIFT-based	SIFT	No	High	Good	Textured
	SIFT-PCA ¹	No	Low	Good	
	SIFT-GLOH ²	No	High	Good	
Grid-based	Grid	No	High	Medium	Both
	Spin images	Yes	Medium	Medium	
Shape-based	Distance function	No	Medium	Good	Outlined
	Delaunay triangulation	Yes	Medium	Good	
	Edge-structures	No	High	Medium	

¹ The dimensionality of the SIFT feature vectors is reduced using Principal Component Analysis (PCA).

² A circular arrangement of patches is used instead of a rectangular arrangement of patches.

A summary of the local region descriptors discussed in this section is provided in Table 3.3. It may be necessary to combine several approaches for describing local features and global features in order to describe user sketches in a robust manner. Chapter 4 will investigate how this can be achieved.

3.3.4 Global Feature Descriptors

Global feature descriptors quantitatively describe the overall appearance of images. An example of a global feature descriptor is the principal components of an image. The locations of local features such as corners, edges and blobs can be used to calculate the principle components.

Principal Component Analysis (PCA) is used to convert a set of observations into a reduced set of values for linearly uncorrelated variables known as *principal components* while retaining as much of the variation of the data set as possible. The 2D case is considered for determining the principal components \bar{u} and \bar{v} of a point set for an image. Let $X = \{x_0, x_1, \dots, x_n\}$ and $Y = \{y_0, y_1, \dots, y_n\}$ be the respective x-coordinates and y-coordinates of the point set. Let $\Sigma \in M_{2 \times 2}$ be the covariance matrix of X and Y then the principal components of the point set is determined from the eigenvalues and eigenvectors

of Σ . Let (λ_0, \bar{v}_0) be the eigenvalue-eigenvector pair with the largest eigenvalue and (λ_1, \bar{v}_1) be the second eigenvalue-eigenvector pair then $\bar{u} = \sqrt{\lambda_0} \bar{v}_0$ and $\bar{v} = \sqrt{\lambda_1} \bar{v}_1$ (Jolliffe 2002).

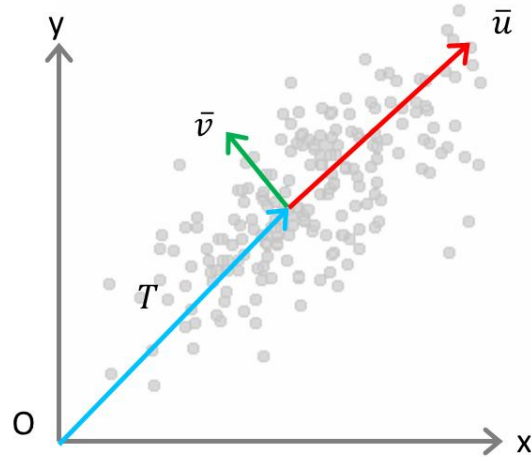


Figure 3.15: Using PCA to estimate the transformation parameters of a point set.

The principal components of the point set can be used to estimate the orientation and scale of the image. The horizontal and vertical scale of the image can be estimated as a constant multiple of the length of the corresponding eigenvectors. The furthest point along each principal axis can also be used to determine the boundaries of the image (Chaudhuri and Samal 2006). The orientation of the image can be estimated from \bar{u} and \bar{v} as shown in Equation (3.1) where ε_a and ε_b are numerical error values and $[a, -b]$ is the top row of the rotation matrix $R = [\bar{u} \ \bar{v}]$. Its location can be estimated as the mean location of the points in the set. These parameters are required for normalising regions so that transformation variant descriptors, such as grid-based descriptors, can be used reliably.

$$\theta = \frac{1}{2}(\cos^{-1}(a) + \sin^{-1}(b)) \quad , \text{ where } R = [\bar{u} \ \bar{v}] = \begin{bmatrix} a & -b \\ b + \varepsilon_b & a + \varepsilon_a \end{bmatrix} \quad (3.1)$$

3.3.5 Discussion

The object recognition phase involves the following four steps (see Figure 3.2):

- Step 1. Detect the features of the query image provided.
- Step 2. Describe the query features quantitatively.
- Step 3. Match the query's features with the features of each candidate image.
- Step 4. Classify the query image using a similarity/dissimilarity measure.

This section discussed several methods from computer vision for steps one and two. In particular, it is important to note that describing user sketches presents two main challenges.

Firstly, there is no texturing information available. Secondly, the user's sketches may contain several anomalies (see Section 3.3.2.4). It is therefore necessary to consider the strengths and weaknesses of each approach when describing user sketches. The next section will discuss how images can be compared for the purposes of performing object recognition. This involves performing steps three and four.

3.4 Comparing Images

The recognition phase of the object recognition process involves comparing the features of a query image with the features of candidate images and determining which object is most likely represented by the query image (see Section 3.2). This section reviews several methods for matching the features of a query image with the features of a candidate image. It also provides an overview of several classification approaches and suggests a classifier suitable for classifying user sketches.

3.4.1 Feature Matching

The feature matching problem involves matching the features of a training image to the features of a query image (Torresani, Kolmogorov and Rother 2008). Object recognition is facilitated by searching for feature correspondences across several training images. This section will provide an overview of two different approaches that have been proposed for finding feature correspondences.

The first approach utilises the appearance of the features and discards spatial and structural information. The features are matched only on their appearance. This approach is often referred to as the *bag of features* approach (Dance *et al.* 2004). Each feature of the query image is matched to the corresponding feature of the candidate image which has the most similar descriptor. The main advantage of the bag of features approach is its simplicity. The method also provides invariance to affine transformations. The main disadvantage of the approach is that it relies heavily on the descriptive capability of the feature descriptor. Feature descriptions are therefore required to be clearly distinguishable from similar features and they should avoid encoding irrelevant variations such as noise. Another disadvantage is that it does not consider the geometric structure of the features of the image.

The second approach utilises structural models. A *structural model* is a mathematical model that measure the correspondence of features while considering their appearances and geometric structure (Zhang *et al.* 2007). Writing the structural model as an optimisation

problem is one way to formulate the feature matching problem (Torresani *et al.* 2008). In this case, the model serves as an objective function $E(\bar{x})$ where \bar{x} is the correlation vector to be optimised for two feature sets P' and P'' . Each entry of \bar{x} corresponds to a pair $c \in P' \times P''$ where $x_c = 1$ if c is a matching configuration, otherwise $x_c = 0$. The problem is formulated to optimise a function $E(\bar{x}) = \sum_{i=0}^{M-1} \lambda_i E_i$ where E_i is a cost function for the i^{th} feature property and λ_i is a weight for scaling the importance of the i^{th} cost function. Torresani *et al.* (2008) matched features based on their appearance $A(\bar{x})$, occlusion $O(\bar{x})$, geometric compatibility $G(\bar{x})$ and spatial proximity $S(\bar{x})$, $M = 4$. Therefore the problem involved finding an \bar{x} that minimises $E(\bar{x})$ as shown in Equation (3.2).

$$E(\bar{x}) = \lambda_A A(\bar{x}) + \lambda_O O(\bar{x}) + \lambda_G G(\bar{x}) + \lambda_S S(\bar{x}) \quad (3.2)$$

Formulating the feature correspondence problem as an optimisation problem has the advantage of using structural and appearance information. The disadvantage is that the optimisation of the objective function is an NP-hard problem and therefore has very high computational costs (Gold and Rangarajan 1996). The approach may be practical if off-the-shelf solvers can be used.

3.4.2 Classification Approaches

The final step of the object recognition phase is the classification step (see Section 3.2). The codebook is populated with training images which are each associated with a class label C_m , $m \in \mathbb{Z}_0^M$. The probabilistic image classification problem involves finding the best classes $L = \{C_m: m \in \mathbb{Z}_0^M\}$ for a given a tuple $\mathbf{x} = (x_0, x_1, \dots, x_n)$ representing a query image. The tuple can describe the image in terms of pixels or lower dimensional representations such as local features (see Section 3.3). The object corresponding to the best class from L can be assumed to be the object most likely sketched by the user. The remaining classes from L can be considered to be alternative best matches.

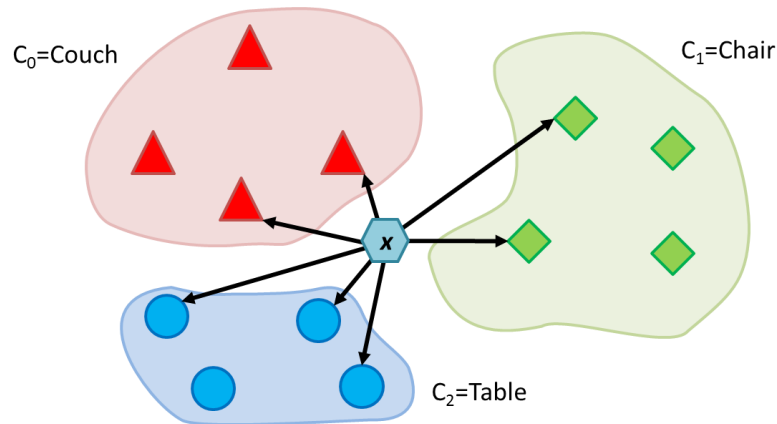


Figure 3.16: The probabilistic image classification problem.

Figure 3.16 provides an example illustrating the classification problem. The codebook is shown to contain three classes, namely C_0 =”couch”, C_1 = ”chair” and C_2 = ”table”. Suppose the tuple x represents a rough sketch of a chair created by the user. The goal of the image classification problem is to determine which class best represents x based on the data contained in the codebook or on classifier training data.

Table 3.4: An overview of several classification approaches (Seetha et al. 2008; Han and Kamber 2006).

Classification Approach	Data Type	Advantages	Disadvantages
Naïve Bayesian	Categorical, Numerical	Simple to implement.	Naïve Bayes assumption usually violated.
Artificial Neural Networks	Numerical	Tolerant to noisy input.	Over fitting is possible when too many attributes are used.
Support Vector Machines	Numerical	Over fitting is unlikely to occur.	Difficult to determine optimal parameters when the data is not linearly separable.
Decision Trees	Categorical, Numerical	Not sensitive to outliers.	Low generalisation performance with high dimensionality.
k -Nearest Neighbours	Numerical	Distance function implicitly defines decision boundaries.	High requirements for storing class instances.

Table 3.4 provides an overview of several approaches that can be taken for classifying images based on their appearance. This section provides an overview each of these classification approaches in order to determine which classifier can be used for recognising the objects sketched by the user. The classifiers include Naïve Bayes classifiers, Artificial Neural Networks, Support Vector Machines, Decision Trees and k -Nearest Neighbours classifiers.

3.4.2.1 Naïve Bayesian

Let H_m be the hypothesis that the tuple \bar{x} belongs to the class C_m . The classification problem involves determining the posterior probability $P(H_m|\bar{x})$ of H_m conditioned on \bar{x} . $P(H_m|\bar{x})$ reflects the probability of \bar{x} belonging to class C_m . Let $P(H_m)$ be the prior probability of H_m , then $P(H_m)$ reflects the probability of class C_m being selected regardless of the tuple values. Let $P(\bar{x}|H_m)$ be the posterior probability of \bar{x} conditioned on H_m , and let $P(\bar{x})$ be the prior probability of tuple \bar{x} occurring.

$$P(H_m|\bar{x}) = \frac{P(\bar{x}|H_m)P(H_m)}{P(\bar{x})} \quad (3.3)$$

Bayes' Theorem relates these probabilities as shown in Equation (3.3) (Han and Kamber 2006). The goal is to find a class C_m that maximises $P(H_m|\bar{x})$. $P(\bar{x})$ is constant for all the classes. If $P(H_m)$ is not known then let $P(H_m) = P(H_{m+1}) \forall k \in \mathbb{Z}_0^{M-1}$. The problem reduces to maximising $P(\bar{x}|H_m)$. The Naïve Bayesian Classifier assumes that the attributes or dimensions of \bar{x} are conditionally independent and that Equation (3.4) holds.

$$P(\bar{x}|H_m) = \prod_{i=0}^n P(x_i|H_m) = P(x_0|H_m) \times P(x_1|H_m) \times \dots \times P(x_n|H_m) \quad (3.4)$$

The values for $P(x_i|H_m)$ can be calculated in two ways. If the values of x_i is categorical then $P(x_i|H_m)$ is calculated as the number of training tuples of class C_m having the value of x_i in the i^{th} dimension. If the values of x_i are continuous then $P(x_i|H_m)$ is calculated from the Gaussian function where the mean and standard deviation is determined from the training tuples of C_m . If all the attributes or dimensions of \mathbf{x} are not conditionally independent, then Bayesian networks can be used to model the relationships between these variables using a directed acyclic graph and Conditional Probability Tables (CPTs). The topology of the network is manually defined. The CPTs can be trained using a gradient descent strategy. McCann and Lowe (2011) showed that the Naïve Bayesian Classifier can be combined with the k -Nearest Neighbours classification approach to effectively classify low resolution images.

3.4.2.2 Artificial Neural Networks

An Artificial Neural Networks (ANN) is another type of network that can be used for image classification. ANNs consist of connected input, hidden and output units that receive and send signals. The connections between the units are weighed. If the sum of the weighed input for a

unit is above a trained threshold then the unit activates and sends its signal along the output connections. The weights can be determined by training the ANN with training tuples and their expected outputs. The various class labels for the image are encoded by the ANN output signal. ANNs can be used to classify low resolution images (Xu and Wei 2012; Mustapha, Lim and Jafri 2010). However, three major problems can be identified for using ANNs to classify high resolution images such as sketches (Engelbrecht 2002):

- If over fitting occurs when an ANN is trained then it loses the ability to generalise to unseen input.
- A large training set of images is required for training a reliable ANN. The set should include validation images that can be used to prevent over fitting. This is problematic because it is difficult to create a sufficiently large set of user sketches.
- It is difficult to model the image classification problem using ANNs. It involves designing an appropriate ANN architecture and representing each image as a single input vector. Images are typically described in terms of features which are in turn represented by description vectors.

3.4.2.3 Support Vector Machines

Support Vector Machine (SVM) classifiers can be used for classifying images or image features. A SVM classifier is an algorithm that constructs a Maximum Marginal Hyperplane (MMH) that separates the tuples of two classes $C_0 = -1$, $C_1 = 1$ from one another. If the dataset is not linearly separable then the tuples can be transformed into a higher dimensional space. If the tuples are linearly separable in the higher dimensional space then a MMH can be determined. The two regions separated by the MMH are given by Equation (3.5).

$$C_m(w_0 + \bar{w} \cdot \bar{x}) \geq 1, \forall m \in \mathbb{Z}_0^M \quad (3.5)$$

The bias w_0 and MMH weight vector \bar{w} are determined using a Lagrangian formulation. The problem can then be solved using the Karush-Kuhn-Tucker (KKT) conditions (Han and Kamber 2006; Mitra, Shankar and Pal 2004). SVM classifiers can be combined for classifying more than two classes.

3.4.2.4 Decision Trees

Decision tree classifiers assume that the classification problem can be broken down into a series of rule-based tests or decisions. The rule system is represented in terms of a tree structure where the vertices represent the attributes being tested and the edges represent possible values or ranges. Decision trees are learned by means of decision tree induction methods such as the Iterative Dichotomiser algorithm (Quinlan 1985). It has been shown, however, that decision trees have low generalisation performance when the dimensionality of \bar{x} is very high (Geurts 2002). Maree *et al.* (2004) showed, however, that decision trees can be learnt and used to classify tuples with lower dimensionalities by considering smaller regions of the image.

3.4.2.5 k -Nearest Neighbours

The classification approaches discussed so far construct classification models before receiving query tuples to classify. The generalisation of image data is therefore modelled during the training phase of the classifier. Training tuples are used to construct the classification model and the generalisation performance of the classifier depends on the training data provided. An alternative approach is to classify query tuples based on their similarity to stored training tuples known as *class instances*. The class instances then represent the classification model (Han and Kamber 2006).

The k -Nearest-Neighbours (k -NN) algorithm classifies a tuple \bar{x} as follows. Each class C_m is associated with a set of class instances $\{\bar{p}_{m_i} : i \in \mathbb{Z}_1^n\}$. The k -NN algorithm finds the k best class instances that minimise a distance function $E(\bar{x}, \bar{p}_{m_i})$. If the Euclidean distance given by Equation (3.6) is used, then it is important to scale the individual dimensions of the search space appropriately so that the Euclidean norm is not biased along the dimensions with large values. Other distance metrics for measuring the dissimilarity between images can also be used, such as the error involved when matching local features (see Section 3.4.1).

$$E_{\text{Euclidean}}(\bar{x}, \bar{p}_{m_i}) = \sqrt{\sum_{j=0}^n (x_j - p_{m_{ij}})^2} \quad (3.6)$$

3.4.2.6 Discussion

This section discussed several classification approaches that can be used for classifying the sketches created by the user. This included methods that construct classification models

during the training stage of object recognition, such as Bayesian Classification, Artificial Neural Networks, Support Vector Machines and Decision Trees.

A sufficient training set consisting of user sketches is required for each class in order to train these classification models with acceptable generalisation performance. This is problematic because it is difficult to produce a training dataset when the different users can each sketch objects in an arbitrary fashion. Another problem is that these methods represent images in terms of individual tuples or vectors. However, images are typically described using multiple features that are each represented with a feature description vector (see Section 3.3). The k -Nearest-Neighbour classification approach addresses this problem by classifying images based on a similarity measure. This makes it possible to use image descriptors in order to classify images. The images do not have to be represented with vectors of high dimensionality. The k -Nearest-Neighbour classification approach therefore provides a feasible method for classifying user sketches from limited training data.

3.5 Rigid Body Pose Estimation

A rigid body is a non-deformable object that can only be manipulated using translation, rotation or scaling transformations (Fitzpatrick and West 2001). Estimating the pose of a rigid body involves approximating the transformation T needed to map the body's model from its local coordinate system to the coordinate system represented in the image (see Figure 3.17).

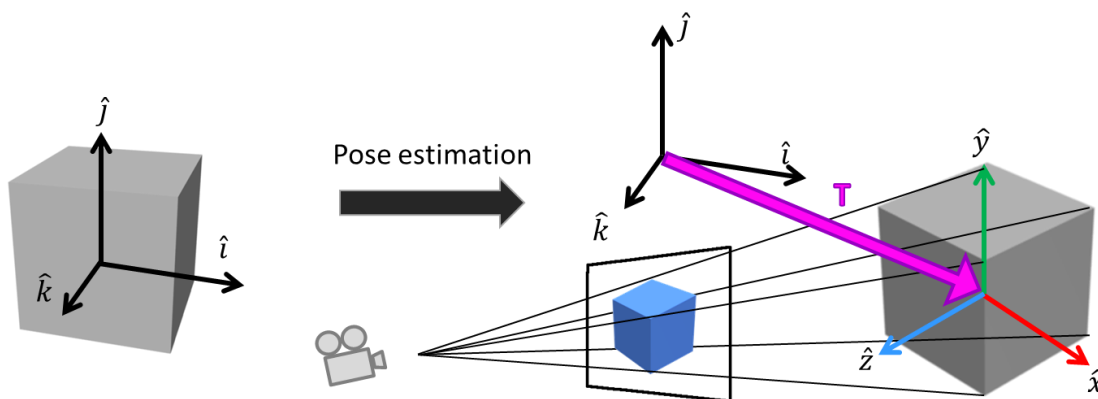


Figure 3.17: The pose estimation problem (Rosenhahn, Perwass and Sommer 2004).

The scale of the rigid body is usually assumed to be constant in order to facilitate monocular depth perception (Lee and Funkhouser 2008). This section reviews three approaches for estimating the pose of a rigid body from an image:

1. Object recognition for recognising the pose of the rigid body.

2. Analytical pose estimation methods.
3. Numerical pose estimation methods.

3.5.1 Recognising Poses

One of the simplest pose estimation approaches is to recognise the model's approximate pose from a codebook instead of determining its exact pose. The idea is to index each object into a codebook from several views. An object recognition algorithm is used to determine which object is present and what its approximate pose is. Shape descriptors are commonly used to describe the boundaries of each object viewed from different angles (see Section 3.3.3.3) (Funkhouser *et al.* 2003; Hou and Ramani 2006; Loffler 2000). This approach has also been proposed as a pre-processing step for high-precision pose estimation algorithms that require an initial pose estimate (Shin and Igarashi 2007; Lee and Funkhouser 2008). Figure 3.18 shows an example of how a model can be sampled to generate a boundary-based description for each view. A representative number of samples are taken to index the anticipated orientations of the object. Shin and Igarashi (2007) used sixteen reference views and a centroid Fourier shape descriptor to build their codebook. The shape descriptor is used to find the closest codebook entry from which the estimated pose can be determined.

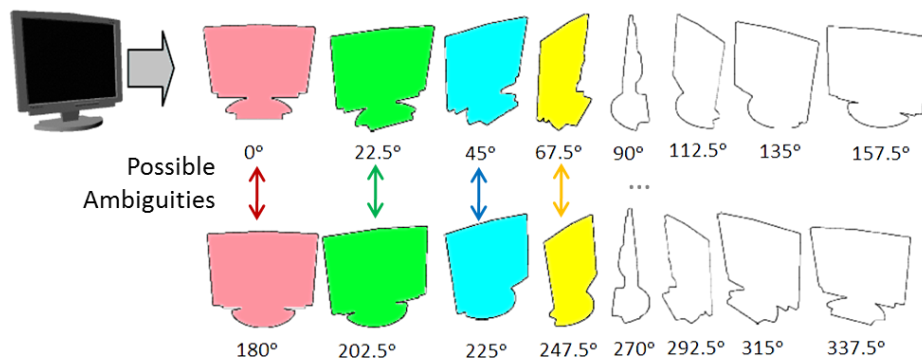


Figure 3.18: Indexing a model from sixteen reference views using a shape descriptor (Adapted from Shin and Igarashi 2007).

Although the pose estimation approach is simple to implement, it relies heavily on the quality of the codebook and the discriminative capability of the feature descriptor. Feature descriptors that do not include internal detail, such as boundary-based descriptors, cannot be used to distinguish views of objects that have similar boundaries. Front/back ambiguity and symmetry is therefore a problem. Figure 3.18 shows an example of where front/back (0° and 180°) and symmetry ambiguity (67.5° and 247.5°) can be problematic. Another limitation of the method is that it can only recognise poses that have been indexed in the codebook.

3.5.2 Analytical Pose Estimation

A more direct approach to the pose estimation problem is to determine the required transformation parameters analytically. The problem is usually formulated as a Perspective-n-Point (PnP) problem (Lepetit and Fua 2005). The PnP problem involves determining the position and orientation of the camera from a set of n 3D points with their corresponding 2D perspective projections. Early research on the PnP problem produced several analytical solutions for small values of n (Haralick, Lee and Ottenburg 1991; Navab and Faugeras 1993; Abidi and Chandra 1995; Yuyan, Iyengar and Jain 1994).

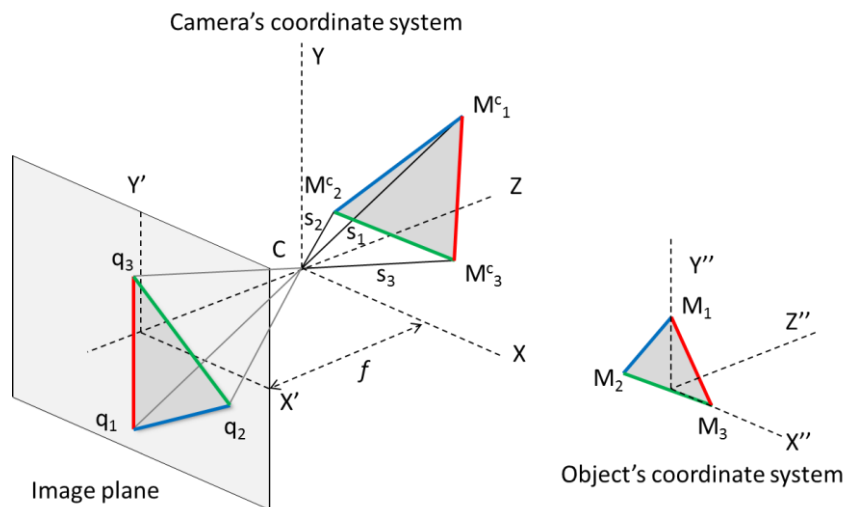


Figure 3.19: A diagram showing the variables involved in solving a P3P problem using a pinhole camera model (Abidi and Chandra 1995).

In the 1990s, the P3P problem was approached using a pinhole camera model (Abidi and Chandra 1995). Figure 3.19 illustrates the formulation of the P3P problem using this camera model. For the purposes of pose estimation, assume that a 3D model of an object is available and that three correspondences between the 3D points \mathbf{M}_i and image points \mathbf{q}_i are known. Also assume that the camera is calibrated, i.e. the focal length f of the camera is known. The P3P problem involves finding the points \mathbf{M}_i^c within the coordinate system of the camera located at \mathbf{C} . Each point \mathbf{M}_i^c is found by estimating the distances $s_i = \|\mathbf{M}_i - \mathbf{C}\|$ between \mathbf{M}_i and \mathbf{C} . The distances s_i are calculated by finding the roots of a fourth degree polynomial which is derived from constraints found in the triangles $\Delta \mathbf{C}\mathbf{M}_i\mathbf{M}_j, i \neq j$. Once the points \mathbf{M}_i^c are known, the transformation that moves the points from \mathbf{M}_i to \mathbf{M}_i^c is taken as the solution to the P3P problem. The transformation can be found using 3D registration algorithms such as the Iterative Closest Point algorithm (see Section 3.5.3.2) (Jost 2003). The transformation is not unique because multiple polynomial roots exist. In fact, the P3P problem either has an

infinite number of solutions or at most four solutions (Gao, Hou and Tang 2003). An additional correspondence is often required to resolve the ambiguity.

More recently, several analytical PnP solutions were proposed that are based on algebraic methods (Moreno-Noguer, Lepetit and Fua 2007; Bujnak and Kukelova 2008; Triggs and Quan 2000; Wu and Hu 2006; Ansar and Daniilidis 2003; Fiore 2001). The most popular approach is to estimate the unknown point distances in the camera's coordinate system by formulating the geometric problem as a system of polynomials. The system of polynomials is then written as an equivalent linear system which is solved using the hidden variable method, the Gröbner basis solver or the singular value decomposition method.

The main advantage of analytical methods is that unique solutions can be found quickly, and in some cases, in $O(n)$ time (Moreno-Noguer *et al.* 2007). The main disadvantage of using analytical methods is that they are specialised for small cases of n . PnP solutions with a small n are highly sensitive to projection errors (Dementhon and Davis 1995). This is problematic because projection errors occur when local features are detected at slightly incorrect locations or when the correspondences between local features and their 3D points are incorrectly determined. These errors are especially prominent when dealing with rough user sketches. Numerical PnP solutions are often preferred over non-iterative solutions when stability is more important than efficiency.

3.5.3 Numerical Pose Estimation

Numerical pose estimation methods approach the PnP problem by estimating the pose and improving the estimated pose iteratively (Didier, Ababsa and Mallem 2008). This can be achieved by modelling the pose estimation problem in terms of a set of geometric constraints and estimating the rotation and translation components of the pose using Least-squares model fitting techniques (Horn, Hilden and Negahdaripour 1988). Least-squares model fitting techniques are typically very sensitive to outliers; however, improved model fitting techniques have been proposed in order to produce more stable iterative pose estimation algorithms (Hanson and Kumar 1994).

Numerical pose estimation methods often relax the perspective projection model. The Orthogonal Iteration (OI) algorithm assumes the pinhole camera model (see Section 3.5.2) in order to iteratively minimise an error metric which is based on the alignment of the projected 2D points and the 3D points of the posed model (Lu, Hager and Mjølness 2000). The POSIT algorithm assumes the scaled orthographic projection model instead of the perspective

projection model (Dementhon and Davis 1995). The simplified camera model is used to estimate the pose of the object.

This section provides a detailed discussion on the POSIT algorithm as well as a discussion on the Iterative Closest Point (ICP) registration algorithm. The advantages and disadvantages of using numerical pose estimation approaches for estimating the pose of rigid bodies contained in user sketches is also discussed.

3.5.3.1 POSIT

The Pose from Orthography and Scaling with ITERations (POSIT) algorithm consists of two parts. The first part is the Pose from Orthography and Scaling (POS) algorithm. The POS algorithm finds the rotation matrix and translation vector of the pose by assuming the Scaled Orthographic Projection (SOP) camera model and solving a linear system of equations. The second part (POS with Iterations) iteratively refines this pose by computing better scaled orthographic projections of the model points and applying the POS algorithm again using the improved projections (Dementhon and Davis 1995).

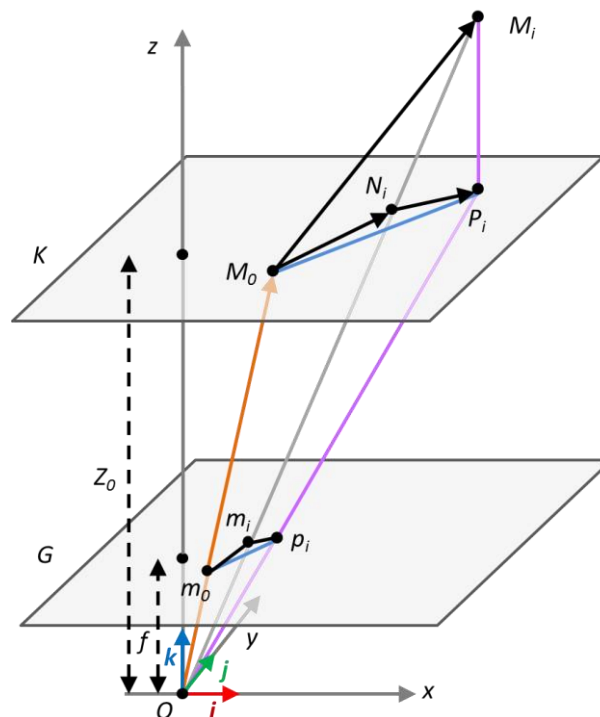


Figure 3.20: The perspective projection and scaled orthographic projection of object points to image points (Dementhon and Davis 1995).

Figure 3.20 illustrates the pinhole camera model approximated by the POSIT algorithm. The image plane G is a distance f (focal length) away from the origin O . The unit vectors of the camera's coordinate system are \hat{i} , \hat{j} and \hat{k} respectively. The object points are M_i known in the

object's coordinate system for all $i \in \mathbb{Z}_0^n$. \mathbf{M}_0 is called the *reference point* of the object. The image points $\bar{\mathbf{m}}_i = (x_i, y_i)$ of the object points are also known for each $i \in \mathbb{Z}_0^n$. The points $\bar{\mathbf{p}}_i = (x'_i, y'_i)$ are the scaled orthographic projections of \mathbf{M}_i . The coordinates (X_i, Y_i, Z_i) of \mathbf{M}_i in terms of the camera's coordinate system are not known because the object's pose is unknown. The POSIT algorithm estimates the pose of the object by calculating the rotation matrix and translation vector of the object without calculating (X_i, Y_i, Z_i) explicitly.

Scaled Orthographic Projection (SOP) is used to approximate the perspective camera model as follows. The approximation assumes that the depth Z_i of the points $\bar{\mathbf{M}}_i$ are close enough to each other for the depth Z_0 of the reference point $\bar{\mathbf{M}}_0$ to be used to project the points $\bar{\mathbf{M}}_i$ to $\bar{\mathbf{p}}_i$ as shown in Equation (3.7). $s = \frac{f}{Z_0}$ is called the *scaling factor*.

$$x'_i = fX_i/Z_0, \quad y'_i = fY_i/Z_0, \quad (3.7)$$

The geometric interpretation of SOP is illustrated in Figure 3.20. Each point $\bar{\mathbf{M}}_i$ is orthographically projected in the plane K located at the reference point $\bar{\mathbf{M}}_0$ in order to obtain $\bar{\mathbf{P}}_i$. Each point $\bar{\mathbf{P}}_i$ is then projected onto the image plane G at $\bar{\mathbf{p}}_i$ using perspective projection. $\bar{\mathbf{m}}_i$ is the "true" perspective projection of $\bar{\mathbf{M}}_i$ and $\bar{\mathbf{p}}_i$ is the SOP approximate. Dementhon and Davis (1995) proved that $\bar{\mathbf{M}}_i$ and $\bar{\mathbf{p}}_i = (x'_i, y'_i)$ are related as shown in Equation (3.8), Equation (3.9) and Equation (3.10) where $\bar{\mathbf{I}} = \frac{f}{Z_0} \hat{\mathbf{i}}$ and $\bar{\mathbf{J}} = \frac{f}{Z_0} \hat{\mathbf{j}}$.

$$\overline{\mathbf{M}_0 \mathbf{M}_i} \cdot \bar{\mathbf{I}} = x_i(1 + \varepsilon_i) - x_0 = x'_i - x_0 = \xi_i \quad (3.8)$$

$$\overline{\mathbf{M}_0 \mathbf{M}_i} \cdot \bar{\mathbf{J}} = y_i(1 + \varepsilon_i) - y_0 = y'_i - y_0 = \eta_i \quad (3.9)$$

$$\varepsilon_i = \frac{1}{Z_0} \overline{\mathbf{M}_0 \mathbf{M}_i} \cdot \hat{\mathbf{k}}, \text{ where } \overline{\mathbf{M}_0 \mathbf{M}_i} = \bar{\mathbf{M}}_i - \bar{\mathbf{M}}_0 \quad (3.10)$$

If values are given to ε_i then Equation (3.8) and Equation (3.9) provide a system of linear equations that can be used for solving $\bar{\mathbf{I}}$ and $\bar{\mathbf{J}}$. The linear system of equations is given in Equation (3.11) and Equation (3.12) where the rows of \mathbf{A} are the object points relative to the object's coordinate system.

$$\mathbf{A}\bar{\mathbf{I}} = \bar{\mathbf{x}}', \text{ where } \bar{\mathbf{x}}' = (\xi_0, \xi_1, \dots, \xi_n) \quad (3.11)$$

$$\mathbf{A}\bar{\mathbf{J}} = \bar{\mathbf{y}}', \text{ where } \bar{\mathbf{y}}' = (\eta_0, \eta_1, \dots, \eta_n) \quad (3.12)$$

The linear system of equations is solved for $\bar{\mathbf{I}}$ and $\bar{\mathbf{J}}$ by finding a least squares solution using the pseudoinverse matrix of \mathbf{A} . POSIT requires the object points to be non-planar in order to solve this system of equations.

ε_i directly influences the accuracy of the estimated pose. ε_i is initially set to zero and then improved as the algorithm iterates. $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ are calculated by normalising $\bar{\mathbf{I}}$ and $\bar{\mathbf{J}}$. The cross product of $\hat{\mathbf{i}}$ and $\hat{\mathbf{j}}$ yields $\hat{\mathbf{k}}$. The next value of Z_0 is given by $Z_0 = 2f / (\|\bar{\mathbf{I}}\| + \|\bar{\mathbf{J}}\|)$ where $\|\bar{\mathbf{u}}\|$ is the Euclidian norm of $\bar{\mathbf{u}}$. An improved value for ε_i is calculated using Equation (3.10). $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ form the rows of the object's rotation matrix \mathbf{R} . The translation vector is calculated as $\overline{\mathbf{OM}_0} = \frac{1}{s} \overline{\mathbf{Om}_0}$.

3.5.3.2 Iterative Closest Point (ICP) Registration

The Iterative Closest Point (ICP) registration algorithm determines the rotation and translation transformation required to align a set of points $V = \{\bar{\mathbf{v}}_i; i \in \mathbb{Z}_0^n\}$ as close as possible to another set of points $U = \{\bar{\mathbf{u}}_i; i \in \mathbb{Z}_0^n\}$ as shown Figure 3.21. The ICP registration algorithm can be used to estimate the pose of an object if the posed object points are known in terms of the camera (see Section 3.5.2).

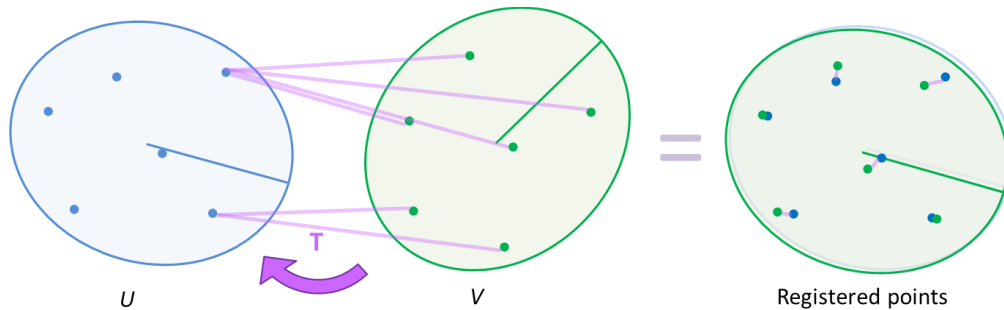


Figure 3.21: An illustration of the 2D case of the point registration problem.

The ICP registration algorithm iterates four steps in order to estimate the transformation required to align the points of V to the points of U as closely as possible. Figure 3.22 provides the pseudocode for the ICP registration algorithm. It iterates until no significant improvement β in $E(\mathbf{T})$ can be observed or the maximum number of iterations have been executed. The translation vector can be estimated at step 3 as the difference of the mean of U and V respectively: $\bar{\mathbf{t}} = \frac{1}{n} \sum_{i=0}^n \bar{\mathbf{u}}_i - \frac{1}{n} \sum_{i=0}^n \bar{\mathbf{v}}_i$.

DO

Step 1. Compute the Nearest Neighbour (NN) point such that $\|\bar{\mathbf{v}}_j - \bar{\mathbf{u}}_i\|$ is a minimum for every correspondence $C_k = (\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_j) = (\bar{\mathbf{x}}_k, \bar{\mathbf{p}}_k)$.

Step 2. Weigh each correlation C_k with an weight w_k , e.g. $w_k \propto 1/\|\bar{\mathbf{x}}_k - \bar{\mathbf{p}}_k\|$

Step 3. Compute the best transformation $\mathbf{T} = (\mathbf{R}, \bar{\mathbf{t}})$ where \mathbf{R} is the rotation matrix and $\bar{\mathbf{t}}$ is the translation vector of \mathbf{T} that minimises:

$$E_t = E(\mathbf{T}) = \frac{1}{W} \sum_{k=0}^n w_k \|\mathbf{R}\bar{\mathbf{p}}_k + \bar{\mathbf{t}} - \bar{\mathbf{x}}_k\|, \text{ where } W = \sum_{k=0}^n w_k$$

Step 4. Apply transformation \mathbf{T} to the points in V .

WHILE $E_{t-1} - E_t > \beta$ **REPEAT**

Figure 3.22: Pseudocode for the ICP registration algorithm (Jost 2003).

Singular Value Decomposition (SVD) of the correlation matrix of U and V can be used to estimate the rotation matrix for \mathbf{T} (Golub and Loan 1996). The NN search in step 1 can be performed efficiently at each iteration using a kd-tree for spatial indexing in $O(n \log(n))$ time instead of using a $O(n^2)$ exhaustive search (Zhang 1994).

3.5.3.3 Discussion

The main advantage that numerical pose estimation algorithms provide over analytical ones is that they are suitable for PnP problems with a large n value. This allows numerical pose estimation approaches to be significantly more accurate and stable than analytical PnP solutions (Dementhon and Davis 1995). Numerical PnP solutions are also less susceptible to the jitter caused by noise and feature errors. User sketches are often inaccurate compared to the images used during the training phase. The sketches are likely to contain the anomalies identified in Section 3.3.2.4. It is therefore possible that feature detection and matching errors occur during the recognition phase. Numerical pose estimation approaches are not affected as severely by this issue as analytical pose estimation approaches if a large enough n is used.

A common disadvantage of numerical approaches is that they are usually less efficient than analytical approaches (Didier *et al.* 2008). Initialisation of the first estimated pose may also affect the convergence of the algorithm. If the first estimate is incorrectly chosen then the algorithm may require many iterations to converge, or it may fail to converge to a globally optimum pose. However, it has been shown that iterative pose estimation approaches converge to a globally optimum estimate pose in most cases (Lu *et al.* 2000).

3.6 Articulated Body Pose Estimation

The process of sketching a character involves planning the posture of the character, defining its proportions and elaborating on its appearance as illustrated in Figure 3.23. A blue pencil is often used to sketch the planning aspects of the character and a black pencil is used for the final aspects of the character. A storyboard artist usually sketches a stick figure first for planning the pose of the character. Next, the artist fleshes out the skeleton by sketching the volume of the character's various body parts and indicating elements with structural importance such as joints, hips and lines of symmetry. The artist then uses the resulting "balloon person" as a foundation for sketching the final appearance of the character.

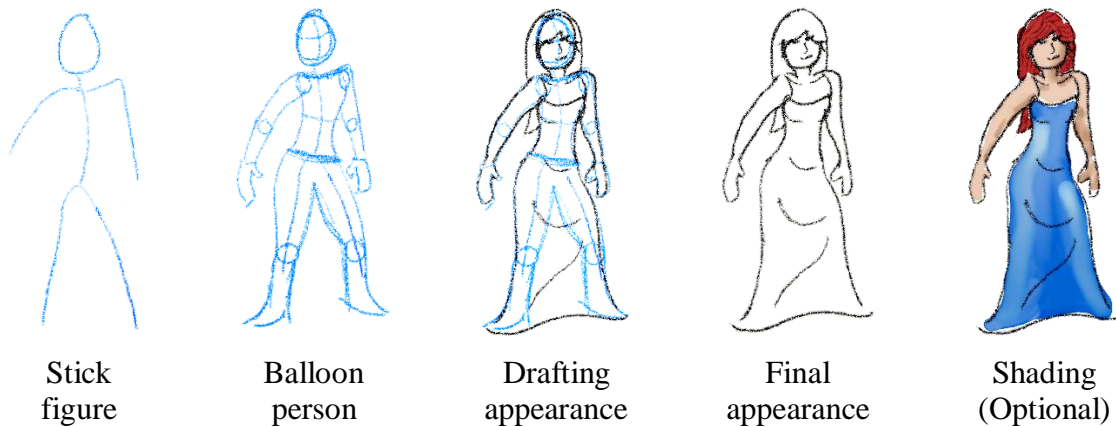


Figure 3.23: The process of sketching a character (Glebas 2008).

At this stage the artist can elaborate on the physical appearance of the character. This may include details such as gender, clothing, hairstyle, facial expression, etc. An optional step is to shade the character in order to illustrate its colouring, texturing and lighting.

Each character in the user's sketch is represented by an articulated body. The articulated body defines the appearance of the character, its location and orientation on the set, and the pose of its individual body parts. Articulated body pose estimation methods are required in order to estimate these parameters. There are essentially two approaches that can be utilised in order to achieve this goal. The first approach is to estimate the character's pose from the character's appearance using methods from computer vision. The second approach is to apply sketch-based posing methods on the stick figure sketched during the planning stage (see Figure 3.23).

3.6.1 Vision-based Posing Methods

Many methods have been proposed in computer vision for estimating the pose of an articulated body from an image. They can be classified into three major groups (Moeslund, Hilton and Kruger 2006):

1. Model-free pose estimation
2. Indirect-model pose estimation
3. Direct-model pose estimation

Model-free pose estimation techniques can be separated into two approaches. The first approach, namely the probabilistic part assembly approach, is a bottom-up 2D pose estimation technique. The method detects the likely locations of the various body parts of the character. Sidenbladh *et al.* (2000) proposed a body part detector for tracking humanoid body parts. Figure 3.24 (a) shows an example of a template used to define an abstract humanoid body. The template is also called a *body plan*. The parts of the body plan are assembled in order to obtain a configuration that best matches the pose observed in the image (see Figure 3.24 (b, c)). The resulting assembly can then be used to estimate the 3D pose of the character.

The second model-free pose estimation technique estimates the character's pose based on training data. The shape of each pose from the training set is encoded and compared to the query image using contour descriptors (see Section 3.3.3.3) (Brand 1999; Howe 2004). Example-based model-free methods are limited to the poses provided in the training dataset.

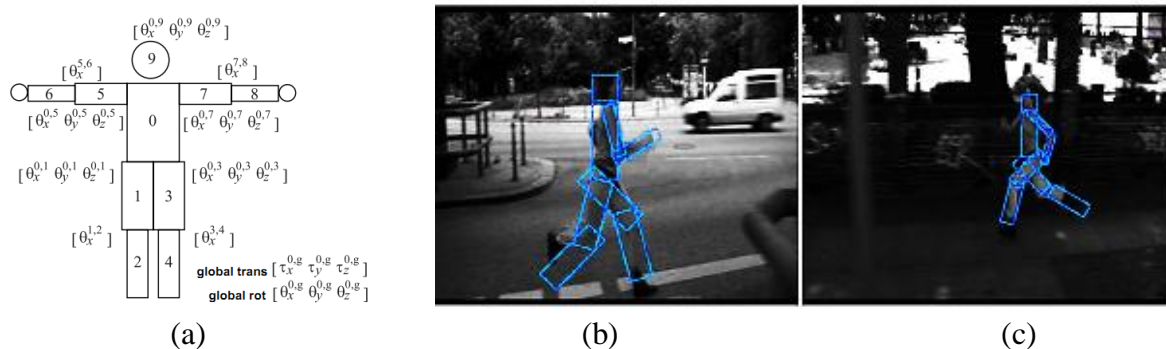


Figure 3.24: Vision-based pose estimation using (a) a body plan (Sidenbladh *et al.* 2000) and (b,c) part assembly (Ramanan, Forsyth and Zisserman 2005).

Indirect-model pose estimation techniques are used to construct the shape and pose of a character from a 3D visual hull without an a priori model of the subject (Cheung, Baker and Kanade 2003; Miki *et al.* 2002). Indirect-model pose estimation techniques require several images from multiple viewpoints in order to construct the visual hull. They are therefore not suitable for monocular character pose estimation from single view storyboard sketches.

Direct-model pose estimation techniques make use of an explicit model in order to synthesise the 3D pose observed in the image. The model provides information about the kinematics, shape and appearance of the character. Monocular direct-model pose estimation has proven to be a very difficult problem to solve; however, several direct-model approaches have been proposed. These approaches combine the probabilistic approach with higher level knowledge about human kinetics (Navaratnam *et al.* 2005; Wachter and Nagel 1997). The poses are limited to those that do not violate kinematic constraints (Bregler, Malik and Pullen 2004; Wachter and Nagel 1997). Inverse Kinematics (IK) approaches that are based on models learned from human motion have also been proposed (Grochow *et al.* 2004; Ong and Hilton 2006). IK involves calculating the orientations of the body parts in order to allow characters to reach out to specific points. Learnt IK methods are limited to specific motion models and lack general posing capabilities (Moeslund *et al.* 2006).

Vision-based pose estimation methods are able to process more complex and descriptive elements of an image in order to estimate character poses in the user's sketch. The methods are able to take into account the shape, texturing and colouring of each part of the character sketched by the user. However, it was identified in Chapter 2 that the characters sketched by the user should be as minimalistic as possible so that they can be sketched as quickly as possible. If sketching a character's stick figure is all that is required, then it would be preferable to sketching a detailed character. This presents a problem for vision-based pose estimation methods because they are designed to operate on images that explicitly show the details of each articulated body. The next section discusses methods that can be used to estimate the character's pose from the user sketched stick figure.

3.6.2 Sketch-based Posing Methods

Several methods have been proposed for articulated body pose estimation from sketches. The pose of the articulated body can be estimated indirectly from the sketch or it can be estimated directly.

Indirect sketch-based posing methods use the sketch for guiding the pose estimation process. The pose is not directly extracted from the user's stick figure. A popular indirect approach is to compare the sketch with known poses from a motion-tracking database and find the closest pose (Lin 2006). The method has the disadvantage of only being able to recognise poses which are known in the database. Inverse Kinematics (IK) can also be used to determine the pose (Vaidya, Shaji and Chandran 2006; Chaudhuri *et al.* 2004). IK algorithms are typically

provided with joint angle constraints in order to produce acceptably realistic results. The disadvantage of IK methods is that they provide limited control of how bones should reach out to their targets.

Direct sketch-based posing methods determine the pose directly from the user's sketch. This requires each bone of the articulated body to be associated with a corresponding line segment in the sketch. The simplest way of doing this is to require the artist to manually label the sketch (Lin 2006; Mao and Qin 2005). The advantage of requiring the artist to manually label the sketch is that any level of fidelity can be used. Unfortunately, the task of manually labelling line segments to their corresponding bones is very tedious and time consuming.

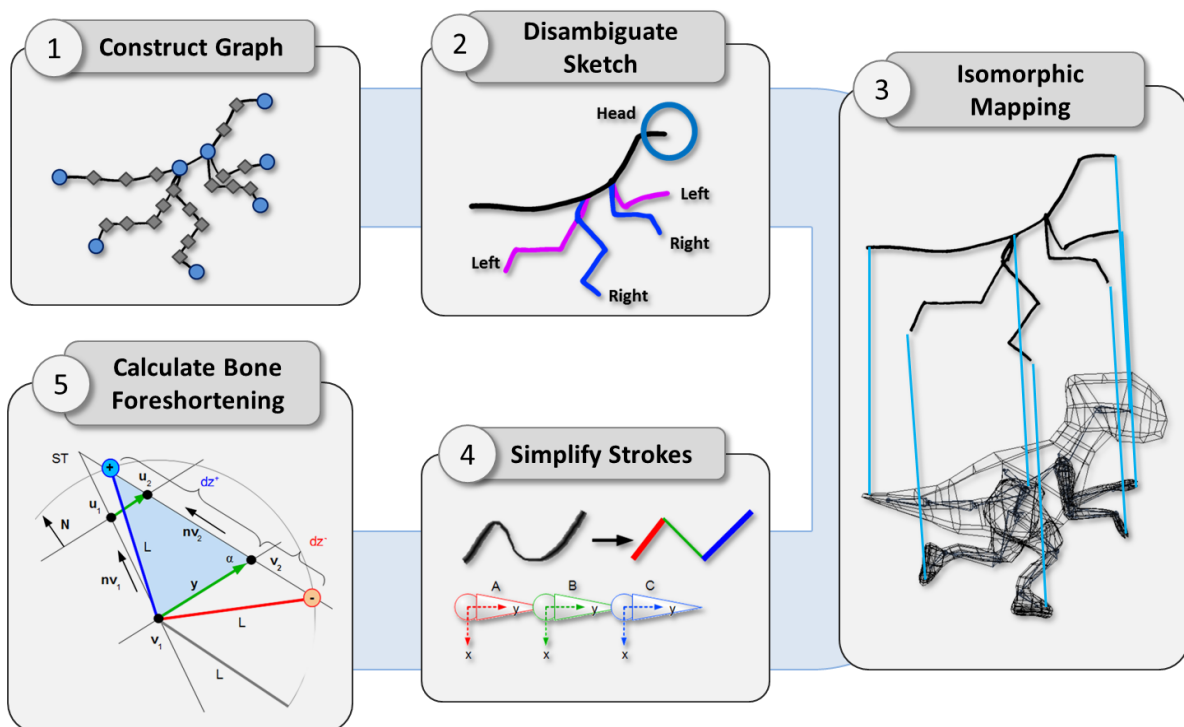


Figure 3.25: A direct sketch-based rigid body posing method (Matthews and Vogts 2011).

Automated labelling methods have been proposed such as using image degrading techniques to extract the stick figure (Davis *et al.* 2003). Previous research⁴ conducted by the author of the dissertation proposed a method for extracting and posing articulated bodies as they are sketched. The direct sketch-based rigid body posing approach is shown in Figure 3.25. The method constructs a directed tree that represents the stick figure as the user sketches (Step 1).

⁴This research on sketch-based articulated figure animation was published in the following paper: Matthews, T. and Vogts, D. (2011): A sketch-based articulated figure animation tool. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT) The Pavilion Conference Centre, V&A Waterfront, Cape Town, South Africa.*

Colour coding is used to disambiguate the topology of the stick figure (Step 2). An isomorphic mapping is then determined between the graph of the stick figure and the armature of the model (Step 3). The strokes of the stick figure are then simplified so that there is a one-to-one mapping between the bones of the armature and the line segments of the stick figure (Step 4). The pose of the articulated body can then be determined by calculating the foreshortening of each bone recursively (Step 5) (Matthews and Vogts 2011).

The main problem with automatic skeleton association methods is that there may be many associations between the user's sketch and the articulated body. Depth ambiguity is also a problem that adds to the ambiguous nature of the articulated body pose estimation problem. In most cases, there are many poses that satisfy the sketch to some extent. For this reason, the solution space is culled using known information about the model and the sketch. Invalid solutions are usually removed using joint angle constraints (Davis *et al.* 2003; Lin 2006; Matthews and Vogts 2011). The remaining solution set can be further reduced using information from the sketch. Mao *et al.* (2005) proposed using multiple strokes to indicate joints and bones that are closer to the viewer. However, this approach may clutter the sketch as more strokes are added. Pressure information from the sketching device can be used to solve depth-related ambiguity.

3.7 Conclusions

This chapter answered the second research question identified in Chapter 1, namely *what methods are available for extracting information from sketches?* This research question was answered by answering the sub-questions provided in Section 3.1.

Q_{2.1} was answered by reviewing several methods for detecting and describing image features. The first part of the sub-question was answered by discussing corner detectors, region detectors and edge detectors. It was established that there are two main challenges when describing user sketches quantitatively. Firstly, there is no texturing information available because the user's sketches are not shaded and they consist primarily of minimalistic strokes. The second challenge is that the sketches may contain several anomalies (see Section 3.3.2.4). Blob-based region detectors, such as the DoG detector, were identified as being the most suitable approach for detecting local features from user sketches.

The second part of sub-question Q_{2.1} was answered by reviewing several local feature descriptors and two global feature descriptors. It was proposed that SIFT-based descriptors be combined with Grid-based descriptors in order to describe user sketches reliably.

Q_{2.2} was answered by reviewing feature matching methods and approaches for classifying user sketches (a class for each prop and symbol). It was determined from literature that the bag of features approach is suitable for finding feature correspondences. The classification problem was addressed by reviewing Bayesian classifiers, ANNs, SVMs, decision trees and k -NN classifiers. It was determined that the k -NN classification approach is the most suitable for classifying user sketches. This is because the k -NN classifier makes it possible to classify images based on their local features. The images do not have to be represented with single, high-dimensional vectors. The k -Nearest-Neighbour classification approach therefore provides a feasible method for classifying user sketches from limited training data.

Q_{2.3} was answered by reviewing three approaches for estimating the pose of a rigid body (such as props), namely pose recognition, analytical pose estimation and numerical pose estimation. It was determined that the pose recognition approach alone is not suitable for estimating the pose of a rigid body because of front/back ambiguity and symmetry. It was also determined that numerical pose estimation approaches provide more stability than analytical pose estimation approaches. The sketches are likely to contain the anomalies identified in Section 3.3.2.4 and it is possible that feature detection and matching errors will occur during the recognition phase. Numerical rigid body pose estimation approaches are not affected as severely by these errors as analytical approaches if a large enough n is used.

Q_{2.4} was answered by reviewing two approaches for estimating the pose of characters. The characters are represented by articulated bodies. An overview of several vision-based posing methods was provided and it was determined from the requirements identified in Chapter 1 that vision-based articulated body posing methods are not suitable for interpreting the characters sketched in the storyboards, because the characters will be sketched using minimalistic stick figures. It was proposed that a direct sketch-based articulated body pose estimation approach is used in order to provide the user with a quick and easy method for sketching characters.

The next chapter investigates how a sketch-based pre-visualisation authoring tool using a storyboarding approach can be designed and implemented. The design includes selected methods reviewed in this chapter in order to automatically interpret user sketches. Chapter 4 will answer the third research question, namely *how can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?*

Chapter 4:

Design and Implementation

4.1 Introduction

Chapter 2 proposed several requirements for sketch-based pre-visualisation authoring using a storyboarding approach, and Chapter 3 provided a review of several approaches that can be used for interpreting the storyboard sketched by the user. This chapter addresses the key requirements which were identified in Chapter 2 (see Table 2.7) by discussing the design and implementation of a proof of concept prototype sketch-based graphical user interface for authoring pre-visualisations using a storyboarding approach.

The third research question identified in Chapter 1 is addressed in this chapter, namely Q_3 : *how can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?* The following sub-questions are answered:

- Q_{3.1}: How can sketch-based pre-visualisation authoring using a storyboarding approach be integrated into existing pre-visualisation authoring frameworks?
- Q_{3.2}: How can the data required for the proposed approach be structured and represented?
- Q_{3.3}: How can a Graphical User Interface be designed to support the proposed approach?
- Q_{3.4}: How can the content sketched by the user be interpreted automatically to determine the 3D context for each shot?
- Q_{3.5}: How can the proposed design be implemented?

The chapter begins by discussing the general framework of pre-visualisation authoring environments and then proposes a new framework for the user interface layer of the general framework (see Section 4.3). The chapter continues by discussing how the data required for

authoring pre-visualisations using the proposed approach can be structured and represented. A detailed discussion on the design of the user interface and the required algorithms is provided. In particular, the algorithm design section investigates several computer vision methods for representing and comparing user sketches (see Section 4.6). It also discusses how objects are recognised and posed, and how the camera is estimated for a storyboard sketch. The chapter concludes by discussing how the prototype was implemented (see Section 4.7).

4.2 A Generalised Framework for Authoring Pre-visualisations

A framework is defined in this research as a large-scale re-usable design that shows how interacting components of a software system are composed together (Johnson 1997). Several frameworks have been proposed for creating cinematic content using game-engines. Notable frameworks include the SAIBA framework (see Section 2.6.1), the Zuzen framework and the Zocalo (Mimesis) framework (Young *et al.* 2004; Kopp *et al.* 2006). This section provides a brief generalisation of these frameworks.

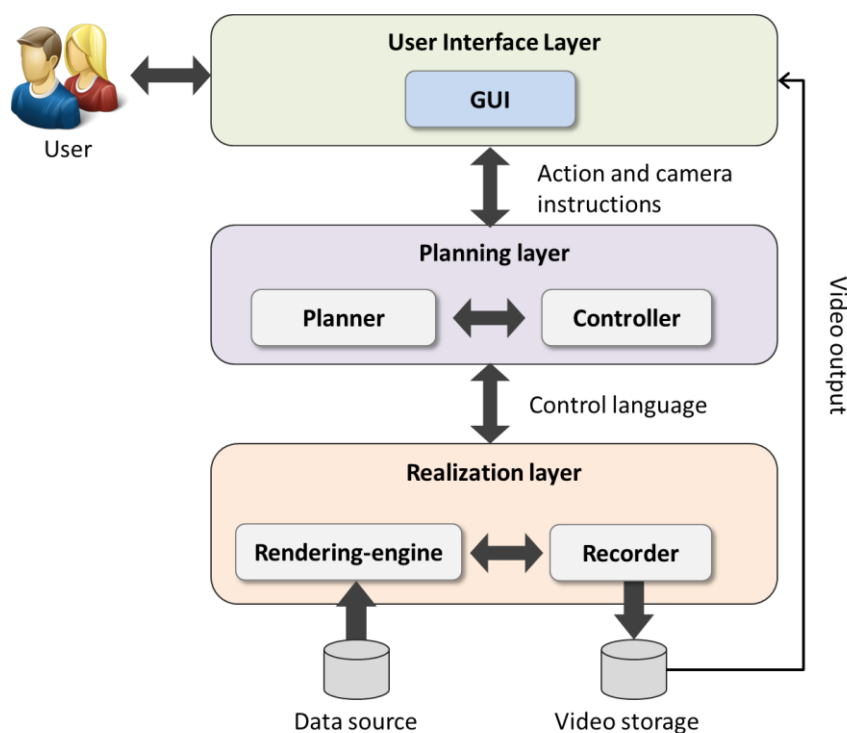


Figure 4.1: A generalised overview of frameworks for generating pre-visualisations (Young *et al.* 2004; Jhala and Young 2006).

Figure 4.1 illustrates a generalised framework for generating pre-visualisations using game-engines. The components of the framework are organised in three layers. The layers communicate with each other using the Extensible Markup Language (XML) via TCP/IP

connections. The layers include the user interface layer, the planning layer and the realisation layer. The following paragraphs discuss the framework from the bottom up.

The realisation layer is responsible for generating the pre-visualisation based on control language commands supplied by the planning layer. The realisation layer contains a rendering-engine that is responsible for generating the final output of the pre-visualisation. The data required by the rendering-engine is obtained from a data source which contains character models, prop models, set models, sounds, animations and textures. The realisation layer also contains a recorder component that captures the pre-visualisation and stores it on a file server. The user interface layer loads the pre-visualisation so that it can be viewed by the user. The BML Realization System and SmartBody are examples of realisation engines that can be used in the realisation layer (Thiebaut et al. 2008; Aleksandra Čereković, Tomislav Pejša and Igor Pandžić 2010).

The planning layer is responsible for processing action instructions and camera instructions from the user interface layer in order to produce control language scripts that can be processed by the components in the realisation layer. The planning layer contains a planner and a controller. The planner is responsible for processing high level action instructions and camera instructions in order to schedule the commands into a plan that can be interpreted by the controller component. Action instructions are used to control the behaviour of the characters in each scene. Camera instructions are used to specify the shots for each scene. Typically, the instructions are at a high level, e.g. “move to X” and “close-up shot of Y”. The planner can also process underspecified plans using domain knowledge and guidelines in order to generate complete plans. Darshak is an example implementation of such a planning component (Jhala and Young 2006). It accepts a sequence of commands and uses a planning algorithm to generate a complete plan that depicts what each character, prop and camera does throughout the pre-visualisation. The planning layer also contains a controller that converts the plan into a set of commands that can be processed by the realisation layer.

The user interface layer provides the user with a way of interacting with the tool in order to author the pre-visualisation. A GUI is provided that generates action instructions and camera instructions for the next layer. MWorld is an example GUI component which is used in the Zocalo framework (Young *et al.* 2004). The Unreal Tournament 2003 game-engine was modified in order to create the MWorld authoring GUI for the Zocalo framework. Longboard is another example GUI that is used within the Zocalo framework. It provides the user with a

sketch-based storyboarding tool for authoring pre-visualisations. Section 2.6.2.4 provided a review on the features provided by Longboard and similar approaches.

Existing GUI components for the user interface layer employ drag and drop interfacing techniques. Furthermore, storyboarding interfaces (such as Longboard) do not interpret the sketches created by the user in order to understand the 3D context of each scene. This research aims to contribute by investigating how sketch-based storyboarding GUIs can be improved by applying methods from Computer Graphics and Computer Vision. A framework for such a GUI is proposed in the next section.

4.3 A Framework for a Sketch-based Storyboarding GUI

This research proposes a framework called SISPA (Storyboarding Interface for Sketch-based Pre-visualisation Authoring) for the user interface layer in order to support sketch-based pre-visualisation authoring using a storyboarding approach. Figure 4.2 illustrates the proposed framework.

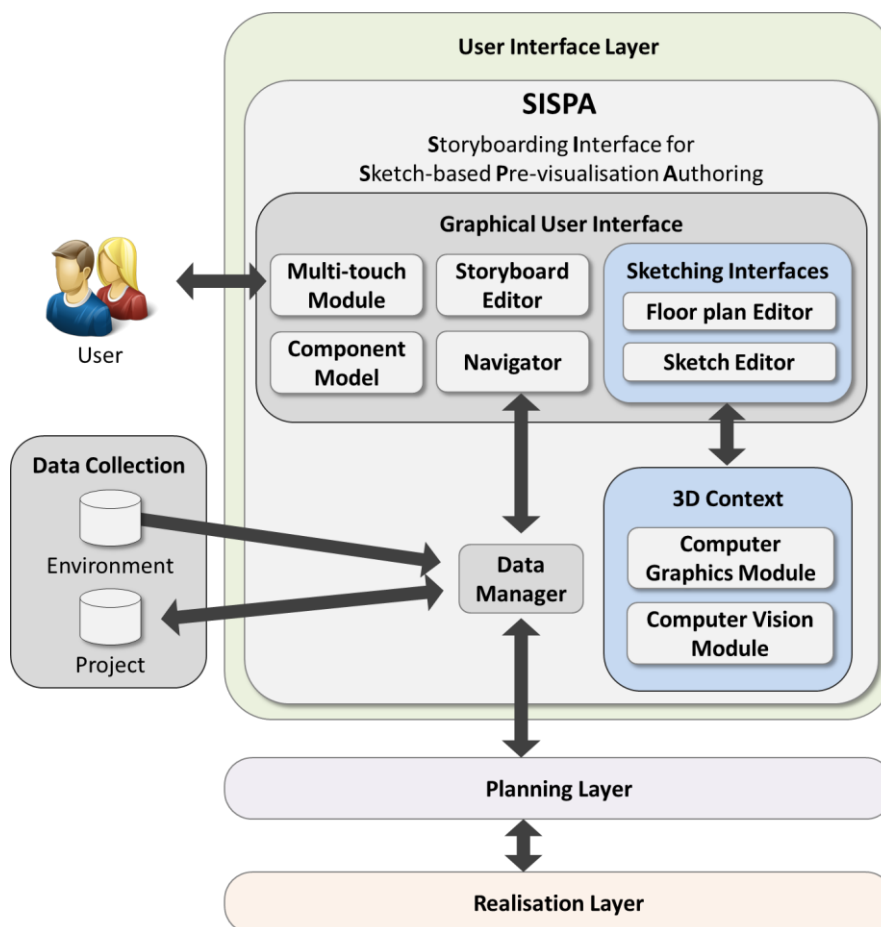


Figure 4.2: SISPA: A framework for sketch-based pre-visualisation authoring using a storyboarding approach.

SISPA has four main components, namely the Data Manager, the GUI, the Data Collection component and the 3D context component. Each component is made out of several subcomponents. This chapter discusses the design and implementation of these components and subcomponents.

At the core of the framework is the Data Manager. It is responsible for transferring data between the Data Collection component, the GUI and the Planning Layer. The Data Collection component (see Section 4.4) involves storing data from the environment and from the project being authored. Environmental data includes the script, 3D models and animations for characters and props, GUI data, and computer vision training data. Project data includes user provided data for authoring the pre-visualisation, e.g. shooting strategies, storyboards and sketches.

The GUI component (see Section 4.5) provides the user with a touch-friendly sketch-based GUI for storyboarding and staging. The GUI component has a Multi-touch Module for processing touch input and gestures. It also contains a Component Model which includes a framework of GUI controls that allow the user to interact with the tool. The Navigator is responsible for controlling the navigational flow between the various parts of the GUI. This includes a Storyboard Editor and two sketching interfaces. The first sketching interface is the Floor Plan Editor. It allows the user to create floor plans in order to diagram the shooting strategy. The second sketching interface is the Sketch Editor which allows the user to sketch the individual storyboard panels.

The framework contains a 3D Context component (see Section 4.6) which provides a 3D context to the floor plan editor and the Sketch Editor. The 3D Context component contains a Computer Graphics Module which allows the user to work directly on the 3D virtual environment. It also contains a Computer Vision Module which is responsible for interpreting the user's sketches.

4.4 Data Collection

The data collection component of the SISPA framework models environmental data and project data. The environmental data includes all the data required for the GUI to function and render the required assets. The user's project data includes all the data entered by the user in order to author the pre-visualisation, e.g. shooting strategy and storyboard sketches.

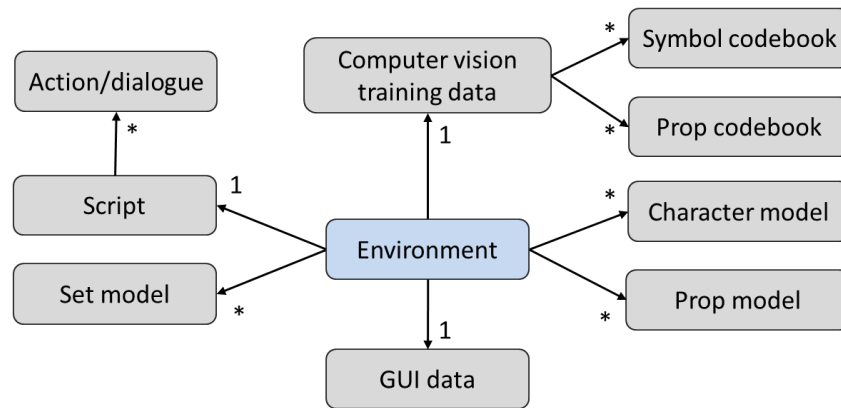


Figure 4.3: The Environmental Data Model.

The Environmental Data Model is illustrated in Figure 4.3. The environment includes a script, assets, GUI data and computer vision training data. The script contains a collection of action and dialogue elements. The assets are represented by 3D models. These include 3D models for sets, characters and props (see Figure 4.4). GUI data include the icons and images for functions and GUI controls. The computer vision training data includes codebooks required for recognising and props estimating the pose of 2D symbols (such as characters, shots and emotions) and 3D props (see Section 4.6). The environmental data is represented using the Extensible Markup Language (XML). Appendix B provides an example of the XML documents used to store references to script data, assets and training data.

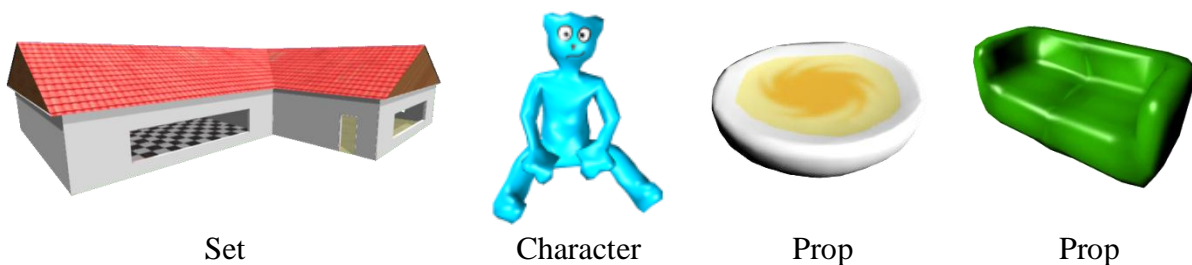


Figure 4.4: Example assets that form part of the environment.

The Project Data Model represents the user's project data in terms of the script (see Figure 4.5). The script is structured in terms of scenes and dramatic blocks as discussed in Section 2.3. Each dramatic block is associated with a single floor plan and a single storyboard.

The floor plan is used to create the shooting strategy for a particular dramatic block (see Section 2.4). It illustrates where the action takes place as well as the required props and characters. The character blockings, camera shots and motion paths are indicated on the floor plan. The floor plan also requires the user to associate each shot with the characters that appear in it.

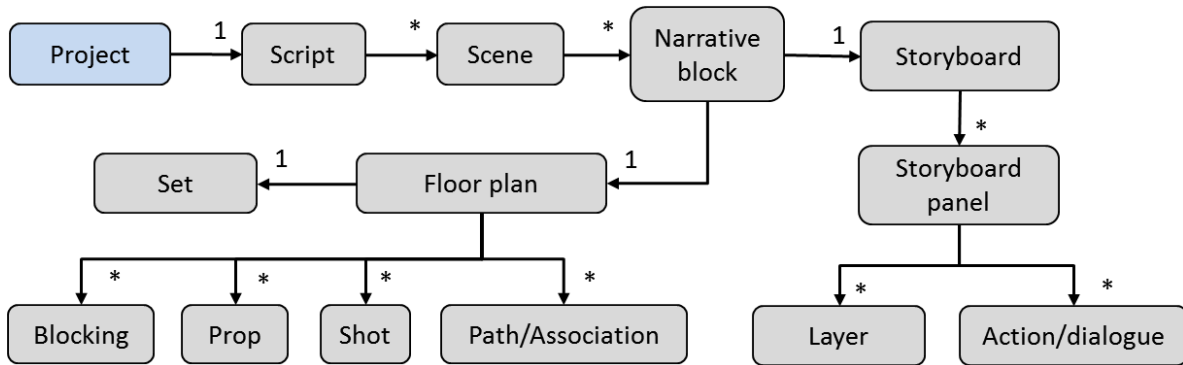


Figure 4.5: The Project Data Model.

The storyboard for a particular dramatic block contains several storyboard panels. Each storyboard panel is associated with several action/dialogue script elements which describe the character action and narrative content of the corresponding shots. If a storyboard panel is sketched by the user then it also contains a layer for each prop and character. This allows the user to sketch and edit the individual props and characters separately. It is important to note that the layering concept allows sketched objects to be separated so that each object can be recognised and posed individually (see Section 4.6).

The project data model influences the design of the GUI. The navigation between the various components of the GUI should support the logical structure of the user's data. The next section will discuss the design of the overall GUI, the navigational flow between its subcomponents and the design of each individual subcomponent.

4.5 Graphical User Interface Design

This section discusses the design of the proposed sketch-based GUI for authoring pre-visualisations using a storyboarding approach. The GUI is designed to be used on touch-enabled tablets that allow users to create and annotate sketches using a stylus. The GUI is implemented and evaluated on the Asus EEE EP121 (see Table 4.1 and Figure 4.6.)

Table 4.1: Important specifications of the Asus EEE EP121 tablet.

Feature	Specification
Processor	Intel Core i5 470um, 1.33 GHz
Memory (RAM)	4 GB
Graphics Acceleration	Integrated, 166 MHz
Stylus-enabled	Yes
Multi-touch support	Two points, Capacitive
Display	12.1" LED Backlight WXGA Screen



Figure 4.6: The Asus EEE Slate EP121 used for prototyping (Asus 2012).

This section begins by discussing the look and feel of the GUI. It also discusses the design considerations taken into account in order to take advantage of the tablet's support for multi-touch interaction. The section continues by discussing the overall layout of the subcomponents of the GUI and how the user navigates between them. It also discusses the functions each subcomponent supports. In particular, it focuses on the subcomponents that support sketch-based input. This includes the floor plan editor and the storyboard editor.

4.5.1 Look and Feel

Existing pre-visualisation authoring GUIs make use of traditional WIMP interfaces, e.g. animation and modelling tools (see Section 2.6.2). Furthermore, sketch-based pre-visualisation authoring tools, such as Longboard, also follow this design approach. The problem with the approach is that the traditional WIMP design approach works well for desktop user interfaces but it does not work well for touch-based user interfaces that operate on portable touch enabled devices such as tablets. The sketch-based design proposed by this research aims to contribute by designing for a touch-enabled portable device that can support sketch-based storyboarding. The design is touch-friendly and supports several of the gestures provided by smartphones and tablets (McKenzie 2011; Nielsen *et al.* 2003). The User Interface (UI) design includes a multi-touch module for recognising the user's gestures. Figure 4.7 (a-g) illustrates the gestures that the user can use to interact with the UI. The multi-touch module supports taps, single finger gestures and double finger gestures. The implementation of the multi-touch module is discussed in Section 4.7.3.3.

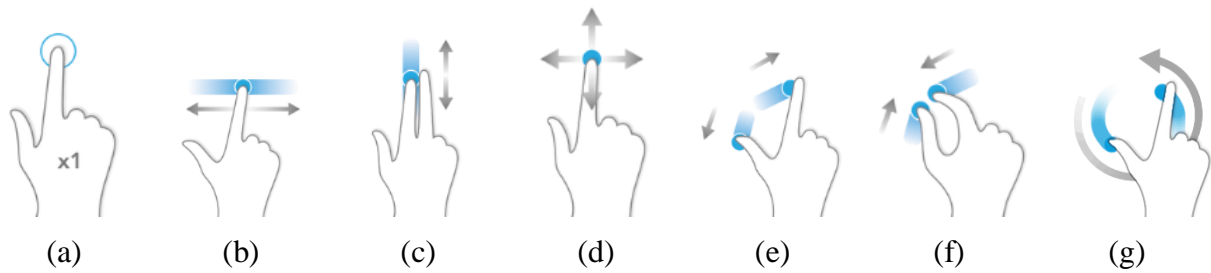


Figure 4.7: Touch gestures for performing a (a) Tap (b) Horizontal flick/scroll (c) Vertical flick/scroll (d) Drag (e) Spread/enlarge (f) Pinch/shrink (g) rotate.

The component model is designed to provide UI components that appear similar to ones used by the well adopted and touch-friendly iOS and Android operating systems (Gadhavi and Shah 2010). The design supports touch interaction by limiting the minimum size of all the UI components that can be interacted with. The icons are large enough for the user to touch comfortably using a single finger. The design also takes a minimalistic approach by avoiding unnecessary lines, bevels and nested panels that are often seen in traditional WIMP interface designs. The user is provided with touch feedback in the form of animations and component highlighting effects. Appendix C provides a summary of the touch gestures used for manipulating objects and navigating throughout the GUI. The design of the Component Model is illustrated throughout this section using screenshots of the final design.

4.5.2 Interface Layout and Navigation

The subcomponents of the GUI are organised within three layers as shown in Figure 4.8. These layers include the story layer, the dramatic block layer and the shot layer. The granularity of the user’s view of the story depends on the currently active layer. The story layer provides an overview of the entire story. The dramatic block layer provides the user with access to all the assets for that particular dramatic block. This includes the relevant script segment, the floor plan with the shooting strategy and the storyboard. The shot layer provides the user with access to the individual storyboard panels which represent the individual shots in the shooting strategy. The user can navigate between the various layers using the spread gesture, the pinch gesture and the zoom “in/out” buttons. Tapping the “zoom in” button at the bottom right of each screen, or using the spread gesture, zooms in to view the story at a finer granularity. Similarly, using the pinch gesture, or tapping on the “zoom out” button, views the story at a coarser granularity.

The story layer contains a single screen, namely the story viewer. This screen provides the user with a list of scenes and dramatic blocks in order to illustrate an overview of the story. It

also allows the user to navigate to a particular dramatic block. This is done by selecting the dramatic block by tapping it once. Using the spread gesture, or tapping the “zoom in” button, will take the user to the layer for the selected dramatic block.

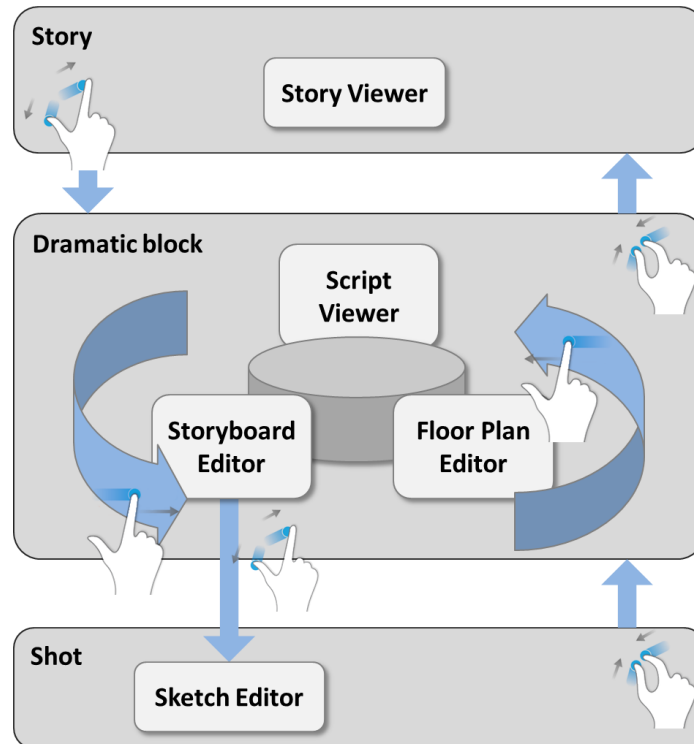


Figure 4.8: GUI layout and navigation.

The dramatic block layer contains the script viewer, the floor plan editor and the storyboard editor. The screens are organised around a conceptual cylinder. The user can rotate the screens around the cylinder from left to right or vice versa using the flick gesture. The GUI indicates neighbouring screens using tabs to the left and right edges of the device’s display. The script viewer shows the part of the script for the current dramatic block. The floor plan editor allows the user to diagram the shooting strategy for the current dramatic block. The storyboard editor can be used for adding, removing and rearranging storyboard panels. An individual storyboard panel can be selected by tapping its image. If a storyboard panel is selected then the user can navigate to the next layer using the “zoom in” button or the spread gesture.

The shot layer contains the Sketch Editor. It allows the user to visualise a specific storyboard panel and edit it. The Sketch Editor provides a sketching interface for sketching the individual storyboard panels. It can also extract information from the user’s sketch using algorithms from computer vision (see Section 4.6) in order to create a 3D pre-visualisation of the particular shot.

The overall layout of the user interface supports the logical structure of the user's data. This makes it easier for the user to quickly navigate to the various parts of the story. The remaining subsections discuss the subcomponents of the GUI in greater detail. The components that only require touch-based interaction are briefly discussed. This includes the story viewer, the script viewer, and the storyboard editor. A detailed discussion is provided about the design of the subcomponents requiring sketch-based input. This includes the floor plan editor and the Sketch Editor.

4.5.3 Story Viewer and Script Viewer

The Story Viewer provides a way of navigating between the dramatic blocks of the story. It shows a two-level hierarchical list of the story. The first level lists the scenes of the story in alphabetical order. The second level lists the dramatic blocks for each scene numerically. It also provides a “delete” button for clearing all the data for the current project. A confirmation dialogue box appears to confirm the deletion in order to prevent accidental data loss.

The Script Viewer displays the part of the story's script for the current dramatic block. It provides a list of items for illustrating action and dialogue (see Section 2.3). These items can be used to populate the action/dialogue entries of the individual storyboard panels. The need for entering text into the tablet is therefore removed. This design decision is based on the fact that entering text on portable touchscreen devices is difficult due to the lack of tactile feedback (Hoggan, Brewster and Johnston 2008).

4.5.4 Storyboard Editor

The Storyboard Editor shows the storyboard for the current dramatic block (see Figure 4.9). It allows the user to add new, blank storyboard panels and remove existing ones using the “add” button (plus) and the “remove” button (cross) respectively. Removing a storyboard panel also removes the corresponding shot in the floor plan. The user can add action/dialogue elements to each storyboard panel. This is achieved by tapping a panel to select it and then tapping the “dialogue” button (speech balloon) in order to show the script fragment selection dialogue box. The user selects the script fragments that are to be placed into the dialogue/action area of the storyboard panel. The user can also rearrange the storyboard panels using the arrow buttons.

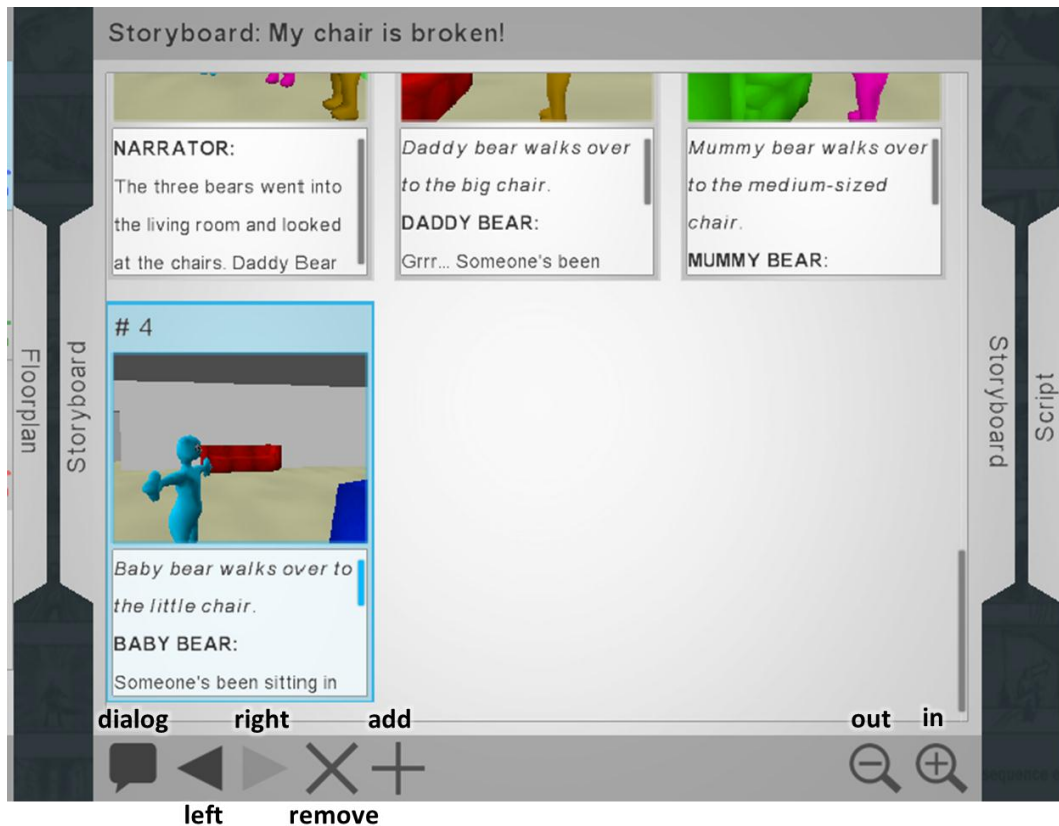


Figure 4.9: A screenshot of the storyboard editor.

The currently selected storyboard panel determines which action/dialogue list box can be scrolled (shown with a blue scrolling indicator). The entire storyboard cannot be scrolled while a storyboard panel is selected (its scrolling indicator is greyed out). The user is required to deselect the currently selected storyboard panel so that the scrolling indicator for the entire storyboard activates. This design decision was made because informal testing showed that it was difficult to scroll if both the storyboard and the storyboard panels received scroll input.

4.5.5 Floor Plan Editor

Directors and storyboard artists visualise the staging and shooting strategy for each dramatic block by sketching overhead views called floor plans (see Section 2.4). They are used to illustrate the contents of the set. Floor plans also illustrate the movements of the characters and the camera on the set. The Floor Plan Editor allows the user to sketch props directly on an interactive top-down view of the 3D virtual environment as shown in Figure 4.10. It also allows the user to sketch the shooting strategy by annotating the floor plan. The annotations are used to indicate character blocking and camera movement. The Floor Plan Editor has three parts. It has an interactive editor that supports sketching and touch-based manipulation, a toolbar at the bottom of the screen and a list of props placed at the right of the screen.

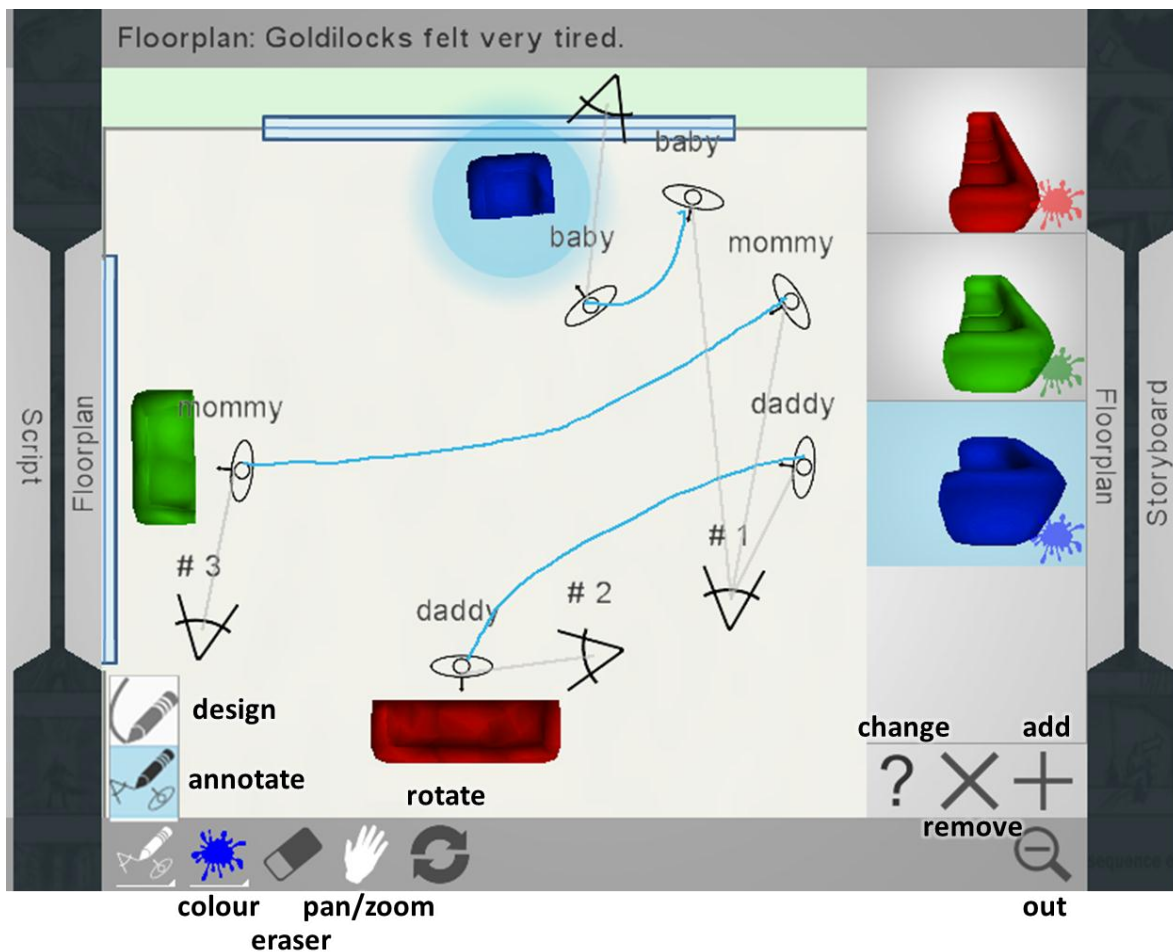


Figure 4.10: A screenshot of the floor plan editor.

The tool bar provides several tools for manipulating objects on the floor plan. It has two sketching modes, a colour chooser, an eraser toggle, a “pan/zoom” toggle and a rotation toggle. The “design” sketching mode is used for sketching props in order to design the set. The “annotate” sketching mode is used for annotating the script in order to perform the staging for the dramatic block. The colour chooser allows the user to specify the colour of a prop. This is important for a pose estimation algorithm that will be discussed in Section 4.6.6. The eraser is useful for erasing props, symbols and parts of sketches. The “pan/zoom” toggle instructs the floor plan editor to use touch input to move objects and change the view of the set instead of navigating to other components of the GUI. The rotation toggle is similar to the movement toggle except that it is used for rotating objects. The “pan/zoom” toggle and the rotation toggle are mutually exclusive.

The prop list can be used to add a new prop. This is achieved by tapping the “add” button (plus). The new, blank prop is shown in the list but not on the floor plan. This is because the 3D model of the prop is unknown. While in the “design” sketch mode, the user sketches the prop as it would be seen from the top-down view (see Figure 4.11 (a)). An object recognition

algorithm recognises which prop has been sketched and shows a progress bar below the prop in the list. The user's sketch is substituted with a 3D model and placed in the 3D virtual set when the algorithm finishes. The prop is stacked on top of another prop if the user sketched them overlapping, e.g. a bowl on a table. A prop can be selected by tapping on it on the floor plan or in the prop list. The selected prop can be removed by tapping the "remove" button (minus). If the incorrect prop is recognised then the user can select the prop and change it to the correct prop by tapping the "change" button. A list of candidate props is shown from which the user can select the correct prop.

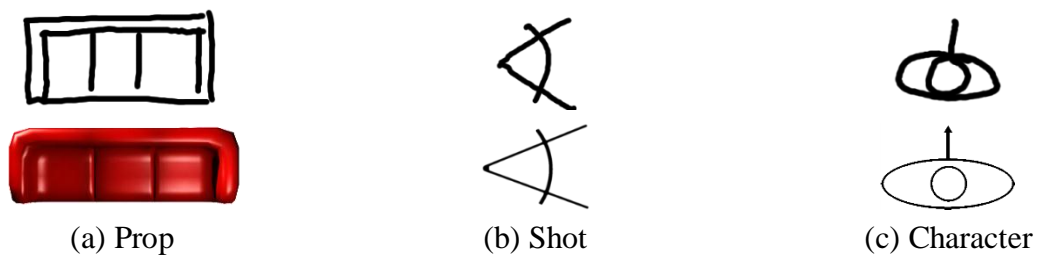


Figure 4.11: Sketches with their matching objects.

The user can sketch the shooting strategy for the current dramatic block by changing the sketching mode to "annotate". The user can add new annotation symbols by sketching directly onto the floor plan without adding the symbols explicitly using a button. The purpose of the prop list is to allow the user to sketch and select props that overlap each other. It is assumed that shots and characters never overlap. This design prevents the user from having to manage two lists or mixing annotation symbols with props in the same list. To indicate a new shot, the user sketches the shot symbol as shown in *Figure 4.11* (b). An object recognition algorithm recognises and places the shot on the floor plan. An animated spinning donut provides feedback to the user while the algorithm is running. Similarly, the user can add character blockings by sketching character symbols on the floor plan as shown in *Figure 4.11* (c). The user is prompted to select the desired character when a character symbol is recognised. Shots and character blockings can be connected with paths in order to indicate motion, as shown in *Figure 4.10*.

Solid blue paths indicate character motion and dashed blue paths indicate camera movement. Sketching a path from a camera symbol to a character symbol, or vice versa, associates the shot with the character blocking. The association is visualised with a straight grey line connecting the camera symbol and the character symbol. This is useful for specifying in which shots the characters appear. Moving or erasing connected shots or character blockings removes all the connected motion paths and associations. The tool automatically adds the

corresponding storyboard panels to the storyboard for each shot indicated on the floor plan. Each storyboard panel shows the 3D scene with the indicated props and characters. The shots and corresponding storyboard panels are automatically numbered and managed. The shot numbers are automatically updated when adding new shots, removing existing shots and reordering shots.

4.5.6 Sketch Editor

The Sketch Editor provides the user with a sketching interface for sketching the individual panels of the storyboard. The user begins with a blank sketch and then sketches the props seen in the shot. The props are recognised and then positioned and orientated in a 3D space. The tool can be used for establishing the position and orientation of the camera automatically once all the props have been recognised. The design also allows the user to work with a storyboard panel with a known camera. When the position and orientation of the camera are known then the user can sketch the characters that are visible in the shot. Their body posture and emotional expressions are extracted directly from the user's sketches. The Sketch Editor also allows the user to manually adjust the camera using touch gestures. It is possible to sketch props directly onto the 3D visualisation of the scene in order to add additional props. This section will discuss the various tools offered by the Sketch Editor for sketching new shots, adding characters and adjusting existing shots.

4.5.6.1 Sketching New Shots

The overall design of the Sketch Editor is similar to the Floor plan Editor. It has an interactive editing area that supports sketching and touch-based manipulation, a toolbar at the bottom of the screen and a list of layers at the right of the screen (see Figure 4.12). The Sketch Editor can be used to sketch a new storyboard panel from a blank sketch. The user begins by adding a new prop layer. The GUI shows a screen with a list of possible layer options. The options include a layer type for props and a layer type for each character in the story. The user taps the "accept" button after selecting the layer type for a new prop. A blank layer entitled "unknown prop" is added to the list of layers on the right side of the screen. The user sketches the prop as it should appear in the shot. Each prop and character is sketched on a separate layer. The colours of each prop layer should match the colour of the corresponding props in the floor plan. This UI design is a performance related requirement of the camera estimation algorithm (see Section 4.6.6).

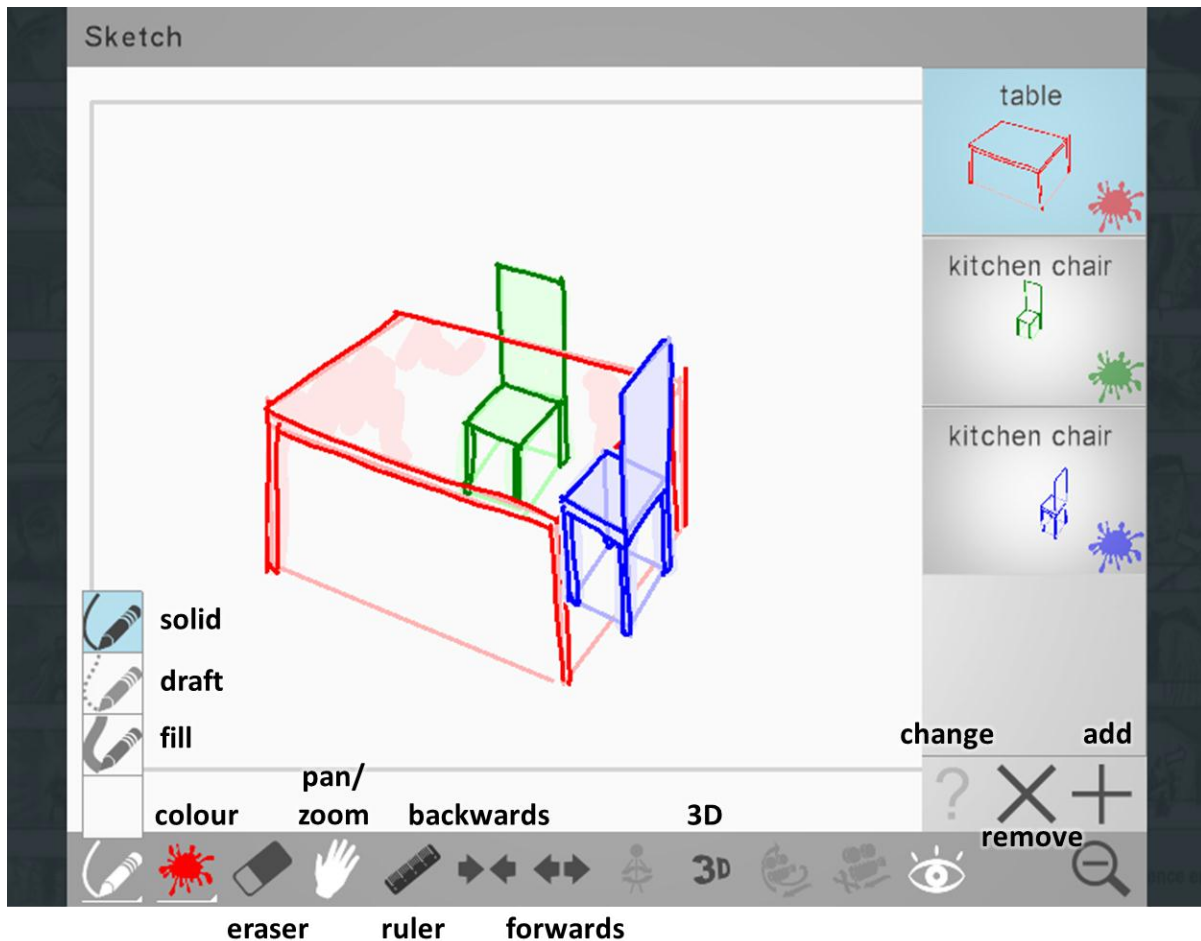


Figure 4.12: A screenshot of the Sketch Editor for sketching a new shot.

The Sketch Editor supports four sketching modes. The solid pencil is used for sketching the final version of each layer. The final version of a layer is used for recognising the object and estimating its pose. The draft pencil is used for quickly drafting the contents of the layer. This makes it easier to sketch the final version of the layer. The fill pencil allows the user to add shading to the layer. Complete sketches of each prop and character are required by the computer vision algorithms discussed in this chapter. To make it easier to visualise the sketch, the fill pencil can be used to “colour in” parts of a layer that should occlude layers below it. For example, the blue chair is closer to the viewer than the red table. Both objects have been completely sketched but the chair appears to block the view of the corner of the table. The emotion pencil is used to sketch the facial expressions of the characters. It is only visible when the selected layer is a character (see Figure 4.13).

The toolbar at the bottom of the screen also shows several tools. The sketch mode chooser allows the user to change the current sketching mode. The colour chooser is used to specify the colour for the currently selected layer. The user can use the stylus to erase parts of the currently selected layer if the eraser is toggled. The eraser will erase strokes that match the

current sketching mode. For example, if the sketching mode is set to drafting then the eraser will only erase draft strokes. Solid strokes, fill strokes and emotion strokes will not be erased. The “pan/zoom” function is similar to the one provided by the floor plan editor. The toolbar contains a ruler toggle and two layer ordering buttons. The ruler allows the user to draw straight line segments easily. Layers can be reordered by tapping the “move forwards” and “move backwards” buttons.

An algorithm processes each layer after the user finishes sketching. It begins five seconds after the last stroke is added to the layer. If the user continues sketching while the algorithm is running then the algorithm is interrupted. It then waits another five seconds after the user finishes making the change. The user’s sketch can be converted into a 3D visualisation of the shot once all the layers have been sketched and processed.

The camera estimation algorithm is called once the user taps the “3D” button. A list of ranked candidate shots is presented to the user. The shots are ranked on the quality of the estimated camera. The storyboard panel’s background changes to a view of the 3D scene when the user selects the desired shot. The user can select an alternative shot by using the “change” button and selecting a different candidate shot. Figure 4.13 shows a screenshot of the 3D shot generated for the sketch provided in Figure 4.12.

4.5.6.2 Adding Characters

Characters can be added to the shot once the 3D context of the camera is known. Storyboard artists often begin sketching characters by sketching simple stick figures to plan their posture and proportions. The artist then “fleshes out” the stick figure in order to sketch a complete character (see Section 2.5.3). The Sketch Editor allows the user to add characters to the 3D scene by indicating their body posture and position on the set using stick figures. To do this, the user adds a new character layer. The tool provides a list of options, showing a “prop” option and an option for each character. Selecting the desired character adds a new character layer for the character. The user can sketch the character’s stick figure using the solid pencil. The eraser can be used for erasing parts of the stick figure. An algorithm analyses the stick figure as it is sketched. The character is then placed in the shot with the indicated posture when the stick figure is completed. The stick figure changes from a uniform black colour scheme to a red, blue and black colour scheme. If the character posing algorithm recognises the stick figure correctly then the red part of the stick figure indicates the character’s right half

and the blue side indicates the character's left half. The character is assumed by default to face the camera.

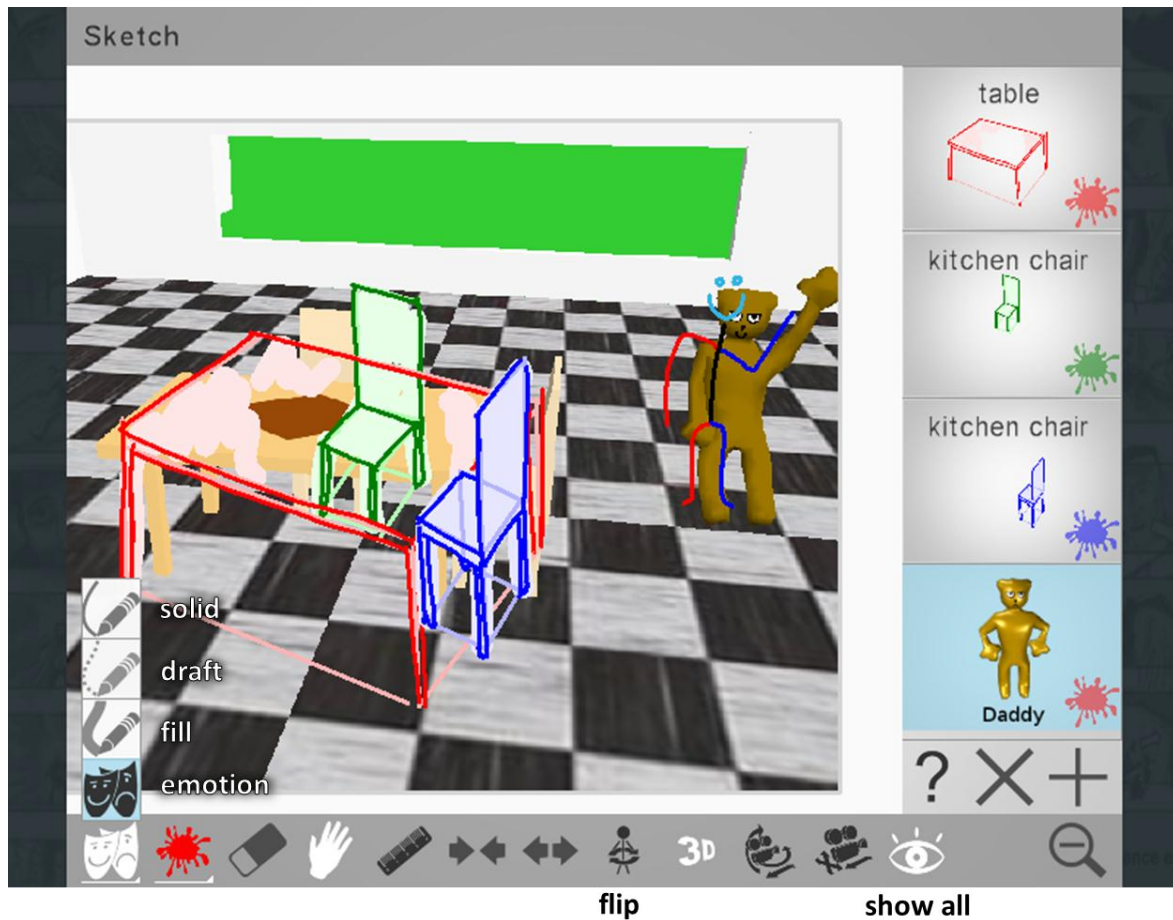


Figure 4.13: A screenshot of the Sketch Editor for adding a character.

In order to sketch characters that are facing away from the camera, the user can enable the “flip” toggle. This will interchange the red and blue colours of the stick figure and turn the 3D character model 180° on the set. Characters that are added in the Sketch Editor are also automatically added to the floor plan for the dramatic block if they do not exist yet.

The facial expressions of each character can be specified using the emotion pencil to sketch the character's facial expression as shown in Figure 4.13. Emotion strokes are indicated in blue and they are associated with the layer for the particular character. The user illustrates the expression by sketching an emotion symbol on the character's face. Figure 4.14 shows the default emotion and three example emotions. An algorithm processes the sketched emotion symbol to determine which emotion was sketched. The texture of the 3D character model is updated to reflect the facial expression on the character's face. The current GUI implementation supports a default facial expression and three facial expressions (happy, sad

and angry) for each character. The emotions can be extended by providing additional training data and corresponding emotion textures for the 3D models.

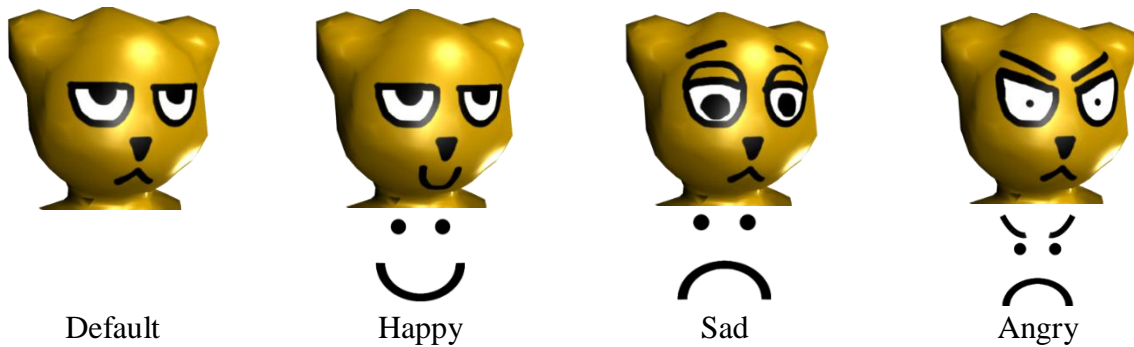


Figure 4.14: Example facial expressions with their matching emotion symbols.

4.5.6.3 Adjusting Shots

The Sketch Editor also allows the user to adjust an existing shot for which the camera is already known. Figure 4.15 shows a screenshot of the Sketch Editor showing the same 3D scene but from a different view. The toolbar provides a “rotate camera” toggle and a “move camera” toggle that can be used to rotate the camera and move the camera in 3D space respectively.

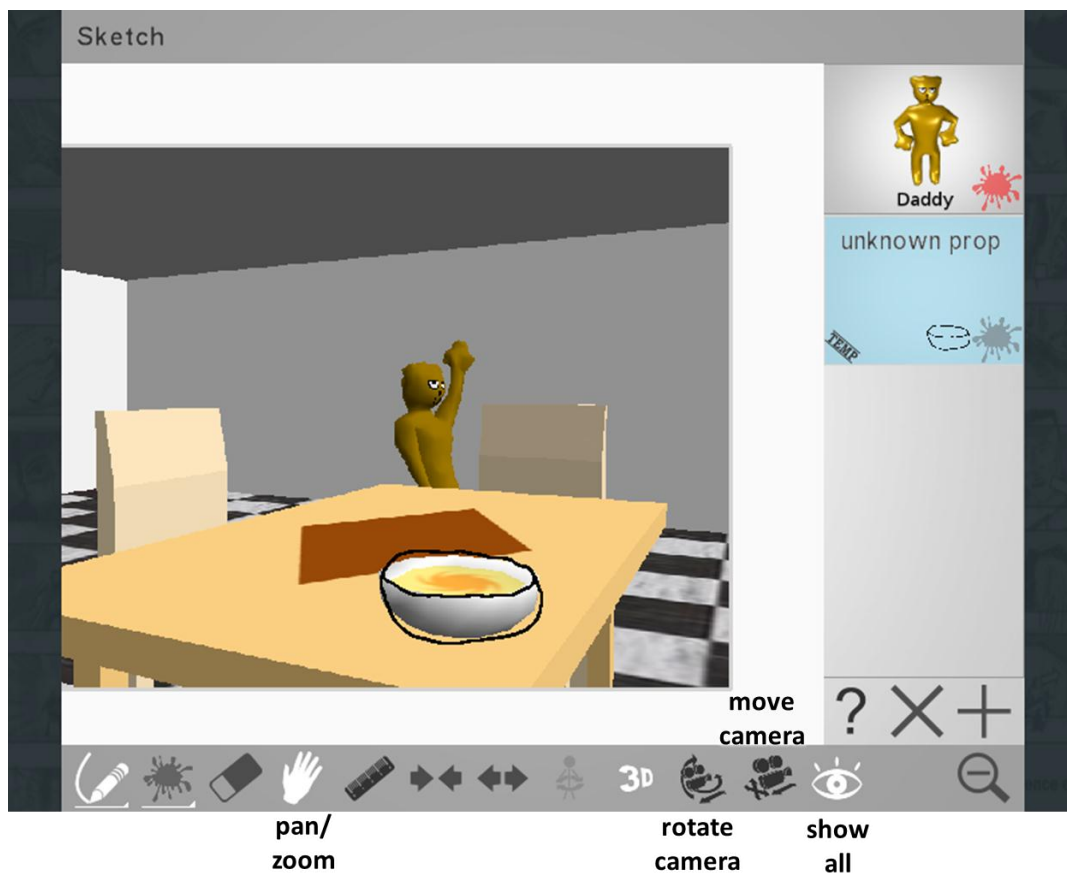


Figure 4.15: A screenshot of the Sketch Editor for adjusting an existing shot.

Camera rotation is in the opposite direction to the drag direction. For example, if the user drags the viewing area to the left then the camera will rotate to the right. This design is consistent with the way the user pans the view in the floor plan editor. The spread gesture will move the camera forwards and the pinch gesture will move the camera backwards. Similarly, if the “move camera” toggle is enabled then dragging the viewing area will move the camera within the current viewing plane. The movement of the camera is in the opposite direction to the drag gesture. Spreading and pinching while moving the camera will also move the camera forwards and backwards.

The user can also sketch props directly onto a storyboard panel with a known 3D camera. This can be done by adding a new prop layer and sketching the prop as it should appear in the shot. The layer is marked as temporary with a “temp” icon. An algorithm recognises and places the prop in the virtual set when the user finishes sketching the prop. It intelligently places the prop so that props can be stacked or placed on the ground. For example, drawing a bowl on the table will result in the prop being positioned on the table’s surface as shown in Figure 4.15. The temporary layer is removed after the prop is placed in the 3D set. Props and characters that are added using the Sketch Editor are automatically added to the floor plan for the relevant dramatic block. Characters are also automatically associated with the relevant shot for the storyboard panel.

4.5.7 Discussion

Existing GUI designs for authoring pre-visualisations employ traditional WIMP user interfaces that follow the point-and-click interaction approach. It was mentioned in Section 2.6.3 that this method does not intuitively fit into the pre-production phase because its activities mainly involve writing, diagramming and sketching. Existing sketch-based GUIs, such as Longboard, do not automatically interpret sketches and they do not fully support the multi-touch environment provided by today’s multi-touch portable devices (Jhala *et al.* 2008).

This section discussed a GUI design that provides a touch-friendly environment for sketching floor plans and storyboards in order to author pre-visualisations. The sketch-based GUI is designed to be similar to traditional paper-based approaches in order to allow the user to sketch floor plans and storyboards quickly and easily. A stylus can be used to sketch a storyboard directly on a tablet computer (see Figure 4.6) and use gestures to move and resize the viewing area of the storyboard “page”. The design is also structured around the activities

performed during the pre-production phase. This supports GUI navigation that is logical to the user.

Several algorithms are required in order to provide the user with a sketching environment that is capable of interpreting user sketches. These include algorithms for recognising the 2D symbols and 3D objects that the user sketches. It also includes algorithms for estimating the location of each object, orientating it, and determining the camera parameters required in order to obtain the shot sketched by the user. Algorithms for posing and positioning characters from stick figures are also required in order to support the GUI design. The next section discusses the design of these algorithms and how they are used by the GUI.

4.6 Algorithm Design

The different types of elements that are required to be interpreted in order to author *minimalistic* pre-visualisations were discussed in Section 2.9. This section discusses the design of the algorithms required by the Computer Vision Module in order to interpret sketches containing these elements. It begins by identifying the key requirements for interpreting the user's sketches in order to support the elements for *minimalistic* pre-visualisation authoring. It provides a brief discussion on how vector data and raster data is converted, and continues to discuss several approaches for describing sketches. A discussion on the approach used to recognise objects from user sketches is then provided. Methods used for estimating the pose of props and characters are discussed for known shots and unknown shots. The algorithm design is concluded with a discussion on estimating the camera from a user's sketch.

4.6.1 Requirements for Interpreting Sketches

The Computer Vision Module is required to automatically interpret floor plans and storyboards. Table 4.2 summarises the different types of objects that should be interpreted. These include the symbols, props and characters. The algorithms used by the Computer Vision Module are required to reliably describe, recognise and pose objects sketched by the user within a reasonable amount of time (a few seconds or less with the hardware discussed in Section 4.5). The image description method used should be invariant to changes in the translation, scaling and rotation of the image. Local feature correspondences are required in order to estimate the orientation and location of the 3D props found in storyboard sketches.

The Computer Vision Module is also required to estimate the location and orientation of the camera from the user's sketch.

Table 4.2: Objects that are required be interpreted by the Computer Vision Module.

Object	Space	Type of Body	Requirements
Floor plan symbol	2D	Rigid	Recognise the symbol and estimate its 2D orientation and location.
Floor plan prop	2D	Rigid	Recognise the prop and estimate its 2D orientation and location.
Storyboard symbol	2D	Rigid	Recognise the symbol.
Storyboard prop	3D	Rigid	Recognise the prop and estimate its 3D orientation and location.
Storyboard character	3D	Articulated	Estimate the 3D location and orientation of the character and orientate its body parts.

4.6.2 Raster Representations and Vector Representations

The algorithms discussed in this section make use of raster-based representations and vector-based representations. A *raster image* is represented by a grid of pixels. Photos are often stored using raster images because they are capable of illustrating fine details and textures. The disadvantage of the approach is that pixelisation anomalies occur when raster images are transformed. Vector images do not suffer from this problem because they are represented using geometric shapes, polygons and curves (Hill and Kelley 2007). The sketches created using the sketching components discussed in Section 4.5.5 and Section 4.5.6 are represented in a vector-based format. The user's strokes are represented by collections of line segments called *edgels* (Shotton *et al.* 2008). Some of the algorithms discussed in this section operate on edgels and other algorithms require them to be sampled into a raster image. This subsection discusses how the Computer Vision Module is designed to convert between the two image representations.

4.6.2.1 Sampling Edgels from a Raster Image

Edgels are extracted from raster images by applying two steps:

- Step 1. Apply a series of image filters to create an edge map.
- Step 2. Apply an edge tracing algorithm to find the edges.

Image filters are applied to the original raster image to create an edge map (see Section 3.3.2). The edge map indicates where the edges are in the original raster image. The value for each individual pixel of an edge map provides a measure of the strength of the edge response at the

pixel location. The original image is converted into a floating point greyscale image. The Laplacian filter used by the Marr-Hildreth edge detector is then applied to the greyscale image (Marr and Hildreth 1980). This produces a raster image that has negative values and positive values along edge responses. Pixels that have near-zero values are non-edge responses. The absolute magnitude of the pixel intensity indicates the strength of the edge response at that location. The resulting edge map is blurred slightly using a bilinear image filter. This reduces the noise of the image and thickens the edges. If an edge response is in the interval of $[0.005,1]$ or $[-1,-0.005]$ then the pixel value is set to one, otherwise it is set to zero. The resulting edge map is blurred using the bilinear filter in order to reduce noise and thicken the edges. Figure 4.16 (b) shows the edge map that is obtained by applying the series of filters on the original grey scale raster image obtained from Figure 4.16 (a).

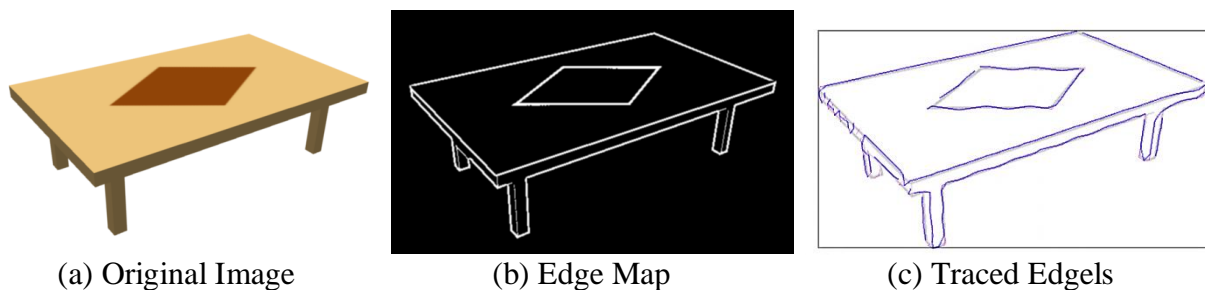


Figure 4.16: Sampling edges from a raster image.

The next step is to extract the edgels from the edge map. This is achieved by tracing the edges in the edge map. Figure 4.16 (c) illustrates the edgels identified from the edge map provided in Figure 4.16 (b). The design of the Computer Vision Module includes an edge tracing algorithm. It calls a tracing subroutine for each non-zero pixel of the pixel map. The subroutine starts at a particular pixel location and adds edgels to an edgel collection as it follows the pixels with strong edge responses along on the edge map. It also clears the pixels it encounters on the edge map as it progresses. This prevents the algorithm from iterating indefinitely and it also reduces the tracing time required. The trace subroutine considers a pixel and then erases a small area of pixels around it. The subroutine then considers the local region of non-zero pixels in order to find pixels with strong edge responses. The candidate edgel is the line segment connecting the current pixel to the candidate pixel. Each candidate pixel is scored on the quality of the candidate edgel. An error value measures how perpendicular the candidate edgel is to the image gradient and how parallel the candidate edgel is to the previously found edgel. The candidate edgel with the lowest error is selected and the subroutine continues from the end of the selected edgel until no further candidate

edgels can be found. The edge tracing algorithm produces a set of edgels while minimising tracing anomalies such as zigzags.

4.6.2.2 Sampling a Raster Image from Edgels

Some of the methods applied in the design of the Computer Vision Module, such as SIFT, require raster-based input (see Section 3.3.3). The problem is that the sketches recorded by the sketching components of the GUI represent the user's sketches using a collection of edgels. It is therefore necessary to sample a raster-based representation of these edgels in order to take advantage of raster-based computer vision methods. This is achieved by sampling a raster image from the edgels. This is achieved in two steps:

- Step 1. Place and orientate a rectangular grid or elliptical grid over the edgels.
- Step 2. Set the intensity of each pixel of the raster image by measuring the distance from the corresponding grid cell to the closest edgel.

The first step involves placing a grid over the edgels. The grid can be a rectangular arrangement of cells organised in rows and columns or an elliptical arrangement of cells organised in tracks and sectors (see Section 3.3.3.2).

The second step is to calculate the intensity of each pixel in the raster image. The distance from the centre of a cell to the nearest edgel used to determine the pixel value for the cell. Figure 4.17 illustrates how the distance from each cell of a rectangular grid arrangement is used to calculate the pixel intensities.

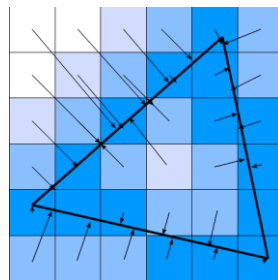


Figure 4.17: Sampling a raster from edgels.

The intensity of each pixel is calculated using Equation (4.1) and Equation (4.2) where I is the value of the pixel, d is the distance from the cell's centre to the closest edgel, D is the maximum distance to an edgel and α is a measure of the sharpness of the raster edges. The raster is sampled at a higher resolution and then reduced to a lower resolution in order to improve the quality of the resulting raster image and reduce aliasing anomalies which are caused by low resolution sampling. The parameters D and α are used to control the thickness

and smoothness of the raster representation of each edgel. Smaller values for D produce thinner edges and larger values of α produce sharper edges. These parameters depend on the requirements of the object recognition algorithm and the limitations of the hardware.

$$I = \begin{cases} S & : d < D \\ 0 & : otherwise \end{cases} \quad (4.1)$$

$$S = \sin\left(\frac{D-d}{D} \cdot \frac{\pi}{2}\right)^\alpha \quad (4.2)$$

4.6.3 Approaches for Describing Sketches

Many of the approaches discussed in Section 3.3 are aimed at processing textured images. The problem is that the sketches provided by the user are outlined and not textured. An image description method is therefore needed to describe the user's sketches and fulfil the requirements identified in Section 4.6.1.

This section discusses the design of several approaches for describing user sketches. Figure 4.18 provides an overview of all the methods and variations investigated during the design of the Computer Vision Module. The green positive blocks show the strengths of each variation and the red negative blocks show their weaknesses. Three overall approaches were investigated. This includes a contour-based method, two part-based approaches and several grid-based methods. The approach implemented by the Computer Vision Module is a combination of the most suitable approaches.

4.6.3.1 Contour-based Approach

The first approach is to use the distance function discussed in Section 3.3.3.3. The distance function measures the distance from the centre of the sketched object to the points along its contour. N distances are sampled between the angular interval of $[0, 2\pi]$. The distance function is transformed into the frequency domain using the one dimensional Discrete Fourier Transform in order to approximate the contour and reduce the dimensionality of the descriptor (D. Zhang and Lu, 2002). The descriptor consists of a histogram of the Fourier transformation coefficients. The coefficients are calculated using the equation $a_k = \frac{1}{N} \sum_{n=0}^{N-1} r(n) e^{-2\pi jkn/N}$ where N is the number of points on the shape and j is the imaginary unit number.

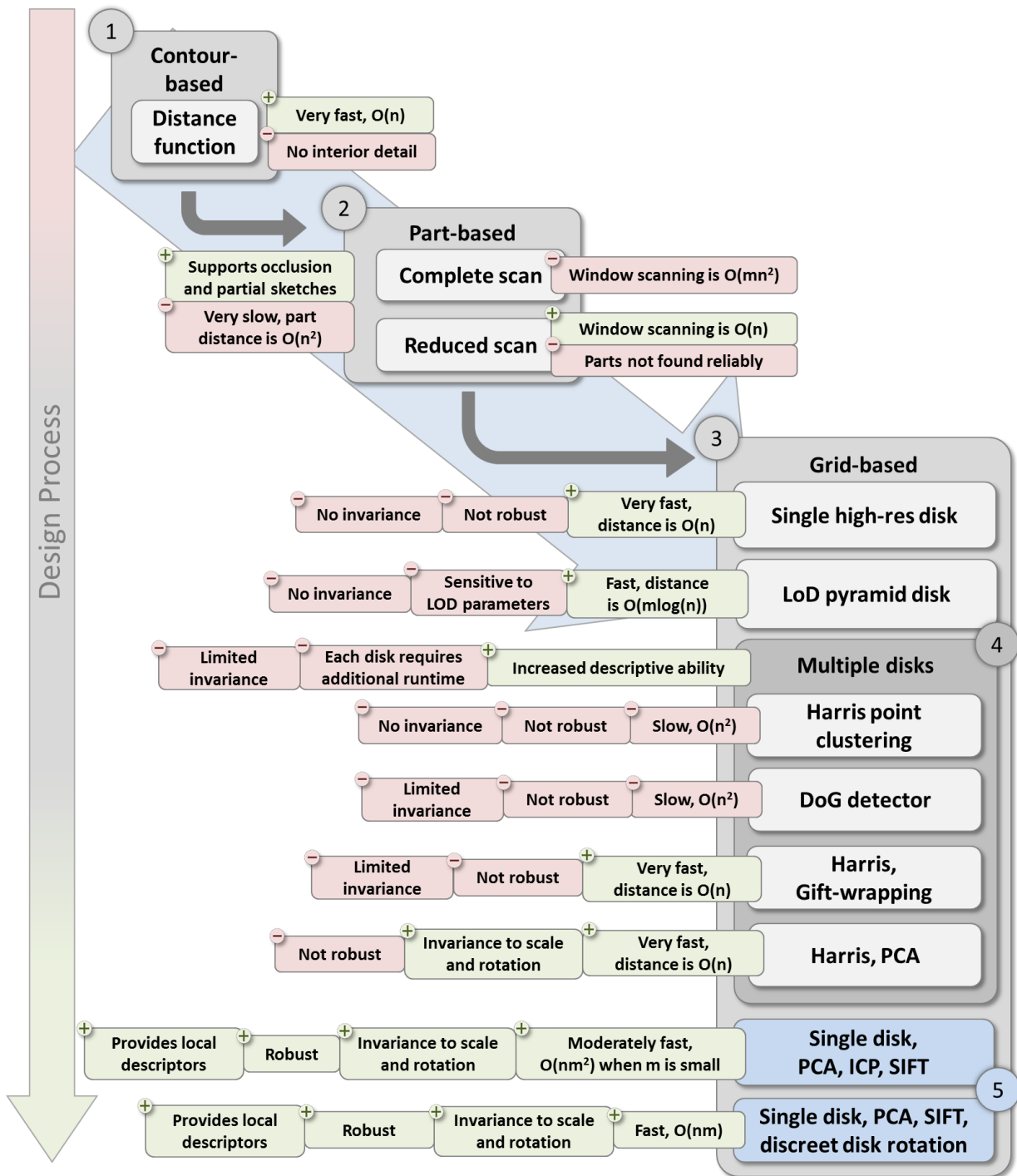


Figure 4.18: Overview of image descriptors investigated during the algorithm design process.

A query image is compared with a candidate image by comparing their descriptors. Contour-based descriptors can be compared with $O(n)$ time, where n is the number of bins each descriptor contains. The method is therefore very fast. It is not rotationally invariant; however, if n is kept small then the query image descriptor can be resampled for a number of rotations in order to minimise the distance between the descriptors and determine the rotation parameter. If m is the number of rotations considered, then the rotationally invariant variation of the contour-based approach therefore runs in $O(nm)$ time.

The problem with the overall approach is that it only considers the contour of the user's sketch. It cannot, for example, distinguish between a double-seat couch and a triple-seat couch when the edgels of both images are normalised into a unit square. The approach does not meet the requirements specified in 4.6.1, namely robust description and scale invariance.

4.6.3.2 Part-based Approach

The second approach is to extract internal details and contour-based features using the part-based object recognition approach proposed by Shotton *et al.* (2008) for structural region-based shape description. The descriptor provides a way of describing the various parts of an image. The parts are allowed to be disjoint and incomplete and contain internal detail. Each part is a cluster of *contour fragments*. A contour fragment is a cluster of edgels that are similar in location and appearance. The Orientated Chamfer Distance (OCD) is used to compare the appearance of the edgels in order to create contour fragments (see Section 3.3.3.3). The contour fragments are clustered together based on their location using the k-medoids clustering algorithm which uses $O(n^2)$ time. The total time required for creating a descriptor for each training image is therefore worse than $O(n^2m^2)$, where n is the number of edgels and m is the number of contour fragments. The approach does not require a descriptor to be created for each query image. Instead, the query image is searched for the expected parts from a training image. This is achieved by sampling the edgels from the query using sliding windows of various sizes. The edgels of the training window are then compared with the edgels of each training image part using the OCD metric. Shotton *et al.* trained a classifier using an evolutionary algorithm in order to determine if the parts found in the query image match the parts in the training image adequately. This is problematic because it requires a representative sample of sketches in order to train the classifier. Instead, a heuristic classifier was designed for this research.

Consider Equation (4.3) and Equation (4.4). P is the probability of a query image with M parts to be correctly associated with the parts of a candidate image. E_m is the OCD from the m^{th} query part to the m^{th} candidate part and E_m^* is the expected OCD. α is used to control the error tolerance when classifying parts as correct or incorrect. W is a classification parameter that controls the shape of the probability function. High values of W allow probabilities around 50% to be widely spread and probabilities close to 0% and 100% to be compact. The part-based image description approach becomes more robust as M increases.

$$P = \frac{1}{1 + e^{-H}} \quad , \text{ where} \quad (4.3)$$

$$H = \sum_{m=0}^M \frac{W}{M} (2S_m - 1) \quad \text{and} \quad S_m = \begin{cases} 1 & : E_m < (1 + \alpha)E_m^* \\ 0 & : \text{otherwise} \end{cases} \quad (4.4)$$

The greatest problem with applying the part-based approach for the Computer Vision Module is that the algorithm has very high computational requirements. This is because the OCD metric is $O(n^2)$ and the scaled sliding window search takes $O(mn^2)$ time. A variant of this approach was investigated in order to reduce the scanning requirement.

The gradient of a high-level scale space image is used in order to guide the search for a particular part in $O(n)$ time (see Section 3.3.2.2). The approach is similar to the gradient ascent method used for local maximisation problems. It assumes that each part is located at the centre of a dense area containing edges. It also assumes that each query part is near the relative expected location of the corresponding candidate part and that their relative scales are the same. Informal experimentation showed that this is not always the case. For example, if two legs of a table are close together then it cannot be assumed that there exists a single part between the two table legs. If the parts are identified separately in the training image then the parts need to be identified separately in the query image. This is why a complete search using a scalable sliding window is necessary (at least around the expected locations).

Figure 4.19 shows an example with 30 parts after scanning a sketch made of a bed during informal experimentation. Each part is annotated with the value of S_m . This example returns a high P value for the correct candidate image because a large M value was chosen and the parts were located using a complete search. A single comparison took at least one minute using the hardware discussed in Section 4.5.1. Decreasing the number of parts, clustering edgels, and using the $O(n)$ searching approach reduced this time down to 2 seconds per comparison. This is still slow and the approach begins to lose its effectiveness at this point.

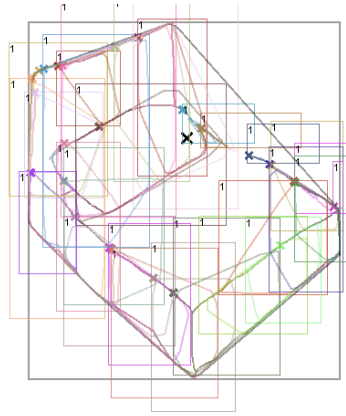


Figure 4.19: Part-based image description.

4.6.3.3 Grid-based Approaches

The third approach applies a simple grid-based descriptor for the image. An elliptical grid is rasterised over the axis aligned bounding rectangle of the edgels contained in the sketch or training image (see Section 3.3.3.2 and Section 4.6.2.2). The resulting *disk* is a global feature descriptor that illustrates the presence of edgels. Figure 4.20 (b) shows the disk for a user sketch shown in Figure 4.20 (a). If the disk is unrolled then it illustrates the edgel distribution of the image in polar coordinates as shown in Figure 4.20 (c).

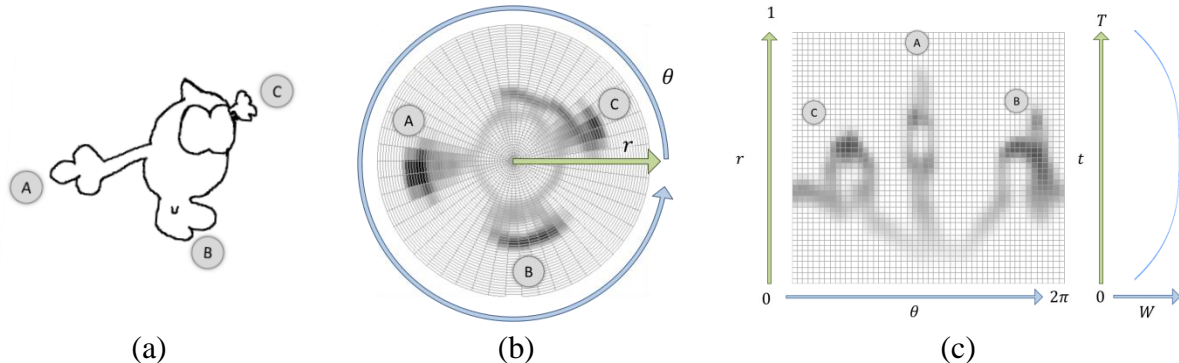


Figure 4.20: Describing a user sketch (a) using a grid-based approach (b-c).

Query images are compared to candidate images by comparing their disks. Each disk is expressed of the descriptor as a single $n = ts$ dimensional vector D where t is the number of tracks and s is the number of sectors. The distance between the two disks is taken to be the square of the Euclidean norm of the difference between the query descriptor and the candidate descriptor (Sajjanhar and Lu 1997). A single comparison therefore takes $O(n)$ time. The approach has low computational requirements but it has several limitations.

Its effectiveness depends on the resolution (descriptive ability) of the disk. If the resolution is too low then the disk cannot be used to distinguish between different images that appear

similar at a low resolution. If the resolution is too high then the disk cannot be used to measure the distance between images accurately if distortion, scaling or rotation is present. For example, if the part of the sketch in Figure 4.20 (a) marked “A” is sketched at a slightly higher position, then the resulting query disk has a greater squared Euclidean distance to the corresponding candidate.

Another limitation to the approach is that the descriptor is very sensitive to the location of the disk on the image. Moving the disk changes the tracks close to the centre significantly and it changes the outer tracks moderately. The changes are less severe near the middle tracks of the disk. This problem is addressed by applying a weight to each track so that the disks can be displaced slightly without causing a large deviation when measuring the distance between two disks. Informal experimentation showed that $W = -8 \cdot |r^3| + 1$ is a good weight function where $r = \frac{t}{T} - \frac{1}{2}$, t is the current track and T is the number of tracks. The approach required prior edgel normalisation because it is not invariant to rotation or scaling.

The problem relating to distortion-related sensitivity of the disk approach is connected to the use of a fixed resolution. This problem is addressed with a Level of Detail (LoD) pyramid disk design. The concept of a LoD disk is similar to the concept of the scale space of an image (see Section 3.3.2.2). The LoD disk contains a pyramid of disks starting with a high resolution disk at the base and ending with a low resolution disk at the apex. The comparison between two LoD disks is similar to normal disk comparison except that it includes how the scale affects the resulting distance measurement. This is achieved by starting at the apex disk and then progressing towards the base disk. If the distances between the disks decrease by moving down the pyramid then the algorithm continues to do so until an improvement less than β is measured. The distance between each query-candidate pair is divided by the dimensionality of the descriptors at the scale of the pair. This prevents low resolution disks from biasing the search. A single comparison runs in $O(m \log(n))$ time, where n is the dimensionality of base disk and m is the number of scale space layers. Informal experiments showed that this approach provides improved recognition accuracy. However, the robustness of the approach is sensitive to the LoD parameters chosen, such as β , n and m . The method improves on the simple disk-based approach but it is not invariant to rotation.

4.6.3.4 Sampling Multiple Disks

The fourth approach attempts to increase the descriptive ability of the Computer Vision Module by sampling multiple disks from the image. Measuring multiple disks involves

determining the location, orientation and scale of each disk consistently. Several variants of the approach were attempted in order to rasterise multiple disks from training images and query sketches.

The first variant attempts to use the Harris corners of the image in order to estimate the location of each of the m disks (see Section 3.3.2.1). The k-medoids clustering algorithm is used to identify $k = m$ clusters. The centre of each cluster is used to position a disk (see Figure 4.21 (a)). Query images and candidate images are compared by measuring descriptor distances between the best matching disk pairs. The main disadvantage of this approach is that it relies heavily on the placement of the disks. Informal experimentation shows that the k-medoids clustering variant of the approach does not produce consistent disks for images that are slightly rotated or distorted.

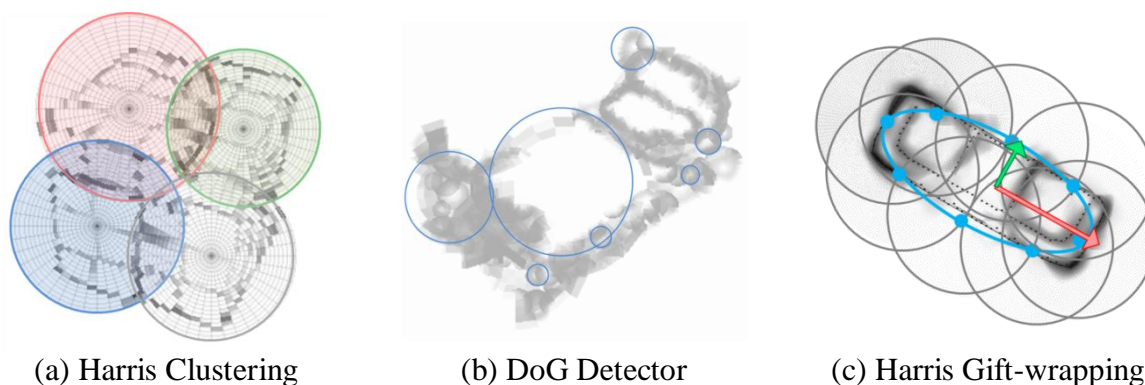


Figure 4.21: Approaches for sampling multiple disks from an image.

The second variant attempts to use the Difference of Gaussian (DoG) local feature detector in order to place and scale the disks for the image. The scale space is constructed on a high resolution edge map of the image. The m largest blobs are used to place the disks and they are orientated along the gradient of the blurred edge map (see Figure 4.21 (b)). The use of the DoG local feature detector introduces several problems. It does not always return a complete set of disk locations and sizes. The approach provides limited invariance to rotation and distortion and it is costly to construct the scale space for a high resolution edge map.

The third and fourth variants attempt to place the individual disks at specific locations so that they cover the entire image space. This is done by normalising the edgels of each image by removing its translation, scaling and rotation. The disks are placed around an ellipse at pre-determined positions (see Figure 4.21 (c)). The problem with using this approach is that the transformation parameters required for normalisation need to be calculated. The third approach achieves this by first calculating the boundary around the Harris corner points using

the Gift wrapping algorithm in $O(nm)$ time (Jarvis 1973). The boundary points are then clustered into two clusters which are connected by a line segment. This line segment is taken to be the horizontal axis of the bounding rectangle of the image. The vertical axis is taken to be perpendicular to the horizontal axis such that a right hand coordinate system results. The coordinate system is then used to normalise the query image the training images. The problem with the clustering approach is that the centres of the two clusters do not always provide an accurate representation of the image's coordinate system. The fourth variant of this approach applies Principal Component Analysis instead to provide more accurate and consistent coordinate systems for the query images and training images (see Section 3.3.4).

Using multiple disks to represent an image does increase the descriptive ability of the Computer Vision Module. The main limitation of the approach is that its effectiveness is influenced by the location, orientation and scaling of the individual disks. Image distortion and rotation is therefore problematic for this approach. Another disadvantage of using multiple disks is that the time required to compare each image increases linearly with each disk added to the descriptor. The Computer Vision Module requires a robust image description method while being invariant to changes in rotation, scale and translation. The final approach taken takes the lessons learnt from the previous approaches discussed so far and provides an approach for describing objects in user sketches.

4.6.3.5 Combined Approach

The image description approach used by the Computer Vision Module should be able to describe and compare images reliably and within a reasonable amount of time (a few seconds or less with the hardware discussed in Section 4.5). The requirements identified in Section 4.6.1 states that the approach should be invariant to changes in the translation, scaling and rotation of the image and it should provide local feature correspondences so that rigid body pose estimation can be performed. This poses five main problems:

1. The objects in the sketch should be separated.
2. The description and comparison of each object should be invariant to translation, scaling and translation.
3. The method should be able to describe simple objects like symbols and top-down views of props.
4. The methods should be able to describe detailed objects like 3D views of props.
5. The description of detailed 3D props should include local features.

These problems are solved by using a combination of image description approaches. The approach involves two stages, namely the description stage and the comparison stage.

The *description stage* involves creating a description for each object in the sketch. Each prop and symbol is sketched on separate layers or isolated within bounding rectangles (see Section 4.5.6.1). Figure 4.22 provides a pseudocode illustration of the description stage.

```

Step 1. Normalise the edgels with PCA into a unit square.
Step 2. Sample low resolution disk  $L$ .
Step 3. Sample high resolution disk  $H$ .
Step 4. Find SIFT features  $\{F_i\}$  of  $H$ .
Step 5. Group the SIFT features  $\{F_i\}$  into  $M$  bags  $\{B_m\}$ .
Step 6. Construct a kd-tree from  $\{B_m\}$ .

```

Figure 4.22: Pseudocode for describing objects in sketches.

The edgels of the user's sketch are normalised to a unit square after removing the translation and rotation of the image using Principal Component Analysis (PCA) (see Section 3.3.4). The principal components are two perpendicular vectors that align with the principal axis of the edgels' local coordinate frame. The magnitude of each principal component is proportional to the variance of the edgels along the corresponding principal axis. A bounding rectangle is placed around the local coordinate frame of the edgels. The width and height of the rectangle is set to a constant multiple of the length of the corresponding principal components. The angle from the x-axis of the parent coordinate frame to the first principal axis represents the rotation of the image. The vector from the parent coordinate frame's origin to the origin of the local coordinate frame represents the translation of the image. Normalisation involves removing the translation, rotation and scaling of the image (in this order) so that the edgels of the image fits into a unit square. The principal components identified by using PCA depend on the distribution of edgels. If the resulting local coordinate frame is a left-handed coordinate frame then the second principal component is mirrored. This ensures that all the coordinate frames are right-handed. It is important to ensure that the images are normalised consistently; otherwise the overall method cannot be invariant to scaling or rotation. The overall description method is therefore scale and translation invariant. Rotational invariance is achieved during the comparison stage.

The object in the image is described using a single disk approach because it allows for the fastest queries (see Section 4.6.3.3). Two resolutions of the disk are used. A lower resolution disk is used for recognising images that are simpler to represent. These include the top-down views of the props and symbols sketched in the floor plan editor and emotions sketched in the

sketched editor. A higher resolution disk is used for describing sketched props that are illustrated from a 3D viewpoint. Local SIFT-based features are extracted from the high resolution disk using the DoG local feature detector (see Section 3.3.2.2 and Section 3.3.3.1). The SIFT features are then grouped into local feature bags which are indexed using a kd-tree (see Section 3.4.1). The local features are used to determine the points from the query image that correspond with the points on the candidate image.

The *comparison stage* involves measuring the dissimilarity between two image descriptions. Figure 4.23 provides a pseudocode illustration of how the dissimilarity measure is calculated.

```

IF a low resolution disk is used THEN DO
    Step 1. Determine the rotation of the low resolution query disk.
    Step 2. RETURN the SSE between the low resolution query disk and
           the candidate disk.
ELSE
    Step 1. Determine the rotation of the high resolution query
           disk.
    Step 2. Set  $E = 0$ 
    Step 3. FOR EACH SIFT feature  $F$  on the query disk  $\{F_i\}$  DO
            Step a. Find the closest bag  $B^*$  of SIFT features from
                   the candidate image using the candidate NN
                   kd-tree.
            Step b. Find the SIFT feature  $F^*$  in  $B^*$  that is the
                   most similar to  $F$ .
            Step c.  $E \leftarrow E + \text{Distance}(F, F^*)$ 
            END FOR
    Step 4. RETURN  $\frac{E}{N}$ , where  $N$  is the number of features in  $\{F_i\}$ 
END IF

```

Figure 4.23: Pseudocode for measuring the dissimilarity of objects in sketches.

The first step in comparing low resolution disks and high resolution disks is to determine the rotation of the object in the query image. The angle of rotation θ between the query disk and a candidate disk is required in order to measure the distance between the descriptors. This angle is then converted into an integer index $j = S\theta/2\pi$ where j indicates how many sectors the candidate disk should be rotated before comparing it to the query disk, and S is the total number of sectors for each disk. Two approaches for calculating θ have been found to work well during the design process.

The first approach involves constructing a second disk of a lower resolution with S' sectors for the query image and the training image. The lower resolution candidate disk is iteratively rotated by $2\pi/S'$, and compared with the candidate image. This approach works well when the rotation of the image does not have to be precisely estimated and short runtimes are required.

The second approach for determining the rotation of the query image is to use the Iterative Closest Point (ICP) registration algorithm (see Section 3.5.3.2). The performance of the ICP registration algorithm is improved by using a kd-tree spatial index on a reduced set of points (Bentley 1975). The final rotation angle is calculated using Equation (4.5) from the rotation component of the resulting registration transformation where ε_a and ε_b are numerical error values and $[a, -b]$ is the top row of the rotation matrix R returned by the ICP registration algorithm. This approach works well if a precise estimate is required and a longer run time is acceptable. The disk is rotated by integer index of $j = S\theta/2\pi$ sectors.

$$\theta = \frac{1}{2}(\cos^{-1}(a) + \sin^{-1}(b)) \quad , \text{ where } R = \begin{bmatrix} a & -b \\ b + \varepsilon_b & a + \varepsilon_a \end{bmatrix} \quad (4.5)$$

The distance from the query disk to the candidate disk is determined as follows. If a low resolution disk is used then the distance is measured as the square of the Euclidean norm of the difference between the query disk and the candidate disk (Sum of Squared Error (SSE)). If a high resolution disk is used then the distance is measured using the local SIFT-based features. The nearest feature bag B_i^* of the candidate grid for each local feature F_i of the query grid is determined using NN kd-trees in $O(\log(n))$ time. The best local feature F_i^* in the feature bag B_i^* is then mapped to F_i . The distance between the disks is calculated as the average dissimilarity between the feature pairs $\{(F_i, F_i^*)\}$.

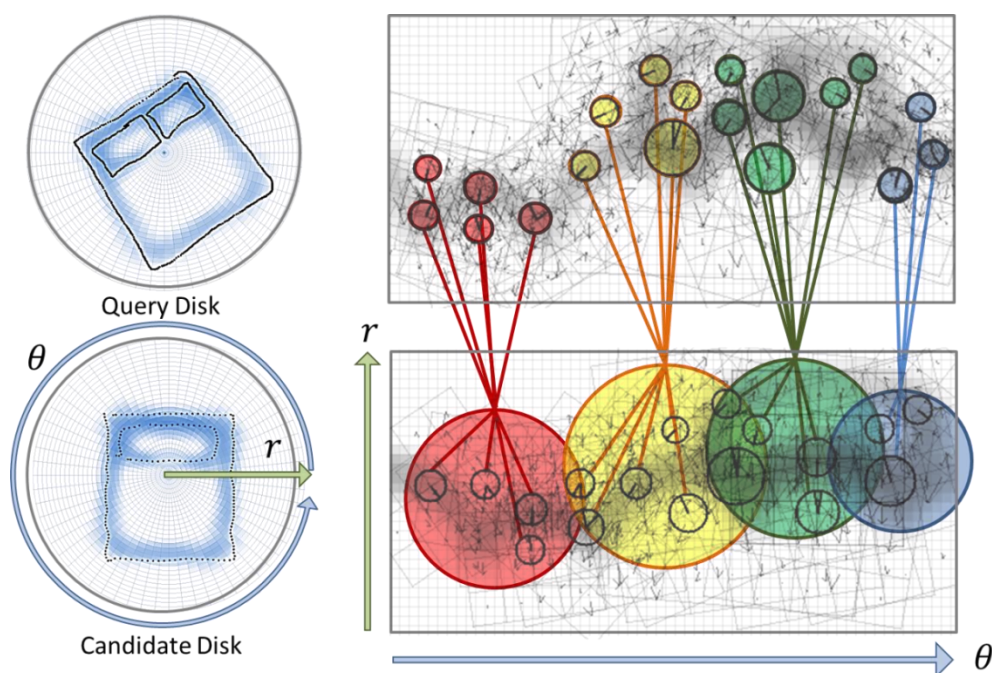


Figure 4.24: Comparing two disks with four SIFT feature bags.

Figure 4.24 provides an example showing how the feature pairs are determined. The scale and direction of each blob is illustrated by a black circle with a direction indicator. The grids around the blobs are the SIFT-keys. Each blob of the query disk is mapped to a single feature bag of the candidate disk. The blob is then mapped to the feature in the feature bag that has the most similar SIFT-key.

Local feature correspondences are required in order to perform pose estimation. The point correspondences are also required to be in the original space of the query sketch or training image. A dictionary mapping is therefore used to map the normalised local feature locations to their original locations.

4.6.3.6 Discussion

This section provided a detailed discussion of all the approaches investigated during the design process for describing user sketches. A combined approach was discussed which is invariant to translation, scaling and rotation transformations, and determines local feature correspondences. The combined approach therefore fulfils the sketch description requirements identified in Section 4.6.1.

4.6.4 Recognising Objects from Sketches

The Computer Vision Module is designed to follow the general object recognition process outlined in Section 3.2 (see Figure 4.25). The approach discussed in Section 4.6.3.5 is used during the training phase and the recognition phase for detecting and describing features.

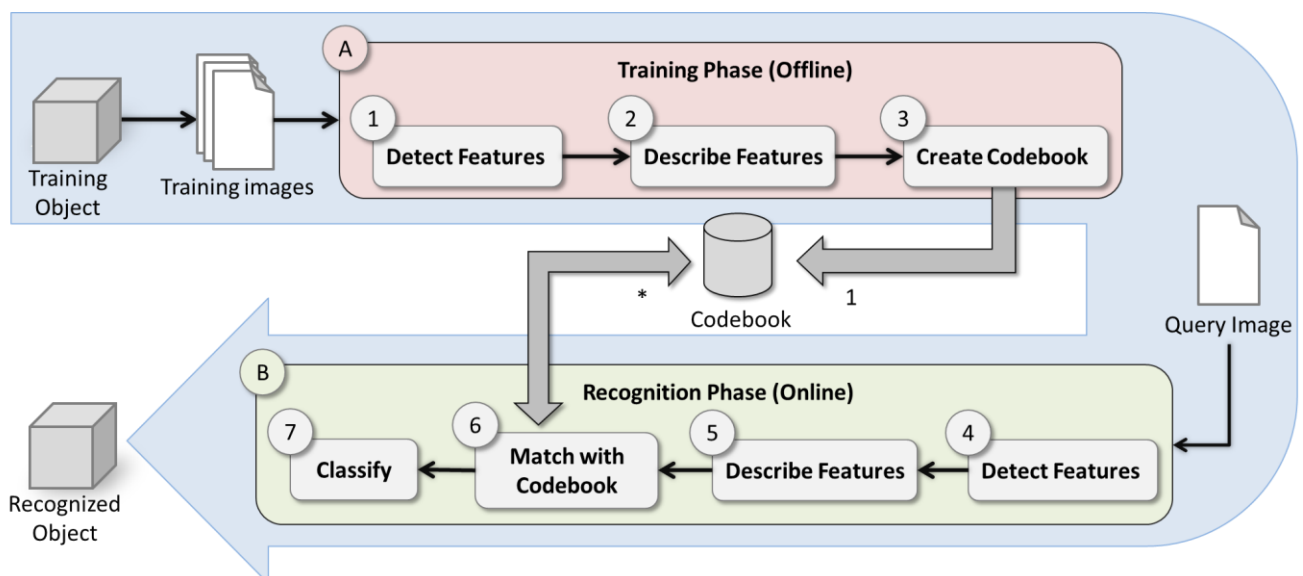


Figure 4.25: The general process followed by object recognition algorithms (Pellegrini 2007).

This section discusses how the training images are sampled and described in order to populate the codebook. It also discusses how the objects sketched in floor plans and storyboards are classified in order to support object recognition.

4.6.4.1 Training Phase

The Computer Vision Module is required to recognise 2D objects, such as top-down views of the props sketched using the floor plan editor and symbols used for staging and illustrating the facial expressions of each character. The Computer Vision Module is also required to recognise 3D objects from 2D sketches in order to interpret contents of the storyboard panels sketched by the user. The necessary objects need to be stored and associated with 2D images so that they can be used for recognition by the Computer Vision Module. The environment data collection contains a set of codebooks that store this data.

The training images used for the floor plan symbols and character emotions were created manually. Each symbol is associated with one or more training images as shown in Figure 4.26. Additional emotion symbols can be added to the environmental data collection (see Section 4.4).



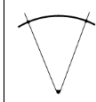
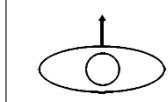


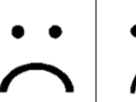
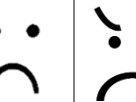


Shot			Character	Happy		Sad		Angry	
									

Figure 4.26: The training images used for recognising symbols.

The training images used for the props are generated automatically using the Computer Vision Module. The props are rendered with a white background. The light source is placed in the centre of the viewing area behind the image plane. This produces images that clearly show important edges that outline the shape and interior of the 3D model of each prop. Two codebooks are generated for each prop. The first codebook contains orthographic top down images of the prop's 3D model. The second codebook contains the images sampled from the various perspective views of the prop's 3D model.

The perspective views are selected in a manner similar to that of Shin and Igarashi (2007). Shin and Igarashi sampled 16 reference views for each 3D model. Informal experimentation determined, however, that using 16 views is insufficient for recognising 3D objects using the image descriptor implemented in the Computer Vision Module. Instead, $24 \times 4 = 96$ reference views are sampled. Four vertical angles and 24 horizontal angles are used for each

vertical angle as illustrated in Figure 4.27. It is assumed that the props will not be sketched from the button up. The large codebook size provides the training data required for effective pose recognition (see Section 3.5). If too little training data is provided then the correlations returned by the image descriptor will be inaccurate and the pose estimation algorithm will yield poor results.

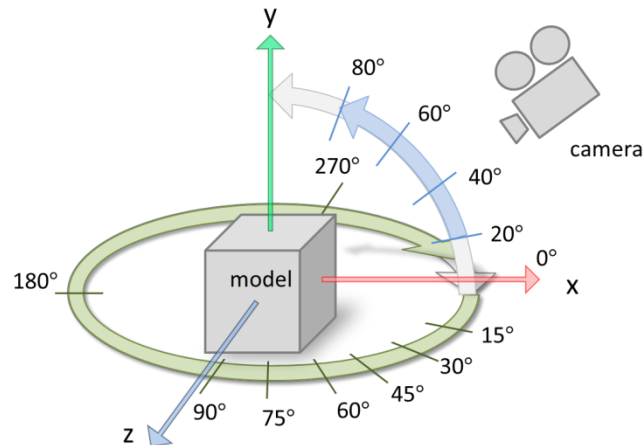


Figure 4.27: The 3D model sampling method.

The next step is the training phase. It involves describing each training image in order to populate the codebook. The codebook contains a collection of objects that are each associated with image description data. This includes the data necessary for storing disks, kd-trees, SIFT features and point dictionaries. The positions of the local features are 2D. Some pose estimation methods such as POSIT require 3D point correspondences (see Section 3.5). The 3D data for the training images is acquired using the Computer Graphics Module. Each 3D model is rendered on the front buffer in graphics memory. The buffer contains a (R, G, B, A, ρ) tuple for each pixel (x, y) where the (R, G, B, A) is the colour component and ρ is the pseudo-depth value for the pixel. The location of the corresponding 3D point in space is calculated by un-projecting (x, y, ρ) into the model-view coordinate system. The corresponding 3D point of each local feature is stored in the codebook. The codebook also contains a model-view matrix for each 3D sample taken. The model-view matrix specifies the camera transformation used when the sample was taken.

4.6.4.2 Recognition Phase

The first step performed during the recognition phase is to describe the sketch using the approach discussed in Section 4.6.3.5. The sketch is classified by comparing this description with the descriptions of candidate objects from the codebook. The orthographic codebook is

used if the user is sketching symbols or top-down views of props. Otherwise, if the user is sketching props in perspective then the arbitrary-view codebook is used.

The object sketched by the user is classified using the k -Nearest Neighbours (k -NN) classifier (see Section 3.4.2.5). The k -NN classifier allows the Computer Vision Module to classify sketches using the dissimilarity measure provided by the image description approach discussed in Section 4.6.3.5.

```

Step 1. Measure the distance  $d_i$  from the query description  $Q$  to each
        candidate description  $C_i$  in the codebook. Add the distance-
        candidate pair  $(d_i, C_i)$  to an ordered dictionary  $D$  such that  $D$ 
        contains a maximum of  $k$  entries.
Step 2. Let  $N$  be the label of the 3D model that the occurs most
        frequently in  $D$ .
Step 3. IF  $N$  occurs only once in  $D$  THEN DO
        RETURN the candidate  $C_i^*$  with the shortest distance  $d_i^*$ .
        ELSE
        RETURN the candidate  $C_i^*$  labelled  $N$  with the shortest
        distance  $d_i^*$ .
END IF

```

Figure 4.28: Pseudocode for the k -NN classifier used to classify query images.

Figure 4.28 shows the pseudocode for the k -NN classifier. An ordered dictionary containing (distance, candidate) pairs is maintained with a maximum of k entries. The class label N that occurs the most in the dictionary (and more than once) is considered to represent the most probable object sketched by the user. The candidate C_i^* labelled N with the smallest d_i^* is associated with the user's sketch.

Informal experimentation determined that $k = 3$ is suitable for classifying floor plan symbols and character emotions (each symbol class has three or less instances). $k = 100$ was found to be a suitable value for classifying perspective sketches of props (each prop class has 96 instances). The top $n = 10$ alternative candidates are presented to the user if it is necessary to manually choose a different prop.

4.6.4.3 Discussion

This section discussed how the objects sketched by the user are recognised. The process involves sampling images of the each object that can be sketched and then generating a codebook that contains descriptors for each training image. The k -NN classifier is used to determine which object is most likely represented by the user's sketch.

The codebooks containing the 3D views of each prop are augmented with 3D point data so that each 2D local feature on the training image has a corresponding 3D point in the space of the 3D model. This data is important for estimating the position and orientation for rigid bodies like props. The next section discusses how the Computer Vision Module is designed to estimate the pose of each prop and character in the user sketches.

4.6.5 Pose Estimation

The Computer Vision Module is also required to place each prop and character into the 3D virtual environment once they have been recognised from the user's sketch. The process of determining location and orientation of each prop and character is known as *pose estimation* (see Section 3.5 and Section 3.6). This subsection discusses two cases for estimating the pose of a prop or character. The first case is when the camera is unknown and the user sketches props on a blank sketch using the Sketch Editor (see Section 4.5.6.1). The second case is when the camera is known and the user sketches props and characters over the rendered image of the known 3D virtual environment.

4.6.5.1 Estimating the Pose of Rigid Bodies with an Unknown Camera

Each prop is represented by a 3D model that is placed and orientated on the set. The 3D model is a rigid body with no deformable or moveable parts. The location and orientation of an object is also known as its *pose*. The floor plan provides the pose of each prop relative to the set. When the user sketches the prop using the Sketch Editor, its pose is relative to the view the prop is sketched from. The problem is that the camera's location and orientation are unknown. The Sketch Editor is responsible for automatically interpreting the user's sketch in order to determine the camera parameters for placing the shot in the floor plan as shown in Figure 4.29 (a).

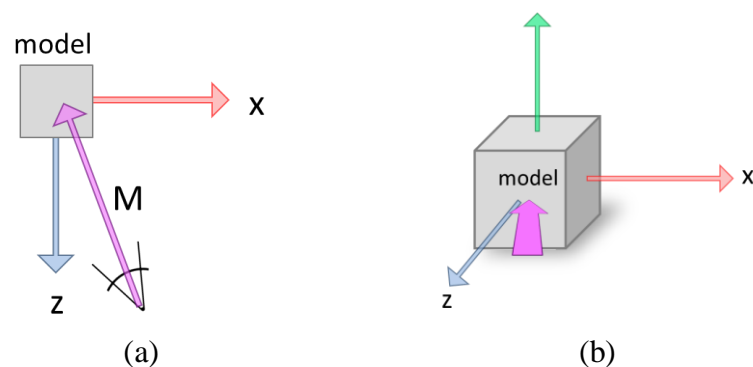


Figure 4.29: The pose estimation problem as viewed from (a) top-down, and (b) the shot sketched by the user.

The Computer Vision Module estimates these parameters by calculating the model-view matrix for each prop the user sketches. These model-view matrices are then combined in order to estimate the final camera for the sketched shot (see Section 4.6.6). The resulting model-view matrix is then used to place and orientate the camera as shown in Figure 4.29 (b).

The model-view matrix for a prop is estimated from its sketch as follows. The prop is assumed to be positioned at the origin with its default orientation. The model-view matrix is estimated using a combination of the pose recognition approach and iterative pose estimation (see Section 3.5) (Shin and Igarashi 2007; Lee and Funkhouser 2008). The pose recognition approach utilises the image descriptor selected in Section 4.6.2. The codebook is searched for a candidate sample that best matches the prop illustrated in the user's sketch (see Section 4.6.4.2). The model-view matrix contained in the codebook for the matching candidate prop is used to estimate the rotation for the camera. The POSIT iterative pose estimation algorithm is then used to determine the camera's translation and refine the camera's rotation (Dementhon and Davis 1995). The POSIT algorithm requires a set of correspondences between 2D image points and 3D model points (see Section 3.5.3.1). The image descriptor used by the Computer Vision Module is used to determine the required correspondences (see Section 4.6.3.5).

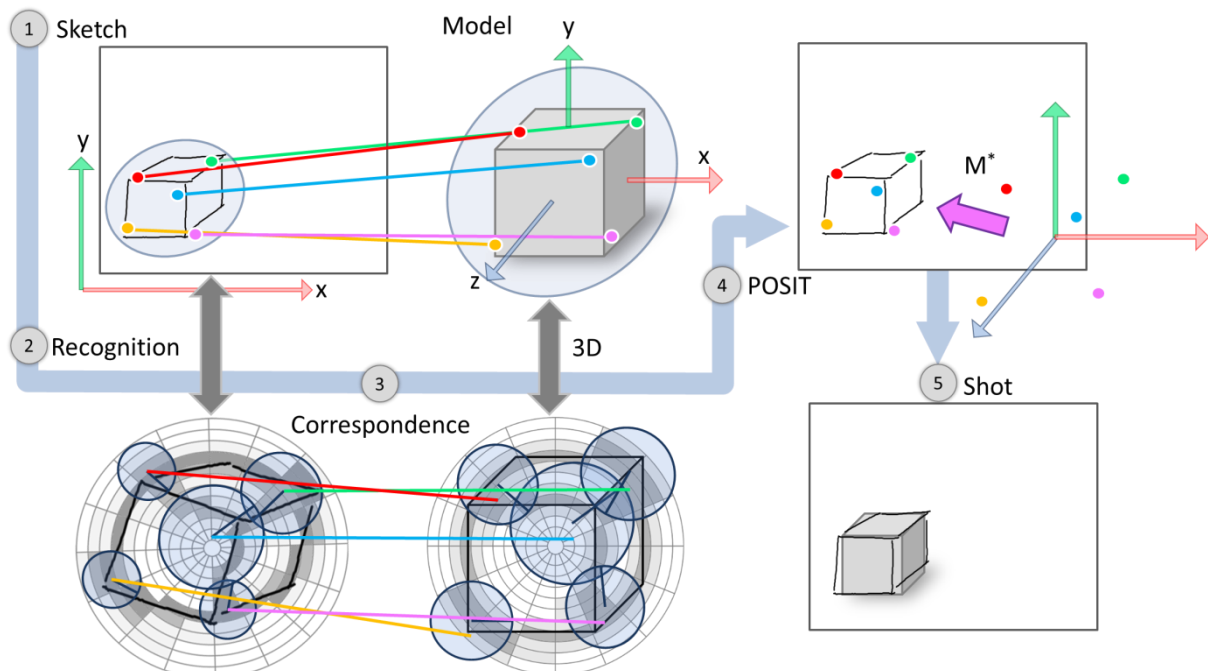


Figure 4.30: The process of estimating the pose of a rigid body sketched by the user.

Figure 4.30 illustrates the pose estimation process implemented by the Computer Vision Module. The first step requires the user to sketch each prop in the camera frame as it should

appear in the final shot. Object recognition is then performed in the second step in order to find the best sample from the codebook that matches the sketched prop. Correspondences between the features in the query image and features in the best candidate sample are determined during the third step. The required $2D \leftrightarrow 3D$ correspondences are obtained by mapping the original points from the sketch for each respective query feature to the locations of the 3D features stored in the prop's codebook.

The dictionary stored in the prop's codebook is used to obtain the corresponding 3D points of the model for each SIFT feature. The POSIT algorithm is used in the fourth step to estimate the translation component of the model-view matrix and refine the orientation component. It takes a set of 2D image points from the user's sketch and a set of corresponding 3D model points as input. The POSIT algorithm yields better results if it is provided with a rough estimate of the model's rotation (Dementhon and Davis 1995). The 3D points are therefore rotated around the origin using the rotation component of the model-view matrix obtained from the codebook. This is a necessary step because the quality of the pose returned by POSIT depends heavily on the quality of the $2D \leftrightarrow 3D$ correlations with which it is provided. It is not unusual for the correlations to be inaccurate, because they are based on low resolution local feature descriptions (SIFT) and distorted input. Every imperfection in the user's sketch worsens the orientation estimation returned by the POSIT algorithm. This is why the codebook model-view matrix is used for estimating the overall orientation of the prop's pose and the POSIT orientation is used for minor orientation refinements. The final step is to combine the estimated model-view matrix for each prop in order to estimate the final shot. Section 4.6.6 will discuss this step in greater detail.

4.6.5.2 Estimating the Pose of Articulated Figures and Rigid Bodies with a Known Camera

If the camera's model-view matrix is known then the pose of each character and prop can be estimated by un-projecting the relevant points from the sketch into the world coordinate system using the Computer Graphics Module. Posing characters also involves determining the orientation of the individual bones in order to estimate the character's posture for the sketch. Figure 4.31 illustrates the seven coordinate systems involved when un-projecting the user's sketches into the 3D environment so that characters and props can be posed.

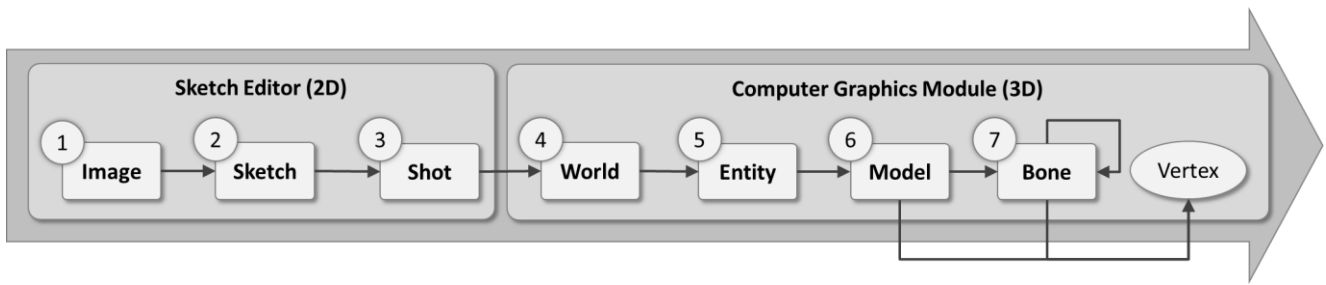


Figure 4.31: The coordinate systems involved for un-projecting an image point.

The Sketch Editor is responsible for un-projecting the 2D position of the mouse cursor (the stylus/pen) into the image space to which the 3D virtual environment is rendered. This image space is represented by the “shot” coordinate system and it is different from the “image” coordinate system to which the GUI is rendered. The “sketch” coordinate system is intermediate to the “Image” coordinate system and the “Shot” coordinate system. It allows the user to pan and scale the current view of the shot. The Computer Graphics Module is responsible for un-projecting 2D points from the “shot” coordinate system to the “model” coordinate system or a leaf “bone” coordinate system.

The position of an entity on the set is calculated by un-projecting a predetermined anchor point from the user’s sketch. The anchor point for a prop is located at the middle of the lower edge of the bounding rectangle of the user’s sketch of the prop (see Figure 4.32 (a)). The anchor point for a character is located halfway between its feet (see Figure 4.32 (b)). Prop anchor points can be placed on the floor or on top of other props. Character anchor points can only be placed on the floor of the set.

If the shot coordinates of the point are represented by $\bar{x} = (x, y, \rho)$ where ρ is the point’s pseudo-depth and $\bar{X} = (X, Y, Z) = P^{-1}(\bar{x})$ is a function that un-projects \bar{x} to \bar{X} in world coordinates, then a ray is constructed from $\bar{X}_0 = P^{-1}(x, y, 0)$ to $\bar{X}_1 = P^{-1}(x, y, 1)$. The ray is intersected with the floor and the top-most surface of the 3D bounding box of each prop. The intersection point with the highest Y coordinate is taken as the 3D correspondence of the 2D anchor point. The entity is then placed on the 3D anchor point. If the entity is a prop then the methods discussed in the previous section are used to determine its orientation. If the entity is a character then the character is assumed to face the camera. If the “flip” option is enabled then the character faces away from the camera. The location and orientation of the character or prop is now known. In other words, the matrix for the “entity” coordinate system has been estimated. If the entity is a character then it is also necessary to estimate the orientations of the individual bones in order to pose the character from the user’s sketch.

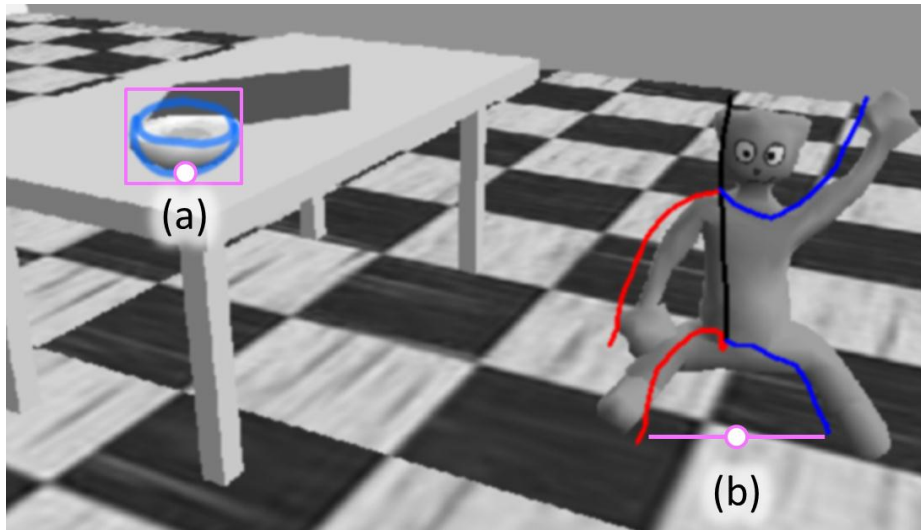


Figure 4.32: Posing props and characters with a known camera.

Figure 4.32 (b) illustrates how the user sketches the pose of a character (see Section 4.5.6.2). The pose of a character is indicated with a red, black and blue coloured stick figure. The red side represents the character's right side and the blue side represents the character's left side. The stick figure is analysed using an algorithm designed from previously published work.⁵ The algorithm accepts a colour coded stick figure and the corresponding armature. The armature is then recursively configured in order to match the pose illustrated by the stick figure while remaining within bone rotational constraints. The stick figure is represented using a tree structure. The tree is automatically constructed as the user sketches. The original algorithm requires user to manually colour code the individual bones in order to resolve ambiguities. In this research, the bones are automatically colour coded. This is because the current design is limited to humanoid characters. It simplifies the sketching process because the user no longer has to set up the colour coding for the armature.

The stick figure is automatically colour coded as follows. The vertex having a degree of four is assumed to be the torso. The shortest edge is assumed to be the head and it is coloured black. The leftmost edge is assumed to be the right arm and it is coloured red. The rightmost edge is assumed to be the left arm and it is coloured blue. The remaining edge is assumed to be the back and it is coloured black. The vertex at the end of the back is assumed to be the pelvis. The leftmost edge from the pelvis is assumed to be the right leg and it is coloured red.

⁵ This research on sketch-based articulated figure animation was published in the following paper: Matthews, T. and Vogts, D. (2011): A sketch-based articulated figure animation tool. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT)* The Pavilion Conference Centre, V&A Waterfront, Cape Town, South Africa.

The rightmost edge from the pelvis is assumed to be the left leg and it is coloured blue. If the “flip” mode is enabled then the red and blue colours are exchanged.

If there is a problem with finding the required edges or vertices then the sketch is taken to be invalid and the algorithm terminates. Otherwise, if the colour coding was successful then the algorithm continues to find a single isomorphic mapping between the sketched stick figure and the armature of the character. The recursive posing algorithm then traverses the strokes of the stick figure and orientates the associated bone chains by taking bone foreshortening into account (Matthews and Vogts 2011).

4.6.5.3 Discussion

This subsection discussed how the pose of each prop is estimated from a user’s sketch. If the camera is known then the prop is correctly placed into the 3D virtual environment. If the camera is unknown then the pose of each of the props is used to estimate the camera. This is achieved by consolidating the model-view matrices of the props while maintaining the spatial relationships between them. The consolidated model-view matrix is used to estimate the camera for the shot sketched by the user. The following section discusses how this is achieved.

4.6.6 Camera Estimation

The Computer Vision Module is designed to be able to estimate the camera for a shot sketched using the Sketch Editor. This is achieved by estimating the pose of each individual prop for an unknown camera using the method discussed in Section 4.6.5. The location and orientation of each prop on the floor plan is then combined with the information gathered from the pose estimation algorithm. Figure 4.33 illustrates the camera estimation problem using an example.

The example shows a floor plan containing a table with two chairs and a sketch with the corresponding layers (see Section 4.5.6.1). Let $P = \{P_n : n \in \mathbb{Z}_0^{N-1}\}$ be the set containing the N props of the floor plan. Let $\{L_n : n \in \mathbb{Z}_0^{N-1}\}$ be the set containing the corresponding layers for each prop. The user is required to sketch the props in the correct colour so that the prop-layer correspondences are known. Let $N_n \in M_{4 \times 4}$ be the transformation matrix representing the location and orientation of P_n on the floor plan.

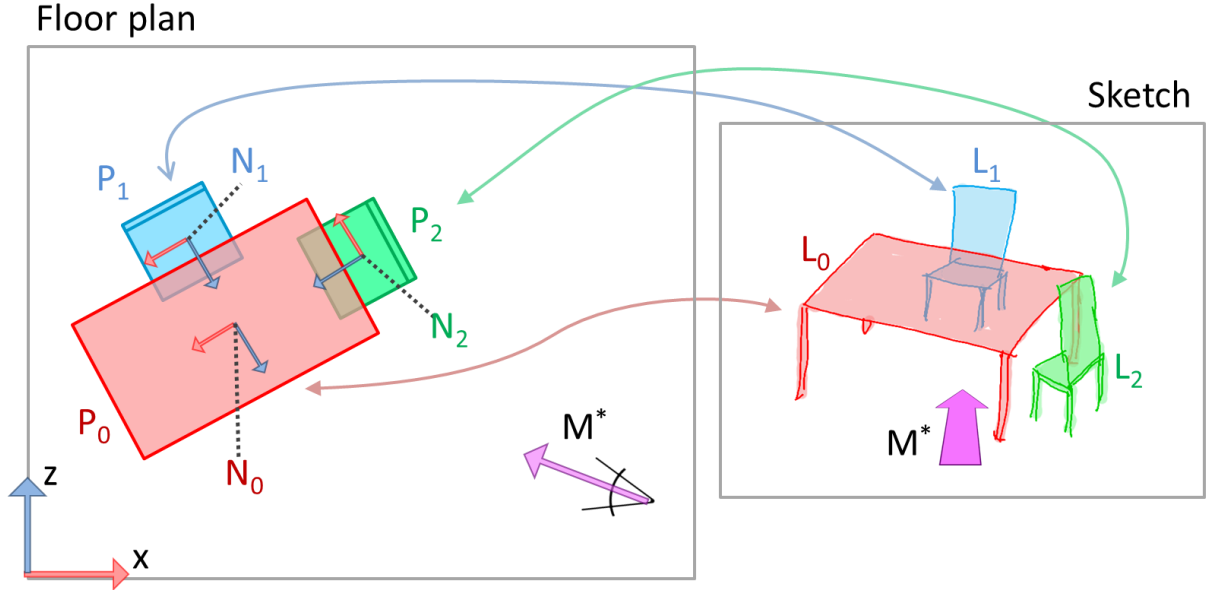


Figure 4.33: The camera estimation problem.

Each layer L_n is associated with a set of K candidate poses $\{(P_n, Q_{n,k}): n \in \mathbb{Z}_0^{N-1}, k \in \mathbb{Z}_0^{K-1}, Q_{n,k} \in M_{4 \times 4}\}$. If P_n is placed at the origin with its default orientation ($N_n = I_4$) then $Q_{n,k}$ is the k^{th} candidate model-view matrix that transforms the prop so that it is projected onto the user's sketch when rendered. However, P_n is not necessarily placed at the origin with its default orientation. It is therefore necessary to express the model-view matrix of the camera $M_{n,k} \in M_{4 \times 4}$ for the n^{th} prop while including its location and orientation on the floor plan. $M_{n,k}$ is determined as follows.

Let V be a vertex of the 3D model of prop P_n at $\bar{x} \in \mathbb{R}^4$. The position of V on the floor plan is $\bar{x}_f = N_n \bar{x}$. The 3D space of the floor plan is the same as the 3D space of the 3D virtual environment where the shot is taken. This implies that the world position of V is \bar{x}_f . The problem involves positioning V so that it is projected onto the user's sketch when the prop is rendered. This position \bar{x}_c is calculated by removing the prop's floor plan transformation from V and then applying the estimated pose: $\bar{x}_c = Q_{n,k} N_n^{-1} \bar{x}_f$. The model-view matrix of the camera is therefore $M_{k,n} = Q_{n,k} N_n^{-1}$ for the n^{th} prop and the k^{th} candidate match.

The camera estimation problem involves selecting the best model-view matrix for each prop so that they can be consolidated into a single model-view matrix. The model view matrix for each prop represents the displacement and orientation of the same camera. These matrices differ slightly due to computer vision error. Recognising the wrong poses or obtaining poor

correlations causes the coordinate frames of the model-view matrices $M_{k,n}$ to misalign. It is important to select the best candidate pose for each prop in order to minimise this error.

$$\min_{\bar{k} \in \mathbb{Z}^N} E(k_0, k_1, \dots, k_{N-1}) = \sum_{n=0}^{N-1} \sum_{\substack{m=0 \\ m \neq n}}^{N-1} d(M_{n,k_n}, M_{m,k_m}, P_n, P_m) \quad (4.6)$$

Equation (4.6) expresses the camera estimation problem as an integer minimisation problem. The problem involves finding a vector $\bar{k} = (k_0, k_1, \dots, k_{N-1})$ where $k_n \in \mathbb{Z}_0^{K-1}$ is the index of the best candidate pose for prop P_n that minimises E . $d(A, B, P)$ is a function for measuring the misalignment of two coordinate frames represented by the transformation matrices A and B for a particular prop P . The misalignment of A and B is measured for a prop using bounded coordinate systems (Huang *et al.* 2009). The basis vectors $\bar{\mathbf{u}}_i^A, \bar{\mathbf{u}}_i^B \in \mathbb{R}^4$ of each coordinate system are rescaled to match the dimensions of the 3D models of the respective props. If the camera coordinate frames are written as $A = [\bar{\mathbf{u}}_0^A \ \bar{\mathbf{u}}_1^A \ \bar{\mathbf{u}}_2^A \ O^A]$ and $B = [\bar{\mathbf{u}}_0^B \ \bar{\mathbf{u}}_1^B \ \bar{\mathbf{u}}_2^B \ O^B]$ where $O^A, O^B \in \mathbb{R}^4$ are the origins of A and B respectively then the misalignment of A and B is given by Equation (4.7).

$$d(A, B, P^A, P^B) = \|O^A - O^B\| + \sum_{i=0}^2 \|\bar{\mathbf{u}}_i^A - \bar{\mathbf{u}}_i^B\| \quad (4.7)$$

Solving the minimisation problem yields a vector $\bar{k}^* \in \mathbb{Z}^N$ of indices that select the best candidate model-view matrix for each prop. Let $\{C_n \in M_{4 \times 4} : n \in \mathbb{Z}_0^{N-1}\}$ be the set of N best model-view matrices. The set of matrices is consolidated by decomposing each into a coordinate frame $C_n = [\bar{\mathbf{u}}_0^n \ \bar{\mathbf{u}}_1^n \ \bar{\mathbf{u}}_2^n \ O^n]$ and constructing the final model-view matrix M^* as shown in Equation (4.8). M^* is used to specify the camera transformation required to view the world space. M^{*-1} provides the location and viewing direction of the camera on the floor plan so that the corresponding shot symbol can be placed automatically (see Section 4.5.5).

$$M^* = \left[\frac{1}{N} \sum_{n=0}^{N-1} \bar{\mathbf{u}}_0^n \mid \frac{1}{N} \sum_{n=0}^{N-1} \bar{\mathbf{u}}_1^n \mid \frac{1}{N} \sum_{n=0}^{N-1} \bar{\mathbf{u}}_2^n \mid \frac{1}{N} \sum_{n=0}^{N-1} O^n \right] \quad (4.8)$$

4.7 Implementation

The components of the proposed framework for a sketch-based storyboarding tool for authoring pre-visualisations were implemented as part of the design and implementation process. An implementation of each algorithm was required in order to investigate its effectiveness and efficiency during the algorithm design process (see Section 4.6). The

implementation of the GUI was required so that the GUI design and the sketch-based storyboarding approach could be evaluated (see Chapter 5).

This section discusses the implementation of the GUI and algorithm design described in this chapter. It begins by discussing the tools that were used in order to implement the prototype and create the required environmental data (see Section 4.4). It continues by discussing third party implementations and libraries that were used in order to implement selected subcomponents of the prototype. The section concludes with a discussion on the implementation of the components of the prototype.

4.7.1 Implementation Tools

The prototype was implemented in the C# programming language on the .NET framework (Microsoft 2012). The language provides many modern programming features that allowed for faster and easier prototyping. The object orientated nature of language and support for operator overloading, delegates and events proved useful for implementing components and frameworks. A small utility framework for representing working with structures was implemented. Example mathematical structures that were implemented include classes for graphs, matrices, and quaternions. The utility framework and implementation of algorithms from previous research⁶ were reused in the implementation of the prototype.

The 3D models required by the prototype for representing the set, props and characters were created using Blender 3D (see Section 2.6.2) (Blender Foundation 2012). The modelling and animation tool was used for creating the meshes for each 3D model. A generic character was created with an attached armature. The generic character was modified in order to create the four characters used in the Goldilocks scenario. A Python script was created to export the 3D model data into an XML format that is read by the prototype tool (see Appendix B).

The tablet computer used for the implementation and evaluation of the prototype runs on the Microsoft Windows 7 operating system. Windows 7 provides sufficient support for third party libraries (see Section 4.7.2). It also provides a native Application Programming Interface (API) for retrieving multi-touch input, which was necessary for the implementation of the prototype. Windows 7's graphical user interface is based on the Windows Icons Menu

⁶ This research on sketch-based articulated figure animation was published in the following paper: Matthews, T. and Vogts, D. (2011): A sketch-based articulated figure animation tool. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists (SAICSIT)* The Pavilion Conference Centre, V&A Waterfront, Cape Town, South Africa.

Pointers (WIMP) approach. It is designed to receive input from the mouse and keyboard. The operating system supports touch interaction and multi-touch gestures; however, the default Windows 7 component model is not touch friendly. This is because the buttons are too small to touch comfortably. The components do not respond as well to touch input as they do with mouse interaction, e.g. scrollbars are hard to use. The user interface of the prototype was implemented using its own, touch-friendly component model instead of the default Windows component model.

4.7.2 Third Party Libraries Used

The prototype tool was implemented using two third party libraries. The first library is called the Tao Framework (Mono 2012). It provides an API for accessing the graphics hardware of the tablet computer so that 3D content can be presented in real time. The Tao Framework provides a wrapper class library for accessing OpenGL API. The OpenGL API is used to call graphics routines that are used to send geometry and graphics data to the graphics hardware (Hill and Kelley 2007; Khronos Group and Silicon Graphics 2012).

The second library is called the EMGU CV (EMGU 2012). The library provides wrapper classes to the OpenCV image processing library (Intel Corporation, Willow Garage and Itseez 2012). The OpenCV API provides access to efficient implementations of image processing routines that are used in computer vision. The API provides an implementation of the POSIT algorithm used for performing pose estimation (see Section 3.5) (Dementhon and Davis 1995). An implementation of the SIFT local feature descriptor was used for extracting local features and generating their image descriptors (Lowe 2004b; Tabibian 2005).

4.7.3 Component Implementation

This section provides a brief discussion of the implementation of the components of the GUI and the 3D Context component (see Figure 4.2). It begins by discussing the implementation of the Computer Graphics Module and the custom component model used to implement the GUI. It continues to discuss the implementation of the multi-touch module and the graphical user interface. The section then concludes with a brief discussion of the implementation of the Computer Vision Module.

4.7.3.1 Computer Graphics Module

The Computer Graphics Module was implemented in OpenGL using the Tao Framework. It provides an environment for rendering content by encapsulating OpenGL routines in order to

render 3D models and 2D content for presenting the user interface. The Computer Vision Module also provides useful functions for rendering content off-screen and un-projecting points from the image buffer into the 3D world. These functions were used to implement the adjustable 3D view provided by the Sketch Editor. The un-project function was used to obtain the 3D data required for sampling and training images. It was also used for performing pose estimation if the model-view matrix is known (see Section 4.6.5). The Computer Graphics Module was also used to implement the component model for the prototype's GUI.

4.7.3.2 Component Model

A custom component model was implemented to provide an environment for building touch-friendly GUIs. It contains controls that are similar to several Windows controls, such as buttons, list boxes, labels, panels and image boxes. Panels can be nested similarly to the containers implemented in the Windows component model. It also provides a 3D scene control which is used for 3D visualisation within a specific area on the screen.

The component model was implemented in OpenGL using the Computer Graphics Module. This enabled the GUI to call OpenGL routines directly for rendering 3D content when required. The design of the component model was guided by Android GUI design guidelines (McKenzie 2011).

The custom components were implemented so that they can be touched and manipulated comfortably. Figure 4.34 provides a comparison between the example components from the default Windows component model and the custom component model. The red circles indicate the area required for the user to interact comfortably with each component. The elements of each Windows control of the windows component model are too small to touch comfortably. The controls provided by the custom component model have larger regions that can be interacted with. The use of text is also minimised and replaced with informative icons. The custom component model supports all the touch gestures discussed in Section 4.5.1.

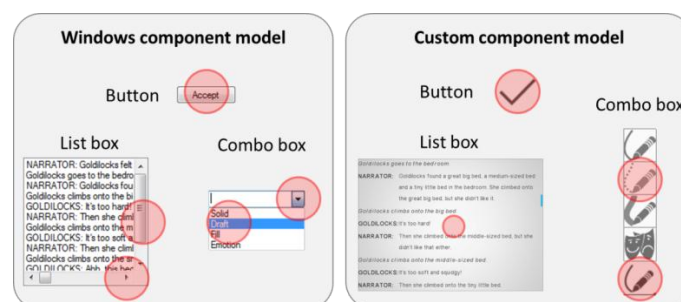


Figure 4.34: A comparison between two component models: Windows default and custom.

4.7.3.3 Multi-touch Module

The multi-touch module was implemented for analysing touch-based input in order to capture the touch gestures discussed in Section 4.5.1. The multi-touch module sends event messages to the component model for each touch gesture performed by the user. The module handles two cases of touch input. The first case is when only one touch point is active and the second case is when two touch points are active.

If there are no active touch points then the multi-touch module registers a single-touch gesture at the touched location. If the user releases the multi-touch surface then a tap gesture is invoked. If the user's fingertip is moved then the multi-touch module updates the position of the first touch point and invokes the drag gesture. If the movement of drag gesture is large enough along an axis then the flick gesture is invoked.

If the user touches the screen with a second finger while the multi-touch module is handling the single-touch case then it handles the multi-touch case for the second touch point and single-touch case for the first touch point. The multi-touch case involves calculating the length and orientation of the line segment connecting the two touch points. If the user moves the first touch point or the second touch point then the scaling, rotation and translation parameters for that gesture are calculated. A manipulation gesture is invoked with the required scaling, rotation and translation parameters. If the scaling is large enough then the relevant spread or pinch gestures are invoked.

4.7.3.4 Computer Vision Module

The Computer Vision Module was implemented using the utility framework discussed in Section 4.7.1 as well as computer vision libraries and algorithm implementations discussed in Section 4.7.2. It implements routines for extracting edges from raster images in order to generate query images and sample images. All the approaches discussed in Section 4.6.3 were implemented, however the final approach was selected for generating image descriptions, measuring the distance between these descriptions and finding local feature correspondences. The object recognition routine was implemented with an ordered dictionary that sorts candidate objects based on their distance from the query image descriptor. The pose estimation methods discussed in Section 4.6.5 and the camera estimation algorithm discussed in Section 4.6.6 were implemented using the Computer Graphics Module, the utility framework and the POSIT implementation provided by OpenCV.

4.8 Quantitative Performance Evaluation

A quantitative performance evaluation was conducted to measure the scalability and accuracy of the prop recognition and camera estimation algorithms discussed in Section 4.6. The experiment has a limited scope but it evaluates the performance of these algorithms sufficiently within the context of this chapter. This section discusses the evaluation methodology followed and the resulting outcomes.

4.8.1 Methodology

The quantitative performance evaluation measured the following three performance metrics:

1. Prop recognition scalability: The total time (in seconds) required to recognise all the props in the user's sketch.
2. Camera estimation scalability: The total time (in seconds) required to estimate the camera from the user's sketch.
3. Camera estimation accuracy: Measured by scoring the estimated camera as follows: 0% (estimation failed), 50% (camera requires adjustment) or 100% (camera requires minor (or no) adjustment).

These performance metrics were measured against a varying number of props (up to six props) with varying shapes in order to determine the scalability and accuracy of the prop recognition and camera estimation algorithms. This was achieved by creating six template sketches as listed in Table 4.3. Each template was sketched $n = 10$ times and the above metrics were measured for each sketch.

Table 4.3: The sketches used for quantitative performance evaluation.

Template	Props
1	Table
2	Big couch, medium couch
3	Big couch, medium couch, small couch
4	Table, kitchen chair, 2 bowls
5	Big couch, medium couch, table, bowl, kitchen chair
6	Table, 3 kitchen chairs, 2 bowls

The quantitative performance evaluation was conducted on a desktop personal computer with the specifications given in Table 4.4.

Table 4.4: Quantitative performance evaluation hardware specifications

Feature	Specification
Processor	Intel Core i7 3.4 GHz
Memory (RAM)	8 GB
Graphics Acceleration	Dedicated, AMD Radeon HD 6900 Series

4.8.2 Performance Results

The total time required (t) by the prop recognition algorithm to recognise all the props (n) in the sketch was measured and analysed. Figure 4.35 (a) illustrates the scalability of the prop recognition algorithm in terms of runtime with 95% confidence intervals. The high Pearson product-moment correlation coefficient ($R^2 > 0.9$) indicates a linear correlation between t and n . This implies that the algorithm is $O(n)$.

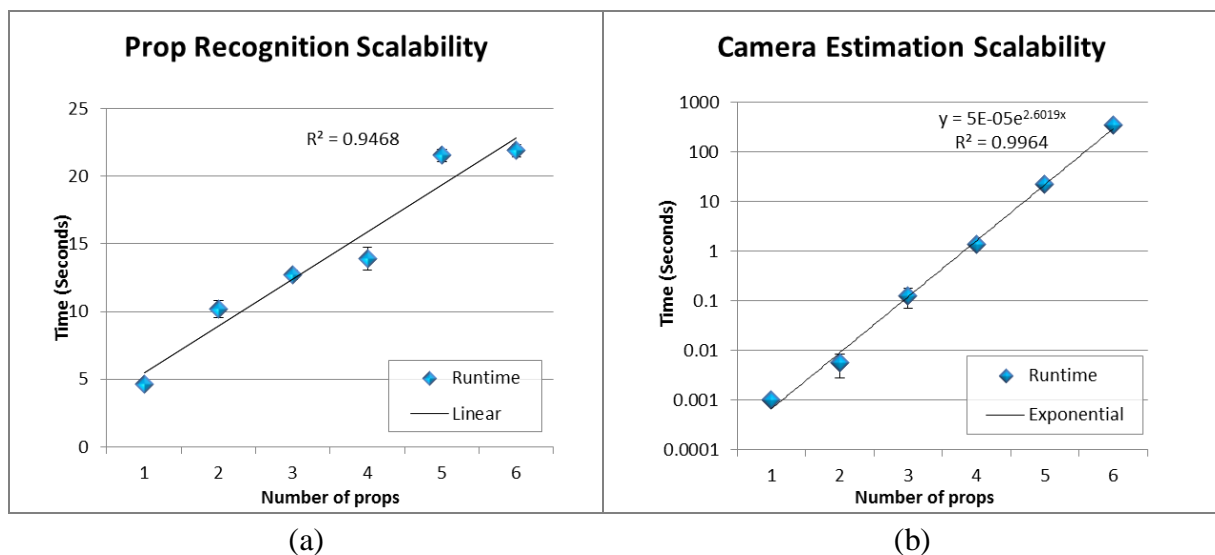


Figure 4.35: (a) Prop recognition scalability and (b) camera estimation scalability ($n=10$).

The total time (T) required by the camera estimation algorithm to estimate the camera of a sketch with n props was also measured and analysed. Figure 4.35 (b) illustrates the scalability of the camera estimation algorithm in terms of runtime with 95% confidence intervals and a logarithmic vertical scale. The high correlation coefficient $R^2 > 0.9$ between $\log_{10}(T)$ and n indicates that the camera estimation algorithm is exponential of nature. This implies that the algorithm is $O(b^n)$ for some b .

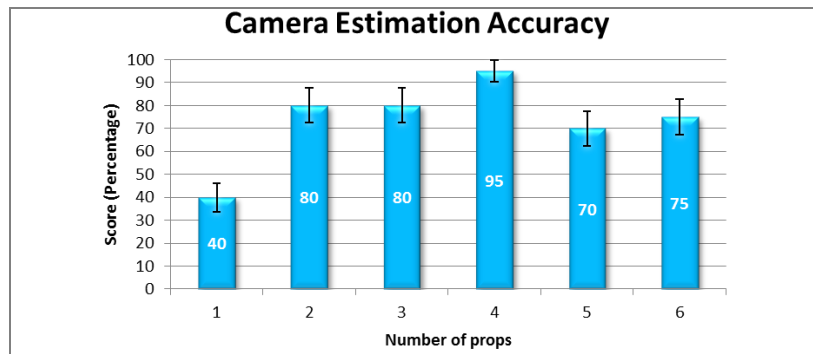


Figure 4.36: Camera estimation accuracy ($n=10$).

The camera estimation accuracy (C) with n props was also measured and analysed. Figure 4.36 illustrates the accuracy of the camera estimation algorithm with 95% confidence intervals. The low correlation coefficient ($0 < R^2 < 0.22$) indicates that C and n are not linearly correlated. However, it can be noted that the accuracy of the algorithm was at least 70% when $n > 1$.

4.8.3 Discussion

The results of the quantitative performance evaluation showed that the camera estimation algorithm grows exponentially. This can be explained as follows. The implemented algorithm minimises the error function (see Equation (4.6)) by exploring every possible value of $\bar{k} = (k_0, k_1, \dots, k_{n-1})$. This operation grows exponentially as the number of props (n) increases. The scalability of this algorithm can be improved in future research by investigating other optimisation methods.

The camera estimation algorithm demonstrated that it could estimate the camera consistently with a score of over 70% if $n > 1$. This is because the algorithm uses the relative location of each prop on the set in order to provide an improved camera estimation. If there is only one prop then the algorithm can only use the prop's appearance to determine the orientation and location of the camera. Furthermore, the prop recognition algorithm could not always successfully recognise and orientate props that are symmetric, lack clearly defined edges or have too many edges (i.e. noisy image descriptors). Quantitative performance evaluation showed that the prop recognition algorithm scales well as the number of props increases (linear growth).

4.9 Conclusions

The chapter addressed the third research question identified in Chapter 1, namely *how can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?* This research question was answered by answering the sub-questions provided in Section 4.1.

Q_{3.1} was answered by proposing a framework for improving the user interface layer of the general pre-visualisation framework discussed in Section 4.2. Q_{3.2} was answered by discussing the data design that describes how the data can be structured in order to implement the proposed approach. Q_{3.3} was answered by discussing the design of a touch-friendly GUI for storyboarding on a tablet. A navigational layout for the required GUI components was proposed. The design of each subcomponent was discussed, with the focus being on the subcomponents receiving sketch-based input. The subcomponents requiring sketch-based input were designed to utilise the Computer Vision Module.

Q_{3.4} was answered by discussing the design for the Computer Vision Module in Section 4.6. The section proposed and discussed several methods for describing the contents of user sketches. It was found that the best approach for describing the user's sketch was to use a single high resolution disk and a low resolution disk for describing sketched objects after they have been normalised using PCA. The ICP and discreet disk rotation methods were applied to achieve rotational invariance. The DoG local region-based feature detector and the SIFT feature descriptor were used to map features in order to find point correspondences. The correspondences proved to be useful for rigid body pose estimation. Object recognition was achieved by using k -NN classifier with the mean SIFT error as the distance function. An algorithm was proposed for estimating the camera for a user sketch by combing information from the floor plan with the pose data estimated for each prop the user sketches.

Q_{3.5} was answered by discussing on how the proposed prototype was implemented. In particular, it was shown how the GUI and Computer Vision Module could be implemented for Windows 7 using a custom component model, OpenGL and OpenCV.

The next chapter discusses the usability evaluation of the prototype tool in order to assess its usability benefits. It investigates the extent to which the sketch-based storyboarding approach supports the user in authoring pre-visualisations effectively and easily.

Chapter 5:

Usability Evaluation

5.1 Introduction

Chapter 4 discussed the design and implementation of a proof of concept prototype sketch-based tool for authoring pre-visualisations using a storyboarding approach. This chapter evaluates the implemented prototype in order to answer the fourth research question identified in Chapter 1 namely, *Q₄: to what extent does a sketch-based storyboarding interface support the user in authoring pre-visualisations in terms of software usability?* The following sub-questions are answered:

- Q_{4.1}: To what extent does the sketch-based storyboarding interface follow the design guidelines for touch-based GUIs that support sketch-based input?
- Q_{4.2}: To what extent does the sketch-based storyboarding interface support the user in terms of performance and user satisfaction?
- Q_{4.3}: To what extent do users prefer the storyboarding approach over the conventional pre-visualisation authoring approaches?
- Q_{4.4}: How can sketch-based pre-visualisation authoring tools with a storyboarding approach be designed in order to support the user in terms of usability?

The chapter begins by investigating and selecting appropriate methods for evaluating the usability of a prototype tool. It continues by providing an analytical evaluation of the prototype design. The results from an empirical evaluation are then reported as well as the observations made during usability testing. The chapter concludes by providing a set of guidelines, based on the findings of this chapter, for designing sketch-based pre-visualisation authoring tools using a storyboarding approach.

5.2 Existing Evaluation Methods

The goal of evaluating the usability of a prototype (system, product or design) is to determine to what extent users can accomplish their own tasks quickly and easily (Dumas and Redish 1999). Traditionally, usability is evaluated through usability testing on desktop computers within controlled environments (Sharp *et al.* 2007). Usability testing involves measuring the user's performance and satisfaction while using the prototype. This is achieved by observing and recording the user's actions in order to identify usability problems.

Usability problems are identified by assessing how efficiently and accurately the user can perform each task. User accuracy can be measured by noting errors and user efficiency can be measured by recording the time required to perform each task. The defining characteristic of usability testing is that it is conducted in a controlled environment with laboratory (or laboratory-like) conditions (Koyani *et al.* 2004). Using a controlled environment isolates the user from day-to-day interruptions. This is important when capturing performance related data such as the time taken per task. Using a control environment ensures that the data collected is not affected by external variables that may distort the data. For example, if a user is busy with a task on a multi-touch surface and fails because a second person touched the screen in order to point something out then, the unanticipated external variable (multi-user collaboration) distorts the usability data. Field studies are conducted when it is important to consider the natural working environment of the user.

Field studies can be conducted remotely where the evaluator is not present during the evaluation of the prototype. The main data collection methods for this evaluation approach are recording interaction data on the prototype automatically and using satisfaction questionnaires. Remote evaluators are not able to observe the user during the evaluation session and important usability issues can be missed. If the evaluator is present during a field study and evaluates the usability of the prototype within the context of the user's natural environment then it is known as an ethnographic usability study. They are useful for identifying opportunities for new technologies, establishing requirements for a design, introducing a new technology or deploying an existing technology in a new context (Balaji *et al.* 2005). Data is recorded by using non-obtrusive observations and interviewing participants outside the use of the prototype.

If the goal of the study is only to identify usability issues and maximise the usability of the prototype then a field study may not necessarily be appropriate. Kaikkonen *et al.* (2005)

found that field studies do not necessarily reveal more usability issues compared to conducting usability testing. This is because the time allocated for evaluation is often limited and field studies require more time and effort to conduct. Tan *et al.* (2006) showed that if there is sufficient time available then field studies reveal more usability issues because participants tend to respond more negatively to usability issues in the field than in the laboratory.

Usability testing and field studies can be very costly in terms of time and effort (Vredenburg *et al.* 2002). Shorter, informal usability tests are often conducted with a smaller population size during the design and implementation phase of the development of the prototype. This ensures that the prototype's GUI is understandable and addresses the needs of the user (Sharp *et al.* 2007). Formal (and informal) heuristic evaluations can be conducted without involving users. Heuristic evaluations are made using guidelines and standards which are known as heuristics in the usability community. Examples of heuristics for evaluating the usability of user interfaces include those developed by Nielsen (1994).

5.3 Selection of Evaluation Methods

The selection of the evaluation methods was influenced by two goals:

- G₁: To measure the extent to which the prototype follows the design guidelines for touch-based GUIs that support sketch-based input.
- G₂: To measure the usability of the prototype in terms of performance and user satisfaction.

The first goal was achieved by performing informal heuristic evaluations on the design of the prototype. The second goal was achieved by conducting usability tests in order to measure the usability benefits of the prototype tool and identify usability issues. The high-level outcome of the usability evaluation was to measure how effectively the user can author pre-visualisations using a sketch-based user interface with a storyboarding approach. The evaluation method was selected because related research showed that sketch-based GUIs can be evaluated successfully through usability testing (Shin and Igarashi 2007; Kelleher 2006; Matthews and Vogts 2011).

5.4 Analytical Evaluation Design

The prototype design was evaluated analytically by the researcher. This was achieved by conducting informal heuristic evaluations. The extent to which the prototype design satisfied

existing guidelines or standards (heuristics) was determined. This involved inspecting and analysing the prototype design using these heuristics in order to identify usability issues (Sharp *et al.* 2007). The following heuristics were compiled into a checklist (see Appendix F) and applied:

- Nielsen's heuristics were used to evaluate the design of the graphical user interface (1994).
- The design guidelines proposed by Salo *et al.* (2012) were used to evaluate the prototype's 2D/3D touch interaction design.
- The guidelines proposed by Wais *et al.* (2007) was used to evaluate the prototype's sketch-based recognition design.

The outcomes of the informal heuristic evaluation included a set of usability issues for each heuristic. The evaluation method also required example instances to be identified for each usability issue. The next section will discuss the resulting outcomes of the analytical evaluation.

5.5 Analytical Evaluation Results

The findings made during the heuristic evaluation of the prototype GUI design are presented in this section. A discussion on the extent to which the design complies to (or violates) each heuristic is provided (see Appendix F). The resulting usability issues are then summarised at the end of this section.

5.5.1 Nielsen's Heuristics for the GUI Design

The findings made by applying Nielsen's Heuristics (1994) are as follows:

1. *Visibility of system status*

The design of the GUI utilises progress bars and animated icons to illustrate the progress of processes and system responsiveness. This is achieved by executing processor intensive tasks in separate threads.

2. *Match between system and real world*

The GUI is designed from the director and storyboard artist's perspective. It uses terminology that is familiar to the user, e.g. staging and storyboarding. No technical jargon is used, e.g. articulated figures and meshes.

3. *User control and freedom*

The GUI is designed with navigational buttons, tabs and screen titles that show where the user is at all times. The user can use the appropriate gesture or the “go out” button if the incorrect screen is opened accidentally. The sketch editing components also allow the user to correct mistakes using an eraser. Objects that have been placed on the floor plan can be manipulated using gestures. The current design and implementation of the GUI does not support undo/redo functionality.

4. *Consistency and standards*

The functions of the GUI are consistently designed and laid out across the various screens. The sketch editing components have similar sketch editing tools. The Android touch GUI guidelines were followed during the GUI design (see Section 4.5.1).

5. *Error prevention*

Errors are prevented by disabling functions that are not applicable based on the active state of the GUI. Dialogue boxes are used to provide meaningful instructions if invalid user actions are performed. The dialogue boxes are built into the GUI and do not make use of the default Windows dialogue windows.

6. *Recognition rather than recall*

The information for each narrative block is visible throughout the design. The floor plan shows the set and the shooting strategy and the action/dialogue entries for the storyboard can be selected directly from the script. The current design and implementation require the user to maintain a mental map of the floor plan and the colour of each prop.

7. *Flexibility and efficiency of use*

Props, symbols and storyboard sketches can be created quickly by sketching them directly as they should appear. The user can sketch while the prototype is recognising existing sketches.

8. *Aesthetic and minimalistic design*

Each screen contains the minimum number of tools required to perform the required tasks. Informative icons are used instead of unnecessary text. The elements of the GUI are designed with a touch-friendly look and feel (see Section 4.5.1). The design is aesthetic and minimalistic.

9. *Recognise, diagnose and recover from errors*

Understandable error messages are provided if the user attempts an action that would place the prototype in an erroneous state. If object recognition fails to recognise the correct prop during set design then the user can select the correct prop from a list of alternative props. Currently, options for changing floor plan symbols and emotion symbols are unavailable. The user is required to erase the symbol and sketch it again.

10. *Help and documentation*

The GUI was designed so that it could be used without the need for referring to documentation. The current design does not support online documentation.

5.5.2 Heuristic Evaluation for Touch-based Interaction with a 2D/3D Interface

The findings made by evaluating the touch interaction design for the 2D/3D prototype GUI against the guidelines proposed by Salo *et al.* (2012) are as follows:

1. *Provide onscreen touch gestures*

The design allows the user to directly manipulate objects in the editing environment using tap, drag, rotate and scale gestures. Spread, pinch and flick gestures are available for navigating between screens (see Figure 4.7).

2. *Avoid gestures which are too similar to each other*

The gestures can be easily distinguished based on their function and the context in which they are used. For example, spreading or pinching on a component that allows editing causes the view to zoom in and out respectively. Performing a spread or pinch gesture on the screen and disabling the pan/zoom/rotate function causes the prototype to navigate to the relevant screen. The current design and implementation of the prototype does not provide online guidance showing how the gestures are performed.

3. *Reduce the need for overlaying UI controls*

The editing components do not contain any controls which are overlaid on the working area. The tools and functionalities are located to the south and east panels of the screen (see Section 4.7.3).

4. *Only use 2D icons in 2D space*

The icons used in the design of the GUI are only used in a 2D context. 2D icons are used to indicate symbols and characters in the floor plan editor. The floor plan is a top-down 2D orthographic projection of the 3D virtual environment. It appears to be a simple 2D floor plan of the environment.

5. *Use simple, large and consistent icons*

Simple and meaningful icons are used to represent the tools that the user can interact with. The icons are large enough to be touched comfortably and they are not placed too close to one another. The designs, locations and functions are consistent for each icon throughout the GUI design.

5.5.3 Heuristic Evaluation for Sketch-based Recognition

The findings made by evaluating the sketch-based recognition design against the guidelines proposed by Wais *et al.* (2007) are as follows:

1. *Efficient and reliable recognition triggers*

The trigger for recognising sketches is automatic. The recognition is triggered after a specific amount of time after the user completes the last stroke of the sketch. The design does not currently support user controlled recognition triggers.

2. *Separate recognised and unrecognised objects*

Objects that have not been recognised are illustrated using the user's original strokes. Recognised sketches, such as props, characters and shots are replaced with the respective 3D models and symbol icons to indicate that recognition has been completed.

3. *Minimise the clutter and user sketch transformations*

The design allows the user to enlarge and adjust the view of the sketching area. This allows the user to sketch the floor plan and individual storyboard panels without cluttering the working environment. The user's sketches are also automatically hidden once they have been interpreted and the 3D pre-visualisation has been prepared. The user can choose to unhide these sketches as well. Recognised objects can be manipulated but the user's original strokes are not modified or transformed at all.

4. *Allow errors to be corrected after sketching*

The design provides an eraser for performing corrections in sketches. Movement and rotation tools are available for adjusting objects. Alternatives are presented to the user when performing object recognition.

5. *Provide predictable and understandable recognition errors*

The current design and implementation of the GUI does not support online assistance for sketching objects. The task list was used to illustrate how objects should be sketched when performing the staging tasks and sketching storyboard panels (see Appendix H).

5.5.4 Usability Issues

This subsection reports on the usability issues identified during the analytical evaluation of the GUI component designed and implemented for the SISPA framework (see Figure 4.2):

- The GUI requires undo/redo functionality so that the user can easily correct mistakes.
- The GUI requires the user to maintain a mental map of the floor plan while sketching the props for a shot on a blank storyboard panel. This can be resolved by providing a small preview area of the floor plan that shows the individual props, where they are on the set and which colour they are. The user should be able to hide and retrieve the preview window using a swipe gesture so that it does not obstruct the working area.
- The GUI does not allow the user to change recognised floor plan and emotion symbols. This can be resolved by adding a popup menu that shows alternatives when the user touches the symbol.
- The design and implementation of the prototype does not support built-in help that documents how the various features of the GUI are used, how objects are sketched and how touch gestures are performed. This can be resolved by integrating a sub GUI that provides a help system similar to the Windows Help system. It should follow the design guidelines for touch-based interfaces and illustrate how gestures are performed graphically.

5.6 Empirical Evaluation Design

A usability evaluation was conducted in order to measure the usability of the prototype tool and identify usability benefits and issues (see Appendix D, Appendix E and Appendix G). The evaluation also measured to what extent the participants preferred the sketch-based

storyboarding approach over conventional pre-visualisation authoring tools, based on their personal experience. This section discusses how the evaluation was designed in order to make these measurements. It discusses the background of the participants used for the evaluation, the evaluation metrics measured and the instruments used. The section concludes by discussing the task list and the experimental setup and procedure.

5.6.1 Participants

Tullis and Albert (2008) recommends using five participants per significantly different class of user. Two user classes were considered, namely users with a background related to filmmaking and participants with a background related to Information Systems. A representative sample of ten students was drawn from the Department of Photography; the Department of Journalism, Media and Philosophy; and the Department of Computing Sciences at the Nelson Mandela Metropolitan University (NMMU) (see Figure 5.1 (a)).

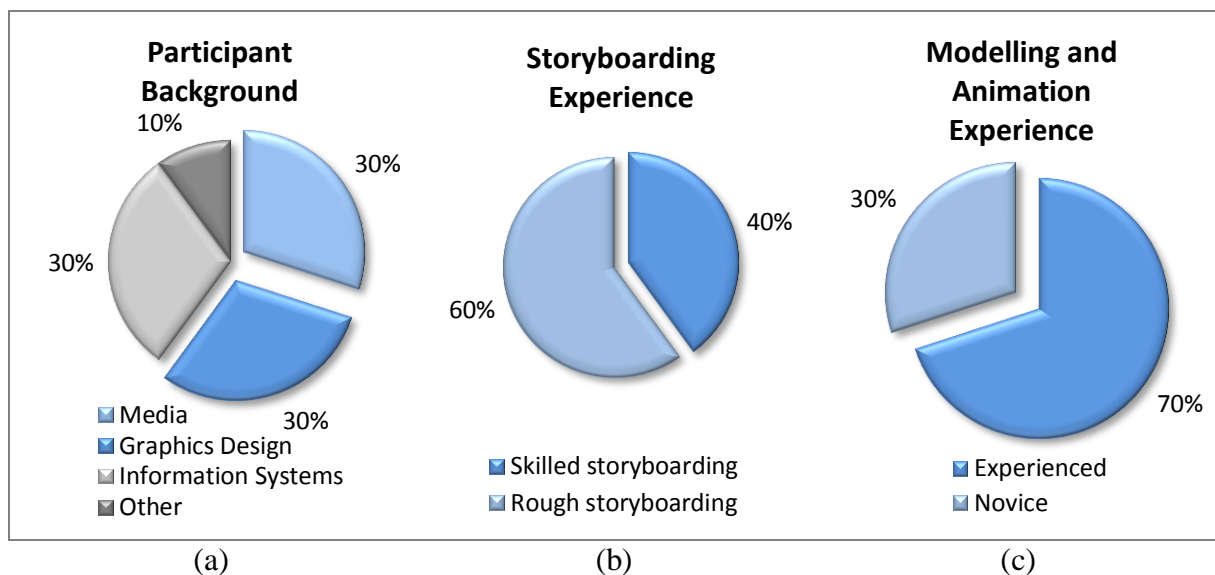


Figure 5.1: Participant (a) background (b) storyboarding experience and (c) modelling and animation experience ($n=10$).

60% of the participants had a film related or media related background. The participants were required to have at least some experience in sketching storyboards. 40% of the participants were skilled at sketching storyboards and 60% had basic storyboarding experience (see Figure 5.1 (b)). 70% of the participants had experience with using 3D modelling and animation tools (see Figure 5.1 (c)).

Participants were asked to specify if they had any experience in activities performed during the filmmaking process (see Figure 5.2). All of the participants reported that they had storyboarding experience. It should be noted that nine of the ten participants reported that

they had experience with writing or reviewing scripts, and two of the participants reported that they had performed computer animation for a project. Participants also had experience in planning, shooting, editing and performing other filmmaking activities such as directing and creating shot lists.

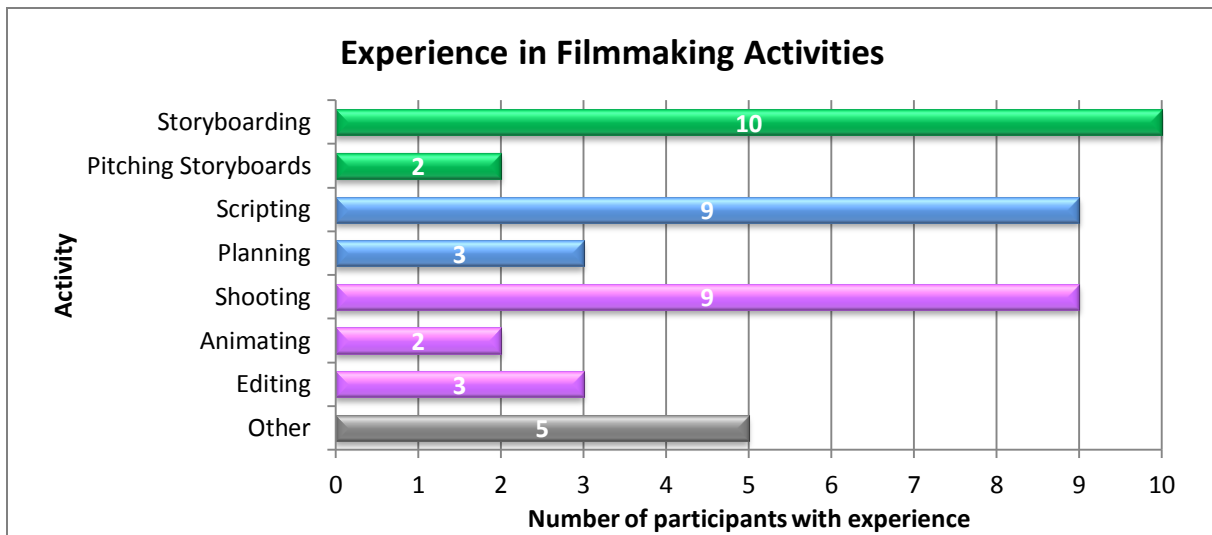


Figure 5.2: Participant experience with filmmaking activities.

5.6.2 Evaluation Metrics

The ISO9241-11 standard defines usability as the “*extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*” (1998). The definition refers to three metrics for measuring the usability of the prototype:

1. **Effectiveness** is a performance metric that measures the accuracy and completeness of the tasks performed by the user. For example, the user performed nine out of ten tasks successfully and made three errors in the process.
2. **Efficiency** is a performance metric that measures the time taken or the number of user actions required to complete tasks successfully. For example, the user took 2 minutes to complete the task and touched the touch display 40 times during the use of the system.
3. **Satisfaction** as a satisfaction metric for measuring the positive and negative attitudes towards the product. For example, the user commented that the GUI is easy to use but needs a feature.

Performance metrics were measured by observing the success level, time required and errors made by the user for each task performed during usability testing. The physical effort

expended in using the prototype was also recorded as performance metrics. This was achieved by automatically logging the touch-based gestures and stylus taps/strokes performed by the user. The learnability of the prototype was evaluated by measuring how the time and physical effort required to complete tasks successfully changed as the users became more proficient with the prototype. Cognitive effort was measured as part of the satisfaction questionnaire. Quantitative data and qualitative data were collected using satisfaction questionnaires in order to measure user satisfaction. The next section discusses the instruments that were used in order to conduct the usability test and gather the required information.

5.6.3 Instruments

Each participant was presented with a list of tasks that needed to be performed in order to evaluate the prototype (see Appendix H). The participants were each asked to complete a pre-task questionnaire for collecting background information (see Section 5.6.1 and Appendix I). The evaluation was divided into two parts. The first part of the evaluation covered all the features that support the staging activity. The second part of the task list covered the features required for performing the storyboarding activity.

The user's effectiveness in performing each task was measured and recorded on a performance sheet for each part of the evaluation (see Appendix J). The performance sheet recorded whether the participants were able to complete each task successfully and accurately. It also recorded the time required and the number of errors made for each task. The participants were observed in order to identify usability issues.

User satisfaction was measured using a user satisfaction questionnaire for each part of the evaluation (see Appendix K and Appendix L). The user satisfaction questionnaires were adapted from the Computer System Usability Questionnaire (CSUQ) (Lewis 1995). It measures the user's cognitive load using a 5-point semantic differential scale. Overall user satisfaction and views of the usability of the prototype were measured using a 5-point Likert scale.

5.6.4 Tasks

The task list used for evaluating the prototype tool was designed for performing the staging and storyboarding for a specific scenario story, namely *Goldilocks and the Three Bears* (see Appendix H). The prototype was preloaded with a script, set, props and characters. The participant was required to author a small part of the Goldilocks story. The task list was

divided into two parts. The first part required the user to sketch the design of the set for the living room and create a floor plan that illustrated the shooting strategy for the particular dramatic block. The second part of the evaluation required the user to design the set for the kitchen and sketch a 3D illustration of the shot for the storyboard of the dramatic block. The tasks included the following:

1. Staging part (narrative block: *“My chair is broken!”*)
 - 1.1. Navigate from the story viewer to the floor plan editor for the narrative block.
 - 1.2. Design the set for the narrative block.
 - 1.3. Block out each character.
 - 1.4. Indicate the movement of each character.
 - 1.5. Specify the position and direction of each shot.
 - 1.6. Indicate the movement of a shot.
 - 1.7. Associate each shot with the characters that appear in it.
 - 1.8. Navigate to the storyboard editor and populate the dialogue/action entries.
2. Storyboarding part (narrative block: *“My porridge has been eaten!”*)
 - 2.1. Design the set for the narrative block.
 - 2.2. Create a new, blank storyboard panel.
 - 2.3. Sketch shot #1 for the narrative block indicating props and characters.
 - 2.4. Indicate the emotional expression on each character’s face.
 - 2.5. Navigate to the floor plan editor and add a second shot #2.
 - 2.6. Navigate to the Sketch Editor for shot #2.
 - 2.7. Use the camera manipulation tools of the Sketch Editor to adjust shot #2.
 - 2.8. Add props to shot #2 by sketching them on the 3D rendering.
 - 2.9. Add characters to shot #2.
 - 2.10. Use the floor plan editor to indicate the motion of each character.

5.6.5 Experimental Setup and Procedure

Usability testing was conducted in a controlled environment. The prototype was installed on an Asus EEE Slate EP121 tablet computer. The tablet was placed on a flat table surface with a chair for the participant and a chair for the evaluator. The table was prepared by laying out the required instruments and loading the prototype. A notepad and stopwatch was also used for recording time-per-task data and noting usability issues.

The evaluation of the prototype proceeded as follows. A single evaluation was conducted at a time for each participant. The procedure of the evaluation, as well as the goals of the study, were briefly explained to each participant. The participant provided consent for taking part in the evaluation (see Appendix D). The Pre-Task Questionnaire was completed by the participant in order to collect background information. The user was given a short training session once the prototype was loaded and the participant completed the Pre-Task Questionnaire (see Appendix I). The participant was then asked to complete the tasks listed in the staging part of the evaluation. The participant then completed the staging Post-Task User

Satisfaction Questionnaire (see Appendix K). Similarly, the participant was asked to complete the second part of the evaluation and fill in the storyboarding Post-Task User Satisfaction Questionnaire (see Appendix L).

Observed usability issues and performance data were recorded on the notepad and the Performance Sheet throughout the evaluation (see Appendix J). The prototype performed automatic logging of all the touch and stylus input provided by the user. This was achieved by recording each touch event and mouse event invoked by the GUI for each screen. An image of the screen and its event data was captured and written to secondary storage each time the participant navigated to a different screen.

5.7 Empirical Evaluation Results

The usability of the proof of concept prototype implementation of the SISPA framework was measured empirically in terms of performance metrics (effectiveness, learnability and efficiency) and user satisfaction metrics (see Section 5.6.2). This subsection will report on the empirical evaluation results as well as the observations made during the usability tests.

5.7.1 Performance Results

This subsection discusses the performance results obtained from analysing performance data collected from usability testing. The performance results shows how effectively and efficiently the users could perform each task using the prototype implementation of the SISPA framework. The task success rate, the time per task, the errors per task, the efficiency of performing tasks and the learnability of the GUI are discussed.

5.7.1.1 Task Success Rate

The success rate of each task was measured by recording whether each task was completed successfully or not and whether there were any problems. The success of each task for each participant was recorded as being completed successfully with “no problems” or “some problems”. If the task was not completed successfully then it was recorded as “failed”. Figure 5.3 (a) shows the success rates of the staging tasks and Figure 5.3 (b) shows the success rates for the storyboarding tasks.

Most of the tasks received a 100% success rate except for task 2.9 which received a success rate of 90%. The participant performed the task correctly but an implementation issue caused the prototype to become non-responsive (memory leak caused by off-screen rendering).

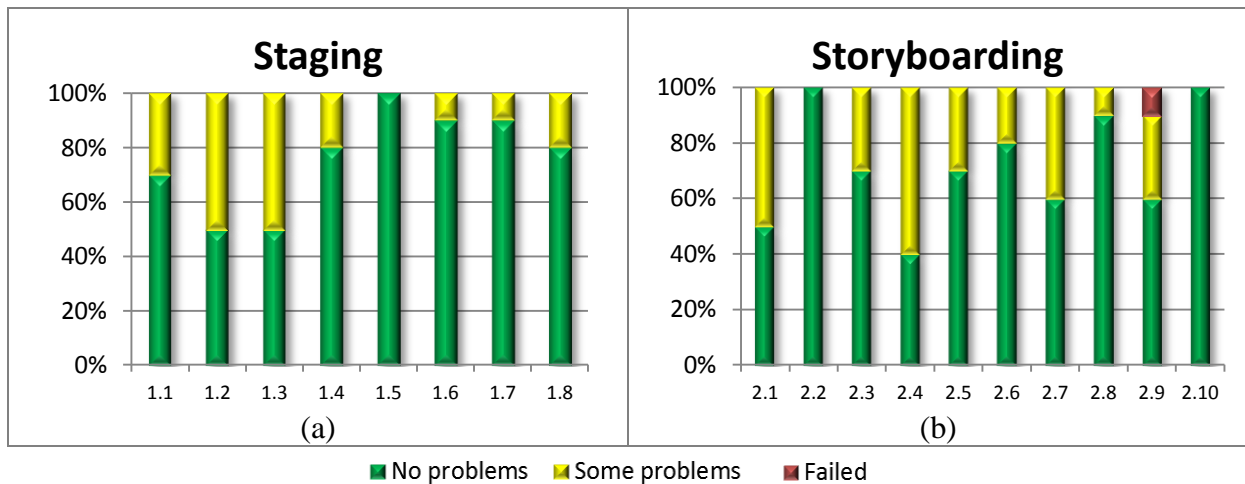


Figure 5.3: Task success rates for (a) staging, and (b) storyboarding ($n=10$).

The evaluation continued after the issue was resolved. It can be concluded that both the staging and storyboarding functionalities of the prototype design and implementation effectively supported the user's tasks due to the high success rate for each task.

5.7.1.2 Errors

Several errors were recorded while observing the participants during usability testing. The errors were recorded on the performance sheet and given severity ratings. The errors were rated as follows.

- Tasks completed without any problems were rated as “no errors”.
- Issues that did not contribute to task failure but caused the user to become annoyed or frustrated were rated with “low severity”.
- Issues that contributed to task failure but did not directly cause it were rated with “medium severity”.
- Issues that contributed directly to the failure of a task were rated with “high severity”.

Figure 5.4 (a) and Figure 5.4 (b) show the errors observed for the staging tasks and the storyboarding tasks. Most of the “low severity” errors were observed during tasks 1.2, 1.3, 2.1 and 2.4. These tasks involved sketching props, symbols and characters. User frustration was caused by recognition errors (k -NN with $k = 1$) and the automatic recognition trigger. The automatic trigger invokes the recognition algorithm after five seconds of completing a sketch. Some of the participants ($n=2$) felt that five seconds was not sufficient time and others ($n=8$) felt comfortable with the trigger delay. There was a medium severity error observed when a participant attempted task 2.4. The participant attempted to sketch the back, left arm and right arm of a character's stick figure without lifting the stylus.

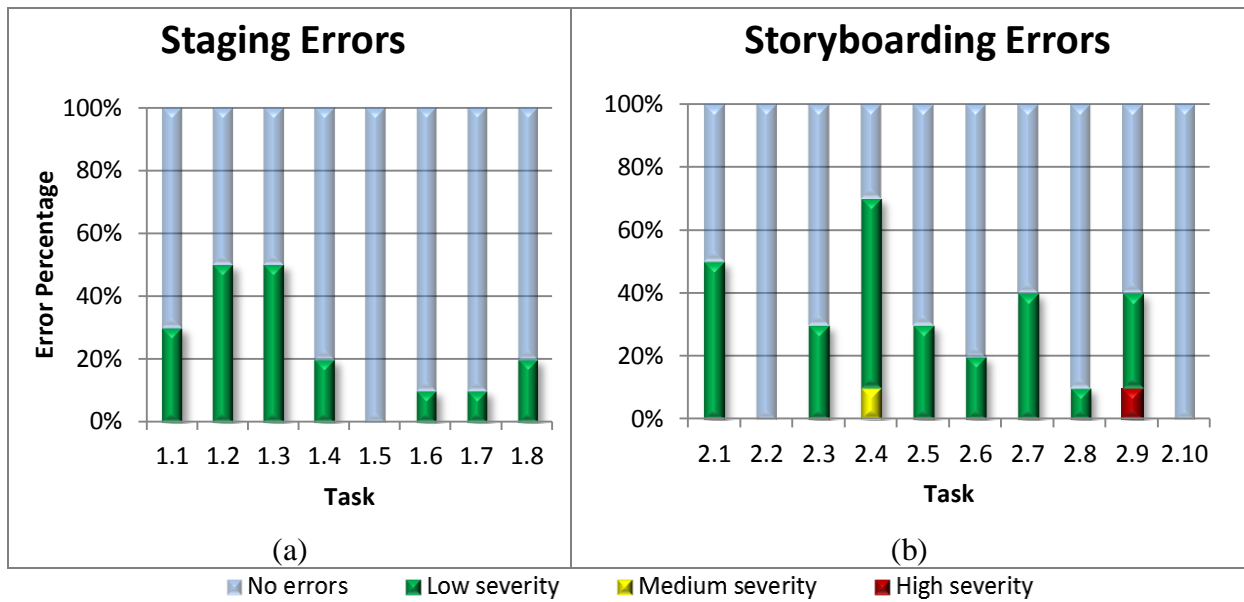


Figure 5.4: Errors made during (a) staging, and (b) storyboarding ($n=10$).

This meant that the back, left arm and right arm were represented with a single stroke. The Computer Vision Module created a tree structure to represent the character's armature as the participant sketched (see Section 4.6.5). It could not successfully construct the tree because the sketch only contained a single stroke. This is a limitation of the character pose estimation method. The participant was required to erase the stick figure and sketch the back and arms using two strokes. The high severity error in task 2.9 was caused by the prototype becoming non-responsive (the memory leak issue was resolved).

Accuracy is a key requirement for completing tasks effectively and efficiently (Stone *et al.* 2005). It was found that 100% of the staging tasks performed and 98% of all the storyboarding tasks performed were without any medium/high severity errors. It can be concluded that the participants could perform the staging tasks and storyboarding tasks accurately.

5.7.1.3 Time on Task

The time required by each participant for each task was recorded on the performance sheet and analysed. Figure 5.5 shows the average time required (in minutes) for each non-trivial task with 95% confidence intervals. Only tasks that required more than one minute to complete are shown. Outliers were removed if they were above $AVG + STD * \beta$ minutes where AVG is the average time for a task, STD is its standard deviation and $\beta = 2.2$. β was determined empirically in order to minimise data skewing. Two outlier times out of the 80 recordings made were removed.

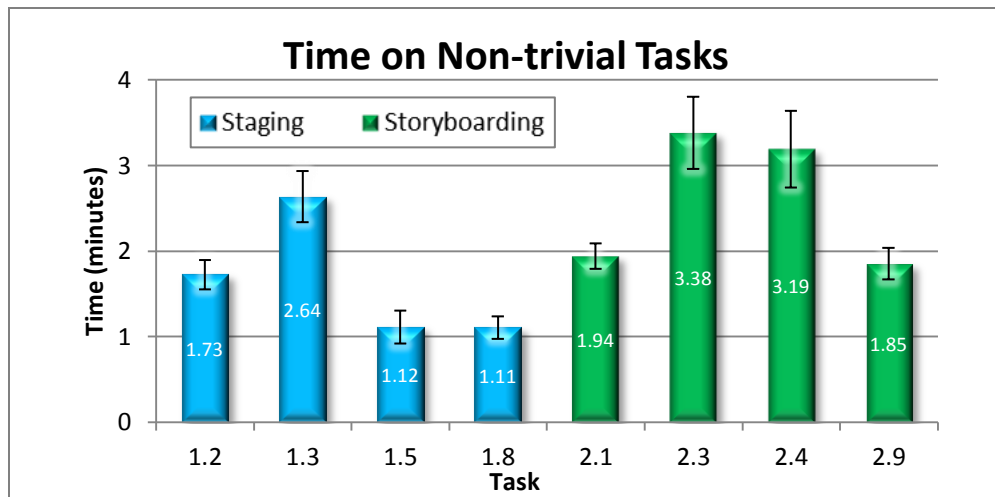


Figure 5.5: Time taken to complete non-trivial tasks ($n=10$).

Tasks 1.2, 1.3 and 1.5 involved sketching the design of the set (with three props) and the shooting strategy for the dramatic block. The participants were able to perform tasks 1.2, 1.3 and 1.5 under a total time of six minutes. This time was spent mostly on sketching, adjusting objects and correcting object recognition errors. Task 1.8 involved navigating to the storyboard and populating the generated storyboard panels with action/dialogue entries. The task required one minute on average. It was observed that two of the participants attempted to use the navigation gesture while being in the move/zoom/pan editing state. They waited to see if the system was responding and then noticed and corrected the state issue. Task 2.3 and task 2.4 required the most time because they required the participant to sketch each prop and character for the shot as it would appear in 3D. The participants achieved this in an average of seven minutes. Task 2.9 also required the user to sketch characters like task 2.4 but it required one minute less time to complete.

5.7.1.4 Learnability

Learnability is measured by examining the time required to complete tasks as the participant gains experience (Tullis and Albert 2008). This is achieved by collecting time-per-task data for multiple trial evaluations (2 to 4 trials). Each evaluation took one hour to complete. It was not feasible to conduct usability tests lasting two or three hours each. Instead, the task list was designed to include repetition of staging and storyboard sketching activities. Similar staging tasks and similar storyboarding tasks were grouped into two groups each as shown in *Table 5.1* (see Appendix H).

Table 5.1: Task grouping for measuring learnability.

Activity	Description	Group	Tasks
Staging	Add props and shots	Staging A	1.2 and 1.5
		Staging B	2.1 and 2.5
Sketching	Sketch characters and indicate facial expressions	Sketching A	2.4
		Sketching B	2.9

Figure 5.6 shows the total time required to complete each group of tasks with 95% confidence intervals. The learnability of the design accounted for a 20% reduction of the time required for performing the staging tasks as the participants became more proficient with using the GUI. A 42% time reduction was measured for performing the sketching tasks. It can be concluded that the participants were able to learn how to use the GUI as they gained more experience of using it.

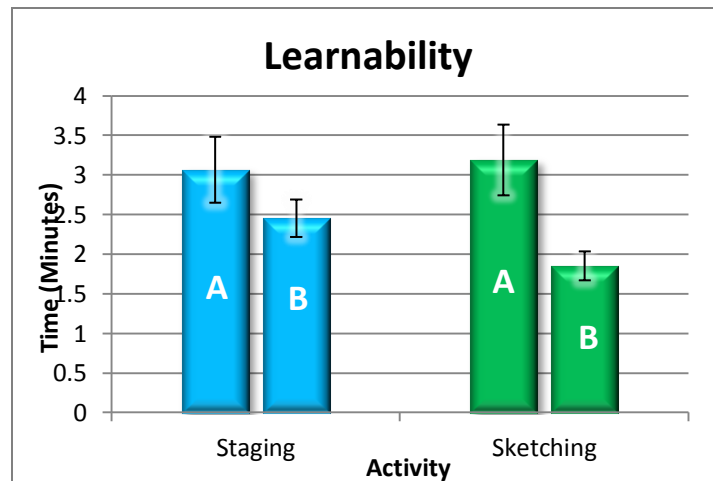


Figure 5.6: Learnability results (n=10).

5.7.1.5 Efficiency

The Common Industry Format defines efficiency as the “*level of effectiveness achieved to the quantity of resources expended*” (NIST 2001). The amount of resources expended can be measured as the time used, or the amount of effort the user expended, per task. Effort is measured as either being physical effort (e.g. number of touches, gestures and strokes) or cognitive effort. This subsection will report on the efficiency of the successfully completed tasks in terms of task time and physical effort.

Figure 5.7 shows the task efficiency measured in terms of success per minute for each non-trivial task. Only tasks with similar complexities are compared task efficiency is influenced

by its difficulty and time requirements of the task (Tullis and Albert 2008). The tasks are grouped into three levels of complexity namely “low”, “medium” and “high”.

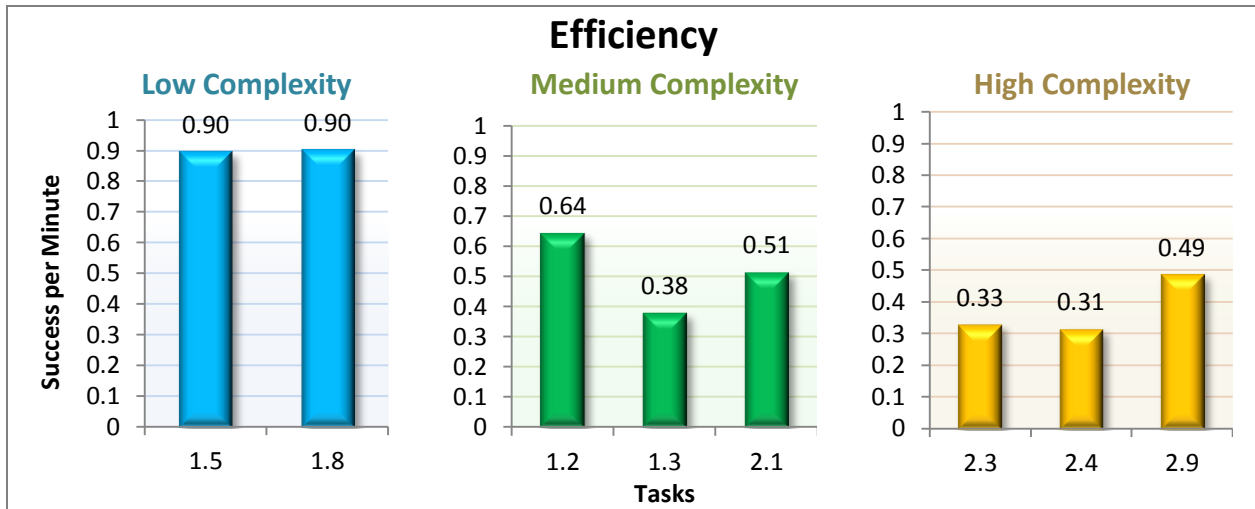


Figure 5.7: Success per minute efficiency results for non-trivial tasks ($n=10$).

Tasks 1.3, 2.3 and 2.4 had relatively low efficiency values. The efficiency of task 1.3 was reduced because of object recognition errors and automatic recognition trigger delays (it involved sketching six symbols). The efficiency of task 2.3 was reduced because it had high time requirements (it involved sketching a 3D scene with three props). The efficiency of task 2.4 was reduced because of a thresholding issue with the automatic interpretation of user stick figures (see Section 5.7.3). The overall average efficiency was calculated for the non-trivial tasks. The efficiency percentage for each level of task complexity was calculated using Equation (5.1) (Tullis and Albert 2008).

$$\text{Efficiency Percentage} = \frac{\text{Average efficiency}}{\text{Maximum efficiency}} \quad (5.1)$$

Table 5.2 shows the average efficiency for each complexity level. It was found that the average efficiency for performing non-trivial tasks was 86%. It can therefore be concluded that the participants were able to perform the tasks efficiently in terms of task success per minute.

Table 5.2: Calculation of the average efficiency for non-trivial tasks.

Complexity Level	Maximum Efficiency	Average Efficiency	Efficiency Percentage
Low	0.9	0.9	100%
Medium	0.64	0.51	80%
High	0.49	0.38	78%
Average Efficiency			86%

Efficiency was also measured in terms of the physical effort expended. This was achieved by recording the number of gesture events and stylus events invoked by the GUI as each participant performed the tasks. The data was stored each time the participant navigated away from a screen. The event data was grouped into two groups for staging and storyboarding similarly to the previous subsection, namely “staging A”, “staging B”, “sketching A” and “sketching B”. Figure 5.8 (a-e) shows the average number of taps, drags, manipulations, stylus strokes and stylus release events that were invoked for each group with 95% confidence intervals. It is interesting to note that the physical effort required for staging and storyboarding reduced from A to B for most event types. The manipulation effort increased slightly from “sketching A” to “sketching B”. This is because the participants preferred seeing an overview of the sketch during the “sketching A” tasks. The participants adjusted the view of the sketch more often during the “sketching B” tasks. It can be concluded that the participants became more efficient in terms of physical effort as the GUI was learned.

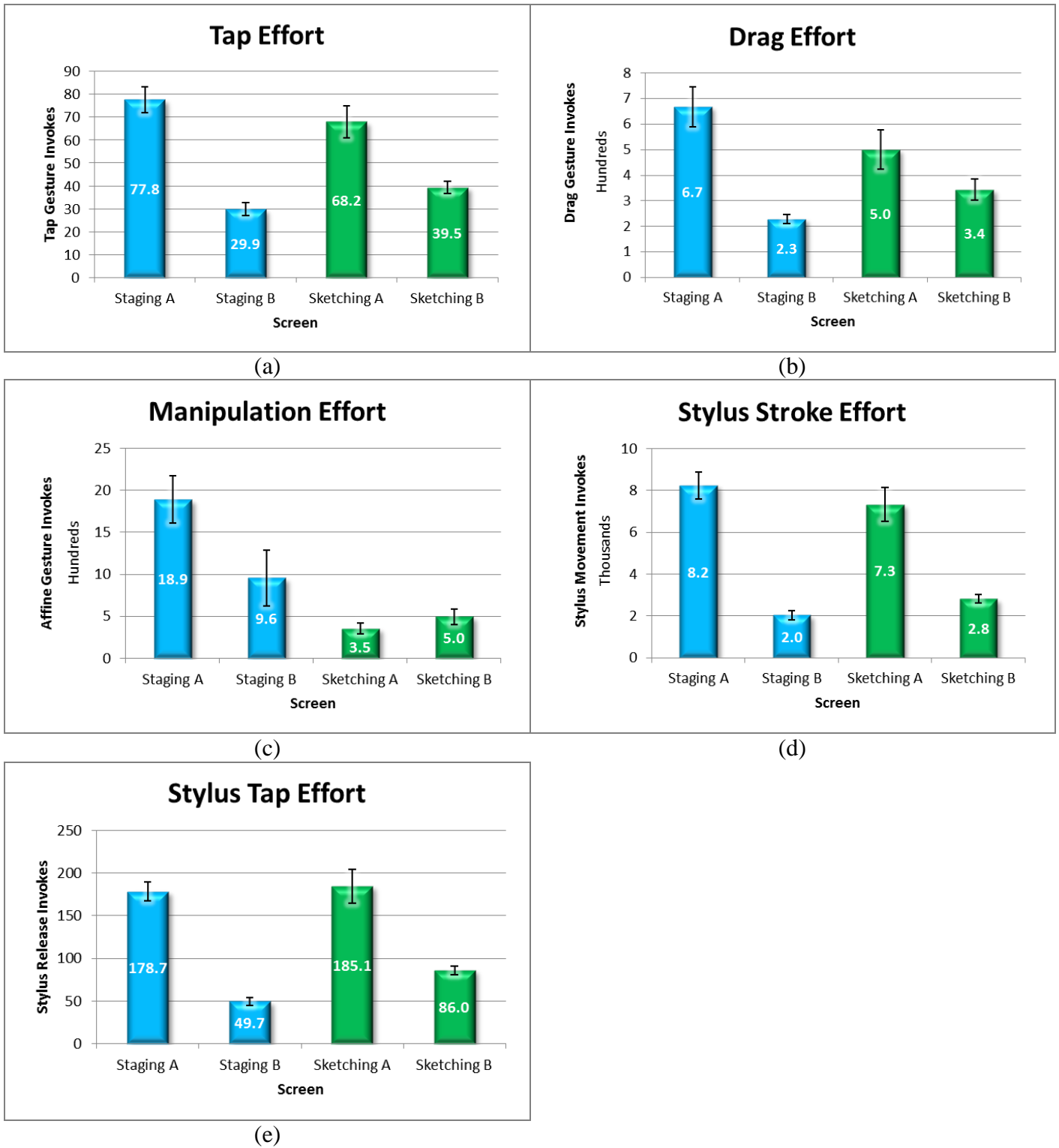


Figure 5.8: Physical effort efficiency results (n=10).

5.7.2 Satisfaction Results

This subsection discusses the satisfaction results obtained from analysing the data from the user satisfaction questionnaires for the staging activities and the storyboarding activities (see Appendix K and Appendix L). User satisfaction was measured in terms of cognitive load, overall satisfaction and usability using 5-point semantic differential scales and 5-point Likert scales. The user satisfaction questionnaires also included sections for collecting qualitative data describing positive, negative and general aspects of the GUI. This subsection reports on the common themes identified from the qualitative data.

5.7.2.1 Workload

The cognitive load was low for both the staging tasks and the storyboarding tasks (see Figure 5.9). The average ratings given for the mental demand and physical demand were in the “low” rank vicinity. The physical demand of the storyboarding tasks was higher than the physical demands of the staging tasks. Similarly, the storyboarding tasks required more effort and time, and caused more user frustration compared to the staging tasks. This can be explained by the fact that sketching shots in 3D perspective requires more skill, effort and time than sketching 2D symbols for a floor plan. This is supported by the efficiency data collected (see Section 5.7.1).

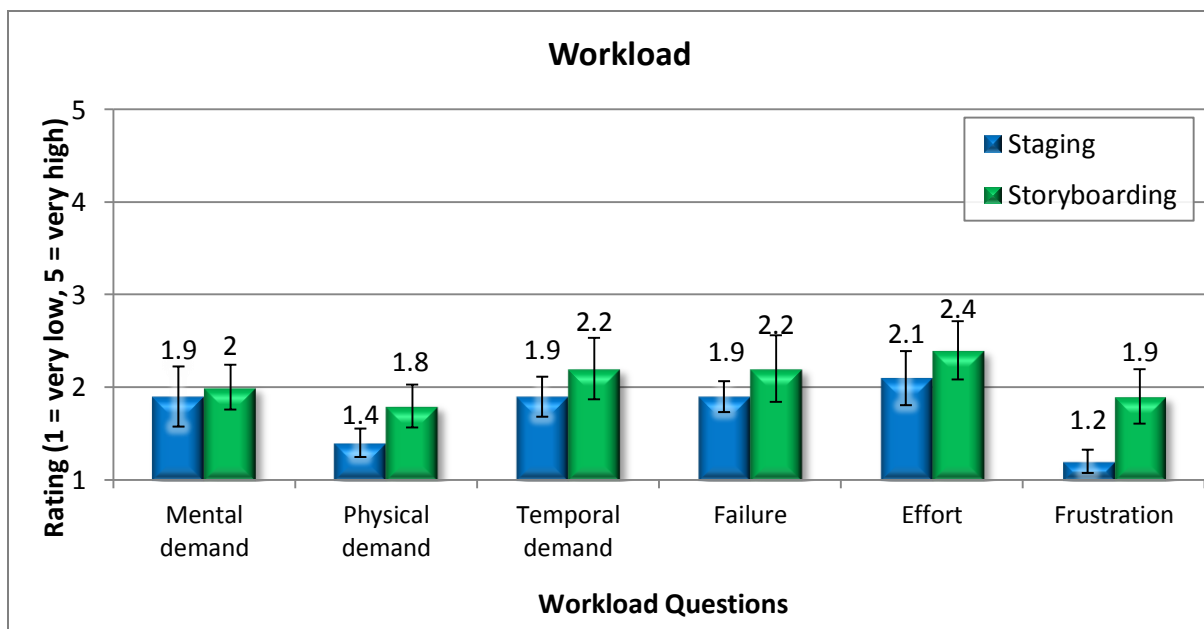


Figure 5.9: The cognitive load using a 5-point semantic differential scale ($n=10$).

The staging tasks received the lowest frustration rating, suggesting that it was easy for the participants to sketch the floor plans. Overall, the participants rated their performance

(success rates) for the staging tasks and the storyboarding tasks as being, on average, better than “high” (shown with a “low” average failure rating in Figure 5.9). It can be concluded that the participants reported to be able to successfully complete the tasks while being under low levels of cognitive and physical load.

5.7.2.2 Overall Satisfaction

Overall user satisfaction usability metrics were measured using 5-point Likert scales. Figure 5.10 shows the overall satisfaction reported by the participants for the staging tasks and the storyboarding tasks with 95% confidence intervals.

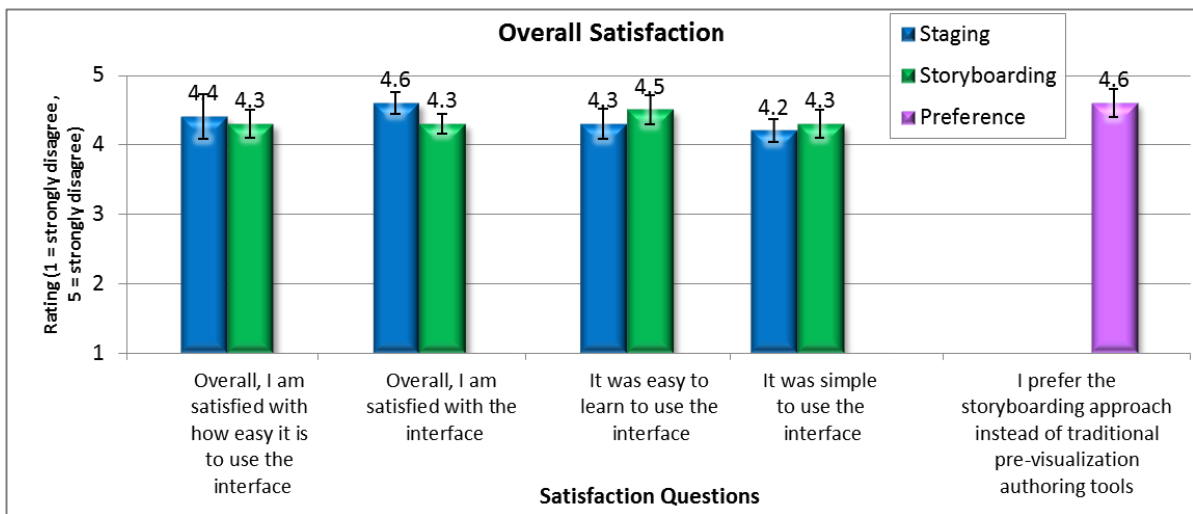


Figure 5.10: Overall satisfaction results using a 5-point Likert scale ($n=10$).

The participants’ average response on the ease of use, overall satisfaction, learnability and simplicity of the staging and storyboarding GUI components were all positive (average was 4.4 = good). The overall satisfaction section of the storyboarding questionnaire also contained a question regarding the preference between the storyboarding approach and the conventional pre-visualisation authoring approach.

An average rating of 4.6 was given in favour of the storyboarding approach. Two participants mentioned that they would have preferred having access to the conventional pre-visualisation authoring methods *and* the proposed storyboarding approach. The following statements were made by participants to support the high rankings they gave the storyboarding approach:

- “The ability to use the 2D and 3D interface. This renders paper obsolete.”
- “...a great fusion of paper sketching and 3D modelling.”
- “This process makes much more sense.”
- “This will make the production process much easier for both skilled and non-skilled users”

- “Most of the other authoring tools are truly complex to learn..., but with the sketched-based authoring it was the opposite.”
- “Most of those (other) programs aren’t as simple to use...”

5.7.2.3 Usability Results

The satisfaction questionnaire was used to collect self-reported data about the usability of the sketch-based interface for staging and storyboarding on 5-point Likert scales. Figure 5.11 shows the self-reported usability results with 95% confidence intervals.

The participants reported an average rating of 4.1 = good for how easily, efficiently, productively and effectively the staging and storyboarding tasks could be performed. The participants also reported that the staging and storyboarding interfaces had the functionality they expected and that they were comfortable with the touch interaction provided by the GUI. The slight differences between the ratings given between the staging and storyboarding interfaces are motivated by analysing the qualitative feedback provided by the participants.

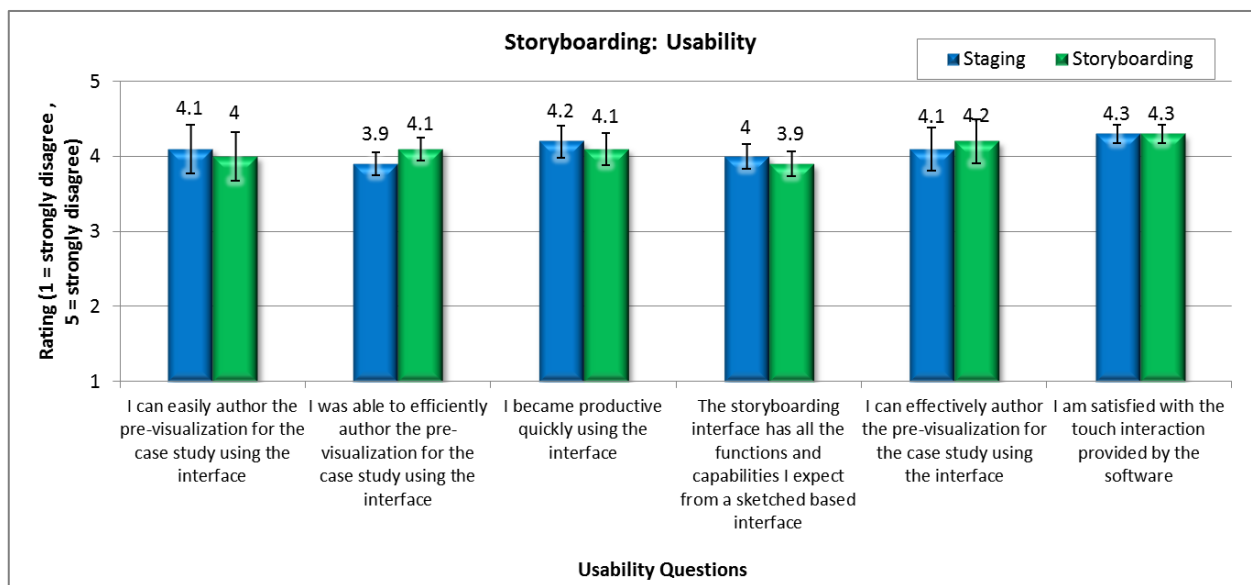


Figure 5.11: Usability results using a 5-point Likert scale (n=10).

5.7.2.4 Qualitative Feedback

The post-task user satisfaction questionnaires included sections for commenting on the most positive, the most negative and general aspects of the staging interface and the storyboarding interface. Analysis of the qualitative data revealed several common themes. General comments were moved to the sections for positive and negative comments respectively.

Table 5.3 shows the most positive comments made regarding the staging interface. Five participants commented that the props, characters and shots were easy to draw using the floor plan editor. This was supported by four participants commenting that the staging interface

does not require artistic talent. Six participants commented that the staging interface was simple, fun to use and easy to learn.

Table 5.3: Most positive comments for the staging interface.

Description	n
Simple and fun to use.	6
Easy to learn.	6
Easy to draw props, characters and shots to place them in the scene.	5
No artistic talent required.	4
Good visual feedback.	3
The icon for each function is easy to understand.	3
Easy to navigate between screens.	2
Would make production simpler.	2

Table 5.4 shows the comments that were made about the most negative aspects of the staging interface. In general, the participants commented about the lack of double tap support (n=1), difficulties recognising character symbols (n=1) and non-responsiveness during the rendering of relatively hi-polygon scenes (n=1). Two participants commented on the limited number of props and characters provided. The proof of concept prototype was implemented in order to support the Goldilocks scenario for user evaluation. Additional 3D assets can be included in the environment by creating the 3D models, adding them and training them (see Section 4.4 and Section 4.6.4). Five participants commented that they would like more functions in general, e.g. undo/redo.

Table 5.4: Most negative comments for the staging interface.

Description	n
Would like more functions.	5
Limited number of assets.	2
No double tap gesture for accepting options.	1
Difficulties recognising character symbols.	1
Lags when rendering highly populated scenes.	1

Table 5.5 shows the most positive comments that were made about the storyboarding interface. Four participants commented that being able to automatically convert user sketches into 3D shots is a useful feature. Five participants found the storyboarding interface to be easy to use and learn. Four participants reported that it was efficient to use as well.

Table 5.5: Most positive comments for the storyboarding interface.

Description	n
Easy to use.	5
Easy to learn.	5
Converting sketches to 3D is a useful feature.	4
The storyboarding interface is efficient to use.	4
Organisation and manipulation of objects is easy.	3
It is simple to place characters.	2

Table 5.6 shows the most negative comments made about the storyboarding interface. It was commented that there is no double tap gesture for accepting options (n=2) and that the storyboarding interface needs more functions, e.g. undo/redo. One of the participants commented that it was difficult to sketch in 3D perspective and that it would be easier to use a drag-and-drop approach so that sketching would not be necessary for creating shots. The focus of this research is to investigate sketch-based methods. Implementing and evaluating a drag-and-drop approach as well was therefore out of scope. Two participants commented that the props and characters had low visual appeal and that *“it would be ideal if the visuals, ability and options matched that of the Sims game”* (Electronic Arts 2009). Low polygon 3D models were used because of the hardware limitations of the tablet computer. It did not provide any dedicated graphics acceleration. Limited graphics acceleration was supported by the integrated graphics chipset provided by the Intel Core i5 processor (Intel Corporation 2012).

Table 5.6: Most negative comments for the storyboarding interface.

Description	n
No double tap gesture for accepting options.	2
The assets have low visual appeal and more of them are desired.	2
The storyboarding interface needs more functions.	1
It is difficult to sketch in a 3D environment.	1

5.7.3 Observations

The participants were observed during usability testing in order to identify usability issues. The usability issues were categorised and given severity ratings (low, medium, high) similar to the ratings given during the measurement of the task success metric.

Table 5.7: Observed usability issues (n=10)

Category	Issue	Severity	n
A.) Symbols	1) Large bounding box, for symbols/props	high	1
B.) Stick figures	1) Stick figure snap thresholds	medium	4
	2) Stick figure head and overlap	medium	2
C.) Touch	1) Double tap to submit/activate	low	2
	2) Select character on view touch	low	1
D.) 3D viewing	1) 3D viewing axis inversion	low	3
E.) GUI state	1) Mode for manipulation/gesture	low	6
	2) New layer for annotations misunderstanding	low	5
	3) Forgetting to add layers for each prop/character	low	3

Table 5.7 summarises the usability issues observed. There was one issue identified with high severity in the “symbols” category where the symbols sketched by the user could not be differentiated. The issue was caused by the way the prototype implemented symbol bounding. The user’s strokes were used to indicate the bounds of each symbol. The issue was that one participant preferred sketching each symbol five or six times its normal size on the floor plan. The boundaries of each symbol overlapped and the prototype was placed in an erroneous state, thereby causing task failure. This issue was resolved by using the boundary area of the recognised object and not the user’s sketch. The user could then sketch the props and symbols at any scale without compromising the state of the floor plan editor.

Two medium severity issues were identified when the participants sketched stick figures in order to add characters to a shot. Issue B.1 was caused by thresholds that determine when the strokes should be attached to the vertices of the user’s stick figure. It was found that when the participant sketched a small stick figure for a distant character then the stroke connecting thresholds were too large. Conversely, when the characters were sketched to be close to the camera, then the thresholds were too small. This issue can be resolved by dynamically adjusting the thresholds according to the scale of the stick figure.

Issue B.2 was caused by a limitation in the character pose estimation method. The Computer Vision Module creates a tree structure to represent the character’s armature as the participant sketches (see Section 4.6.5). If the user attempts to sketch the stick figure’s head using a circle or attempts to sketch the figure using one stroke then the algorithm cannot correctly

associate the stick figure with the armature. The participants were required to erase the stick figure and sketch the neck, back, arms and legs with separate strokes.

Three low severity, touch related issues were identified. Two participants attempted to perform a double tap gesture to select and open story items from the story viewer (issue C.1). This issue can be resolved by incorporating the double tap gesture into the multi-touch module and the GUI implementation. One participant attempted to select a character layer from the Sketch Editor by touching the character in the working area instead of the list of layers (issue C.2). This issue can be resolved by using a picking method for 3D object selection from the rendered image (e.g. the selection buffer in OpenGL).

Issue D.1 involved moving and rotating the camera in 3D spacing using the drag gesture. It was observed that three participants tried to move the camera towards a direction by dragging in the same direction. The method for panning the view of a 3D scene was designed to be consistent when panning 2D views. The participants were able to orientate themselves in 3D space when interacting with the touch based GUI. The issue can be resolved by including a function that allows the participant to choose whether camera movement and rotation should be inverted along the horizontal and vertical axis or not.

There were several GUI state related, low severity issues. The screens that required sketch-based input allowed the participants to pan the view of the current sketching area. The “pan/zoom” toggle was used to distinguish between panning gestures and navigation gestures. In issue E.1, six participants forgot about this GUI state and attempted to navigate while the GUI was in a “pan/zoom” mode. The participants noticed this and disabled the toggle to navigate. The issue can be resolved by removing the “pan/zoom” toggle and restricting flick gestures to the tabs on the sides of the screen.

Issue E.2 and issue E.3 involved layer management. Five participants thought that a new layer was required for creating an annotation symbol and three participants forgot to add new layers between sketching props. The layer concept was introduced in order to allow the user to sketch stacked props, e.g. sketching a bowl on a table. However, it placed additional cognitive load on the participants. Issue E.2 and issue E.3 can be resolved by removing the layer concept from the floor plan editor and using bounding boxes to distinguish props (similarly to how symbols are distinguished). A “stack” toggle can then be added to allow the user to sketch stacked props when required.

5.8 Design Guidelines

Several design recommendations can be made based on the results from the analytical and empirical evaluations. The lessons learned from the evaluation have been compiled into ten guidelines for designing sketch-based pre-visualisation authoring tools with a storyboarding approach. The design guidelines are as follows:

1. Design for sketch-based input based on real world deliverables

The design of the GUI should be based on real world deliverables such as scripts, floor plans and storyboards. The tool should be able to analyse sketch-based input automatically in order to interpret the user's sketches of props, characters and floor plans. Algorithms that interpret the user's sketches should be executed in separate threads if they are computationally expensive. The user should be able to sketch at interactive rates without abrupt interruptions. It is also important that the algorithms used by the Computer Vision Module are sufficiently robust in order to process rough sketches (see Section 4.6). This will allow artistic and non-artistic users to be able to use the sketch-based pre-visualisation authoring tool.

2. Minimise clutter in user sketches

The GUI components that accept sketch-based input should provide a means for reducing the cluttering of user sketches. The view of the working area should be adjustable using zoom and pan gestures so that participants can sketch the props and symbols at the correct scale. Grids should be provided in order to provide the user with an indication of scale. If objects are recognised then their sketches should be replaced by the recognised object. The 3D camera of a storyboard panel should be known before changing the user's 3D sketches into 3D objects. Sketch editors that require users to sketch in 3D should support the layering concept so that props and characters can be sketched, manipulated and analysed separately.

3. Provide help so that the sketched objects can be recognised successfully

The GUI should provide built-in help features to aid the user in sketching objects and using the system. This can be achieved by providing animated tutorials. A "show me how" feature can be included to demonstrate how the stylus can be used to sketch props, symbols and characters so that they can be recognised and posed effectively.

4. Provide user controlled recognition triggers

Automatic recognition triggers should be optional and the main recognition trigger should be user controlled. This can be achieved by providing a “recognise all” button. The recognition processing can work in the background without requiring the user to begin the process, but it should be the user’s responsibility to change the sketches into their respective objects. An optional time-delayed automatic recognition trigger can be provided, but it should not be the only option.

5. Provide comprehensive editing functionality for the sketching interface

The interface components requiring sketch-based input should support common editing functionalities provided by editing software, such as selecting, clipboard functionality and undo/redo functionality. The space available to include these touch-friendly tools may be limited. If this is the case then a tabbed toolbar can be used to organise the tools. If the sketching interface allows 3D viewing then the user should be able to choose whether movement/rotation around the axis should be inverted, depending on the user’s preference.

6. Design for touch-based input

The GUI should be designed to be touch-friendly by following touch interaction design principles (see Section 4.5.1). The GUI should be simple, consistent, minimalistic and aesthetic. Informative icons should be used instead of unnecessary text. Intuitive and logical touch gestures should be provided for interacting with the various components of the GUI.

7. Allow props, characters and symbols to be sketched in any fashion

The method used for interpreting user sketches automatically should allow the user to sketch props, characters and symbols in any fashion. The method used for recognising objects and estimating their pose should not be dependent on how the objects are sketched. For example, if stick figures are used to represent characters then the user should be able to sketch the parts of the stick figure in any order.

8. Minimise the number of GUI states

The design of the touch/sketch-based GUI should minimise the number of states required to interact with the tool. The GUI should be designed so that the touch gestures are specific to relevant GUI components so that a “navigate/manipulate” state is not needed. In addition to minimising touch interaction states, the GUI should only apply the layering concept in the Sketch Editor. Having layers for individual props in the Floor Plan Editor adds to the cognitive load on the user. The props’ bounding boxes can be used to differentiate props. A “stack” toggle can be used to allow props to be placed on top of one another, e.g. bowls on a table.

9. Reduce cognitive effort required for sketching the storyboard panels

The user should not be required to maintain a mental image of the props on the floor plan while sketching a new shot. The cognitive effort required for remembering where props are relative to each other should be reduced. This can be done by adding a floor plan preview to the Sketch Editor. The user should be able to toggle the visibility of the floor plan preview.

10. Provide touch-based alternatives for adding props, characters and shots to the scene

Additional touch-based methods should be available to users for adding props and symbols to the floor plan. For example, performing a long press gesture at a point on the set can show a context menu for the user to select a prop, character or shot. The user can then use the drag and rotate gestures to adjust the pose of the object. This provides an alternative method for users who prefer not to sketch the individual objects during the staging activity.

These design guidelines are supported by the findings of the analytical and empirical evaluation of the sketch-based prototype tool for authoring pre-visualisations using a storyboarding approach. Guidelines 1 to 6 are based on existing heuristics for designing sketch-based GUIs that support touch interaction (see Section 5.5). Guidelines 7 to 10 are based on the findings of the empirical evaluation. They extend existing design heuristics by requiring that objects be sketched in any fashion, that the number of GUI states is minimised, that the cognitive effort required for sketching shots is reduced and that touch-based alternatives are available for adding objects to the scene.

5.9 Conclusions

This chapter answered the fourth research question listed in Chapter 1, namely *to what extent does a sketch-based storyboarding interface support the user in authoring pre-visualisations in terms of software usability?* The chapter answered this research question by providing an analytical and empirical evaluation of the GUI design discussed in Chapter 4.

The results from the analytical evaluation showed that the design followed proven guidelines for overall GUI design, touch interaction and sketch-based recognition. The usability of the proof of concept prototype was evaluated empirically through usability testing with ten participants. Usability was measured in terms of effectiveness, efficiency and user satisfaction. It was found that the prototype allowed users to effectively author pre-visualisations, with an average success rate of 99% for non-trivial tasks. 100% of the staging tasks and 98% of all the storyboarding tasks were completed with only minor errors. The participants could complete the tasks efficiently in terms time and effort. It was also found that the time and effort required decreased as the participants learned to use the prototype. This shows that the prototype was easy to learn.

User satisfaction was measured using satisfaction questionnaires. The participants responded positively by reporting that the prototype required low cognitive effort to use. The participants also responded that they could achieve high performance levels without feeling frustrated or pressured for time. The average response on the ease of use, overall satisfaction, learnability and simplicity of the staging and storyboarding GUI components was 4.4 out of five. Positive usability responses were reported with an average rating of 4.1 out of five. The positive results show that the prototype design supports the user in terms of software usability.

The participants were asked whether they preferred using conventional pre-visualisation authoring approaches, the proposed sketch-based storyboarding approach or a combination of the two approaches. Interestingly, an average rating of 4.6 out of five was given in favour of the storyboarding approach. This chapter concluded by proposing ten design guidelines based on the lessons learned from the evaluation (see Section 5.8).

The next chapter concludes the dissertation by discussing how the objectives of this research were achieved and what theoretical and practical contributions were made. The chapter also proposes future research opportunities.

Chapter 6:

Conclusions

6.1 Introduction

The main objective of this research was to investigate how sketch-based user interfaces and methods from computer vision can be used for supporting pre-visualisation authoring using a storyboarding approach. Problems with existing pre-visualisation authoring tools were identified and a sketch-based storyboarding approach was proposed in order to address these problems. Appropriate computer vision methods were identified and incorporated in the design of a new framework, namely SISPA, for the proposed approach. A proof of concept prototype was implemented for the SISPA framework so that a usability evaluation could be conducted in order to measure the extent to which the approach supports the user in authoring pre-visualisations. The findings of the evaluation were used to propose a set of guidelines for designing future sketch-based pre-visualisation authoring tools that utilise computer vision methods in order to automatically interpret user sketches.

This chapter concludes this research by discussing how the research objectives were achieved and summarising the theoretical and practical contributions made. The limitations of this research and the problems that were encountered are also discussed. The chapter closes by making recommendations for future research in sketch-based pre-visualisation authoring.

6.2 Achievement of Research Objectives

The thesis statement for this research is as follows, as presented in Chapter 1:

Storyboarding with sketch-based interfacing techniques can be used effectively for authoring pre-visualisations without impairing software usability.

The main objective of this research was to investigate how sketch-based user interfaces and methods from computer vision can be used for supporting pre-visualisation authoring using a storyboarding approach. This section discusses how the following sub-objectives were met in order to achieve the main research objective:

- O₁: To investigate how pre-visualisations are currently created by focusing on the methods and theories already developed.
- O₂: To investigate what methods are available for extracting information from sketches.
- O₃: To design and implement a sketch/touch-based storyboarding tool for authoring pre-visualisations.
- O₄: To measure the extent to which a sketch-based storyboarding interface supports the user in authoring pre-visualisations in terms of software usability.

6.2.1 Achievements

This research showed that sketch-based interfacing techniques can be used effectively for authoring pre-visualisations without impairing software usability. This was supported by addressing the research questions presented in Chapter 1 in order to address the above-mentioned research objectives.

Q₁: What methods and theories have been developed for authoring pre-visualisations?

The first research question was answered in Chapter 2. This involved reviewing the five phases of the filmmaking process namely, development, pre-production, production, postproduction and distribution. The chapter included a detailed review of the activities performed during the pre-production phase, namely script analysis, staging, storyboarding and pre-visualisation. It was found that floor plans are useful for documenting the shooting strategy for each individual dramatic block (see Section 2.4 and Table 2.5). The review also discussed how storyboard artists illustrate characters, emotions, environments and dynamics using sketches, annotations and graphical devices (see Section 2.5 and Table 2.6). This investigation was necessary in order to align the proposed authoring approach with existing filmmaking methods and theories.

The chapter reviewed textual and graphical pre-visualisation authoring approaches, including game-engine code, control languages, Director Notation, modelling and animation tools, simplified animation tools and sketch-based animation tools. It was found that each approach provided an advantage over the previous approach and that existing pre-visualisation tools

focus on authoring detailed animated pre-visualisations using click-and-drag interaction methods. The most intuitive and user friendly approach was identified to be the sketch-based storyboarding approach. It makes the transition from storyboarding to pre-visualisation authoring smoother in the sense that sketching a storyboard and sketching a pre-visualisation is almost the same thing. The need for such an authoring tool was confirmed during an interview with a producer from Triggerfish Animation Studios. A set of requirements was identified from the literature review and the interview and summarised in Section 2.8. In particular, it was determined that the tool should be able to interpret the sketches made by the artist during staging activities and storyboarding activities. It was therefore necessary to investigate methods for extracting information from user sketches.

Q₂: What methods are available for extracting information from sketches?

The second research question was answered in Chapter 3 by discussing methods from computer vision literature for extracting information from user sketches. Two main challenges were identified. Firstly, there is no texturing information available because the user's sketches are not shaded and they consist primarily of minimalistic strokes. The second challenge is that the sketches may contain anomalies such as inconsistent, missing or unnecessary strokes (see Section 3.3.2.4). The feasibility of each method for operating on user sketches was discussed. The methods included approaches for describing images, comparing images and estimating the pose of rigid bodies and articulated bodies.

The chapter reviewed several feature detectors and descriptors (see Section 3.3). It was established that Blob-based region detectors, such as the DoG detector, were the most stable approach for detecting local features from sketches. It was also found that the best approach to describing sketch features is to combine feature description approaches. SIFT-based descriptors can be combined with Grid-based descriptors in order to describe sketches reliably. Literature suggested that shape-based local feature descriptors are also suited for describing outlined images such as sketches.

Approaches for matching features and classifying user sketches were reviewed from literature. It was determined that the bag of features approach and structured feature matching approaches are both suitable for finding feature correspondences. Bayesian classifiers, ANNs, SVMs, decision trees and k -NN classifiers were considered. It was determined that the k -NN classification approach is the particularly suitable for classifying user sketches. This is because the k -NN classifier makes it possible to classify images based on their local features.

The images do not have to be represented with single, high-dimensional vectors. The k -Nearest-Neighbour classification approach is particularly suitable for classifying user sketches from limited training data.

Several approaches were reviewed for estimating the pose of rigid bodies (such as props) and articulated bodies (such as characters). It was determined that pose recognition is not suitable for estimating the pose of a rigid body because of front/back ambiguity and symmetry. The sketches are also likely to contain the above-mentioned anomalies (see Section 3.3.2.4). The literature review revealed that numerical rigid body pose estimation approaches such as POSIT provide more stability than analytical approaches (see Section 3.5). Based on the requirements identified in Chapter 2, it was determined that vision-based articulated body posing methods are not suitable for interpreting the characters, because they will be sketched using minimalistic stick figures. It was determined that the direct sketch-based articulated body pose estimation approach should be used instead in order to provide the user with a quick and easy method for sketching characters.

Q₃: How can the authoring of pre-visualisations be supported using a storyboarding metaphor and sketch/touch-based interfacing techniques?

The third research question was answered in Chapter 4 by proposing a new framework called SISPA for improving the user interface layer of the general pre-visualisation framework (see Section 4.2. and Section 4.3). The design and implementation of a proof-of-concept prototype that implements the components of the SISPA framework was discussed.

A touch-friendly Graphical User Interface (GUI) design was proposed for storyboarding on a tablet computer. A GUI component layout supporting the filmmaking activities identified in Chapter 2 was discussed, as well as an intuitive method for navigating between them. The components requiring sketch-based input were designed to utilise the computer vision approaches identified in Chapter 3 for extracting information from sketches.

Chapter 4 discussed several methods for describing the contents of user sketches using the approaches reviewed in Chapter 3. It was found that the best approach for describing the user's sketch was to combine a single high resolution disk (grid-based) and a low resolution disk for describing sketched objects after they have been normalised using Principal Component Analysis (PCA). The resulting descriptor was found to be invariant to scaling. The Iterative Closest Point (ICP) registration and discreet disk rotation methods were applied to achieve rotational invariance. The image correspondence was determined using the

Difference of Gaussian (DoG) local region-based feature detector. The SIFT feature descriptor was used to match features in order to find 2D-3D point correspondences. The correspondences proved to be useful for rigid body pose estimation with a known or unknown camera. Object recognition was achieved by using k -NN classifier with the mean SIFT error as the distance function. An algorithm was proposed for estimating the camera for a user sketch by combining information from the floor plan with the pose data estimated for each prop in the user sketch. A discussion was included on how the proposed touch-friendly GUI could be implemented on a multi-touch tablet using OpenGL and OpenCV.

Q₄: To what extent does a sketch-based storyboarding interface support the user in authoring pre-visualisations in terms of software usability?

The fourth research question was answered in Chapter 5 by providing an analytical and empirical evaluation of the proof-of-concept prototype discussed in Chapter 4. Ten design guidelines were proposed, based on the lessons learned from the analytical and empirical evaluation (see Section 5.8).

The results from the analytical evaluation showed that the design followed proven guidelines for overall GUI design, touch interaction and sketch-based recognition interfaces. The usability of the proof-of-concept prototype was evaluated empirically through usability testing using ten participants. 40% of the participants were skilled at sketching storyboards and 70% of the participants had experience with using 3D modelling and animation tools. The usability of the prototype was measured in terms of effectiveness, efficiency and user satisfaction. The evaluation involved performing a set of tasks for authoring a pre-visualisation in a small scenario. Tasks were included for staging, storyboarding and sketching.

The usability testing showed that the prototype allowed users to effectively author pre-visualisations using the prototype with an average success rate of 99% for non-trivial tasks. It was found that the participants could complete the tasks efficiently in terms of time and effort (physical and cognitive). It was also found that the time and physical effort required decreased as the participants learned to use the prototype. This showed that the prototype was easy to learn.

The participants responded positively on the user satisfaction questionnaires by reporting that they could achieve high performance levels with minimum effort without feeling frustrated or pressured for time. The participants also reported that the prototype required low cognitive and physical effort to use. The average response on the ease of use, overall satisfaction,

learnability and simplicity of the staging and storyboarding GUI components was 4.4 out of five. Positive usability responses were reported with an average rating of 4.1 out of five.

The positive performance and user satisfaction results show that the design discussed in Chapter 4 supports the user in terms of software usability. The participants were also asked whether they would prefer using conventional pre-visualisation authoring approaches, the proposed sketch-based storyboarding approach or a combination of the two approaches. Interestingly, an average rating of 4.6 out of five was given in favour of the storyboarding approach. Two of the ten participants responded in favour of a combined approach.

6.2.2 Summary

The research objectives defined in Chapter 1 were supported by this research. It was shown how a sketch-based pre-visualisation authoring tool that uses a storyboarding approach could be designed and implemented. The results of the prototype evaluation showed that the approach supports the user in pre-visualisation authoring. This result supports the thesis statement of this research. This research resulted in the following achievements:

- Investigation of the filmmaking process and the activities performed during the pre-production phase;
- Identification of several problems with existing pre-visualisation authoring approaches;
- Identification of the requirements for sketch-based pre-visualisation authoring tools that employ the storyboarding approach;
- Selection of the most appropriate computer vision methods for extracting information from user sketches in order to recognise and pose objects in the user's sketches;
- Development of a framework (SISPA) for improving the user interface layer of the general pre-visualisation framework using a sketch-based storyboarding approach;
- Design and implementation of a proof-of-concept prototype tool that implements the components of the SISPA framework;
- Evaluation of the usability of the prototype tool and the extent to which the proposed approach supports the user in authoring pre-visualisations; and
- Proposal of a set of guidelines for designing sketch-based pre-visualisation authoring tools that automatically interpret user sketches and follow the storyboarding approach.

6.3 Research Contribution

This research has several theoretical and practical contributions. The theoretical contributions are related to the general pre-visualisation authoring framework. It also contributed by showing how computer vision and sketch/touch-based interaction methods could be used to support pre-visualisation authoring. The practical contributions are related to the implementation of the approaches for supporting sketch-based pre-visualisation authoring.

6.3.1 Theoretical Contributions

A set of requirements for sketch-based pre-visualisation authoring was identified from filmmaking literature and an interview with an animation studio called Triggerfish Animation Studios (see Section 2.8). The requirements included a list of objects, symbols, annotations, graphical devices and other elements that are used in floor plans and storyboards during the pre-production phase. It also included requirements regarding functionality, usability, information and semantics and technical aspects. These can be used as a guideline for determining what elements a sketch-based pre-visualisation authoring tool should support.

A new framework namely SISPA was proposed for the user interface layer of the general pre-visualisation authoring framework. The SISPA framework includes the following main components (see Figure 6.1):

- *Data Manager*: The Data Manager is responsible for transferring data between the data collection component, the GUI and the planning layer.
- *Data Collection*: The Data Collection component stores data from the environment and data from the project being authored.
- *Graphical User Interface*: The GUI provides the user with a touch-friendly sketch-based interface for storyboarding and staging. It provides support for multi-touch interaction, navigation, editing storyboards, and sketching floor plans and storyboards.
- *3D Context*: The 3D Context component provides a 3D context to the floor plan editor and the Sketch Editor. It contains a Computer Graphics Module which allows the user to work directly on the 3D virtual environment. It also contains a Computer Vision Module which is responsible for interpreting the user's sketches.

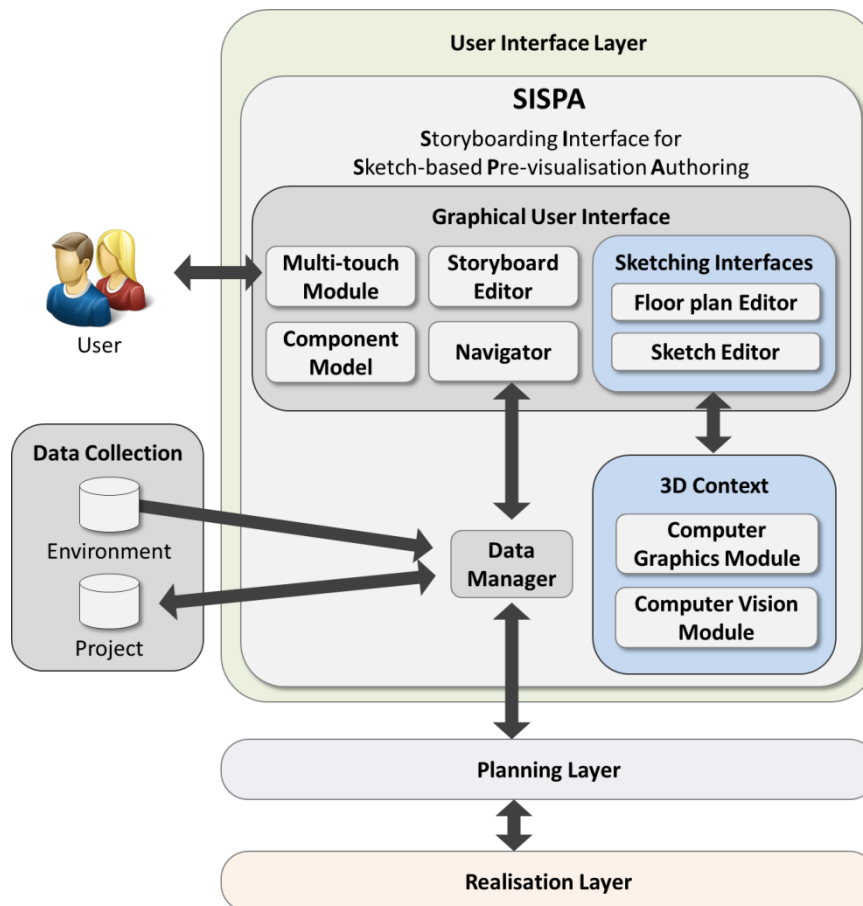


Figure 6.1: The SISPA framework.

The findings made during the design and implementation of the Computer Vision Module can be used to assist future research in describing images quantitatively for the purpose of interpreting rough sketches. The investigation revealed the strengths and weaknesses of each approach and it showed how the methods can be combined in order to describe rough user sketches reliably (see Section 4.6.3).

An algorithm for estimating the unknown camera represented in the user's sketch is also proposed (see Section 4.6.6). The algorithm combines the information known from the floor plan and the poses estimated from the props sketched by the user in order to estimate the camera parameters for the storyboard panel. Existing sketch-based storyboarding tools do not automatically interpret the user's sketches in order to establish the camera from the sketched environment (see Section 2.6.2.4).

A set of guidelines was proposed for designing sketch/touch-based pre-visualisation authoring tools that automatically interpret user sketches and follow the storyboarding approach (see Section 5.8). The guidelines were derived from existing heuristics and the lessons learned from the usability evaluation of the prototype conducted during this research.

The guidelines can be used by future researchers and developers for designing similar pre-visualisation authoring tools.

6.3.2 Practical Contributions

The practical contributions of this research include the proof-of-concept prototype tool which is one of the first touch-friendly storyboarding tools that can interpret user sketches in order to estimate the camera, place props and characters in the 3D virtual environment and author pre-visualisations using a stylus-enabled multi-touch tablet computer. The prototype tool was designed and implemented modularly and it can be extended easily for future research. The results of the usability evaluation are also part of the practical contributions made in this research. Future researchers can use the evaluation results for comparative studies.

6.4 Limitations and Problems Encountered

This research was limited to investigating how minimalistic pre-visualisations can be authored using a sketch-based storyboarding interface. Sketch-based methods for authoring detailed (animated) pre-visualisations were not investigated. The scope of the research was therefore limited to a selected number of storyboard elements, annotations and graphical devices (see Section 2.8). The environmental data used during the evaluation of the proof-of-concept prototype tool was also limited. A small scenario with a single set, six props and four characters was used (eleven low polygon 3D models). This was due to limitations on the hardware used for usability testing.

The problems encountered during this research include the above-mentioned hardware limitation. The tablet computer (see Section 4.5) used did not provide any dedicated graphics hardware acceleration. Limited graphics acceleration was supported by the integrated graphics chipset provided by the Intel Core i5 processor. The additional graphics processing requirements on the processor increased the run times of the computer vision algorithms. This lowered the prototype's performance compared to using a desktop computer with dedicated graphics hardware.

6.5 Future Research

There are several research opportunities for authoring pre-visualisations using a sketch/touch-based storyboarding approach. Future research could investigate how the framework proposed in this research (SISPA) can be combined with existing pre-visualisation frameworks in order

to generate detailed animated pre-visualisations (see Section 4.2). This involves designing interfaces between the different components of the sketch-based storyboarding GUI and the pre-visualisation framework that can translate the story representation into control language scripts (see Section 2.6.1.2).

The research could also be extended to include more elements from the requirements identified in Section 2.8. This would introduce new problems for interpreting user sketches that can be solved using computer vision and sketch-based interfacing methods. These problems include the creation of continuous animations from the still storyboard images as well as dealing with sketched elements that are not valid or accurate in 3D space (see Section 2.7.3.2).

There is also a need to investigate how sketch-based storyboarding tools can be incorporated into existing high-end modelling and animation tools (see Section 2.6.2.2) and simplified animation tools (see Section 2.6.2.3). The qualitative results from the usability evaluation conducted in this research suggest that sketch-based storyboarding methods may be useful for modelling and animation experts.

Future research could also involve evaluating the proposed authoring approach using comparative studies and field studies. A comparative study could compare the sketch-based storyboarding approach with existing click-and-drag approaches in terms of usability and usefulness. A field study could be conducted over a period of time to investigate how the sketch-based storyboarding approach can be adopted in the filmmaking industry for authoring pre-visualisations.

References

- Abidi, M.A. and Chandra, T. (1995): A New Efficient and Direct Solution for Pose Estimation Using Quadrangular Targets: Algorithm and Evaluation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. : pages 534–538
- Adobe (2012): *Photoshop* [online]. Available at: <http://www.photoshop.com/> [Accessed 18 November 2012].
- Aleksandra Čereković, Tomislav Pejša and Igor Pandžić (2010): A Controller-based Animation System for Synchronizing and Realizing Human-like Conversational Behaviors. In *Development of Multimodal Interfaces: Active Listening and Synchrony*. Springer Berlin / Heidelberg : pages 80–91
- Ansar, A. and Daniilidis, K. (2003): Linear Pose Estimation from Points or Lines. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. : pages 578–589
- Asus (2012): *Eee Slate EP121* [online]. Available at: http://www.asus.com/Eee/Eee_Pad/Eee_Slate_EP121/ [Accessed 28 October 2012].
- Autodesk (2011a): *Autodesk 3ds Max Products Products: Features* [online]. Available at: <http://usa.autodesk.com/3ds-max/features/> [Accessed 20 June 2011].
- Autodesk (2011b): *Autodesk Maya Feature Compare* [online]. Available at: <http://usa.autodesk.com/maya/compare/> [Accessed 21 June 2011].
- Autodesk (2011c): *Autodesk MotionBuilder Real-Time 3D Character Animation Software* [online]. Available at: <http://usa.autodesk.com/adsk/servlet/pc/index?id=13581855&siteID=123112> [Accessed 20 June 2011].
- Autodesk (2011d): *Autodesk Softimage Features* [online]. Available at: <http://usa.autodesk.com/adsk/servlet/pc/index?siteID=123112&id=13571400> [Accessed 20 June 2011].
- Balaji, T.S., Holtzblatt, K., Kangas, E., Kates, J., Kinnunen, T., Landers, B., Page, C., Moritz, B., Bloom, J., Chipcase, J., Lehtikoinen, J. and Rondeau, D. (2005): Designing for the mobile device: experiences, challenges and methods. In *Communications of the ACM*. ACM
- Barrow, H., Tenenbaum, J., Bolles, R. and Wolf, H. (1977): Parametric correspondence and chamfer matching: Two new techniques for image matching. In *International Joint Conference On Artificial Intelligence*. Cambridge, USA: Morgan Kaufmann : pages 659–663

-
- Battiatto, S., Gallo, G., Puglisi, G. and Scellato, S. (2007): SIFT Features Tracking for Video Stabilization. In *Proceedings of the 2007 International Conference on Image Analysis and Processing*. : pages 825–830
- Bentley, J.L. (1975): Multidimensional binary search trees used for associative searching. In *Communications of the ACM*. : pages 509–517
- Blender Foundation (2012): *Blender Foundation* [online]. Available at: <http://www.blender.org/blenderorg/blender-foundation/> [Accessed 18 October 2012].
- Brand, M. (1999): Shadow puppetry. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Kerkyra , Greece : pages 1237–1244
- Bregler, C., Malik, J. and Pullen, K. (2004): Twist Based Acquisition and Tracking of Animal and Human Kinematics. *International Journal of Computer Vision*, 56(3), pages 179–194.
- Bujnak, M. and Kukelova, Z. (2008): A general solution to the P4P problem for camera with unknown focal length. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*. : pages 1–8
- Bull, G. and Kajder, S. (2004): Digital storytelling in the language arts classroom. *Learning & Leading with Technology*, 32(4), pages 46–49.
- Canny, J. (1986): A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6).
- Cassell, J., Vilhjálmsón, H.H. and Bickmore, T. (2001): BEAT: the Behavior Expression Animation Toolkit. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM
- Chakravarthy, A., Beales, R., Jung, Y., Wagner, S., Jung, C., Yannopoulos, A., Koutsoutsos, S., Schiffmann, R., Hedtke, R. and Saenen, I. (2010): A Notation Based Approach To Film Pre-vis. In *Proceedings of the Conference for Visual Media Production*. Los Alamitos, CA, USA: IEEE Computer Society : pages 58–63
- Chakravarthy, A., Beales, R., Walland, P. and Yannopoulos, A. (2009): ANSWER: A Semantic Approach to Film Direction. In *Proceedings of the International Conference on Internet and Web Applications and Services*. Los Alamitos, CA, USA: IEEE Computer Society : pages 645–648
- Chaudhuri, D. and Samal, A. (2006): A simple method for fitting of bounding rectangle to closed regions. *Pattern Recognition*, 40(7), pages 1981–1989.
- Chaudhuri, P., Kalra, P. and Banerjee, S. (2004): A System for View-Dependent Animation. *Computer Graphics Forum*, 23(3), pages 411–420.
- Cheung, K.M.G., Baker, S. and Kanade, T. (2003): Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In

-
- Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* : pages 77–84
- Clipartoday (2011): *Clipart Today* [online]. Available at: http://www.clipartoday.com/clipart/cartoons/cartoon/cartoon_266948.html [Accessed 25 July 2011].
- Clker (2011): *Light Bulb clip art* [online]. Available at: <http://www.clker.com/clipart-6937.html> [Accessed 26 July 2011].
- Dance, C., Willamowski, J., Fan, L., Bray, C. and Csurka, G. (2004): Visual Categorization with Bags of Keypoints. In *Proceedings of the ECCV International Workshop on Statistical Learning in Computer Vision.* : page 22
- Davis, J., Agrawala, M., Chuang, E., Popović, Z. and Salesin, D. (2003): A Sketching Interface for Articulated Figure Animation. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* San Diego, California: Eurographics Association : pages 320–328
- Decarolis, B., Pelachaud, C., Poggi, I. and Steedman, M. (2004): APMML, a Mark-up Language for Believable Behavior Generation. *Life-Like Characters: Tools, Affective Functions, and Applications*, pages 65–86.
- Dementhon, D.F. and Davis, L.S. (1995): Model-Based Object Pose in 25 Lines of Code. *International Journal of Computer Vision*, 15(1), pages 123–141.
- Didier, J.-Y., Ababsa, F. and Mallem, M. (2008): Hybrid camera pose estimation combining square fiducials localisation technique and orthogonal iteration algorithm. *International Journal of Image and Graphics*, pages 169–188.
- Dumas, J.S. and Redish, J.C. (1999): *A Practical Guide to Usability Testing*, Intellect Books.
- EMGU (2012): *EMGU CV* [online]. Available at: http://www.emgu.com/wiki/index.php/Main_Page [Accessed 7 November 2012].
- Electronic Arts (2009): *The Sims 3* [online]. Available at: <http://www.ea.com/the-sims-3> [Accessed 20 October 2012].
- Engelbrecht, A.P. (2002): *Computational Intelligence: An Introduction*, Wiley.
- Ferilli, S., Basile, T.M.A., Esposito, F. and Biba, M. (2011): A Contour-based Progressive Technique for Shape Recognition. In *Proceedings of the International Conference on Document Analysis and Recognition.* : pages 723–727
- Fiore, P.D. (2001): Efficient Linear Solution of Exterior Orientation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(2), pages 140–148.
- Fitzpatrick, J.M. and West, J.B. (2001): The Distribution of Target Registration Error in Rigid-Body Point-Based Registration. *IEEE Transactions on Medical Imaging*, 20(9), pages 917–927.

-
- Forton, G. (2011): *Gerald Forton's Website* [online]. Available at: <http://www.home.earthlink.net/~movieboards/fortonboard1.html> [Accessed 12 November 2011].
- Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin Princeton, D. and Jacobs, D. (2003): A Search Engine for 3D Models. *ACM Transactions on Graphics*, 22(1), pages 83–105.
- Gadhavi, B. and Shah, K. (2010): *Analysys of the Emerging Android Market*. San José State University.
- Gao, X.-S., Hou, X.-R. and Tang, J. (2003): Complete solution classification for the perspective-three-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8), pages 930–943.
- Geurts, P. (2002): *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. University of Liege, Belgium.
- Glebas, F. (2008): *Directing the Story: Professional Storytelling and Storyboarding Techniques for Live Action and Animation* illustrate, Focal Press.
- Gold, S. and Rangarajan, A. (1996): A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4), pages 377–388.
- Goldman, D.B., Curless, B., Salesin, D. and Seitz, S.M. (2006): Schematic Storyboarding for Video Visualization and Editing. *ACM Transactions on Graphics*, 25(3), pages 862–871.
- Golub, G.H. and Loan, C.F. (1996): *Matrix Computations* 3rd edition,
- Greenhead, B. (2011): *Take-off cartoon* [online]. Available at: <http://www.cartoonstock.com/directory/t/take-off.asp> [Accessed 15 November 2011].
- Grochow, K., Martin, S.L., Hertzmann, A. and Popović, Z. (2004): Style-Based Inverse Kinematics. In *Proceedings of the 2004 SIGGRAPH Conference*. Los Angeles, California: ACM : pages 522–531
- Han, J. and Kamber, M. (2006): *Data Mining: Concepts and Techniques* 2nd edition, San Francisco, California, USA: Diane Cerra.
- Hanson, A.R. and Kumar, R. (1994): Robust methods for estimating pose and a sensitivity analysis. *Computer Vision and Image Understanding*, 60(11), pages 313–342.
- Haralick, R.M., Lee, D. and Ottenburg, K. (1991): Analysis and Solutions of The Three Point Perspective Pose Estimation Problem. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. : pages 592–598
- Harris, C. and Stephens, M. (1988): A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*. : pages 147–151

-
- Hill, F.S. and Kelley, S.M. (2007): *Computer Graphics Using OpenGL* Third, Pearson Education.
- Hofstee, E. (2009): *Constructing a Good Dissertation: A Practical Guide to Finishing a Master's*, Johannesburg, South Africa: EPE.
- Hoggan, E., Brewster, S. and Johnston, J. (2008): Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proceedings of the CHI conference on Human factors in computing systems*. New York, New York, USA: ACM : page 1573
- Horn, B.K.P., Hilden, H.M. and Negahdaripour, S. (1988): Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America*, 5(7), pages 1127–1135.
- Hou, S. and Ramani, K. (2006): Sketch-based 3D Engineering Part Class Browsing and Retrieval. In *Proceedings of the EUROGRAPH-ICS Workshop on Sketch-Based Interfaces and Modeling*. : pages 131–138
- Howe, N.R. (2004): Silhouette Lookup for Automatic Pose Tracking. In *Proceedings of the Computer Vision and Pattern Recognition Workshop*. : pages 15–22
- Huang, Z., Shen, H.T., Shao, J. and Zhao, X. (2009): Bounded Coordinate System Indexing for Real-time Video Clip Search. *Transactions on Information Systems*, pages 1–33.
- ISO9241-11 (1998): *Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability.*,
- Innoventive Software (2009): *FrameForge Previz Studio* [online]. Available at: <http://www.frameforge3d.com/Products/> [Accessed 20 April 2011].
- Intel Corporation (2012): *3rd Generation Intel® Core™ i5 Processor* [online]. Available at: <http://www.intel.com/content/www/us/en/processors/core/core-i5-processor.html> [Accessed 15 November 2012].
- Intel Corporation, Willow Garage and Itseez (2012): *OpenCV* [online]. Available at: <http://opencv.org/> [Accessed 7 November 2012].
- Jarvis, R.A. (1973): On the Identification of the Convex Hull of a Finite Set of Points in the Plane. *Information Processing Letters*, pages 18–21.
- Jhala, A., Rawls, C., Munilla, S. and Young, R.M. (2008): Longboard: A Sketch Based Intelligent Storyboarding Tool for Creating Machinima. In *Proceedings of FLAIRS '08*, pages 386–391.
- Jhala, A. and Young, R.M. (2006): Representational Requirements for a Plan Based Approach to Automated Camera Control. In *Proceedings of the Second Conference on Artificial Intelligence and Interactive Digital Entertainment*. Marina del Rey, CA
- Johnson, R.E. (1997): Components, Frameworks, Patterns. In *Proceedings of the 1997 symposium on Software reusability*. ACM : pages 1–23

- Jolliffe, I.T. (2002): *Principal Component Analysis* 2nd edition,
- Jost, T. (2003): A Multi-Resolution ICP with Heuristic Closest Point Search for Fast and Robust 3D Registration of Range Images. In *Proceedings of the International Conference on 3-D Digital Imaging and Modeling*. : pages 427–433
- Jung, Y., Wagner, S., Jung, C., Behr, J. and Fellner, D. (2010): Storyboarding and Pre-Visualization with X3D. In *Proceedings of the 15th International Conference on Web 3D Technology*. Los Angeles, California : pages 73–82
- Jung, Y.A. (2009): *PML 2.1 Specification* [online]. Available at: www.answer-project.org/innovations/PMLSpecification.pdf [Accessed 18 April 2011].
- Kadir, T. and Brady, M. (2003): Scale Saliency: a novel approach to salient feature and scale selection. In *Visual Information Engineering*. : pages 25–28
- Kadir, T., Zisserman, A. and Brady, M. (2004): An affine invariant salient region detector. *Image Rochester NY*, 3021, pages 228–241.
- Kaikkonen, A., Kallio, T., Kankainen, A., Kekalainen, A. and Cankar, M. (2005): Usability Testing of Mobile Applications : A Comparison between Laboratory and Field Testing. *Journal of Usability Studies*, 1(1), pages 4–16.
- Ke, Y. and Sukthankar, R. (2004): PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. : pages 506–513
- Kelleher, C. (2006): *Motivating Programming using storytelling to make computer programming attractive to middle school girls*. Carnegie Mellon University.
- Khronos Group and Silicon Graphics (2012): *OpenGL* [online]. Available at: <http://www.opengl.org/> [Accessed 7 November 2012].
- Kopp, S., Jung, B., Leßmann, N. and Wachsmuth, I. (2003): Max – A Multimodal Assistant in Virtual Reality Construction. *Künstliche Intelligenz*, 17.
- Kopp, S., Krenn, B., Marsella, S., Marshall, A.N., Pelachaud, C., Thórisson, K.R. and Vilhjálmsón, H. (2006): Towards a Common Framework for Multimodal Generation The Behavior Markup Language. In *Intelligent Virtual Agents*. Springer Berlin / Heidelberg : pages 205–217
- Kopp, S. and Wachsmuth, I. (2004): Synthesizing multimodal utterances for conversational agents. *Computer Animation and Virtual Worlds*, 15, pages 39–52.
- Koyani, S.J., Bailey, R.W., Nall, J.R., Allison, S., Mulligan, C., Bailey, K. and Tolson, M. (2004): *Research-Based Web Design & Usability Guidelines*, GSA.
- Labschütz, M. and Krösl, K. (2011): Content Creation for a 3D Game with Maya and Unity 3D. In *The 15th Central European Seminar on Computer Graphics*.

- Lazebnik, S., Schmid, C. and Ponce, J. (2003): A Sparse Texture Representation Using Affine-Invariant Regions 2 . Building the Representation. In *Computer Vision and Pattern Recognition*. : pages 19–324
- Lee, J. and Funkhouser, T. (2008): Sketch-Based Search and Composition of 3D Models. In *Proceedings of the EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling*.
- Lepetit, V. and Fua, P. (2005): Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, pages 1–89.
- Levin, D.M. (1988): *The opening of vision: nihilism and the postmodern situation*, Routledge.
- Lewis, J.R. (1995): IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1), pages 57–78.
- Life at the Pond (2012): *Life at the Pond* [online]. Available at: <http://lifeatthepond.com/> [Accessed 23 October 2012].
- LightWave (2012): *LightWave* [online]. Available at: <https://www.lightwave3d.com/> [Accessed 18 October 2012].
- Lin, Y. (2006): 3D character animation synthesis from 2D sketches. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*. Kuala Lumpur, Malaysia: ACM : pages 93–96
- Loffler, J. (2000): Content-based Retrieval of 3D Models in Distributed Web Databases by Visual Shape Information. In *Proceedings of IEEE International Conference on Information Visualization*. IEEE Computer Society : pages 82–87
- Long, B. and Schenk, S. (2002): *The Digital Filmmaking Handbook*, Charles River Media.
- Lowe, D.G. (2004a): Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2).
- Lowe, D.G. (2004b): Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image. *The University of British Columbia*, (US Patent 6,711,293).
- Lowe, D.G. (1999): Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. : pages 1150–1157
- Lu, C.P., Hager, G.D. and Mjolsness, E. (2000): Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6), pages 610–622.
- Luckow, R. (2010): Unnamed things creating a controlled vocabulary for the description of animated moving image content. *Journal of Digital Asset Management*, 6(3), pages 153–157.

-
- Machado, T.L. de A., Gomes, A.S. and Walter, M. (2009): A comparison study: sketch-based interface versus wimp interfaces in three-dimensional modeling tasks. In *Proceedings of the Latin American Web Congress*. : pages 29–35
- Mamer, B. (2008): *Film Production Technique: Creating the Accomplished Image 5*, illustr, Wadsworth Cengage Learning.
- Mao, C. and Qin, S.F. (2005): A Sketch-Based Gesture Interface for Rough 3D Stick Figure. In *Proceedings of Eurographics Workshop on Sketch Based Interfaces and Modeling*. Dublin: Eurographics
- Maree, R., Geurts, P., Piater, J. and Wehenkel, L. (2004): A generic approach for image classification based on decision tree ensembles and local sub-windows. In *Proceedings of the 6th Asian Conference on Computer Vision*. : pages 860–865
- Marino, P. (2004): *3D Game-Based Filmmaking: The Art of Machinima*, Scottsdale, AZ: Paraglyph Press.
- Marr, D. (1982): *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, New York: Freeman and Company.
- Marr, D. and Hildreth, E. (1980): Theory of edge detection. In *Proceedings of the Royal Society of London*. : pages 187–217
- Matas, J., Chum, O., Urban, M. and Pajdla, T. (2004): Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 22(10), pages 761–767.
- Matthews, T. and Vogts, D. (2011): A sketch-based articulated figure animation tool. In *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*. Cape Town, South Africa: ACM : pages 151–160
- McCann, S. and Lowe, D.G. (2011): Local Naive Bayes Nearest Neighbor for Image Classification. *Computer Vision and Pattern Recognition*.
- McKenzie, D. (2011): *Designing For Android Tablets* [online]. Available at: <http://mobile.smashingmagazine.com/2011/08/09/designing-for-android-tablets/> [Accessed 29 October 2012].
- Microsoft (2012): *Microsoft .Net* [online]. Available at: <http://www.microsoft.com/net> [Accessed 7 November 2012].
- Miki, I., Trivedi, M.M., Hunter, E. and Cosman, P.C. (2002): Human Body Model Acquisition and Motion Capture Using Voxel Data. In *Proceedings of the Second International Workshop on Articulated Motion and Deformable Objects*. : pages 104–118
- Mikolajczyk, K. and Schmid, C. (2005): A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), pages 1615–1630.

- Mikolajczyk, K. and Schmid, C. (2002): An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision*. : pages 1–7
- Mikolajczyk, K. and Schmid, C. (2001): Indexing based on scale invariant interest points. In *Proceedings of the IEEE International Conference on Computer Vision*. Vancouver, BC, Canada : pages 525–531
- Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. and Gool, L.V. (2005): A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2), pages 43–72.
- Mitra, P., Shankar, B.U. and Pal, S.K. (2004): Segmentation of multispectral remote sensing images using active support vector machines. *Pattern Recognition Letters*, 25(9), pages 1067–1074.
- Moeslund, T.B., Hilton, A. and Kruger, V. (2006): A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2), pages 90–126.
- Mono (2012): *Tao* [online]. Available at: <http://www.mono-project.com/Tao> [Accessed 7 November 2012].
- Moreno-Noguer, F., Lepetit, V. and Fua, P. (2007): Accurate Non-Iterative $O(n)$ Solution to the PnP Problem. In *Proceedings of the International Conference on Computer Vision*. : pages 1–8
- Moviestorm (2011): *The Moviestorm guide to previsualisation* [online]. Available at: <http://www.moviestorm.co.uk/hub/professional> [Accessed 28 November 2011].
- Mustapha, M., Lim, H. and Jafri, M. (2010): Comparison of Neural Network and Maximum Likelihood Approaches in Image Classification. *Journal of Applied Sciences*, 20(22), pages 2847–2854.
- NIST (2001): *Common Industry Format for Usability Test* [online]. Available at: www.nist.gov/iusr [Accessed 14 November 2012].
- Nadernejad, E. and Sharifzadeh, S. (2008): Edge Detection Techniques Evaluations and Comparisons. *Applied Mathematical Sciences*, 2(31), pages 1507–1520.
- Navab, N. and Faugeras, O. (1993): Monocular Pose Determination from Lines: Critical Sets and Maximum Number of Solutions. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. : pages 254–260
- Navaratnam, R., Thayananthan, A., Torr, P.H.S. and Cipolla, R. (2005): Hierarchical Part-Based Human Body Pose Estimation. In *Proceedings of the 2005 British Machine Vision Conference*.
- Nielsen, J. (1994): Usability Inspection Methods. In *Proceedings of the Conference Companion on Human Factors in Computing Systems*. ACM : pages 413–414

- Nielsen, M., Störring, M., Moeslund, T.B. and Granum, E. (2003): A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In *Gesture-Based Communication in Human-Computer Interaction*.
- Nitsche, M. (2008): Experiments in the Use of Game Technology for PreVisualization. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*. Toronto, Ontario, Canada: ACM : pages 160–165
- Oates, B.J. (2006): *Researching information systems and computing*, SAGE.
- Ong, E.-J. and Hilton, A. (2006): Learnt inverse kinematics for animation synthesis. *Graphical Models*, 68(5-6), pages 472–483.
- Oxford, D. (2010): “*storyboard*” *Oxford Dictionaries*. [online]. Available at: <http://oxforddictionaries.com/definition/storyboard?region=us> [Accessed 12 October 2011].
- Pellegrini, S. (2007): Articulated Object Recognition. *Society for Optics and Photonics*, pages 14–24.
- PicturesOf (2011): *A Retro Cartoon of a Man Disoriented and Dizzy* [online]. Available at: <http://www.picturesof.net/pages/100603-003024-326053.html> [Accessed 15 November 2011].
- Piwek, P., Krenn, B., Schröder, M., Grice, M., Baumann, S. and Pirker, H. (2004): RRL: A Rich Representation Language for the Description of Agent Behaviour in NECA. *Net Environment for Embodied Emotional Conversational Agents*.
- Power Production Software (2011): *Storyboard Quick* [online]. Available at: http://www.powerproduction.com/storyboard_quick.html [Accessed 3 July 2011].
- Pramaggiore, M. and Wallis, T. (2008): *Film: A Critical Introduction*,
- Proferes, N.T. (2005): *Film Directing Fundamentals*, Focal Press.
- Quinlan, J.R. (1985): Induction of Decision Trees. *Machine Learning*, 1(1), pages 81–106.
- Rabiger, M. (2003): *Directing: Film Techniques and Aesthetics 4*, illustr, Focal Press.
- Ramanan, D., Forsyth, D.A. and Zisserman, A. (2005): Strike a pose: tracking people by finding stylized poses. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society : pages 271–278
- Reallusion (2012): *iClone 5* [online]. Available at: <http://www.reallusion.com/iclone/> [Accessed 20 October 2012].
- Rijsselbergen, D.V., Keer, B.V.D., Verwaest, M., Mannens, E. and Walle, R.V. de (2009): Movie script markup language. In *Proceedings of the 9th ACM symposium on Document engineering*. ACM : pages 161–170

- Rizzo, M. (2005): *The Art Direction handbook for Film*,
- Robin, B.R. (2008): Digital storytelling: A powerful technology tool for the 21st century classroom. *Theory Into Practice*, 47(3), pages 220–228.
- Rosenhahn, B., Perwass, C. and Sommer, G. (2004): *Foundations about 2D-3D Pose Estimation* [online]. Available at: <http://homepages.inf.ed.ac.uk/rbf/CVonline/> [Accessed 12 January 2012].
- Roth, P.M. and Winter, M. (2008): Survey of appearance-based methods for object recognition. *Institute for Computer Graphics and Vision Graz University of Technology Austria Tech Rep*.
- Sadik, A. (2008): Digital storytelling A meaningful technology-integrated approach for engaged student learning. *Educational Technology Research and Development*, 56, pages 487–506.
- Sajjanhar, A. and Lu, G. (1997): A Grid Based Shape Indexing and Retrieval Method. *Computer Journal on Multimedia Storage and Archiving Systems*, 29, pages 131–140.
- Salo, K., Arhippainen, L. and Hickey, S. (2012): Design Guidelines for Hybrid 2D / 3D User Interfaces on Tablet Devices A User Experience Evaluation. In *Proceedings of the Fifth International Conference on Advances in Computer-Human Interactions*. ACHI : pages 180–185
- Saunders, M., Lewis, P. and Thornhill, A. (2009): *Research Methods for Business Students*, Pearson Education.
- Screen Africa (2012): *Zambezia wins at Durban Film Fest* [online]. Available at: <http://www.screenafrica.com/page/news/festivals/1343610-Zambezia-wins-at-Durban-Film-Fest#.UIWkAsV196w> [Accessed 23 October 2012].
- SeagullsCalling (2011): *SeagullsCalling* [online]. Available at: <http://seagullscalling.com/?p=189> [Accessed 15 November 2011].
- Seetha, M., Muralikrishna, Deekshatulu, B.L., Malleswari, B.L., Nagaratna and Hegde, P. (2008): Artificial Neural Networks and Other Methods of Image Classification. *Theoretical and Applied Information Technology*, pages 1039–1053.
- Shahabi, C. and Safar, M. (2007): An experimental study of alternative shape-based image retrieval techniques. *Multimedia Tools and Applications*, 32(1), pages 29–48.
- Sharp, H., Rogers, Y. and Preece, J. (2007): *Interaction design: beyond human-computer interaction*, Wiley.
- Shelly (2011): *Escape* [online]. Available at: <http://zumbaspot.blogspot.com/> [Accessed 28 November 2012].

- Shin, H. and Igarashi, T. (2007): Magic Canvas: Interactive Design of a 3-D Scene Prototype from Freehand Sketches. In *Proceedings of the 2007 Graphics Interface Conference*. Montreal, Canada: ACM : pages 63–70
- Shotton, J., Blake, A. and Cipolla, R. (2008): Multi-Scale Categorical Object Recognition Using Contour Fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7), pages 1270–1281.
- Side Effects Software (2012): *Side Effects Software* [online]. Available at: <http://www.sidefx.com/> [Accessed 18 October 2012].
- Sidenbladh, H., Black, M.J. and Fleet, D.J. (2000): Stochastic Tracking of 3D Human Figures Using 2D Image Motion. In *Proceedings to the European Conference on Computer Vision*. : pages 702–718
- Simon, M. (2006): *Storyboards: motion in art* 3rd edition, Focal Press.
- Skorupski, J. (2009): Storyboard Authoring of Plan-Based Interactive Dramas. In *Proceedings of the 4th International Conference on Foundations of Digital Games*. Orlando, Florida: ACM : pages 349–351
- StockphotoPro (2011): *Race* [online]. Available at: http://www.stockphotopro.com/photo_of/race/6761373ZNF/___race__action__cartoon_ [Accessed 11 August 2011].
- Stone, D., Jarrett, C., Woodroffe, M. and Minocha, S. (2005): *User Interface Design and Evaluation*, San Francisco, California, USA.
- Tabibian, B. (2005): *SIFT Implementation* [online]. Available at: <https://sites.google.com/site/btabibian/projects/3d-reconstruction/code> [Accessed 6 June 2012].
- Takalani (2011): *Takalani Sesame* [online]. Available at: <http://www.takalanisesame.com/about.html> [Accessed 23 October 2012].
- Tan, G.C.B., Duh, H.B. and Chen, V.H. (2006): Usability Evaluation for Mobile Device : A Comparison of Laboratory and Field Tests. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. ACM : pages 181–186
- Tao, Y. and Grosky, W.I. (1999): Delaunay triangulation for image object indexing: a novel method for shape representation. In *Proceedings of the SPIE Symposium on Storage and Retrieval for Image and Video Databases*. : pages 23–29
- The Previsualisation Society (2012): *Specific Types of Previs* [online]. Available at: <http://previsociety.com/specific-types-of-previs/> [Accessed 4 October 2012].
- Thiebaut, M., Marsella, S., Marshall, A.N. and Kallmann, M. (2008): SmartBody: Behavior Realization for Embodied Conversational Agents. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*. Estoril,

- Portugal: International Foundation for Autonomous Agents and Multiagent Systems : pages 151–158
- Tomaric, J. (2010): *Filmmaking: Direct Your Movie from Script to Screen Using Proven Hollywood* illustrate, Focal Press.
- Toon Boom Animation Inc. (2011): *Storyboard Pro* [online]. Available at: <http://beta.toonboom.com/professionals/storyboard-pro/features> [Accessed 4 July 2011].
- Toon Boom Animation Inc. (2012): *Storyboard Pro 3D* [online]. Available at: <http://beta.toonboom.com/professionals/storyboard-pro-3d/features> [Accessed 21 October 2012].
- Torresani, L., Kolmogorov, V. and Rother, C. (2008): Feature Correspondence via Graph Matching: Models and Global Optimization. In *Proceedings of the European Conference on Computer Vision*. Springer : pages 596–609
- Triggerfish Animation Studios (2012a): *Triggerfish Animation Studios* [online]. Available at: <http://www.triggerfishstudios.com/en/> [Accessed 23 October 2012].
- Triggerfish Animation Studios (2012b): *Zambezia* [online]. Available at: <http://www.zambeziamovie.com/index.html> [Accessed 23 October 2012].
- Triggs, B. and Quan, L. (2000): Camera Pose Revisited - New Linear Algorithms. *Esprit*, pages 6–8.
- Tullis, T. and Albert, W. (2008): *Measuring the User Experience* 2nd edition, Burlington, MA: Morgan Kaufmann.
- Tuytelaars, T. and Gool, L.J.V. (1999): Content-based Image Retrieval based on Local Affinely Invariant Regions. *Visual Information and Information Systems*, 1614, pages 493–500.
- Tuytelaars, T. and Gool, L.V. (2004): Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59(1).
- Vaidya, A.S., Shaji, A. and Chandran, S. (2006): Vision-Based Posing of 3D Virtual Actors. In *Proceedings of the 2006 Asian Conference on Computer Vision*. : pages 91–100
- Vilhjálmsón, H. (2004): Animating Conversation in Online Games. In *Proceedings of the 2004 International Conference on Entertainment Computing*. Springer Berlin / Heidelberg : pages 193–218
- Vilhjálmsón, H., Cantelmo, N., Cassell, J., Chafai, N.E., Kipp, M., Kopp, S., Mancini, M., Marsella, S., Marshall, A.N., Pelachaud, C., Ruttkay, Z., Thorisson, K.R., Welbergen, H. and Werf, R.J. (2007): The Behavior Markup Language Recent Developments and Challenges. In *Proceedings of the 7th international conference on Intelligent Virtual Agents*. Paris, France: Springer-Verlag : pages 99–111

- Vilhjálmsón, H.H. (2005): Augmenting Online Conversation through Automated Discourse Tagging. In *Proceedings on the Hawaii International Conference on System Sciences*. IEEE Computer Society
- Vredenburg, K., Mao, J.-Y., Smith, P.W. and Carey, T. (2002): A survey of user-centered design practice. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, New York, USA: ACM Press : page 471
- Wachter, S. and Nagel, H. (1997): Tracking of persons in monocular image sequences. In *Proceedings of the Nonrigid and Articulated Motion Workshop*. San Juan , Puerto Rico : pages 2–9
- Wais, P., Wolin, A. and Alvarado, C. (2007): Designing a Sketch Recognition Front-End : User Perception of Interface Elements.
- Walker, M. (2000): *The Lexicon of Comicana*, Iuniverse Inc.
- Wu, Y. and Hu, Z. (2006): PnP Problem Revisited. *Journal of Mathematical Imaging and Vision*, 24(1), pages 131–141.
- Xu, M. and Wei, C. (2012): Remotely sensed image classification by complex network eigenvalue and connected degree. *Computational and mathematical methods in medicine*.
- Ye, P. and Baldwin, T. (2008): Towards Automatic Animated Storyboarding. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. Chicago, Illinois: AAAI Press : pages 578–583
- Young, R.M., Riedl, M.O., Branly, M., Jhala, A., Martin, R.J. and Saretto, C.J. (2004): An architecture for integrating plan-based behavior generation with interactive game environments. *Journal of Game Development*, 1(1), pages 1–29.
- Yuyan, W., Iyengar, S.S. and Jain, R. (1994): A New Generalized Computational Framework for Finding Object Orientation Using Perspective Trihedral Angle Constraint. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10), pages 961–975.
- Zhang, D. and Lu, G. (2002): Shape Based Image Retrieval Using Generic Fourier Descriptors. *Signal Processing: Image Communication*, 17, pages 825–848.
- Zhang, J., Marszałek, M., Lazebnik, S. and Schmid, C. (2007): Local Features and Kernels for Classification of Texture and Object Categories: An In-Depth Study. *International Journal of Computer Vision*, 73(2), pages 213–238.
- Zhang, Z. (1994): Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2), pages 119–152.

Appendix A :

An Interview with Triggerfish Animation Studios

Interview with Producer/Animator

Position responsibilities

Overseeing and influencing every aspect of computer generated animation.

Interview proceedings

<Recording begins>

Interviewee: As much as possible we tried to indicate basing movement of characters so that we can get a basic idea what the framing would be like and if there where camera moves.

Timothy: So the storyboard artist would use a storyboarding tool and sketch characters and a light background with basic movement. Do you guys do animation on the camera as well? Panning and zooming?

Interviewee: Yes, you would. Let me see.. (video)

Dieter: I think there's some zooming at the end.

Timothy: If we look at this image, there's a lot of information we can extract and automatically put in your pre-vis software. At the moment, you're asking your animators to block the characters into basic posing <Interviewee agrees>. It might be possible to automate that.

Dieter: You draw this and the system will place the characters.

Interviewee: That's interesting, because we are finding that there's a huge jump between drawing something in 2D where your artists can do a lot of cheats where the artists doesn't really take into account perspective in real 3D space. So often, when it gets to the pre-vis guys upstairs they find that this can't really work. I can show you the pre-vis version... (video)

Dieter: There's some information from the storyboard that might be useful, like the shot and lensing.

Interviewee: So that's the next step from storyboarding to pre-vis.

Dieter: The idea for the project is that you draw stuff in storyboards fairly quickly and then you get pre-vis automatically.

Interviewee: I would assume that you would have established the 3D environment beforehand.

Dieter: Yes. There will be two different parts like a top-down map planner – you draw the terrain a bit - and once you have a basic one with key features in place, that if you see it on in screen, you know where this scene is taking place. If you have enough of those, if you want to pose it, you draw a key feature somewhere in the picture and then it would know where to position it.

Timothy: So, you'll still need your content to be created beforehand. And, have a rough layout of your scene, e.g. where rocks and trees would be. Then some algorithm will try to recognise the content from the storyboard and automate the 3D context of the scene and place the characters.

Dieter: The idea is also if there's stuff you want to add, e.g. "I needed a tree here". – You can draw it in the scene in the correct place, if it has enough information.

Timothy: Do you want to stay with the storyboarding software you have at the moment, what was it called?

Interviewee: Yes, Toonboom.

Timothy: Would you consider using a storyboarding tool that can analyse the storyboard to generate pre-vis, or would you rather use an entirely different system?

Interviewee: I think it would be down to usability. If the artist can work quickly in it (the whole idea of storyboarding is speed). If it's easy for them draw the basic of what they need, then I don't mind. On this team, some guys were drawing on paper, others were using toonboom, others were doing it in photoshop – just depending on their preference.

Dieter: Aside from the pre-vis, is there any other output that gets used elsewhere. Does that feed into any of your other processes aside for just being a visual artifact?

Timothy: In other words, do you use your pre-vis poses as input for a higher fidelity version of the film?

Interviewee: Yes. When the animator picks up the scene, he would use the same positioning and the same posing..

Dieter: Is there a link between someone seeing it, and someone creating it down there?

Timothy: Like a digital.. ?

Interviewee: No, it's a manual transition.

Dieter: That's the gap we are trying to bridge.

Timothy: What activities are involved in the filmmaking process, and specifically in the pre-production phase?

Interviewee: The first step is script writing. Once you have the story outlined, it's broken into scenes so that each event in the story is then broken down into a scene – still in script form. Then each scene is given a scene number. Those are then handed over to the storyboard artists, and they would be told: “this is scene 102, this character walks in, he does this, he has a fight with so and so, and then he leaves” Then the storyboard artist would look at that and try to picture the best way of illustrating that with images.

Timothy: So, the shot type is not annotated on the script. It's up to the storyboard artist to decide what shot is.

Interviewee: Yes, but he would be directed by the director. (example) The script would have some basic directions, e.g. “Pumba walks from left to right and looks angrily at so and so”. It might also include information like: “closeup on Xheco and he's looking angry, cut to wide shot of Desatory”

Timothy: So, you have different versions of the script and then you start annotating it <Interviewee agrees>.

Interviewee: More information gets added as you go along. By the time you're briefing the storyboard artists, they should be told that sort of thing.

Timothy: Is your script digital as well, or is it paper-based?

Interviewee: It's digital.

Dieter: How closely do you link that with your storyboards?

Interviewee: It's a bit difficult at first, and it's also frustrating. There is a lot of change in the initial part of film making is very fluid. You would have a scene that starts out at six shots

long and then suddenly goes to eight and then to nine and then back to two shots – very quickly. So numbering is quite a mission

Timothy: So it's a problem of manual management of the storyboard panels and ordering things as your requirements change.

Interviewee: Ideally, what I would like, is to package it so that each shot has a packet of information. Attached to the shot, would be the chunk of script that defines that scene – using some sort of ID number, not a shot number, because it shot numbers change. The IDed packet can then be stored in a database for later reference.

Timothy: So, basically – you want electronic, automatic linking between the elements of your script and the elements of your storyboard. Because, at the moment, that's manual.

Interviewee: Yes

Dieter: And if you swap the order of stuff, the link should still be preserved.

Interviewee: Yes, it causes a lot of chaos throughout the movie. Not just in the storyboarding process.

Timothy: And any other activities after/during storyboarding?

Interviewee: Once you've produced a script, you go to the storyboard artist and they will draw frames of each major action that happens in the scene. Mostly, if it's a simple shot, it would be one image. If there's more than one storyboard beat that's happening in that shot, they'll draw a couple of frames. It's still one shot, even if there's more than one frame.

Timothy: At what level is an action/story beat: e.g. a character turning or something major happening in the scene?

Interviewee: Anything that is important to the story. So, if the character is just standing there throughout the scene, then it's not. But, if he's standing there and someone comes up behind him and he looks to see what that is, that's a beat. And you'd want to include that so that if someone looks at the storyboard then he/she would know to take that into account.

Timothy: And then, the next step: from the storyboard to the pre-visualisation. So, everything is still within the pre-production phase? You haven't started producing anything yet.

Interviewee: Yes, well, the pre-vis stage is more of a production phase because we had to create all of these 3D assets in order to create these pre-vis's. So we consider storyboarding to be pre-production and pre-vis – production.

Timothy: So, this research doesn't fall perfectly in pre-production. It kind of leans over until the content is available.

Interviewee: It depends. Different studios will define pre-production and production in different ways. Ideally, you'd want these processes to run in parallel for quite a long time before you get into real production. Because, this is the check, fast, high turn-over phase where you can make a lot of mistakes and so a lot of working out. Storyboards are a limitation in that they don't give you any 3D information so you can make assumptions that don't turn out to be accurate when you try to put the characters in 3D space.

Dieter: How many alternative possibilities do you try for a particular shot for a scene? Do you make one linear one and make little changes? Or do you try major alterations? What is typical?

Interviewee: Typically, you do as much exploration as possible in the script because it is very fast. In storyboarding there's quite a bit of turnover because it's the first time we see anything in visual form. They'll probably rework that 2 or 3 times.

Dieter: Would you like to be able to go back and look at previous attempts exactly as they were planned? Like the roll-back features to go backwards and forwards.

Interviewee: We do store everything we create and version it, so we are able to go back if we need to.

Timothy: is it well manage, or a manual process?

Interviewee: It's a manual. Well, the storyboarding process was manual but in the pre-vis tools, we've written some tools that automate it which makes it easier.

Timothy: Let's say you've got your storyboards and you've asked your animators to translate them into rough pre-vis. How would you feel, if you changed something major in the scene, for example: the character is no longer sitting and turning his head, he's now flying over the scene. Would you prefer to be able to sketch the frame over as a storyboard frame or rather ask the animator to manually change the scene. What would be easier?

Interviewee: If the automatic conversion from the storyboard to the pre-vis can guarantee animation coherence, then that would be ideal because it would be a faster way of making adjustments. However, if there are slight coherence problems such as characters being placed slightly different in the scene, that would be a problem. For example, if a rock would be

sketched in one place and then in another, then that would be a problem in the continuity of the animation.

Dieter: So, once an asset has been placed, it's basically locked (static assets) in terms of movement.

Interviewee: Unless you specifically direct it to move.

Timothy: One of the largest problems of storyboards is that because they are really rough, they are prone to error. For example, a rock moving. The conversion process will have to be robust and look for these contradictions.

Interviewee: What might be useful – to have a tool that goes back and forward between these, because when you're doing this, you haven't established any of these major things. You know, the actual terrain and layout of the set. Once you have this, you want to be able to quickly adjust character positions which be easier to sketch in. For example, this guys is the wrong place, quickly sketch him here..

Dieter: As far as sketching is concerned. If you saw his video, the way it works, you sketch the skeleton as a stick figure and it would reposition the model the way it would look then. Would that we a problem?

Timothy: It's possible to use more advanced algorithms so that you can draw the outline, but it becomes really difficult.

Interviewee: You basically want the simplest way of doing it. In fact, I'd find the master hierarchy of the model and just move it in the set. But, let's say his head is up, and we'd like to move it down, it would be nice if you could just draw curve of the neck and head and it pops to the right position.

Timothy: It sounds like you want a combination of three things: something that you can storyboard in that does the automatic conversion from the storyboard into pre-vis. But, if you change the pre-vis the storyboard should be updated so they are almost basically the same thing (linked). And thirdly, you want some of the basic animation functionality that animation packages have: like moving, rotating objects, so that you don't have to sketch the contour of the character changing.

Interviewee: Also, you want to make the setup as fast as possible. If I'm an artist, I just want to click on the pivot point and drag this down. So that it's fast. If you have to draw in a line and indicate the weighting of the effect, it would take more time.

Dieter: The idea we had... Most of the stuff we read, they did storyboarding..not like a video, on paper, comic strip drawn by hand. The idea was that you take a tablet and draw each panel with a stylus and it converts it into that strait away.

Timothy: Storyboard Quick does something similar, but it outputs an animatic (No conversion for storyboard to pre-vis)

Dieter: The reason I asked is that the clicking and dragging sounds more like a tool like SoftImage.

Interviewee: Well, we built basic rigs into our characters so we can rig quickly and do those poses.

Dieter: And if for example, as output from this thing you got a timeline that has a set of key frames that didn't have animation, but stepped keys – would that be useful, because we can put in basic animation like if the thing moves – interpolate.

Timothy: We were considering a higher fidelity animation. Let's say you have some pre-canned animations. They don't have to be production quality. In the animatic, you'd do really basic movement, like the character is being translated. It's possible to, if you use some gestures, to tell the system the character is performing a walking animation. So, instead of having a block moving, it would look like zebra is walking.

Dieter: Well, it would do walking but not necessarily sticking to the ground properly.

Timothy: Would that be useful?

Interviewee: Yes it would. Well, we've already gotten around that. We've saved some basic walking clips.

Timothy: Could you please briefly discuss what you do in production and post production?

Interviewee: Once you have an approved pre-vis shot, the next phase would be to take it to performance animation, but at the same time, you have a team who is involved in asset creation and they are building every single piece, prop, sets, characters that's required in the movie. The idea is to have that happening in parallel with your storyboarding and pre-vis phase. In fact, they inform each other quite a bit. We'll see, this camera is looking at this rock. Someone needs to build that rock. And then the camera swings around to look at something else – we need that as well in this amount of detail but it doesn't need to be high-res because the camera doesn't get that close. There's quite a bit of interaction between the modellers and the pre-vis artists.

Dieter: If you do storyboards and you could indicate distance, would a list of assets and level of detail, or the nearest camera looking at that. Do you think that would be useful.

Interviewee: Yes

Dieter: Or how frequently this occurs in the shot, so it should be built first.

Interviewee: Yes that would be very useful. We use our pre-vis to try and determine that. So, we would know: I've got a wide shot here, all of this environment will have to be built and detailed with rocks and grass or whatever the case is. And it's a moving camera and it takes quite a lot of the set. (video)

Pre-vis and content creation almost end at the same time, so that when production of that scene begins – you have all of those assets ready. Then the animators would take the high-res characters and the stripped down version of the set and do the performance animation and the camera settings and do their performance animation. At the same time, a team would be working on the set, optimizing it – to make sure that, based on that camera angle, they'll strip out everything that's not within the camera to make the scene light. And then, they also dress it, so for shot specific camera angle, they'll adjust plant positions for composition for a particular shot. Once the animators finished the shot, the characters within the animation is placed within that final set and it's given to the lighting team. The lighting team will then set up lighting for the shot and render it out. Often, almost always, the rendering is done in several single different passes. The characters will be rendered in one pass, the background will be rendered in another pass, certain aspects of lighting might be rendered in different passes. E.g. specular pass, or a pass for one particular light. So that you have control after the rendering process, you'll have to have a bit of control in terms of lifting the lighting on a particular character, blurring the background.

Dieter: Do you ever indicate lighting information in your storyboards?

Interviewee: Sometimes. (video example)

Timothy: It's possible to indicate lighting information in storyboards, but it doesn't look like these storyboards are that level of fidelity.

Dieter: That's why we are asking. We were thinking of including it.

Interviewee: We often don't in our storyboards, but we do in our pre-vis. For example, there's a scene here where characters are talking to each other in front of a fire. And we indicate a strong light to get a rough idea of where the yellow light is coming from.

Dieter: Do you think it would be useful if you could sketch the lighting information in the storyboard and it would add the lighting to the pre-vis?

Interviewee: I think it would be helpful, yes; especially for scenes where the lighting is a key part of the story. E.g. if there are just standing around and talking to each other in the middle of the day – you’ll probably need it. But if they would be sitting around a camp fire – you’d like to see the glow on their faces.

Timothy: So, we’d assume some kind of default environment light source, and that would be extracted from whatever 3D context has been determined. If there’s any special cases, like a camp fire, the storyboard artist would just indicate using special lighting gestures.

Interviewee: Yes.

Dieter: Do you guys use any special annotations to mean specific things? Like: arrows, squares and stuff that move.

Interviewee: Yes, sometimes the storyboard artist would draw an arrow to indicate that the character can walk off-screen, or the direction he is going, or an arrow to indicate that the camera should push in. But generally, we’ve taken that process out because we can indicate that in the animatic.

Timothy: I’m wondering if it would be more suitable to take an animatic as input instead of static images. I’m not sure, what do you think? How does the storyboard artist feel about it? Is he drawing each asset individually and just drag and dropping?

Interviewee: He’ll draw them individually on different layers. E.g., he’d know this guy moves, so he’d draw him on a separate layer. And once he’s finished drawing it, he’d animate the character sort of moving the screen.

Dieter: So, at least the layer concept is well known. Because we were thinking of having different layers for different kinds of information. Like, a layer for scenery, a layer for characters, one for lighting, one for camera movements. <Interviewee agrees>

Timothy: Okay, so to get back to the process...

Interviewee: Once the shots are rendered you get multiple passes for each frame. That goes to the compositing team. They take all the passes and assemble them into a single frame. They would layer them on top of each other, treat colour, fix any rendering errors – which are easy to fix instead of sending the frames back to the render farm.

Timothy: How serious are these errors? Pixel errors, or... ?

Interviewee: Yes, pixel errors.

Timothy: For example, rays that were calculated incorrectly or when layers are composed and there would be gaps?

Interviewee: Yes, a typical example would be a rendering glitch. Maybe there's a black spot that pops on and off on a characters face because that scene was rendered across several different computers and some of them made some kind of mess. We had problems on the Zambezi project with flickering feathers.

Timothy and Dieter: <It was probably the depth buffer or alpha blending>

Interviewee: If you could get away with painting on it. Then the compositing artist would just fix it. If it was too much of a job, then they'd have to send it back to the render farm to be re-rendered. So, they fix any errors and sweeten the shot – they make it look as beautiful as it can possibly be. They adjust the colours...

Timothy: Like post imaging operations, like blooming...

Interviewee: Yes, exactly - applying face depth of field by blurring the background. Once that's done, the image would be taken to the edit suite.

Timothy: Is this still during the production phase, or are we leaning over to...

Interviewee: This is still production, I think. This is pretty much at the end of production though. Once the image has been through composition, it's done. It's in the edit suite and the editors add audio from that point.

Timothy: Then you guys move on to the post-production phase?

Interviewee: Yes, the post-production phase very quickly would be grading. They would look at all the shots on the timeline and make sure that each individual scene is consistent in terms of the lighting. Because, often when you have different lighters lighting the shot, there'd be slight inconsistencies in colour because it's a different person. So, they'll overall the scene, lift the blacks, adjust colour so that there's a consistency across it. After grading, it would then be taken to the cinema with a protector on the bring screen to see how it would be like on large screen format. Once that's done, they'll be doing the final music and audio effects after because every time you change the edit you lose audio sync.

Timothy: I hear from the animatic that you are already using dialogue. During the storyboarding phase, would it be useful to be able to add music at the storyboarding phase?

Interviewee: Generally we don't bother with music at this phase.

Timothy: Ok, so I see you guys don't use text at the bottom, you just record audio – but having the link between the script and the digital storyboard would be useful.

Interviewee: Yes.

Dieter: You could turn the text on and off like a flag.

Interviewee: It's useful to have the script as well as the audio, because it gives extra information.

Timothy: Then, after you've added the audio then you can move over to the next phase, marketing?

Interviewee: Yes.

Dieter: What format are your scripts in? Is it a text document?

Interviewee: We use a program called Final Draft. Yes, “sngf”. I think it's an XML document.

Timothy: Regarding the style of storyboarding, do all your storyboards look like that where you sketch out contours or do take higher level where you start shading things?

Interviewee: Not unless it's specific to the story.

Timothy: Ok, so it's just important to show the contours of the characters.

Dieter: Basic placement, how they are and basic pose.

Interviewee: You might indicate, if there was a crowd, you would indicate in colour that he's the principle character.

Dieter: How frequent do you use colour in your storyboards?

Interviewee: Very seldom, more likely, this guys would be grey and everyone else would be white.

Timothy: Ok, how accurate do your storyboards predict what you envision the film to be in the beginning.

Interviewee: At first, they are very rough because the physical 3D sets don't exist. The artist is just interpreting what is coming out of the script. The script says there is a cliff. So, the storyboard artist draws a cliff but we do not know how high it is, where it is until we have actually made it in 3D space.

Timothy: I see that you space particular frames in time. How far away can you space them to have a realistic animation? For example if the character's standing there and he's moving to the other side of the scene. Do you show individual shots in between? When is it important to show that the character's here and now he's there?

Dieter: I think that depends on what they are trying to do.

Timothy: <checking questions>Is there anything that we haven't spoken about that you find important to annotate on storyboards? Can you think of anything else except arrows and colours?

Interviewee: Reference number I guess, or some shot ID.

Timothy: <checking questions> Ok, are there any limitations of the storyboarding software and the Photoshop pre-vis you have at the moment that you can share you us?

Dieter: Thinks that'll like to improve.

Timothy: Stuff that is problematic for you especially with storyboarding and the pre-vis software.

Interviewee: One of the key things, wanting to be able to manage change so that the information about the shot doesn't get disconnected or lost. Because there are so many aspects describing a scene: there would be script, an image, audio – just to describe a shot. If that changes in the ordering of the sequence of events in the edit, then you want to make sure all that information goes with. At the moment that is a manual process – managed by giving it a shot number. So we know that all text associated with scene 13 shot 5 manually kind of move with it. I'd like to be able to manage it as one unit. All if the information sits at one place and move it as a whole.

Timothy: So, it's basically a limitation with automated management of the pre-vis content. Is there anything else you can think of, except for shot numbers?

Interviewee: I guess the more information you can get into your storyboard, the better. Anything that optimises creating new poses or quickly showing camera moves. The key limitation with storyboards is that it's not in 3D, but it's very fast. So, you can sketch a

character much faster than you can load a model and pose it. So, those are the tradeoffs. In storyboarding, you are working very fast but not very accurate. In pre-vis you are more accurate but a little bit slower. In animation you are working very accurate but very very slow.

Timothy: So, something you'd like to see is mid-way. It would allow you to move between the two.

Interviewee: Yes, if would be possible to be able to represent 3D more at the storyboarding phase.

Timothy: Well, what we can do, is make storyboarding and pre-visualisation appear to be the same thing. For example, using non-photo realistic rendering. You sketch the thing, it recognises the scene, poses the characters and automatically generates the non-photo realistic pre-vis in the same storyboard panel. So that you don't even notice it happened. And then you press the play button and your storyboard comes to life. That would get you the data and still look like a storyboard. <Interviewee agrees>

Dieter: The only down side is asset creation, the order is a bit flipped. So you'd have to have some basic models for these things before you could to this sketching like that.

Interviewee: You know, for us, ideally – we want to get to the pre-vis stage earlier, because this is where we really start to see what's involved in physically making the film. Storyboards are more about story. How do the characters interact? How do we tell the story. Pre-vis is more about how do we technically, physically make this movie. So the sooner we can get to pre-vis, the better. If we could use this as our high-speed story telling phase as well as our phase that informs our asset creation, that would be great. Because, we are working in a 3D medium because we need to get to get to this medium as fast as we can.

Timothy: Do you think a non-photo realistic approach would be nice like a cartoon shader?

Interviewee: At this point, aesthetics are not that important.

Timothy: So, if it looks like this then..<looking at their pre-vis>

Interviewee: ...It's perfect, yes.

Timothy: And the fact that you are sketching on top of this thing, it doesn't matter?

Dieter: So, for example, if you wanted to add a new character here, you might add a new layer, select a character and draw the skeleton and they are in there. Would that kind of thing be helpful?

Interviewee: Yes.

Dieter: One of the issues we have, is that we are aiming for a totally sketch-based interface, that you can draw whatever you want. But then there's always a problem with selection. "Do you want to draw a rock, or trees?" The question is, should you allow them to draw anything and allow it to try and find out what it is, or would be better to select a specific thing and you'd draw it and then it places it? Any ideas on how you think that would be?

Interviewee: I guess it's probably better if you have a preset library of trees, rocks and characters. So, you say "oh, I need a tree" immediately in the back ground. So, I go and find the preset tree drag, draw and you scale it up to the right size and it's set in the scene. One problem, the sketch-based thing is a 2D system. But actually, you want to be able to influence the stuff in a 3D environment.

Dieter: You will be able to, if you know a tree is this tall and you draw it at a certain location and a certain height. If it has enough information about the background, it can place the camera it can place the tree at the correct place and side in the 3D scene.

Interviewee: Let's say, I would like to draw a tree behind this character. If I just draw a tree, how does it know where in space the base of that tree is?

Timothy: It would be based in the size. Assuming the artist has enough skill to indicate that the tree is in the distance, then the algorithm will be able to determine that the tree is in the distance. If he sketches really big, then algorithm will determine that the tree is in front of the character. It would heavily depend on the artist's skill. The alternative, is to use a drag and drop style interface, then it wouldn't matter. But, we'd like to investigate sketch-based methods.

Interviewee: I think if you look at sketch-based methods, the best thing for it is along your current project. To be able pose characters based on sketches. If it's fast and intuitive for the artists and generally sketching is more intuitive than grabbing a control and moving it. If they want to say "I want this guy's neck to bend the other way" and they can just sketch it, then that would be much more useful and fast than also using it to populate a scene, then it probably wouldn't be that useful. And the scope that is huge, whereas if you are limited to posing characters it's more feasible.

Dieter: So, it sounds that character posing is more important than scene creation.

Interviewee: Yes, because it's much more specific. For example, the artist might draw a tree. But what tree is it? What is its shape? If he draws a pine tree, will the algorithm know I must generate a pine tree? Or, will you be pre-selecting content from a database?

Timothy: You create content while you do pre-visualisation and storyboarding and it's like two-way communication. So, if you have that content in low or high fidelity in a database, it's possible to do image recognition then there are algorithms that can look at that content and based on the storyboard you sketch, automatically select the correct content.

Dieter: Well, the closest to what you have.

Interviewee: So then it would be simpler for the artist to draw. Not so much an accurate, spend a lot of time drawing. Just, basically draw a symbol.

Dieter: Or different shape trees. If they can draw the one that's the closest to that particular tree then it would select that one.

Timothy: We are aiming for low fidelity storyboards, so most of the image recognition algorithms will fail.

Dieter: For us, we consider this low. Do you consider this high?

Timothy: In these storyboards, there are only basic contours. You can't use feature based recognition methods. <Dieter agrees> And, a very vague future work question. Are there any advances in pre-visualisation that'd you like to see. Something you don't have at the moment, that would be nice. For example, the crowd thing we spoke of earlier.

Interviewee: Being able to represent more complexity with speed. Because, generally, pre-vis is sort of the blue print of the movie. You try to get as much information in there as possible but get it turn over very quickly. The more information you can get in there, the better. But it needs to be quick. So, you'll need to be able to put in a crowd quickly and say "I've got a crowd", because if you take a crowd and put it in one of these scenes, it would just hang. Because, there is just too much information. The crowd would be full of rigged characters and those characters would be heavy.

Timothy: So, to generalise. You want very densely populated scenes to be animated for pre-visualisation without the costs processing and asking staff to do the manual posing.

Interviewee: Yes. One of the other problems is determining the level of detail. For some people, this would be sufficient detail. For example, for the animators doing the performance, this is probably sufficient detail. They know my character stands here and moves to here. For the director, this may be insufficient detail. “What’s behind those characters?” “I need to think of the composition of this shot as well”. The last time we pre-vised the movie on Zambezia, we didn’t even include any texture on the characters. So it was all grayscale. This time, we took it a little bit further, because we could. We took a bit more time to prepare the characters on that level.

Timothy: <Looking at pre-vis> It looks like they are just vertex coloured. <Interviewee agrees> And the animal on the right has a high poly count and the one on the left is very low. It seems like there are inconsistencies. Is that serious?

Interviewee: No, we created the characters beforehand and we did polygon reduction on them to get them down to low-res version.

Timothy: Oh, and the characters you created. Where they rigged at that stage?

Interviewee: Yes they were rigged, but with a very simple rig.

Timothy: When you reduce the character, it seems like you are losing the rig and then re-rigging it using blocks. <Interviewee agrees> Because, I think it’s possible to reduce the character’s poly count and still keep the skeleton so that you don’t have to re-rig it.

Interviewee: But, we actually need two different rigging systems, because it’s wasteful to have the high quality rig at this stage. It would be heavy and drain resources. So, when we knock the character down, we actually do the modelling phase first. We’ll model the high-res character. Then we’ll model the low-res character. Then we’ll rig the low-res character and separately rig the high-res character. So we have a rig for pre-vis and a separate rig for animation.

Timothy: Ok, it is possible to simplify the skeleton of the character. But, even if you keep the high definition skeleton and you keep the low definition mesh, the computation overhead is still small because the calculations are being done on the vertices and not the skeleton.

Interviewee: Yes, it’s actually the complexity of the rig itself. The constraints, relationships and dependencies in that are built into the skeleton itself that cause it to be slow.

Timothy: Oh, I see. So, the skeleton itself is heavy. And you have high bone counts, like 100 / 50?

Interviewee: Probably not high numbers of actual bones. Maybe 50. It's more about the dependencies. For example, you might have 5 bones in the neck. But in the final rig, you'll need it do be IK/FK switching. You need it to be squashy and stretchy. And all of that create extra computational overhead that you actually wouldn't build onto in the pre-vis rig. So, this example is probably just an IK setup.

Timothy: I'm wondering, but this might be out of scope. It seems like your team is spending a lot of resources translating your high fidelity content into pre-vis content. That process can be automated.

Interviewee: It is to a certain extent, but we have rigging tools that allow riggers to say: "this is the top of the shoulder, this is the top of the foot, go" and it would automatically put in the rig for you. So, it's not 100% automatic, there's definitively a manual aspect to it. But we are gradually getting there. Ultimately, it would be great if we had the final character, put the rig in, press a button and it pops out a low-res character with all the non-essential rig elements stripped out.

Dieter: That could be a project for next year.

Timothy: Ok, well I can't think of any other questions.

Dieter: Ok, well thank you very much for the time you've given us.

Interviewee: It was a pleasure.

<Recording ends>

Appendix B :

XML Data Format Examples

Figure B.1 (a) shows an example of an XML file which stores the name and type of each asset in the environment Figure B.1 (b) shows an example of a script file.

```
<?xml version = "1.0" encoding="UTF-8"?>
<environment name = "Fairytale World">
  <set file = "home"/>
  ...
  <prop file = "bowl"/>
  ...
  <character file = "daddy"/>
  ...
  <script file = "goldilocks"/>
  <psamples file = "fairytale world"/>
  <osamples file = "fairytale world"/>
</environment>

<?xml version = "1.0" encoding="UTF-8"?>
<story title = "Goldilocks and the Three Bears">
  <scene title = "Exploring the house">
    <block title = "Goldilocks felt very tired." set = "home">
      <actiondialog
        entity = ""
        text = "Goldilocks felt very tired. She tip-toed..."
        type = "narration"
      />
    </block>
    ...
  </scene>
  ...
</story>
```

(a) (b)
Figure B.1 (a) An XML environment file. (b) An XML script file.

Each 3D model is stored in a separate file using a custom XML format. The 3D model format used by a sketch-based animation tool was reused for this research (Matthews and Vogts 2011). Figure B.2 shows an example model stored in the custom XML 3D model format.

```
<?xml version = "1.0" encoding="UTF-8"?>
<model>
  <vertices>
    <vertex x="-0.385926" y="0.101384" z="1.736532" nx="-0.200446" ny="0.844295" nz="-0.496933"/>
    <vertex x="0.546882" y="-0.092954" z="1.756701" nx="0.080569" ny="-0.867611" nz="-0.490646"/>
    ...
  </vertices>
  <mesh name="Body" armature="Armature" texture="baby.png">
    <face>
      <vertex index="267" u="0.536138" v="0.895174"/>
      <vertex index="90" u="0.498397" v="0.905074"/>
      <vertex index="376" u="0.515346" v="0.864195"/>
    </face>
    ...
  </mesh>
  <armature name="Armature" mesh="Body">
    <bone name="root" matrix="[]" length="0.512225" start="[0.000000; 0.461900; 1.600119]" >
      <bonevertex vertex="21" weight="0.438869"/>
      ...
      <animation>
        <action name="Unposed">
          <pose frame="1" xroll = "0" yroll = "0" zroll = "0" dx = "0" dy = "0" dz = "0"/>
          <pose frame="10" xroll = "0" yroll = "0" zroll = "0" dx = "0" dy = "0" dz = "0"/>
          ...
        </action>
        ...
      </animation>
      <bone ... >
        ...
      </bone>
    </armature>
  </model>
```

Figure B.2: An XML 3D model file.

Appendix C:

GUI Touch Gestures

Table C.1 and Table C.2 provides a summary of the touch gestures supported by the GUI of the SISPA framework. Single tap gestures for selecting objects not shown in the summary. See Section 4.5.1 for a discussion of the touch gesture types.

Table C.1: The touch gestures for the Story Viewer, Script Viewer and Storyboard Editor.

Screen	State	Gesture	Area	Description
Story viewer	Navigate	Anywhere	Spread	Navigates to the Script Viewer, Floor plan Editor or Storyboard Editor for the selected dramatic block.
			Vertical Scroll	Scrolls the story overview vertically.
Script viewer	Navigate	Anywhere	Pinch	Navigates to the Story Viewer.
		Tabs	Flick Left	Navigates to the Floor Plan Editor.
			Flick Right	Navigates to the Storyboard Editor.
List	Vertical Scroll	Scrolls the script vertically.		
Storyboard Editor	Navigate	Anywhere	Spread	Navigates to the Sketch Editor for the selected storyboard panel.
			Pinch	Navigates to the Story Viewer.
		Tabs	Flick Left	Navigates to the Script Viewer.
			Flick Right	Navigates to the Floor Plan Editor.
		Storyboard	Vertical Scroll	Scrolls the storyboard vertically.
		Dialogue	Vertical Scroll	Scrolls the action/dialogue list vertically.

Table C.2: The touch gestures for the Floor Plan Editor and the Sketch Editor.

Screen	State	Area	Gesture	Description
Floor Plan Editor	Navigate	Anywhere	Pinch	Navigates to the Story Viewer.
			Flick Left	Navigates to the Storyboard Editor.
			Flick Right	Navigates to the Script Viewer.
	Pan/Zoom	Editor	Spread	Zoom in the view of the floor plan.
			Pinch	Zoom out the view of the floor plan.
			Drag	Set (Any): Pans the Editor's view. Prop (Design): Move a prop. Symbol (Annotate): Move a symbol.
	Rotate	Editor	Rotate	Prop (Design): Rotate a prop. Symbol (Annotate): Rotate a symbol.
Any	Prop List	Scroll	Scrolls the prop list vertically.	
Sketch Editor	Navigate	Anywhere	Pinch	Navigates to the Storyboard Editor.
	Pan/Zoom	Editor	Spread	Zoom in the view of the sketch.
			Pinch	Zoom out the view of the sketch.
			Drag	Pans the view of the sketch.
	Rotate Camera	Editor	Spread	Moves the camera forwards.
			Pinch	Moves the camera backwards.
			Drag	Rotates the camera.
	Move Camera	Editor	Spread	Moves the camera forwards.
			Pinch	Moves the camera backwards.
			Drag	Moves the camera within the current viewing plane.
	Any	Layer List	Scroll	Scrolls the layer list vertically.

Appendix D:

Ethics Approval (Application)

NMMU RESEARCH ETHICS COMMITTEE (HUMAN)

SECTION A:(To be filled in by a representative from the Faculty RTI Committee)					
Application reference code:	H HUMAN	12 YEAR	SCI FACULTY	CS DEPARTMENT	015 NUMBER
Resolution of FRTI Committee:	<input type="checkbox"/> Ethics approval given (for noting by the REC-H) <input type="checkbox"/> Referred to REC-H for consideration (if referred to REC-H, electronic copy of application documents to be emailed to Imtiaz.Khan@nmmu.ac.za)				
Resolution date:					
Faculty RTI representative signature:					

1. GENERAL PARTICULARS

TITLE OF STUDY	
a) Concise descriptive title of study (must contain key words that best describe the study): Sketch-based Digital Storyboards for Authoring Film Pre-visualizations	
PRIMARY RESPONSIBLE PERSON (PRP)	
b) Name of PRP (must be member of permanent staff. Usually the supervisor in the case of students): Dr. Dieter Vogts 090101F	
c) Contact number/s of PRP: 0415042089	
d) Affiliation of PRP: Faculty Science Specify here, if "other" Department (or equivalent): Computing Sciences	
PRINCIPLE INVESTIGATORS AND CO-WORKERS	
e) Name and affiliation of principal investigator (PI) / researcher (may be same as PRP): Timothy Matthews Gender: Male	
f) Name(s) and affiliation(s) of all co workers (e.g. co-investigator/assistant researchers/supervisor/co-supervisor/promoter/co-promoter). If names are not yet known, state the affiliations of the groups they will be drawn from, e.g. Interns/M-students, etc. and the number of persons involved: Dr. Dieter Vogts (supervisor) and Mr. Kevin Naudé (co-supervisor)	
STUDY DETAILS	
g) Scope of study: Local	h) If for degree purposes: Master's
i) Funding : NRF grant Additional information (e.g. source of funds or how combined funding is split)	

j) Are there any restrictions or conditions attached to publication and/or presentation of the study results? No If YES, elaborate (Any restrictions or conditions contained in contracts must be made available to the Committee): Not applicable
k) Date of commencement of data collection: 2012/09/03 Anticipated date of completion of study: 30 November 2012
l) Objectives of the study (the major objective(s) / Grand Tour questions are to be stated briefly and clearly): The main objective of this study is to determine how effectively sketch-based user interfaces can support the authoring of animated pre-visualizations.
m) Rationale for this study: briefly (300 words or less) describe the background to this study i.e. why are you doing this particular piece of work. A few (no more than 5) key scientific references may be included: Pre-production is an important phase of the filmmaking process because it involves planning every aspect of the film and preparing for film production (Mamer 2008). The need for film pre-visualization has led to the use of several authoring systems. Systems tailored for film pre-visualization authoring employ drag-and-drop style user interfaces that make authoring a tedious and manual process (Innoventive_Software 2009). Traditional storyboarding techniques can be used to quickly communicate what is happening in each scene. Limited research has been conducted in applying a storyboarding metaphor for authoring film pre-visualizations (Jhala, Rawls et al. 2008; Skorupski 2009). Research has demonstrated that information can be extracted from sketches (Shin and Igarashi 2007; Lee and Funkhouser 2008) and animated characters from sketches (Davis, Agrawala et al. 2003; Chaudhuri, Kalra et al. 2004; Lin 2006). This research will investigate how film pre-visualizations can be effectively authored using sketch-based interfacing techniques and a storyboarding metaphor. A prototype system will be implemented in order to conduct a usability evaluation for assessing the usability benefits of sketch-based pre-visualization authoring using storyboards.
METHODOLOGY
n) Briefly state the methodology (specifically the procedure in which human subjects will be participating) (the full protocol is to be included): A sketch-based pre-visualization authoring tool on a tablet device is being developed and will be used to determine if sketch-based user interfaces can effectively support the authoring of animated pre-visualizations. A user study will be conducted in order to evaluate the usability and authoring effectiveness of the prototype. Participants will be asked to author a pre-visualization from a case study. Performance and self-reported metrics will be evaluated to determine the effectiveness and efficiency of the prototype. These metrics will provide insight into the usability and user satisfaction of the prototype tool. Participants will be given a pre-task questionnaire, a set of tasks detailing the authoring process and thereafter asked to complete a post-task questionnaire.
o) State the minimum and maximum number of participants involved (Minimum number should reflect the number of participants necessary to make the study viable) Min: 15 Max: 30

2. RISKS AND BENEFITS OF THIS STUDY

a) Is there any risk of harm, embarrassment or offence, however slight or temporary, to the participant, third parties or to the community at large? No If YES, state each risk, and for each risk state i) whether the risk is reversible, ii) whether there are alternative procedures available and iii) whether there are remedial measures available. Not applicable
b) Has the person administering the project previous experience with the particular risk factors involved? No If YES, please specify: Not applicable

c) Are any benefits expected to accrue to the participant (e.g. improved health, mental state, financial etc.)? No If YES, please specify the benefits: Not applicable
d) Will you be using equipment of any sort? Yes If YES, please specify: Tablet PC : Asus EEE Slate EP121
e) Will any article of property, personal or cultural be collected in the course of the project? No If YES, please specify: Not applicable

3. TARGET PARTICIPANT GROUP

a) If particular characteristics of any kind are required in the target group (e.g. age, cultural derivation, background, physical characteristics, disease status etc.) please specify: Participants will be required to have basic drawing skills and knowledge of using tablet devices.
b) Are participants drawn from NMMU students? Yes
c) If participants are drawn from specific groups of NMMU students, please specify: Participants will be drawn from the Department of Journalism, Media and Philosophy at NMMU. Participants will also be drawn from Triggerfish Animation Studios in Cape Town.
d) Are participants drawn from a school population? No If YES, please specify: Not applicable
e) If participants are drawn from an institutional population (e.g. hospital, prison, mental institution), please specify: Not applicable
f) If any records will be consulted for information, please specify the source of records: Not applicable
g) Will each individual participant know his/her records are being consulted? No If YES, state how these records will be obtained: Not applicable
h) Are all participants over 18 years of age? Yes If NO, state justification for inclusion of minors in study: Not applicable

4. CONSENT OF PARTICIPANTS

a) Is consent to be given in writing? Yes If YES, include the consent form with this application. If NO, state reasons why written consent is not appropriate in this study.
b) Are any participant(s) subject to legal restrictions preventing them from giving effective informed consent? No If YES, please justify: Not applicable
c) Do any participant(s) operate in an institutional environment, which may cast doubt on the voluntary aspect of consent? No If YES, state what special precautions will be taken to obtain a legally effective informed consent: Not applicable
d) Will participants receive remuneration for their participation? No If YES, justify and state on what basis the remuneration is calculated, and how the veracity of the information can be guaranteed. Not applicable
e) Which gatekeeper will be approached for initial permission to gain access to the target group? (e.g. principal, nursing manager, chairperson of school governing body) HOD of the Department of Journalism, Media and Philosophy (Ms B M Wright) ; a producer from Triggerfish Animation Studios (Mr M Buckland) and the NMMU DVC: Academic (Prof T. Mayekiso).
f) Do you require consent of an institutional authority for this study? (e.g. Department of Education, Department of Health) No

If YES, specify: **Not applicable**

5. INFORMATION TO PARTICIPANTS

- a) What information will be offered to the participant before he/she consents to participate? (Attach written information given and any oral information)
- b) Who will provide this information to the participant? (Give name and role)
Timothy Matthews PI
- c) Will the information provided be complete and accurate? **Yes**
If NO, describe the nature and extent of the deception involved and explain the rationale for the necessity of this deception: **Not applicable**

6. PRIVACY, ANONYMITY AND CONFIDENTIALITY OF DATA

- a) Will the participant be identified by name in your research? **No**
If YES, justify: **Not applicable**
- b) Are provisions made to protect participant's rights to privacy and anonymity and to preserve confidentiality with respect to data? **Yes**
If NO, justify. If YES, specify: **Participants will be assigned, and referred to by anonymous participant numbers in this study and will not be mentioned by name.**
- c) If mechanical methods of observation be are to be used (e.g. one-way mirrors, recordings, videos etc.), will participant's consent to such methods be obtained? **Yes**
If NO, justify: **No mechanical methods of observation are being used**
- d) Will data collected be stored in any way? **Yes**
If YES, please specify: (i) By whom? (ii) How many copies? (iii) For how long? (iv) For what reasons? (v) How will participant's anonymity be protected? **(i) Dr. Dieter Vogts and Mr Timothy Matthews (ii) one copy (iii) Five years (iv) Data analysis and Reporting purposes (v) Participants will be assigned participant numbers**
- e) Will stored data be made available for re-use? **No**
If YES, how will participant's consent be obtained for such re-usage? **Not applicable**
- f) Will any part of the project be conducted on private property (including shopping centres)? **Yes**
If YES, specify and state how consent of property owner is to be obtained: **Permission will be obtained from a producer from Triggerfish Animation Studios, Mike Buckland**
- g) Are there any contractual secrecy or confidentiality constraints on this data? **No**
If YES, specify: **Not applicable**

7. FEEDBACK

- a) Will feedback be given to participants? **No**
If YES, specify whether feedback will be written, oral or by other means and describe how this is to be given (e.g. to each individual immediately after participation, to each participant after the entire project is completed, to all participants in a group setting, etc.): **Not applicable**
- b) If you are working in a school or other institutional setting, will you be providing teachers, school authorities or equivalent a copy of your results? **Not applicable**
If YES, specify, if NO, motivate:

8. ETHICAL AND LEGAL ASPECTS

The Declaration of Helsinki (2000) or the Belmont Report will be included in the references: **No**

If NO, motivate: **The report is not applicable to this study**

(A copy of the Belmont Report is available at the following link for reference purposes: <http://www.nmmu.ac.za/documents/rcd/The%20Belmont%20Report.pdf>)

a) I would like the REC-H to take note of the following additional information:

The latest date for the commencement of data collection is the 3rd of September 2012 and is expected to last until the 30th of September 2012. Thereafter data analysis will be performed and the Masters Dissertation will be submitted at the end of November 2012. The successful completion of the Masters qualification depends on the date at which data collection may commence.

9. DECLARATION

If any changes are made to the above arrangements or procedures, I will bring these to the attention of the Research Ethics Committee (Human). I have read, understood and will comply with the *Guidelines for Ethical Conduct in Research and Education at the Nelson Mandela Metropolitan University* and have taken cognisance of the availability (on-line) of the Medical Research Council Guidelines on Ethics for Research (<http://www.sahealthinfo.org/ethics/>). All participants are aware of any potential health hazards or risks associated with this study.

I am not aware of potential conflict(s) of interest which should be considered by the Committee.

If affirmative, specify: **Not applicable**

09 April 2013

SIGNATURE: **Dr. Dieter Vogts** (Primary Responsible Person)

Date

09 April 2013

SIGNATURE: **Timothy Matthews** (Principal Investigator/Researcher)

Date

10. SCRUTINY BY FACULTY AND INTRA-FACULTY ACADEMIC UNIT

This study has been discussed, and is supported, at Faculty and Departmental (or equivalent) level. This is attested to by the signature below of a Faculty (e.g. RTI) and Departmental (e.g. HoD) representative, neither of whom may be a previous signator.

NAME and CAPACITY (e.g. HoD)

SIGNATURE

Date

NAME and CAPACITY (e.g. Chair:FacRTI)

SIGNATURE

Date

Appendix E :

Ethics Approval (Letter)



**Nelson Mandela
Metropolitan
University**

for tomorrow

**Chairperson of the Research Ethics Committee (Human)
NMMU**

Tel . +27 (0)41 504-2235

• PO Box 77000 • Nelson Mandela Metropolitan University
• Port Elizabeth • 6031 • South Africa • www.nmmu.ac.za

Ref: [H12-SCI-CS-015/Approval]

Contact person: Mrs U Spies

21 September 2012

Dr D Vogt
NMMU
Faculty of Science
Embizweni Building
Room 09-02-18
Summerstrand South Campus

Dear Dr Vogt

SKETCH-BASED DIGITAL STORYBOARDS FOR AUTHORIZING FILM PRE-VISUALIZATIONS

PRP: Dr D Vogt
PI: Mr T Matthews

Your above-entitled application for ethics approval served at the Research Ethics Committee (Human).

We take pleasure in informing you that the application was approved by the Committee.

The ethics clearance reference number is **H12-SCI-CS-015**, and is valid for three years. Please inform the REC-H, via your faculty representative, if any changes (particularly in the methodology) occur during this time. An annual affirmation to the effect that the protocols in use are still those for which approval was granted, will be required from you. You will be reminded timeously of this responsibility, and will receive the necessary documentation well in advance of any deadline.

We wish you well with the project. Please inform your co-investigators of the outcome, and convey our best wishes.

Yours sincerely

A handwritten signature in cursive script that reads "CB Cilliers".

Prof CB Cilliers
Chairperson: Research Ethics Committee (Human)

cc: Department of Research Capacity Development
Faculty Officer: Science

Appendix F :

Heuristics Checklist

Heuristics for GUI Design	Poor	Good ⁷	Excellent
1. Visibility of system errors			✓
2. Match between system and real world			✓
3. User control and freedom		✓	
4. Constancy and standards			✓
5. Error prevention			✓
6. Recognition rather than recall		✓	
7. Flexibility and efficiency of use			✓
8. Aesthetic and minimalist design			✓
9. Recognise and recover from errors		✓	
10. Help and documentation	✗		
Heuristics for Sketch-based Recognition	Poor	Good	Excellent
1. Efficient and reliable recognition triggers		✓	
2. Separate recognised and unrecognised objects			✓
3. Minimize clutter and user sketch transformations			✓
4. Allow errors to be corrected after sketching			✓
5. Predictable and understandable recognition errors	✗		
2D/3D Heuristics for Tablets	Poor	Good	Excellent
1. Provide onscreen touch gestures			✓
2. Avoid gestures which are too similar		✓	
3. Reduce the need for overlaying controls			✓
4. Only use 2D icons in 2D space			✓
5. Use of simple, large and consistent icons			✓

⁷ The extent to which each heuristic applies to the design of the prototype was rated as follows:

- Poor: Serious usability issues have been identified.
- Good: Minor usability issues have been identified.
- Excellent: No usability issues have been identified.

Appendix G:

Informed Consent Form

NELSON MANDELA METROPOLITAN UNIVERSITY
INFORMATION AND INFORMED CONSENT FORM

RESEARCHER'S DETAILS	
Title of the research project	Sketch-based Digital Storyboards for Authoring Pre-visualisations
Reference number	H12-SCI-CS-015
Principal investigator	Timothy Matthews
Contact telephone number (private numbers not advisable)	041 504 2094

A. DECLARATION BY OR ON BEHALF OF THE PARTICIPANT		Initial
I, the participant and the undersigned	(full names)	

A.1. HEREBY CONFIRM AS FOLLOW		Initial
I, the participant was invited to participate in the above-mentioned research project that is being undertaken by		
from	Timothy Matthews	
	Department of Computing Sciences	
Of the Nelson Mandela Metropolitan University		

A.2 THE FOLLOWING ASPECTS HAVE BEEN EXPLAINED TO ME, THE PARTICIPANT			Initial
Aim	The main objective of this study is to determine how effectively sketch-based user interfaces can support the authoring of animated pre-visualisations. The information will be used to/for research purposes		
Procedures	I understand that I am required to use a system in order to evaluate how effectively sketch-based user interfaces can be used to support the authoring of animated pre-visualisations		
Risks	I understand that there are no risks involved in participating in this process		
Confidentiality	My identity will not be revealed in any discussion, description or scientific publications by the investigators		
Voluntary participation / refusal / discontinuation	My participation is voluntary	YES NO	
	My decision whether or not to participate will in no way affect my present or future career/employment/lifestyle	TRUE FALSE	

No pressure was exerted on me to consent to participate and I understand that I may	
---	--

Appendix G: Informed Consent Form

withdraw at any stage without penalisation	
Participation in this study will not result in any additional cost to myself	
I HEREBY VOLUNTARILY CONSENT TO PARTICIPATE IN THE ABOVE-MENTIONED PROJECT:	
Signed/confirmed at	on 20
Signature	Signature of the witness:
	Full name of witness:

B. STATEMENT BY OR ON BEHALF OF INVESTIGATOR(S)					
	I, (name of interviewer)	declare that:			
1.	I have explained the information given in this document to	(name of patient/participant)			
	And / or his / her representative	(name of representative)			
2	He / She was encouraged and given ample time to ask me any questions;				
3	This conversation was conducted in	Afrikaans	English	Xhosa	Other
	and no translator was used <u>OR</u> this conversation was translated into				
	(language)	by	(name of translator)		
4.	I have detached section D and handed it to the participant	Yes	No		
Signed /confirmed at		on		20	
Signature of the interviewer	Signature of witness:				
	Full name of witness:				

Appendix H:



Task List

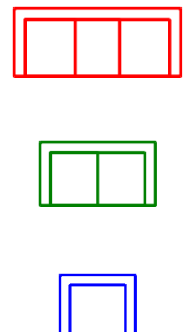
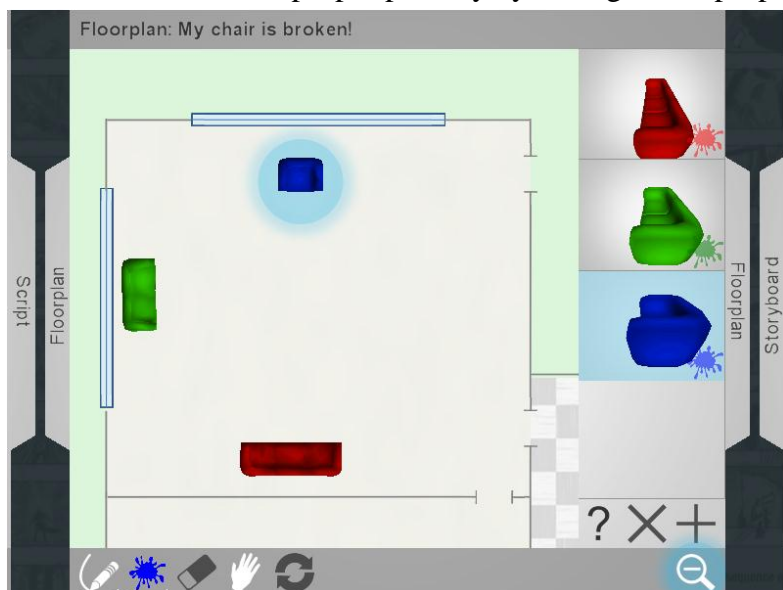
Tasks Description

Pre-visualisations are mock up computer generated previews that are used to visualise a film before or during production. You will be asked to perform the set of tasks using the sketch-based pre-visualisation authoring tool provided in order to author a pre-visualisation for a case study. You will then be given a questionnaire to complete regarding the use of the system. Please feel free to ask me questions at any point if you feel lost or confused and I will assist you. The main focus of the prototype tool is to support effective pre-visualisation authoring using a simple and easy to use sketch-based storyboarding interface.

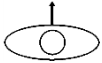
Task 1: Staging

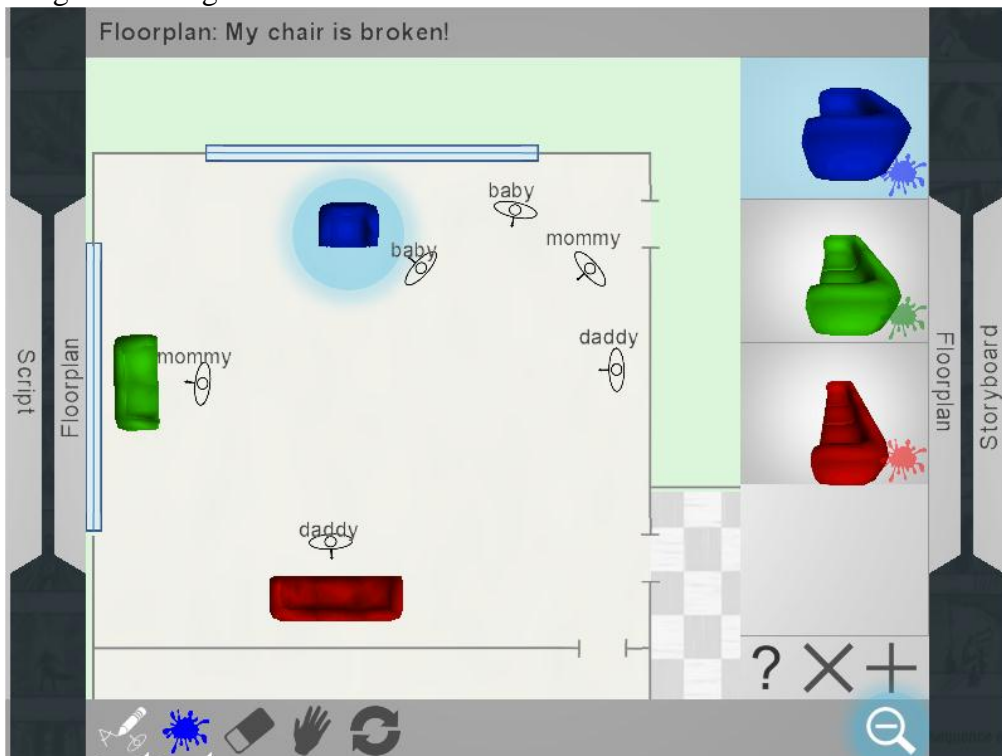
In this task you will be required to perform the staging for one narrative block within the "**Goldilocks and the Three Bears**" case study. The case study begins where the three bears enter the house after Goldilocks falls asleep. The narrative block entitled "**My chair is broken!**" ends after the baby bear becomes hungry.

- 1.1 Select the narrative block entitled "**My chair is broken!**", and navigate to the floor planning screen.
- 1.2 Draw three chairs in the living room: A red 3-seat couch, a green 2-seat couch and a blue 1-seat couch as shown below using the prop pencil.  The props are listed on the right hand of the screen. Draw each prop separately by adding a new prop using the  button.

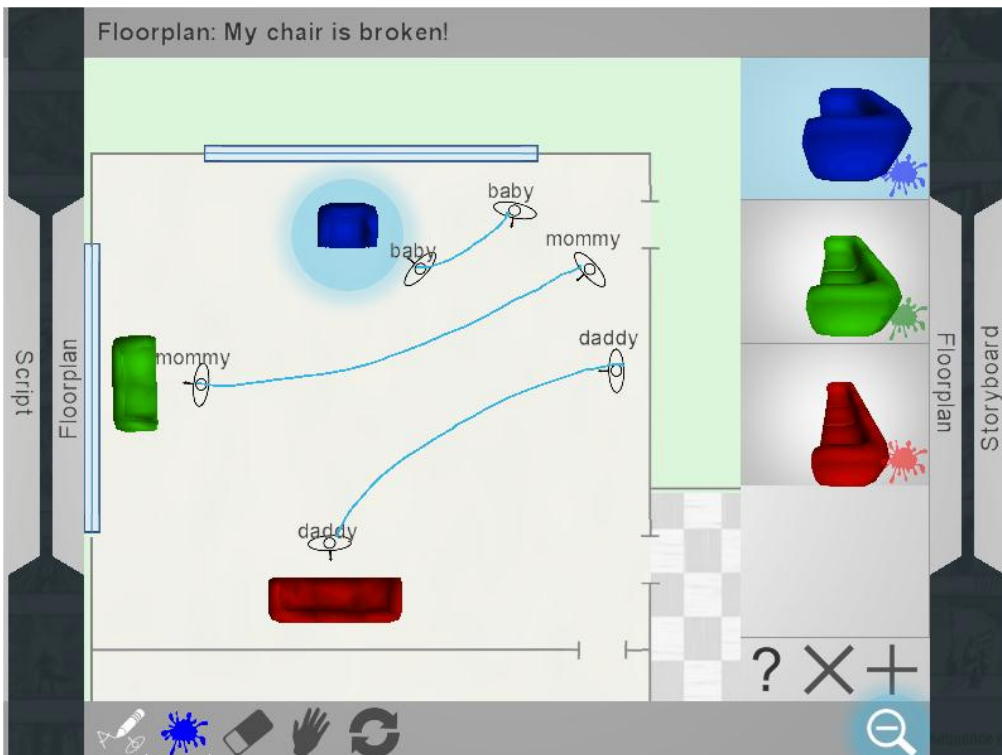



1.3 Select the symbol  using the pencil drop down box.

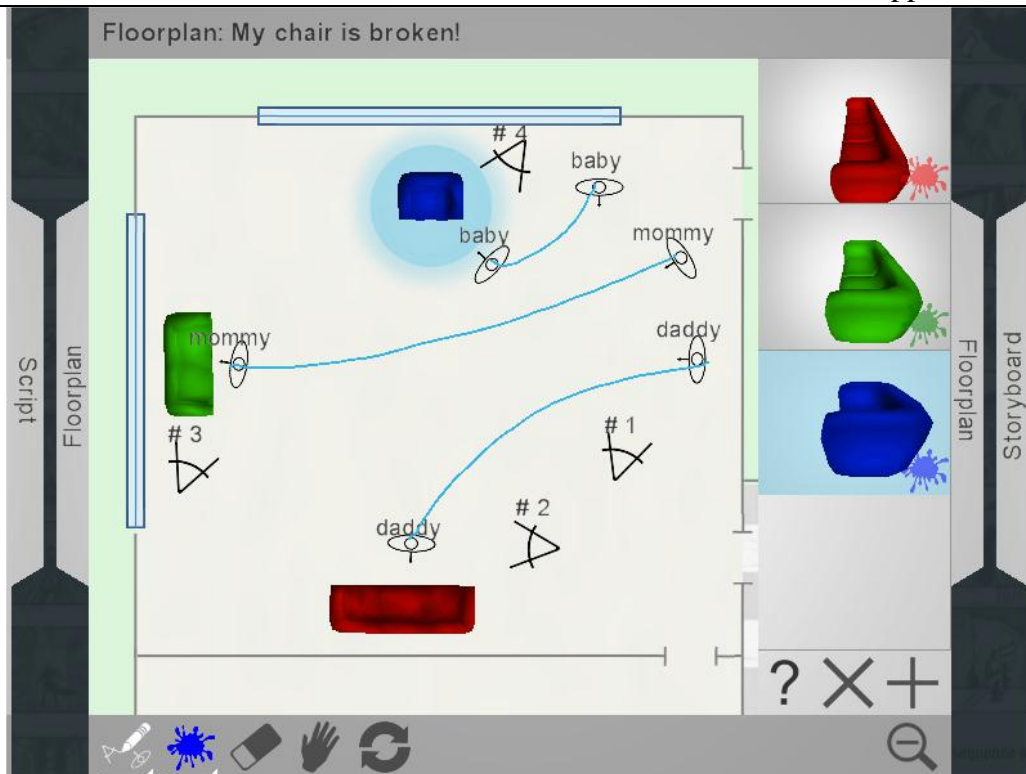
Block the daddy bear, mommy bear and baby bear by drawing character symbols  at their starting and ending locations as shown below.



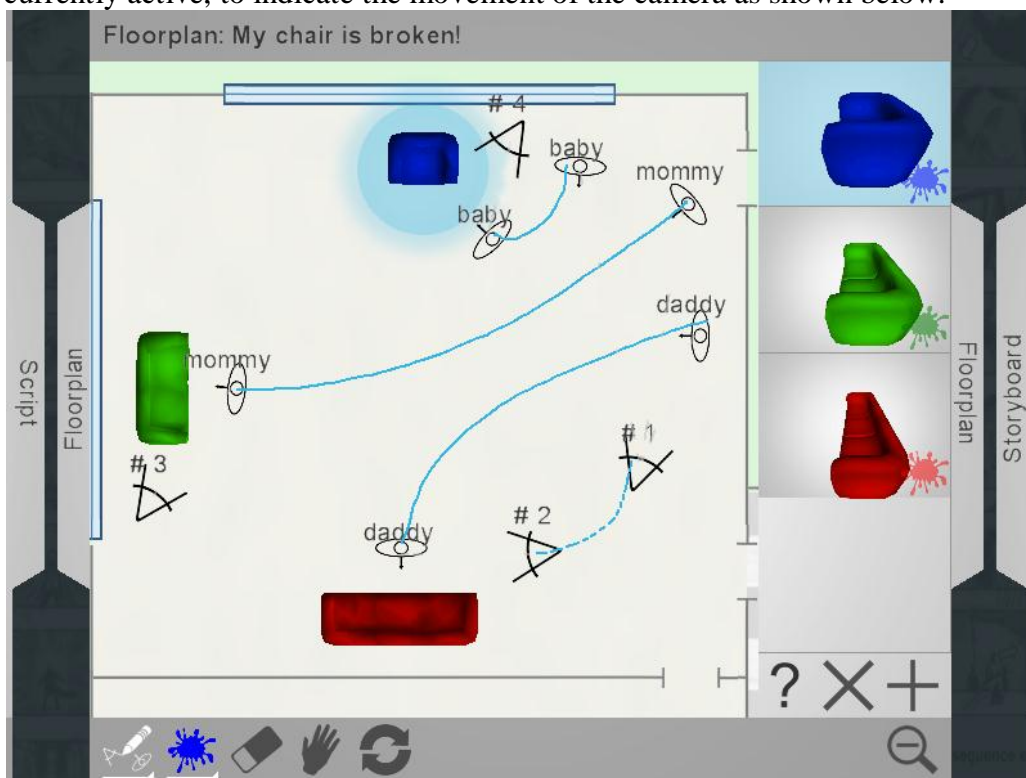
1.4 Connect the bears' starting and ending positions using the annotation pencil, which is currently active, to indicate the movement of each bear as shown below.



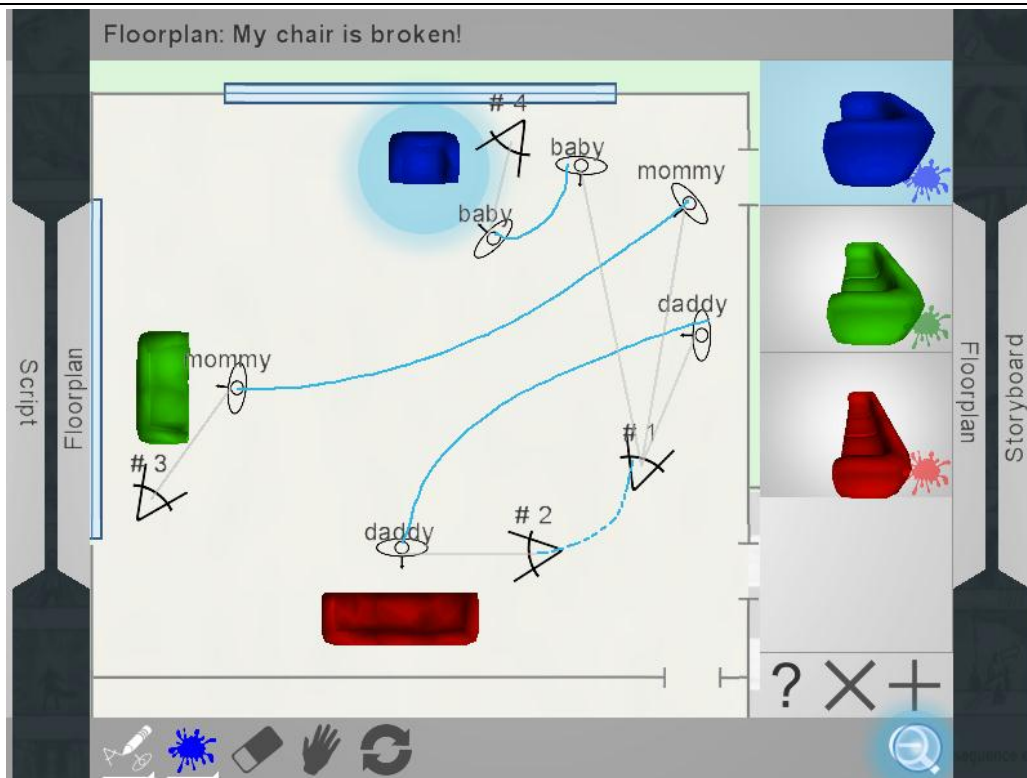
1.5 Draw four cameras on the floor plan by sketching the  symbols as shown below.



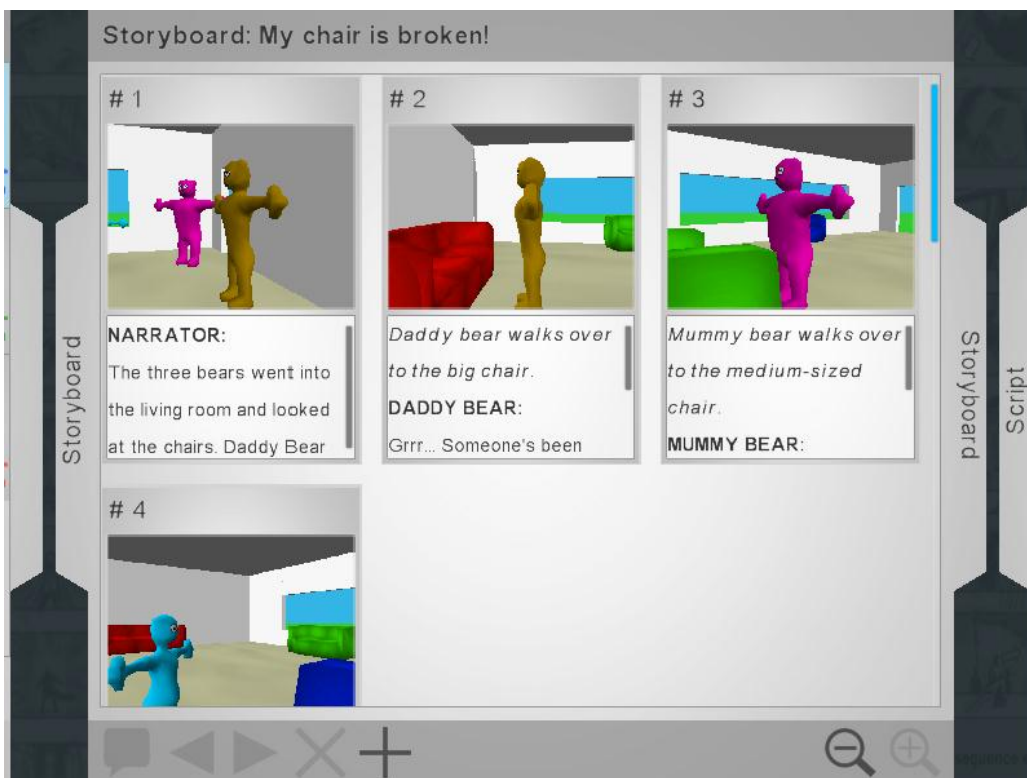
1.6 Connect the first and second camera's starting and ending positions using the annotation pencil, which is currently active, to indicate the movement of the camera as shown below.



1.7 Associate the camera symbols of each shot with the characters' they contain by connecting the shot symbol with the relevant characters as shown below.



1.8 Navigate to the storyboard screen and add dialogue and action  as shown below.

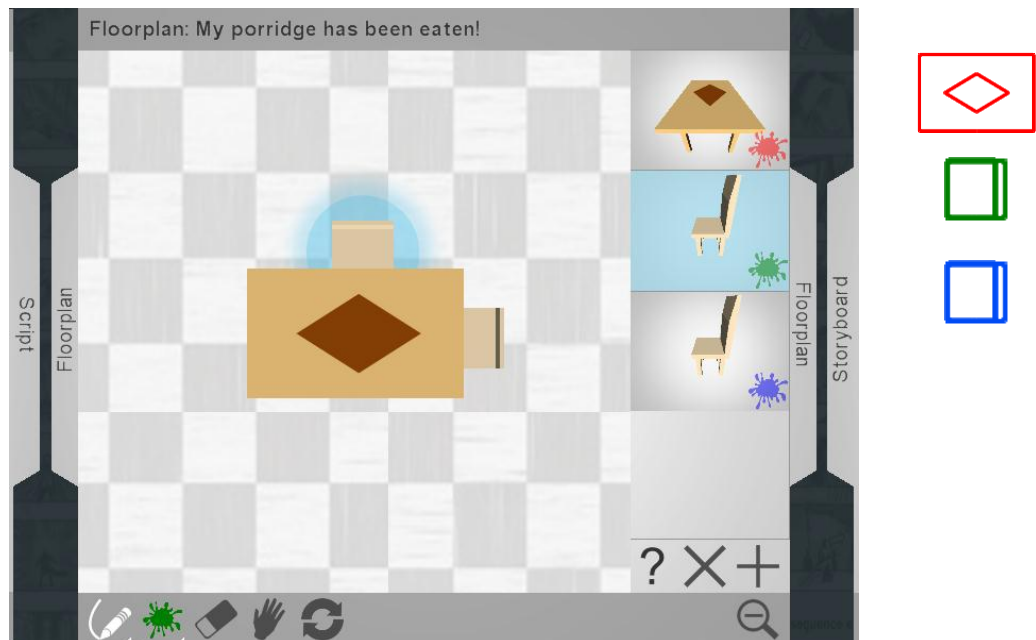


Please complete the questionnaire for Task 1.

Task 2: Storyboarding

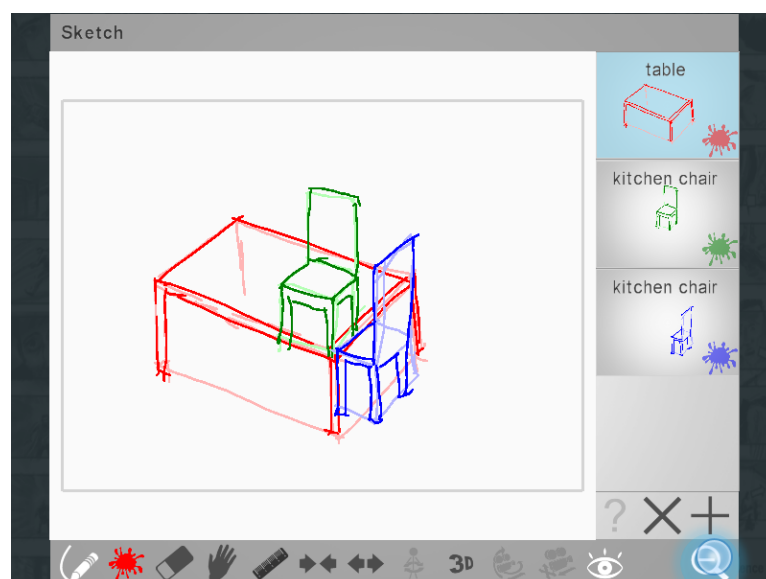
In this task the user will be required to edit the storyboard for two narrative blocks.

- 2.1 Navigate to the narrative block entitled “**My porridge has been eaten!**” and view its floor plan. In the kitchen, add two kitchen chairs and a table as shown in the figure below. The **table is red**, the **top chair is green** and the **right-hand chair is blue**.





- 2.2 Navigate to the storyboard screen and add a new, blank storyboard panel.
- 2.3 Zoom into the storyboard panel and sketch the table and the chairs as shown below. Sketch **each prop on a separate layer** and using the **correct colour**. Use the drafting pencil for drafting and the solid pencil for the final finishing.

Touch the **3D button** **3D** when you are finished.

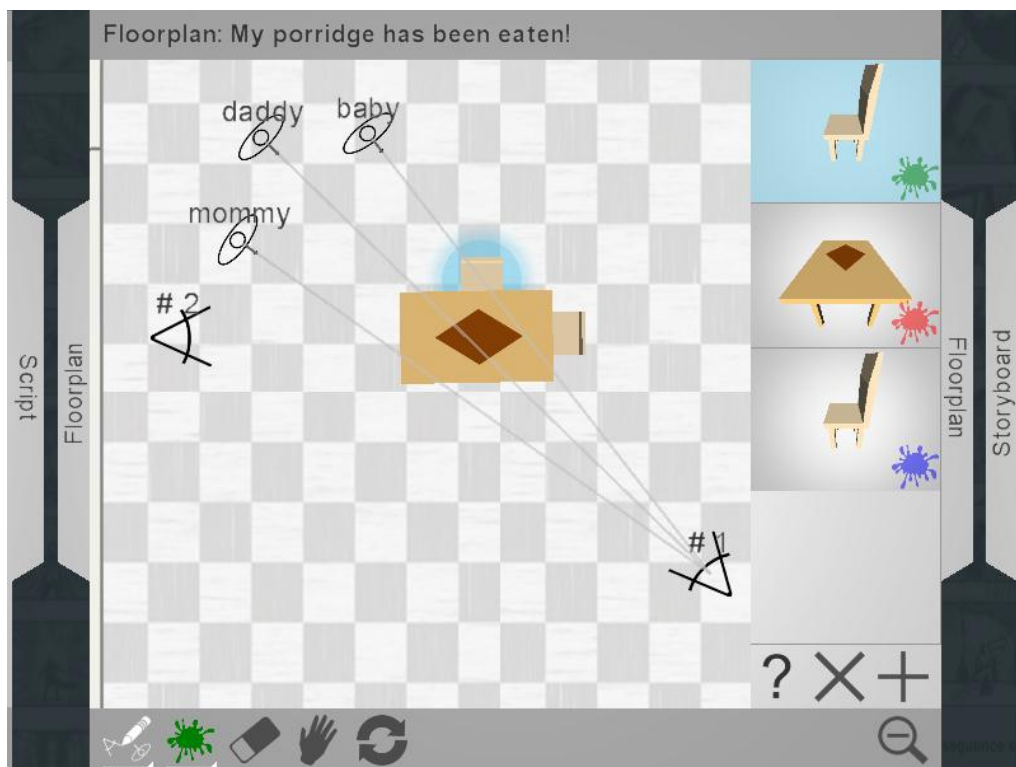


- 2.4 Add and sketch the three happy bears by sketching their stick figures as shown below. Use the



emotion pencil  to indicate the facial expressions of each bear. 

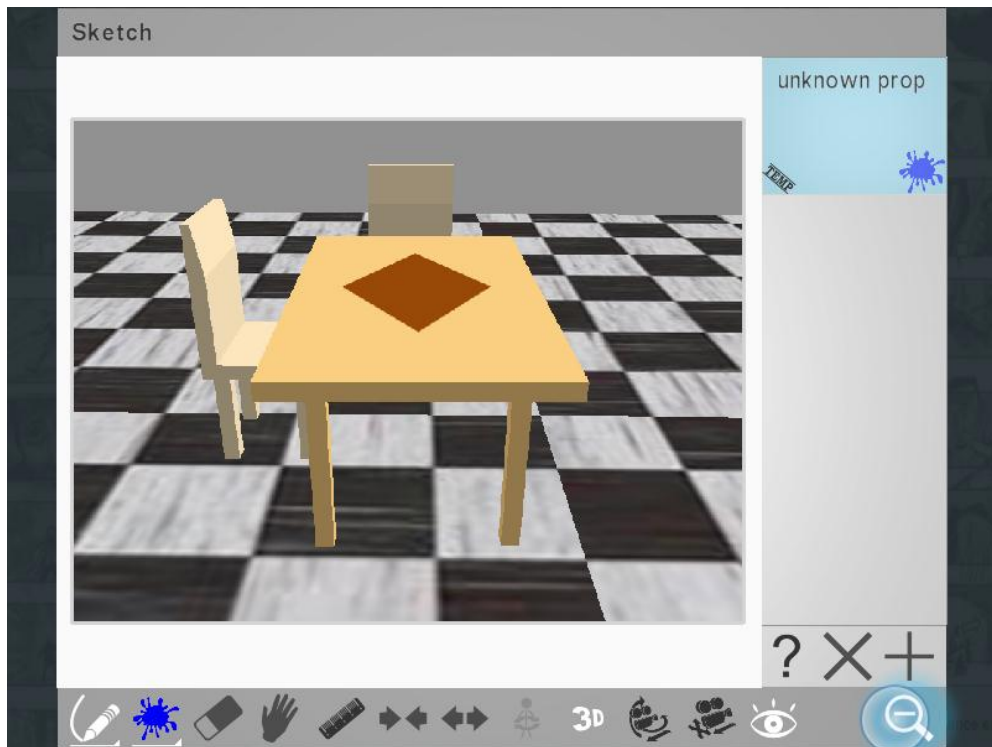


2.5 Navigate to the floor planning screen and add another shot (#2) using the annotation pencil as shown below.

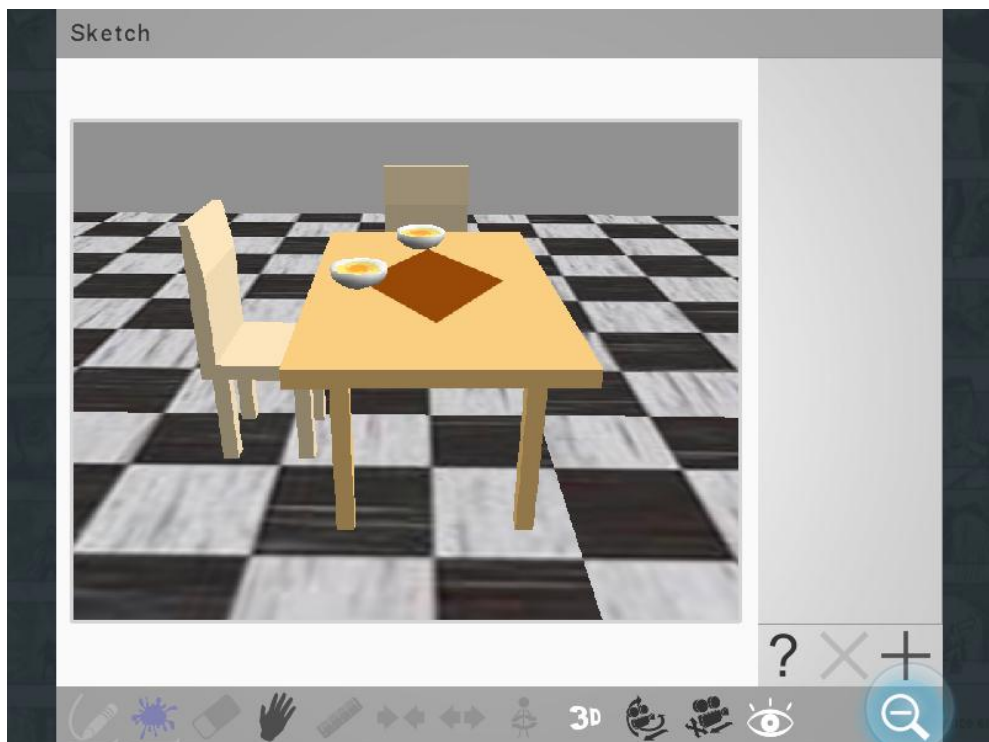



2.6 Navigate to the storyboarding screen and zoom in on the last storyboard panel (shot #2).

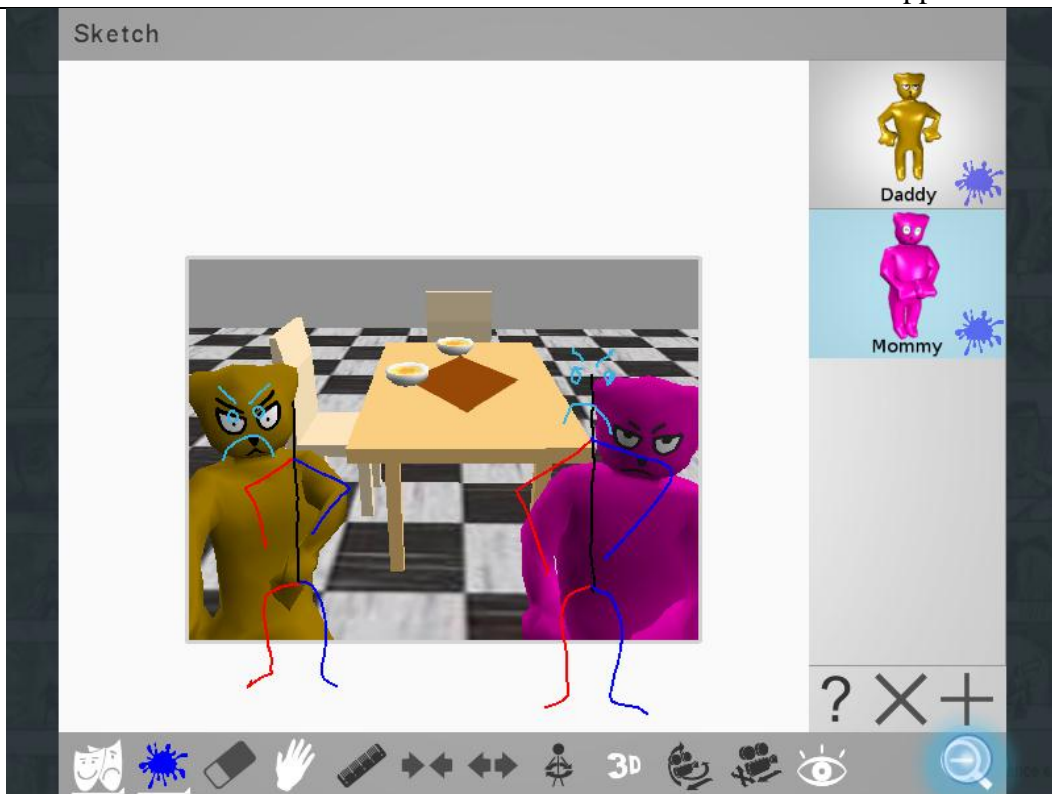
2.7 Using the touch surface to rotate and move the camera using the **move camera**  and **rotate camera**  tools in order to get the shot below.



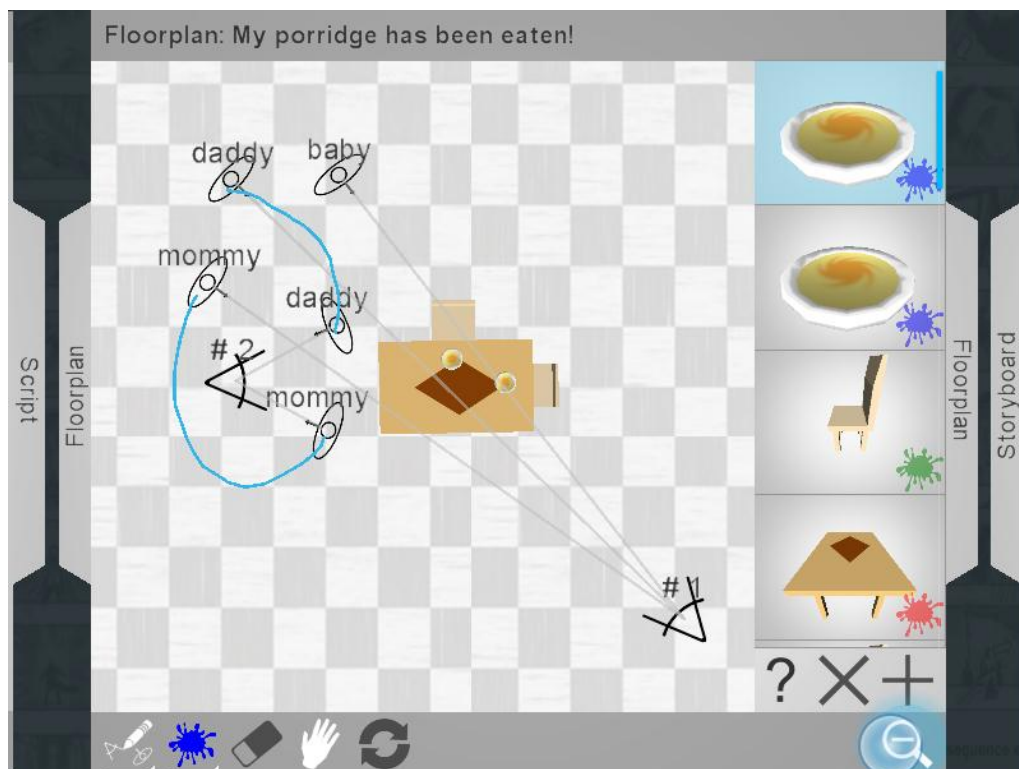
2.8 Add two bowls to the list of layers and sketch them on the table as shown below.



2.9 Add two angry bears around the table as shown below. 



2.10 Navigate to the floor plan and indicate how the bears move in the kitchen as shown below.



Please complete the questionnaire for Task 2.

Thank you for participation!

Appendix I :

Pre-Task Questionnaire

Pre-Task Questionnaire		
1. What is your current occupation (e.g. media student, animator, producer, storyboard artist)		
2. Do you have any experience with sketching storyboards? If so, then please mention what experience you have.	Yes	No
3. Do you have any experience with performing pre-production activities? If so, then please mention what experience you have.	Yes	No
4. Do you have any experience with sketching storyboards? If so, then please mention what experience you have.	Yes	No
5. Do you have any experience with creating 3D computer generated animations? If so, then please mention what experience you have.	Yes	No
6. Do you have any experience with any other tasks related to the pre-production and production stages of filmmaking? If so, then please mention what experience you have.	Yes	No

Appendix J :

Performance Sheet

Task 1: Staging

Task	Success	Time	Errors
1.1			
1.2			
1.3			
1.4			
1.5			
1.6			
1.7			
1.8			

Task 2: Storyboarding

Task	Success	Time	Errors
2.1			
2.2			
2.3			
2.4			
2.5			
2.6			
2.7			
2.8			
2.9			
2.10			

Appendix K:

Post-Task Questionnaire (Staging)

Staging: Post-Task Questionnaire							
A. Cognitive load							
1. Mental demand: How mentally demanding were the tasks?							
	Very Low	1	2	3	4	5	Very High
2. Physical demand: How physically demanding were the tasks?							
	Very Low	1	2	3	4	5	Very High
3. Temporal demand: How hurried or rushed was the pace of the tasks?							
	Very Low	1	2	3	4	5	Very High
4. Performance: How successful were you in accomplishing what you were asked to do?							
	Very Low	1	2	3	4	5	Very High
5. Effort: How hard did you have to work to accomplish your level of performance?							
	Very Low	1	2	3	4	5	Very High
6. Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?							
	Very Low	1	2	3	4	5	Very High
B. Overall satisfaction							
1. Overall, I am satisfied with how easy it is to use the staging screen.							
	Strongly disagree	1	2	3	4	5	Strongly agree
2. Overall, I am satisfied with the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
3. It was easy to learn to use the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
4. It was simple to use the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree

Appendix K: Post-Task Questionnaire (Staging)

C. Usability							
1. I can easily author the pre-visualisation for the case study using the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
2. I was able to efficiently author the pre-visualisation for the case study using the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
3. I became productive quickly using the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
4. The staging screen has all functions and capabilities I expect from a staging tool							
	Strongly disagree	1	2	3	4	5	Strongly agree
5. I can effectively author the pre-visualisation for the case study using the staging screen							
	Strongly disagree	1	2	3	4	5	Strongly agree
6. I am satisfied with the touch interaction provided by the software							
	Strongly disagree	1	2	3	4	5	Strongly agree
D. General							
1. Identify the most positive aspects of the staging interface.							
2. Identify the most negative aspects of the staging interface.							
3. Please provide any general comments or suggestions for improvement for the staging interface.							

Appendix L :

Post-Task Questionnaire (Storyboarding)

Storyboarding: Post-Task Questionnaire							
E. Cognitive load							
1. Mental demand: How mentally demanding were the tasks?							
	Very Low	1	2	3	4	5	Very High
2. Physical demand: How physically demanding were the tasks?							
	Very Low	1	2	3	4	5	Very High
3. Temporal demand: How hurried or rushed was the pace of the tasks?							
	Very Low	1	2	3	4	5	Very High
4. Performance: How successful were you in accomplishing what you were asked to do?							
	Very Low	1	2	3	4	5	Very High
5. Effort: How hard did you have to work to accomplish your level of performance?							
	Very Low	1	2	3	4	5	Very High
6. Frustration: How insecure, discouraged, irritated, stressed, and annoyed were you?							
	Very Low	1	2	3	4	5	Very High
F. Overall satisfaction							
1. Overall, I am satisfied with how easy it is to use the storyboarding interface.							
	Strongly disagree	1	2	3	4	5	Strongly agree
2. Overall, I am satisfied with the storyboarding interface.							
	Strongly disagree	1	2	3	4	5	Strongly agree
3. It was easy to learn to use the storyboarding interface.							
	Strongly disagree	1	2	3	4	5	Strongly agree
4. It was simple to use the storyboarding interface.							
	Strongly disagree	1	2	3	4	5	Strongly agree
5. I would like to use sketched-based authoring software in the future instead of traditional pre-visualisation authoring tools (e.g. Storyboard pro, Maya 3D, paper based storyboards).							
	Strongly disagree	1	2	3	4	5	Strongly agree

Appendix L: Post-Task Questionnaire (Storyboarding)

6. Please motivate the score you selected in the above statement (F.5).

G. Usability

1. I can easily author the pre-visualisation for the case study using the storyboarding interface.	Strongly disagree	1	2	3	4	5	Strongly agree
2. I was able to efficiently author the pre-visualisation for the case study using the storyboarding interface.	Strongly disagree	1	2	3	4	5	Strongly agree
3. I became productive quickly using the storyboarding interface.	Strongly disagree	1	2	3	4	5	Strongly agree
4. The storyboarding interface has all functions and capabilities I expect from a sketched based storyboarding tool.	Strongly disagree	1	2	3	4	5	Strongly agree
5. I can effectively author the pre-visualisation for the case study using the storyboarding interface.	Strongly disagree	1	2	3	4	5	Strongly agree
6. I am satisfied with the touch interaction provided by the software.	Strongly disagree	1	2	3	4	5	Strongly agree

H. General

1. Identify the most positive aspects of the storyboarding interface.
2. Identify the most negative aspects of the storyboarding interface.
3. Please provide any general comments or suggestions for improvement.