Super features: a probabilistic approach to feature matching and correction

by

Yuko Roodt

A dissertation submitted in partial fulfillment of the degree

Doctor Philosophiae

in

Electrical and Electronic Engineering

in the

Faculty of Engineering and the Built Environment

at the

University of Johannesburg

Supervisor : Prof A.L. Nel

Submitted : August 2015

Declaration

I, Yuko Roodt, hereby state that this thesis in entirety is my own, original work and has not been handed in by anyone else for academic purposes. I further state that I know what plagiarism is and that everything done is of my own work except where others has been referred to and clearly referenced. I further understand that any unethical academic behaviour, which includes plagiarism, is regarded by the University of Johannesburg as a serious offence which will result in a disciplinary hearing.

Wednesday, 5 August 2015

Signature:	**Date**
Yuko Roodt	

Acknowledgements

I would like to thank Prof. A.L. Nel for his help, advise and insights. I appreciate the time that he has invested in me during the development of this work. I would also like to acknowledge the support and love provided by my wife, Verüschka, and our family. Without their encouragement and motivation this task would have felt impossible.

Abstract

Reliable detection and matching of visual features is an important problem in the field of computer and robotic vision. It mimics the behaviour and function of the parts of the visual cortex required for object recognition, scene understanding and 3-dimensional reconstruction. Feature detection and matching can be influenced negatively by a number of environmental and optical effects such as occlusions, duplicate features, motion, illumination changes, image noise and 3-dimensional transformations and distortions. Methods of improving the matching results for these situations have been proposed with limited success. A probabilistic feature matching and correction framework is proposed, that uses insights provided by the geometric relationships observed between different features to allow Super-feature clusters to be constructed. These Super-feature clusters allow features to be matched using visual similarity as well as geometric consistency, enabling many matching issues to be resolved. The geometric relationships are combined in a translation, scale, rotation and affine invariant manner, allowing features experiencing global as well as local motion, such as can be found on dynamic, non-rigid and morph-able objects, to be matched. Invalid feature matches can be detected and corrected using the probability density distribution derived from the geometric observations. An efficient method of constructing and evaluating the probability density distribution of the position estimates is provided, allowing the modes of the distribution to be located. The Super-feature matching and correction framework can be used in conjunction with many state of the art feature detectors that provide feature position, rotation and description information, allowing their feature matching accuracy and feature usage to be improved.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Object recognition in the human visual system using features can be motivated from cell level observations in the visual cortex of mammals [1], and also in a number of other species such as the Cephalopod [2] and the Sesarma leptosoma [3]. Visual features act as a form of data compression in the early stages of the visual system allowing only important image information to be processed and retained, and unimportant information to be discarded [4]. This simplifies and reduces the amount of processing resources required for complex vision tasks. Since many Computer Vision algorithms attempt to mimic the behavior and function of the visual cortex, a similar approach is followed for high-level Computer Vision. Matching of features and establishing correspondences between different images is an important problem and forms the basis of many high level vision tasks such as:

- Object recognition and classification [5]
- Content based image retrieval [6]
- Image registration [7], alignment [8], mosaicing [9] and panorama stitching [10]
- Pose estimation [11] and Camera motion tracking [12]

- 3-dimensional reconstruction and modelling [13]
- Robotic and vision based navigation and control [14]
- Augmented reality [15]

Feature detection in digital images consist of detecting and localizing small regions in an image which are considered information rich. The image information surrounding these localized regions are then described in a compact form that enables matching of features. The way in which the compact description is generated can improve matching performance under varying illumination and noisy images. This is achieved by normalizing the description and introducing invariance in the feature descriptor to these variable conditions. The compact form or compressed description of the feature's information also improves the processing requirements when matching large numbers of features. The feature matching process usually involves finding the best match for each feature in one set of features compared to another set of features, using appearance similarity. Each feature is then matched, one by one, using the Manhattan, Euclidean or Hamming distance metrics until the best candidate feature is found that matches the template feature. Feature sets can be obtained from a single image, a range of images, video sequences [16] or pre-trained features of an object captured from multiple views [17].

A number of methods and techniques for detecting and matching features in a digital image have been proposed. These algorithms range from simple binary detectors designed for fast execution times to advanced detectors with sub-pixel localization and invariance to affine transformations. The most well known and most popular of these feature detectors are:

- Scale-invariant feature transform (SIFT) [18]
- Speeded Up Robust Features (SURF) [19]
- Maximally Stable Extremal Regions (MSER) [20]
- Features from Accelerated Segment Test (FAST) [21]
- Binary Robust Independant Elementary Features (BRIEF) [22]

- Oriented FAST and Rotated BRIEF (ORB) [23]
- Binary Robust Invariant Scalable Keypoints (BRISK) [24]

1.2 Feature detection and matching difficulties

There are a number of challenges that feature detectors need to address. Since images represent a 3-dimensional scene projected onto a 2-dimensional image plane, features could exhibit appearance changes and distortions related to noise, affine transformations, rotation, scale changes and occlusions. These distortions and artefacts can hinder reliable detection and description of visual features [25].

Occlusion of image regions can occur naturally when motion is present while matching features from different feature sets. Moving objects can occlude image regions located in the background as well as itself and other objects. This will result in a large number of previously observed features to become unobservable, limiting the matching of these features.

Features detected close to motion boundaries will have weak descriptions since motion can cause large parts of the feature's description region to change, making matching of these features difficult. The same problem can occur due to dynamic shadows and atmospheric effects, which can change the local image region used for describing a feature. The feature description is the primary information that is used to match features, while position information can be used as well when only small, predictable motion is present or the motion model is known or has been estimated. This is not normally the case for most applications, therefore weak feature descriptions will have a severe effect on the matching accuracy.

Another problem is duplicate features. Since only a small local region surrounding a feature is used for feature description during the matching process, many similar features with similar descriptions can occur simultaneously in a feature set detected from an image. The severity and likelihood of duplicate features occurring in a natural photograph can be observed in Figure 1.1. The occurrences of duplicate features can severely impact the matching performance of a feature detector. Since multiple copies of the same feature might be present, selecting the correct feature to match might appear to be a difficult task. Some feature

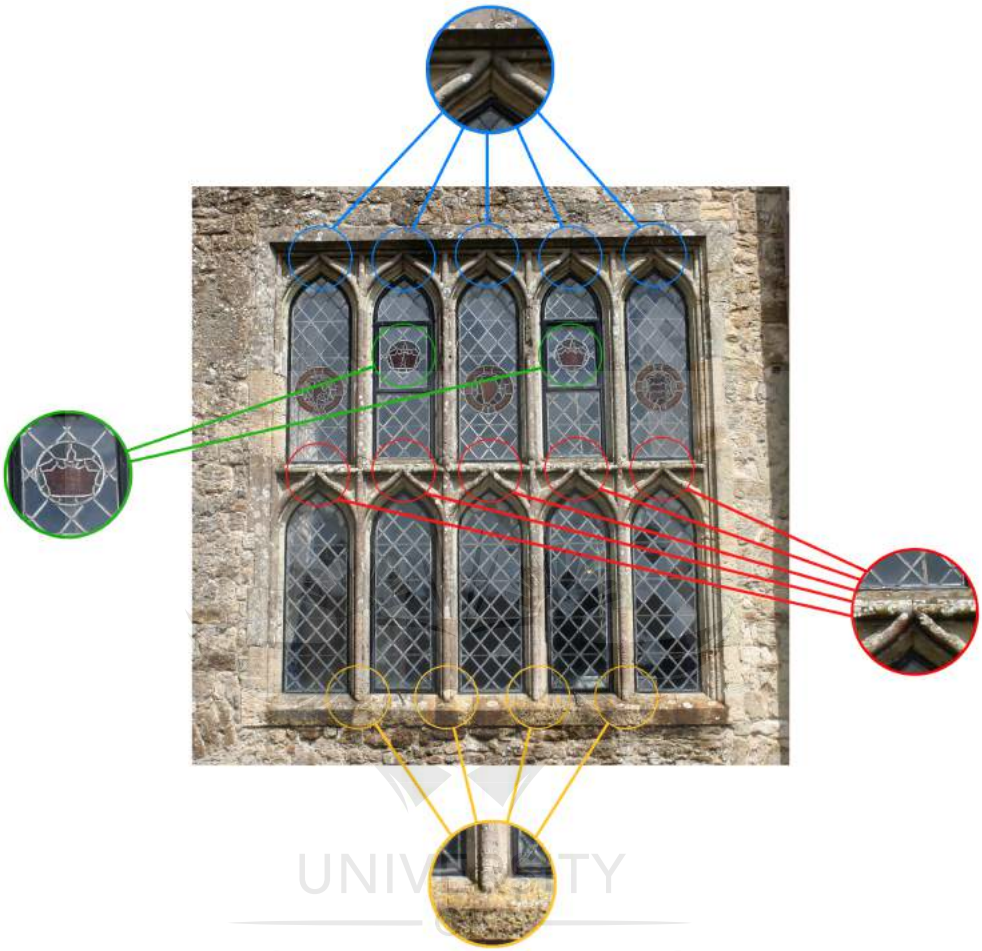


Figure 1.1: Examples of different sets of duplicate features detected in a natural image

detection algorithms realize that these features are difficult to match and choose to discard and ignore features that have multiple possible matches [18]. This is a poor solution to the problem, as many good and descriptive features are removed from the dataset.

A number of techniques have been proposed to improve the matching results. These methods attempt to alleviate these issues by thresholding, discarding and constraining the detected features and matches. The most popular and simplest technique that can be integrated into a feature matching algorithm method is to threshold poor feature matches. Valid feature matches are only made when

a match was found with a sufficiently small matching error. Unfortunately, this only works for some situation; features located on motion borders have poor descriptions that will not produce high matching scores. Similarly, duplicate features might have high matching scores, but have a high probability of being matched incorrectly.

An improvement to this is to determine what the best and second best matching score is, when these match scores are sufficiently close, then the match will be discarded since it cannot be reliably matched as multiple possible match solutions exist [26]. This reduces the number of features that can be matched successfully, which might limit the usefulness of the specific feature detector for some applications.

Methods of tracking features between frames of a video sequence have also been proposed, these methods use position as well as description information to predict a feature's motion, allowing for more accurate matching to be performed [27]. This method of improving feature matches is unfortunately limited to applications that use continuous video sequences and usually only function well when the motion is coherent and predictable.

The method that currently produces the best matching results is based on estimating the motion model from the feature match observations. The type of motion that can be estimated is limited to static scenes that have been transformed by camera motion, where Epipolar geometry and RANSAC is employed to estimate the motion model [10]. Matching features in a scene that is experiencing camera motion as well as having different objects, each with their own dynamic motion, cannot be solved using this technique. Matching of features on dynamic, morphable and non-rigid objects such as the human face or body also pose some difficulty for this method.

1.3 Research Hypothesis

The geometric relationship between a selected primary feature and its neighbouring features, observed in one coordinate frame, can provide insight into the location of the transformed version of that primary feature in another coordinate frame. Grouping of features and their corresponding feature matches will allow

these geometric relationships to be applied to and strengthen the matching process. Each primary feature and its neighbouring features can be grouped to form a Super-feature cluster around the primary feature, which will allow features to be matched using visual similarity and geometric consistency, which will enable stronger, more reliable feature matches to be made.

- Integrating geometric consistency into the feature matching process provided by the Super-feature cluster will allow incorrectly matched features such as features affected by duplicate features, occlusion and motion boundaries to be classified and removed resulting in an increase in the matching accuracy.

- Incorrectly matched features can be improved and corrected, due to the matching search space constraints proposed by the Super-feature cluster, enabling an increase in feature usage.

- The geometric relationships between the different features in the Super-feature cluster, which can be used to localize the primary feature, can be applied in a translation, rotation, scale and an affine invariant manner when the features in a feature sets have been detected to provide feature position and orientation information.

- The feature matching accuracy will improve when features are matched using appearance similarity as well as geometric consistency for scenes experiencing local and global motion of dynamic, non-rigid objects where the motion model is unknown.

1.4 Research methodology

A Design-science research methodology approach will be applied in this thesis. Design-science is a research methodology that is concerned with the development and design of effective artefacts to achieve set objectives, the primary goal of Design-science is utility [28]. On the other hand Natural-science has the primary goal of finding truths, this is achieved by attempting to understand and explain phenomena as a research objective.

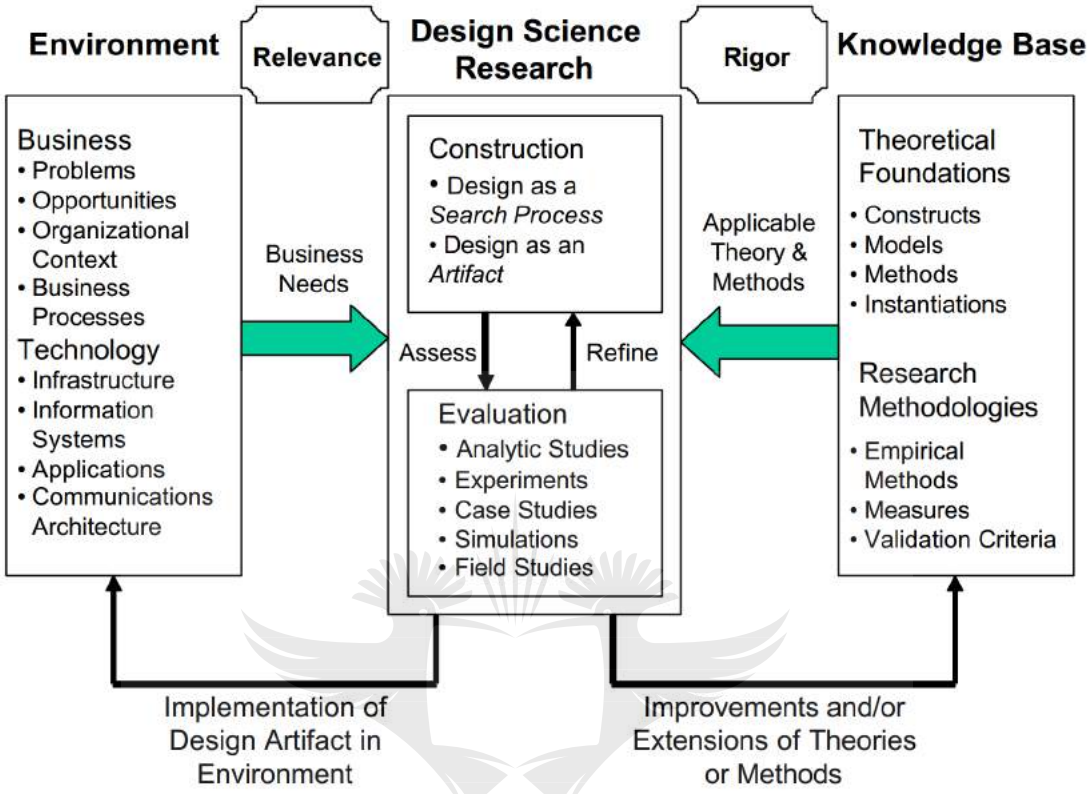


Figure 1.2: Design science research framework [28]

An overview of the Design-science framework is provided in Figure 1.2. It has three primary pillars: the Environment, Design-science research and the Knowledge base, which is connected through the Relevance and Rigour cycle. The environment is composed of the people, business organizations as well as the technology where the problem space or phenomena of interest can be found. This environment contains a number of problems and opportunities also known as business needs defined by the entities in this environment [29]. A benefit of addressing business needs using research ensures that the research is relevant.

There are two primary processes in the Design-science research pillar: Construction and Evaluation which form part of the Design cycle. First, a product or an artefact is built to solve an identified environmental problem defined by the Relevance cycle. This artefact can be one of four different products that are produced by design science research: constructs, models, methods and implemen-

tations [30]. During the artefact design process, scientific theories and methods can be drawn from the knowledge base, this is known as the Rigour cycle.

The next step is to test, assess and evaluate the utility of the solution in solving the problem. Extensive simulations, experimentation and studies can then be used to evaluate the problem solution. The results can then be compared against the original business needs and requirements, allowing the product to be refined and the process of construction and evaluation to be repeated until a desired solution is achieved.

During the Design cycle research, all contribution and additions to the Knowledge base can be produced by building on prior theories and methods. The Design-science research model can contribute in two ways, it can produce design artefacts to benefit and solve problems for the environment or it can improve and extend the current Knowledge base [31].

1.5 Dissertation outline

In Chapter 2, an overview of the feature detection and matching process will be provided as well as the development progress of feature detection, description and matching. The Mean-shift mode finding and clustering algorithm is discussed in Chapter 3. Chapter 4 will provide some insight into feature matching issues and problems and will introduce the Super-feature algorithm, including implementation details. In Chapter 5 the experimental setup and the test datasets will be discussed and the feature matching results obtained with the Super-features algorithm will be provided. Chapter 7 will provide an analysis and breakdown of the observed results and discuss future development opportunities and improvements of the Super-feature algorithm.

Chapter 2

Feature detection and matching

2.1 An overview of the feature detection and matching process

Feature detection is one of the primary functions used in many computer vision algorithms and applications [32]. It enables large amounts of visual data to be processed in an effective and efficient manner by focusing only on regions that are information rich. An overview of the steps involved in the traditional feature detection and matching process, implemented by many vision applications is described and provided in Figure 2.1.

The main objective of feature detection and matching is to find regions or locations that correspond to each other in different images or feature sets. This is achieved by first detecting features in an image or image set. As a minimum requirement, each detected feature requires the position where it was detected as well as a visual description of the feature. The feature description is used for matching features and provides a compact representation that describe the visual appearance of the region surrounding the feature's position. Usually, features that have similar descriptions are related visually and could potentially be matched to each other. Other feature related information can also be derived and included, such as the direction vector, orientation angle, scale and affine transformation of the detected feature [33].

The next step is to find feature matches between different feature sets. The

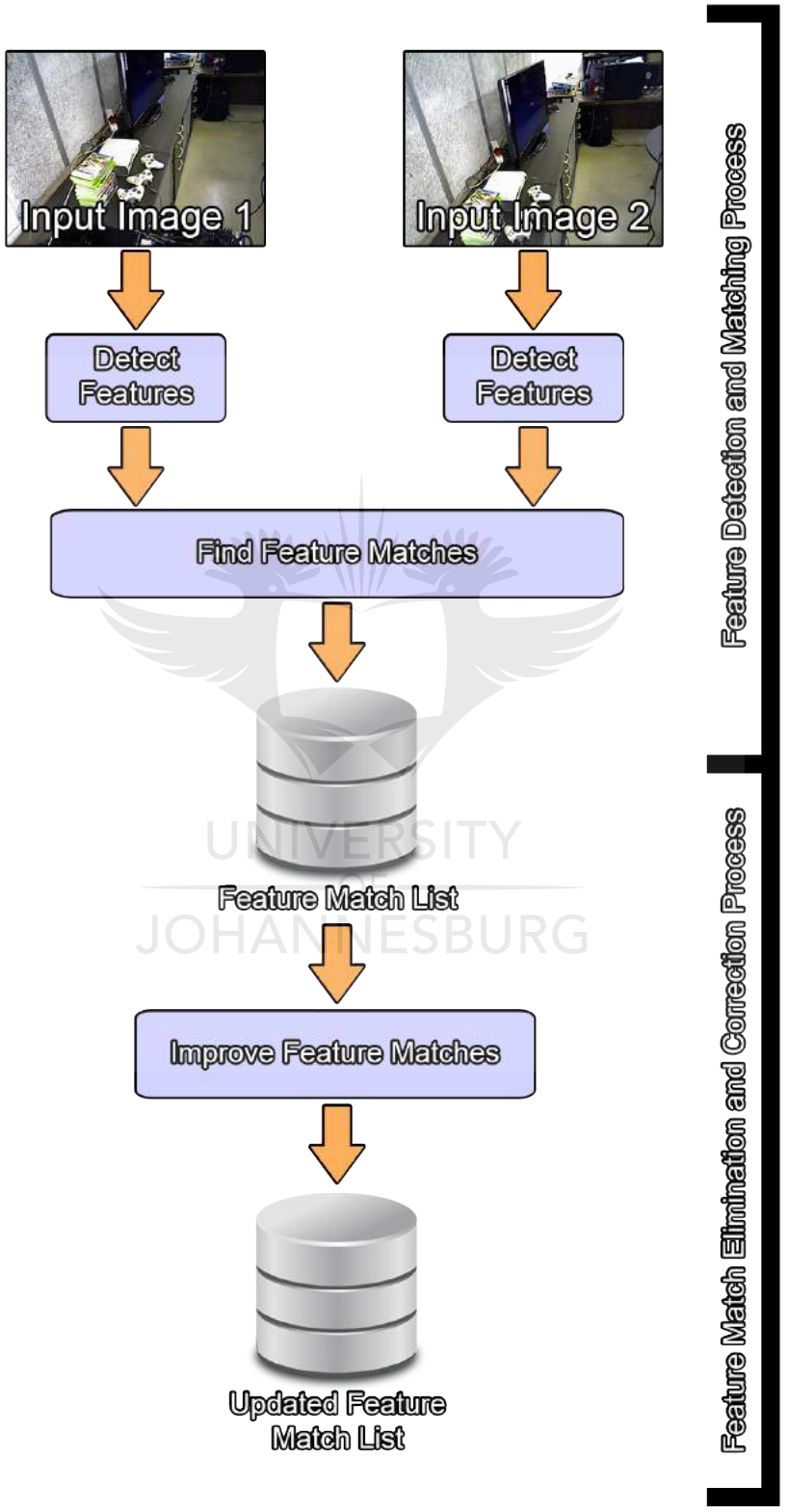


Figure 2.1: Overview of the feature detection and matching process used by many computer vision applications

feature description information is then used to find the best match for each feature. This will result in a match list that provides the feature matching information from one feature set to another. This matching information can then be used to do image processing and computer vision tasks such is alignment, pose estimation or classification. Unfortunately, a large number of incorrect or invalid matches can be present in the match list after the initial feature detection and matching process. These mismatches are usually associated with, or the results of:

- The presence of duplicate features
- Occlusion produced by motion [34]
- Features located on motion boundaries that are poorly localized and have weak descriptions
- Features experiencing extreme scale and affine transformations
- Image noise [35] and compression artefacts [36] degrading feature localization and description
- Out-of-focus blurring and motion blur [37]
- Features affected by dynamic shadows, illumination changes and atmospheric effects [38]
- Insufficient feature descriptiveness
- Poor feature localization repeatability

The addition of a feature match elimination and correction process is employed by most feature detectors to reduce or minimize the effect of the feature match outliers present in the initial feature matches, this extension can be seen in Figure 2.1. To reduce the effect of the invalid feature matches, many algorithms discard features and reduce the match list to only contain the strongest and most stable features.

The two most common techniques used are thresholding of the feature responses and the thresholding of the feature descriptor match error. Thresholding

of the feature response, discards weak features according to the feature's corner, DoG (Difference-of-Gaussians) or LoG (Laplacian-of-Gaussian) response determined by the feature detector. This allows features, which the feature detector believes it will have difficulty localizing in other images, to be removed from the feature lists. Thresholding of the feature descriptor match error, removes poor feature matches from the match list where a matching feature could not be found that was sufficiently similar.

Feature detectors such as SIFT attempt to remove duplicate features from the match list, which it will have difficulty matching reliably [18]. Duplicate features tend to have similar feature descriptions that make them hard to distinguish from each other. These duplicate features will result in invalid matches even if they have high feature responses and descriptor match scores. To be able to determine if there are any duplicate features present in the matching list, the difference between the best feature match error and the second best feature match error is used. If this difference is small, it can be a good indicator that a duplicate feature might exist, and if this difference is large it shows that the feature is unique and that there is a good chance that it will provide a reliable feature match.

Some applications that can constrain the environment and camera, or are able to describe the model of the observed motion, can use the additional motion information to limit the feature matching search space which will improve the matching accuracy [75]. This is typically limited to static scenes and images captured where only camera motion is experienced. When the motion of objects in a scene dominates the image area, the algorithms that estimate the motion model tend to fail, producing poor feature matching results. Tracking of visual features between consecutive frames of a continuous video sequence can also help to limit the feature matching search space [40]. Improving matching through feature tracking can only be used by some applications where continuous video sequences are available and the motion is consistent and predictable.

2.2 Feature detection development

There are three primary types of features that are used in image processing: Edges, Corners and Blobs. Edge features are considered the simplest of the

three feature types and were the first features investigated by computer vision researchers. Marr and Hildreth proposed in 1979 an edge detector based on the early processing stages of the human visual system [41]. Edges were detected in an image by searching for zero-crossings after filtering with different sized Laplacian-of-Gaussian (LoG) kernels. The proposed edge detector suffered from poor localization of curved edges and sometimes detected false or invalid edges.

In 1983 Canny improved on this work by providing a well formulated method for determining edges in images [42]. His method was based on first-order derivatives where he localized the gradient magnitude local maximums of a Gaussian smoothed image. Gaussian filtering was applied as a pre-processing step to reduce the effect of noise on the edge detection process. Non-maximum suppression and hysteresis processes were also applied to remove unwanted edge responses and improve the localization of edges.

Moravec proposed one of the first corner detection algorithms and introduced the concept of "interest points" [43]. He defined a corner as a point in an image with low self similarity. An interest measure was calculated using directional variance over small image regions. The local-maximum of these interest measures were then classified as "interest points". He developed this technique to detect and locate the same features in different images. This method provided poor detection results when corners were at specific angles due to the directional variance calculation being performed only at predefined directions.

A rotation invariant corner detector was proposed by Forstner, which used the ratio between the trace and determinant of the covariance matrix to calculate the local corner response [44]. Pixels with the strongest curvature response in a local region were classified as corners. Forstner also introduced a formal approach to matching sets of interest points between different images using a similarity measure.

A combined corner and edge detector was proposed by Harris and Stephens which is a contour curvature method similar to the corner detector proposed by Forstner [45]. Classification between flat regions, edges and corners could be made using the proposed method. A Hessian matrix or structure tensor was used to measure curvature, where areas of large curvature along a contour were classified as corners and areas of low curvature were classified as edges. The Harris corner

detector is sensitive to scale changes and produced poor detection repeatability in these situations [18]. Multi-scale Harris corner detectors were later introduced to improve the detection rate when scale changes were present [46].

In 1994, Shi and Tomasi extended on the work of Harris and Stephens, and showed how a structure tensor based corner detector can be improved by introducing small modifications to the way that the corner response is calculated [47]. Instead of calculating the trace and determinant of the covariance matrix they calculated and used the minimum eigenvalue as a corner response.

The Smallest Univalue Segment Assimilating Nucleus (SUSAN) feature detector was introduced in 1997. It places a circular mask over every local image region, from which it calculates the size, centroid and second moment [48]. The benefits of this method are that it provides some invariance to noise and that no noise reduction pre-processing steps need to be performed. Other methods typically require pre-smoothing of the image with a Gaussian filter to reduce the impact of noise. SUSAN is also unique compared to other corner detectors in the sense that it does not require the calculation of image derivatives. The SUSAN operator was later improved by introducing an iterative adaptive threshold selection method to calculate an optimal similarity threshold [49]. A ring shaped mask was also introduced to allow the SUSAN operator to detect special and complex corner shapes that its first implementation had difficulty detecting.

High-speed feature detectors based on an Accelerated Segment Test (AST) were recently proposed by Rosten and Drummond which share many similarities with the SUSAN corner detector [21]. The circular mask used by SUSAN was reduced to the border pixels of a circle, thus reducing the amount of pixels that need to be considered when classifying a corner. The border pixels around a candidate corner are evaluated to determine the number of contiguous pixels that occur on the circle. This process is accelerated even further by using machine learning to construct an optimized decision tree. The decision tree is used to stop the processing of potential candidate regions as soon as the likelihood of it being a feature becomes too low. A number of feature detectors based on the AST detector have been proposed, such as FAST [21], FAST-ER [50] and AGAST [51]. These feature detectors differ primarily in how the decision tree is constructed and used. The AST range of detectors share the same goal of low processing

requirements and real-time execution speeds.

Scale invariant corner detectors were introduced in the late Nineties. By repeating the corner detection process on multiple scales of an image, different sized corners could be detected [52]. A drawback of this approach is that the same corner can be detected at multiple scales. The determinant of the Hessian (HoG) was introduced to improve this problem; corners are localized by detecting scale-space maxima of the Hessian corner responses [53]. In 1998 Lindeberg extended this concept by proposing a systematic methodology for selecting appropriate scales for features by observing how normalized Gaussian derivatives change from one scale-space level to another [54].

Instead of detecting corners as features, some researcher experimented with Blob detectors based on Laplacian of Gaussian (LoG) and Difference of Gaussians (DoG) operators. These feature detectors are naturally suited to scale invariant features but do not provide features that are as stable, compared to features detected using corner detectors since these operators respond to corners as well as edges. A method for searching for feature response maximums in the 3D representation or scale-space of an image was originally developed by Crowley and Parker in 1984 [55]. They demonstrated how local maxima in a scale-space representation of bandpass images can be detected and used as landmarks to align and match detected object shapes.

Lowe improved on the work by Crowley and Parker in 1999 by introducing a feature detector invariant to scale, translation, rotation and partially invariant to affine transformations and illumination changes [5]. The Scale Invariant Feature Transform (SIFT) allowed accurate matching between features from different images and demonstrated similar properties to assemblies of neurons used for object recognition in the inferior temporal cortex of primates. SIFT detects features using the DoG blob detector at successive scales and the eigenvalues of the second-order Hessian matrix is used to remove edge responses. One of the primary contributions of SIFT was the introduction of a local histogram of gradients descriptor calculated from the image region surrounding a feature. This helped to reliably identify a feature in another image and simplified the feature matching process. Lowe further improved on the method by adding sub-pixel and sub-scale-level localization for detected features [18].

A number of improvements to SIFT have been proposed, the majority of which made changes to the way that the features are described. Ke and Sukthankar modified the descriptor proposed by Lowe by adjusting the histogram weights using Principal Component Analysis (PCA) [56]. PCA-SIFT produced a more compact local feature descriptor which is more resilient to image deformations and also improves on the computational performance. An updated descriptor with color invariant characteristics based on the color invariance model of Geusebroek et al. [57] was proposed by Abdel-Hakim and Farag [58]. The CSIFT descriptor provided a more descriptive feature description with improved reliability against photometrical distortions. The Pyramid Histogram Of visual Words (PHOW) descriptor was proposed by Bosch et al. in 2007, this descriptor combines appearance and shape information [59]. The PHOW feature description is generated over multiple image scales using the Hue, Saturation and Value (HSV) colourspace. Affine-SIFT (ASIFT) was introduced in 2009 and simulates the affine distortion in an image by varying the camera's longitude and latitude angle over its optical axis [60]. SIFT is then applied to each of the transformed images creating a fully affine invariant feature detector. This can be an expensive process, since it requires the re-detection of features for every simulated image transformation. The Local Intensity Order Pattern (LIOP) feature descriptor was introduced which explores the local intensity relationships of the sample points. Local and global ordinal information surrounding a feature is captured using this method [61]. It was shown that this type of descriptor provided additional invariance to monotonic intensity changes, JPEG compression artifacts and blurring.

Some alternative methods to SIFT have also been proposed. The Noise Invariant Feature Transform (NIFT) was proposed by Roodt et al., the proposed a new feature detector that improved feature detection reliability in the presence of large levels of image noise [62]. The NIFT algorithm made use of an image noise estimation algorithm to determine the level of noise present in the original input image. This initial noise estimate was then used to estimate the amount of noise present in the Gaussian and Difference-of-Gaussian (DoG) scale-space levels that is used to detect features. The noise estimates for each scale-space level was then used to adaptively change the thresholding parameters for that level to allow only strong features to be detected as local extrema that were weaker than the

detected noise could then be discarded. This allowed the noise invariant ability of a feature detector to be improved.

Speeded-Up Robust Features (SURF) was proposed as an improvement over SIFT in 2006, where integral images and filter kernel approximations were used to speed up the feature detection process [19]. An integer approximation to the HoG operator, accelerated with integral images, were used to detect features, and Haar wavelets were used to build the feature descriptors. Similar accuracy and robustness were promised at a reduced computational complexity.

Matas et al. proposed a feature detector based on Maximally Stable Extremal Regions (MSER). This method has many similarities to watershed based segmentation [20]. A connected set of pixels are classified as a maximally stable region when they are brighter or darker then the pixels surrounding the group. MSER provides stable detection results for scenes with viewpoint changes, scale changes and lighting changes but provides poor detection repeatability when blurring is introduced.

2.3 Feature description progress

Lately, a large number of vision researchers focused their efforts on attempting to improve the feature matching process by altering the way image features are described. They released methods and techniques that do not specify how features should be detected, but rather how compact, distinct and repeatable feature descriptions can be generated. Binary Robust Independent Elementary Features (BRIEF), as proposed by Calonder et al., showed how a highly discriminant binary feature descriptor can be constructed using intensity difference tests of constant random sample locations [22]. The features were stored as binary strings that could be compared efficiently using Hamming distance. BRIEF did not provide any scale or rotation invariance. Rublee et al. provided rotation invariance, improved noise invariance and combined this with the FAST detector to create the Oriented FAST and Rotated BRIEF(ORB) feature detector [23]. Binary Robust Invariant Scalable Keypoints (BRISK) proposed by Leutenegger et al. incorporated scale-invariance by calculating a FAST-based detector on different layers of an image pyramid [24]. The feature descriptor used by BRISK

resembles the uniform sampling pattern proposed by Tola et al. in the DAISY descriptor, where pairwise comparisons are performed on the samples to construct the descriptor string [63]. Binary Features from Robust Orientation Segment Tests (BFROST) provided by Cronje, improved on the work of Calonder et al. and Rublee et al. It implemented a FAST-like detector accelerated by using a Graphics Processing Unit (GPU), where the feature descriptor was generated by sampling and comparing the accumulated intensity values of rectangular regions calculated using an integral image [64]. A fast, compact and robust feature descriptor inspired by the retina of the human visual system was introduced by Alahi et al. in 2012 [65]. The Fast Retina Keypoint (FREAK) used a sampling pattern with varying sized receptive fields similar to the ganglion cell distribution in the eye to measure accumulated intensity differences.

2.4 Feature matching progress

Even though much effort has been focused on improving the quality of feature descriptors, feature detectors still have difficulty ensuring that feature matching takes place correctly. When we consider that feature descriptions are generated for a small region surrounding a feature's position, we realize that the description cannot be unique and a number of similar features can occur in the same image. A number of feature matching algorithms and methods based on appearance similarity and geometric consistency have been proposed in an attempt to improve matching results. The majority of feature detectors we have discussed up to this point, ignore spatial layout and rely only on appearance similarity obtained by matching descriptor information. Geometric consistency can be constrained locally as well as globally. Local geometric consistency considers that neighboring features have undergone similar transformations that can be modeled where a global geometric consistency considers that all features in an image have been displaced by a consistent transformation that can be modeled such as the transformation produced by a camera moving through a static scene.

GSIFT, proposed by Mortensen et al., allowed features that were not unique, to be matched by including a global context into the local image feature descriptions [66]. This reduces the number of mismatches of similar features and helps

to resolve local appearance ambiguities. The global context descriptor, however, is not as effective when scale changes are introduced and scale invariance was marked as a potential future improvement. Kai et al. also investigated how the feature matching accuracy of SIFT can be improved by employing a maximum of minimum distance clustering approach [67]. This algorithm is able to cluster related features, which enables it to remove noisy feature matches as well as feature matches that are structurally unrelated.

A number of voting techniques based on Random Sample Consensus (RANSAC) and its variations have been proposed [68]. It enables the estimation of the model parameters produced by the camera's pose changes from the feature correspondence information, even in the presence of large numbers of outliers [69]. Every feature match is then evaluated against the estimated transformation model to determine if it is correct. Bazin et al. extended this idea by formulating the feature matching problem for an unknown parametric model as a mixed integer program that can be solved using linear programs that use the branch-and-bound procedure [70]. Geometric and appearance constraints are combined by globally optimizing a RANSAC-like criterion to simultaneously estimate a global geometric transformation, feature correspondences and classifying potential outliers.

Graph matching optimization techniques have also been proposed. Torresani et al. formulated the matching process as an energy minimization problem of the local appearance and the spatial arrangement of features [71]. This method uses a sub-gradient ascent scheme which suffers from a slow convergence rate, therefore they proposed alternate optimization techniques as a possible direction for future work. An efficient algorithm based on linear programming techniques was introduced by Li et al. Each feature point is represented using an affine combination of its neighboring features where the graph weights are solved using a least squares minimization process [73]. Their method did not specify if feature matches were correct, but was rather used for object recognition using multiple grouped features. This method also had difficulty handling occlusions.

Grauman and Darrell proposed an efficient method of comparing images using an approximation of the Earth Mover's Distance between bags of features present in images [74]. The discrete feature distributions or bags of features are derived from the distinct local invariant features as well as their number of occurrences in

an image, this is achieved without clustering of the features. This algorithm does not allow feature matching to be improved, since the geometric relationships between features are discarded, but it provides an efficient method of finding images with similar feature distributions.

A new technique of detecting and describing features through interest point grouping was proposed by Brown and Lowe [75]. Every possible combination of small groups of interest points is grouped to form features, this results in a very large number of detected features of which only a few are useful. Feature outlier rejection thus forms an important part of this algorithm. Consistent sets of features are found using a Hough transform in a 2-dimensional transformation space. Addition rejection steps are also employed, which make use of the fundamental matrix and epipolar geometry to reject outliers. Many of the original detected features are discarded and the functionality to correct invalidly matched features is not provided by this feature detection method.

An iterative robust feature matching algorithm based on an Alternate Hough and an Inverted Hough transform was proposed by Chen et al. in 2013 [72]. They believe that features located close to each other share coherent homographies that can be used to determine the correctness of matched features. The proposed matching algorithm can only be used with affine invariant feature detectors that are able to detect features that have local affine transformation information and as such, this excludes a large number of the discussed feature detectors, which limits its usefulness.

2.5 Conclusion

An overview of the feature detection and matching process as well as the factors that are responsible for possible feature mismatches, were discussed. A summary of the methods and techniques used to improve feature matching in these situations were also provided, including an extensive overview of the development progress of feature detection, description and matching.

The Super-feature algorithm will allow features to be matched using visual similarity as well as geometric consistency. A similar belief, proposed by Chen et al., where features located close to each other might share coherent transforma-

tions is used by the Super-feature algorithm. The Super-feature algorithm uses a new technique to derive the geometric consistencies that is less constrictive and can be used with any feature detector that provides position, rotation and description information whereas the method proposed by Chen et al. can only be used with affine invariant feature detectors that are expensive to calculate.

The Super-feature algorithm also make use of the Mean-shift algorithm to find the strongest modes in a 2-dimensional probability distribution derived for each feature. In Chapter 3, the application areas and a mathematical description of the Mean-shift algorithm will be provided as well as a visual demonstration of the kernel motion produced by the Mean-shift iterations for a 2-dimensional problem.

Chapter 3

Mean-shift clustering and mode estimation

The following theory will be required to combine the estimation probabilities obtained from the geometric relationships provided by the neighbouring feature matches in the Super-feature cluster, allowing a selected primary feature to be located in a different coordinate frame. A detailed description and breakdown of how the Mean-shift method was used in the Super-feature algorithm will be provided in Chapter 4.

The Mean-shift algorithm is an iterative mode-seeking technique used to estimate the locations of the local maxima or modes of the underling density distribution [76]. It is a non-parametric estimation technique that uses the neighbouring feature space samples to calculate the gradient of the density distribution, allowing it to iteratively ascend to the highest local maximum. Mean-shift is usually used as a clustering technique allowing arbitrary distributed points to be grouped together in the feature space [77]. Typically samples that have converged to the same local maxima are grouped together to form a cluster.

Alternatively, the Mean-shift algorithm is also useful for estimating the global maxima of the probability density distribution of a feature space. A search and comparison is performed to the detected local maxima, allowing the largest maxima to be found. Estimation of the local and global maximums of the density distribution is obtained without physically discretizing and calculating the prob-

ability density distribution from the sample points using a predefined selected filter kernel. Discretizing and sampling the probability density distribution can be an expensive process when a large number of samples are available, especially when feature spaces with higher dimensions are considered.

Clustering of data as well as the estimation of the density distribution modes is an important problem in the fields of machine learning [78] and computer vision [79]. The Mean-shift algorithm has been applied successfully to a number of problems such as:

- Deformable model fitting and alignment [80]
- Classification through clustering [81]
- Image and colour segmentation [82; 83]
- Object tracking [84; 85]

The Mean-shift algorithm does not require knowledge of the number of modes of the density distribution, such as the k-means clustering algorithm, which can be difficult to know and must sometimes be found through experimentation. Only the kernel bandwidth needs to be selected; either a fixed bandwidth can be selected for the Mean-shift algorithm or a dynamic bandwidth parameter can be determined for each sample by deriving the bandwidth from the k-nearest neighbours.

3.1 Deriving the Mean-shift algorithm

Kernel density estimation, also known as the Parzen window density estimation technique, is a method of estimating the probability density distribution from a set of arbitrary distributed points.

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\Big(\frac{x - x_1}{h}\Big) \tag{3.1}$$

Given a set of d-dimensional points defined as $x_{i=1..n}$ where n is the number of points, sampled from an unknown density distribution f. Then the multivariate

kernel density estimator $\hat{f}$ for these points are defined in Equation 3.1 for a filtering kernel K, assuming a single bandwidth parameter h.

$$\begin{aligned}
\text{Gaussian: } K(x) &= e^{-\frac{x^2}{2\sigma^2}} \\
\text{Rectangular: } K(x) &= \begin{cases} 1, & \text{if } a \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \\
\text{Student's t: } K(x) &= \left(1 + \frac{x}{\alpha}\right)^{-\frac{\alpha+D}{2}} \\
\text{Epanechnikov: } K(x) &= \begin{cases} \frac{3}{4}(1-x^2), & \text{if } |x| \leq 1 \\ 0, & \text{otherwise} \end{cases}
\end{aligned} \tag{3.2}$$

The kernel $K(x)$ is a d-variate kernel, which should be non-negative, non-increasing and piecewise continuous [86]. A number of kernels with different characteristics have been proposed such as the Gaussian, Rectangular, Student's t and Epanechnikov kernels, provided in Equation 3.2. The Epanechnikov kernel is considered as an optimum kernel for most applications, as it minimizes the mean-integrated-square-error (MISE) [87].

$$\begin{aligned}
K' &= dK/dt \\
\hat{f}_{h,K}(x) &= \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} K' \left\| \frac{x - x_1}{h} \right\|^2
\end{aligned} \tag{3.3}$$

$$\hat{\bigtriangledown} f_{h,K}(x) \equiv \bigtriangledown \hat{f}_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x - x_i) K' \left\| \frac{x - x_1}{h} \right\|^2 \tag{3.4}$$

By employing the kernel profile notation, the density estimator can be rewritten as Equation 3.3 [88]. The next step in trying to analyse the feature space and the probability density distribution of the feature space would be to estimate the modes of the distribution. The modes of the underlying density distribution will be located where the gradient approaches 0 such that $\bigtriangledown f(x) = 0$. The Mean-shift

algorithm provides an intuitive manner of locating these regions where the gradient approaches 0 without physically calculating the density distribution. The density distribution is typically calculated by discretizing the domain and sampling the density distribution to be able to find the modes. This results in the density gradient estimator provided in Equation 3.4 derived from the gradient of the density estimator in Equation 3.3.

$$\begin{aligned} G(x) &= -K'(x) \\ m_{h,G}(x) &= \frac{\sum_{i=1}^{n} x_i G\left\|\frac{x-x_i}{h}\right\|^2}{\sum_{i=1}^{n} G\left\|\frac{x-x_i}{h}\right\|^2} - x \end{aligned} \tag{3.5}$$

where:

- x is the center of the kernel window
- h is the selected bandwidth parameter

When we consider a radially symmetric kernel that satisfies the conditions previously specified, then the resulting Mean-shift is defined by Equation 3.5 for a set of data points [89]. This method is guaranteed to converge to zero gradient regions. The step size will be large in regions with low density values and small in regions with high densities, resulting in an adaptive gradient ascent algorithm.

$$y_{j+1} = \frac{\sum_{i=1}^{n} x_i G\left\|\frac{y_j-x_i}{h}\right\|^2}{\sum_{i=1}^{n} G\left\|\frac{y_j-x_i}{h}\right\|^2}, \qquad j = 1, 2, ... \tag{3.6}$$

where:

- y_j is the current position of the kernel
- y_{j+1} is the updated position
- j represents the iteration number

The bandwidth parameter h changes the size of the kernel, allowing modes from different scale resolutions to be detected, depending on the application. The final step is to apply the Mean-shift procedure iteratively by computing the

Mean-shift vector $m_{h,G}(x)$ and then translating the position of kernel G by the Mean-shift vector as demonstrated in Equation 3.6. This process is repeated until the system converges and the kernel is not translated any more by further iterations [90]. Depending on the application, either the kernels that have settled on the same mode can either be grouped together to form a cluster, or the unique modes can be found from the grouped kernels. Once the unique modes have been found, the mode with the largest density response can be located as the global maximum.

3.2 Demonstration of Mean-shift iterations in 2-dimensions

A demonstration of the results obtained by running the Mean-shift algorithm a number of times on a 2-dimensional problem can be seen in Figure 3.1. This example shows how a single kernel starting on one of the initial sample positions slowly moves towards the local extrema. At every iteration it calculates the gradient of the probability distribution at its current location by accumulating and weighting the neighbouring samples using its kernel function. The gradient allows it to calculate a shift position that will move the current kernel toward the mode or local extrema. The gradient vector enables it to slowly ascend on the probability density function until it has reached the highest position. This region is where the gradient approaches zero as well as where the local extrema is located. Once a zero gradient has been found, the kernel will cease to move, allowing the algorithm to stop performing any further iteration. Typically, a kernel is created for all of the initial samples. This process will then be performed until all the kernels have converged to their respective local maxima of the probability density function.

3.3 Conclusion

The application areas and uses of the Mean-shift algorithm has been provided, including the mathematical derivation and demonstration of the Mean-shift it-

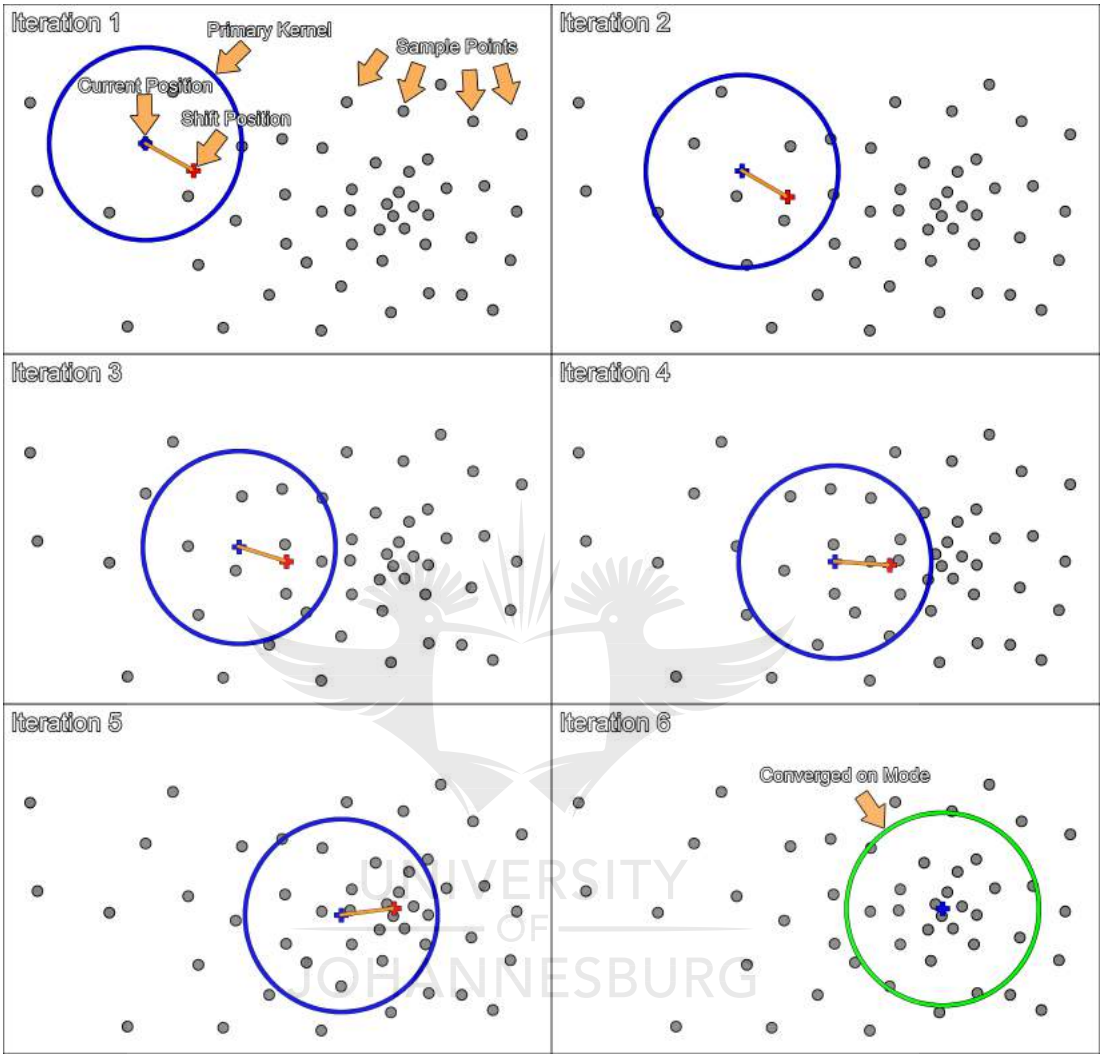


Figure 3.1: Example of kernel shifts produced by the Mean-shift algorithm for different iterations

erations for a 2-dimensional problem. In the next chapter, the Super-feature matching framework will be introduced, which will include a detailed description of the implementation details for each of the algorithm processing steps.

Chapter 4

The Super-feature matching framework

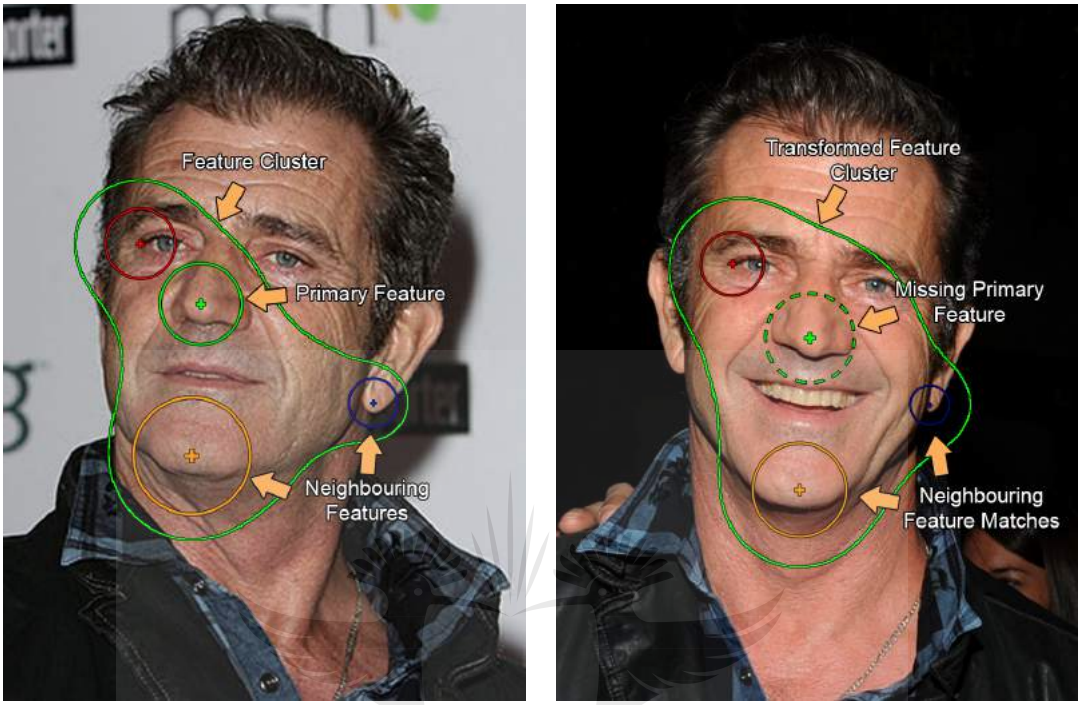
The following is a description of the implementation details of the Super-feature matching and improvement algorithm. The Super-feature algorithm is a new technique, which can be used to group neighbouring features into feature clusters, allowing feature matching to be improved. A method of observing the geometric relationships between different features in a feature cluster in a translation, scale, rotation and affine invariant manner will be provided. After which, a robust and efficient technique of combining these geometric relationships to estimate the transformed location of a selected feature from the feature cluster, located in another coordinate frame in the presence of match outliers will be presented. Finally, an iterative approach will be introduced to detect and correct invalid feature matches by comparing the matched position obtained through visual similarity and the estimated positions derived from the geometric relationships observed between the features in the feature cluster. This will allow features to be matched by not only using visual similarity, but also by incorporating geometric consistency, enabling stronger and more reliable matches to be established.

A problem with integrating geometric consistency into the matching process is that it can limit the types of motion and transformations that can be dealt with, if integrated incorrectly. As previously discussed, current solutions limit processing

to either static scenes with global motion or they require affine information for each feature. Another problem is that the matching algorithms that can handle local motion provide poor invariance to rotation, scale and affine transformations. The proposed technique combines the geometric consistency information in such a way that local as well as global motion of non-rigid, dynamic and morph-able objects can be handled with invariance to translation, scale, rotation and affine changes. In situations where the geometric consistency becomes weak, such as when features are located on motion boundaries, the Super-feature algorithm will provide Multi-mode position estimates. This method can also be integrated and used in conjunction with a wide range of current state of the art feature detectors. To understand the Super-feature algorithm an introduction to feature clusters will now be provided.

4.1 Introduction to feature clusters

Each neighbouring feature match surrounding a selected feature can provide some insight into the transformed location of that feature. The geometric relationship between a selected feature's position and a set of neighbouring feature positions remains relatively constant in a 3-dimensional environment when scale, rotation and translation transformations are applied and the shape is transformed back to a set coordinate frame, thus reversing the effect of these transformations. An example of a feature cluster created for a selected primary feature using neighbouring features can be seen in Figure 4.1.a. After the features in the cluster have experienced some 3-dimensional transformation, the resulting feature cluster can be seen in Figure 4.1.b. Even though the features in the feature clusters have been transformed, they can still provide some insight into the location of the missing primary feature. As an example the location of the primary feature can be determined visually by observing that the primary feature was located somewhere near the centre of all the neighbouring features, as seen in Figure 4.1.a, thus the primary feature will probably again be located relatively close to the centre of the three transformed neighbouring feature matches in the feature cluster observed in Figure 4.1.b. This will allow the feature matching search space to be limited to the region determined by the feature cluster. The cluster



(a) Feature cluster frame 1 (b) Feature cluster frame 2

Figure 4.1: A demonstration of the same feature cluster experiencing 3-dimensional transformations

can constrain the search space by observing the positional relationships between the primary feature and its neighbouring features. These insights, provided by the neighbouring feature matches, can then help locate the primary feature in Figure 4.1.b. This is an oversimplified example of a feature cluster, since it will not always be the case that the neighbouring features will be located around the primary feature. When all neighbouring features are located only on one side of the feature, it will have difficulty estimating the position of the primary feature using the demonstration provided. However, this simplified explanation conveys the idea of how a cluster of features can be constructed and the insight that these features can provide into estimating the position of other features in the same feature cluster, allowing the matched and estimated positions to be compared to determine incorrect feature matches. All features might have been matched

and located in the new coordinate frame, but invalid matches might be present, estimating the position of a selected feature from other features in the feature cluster allows some of these invalid matches to be classified and corrected.

4.2 An overview of the Super-feature match improvement framework

The Super-feature algorithm extends on the concept of feature clustering by matching features using appearance similarity as well as using local geometric consistency provided by the neighbouring features to improve the feature matching accuracy and limit the matching search space. A Super-feature or cluster of features are constructed for each feature using a set number of neighbouring feature matches. The insight provided by the neighbouring features on the geometric relationships of the primary feature will be combined in a translation, scale, rotation and affine invariant manner to estimate the location of the selected feature in a transformed coordinate frame. The Super-feature cluster should remain functioning even when only partial data is available, since all features in the cluster will not always be present or visible and some features might contain incorrect matching information. The Super-feature cluster should thus be able to exist in part and does not have to be whole or complete to be useful. After the position has been estimated from the geometric relationships, then the estimated position can be compared against the matched position of the primary feature to determine if a valid match have occurred. The Super-feature algorithm will provide a new matching framework that can be used with many current state-of-the-art feature detectors. Any feature detector that provides feature position, description and rotation information can make use of the Super-feature algorithm by providing the feature and match lists.

An overview of the feature match improvement framework used by the Super-feature algorithm is provided in Figure 4.2; this proposed framework is part of the feature match elimination and correction process techniques. First, the features are detected using the selected feature detector, and then the initial feature matches can be found between these feature sets. The resulting feature lists and

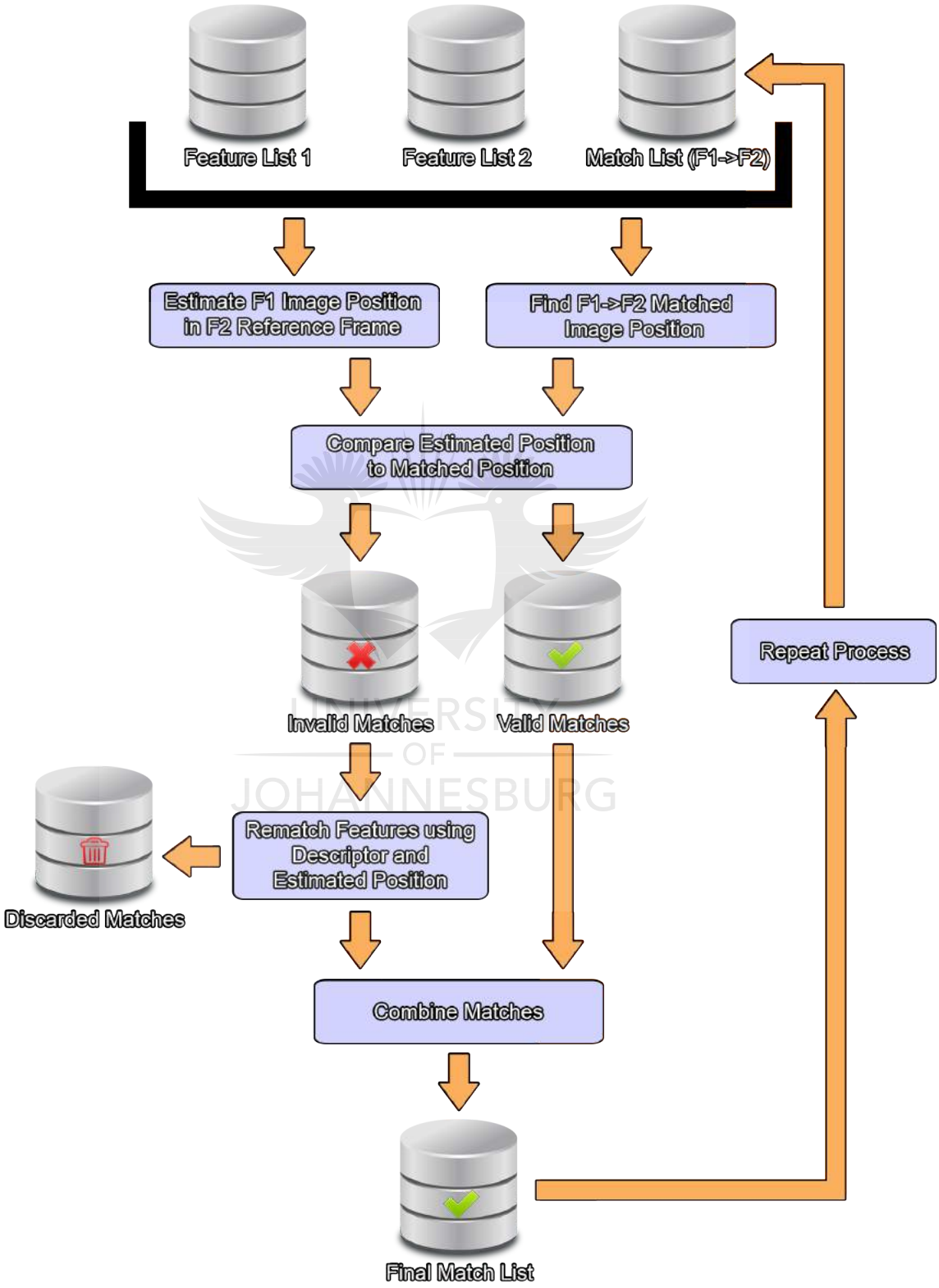


Figure 4.2: Overview of the feature match improvement framework used by the Super-feature algorithm

match list can then be provided as input to the Super-feature algorithm. The Super-feature algorithm will attempt to separate and classify the feature matches into two categories: correct and incorrect matches. The matched position of each feature is extracted and compared against the estimated position derived from the insights provided by the neighbouring features in the Super-feature cluster. When the matched position and the estimated position is similar, the corresponding feature match is marked as a valid match. A disagreement between the matched position and estimated position will result in the feature being marked as invalid.

Each of the feature matches that were classified as being incorrect can then be re-evaluated and the algorithm will attempt to find and improve the invalid matches using the insight provided by the Super-feature cluster. This will reduce the matching search space as the description and estimated position can be used during the matching process. Alternatively, when a true match cannot be found, then additional features can be created at the estimated position of a feature. This can only be done when the estimation reliability is high and the actual feature that should have been matched was not found due to poor feature detection repeatability. This will allow valid feature matches to be created, when the features sets are incomplete and a valid match does not exist. Poor feature sets can occur when low quality feature detectors were used, or when the feature matching problem was difficult due to the image complexity. When a valid match was found that was coherent with the estimated position and the description information, then these corrected feature matches can be included in the valid feature match list which can be used for further processing. Bad matches that could not be corrected are discarded and removed from further processing steps.

The Super-feature algorithm allows two modes of operation. The algorithm can be set to exclude and discard invalid feature matches or can exclude features as well as attempt to correct invalid matches. Depending on the application and processing resources available, the appropriate mode can be selected.

Once the Super-feature match improvement process has been completed, then the entire process can be repeated by using the corrected match list as input rather than the original match list. Mismatched outliers would have been removed from the match list, these outlier could have negatively influenced the estimation process of the previous iteration. In the next iteration, these outliers will no longer

be present, and a more stable estimation solution can be found. By iterating the Super-feature match improvement process a number of times, the system will converge as more and more outliers are removed from the match list and incorrect features matches are corrected, this will result in a more reliable estimation and classification process. Each of the processing steps involved will now be discussed in more detail.

4.3 Initial detection and matching of features

The Super-feature algorithm improves the feature matching and correction process, but does not dictate how features should be detected. This allows the Super-feature algorithm to work in conjunction with a number of feature detectors, as long as these feature detectors provide as a minimum feature position, orientation and description information. The feature orientation provided by the different feature detectors does not necessarily have to provide the exact direction of the corner, it might for instance provide the reverse direction of the corner. These are all valid feature orientations, the only requirement for the feature directions are that they should provide a constant direction for features even though transformations have been applied.

When matching features using the Super-feature algorithm, the first step is to detect features in both frames using the selected feature detector. The initial feature matches should also be established, this can be achieved by performing a brute force nearest-neighbour search using only the description information of each feature. The feature lists for each frame as well as the feature match list can then be sent to the Super-feature algorithm to allow it to improve and correct the feature correspondences.

4.4 An overview of the Super-feature position estimation algorithm

The next step in the Super-feature algorithm, is that each feature of the first coordinate frame's feature list is selected as a primary feature and a Super-feature

cluster will be constructed for each of these primary features. To start building the Super-feature for each primary feature, the k-nearest neighbours of each primary feature need to be found. The k-nearest neighbouring features of the selected primary feature for reference frame 1 is found by performing an index preserving partial selection sort on the distance values obtained by calculating the Euclidean distance between the primary feature's position and its neighbouring feature's position. The geometric relationship between the neighbouring features and the primary feature can then be observed by constructing a voting vector from the position difference. A demonstration of the detected k-nearest neighbour features and the corresponding voting vectors for a selected primary feature can be seen in Figure 4.3. The voting vector's position and rotation are then normalized using

Figure 4.3: An example of the voting vectors derived from the k-nearest neighbours of a selected primary feature

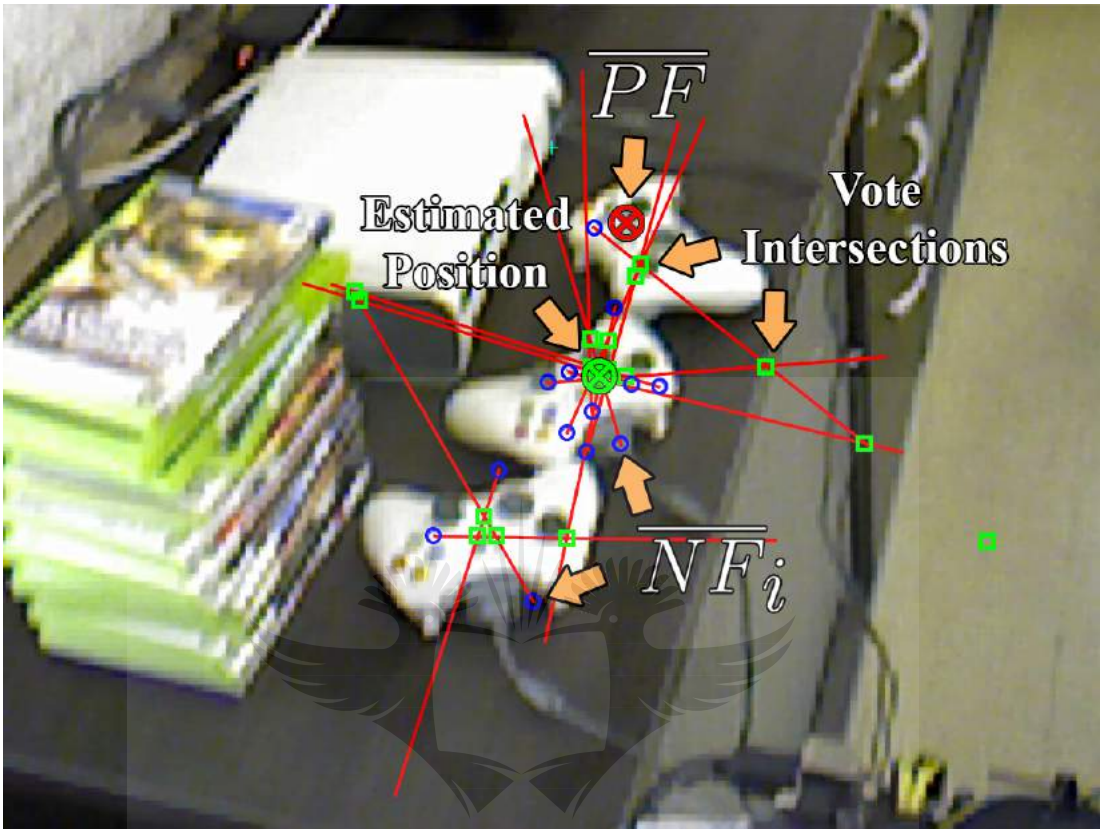


Figure 4.4: An example, demonstrating how the intersection positions of the voting lines were used to estimate the true location of the mismatched primary feature

the neighbouring feature's position and direction, this will allow the voting vector to be transformed and applied to a new coordinate frame. After the voting vectors for each neighbouring feature in reference frame 1 have been constructed, then the voting vectors and their geometric relationships can be applied in reference frame 2. The voting vectors are first, rotated and transformed using the direction and position of the feature in reference frame 2 that was matched to the corresponding neighbouring feature match in reference frame 1.

The next step is to convert each voting vector in reference frame 2 to a voting line starting at the matched neighbouring feature's position and pointing in the direction in which the neighbouring feature believes the primary feature is located. A demonstration of the voting vectors, transformed and applied to the

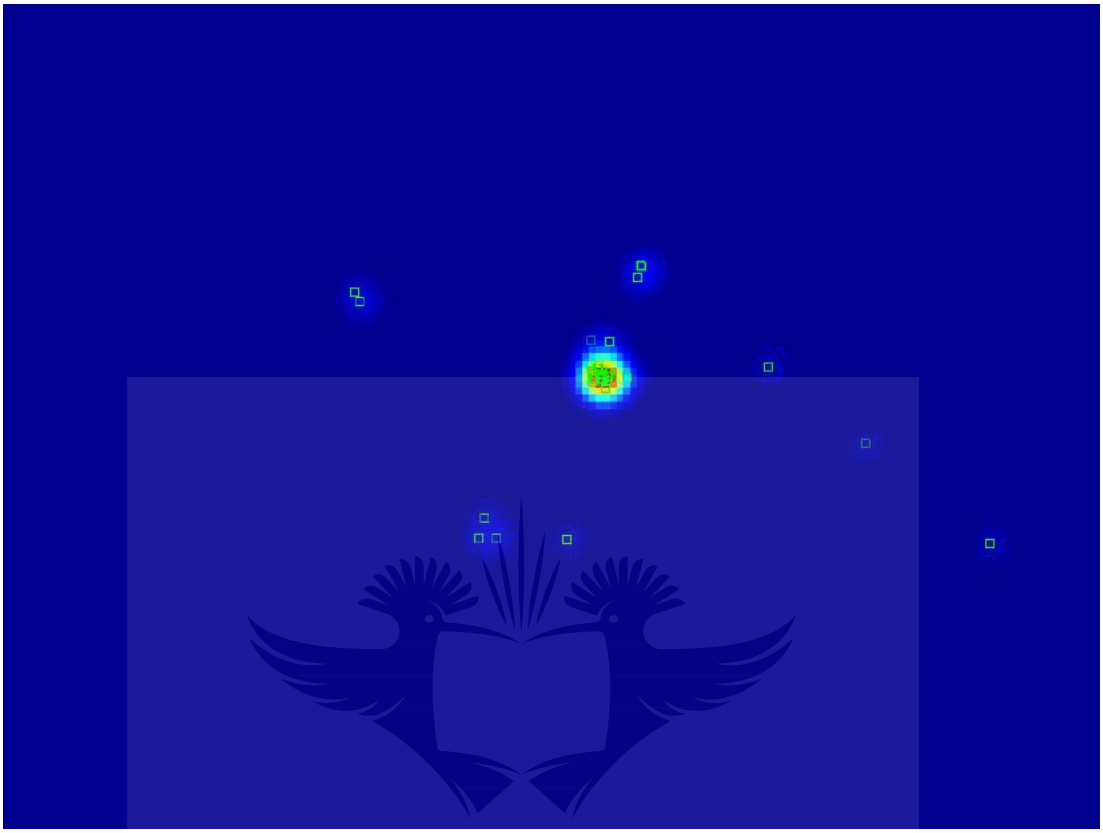


Figure 4.5: The probability density function constructed from the voting line intersections using a Gaussian kernel

neighbouring feature matches for a transformed reference frame 2 can be seen in Figure 4.4. The intersection position between each combination of voting lines can then be determined and used to estimate the position of the primary feature using the geometrical observations of the neighbouring features. Neighbouring features that have been matched incorrectly will generate invalid intersections that do not agree with the true location of the primary feature as seen in Figure 4.4.

Since many outliers can be present, the different intersection estimates need to be combined in an effective and robust manner. A Gaussian kernel with a tune-able bandwidth sigma is placed and accumulated at each intersection position, allowing a probability density function to be constructed representing the

probability of the primary feature's location. The resulting probability density function of the intersections positions can be seen in Figure 4.5. Note how the highest probability response was produced at the true position of the primary feature in the new coordinate frame, and that a low probability response is present at the position in the probability density distribution where the primary feature was incorrectly matched to. Some low probability peaks can be produced by invalid matches, but only the strongest modes are used as possible estimates of the primary feature's location.

Constructing and discretizing the 2-dimensional probability function to find the peaks is a processing resource expensive process, but since only the locations of the local extrema in the probability density distribution are important. A computationally efficient Gaussian weighted Mean-shift algorithm could be employed. The Gaussian weighted Mean-shift algorithm was used to find the locations of the modes of the distribution without constructing the probability density distribution. A list of the strongest modes is found, each representing a possible location for the primary feature in reference frame 2. The final step is to compare the estimated positions to the matched position, allowing the feature matches to be classified as either, valid or invalid.

The primary feature in Figure 4.3 was matched incorrectly in Figure 4.4, but the Super-feature algorithm was able to locate the highest response in the probability density function as seen in Figure 4.5, allowing the mismatched feature to be corrected to the true matching position. Each of the Super-feature position estimation algorithm steps and their implementation information will now be discussed in detail.

4.5 Finding the k-nearest neighbour features and feature matches

The first step is finding the indices in the feature list of the first coordinate frame of the K-nearest neighbours for each of the input features. This can be accomplished, by selecting a feature and then calculating the Euclidean distance from the selected feature's position to all other features in the feature list. A

partial selection sort can then be applied to these distance values to sort them into ascending order up to $K + 1$ elements [91].

To reduce the memory and processing requirements of the algorithm as well as limiting unnecessary memory swaps, the current selected feature is not removed from the feature list before the distance calculations are performed on each element. Therefore, there will always be an element with a distance of 0 that corresponds to the current selected feature itself. This is the reason why the distance values are sorted up to $K + 1$ and not K values, since only the first K-elements with the smallest distances to the primary features need to be found. The first element is discarded and the nearest neighbour index list can then be constructed from the elements of 2 up to $k + 1$, which will result in K elements selected.

```
Input A as list of unsorted values;
Input K as the number of sorted elements;
Set N as the number of elements in A;
Initialize index with [1..N];
for i ← 1 to K do
    for j ← i+1 to N do
        if A[index[j]]<A[index[i]] then
            Swap index[i] and index[j];
        end
    end
end
```

Algorithm 1: Index preserving partial Selection sort

The selected sorting algorithm needs to preserve the indexing information of the neighbouring features associated with the distance values. This will enable and allow access to the feature and match information stored in the data structure lists used by later processing steps. For this reason, the index preserving partial selection sort algorithm was applied to the distance values. It swaps indexing information instead of the physical stored values, the sorting algorithm steps are presented in Algorithm 1. This sorting algorithm allows complex programming

structures to be sorted up to a specified K without unnecessary memory transfer of information stored in complex data structures, reducing processing overhead. It returns a list of index positions that represent the distance values in ascending sorted form.

$$\begin{aligned} feature1\_SubList &= feature1\_List[KNN\_IDs[1..K]] \\ feature2\_SubList &= feature2\_List[match\_List[KNN\_IDs[1..K]]] \end{aligned} \tag{4.1}$$

The next step is to compile the feature sub-lists from the nearest neighbour index list, original feature lists as well as the matching lists as seen in Equation 4.1. Since only the first K-elements with the smallest distances to the primary features need to be found. A compact feature list of the K-nearest features are compiled from the sorted index list. A specific index in $feature1\_SubList$ is related to that same index in $feature2\_SubList$ as a match from the one feature set to the next.

4.6 Calculating the voting vector from the geometric relationships between the neighbouring features and the primary feature

$$\begin{aligned} \theta &= -NF_{rotation} \\ v &= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} (PF_{position} - NF_{position})^T \end{aligned} \tag{4.2}$$

where:

$PF_{position}$	is the position of the selected primary feature
$NF_{position}$	is the position of the neighbouring feature
$NF_{rotation}$	is the detected feature orientation of the neighbouring feature provided by the feature detector

Now that the feature sub lists have been created from the neighbouring features, the geometric relationship between each neighbouring feature and the selected primary feature can be determined. To ensure scale, rotation and affine

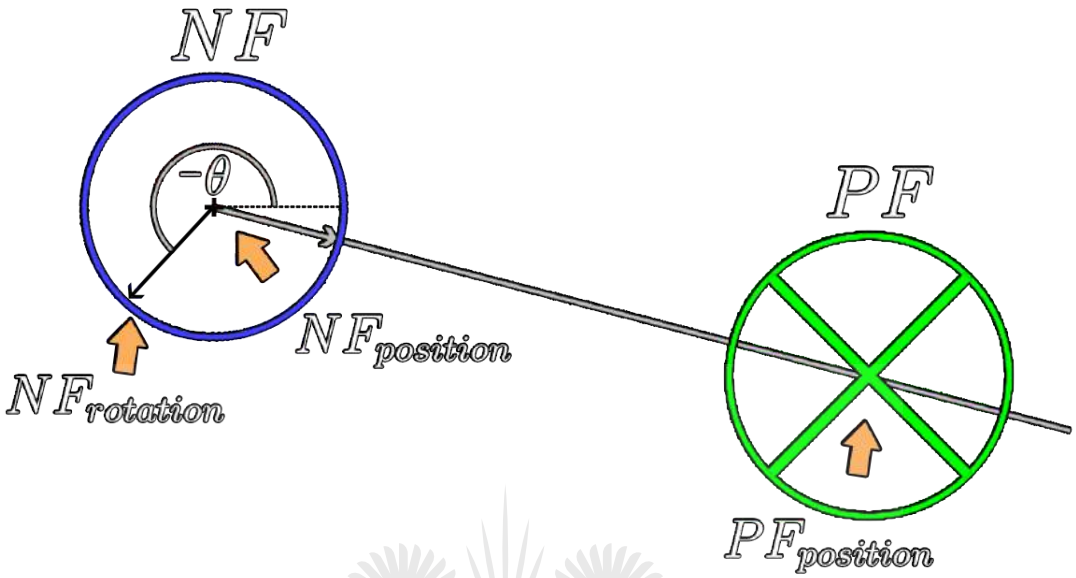


Figure 4.6: Generating a voting vector from the neighbouring feature to the primary feature

invariance, a voting vector is calculated from a neighbouring feature to the primary feature where the rotation is then normalized to a set coordinate frame by transforming the voting vector by the rotation of the neighbouring feature's direction.

For each feature in the neighbouring $feature1\_SubList$, the rotationally normalized voting vector v can be found using Equation 4.2. In the first feature list's coordinate frame, the voting vector can be constructed by subtracting the neighbouring feature's position $NF_{position}$ from the primary feature's position $PF_{position}$. A rotation transformation is then applied to this voting vector to remove the detected direction of the neighbouring feature, resulting in a vector that is located on a normalized coordinate frame. A demonstration of how a voting vector can be constructed from the geometric relationship between a neighbouring and primary feature can be seen in Figure 4.6.

To be able to apply the voting vector to another coordinate frame, the resulting vector needs to be stored after the rotation was normalized using the feature's direction. The inverse of the neighbouring feature's rotation $NF_{rotation}$ is used to

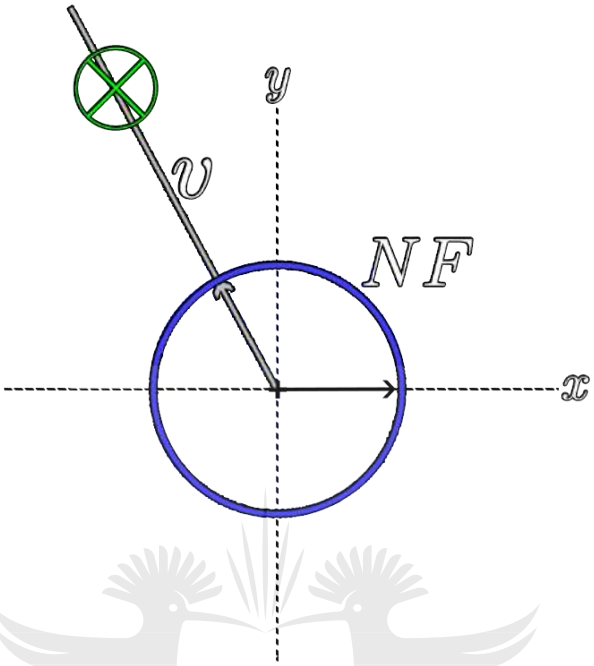


Figure 4.7: Normalization of the rotation of the voting line using the neighbouring feature's rotation

rotate the voting vector so that the neighbouring feature's direction is 0 degrees while preserving the angle between the feature's direction and the determined voting vector, this process can be seen in Figure 4.7. The resulting voting vector v can then be applied to a new coordinate frame when the matching process is initiated on another feature set.

$$\begin{aligned} \beta &= \overline{NF}_{rotation} \\ w &= \begin{bmatrix} cos\beta & -sin\beta \\ sin\beta & cos\beta \end{bmatrix} (v/\left|v\right|)^T \end{aligned} \tag{4.3}$$

where:

$\overline{NF}_{rotation}$	is the direction of the neighbouring feature matched in the new coordinate frame
v	is the voting vector

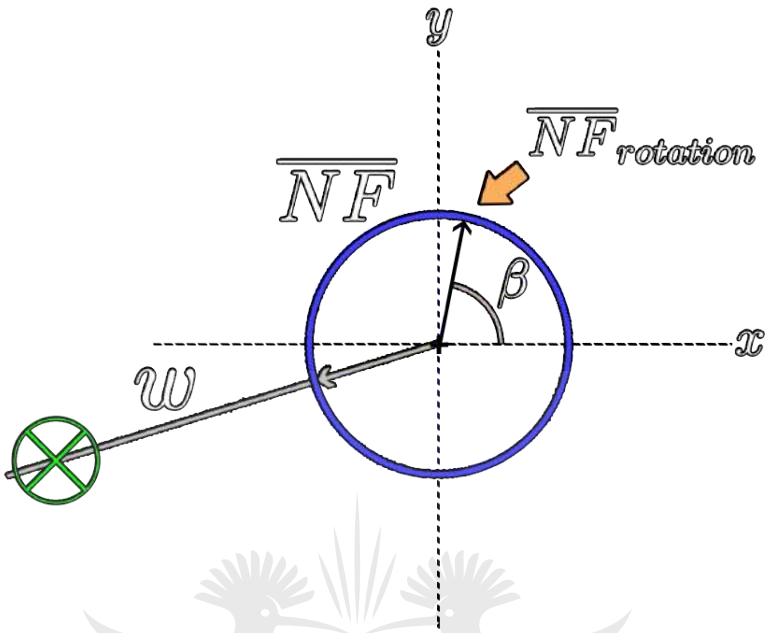


Figure 4.8: Voting line transformed to new coordinate frame

To be able to apply the normalized voting vector v to a new coordinate frame, the voting vector needs to be rotated using the feature orientation of the feature that was matched to the neighbouring feature that was used to construct this voting vector, this will produce a new transformed voting vector w. Transforming the voting line into a new coordinate frame can be calculated using Equation 4.3. The vector w will provide information regarding the direction in which the neighbouring feature believes the primary feature is located.

A demonstration of the voting vector transformed using the direction of the neighbouring feature in a new coordinate frame can be seen in Figure 4.8. Each neighbouring feature can provide some insight into the direction of the primary feature relative to that neighbouring feature. The directional relationship can be detected in one coordinate frame and transferred and applied in a new coordinate frame allowing the primary feature to be located.

4.7 Finding the intersection positions between all voting line combinations

The next step is locating the intersection positions between the different voting vectors, each intersection position provides an estimate of the location of the primary feature in the new coordinate frame. A voting line needs to be constructed from the voting vector to limit the intersections between two voting vectors to only be possible in front of both voting vectors. Currently any intersection between the two infinitely long voting vector will be considered valid.

4.7.1 Calculating the intersection position of two lines

Each voting direction vector has a direction as-well as a position that corresponds to the neighbouring feature's position that was responsible for that specific vector. The voting direction vector can be converted into a line that starts at the vector's position and points in the direction of the direction vector. To be able to construct a line from a vector, two points are defined that lie on this line.

$$\begin{aligned} p1 &= (x_1, y_1) = NF2_{position} \\ p2 &= (x_2, y_2) = p1 + w \end{aligned} \tag{4.4}$$

where:

$NF2_{position}$	is the neighbouring feature's position in frame 2
w	is the voting direction vector
$p1$ and $p2$	are points that lie on the first voting line

The first point is set to be equal to the direction vector's position and the second point can be defined as the vector's position combined with the vector's direction itself, this is demonstrated in Equation 4.4. Similarly, this technique

can be used to define the points $p3$ and $p4$ located on the second voting line L_2.

$$\begin{vmatrix} P_x & P_y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix} = 0 \quad and \quad \begin{vmatrix} P_x & P_y & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{vmatrix} = 0 \tag{4.5}$$

The next step is developing an algorithm to find the intersection position of two selected voting lines. The intersection position $P = (P_x, P_y)$ of two lines L_1 and L_2 can be calculated by simultaneously solving Equation 4.5, where each line was constructed using two distinct points on the voting line such as $p1 = (x_1, y_1)$ and $p2 = (x_2, y_2)$ for L_1 and $p3 = (x_3, u_3)$ and $p4 = (x_4, y_4)$ for L_2 [92].

$$\begin{aligned} P_x &= \begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix} \Big/ \begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix} \\ P_y &= \begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix} \Big/ \begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix} \end{aligned} \tag{4.6}$$

These equations can be combined and solved by calculating the determinants of the points that lie on the individual lines [93], resulting in Equation 4.6.

$$\begin{aligned} (P_x, P_y) = \Bigg(& \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \\ & \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \Bigg) \end{aligned} \tag{4.7}$$

Further decomposition results in Equation 4.7. Unfortunately, this equation assumes infinite lines passing through the selected points, the ideal configuration is that the voting lines should start at the vector position and point in the direction of the vector. Another problem with this solution is that it has difficulty handling parallel lines, the intersection position becomes undefined, but does not provide an appropriate, stable response. There are also a number of duplicate calculations that can be precomputed to reduce the overall processing requirements. To better handle all of these special cases, some changes and improvements were made.

$$
\begin{aligned}
a &= x_1 y_2 - y_1 x_2 \\
b &= x_3 y_4 - y_3 x_4 \\
c &= x_3 - x_4 \\
d &= x_1 - x_2 \\
e &= y_3 - y_4 \\
f &= y_1 - y_2 \\
g &= de - cf \\
P &= (P_x, P_y) = \left(\frac{ac - bd}{g}, \frac{ae - bf}{g} \right) \\
L1\_Proj &= (d, f) \cdot (P_x - x1, P_y - y1) \\
L2\_Proj &= (c, e) \cdot (P_x - x3, P_y - y3) \\
I(p1, p2, p3, p4) &= \begin{cases} 1, & \text{if } g \neq 0 \textit{ and } L1\_Proj < 0 \textit{ and } L2\_Proj < 0 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
\tag{4.8}
$$

where:

$L1\_Proj$ and $L2\_Proj$	are signed distance measures of the projection of the calculated intersection position onto the lines L_1 and L_2, respectively
$I(p1, p2, p3, p4)$	is an intersection flag used to specify if a valid intersection occurred

To ensure that intersections cannot occur behind the voting vectors, a new point is calculated by projecting the intersection position determined using Equation 4.7 onto the two voting lines. This will result in a signed distance measurement that provides the distance of the intersection position on the line from the lines starting position or origin. If this signed distance measurements is positive, then the intersection occurred behind the vector, the intersection is then classified as an invalid intersection. When both signed distance measurements are negative, the intersection is valid and occurred on the two lines starting at their respected origins.

```
Convert vector directions and positions into point p1, p2 on L1 and p3 and
p4 on L2;
Calculate c, d, e, f and g;
if g ≠ 0 then
    Calculate a and b;
    Determine intersection position P;
    Calculate point-to-line projections L1_Proj and L2_Proj;
    if L1_Proj < 0 and L2_Proj < 0 then
        Return valid intersection at point P;
    else
        Return invalid intersection;
    end
else
    Return invalid intersection;
end
```

Algorithm 2: Calculating the intersection position of two lines defined by vectors

In Equation 4.8, the inclusion of the intersection point to line projection comparisons are demonstrated, the pre-computation and reuse of some of the mathematical terms were also introduced. An optimal method of implementing the vector line to vector line intersection function is presented in Algorithm 2. It allows some of the processing steps to be delayed or skipped when parallel lines are

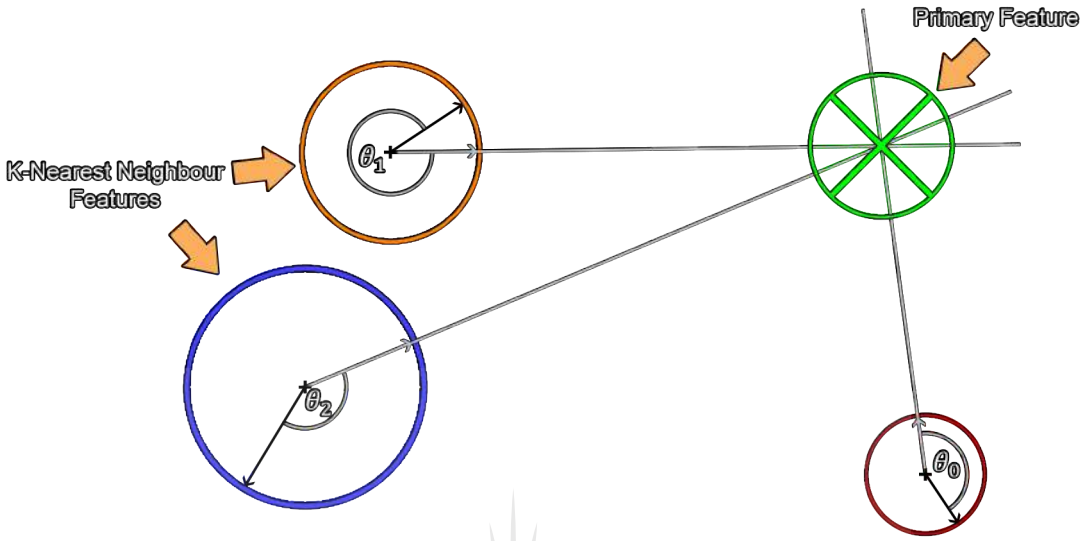


Figure 4.9: Multiple neighbouring features, each observing a single geometric relationship for the primary feature

encountered improving the processing performance. It also limits intersections on infinite lines, producing only intersection in-front of the voting vectors.

4.7.2 Estimating the position of the primary feature using neighbouring feature voting lines

Each neighbouring feature provides insight on the direction in which it believes the primary feature is located. When the directional insights from multiple features are combined it allows the algorithm to estimate the position of the primary feature from the intersections of the individual voting lines. Insights observed from one coordinate frame needs to be transferred to another coordinate frame in a scale, affine, translation and rotation invariant manner.

An example of three different neighbouring features observing a geometric relationship for a selected primary feature in one coordinate frame can be seen in Figure 4.9. Every combination of two neighbouring features provides a single estimate of the position of the primary feature. Thus a number of possible position estimates can be derived that need to be combined to determine the global posi-

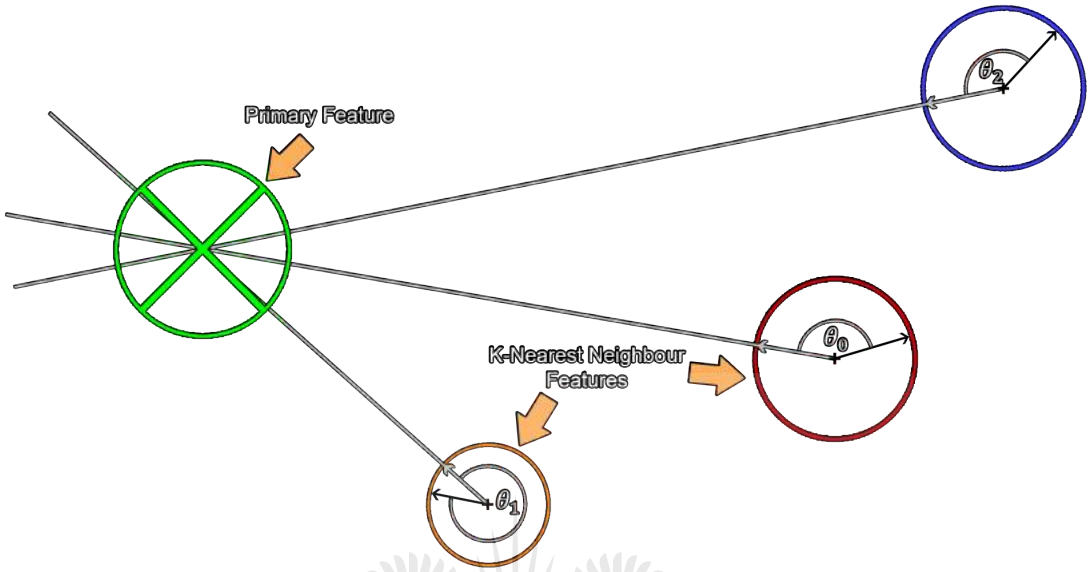


Figure 4.10: Multiple neighbouring features, each applying their geometric relationship in an attempt to locate the primary feature

tion estimate. Some feature matches might have been matched incorrectly, which will result in some of the position estimates to be incorrect when the intersection positions are determined. The algorithm should cater for outliers and incorrect position estimates by fusing the local position estimates in a robust manner to ensure that a reliable position estimate is derived.

When the geometric relationships from multiple neighbouring feature matches derived from one coordinate frame is applied to another, the primary feature can be located. An example of how the intersection positions of the voting lines from multiple neighbouring feature can be used to estimate the location of the primary feature in another coordinate frame can be seen in Figure 4.10. Note that the angle relationship between the feature's direction and the voting angle is preserved to a high degree when translation, scale, rotation and affine transformations are applied between the different coordinate frames as can be observed in Figure 4.9 and Figure 4.10. Handling of affine transformations are the most difficult of the four different transformations, increasing the number of neighbours used by the estimation process will improve the Super-feature algorithms ability to handle this type of transformation.

4.8 Finding the modes of the probability density distribution using a 2-dimensional Gaussian weighted Mean-shift algorithm

Create kernels $K_{i=1..n}$ at each sample point position;
Create copy of initial kernels $\hat{K} = K$;
for *iteration* $\leftarrow$ *1 to max_iterations* **do**
  $convergence\_flag = 1$;
  for $i \leftarrow$ *1 to* n **do**
    $W_{accumilated} = 0$;
    $\overline{K}_{accumilated} = 0$;
    for $j \leftarrow$ *1 to* n **do**
      $d = \sqrt{\sum(K_i - \hat{K}_j)^2}$;
      $W_{gaussian} = e^{\frac{(d/\sigma)^2}{-2}}$;
      $W_{accumilated} = W_{accumilated} + W_{gaussian}$;
      $\overline{K}_{accumilated} = \overline{K}_{accumilated} + (\hat{K}_j * W_{gaussian})$;
    end
    $\overline{K}_i = \frac{\overline{K}_{accumilated}}{W_{accumilated}}$;
    if $\sum |K_i - \overline{K}_i| > \varepsilon$ **then**
      $convergence\_flag = 0$;
    end
    $K_i = \overline{K}_i$;
  end
  if $convergence\_flag = 1$ **then**
    Algorithm has converged, no more iterations are required;
  end
end

Algorithm 3: Gaussian weighted Mean-shift kernel propagation algorithm

The final step in the position estimation algorithm is to combine the different position estimates provided by each voting line pair. A probability density

function of the estimated positions can be constructed through kernel density estimation. A Gaussian kernel is accumulated at each intersection position provided by the insights of the neighbouring feature matches and their corresponding voting lines. This is a computationally expensive process as the resulting probability density function needs to be constructed in 2-dimensions and this process need to be repeated for each of the original feature matches that need to be evaluated.

The Gaussian weighted Mean-Shift algorithm (GWMS) can be used to combine the different position estimates allowing the regions with the strongest probability responses in the probability density function to be located without constructing the probability density function. The local gradients of the probability density function are calculated from the intersection positions and the Gaussian kernel, allowing the kernels of the GWMS algorithm to converge on the modes or local extrema of the distribution.

An overview of the steps involved in the GWMS kernel propagation algorithm is provided in Algorithm 3. The first step is to create kernels at each voting line intersection position. An initial unaltered copy of these kernels will be stored as $\hat{K}$, since the Kernel positions will be updated after each iteration and the original kernel positions are required by some of the algorithm's processing steps. A number of iterations will be performed until the system converges or the maximum number of iterations have been performed. The *max_iterations* parameter was introduced to ensure that the GWMS algorithm always terminates in a timely fashion, the *max_iterations* was set to 1000 for this implementation.

The next step in the GWMS algorithm is to calculate and accumulate the Gaussian weighted positions for all the initial unaltered neighbouring kernels $\hat{K}_j$ compared to the current selected kernel K_i. The Euclidean distance is calculated from the position of the current kernel K_i to the positions of the neighbouring kernels $\hat{K}_j$. A fixed kernel bandwidth σ, as well as the distance is then used to sample the current kernel's Gaussian function to calculate the Gaussian weight $W_{gaussian}$. The normalization coefficient used to ensure that the traditional Gaussian kernel integrates to 1 was removed to save computation time, the normalized kernel that was used in this implementation will have a maximum response of 1. The calculated Gaussian weight will have a stronger weight for kernels that are closer and a smaller weight for kernels that are placed far apart. The different

Gaussian weights calculated for each neighbouring kernel are then accumulated and stored in $W_{accumilated}$.

A new position for the selected kernel K_i needs to be calculated, this is achieved by accumulating the positions of each neighbouring kernel, scaled by the calculated weight for that neighbour. After each neighbouring kernel has been evaluated, the new updated position $\overline{K}_i$ for the selected kernel K_i can then be calculated by dividing the weighted position accumulation $\overline{K}_{accumilated}$ by the accumulated weights $W_{accumilated}$.

Now convergence can be checked, the Manhattan distance is calculated between the previous kernel position K_i and the update kernel position $\overline{K}_i$. If this distance is larger than the convergence bias ε, then the specific kernel has not yet converged and it is still busy ascending toward the local extrema. If the motion distance of all the kernels, between different iterations are less than the convergence bias, then the execution of the Mean-shift iterations can be terminated since a state of convergence has been achieved. As a final step, the update kernel position $\overline{K}_i$ is then set as the kernels new position K_i for the next iteration of the GWMS algorithm.

Initialize all $U_{i=1..n}$ as *true*;
$q = 0$;
for $i \leftarrow$ *1 to n* **do**
  if U_i *is true* **then**
    $q = q + 1$;
    $M_q = K_i$;
    for $j \leftarrow$ *i+1 to n* **do**
      if $\sqrt{\sum(K_i - \hat{K}_j)^2} <= \beta$ **then**
        $U_j = false$;
      end
    end
  end
end

Algorithm 4: Finding the distribution modes from the converged kernels

Now that the kernel propagation algorithm has converged and the intersection kernels have settled on the local extrema. A position list of all the local extrema needs to be constructed by linking the grouped kernels that have settled on the same extrema. An overview of the mode finding algorithm that operates on the converged kernels is provided in Algorithm 4.

A flag U_i is created for each converged kernel K_i, the value of the flag represents, if that kernel is still eligible to be used as a unique mode of the distribution. Since each kernel would have converged on the same position as the modes of the distribution, a number of kernels would have settled on the same local extrema. We want to construct a list of only the unique modes or kernels, thus some kernels that share a local extrema will be discarded. Initially, all kernels are eligible for use as a mode and U_i is initialized as true for every kernel. The first kernel that is found to be eligible is added to the mode list M, its linked cluster partners need to be found and their eligibility need to be removed. The Euclidean distance between the current mode and the rest of the remaining kernels are calculated, if this distance is smaller than the clustering distance threshold β, it means that the kernel was also located on the same mode and its eligibility would be removed by setting U_j equal to $false$. This process will systematically construct the list of unique modes by excluding kernels where the local extrema have already been added to the list.

Only the locations of the strongest modes should be used as position estimates. The probability response of the global maximum is found and used to locate all the local extrema that have similar high responses to the global extrema. Local modes that have probability density responses within 10% of the maximum response were also added to the position estimate list. Therefore, more then one position estimate can be obtained for a feature match, allowing the system to function as a multi-mode estimator.

4.9 Conclusion

The concept of feature clusters were introduced as well as the implementation details of the Super-feature matching and improvement framework. Each of the processing steps involved in the Super-feature algorithm were discussed in detail.

The presented Super-feature algorithm allows neighbouring feature matches to be clustered and the observed geometric relationships between the features in the cluster to be applied to the matching process allowing features to be matched using appearance similarity and geometric consistency. Matching features using appearance similarity and geometric consistency allow more reliable feature matches to be established and the feature matching search space to be reduced. The proposed feature matching framework allows geometric relationships to be obtained and applied in a reliable and robust manner, even when a large number of feature matches in the Super-feature cluster is missing or provide invalid feature match information. Another benefit of the Super-features algorithm is that it was designed to function in a translation, scale, rotation and the affine invariant manner and that it can be used in conjunction with many state-of-the-art feature detectors that provide feature location, rotation and description information.

In Chapter 5 the testing methodology will be presented and experimental tests will be performed to find optimal parameters for the Super-feature algorithm. A number of experiments will also be performed to determine the feature matching reliability and performance results of integrating the Super-feature algorithm to a number of current state-of-the-art feature detectors.

Chapter 5

Results

5.1 Experimental setup

A number of tests were developed and performed to tune and test the capabilities of the Super-feature algorithm and its ability to handle different situations. An overview of the tests that will be performed is provided in Table 5.1. The Super-feature algorithm has a number of parameters that can be tuned. Therefore the first set of tests was performed where the different algorithm parameters were varied over a range of different values, the accuracy results for these configuration settings could then be determined. This allowed us to determine and select a generic set of parameters that will provide reliable results in a wide variety of processing conditions and situations. As a result, these selected parameters were then used for the rest of the testing procedures.

The next set of test that were performed was the transformation invariance tests. These tests allowed us to determine the ability of the Super-feature algorithm to handle Scale, Rotation and Affine transformations, and to what degree the Super-feature algorithm was invariant to these transformations. Individual tests were performed for these simple transformations because it can be difficult to determine where a problem or limitation of the Super-feature algorithm might lie when using a complex or composite transformation test, which might include a combination of different transformations. Consequently, if the Super-feature algorithm is lacking in one of these areas, it could be observed and a potential

Test category	Category sub-tests
1.) Selection of Super-feature algorithm parameters	i.) Gaussian Sigma parameter ii.) Number of neighbours parameter iii.) Number of iterations parameter iv.) Weight threshold parameter
2.) Simple global image motion transformations	i.) Rotation transformation invariance ii.) Scale transformation invariance iii.) Affine transformation invariance
3.) Complex global and local motion transformations	i.) Dominant scene motion with possible occlusions ii.) Motion present on foreground as well as background iii.) Dynamic, non-rigid and morph-able objects and environments iv.) Fast object and camera motion with motion blurring

Table 5.1: Overview of the test categories as well as the sub-tests that will be performed

improvement could be developed.

The last set of tests consisted of composite local as well as global transformations, these tests ranged from simple to complex transformations of the observed scene as well as the objects in the scene. The first tests in this testing category will be simple with dense sets of descriptive features and predictable motion. Each set of tests in this category is increasingly more difficult as the features become less descriptive and the feature sets become more sparse due to the complexity of the observed scene and the motion of the objects in the scene. The MPI-Sintel optical flow dataset was used, this dataset provides the motion vector of each pixel, from one frame to another [94]. Allowing us to compare and test different feature detectors based on different feature detection methodologies to each other using the same dataset. Even though, the MPI-Sintel dataset is synthetic it exhibits the characteristics of natural scenes, it provides a rich set of features that is sufficiently difficult to match. The images in this dataset contain large motions, occlusions, specular reflections, dynamic shadows, motion blurs, defocus blur and atmospheric effects such as blooming, smoke and fog.

5.1.1 Feature detector, implementation information and settings

The Super-feature algorithm doesn't provide the ability to detect features, and should be integrated with current generation feature detectors. Therefore, three different feature detectors were selected and used for testing purposes: SIFT, SURF and MSER. These feature detectors are based on different theories and use different methods of detecting and localizing features. As a result, a common set of detected features between these methods cannot be expected, consequently a dense optical flow field is used to establish a ground truth that can be used to evaluate the feature detectors using a common framework. For SURF [95] and MSER [96], the built-in implementations provided by Matlab R2013A was used to test the application of Super-features in-conjunction with these detectors. The original C implementation of SIFT provided by Lowe was used for testing the SIFT based Super-feature algorithm [97], the compiled version of Lowe's implementation was called using a Matlab wrapper. The default setting proposed by the respected Authors of these feature detection methods was used to perform the tests. Here is a breakdown of the setting used for each feature detection method used for testing.

The SIFT detector was configured with the following setting:

- Feature description size = 128
- Prior smoothing sigma = 1.6
- Number of Octaves = 4
- Number of scales per octave = 3

The SURF detector was configured with the following setting:

- Feature description size = 64
- Metric threshold = 1000
- Number of Octaves = 3
- Number of scales per octave = 4

The MSER detector was configured with the following setting:

- Feature description size = 64 (Based on SURF Descriptor)
- Threshold delta = 2
- Region area range = 30 to 14000
- Maximum area variation = 0.25

5.1.2 Interpreting the results

To understand the result figures, a brief description of each label's meaning and descriptions is provided. In each figure, three different sub-results will be provided per feature detector, nl. Original, Exclusion and Inclusion. Here is a breakdown of each of these figure terms and their descriptions:

- Original - Results obtained by the traditional feature detectors before the Super-feature algorithm were integrated, this usually forms the baseline.
- Exclusion - Super-feature algorithm applied to the original feature matches to exclude features the algorithm believes is incorrect.
- Inclusion - After Super-feature Exclusion was applied, an optional second processing stage can be employed, where the matching position of incorrect features are corrected using the estimated position. Inclusion provides the results of the full Super-feature algorithm.

Applying only Super-feature Exclusion to the original feature matches will result in the most reliable feature detectors, but the feature count might be limited. This is important for some applications, where feature matching reliability is more important than feature count. The Super-feature algorithm with Inclusion of corrected features will perform the second processing stage after Super-feature Exclusion was applied, this will give rise to a higher number of valid features detected, but might result in a slight decrease in reliability and classification accuracy. Applications with a need for a denser set of features will benefit most from the Super-feature Inclusion algorithm.

The matching accuracy is a reliability percentage measure calculated from the number of correct feature matches. Each feature is evaluated against the ground truth motion to determine if it was correctly or incorrectly matched. This is achieved by calculating the positional difference of each feature's matched position to the ground truth transformation position. The ground truth transformed position of a feature is calculated from the feature's original position combined with the ground truth optical flow vector at that pixel location provided by the specified testing dataset. This position error is then compared to a maximum position error threshold selected as a 5 pixel radius around a feature to make the final classification.

The True-positive usage is calculated from the ratio of the number of correct features found after the Super-feature algorithm has been applied, to the number of correct features of the original algorithm without Super-feature integration. This is an important measure since the Super-feature algorithm attempts to maximize the number of True-positive feature usage. The Super-feature Exclusion algorithm will attempt to classify each feature as being either correct or incorrect according to neighbouring feature support, without changing any feature matches. The Super-feature Inclusion algorithm will work similarly to the Exclusion algorithm but it will also try to improve the True-positive usage rate by altering incorrect feature matches which it believes has a high estimation reliability.

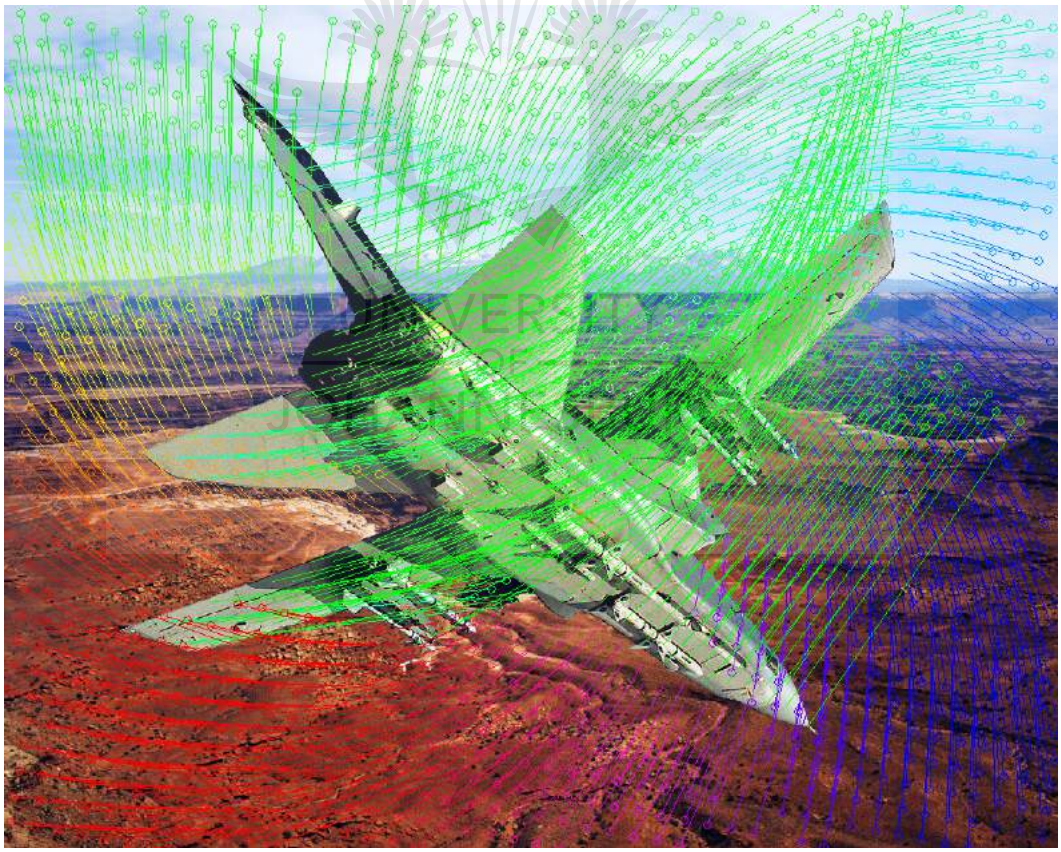
The classification accuracy is determined by comparing the classification results of each algorithm to the ground truth feature classifications, determined from the ground truth transformed position. It is important to evaluate a feature detection algorithm on both the classification accuracy as well as the matching accuracy. This is because a poor feature matching method can discard almost all of the present features in an attempt to improve its matching accuracy. However, since it has discarded the majority of its features, it will reduce its usefulness. In this example case the matching accuracy might be high, but classification accuracy will be low since a large number of useful True-positive features were unnecessarily discarded. An ideal feature detector should have a high matching accuracy as well as a high classification accuracy to ensure that the matching data is used optimally.

5.2 Selection of the Super-feature algorithm parameters

To determine appropriate parameters for the Super-feature algorithm, a natural image dataset was generated by synthetically transforming and fusing photographs of the foreground and the background to form a new image where the ground truth motion is known. This dataset was designed to exhibit composite complex transformations between different foreground and background objects aswell as contain image region occlusions. To be able to compare the algorithm's performance, the synthetically transformed natural dataset needed to provide the ground truth motion vector for each pixel from the Primary image to the Transformed image. Consequently, enabling the testing of different feature detectors based on different feature detection methodologies. Natural photographs of an Aircraft and the Grand Canyon were manually segmented, transformed and composited together to form the test dataset images as seen in Figure 5.1.a and Figure 5.1.b. Different known rotations, scales and affine transformations were applied to the foreground as well as the background image. Occlusions can occur naturally due to the different transformations applied, especially on the borders between the foreground and background. The resulting transformed foreground and background images were then composited together to form the Transformed image presented in Figure 5.1.b. This dataset also contains a number of duplicate features due to the symmetry of the Aircraft as well as the near-stochastic nature of the Canyon image. Since the different applied transformations are known, an optical flow image can be generated that represents the motion of each pixel of the Primary image to the Transformed image, allowing feature matches from the Primary image to the Transformed image to be evaluated. A demonstration of some of the flow vectors can be seen in Figure 5.1.c. The generated dataset was then used to test the effect of changing each algorithm parameter over a range of values and its effect on the matching accuracy.

(a) Primary image used as Frame 1

(b) Transformed image used as Frame 2



(c) Example motion vectors from the Primary image to the Transformed image

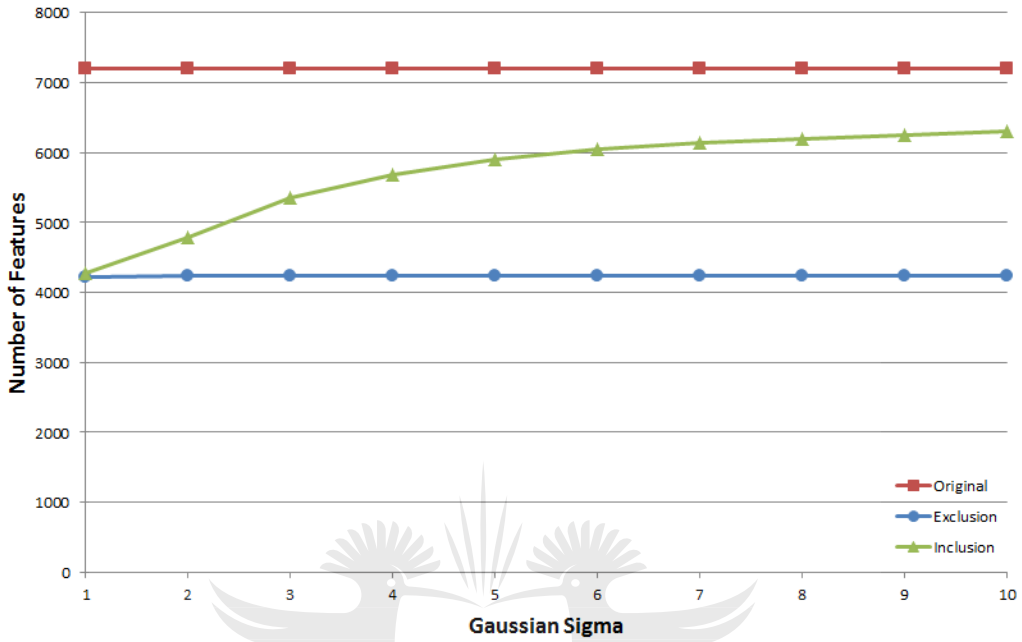
Figure 5.1: An example of the constructed natural test images that were created by transforming the foreground and background using known transformations

5.2.1 Selection of Gaussian Sigma

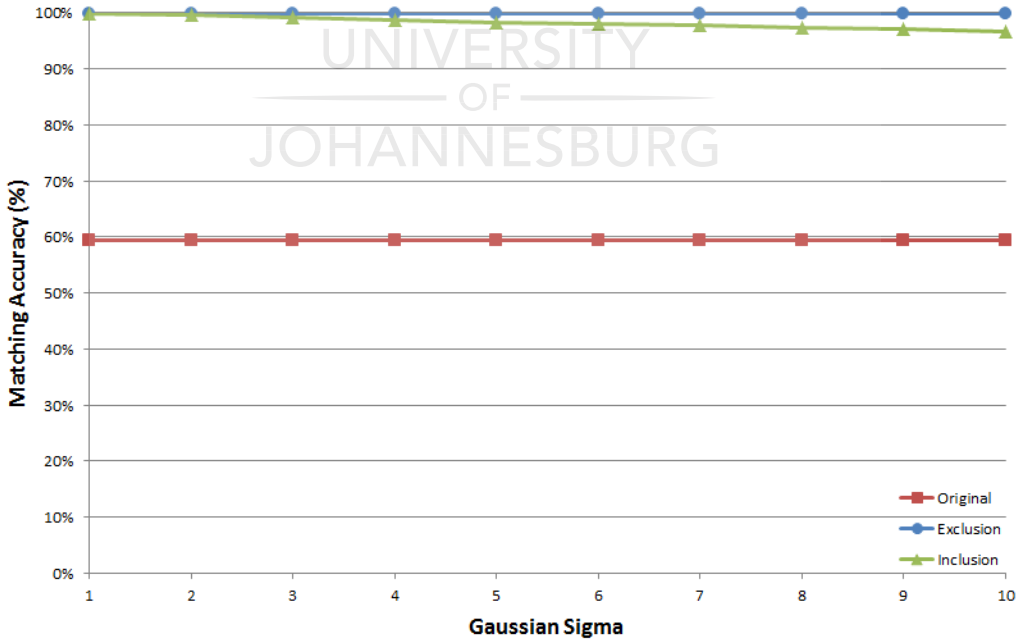
The Gaussian Sigma parameter of the Super-feature algorithm determines the size of the influence region of each pair of neighbouring feature's position estimates on the Probability density function. A larger sigma will relax the accuracy constraint of individual local estimates created by neighbouring feature sets, which will allow them to contribute to a potential global estimate even if they contain some error. This may improve the algorithm's ability to estimate the matching position of a selected feature, even if only a few neighbouring features were matched correctly or complex local transformations are present in the input images. Unfortunately, a negative effect of a larger sigma is that it will reduce the localization accuracy of the final globally estimated position of the feature match. A smaller sigma on the other hand will have the opposite effect, the globally estimated position will be localized more accurately, but individual local estimates may be discarded since the Super-feature algorithm believes that they may contain some estimation error determined from its weak global consistency compared to other local estimates. Therefore a smaller sigma is more ideal for feature detectors that provide a dense set of reliable features and a larger sigma will work better when a sparse set of weak features need to be matched. This is due the fact that the estimation accuracy produced by the neighbouring feature matches decrease as the distance between the neighbours and the primary feature increase. The angle between the neighbouring feature matches and the primary feature can also have an effect, angles close to a right angle tend to produce more accurate estimates.

The results obtained by adjusting the Gaussian Sigma parameter of the Super-feature algorithm over the range of 1 to 10 sigma is provided in Figure 5.2 and Figure 5.3. In these figures, "Original" represents the results of the original feature matches before the Super-features algorithm was applied, "Exclusion" represents the results after the weak feature matches have been removed using the Super-feature algorithm and "Inclusion" represents the results where the Super-features algorithm attempts to correct bad feature matches.

Super-feature Exclusion provided the most reliable results in this test with an average matching score of close to 100% over the entire range of tested Gaussian Sigmas compared to the Original matching accuracy obtained by SIFT, which

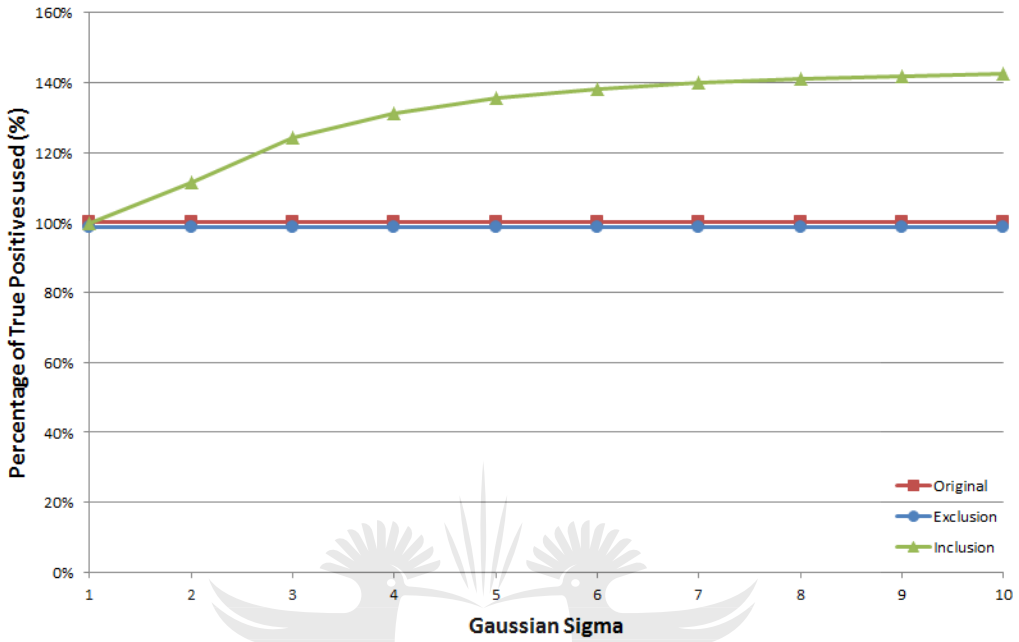


(a) Number of Features

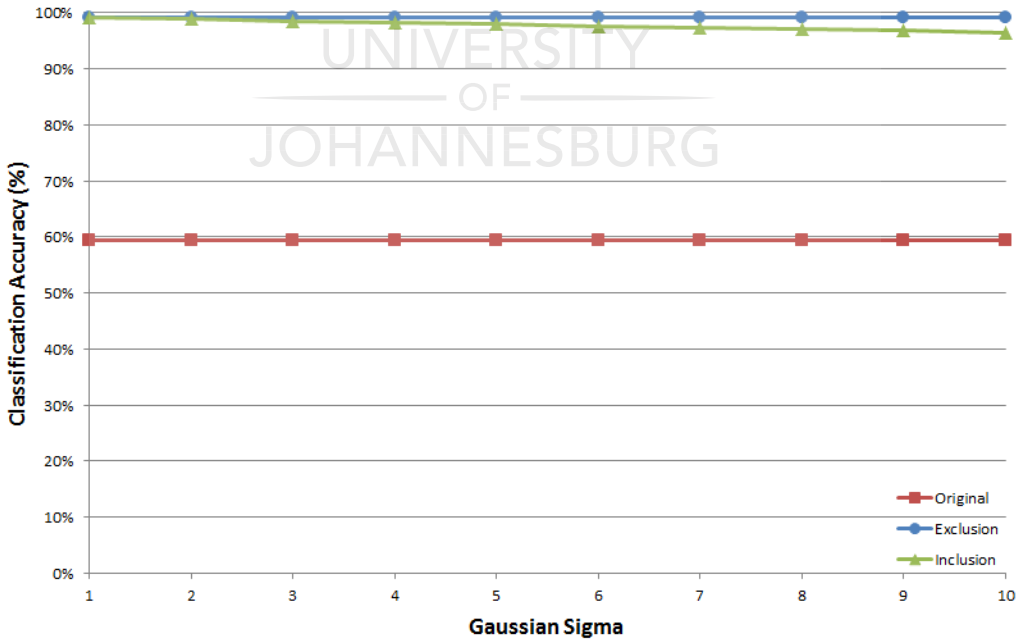


(b) Matching Accuracy

Figure 5.2: The number of features and matching accuracy results obtained by adjusting the Gaussian Sigma parameter of the Super-features algorithm



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.3: The percentage of True-positives used classification accuracy results obtained by adjusting the Gaussian Sigma parameter of the Super-features algorithm

was only 59.4%. Super-feature Exclusion removed poor feature matches from the match list, it removed only feature matches that had a high probability of being incorrect. As a result 58.7% of the total detected features were matched using Super-feature Exclusion. This result comes close to the number of correct features present in the ground truth where only 59.4% of the features were correct, which means that less than 1% was misclassified by the Super-features algorithm and it had a classification accuracy of 99.2% as seen in Figure 5.3.a.

The greatest effect of changing the Gaussian Sigma was the increase of the number of features, as observed for the Super-features Inclusion algorithm. Super-features with Inclusion attempts to correct and improve the features it believes were matched incorrectly and includes them in the final match list. It can be seen that as the Sigma is increased the Super-feature algorithm is better able to estimate the positions of mismatched features. Resulting in more features being corrected and matched, negatively some incorrect matches might be included in the final match list, which could result in a possible decrease in matching accuracy. Therefore a larger Sigma resulted in more feature matches being corrected and included with a slight decrease in matching accuracy.

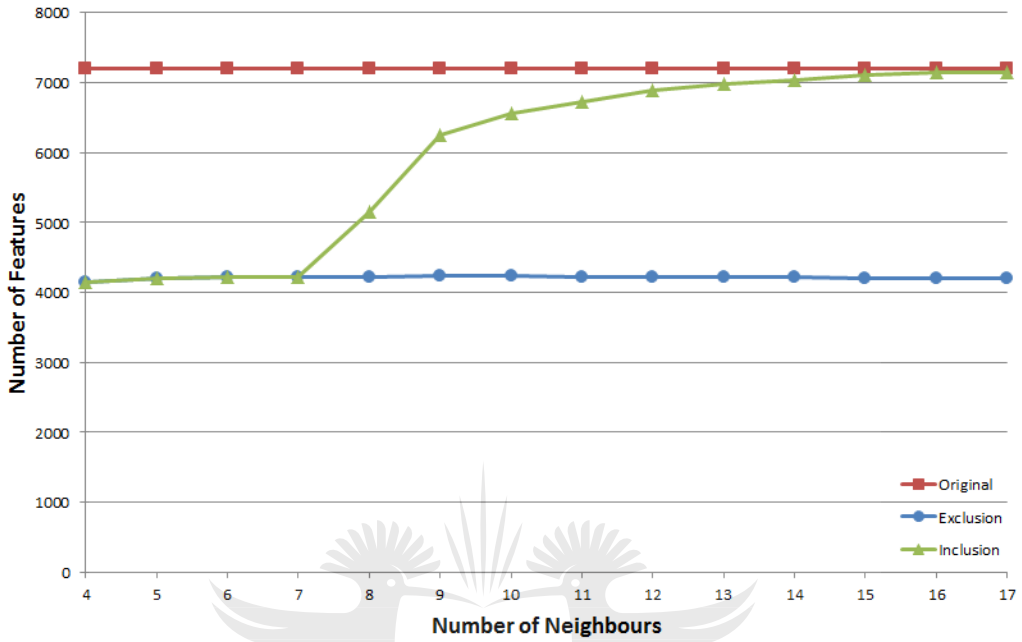
The increase of the Gaussian Sigma negatively impacted the classification accuracy, on the other hand it dramatically improved the percentage of True-positives used as can be observed in Figure 5.3.a. A Gaussian Sigma of 4 was empirically selected as an effective value for the Gaussian Sigma parameter of the Super-feature algorithm, it resulted in an increase of 34.4% more features being detected with only a 1.2% decrease in matching accuracy.

5.2.2 Selection of the number of neighbours

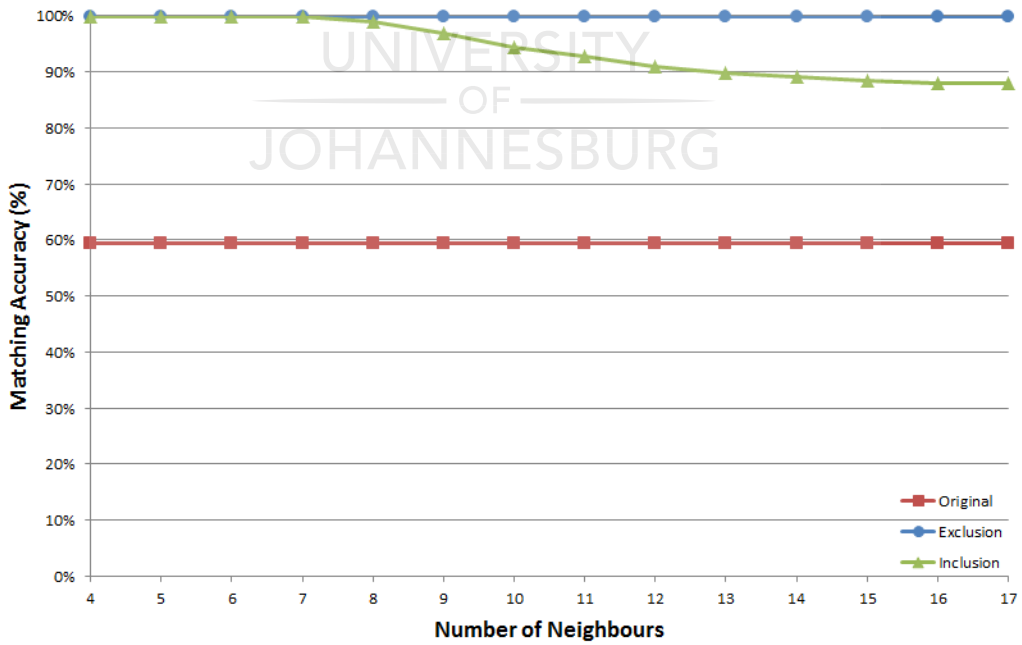
The correct selection of the "number of neighbours" parameter that the Super-feature algorithm will use, is an important issue. A larger "number of neighbours" value will improve the algorithm's ability to provide reliable estimates in the presence of outliers. This will unfortunately come at the expense of increased processing time. Another problem with the selection of a large "number of neighbours" value is that small moving objects that only contain a small set of features can be misclassified as being incorrect. This happens because the

background behind the object will provide more support for the features on the object than the features on the object. This will result in position estimates that assume that the features on the object should be part of the background and thus will be constrained to the motion of the background. The Super-feature algorithm does handle this situation to some extent by providing multi-mode estimates of the underlying density distribution, for each feature where required. In these situations an estimated position proposed by the background and the foreground will be provided, but the Super-feature algorithm is unable to distinguish which one of these possible solutions are correct. The selection of a smaller "number of neighbours" value will allow the Super-feature algorithm to function on smaller objects, but will result in less support being provided for the Super-feature position estimator. Consequently, this can result in an algorithm with poor classification accuracy when large numbers of outliers exist in the match set. On the other hand it will improve and reduce the processing requirements, while increasing its ability to function in situations where the feature sets are sparse or when there are small moving objects present in the images.

The results obtained by changing the "number of neighbours" parameter are provided in Figure 5.4 and Figure 5.5 The Original matching accuracy provided by the feature detector was 59.4%, after Super-feature Exclusion was integrated it provided stable results close to a 100%. Consequently, it shows that even when using only a small number of neighbours, such as 4, in situations where less than 40% of the Original matches are incorrect the Super-features algorithm can provide an improvement over the traditional method. The results obtained by the Super-feature with Inclusion of corrected features unfortunately suffered when a large number of neighbours were selected. Accordingly, it can be observed in Figure 5.4.b that there is a slow decline of the matching accuracy as the number of neighbours are increased. Increasing the number of neighbours can have a positive effect on the number of True-positives used as well as the number of features matched albeit with a reduced matching and classification accuracy as seen in Figure 5.5.a. One of the reasons for the decrease in matching accuracy of the Super-feature Inclusion algorithm is that more multi-mode solutions can occur as the number of neighbours are increased. Unfortunately the Super-feature algorithm can only correct features with single, reliable solutions even if it can

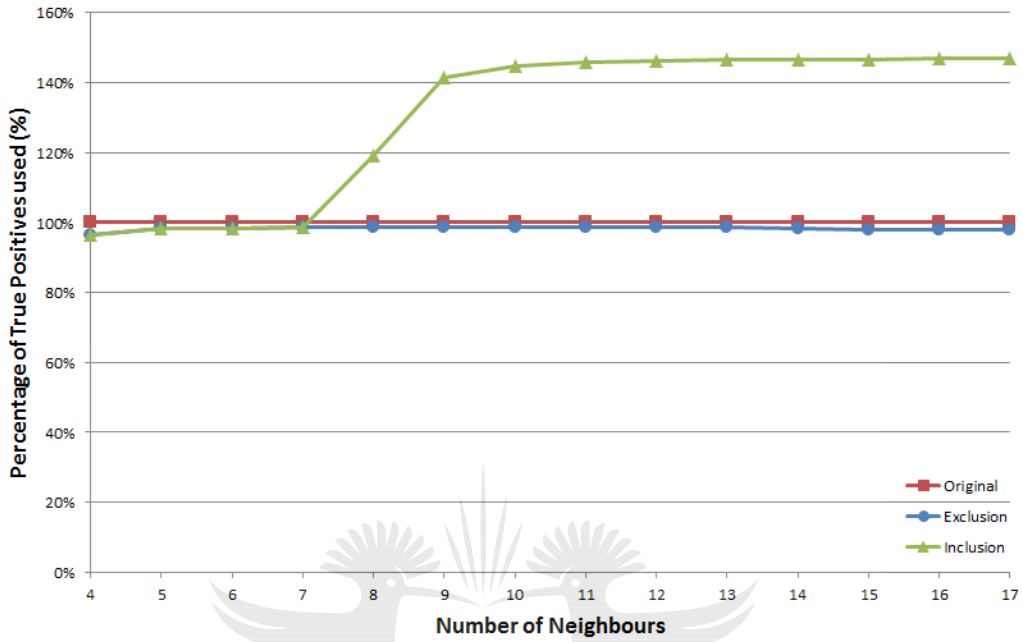


(a) Number of Features

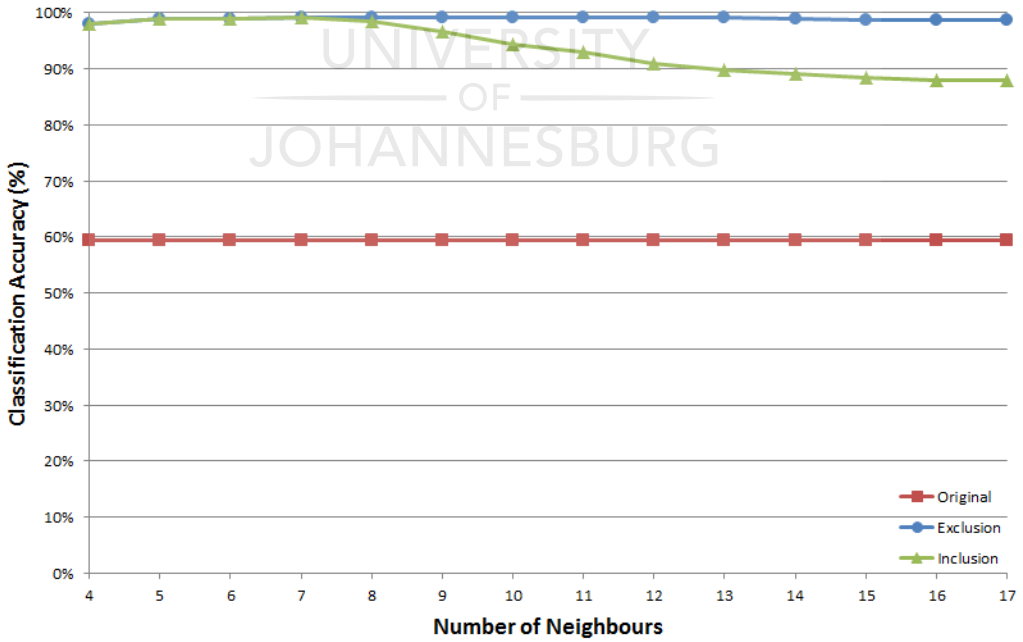


(b) Matching Accuracy

Figure 5.4: The number of features and matching accuracy results obtained by adjusting the "number of neighbours" parameter of the Super-features algorithm



(a) Percentage of True-positives used



(b) Classification Accuracy

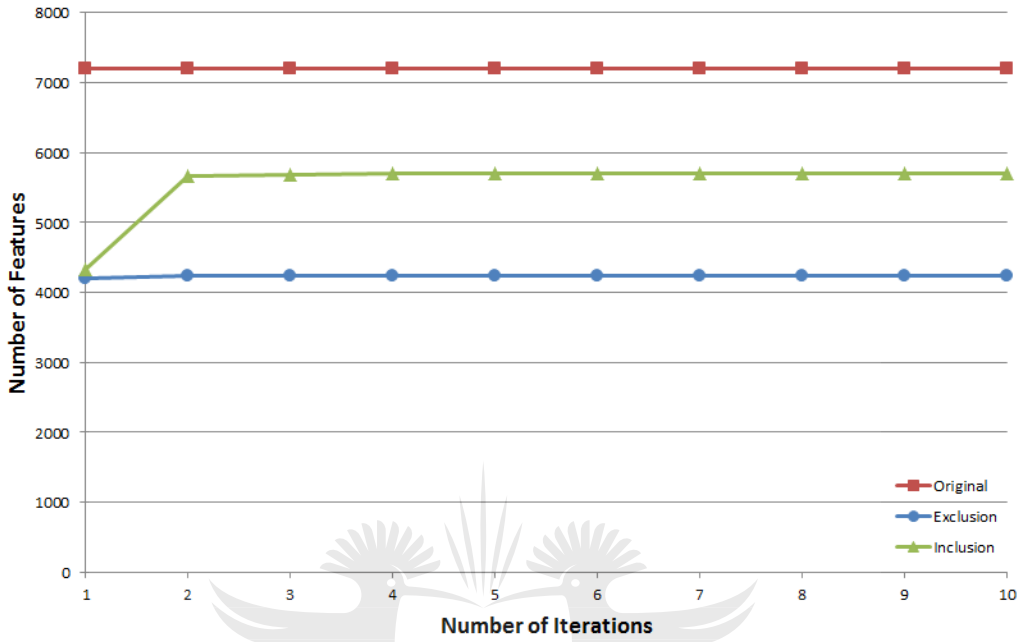
Figure 5.5: The percentage of True-positives used and classification accuracy results obtained by adjusting the "number of neighbours" parameter of the Super-features algorithm

provide the information of multiple possible solutions when they occur. Another possible reason for the reduced matching accuracy is that the support provided by sparse feature regions can be overwhelmed by the feature support provided by stronger neighbouring regions. A "number of neighbours" parameter of 10 was determined to be an effective selection which provided an increase to the number of features detected and the True-positive usage percentage, while maintaining a good matching accuracy as well as reducing the processing requirements.

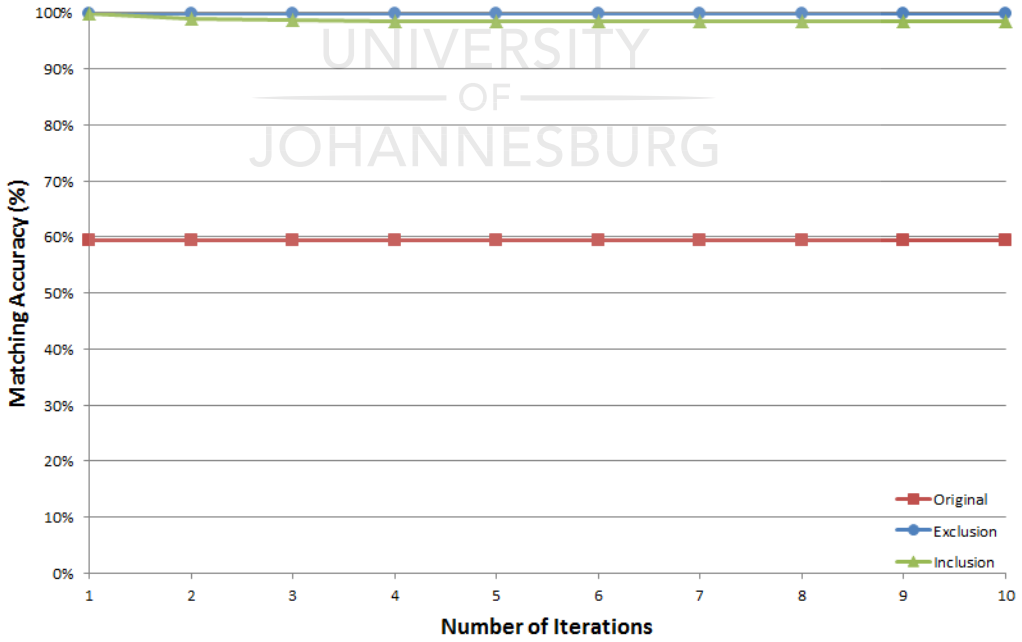
5.2.3 Selection of the number of iterations

The number of iterations parameter of the Super-feature algorithm allows it to repeat the voting process a number of times. Each iteration will repeat the Super-feature process, but will only use feature matches that it believes have a high reliability confidence derived from the previous iteration. During the first iteration, all available features matches will have to be used which could result in some misclassifications, as some potentially incorrect features will provide invalid support for other features. The succeeding iterations improve on the prior iteration's results by providing more insight around the validity of features, resulting in the improvement of the feature matching classification accuracy. This can be attributed to the fact that each iteration is able to remove more and more weak features from the valid feature set, excluding them from the Super-feature algorithm's voting process allowing only strong feature matches to contribute and provide support. In general the Super-feature algorithm tends to converge quickly, usually within only a few iterations after which no further benefit is apparent.

The Super-features algorithm converged completely after 3 iterations for this test dataset, after which no real improvement was visible as seen in Figure 5.6 and Figure 5.7. There was only a slight improvement observed by executing an addition iteration after the second iteration was completed. In some applications where processing resources are constrained, 2 iterations can be used without a significant decrease in matching and classification accuracy. A single iteration is sufficient when only Super-feature Exclusion is required, providing a reliable feature set requiring only a third of the processing resources. From these results

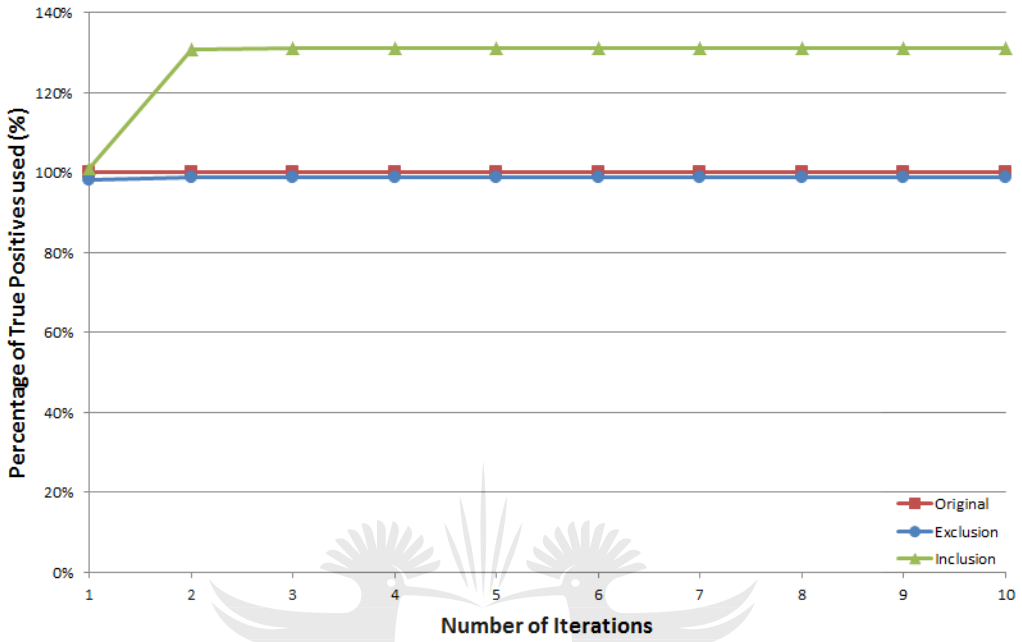


(a) Number of Features

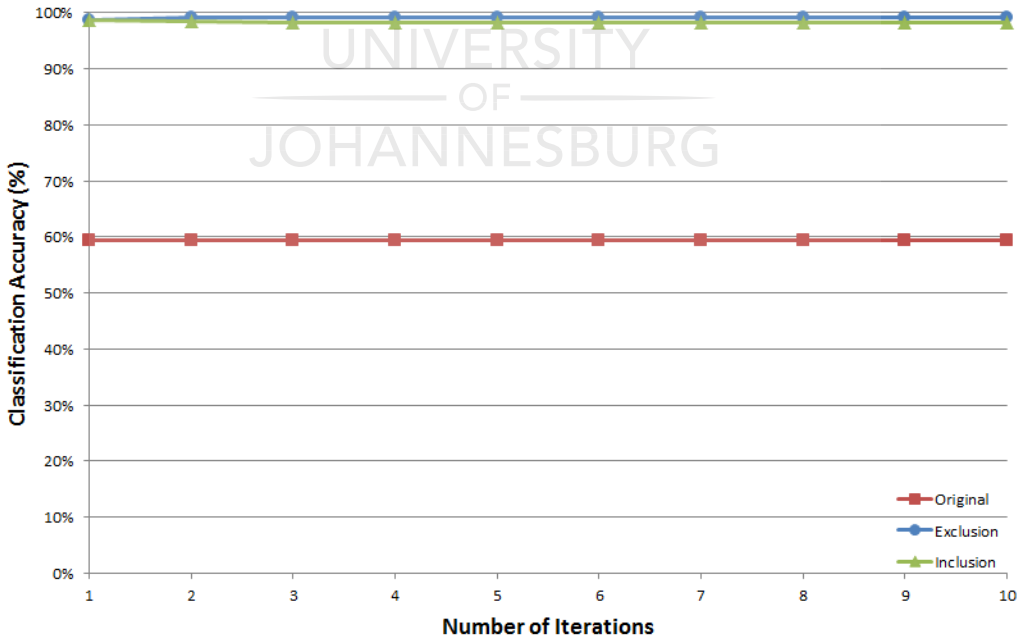


(b) Matching Accuracy

Figure 5.6: The number of features and matching accuracy results obtained by adjusting the number of iterations parameter of the Super-features algorithm



(a) Percentage of True-positives used



(b) Classification Accuracy

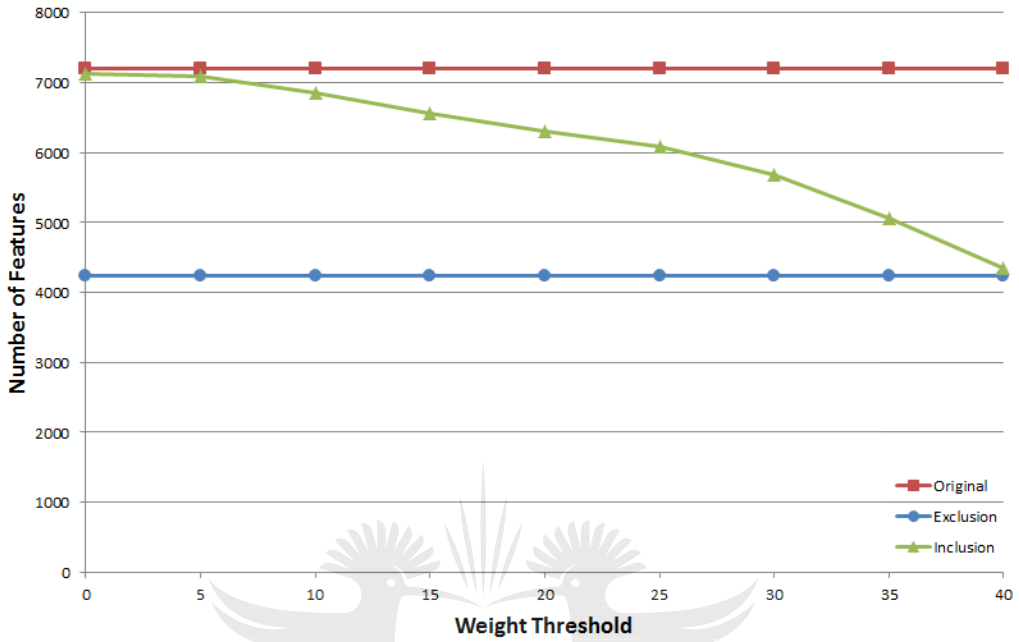
Figure 5.7: The percentage of True-positives used and classification accuracy results obtained by adjusting the number of iterations parameter of the Super-features algorithm

the number of iterations of 3 was selected as an effecting parameter selection for the Super-feature algorithm, this selected number of iterations will provide the best results for most applications.

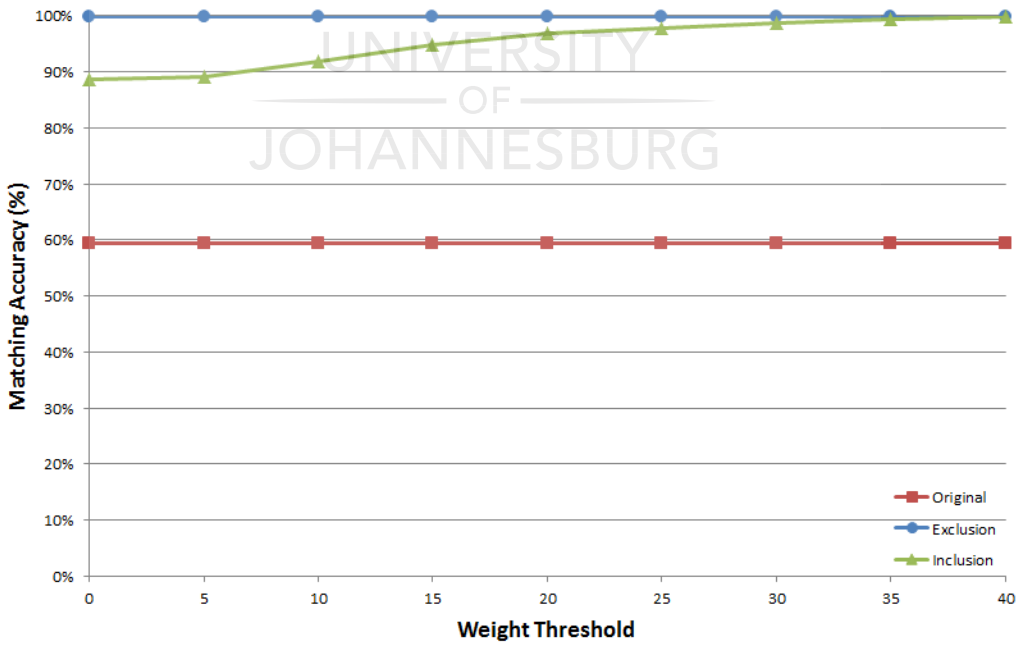
5.2.4 Selection of weight threshold

The weight threshold parameter is specific to the Super-feature correction and Inclusion process and does not impact the Super-feature Exclusion algorithm. Therefore, the Super-feature Exclusion results should not have been included in the results since this parameter does not affect the algorithm's operation, but it was included to provide a baseline against which to compare the Super-feature Inclusion algorithm when the weight threshold is adjusted. In the Super-feature Inclusion algorithm the weight threshold is used to trim and discard incorrect features that have poor estimation reliability, calculated from the Probability density function of the local neighbour feature match votes. Feature matches that have been classified as being incorrect and that have an estimated reliability stronger than the selected weight threshold are corrected by changing their matched positions to the Super-feature's estimated positions derived through the neighbouring feature match voting process. When a higher weight threshold is selected it will result in a strict correction policy, only allowing incorrect features with a strong estimation reliability to be corrected and included in the process. By selecting a low weight threshold the correction policy will be relaxed, resulting in an algorithm that is more lenient when correcting features matches, some errors might occur because of this relaxed constraint.

As previously explained and can be observed in Figure 5.8.a, the number of features produced by the Super-feature Inclusion algorithm decrease as the weight threshold parameter is increased. Consequently, by adjusting the weight threshold, it also has an effect on the matching accuracy, which improved due to the strict thresholding criteria as seen in Figure 5.8.b. When selecting a strict weight threshold (such as 40), only the strongest features will be corrected and included, resulting in a small number of features being selected. An ideal weight threshold value will provide a high matching accuracy, but also correct and include a large number of features. For this reason, a weight threshold of 30 was selected,

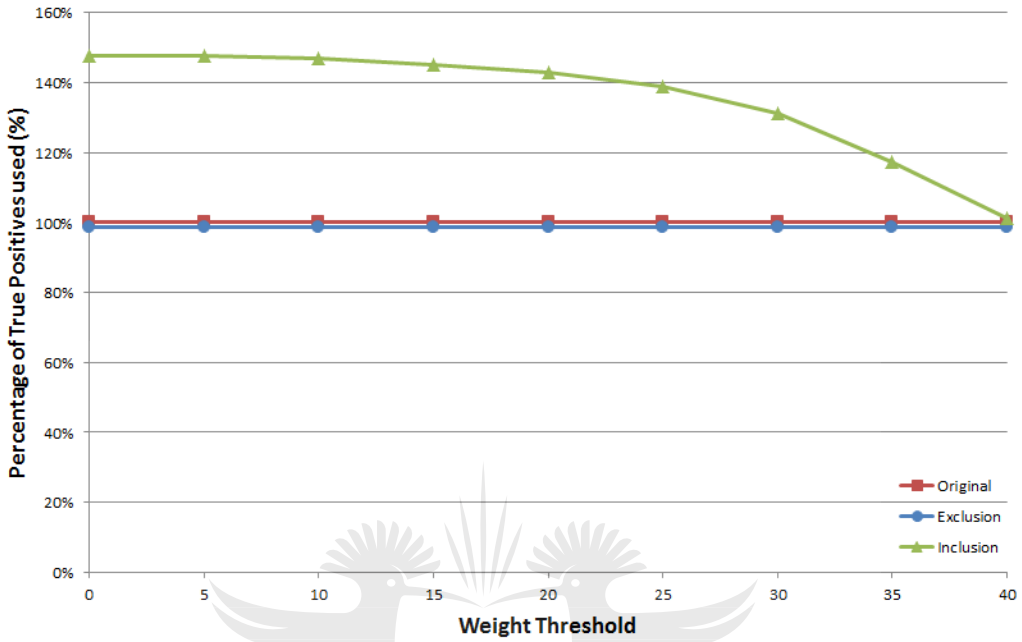


(a) Number of Features

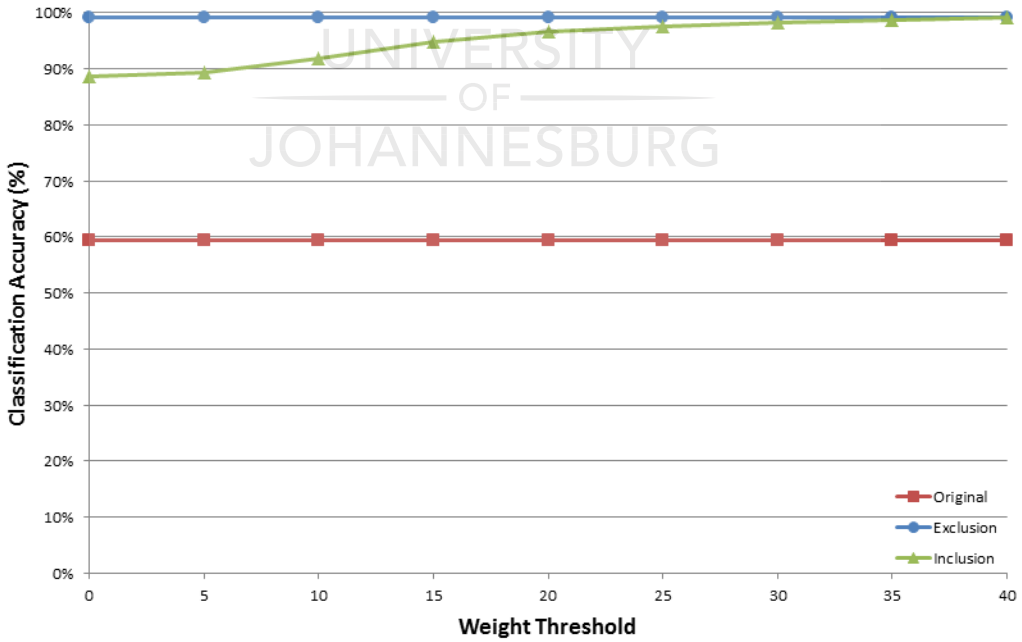


(b) Matching Accuracy

Figure 5.8: The number of features and matching accuracy results obtained by adjusting the Weight Threshold parameter of the voting scheme



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.9: The percentage of True-positives used and classification accuracy results obtained by adjusting the Weight Threshold parameter of the voting scheme

it produced 34% more features than Exclusion but only reduced the matching accuracy by less than 1%. This weight threshold will also allow more True-positives to be used compared to the Original and Super-feature Exclusion feature matches as can be seen in Figure 5.9.

5.3 Simple global image motion transformations

(a) Bridge image (800x600 pixels)[98]

(b) Nature image (1024x768 pixels)[99]

(c) Aircraft image (1280x1024 pixels)[100]

Figure 5.10: Dataset images used for testing global synthetic image transformations

The test dataset images that will be used for testing the ability of the Super-features algorithm are Bridge, Nature and Aircraft, they can be seen in Figure 5.10. These three images have different resolutions and image information densities, which will result in different amounts of features being detected. Since the sparsity of features can influence the Super-features algorithms accuracy results, these images were selected to test a wide range of feature densities. The feature detectors that we used for testing, detected the least amount of features in the Aircraft image, this can be attributed to the large amount of regions in the image that contain low frequency information as-well as man made structure. There is close to double the amount of features in the Bridge image and two and a half times more feature in the Nature image compared to the Bridge image. This is because the Nature image contains a large amount of high frequency image information, which typically produces a large number of small features. The three test images demonstrate a wide variety of real world environments, from man made structures to natural high frequency information common in nature. All three test images also contain a large number of duplicate features such as the windows and door arches that can be seen in the Bridge image, the rocks and grass regions in the Nature image as well as the symmetrical features on the aircraft in the Aircraft image.

To test the Super-features algorithms ability to handle specific situations, we will test the effect of single global image transformations such as scale, rotation and affine transformation on the accuracy results of the different feature detector before and after the Super-features algorithm has been integrated. Complex combinations of these transformations on multiple moving objects and environments will also be tested later. This will demonstrate to what level the Super-features algorithm has invariance to these transformations and as a result show that reliable feature matching can be established in these situations. Each test requires different amounts of rotation, scale and affine transformations to be applied before the accuracy results are determined, a range of transformations will be tested. The test dataset images will be synthetically transformed according to a selected rotation, scale or affine transformation. Since the transformation is known, it allows us to establish a ground truth that can be used to determine if a feature match or the transformation of a feature from the one coordinate frame to the

next is correct. The current generation of feature detectors has some difficulty in accurately localizing features. Therefore, if a feature's matched position is within a 5 pixel radius of the ground truth position, it will be considered a valid match and will be classified as correct. The accuracy results are determined by matching all features from the original untransformed image to the transformed image. Three different feature detectors will be used to detect the feature sets, these feature detectors will be SIFT, SURF and MSER. These feature detectors are well known and common in the field of image processing and computer vision.

5.3.1 Rotation transformation invariance

To be able to test the rotation handling ability of the Super-features algorithm, the input test images were rotated synthetically over the range of 0 to 360 degrees in 10 degree intervals. Example images of the tested rotation transformations applied to the Aircraft image are provide in Figure 5.11. Features were detected using SIFT, SURF and MSER in the original untransformed image and then matched with the same features detected in each of the rotationally transformed images. Thereafter, the matching and classification accuracy was calculated from

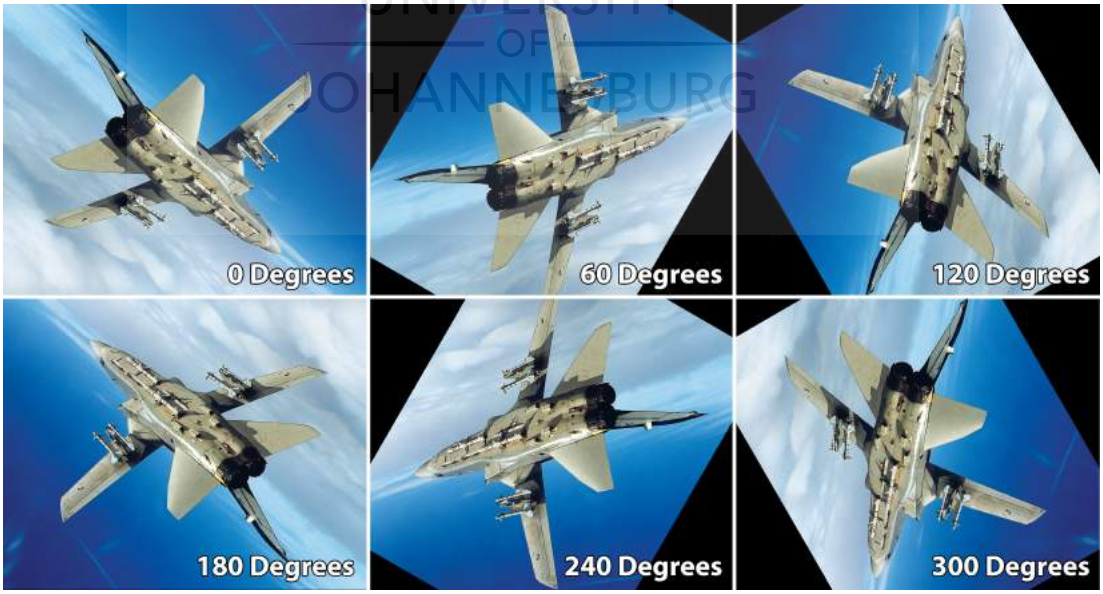
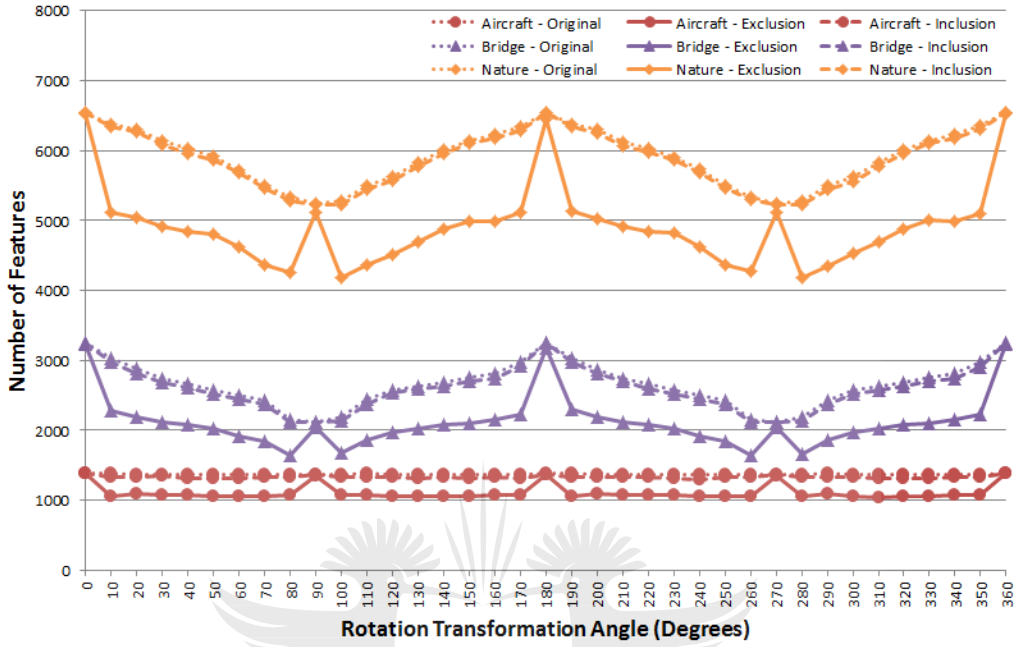


Figure 5.11: Example images of the Aircraft rotation dataset

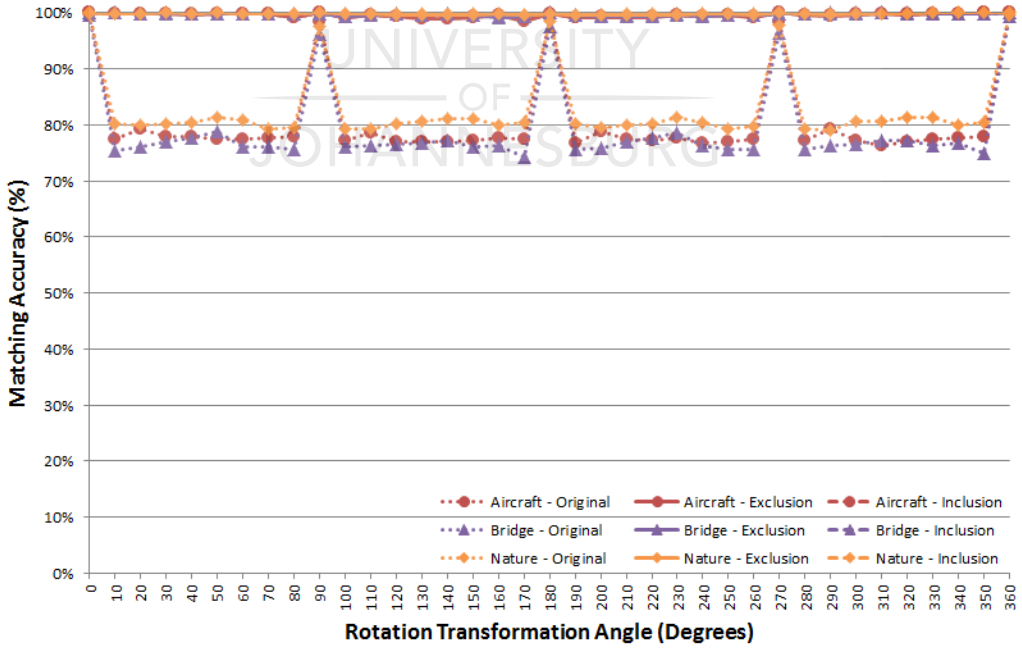
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.12: A comparison of the feature matching results of 20 degree Rotation transformed features by the SIFT and Super-feature algorithm

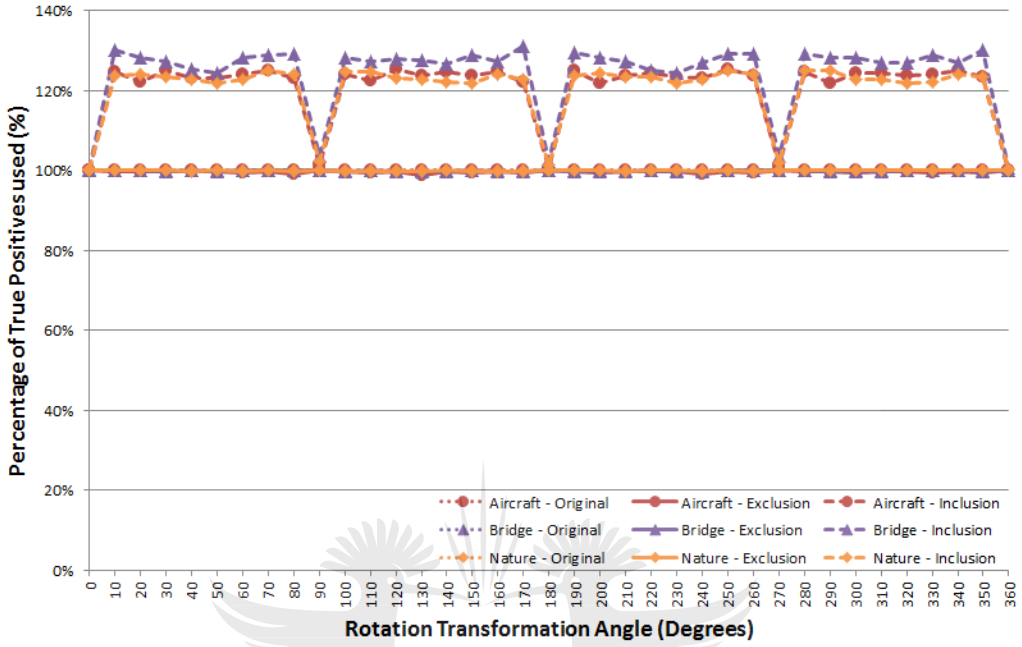


(a) Number of Features

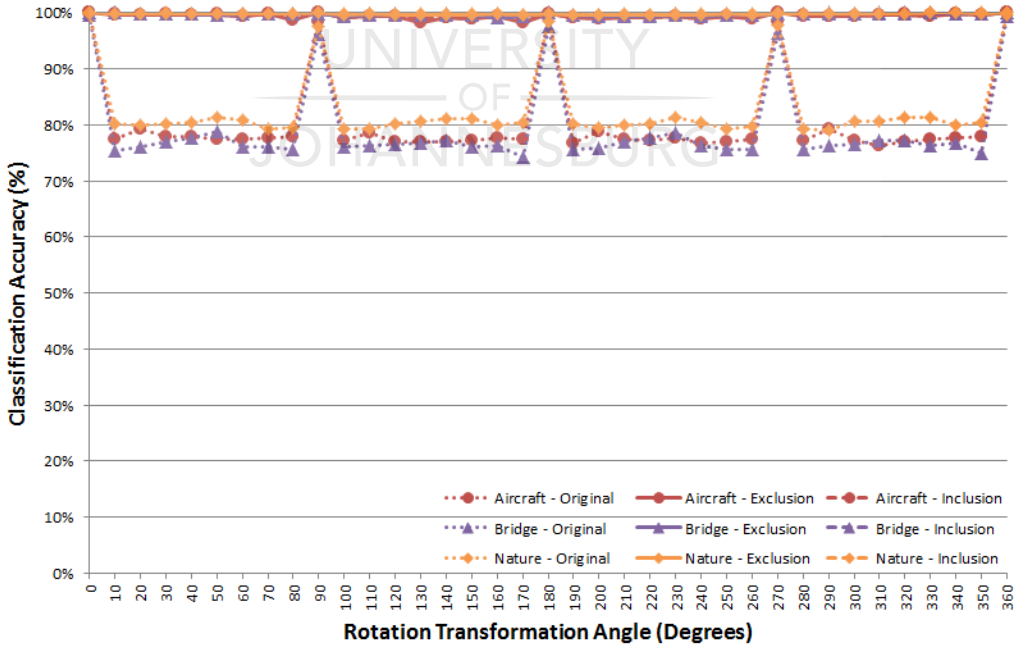


(b) Matching Accuracy

Figure 5.13: The number of features and matching accuracy results obtained by the SIFT based Super-features algorithm for Rotation transformations



(a) Percentage of True-positives used



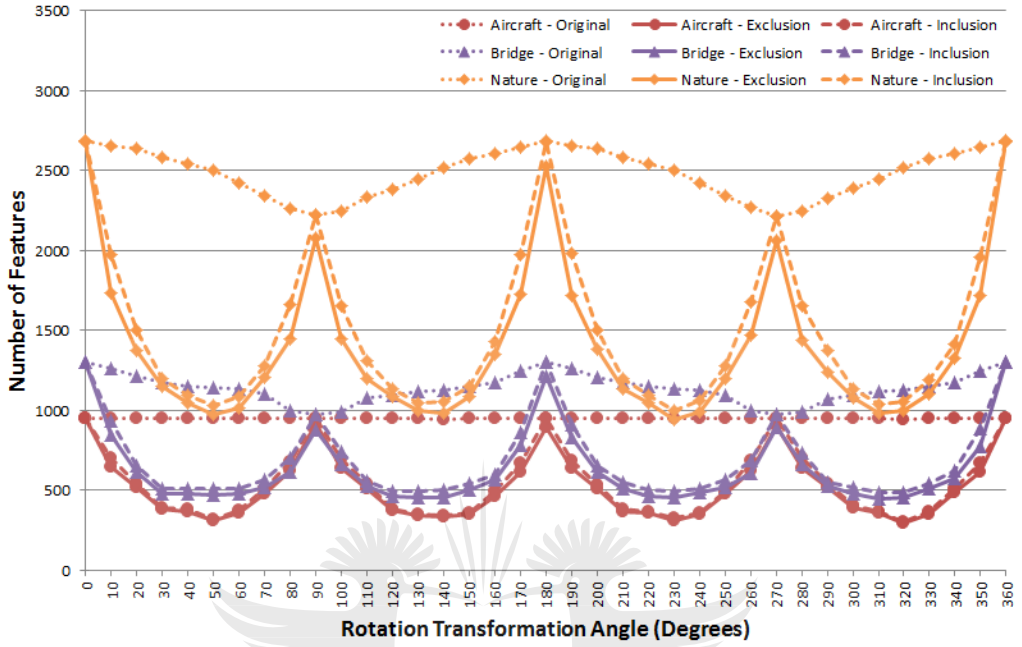
(b) Classification Accuracy

Figure 5.14: The percentage of True-positives used and classification accuracy results obtained by the SIFT based Super-features algorithm for Rotation transformations

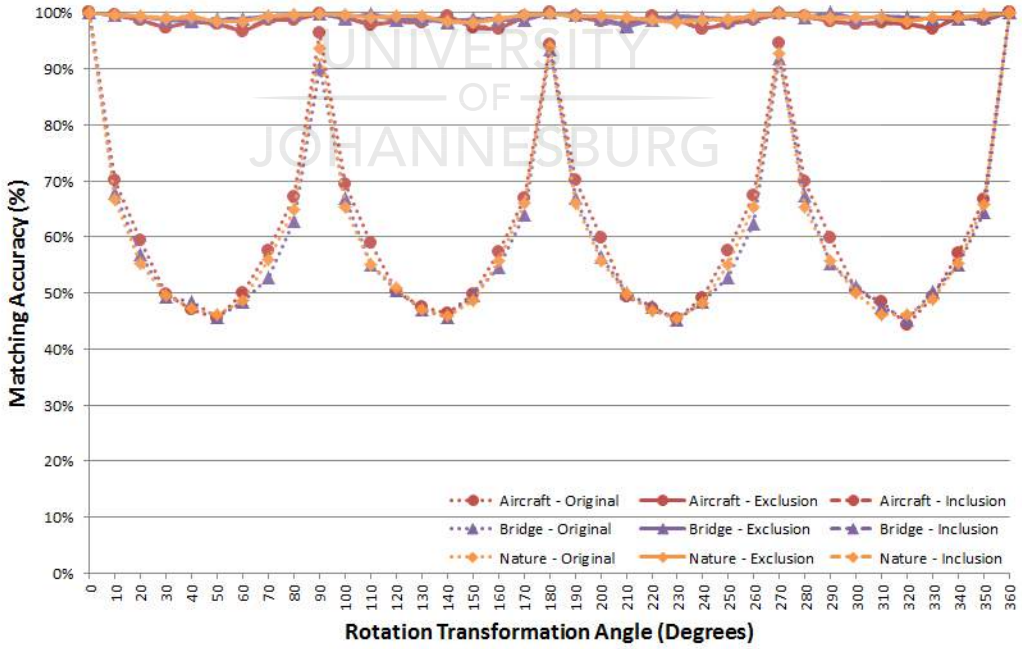
the ground truth flow vectors between these images, which were calculated from the known ground truth rotation angles. It can be observed that more features were detected by the different feature detectors at rotation angles of 0, 90, 180, 270 and 360 compared to the other angles as seen in Figure 5.13.a. Some features were lost because some of the corner regions of the rotated image were transformed outside the image region. Another reason for the observed decrease in the number of features was caused by the blurring that was introduced by sub-sampling of the pixel values caused by the image rotation algorithm. Usually only small high frequency features are lost as a result of the re-sampling process.

The SIFT based Super-feature algorithm produced consistent results over the entire range of tested rotation angles, it maintained a matching accuracy close to a 100% for all rotations. A visual demonstration of the feature matching results of the SIFT and SIFT based Super-feature algorithm can be seen in Figure 5.12. By integrating Super-features in the matching process of SIFT, it produced a substantial matching accuracy improvement of more than 20% compared to the original SIFT feature matching algorithm as seen in Figure 5.13.b. Super-features with inclusion and correction produced similar results to the Super-feature Exclusion algorithm except that the Super-feature Inclusion algorithm was able to make use of more features since it was able to recover some miss-matched features. Consequently, Super-feature Inclusion was able to produced close to a 100% matching and classification accuracy as well as increase the feature count by more than 20% for the majority of the tested rotations. The number of True-positives used also improved after Super-features were integrated compared to the original SIFT feature detection algorithm as seen in Figure 5.14.a. SIFT produced a dens set of features which boosted the performance of the Super-feature algorithm. Neighbouring feature matches were relatively error free with only 20% miss-matches. This allowed the Super-feature algorithm to use the support provided by the neighbouring feature matches to accurately classify feature matches as either correct or incorrect. Similar results were obtained for all three test images which showed that the Super-feature algorithm provides a strong invariance to rotation transformations in many scenarios.

Super-features based on the SURF feature detector produced similar results to the SIFT based Super-feature algorithm, the results are provided in Figure 5.15

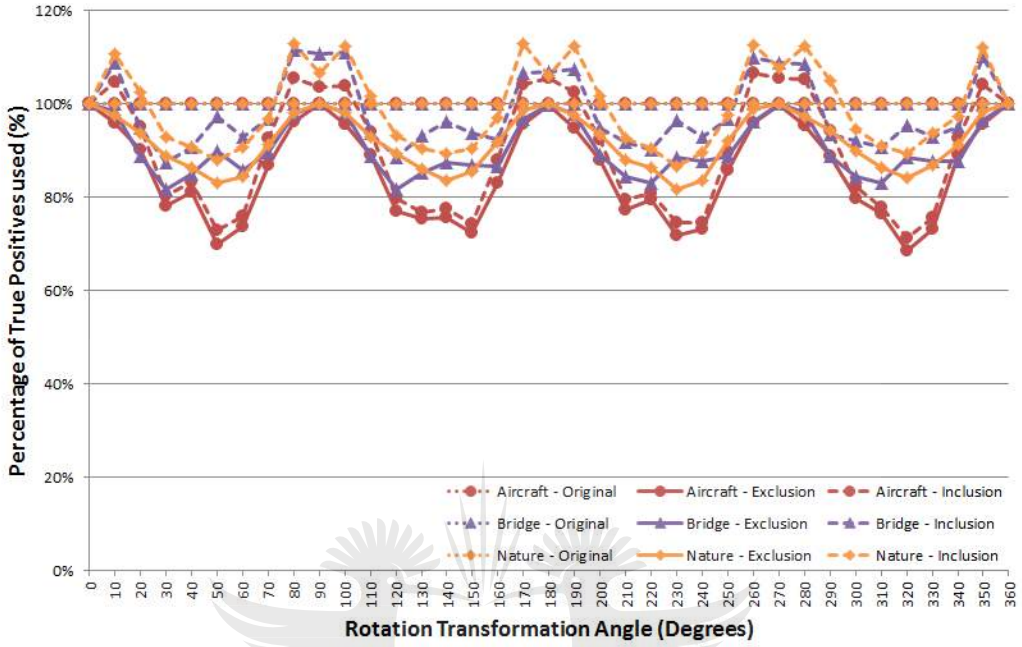


(a) Number of Features

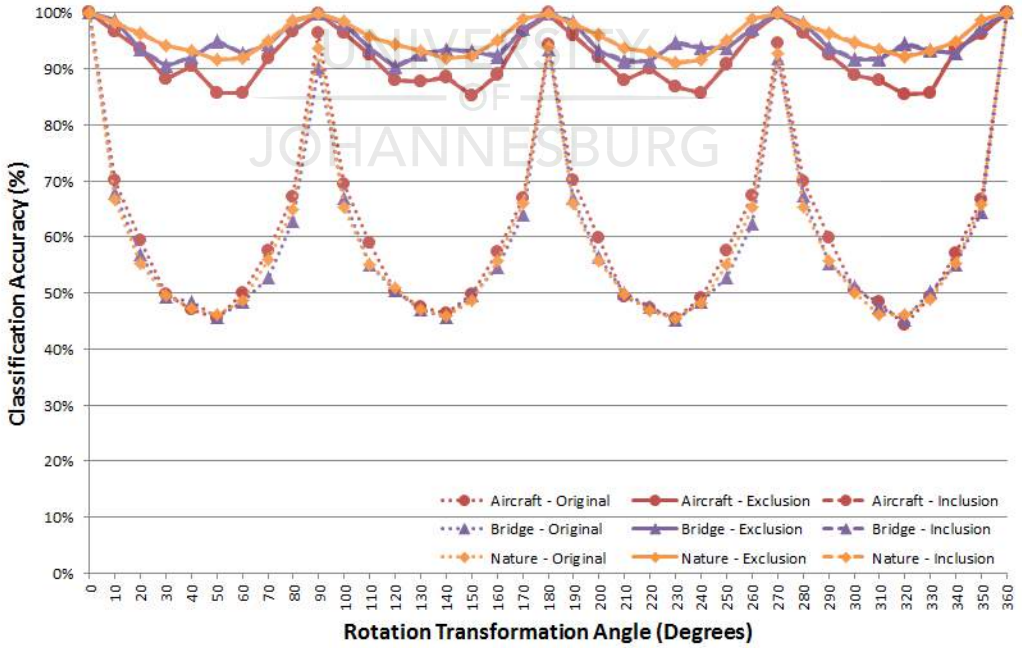


(b) Matching Accuracy

Figure 5.15: The number of features and matching accuracy results obtained by the SURF based Super-features algorithm for Rotation transformations



(a) Percentage of True-positives used

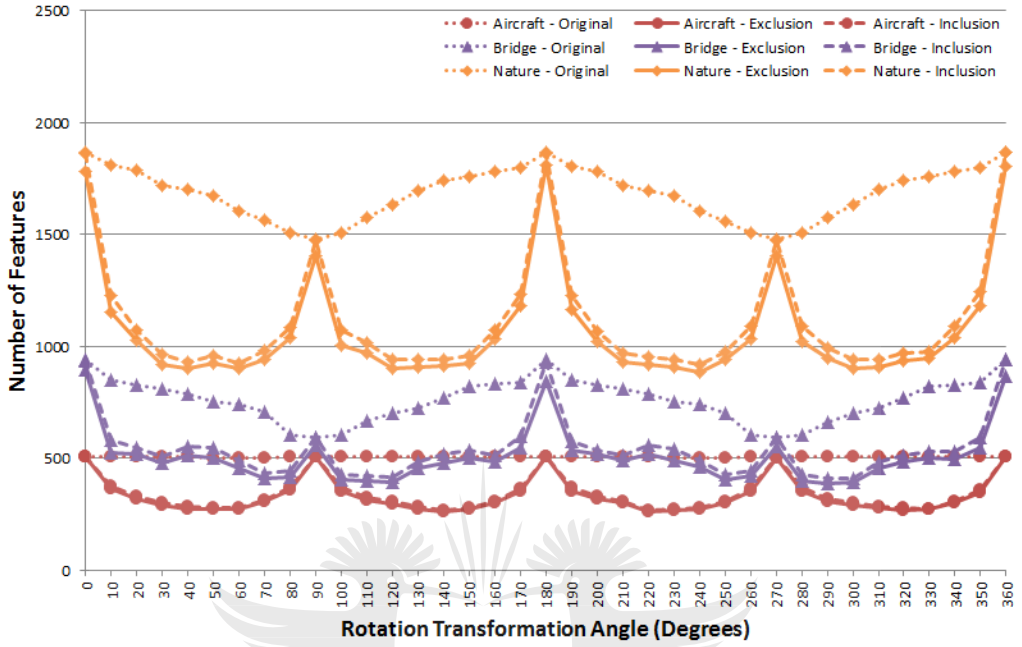


(b) Classification Accuracy

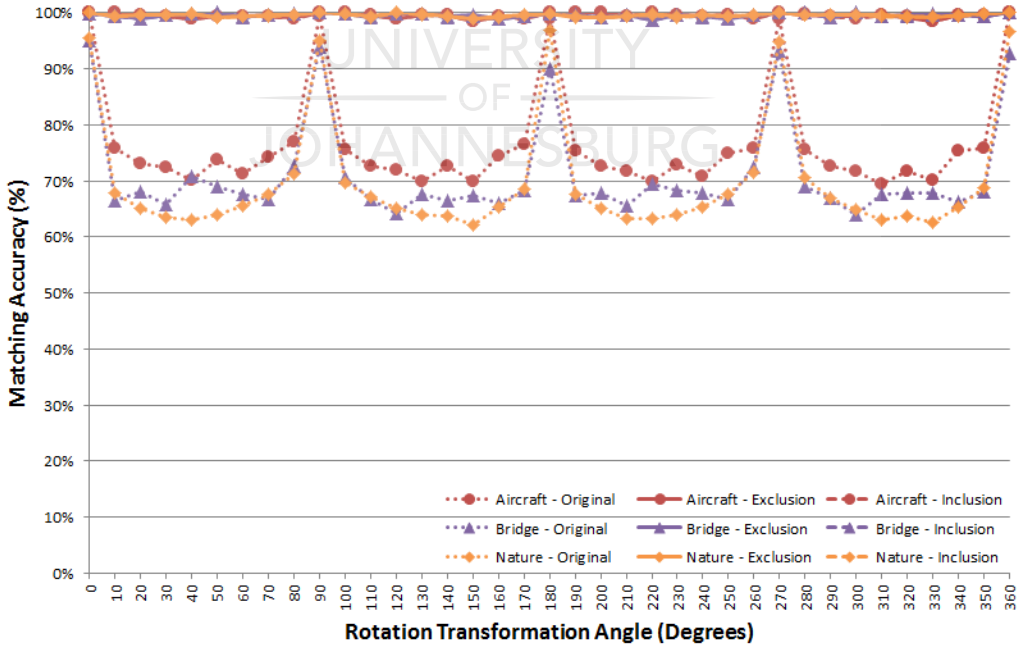
Figure 5.16: The percentage of True-positives used and classification accuracy results obtained by the SURF based Super-features algorithm for Rotation transformations

and Figure 5.16. The default configuration and settings of SURF did not detect as many features as SIFT, also the feature matches that were found contained a substantial amount more mismatches. This makes feature match classification and correction using the Super-feature algorithm more difficult, since more outliers will be present during the estimation process. As seen in Figure 5.15.b the Original SURF matches did not provide strong rotation invariance compared to SIFT, a large number of features were matched incorrectly for rotations between the intervals of 90 degree angles. The poor rotation invariance of the SURF method, negatively impacted SURF's original matching accuracy, for some rotations the matching accuracy dropped by more than 50%. This resulted in a poor matching accuracy of the Original SURF detector which introduced more outliers in the estimation process of the Super-feature algorithm. Thus the impact of the increased number of outliers on the classification accuracy can be seen in Figure 5.16.b. The matching accuracy remained high as seen in Figure 5.15.b but some True-positives were lost since the neighbouring feature matches provided poor support and these features were misclassified and discarded as incorrect matches. The Super-feature algorithm was still able to substantially improve the matching accuracy of SURF even if the SURF detector included more outliers. Consequently, the results obtained by the SURF based Super-feature algorithms were not as good as the results obtained for the Super-feature SIFT algorithm, but it consistently produced results higher than 98% for all rotation angles. From these test results we can observe that the Super-feature algorithm is able to continue functioning reliably even when 50% of the neighbouring feature matches were incorrect. This allowed the Super-feature algorithm to improve on the poor rotation invariant property of the SURF feature detector.

The MSER feature detector, detected the least amount of features compared to SIFT and SURF, these features were sparse in nature which can cause some feature matches to be poorly supported by their neighbouring feature matches. The same decrease in feature count artefact for some rotations observed for SIFT and SURF was also seen for the MSER features, where the amount of features decrease for rotation intervals between 0, 90, 180, 270 and 360 degrees. The MSER and MSER based Super-feature algorithm results can be seen in Figure 5.17 and Figure 5.18. As can be observed from these results, the rotation invari-

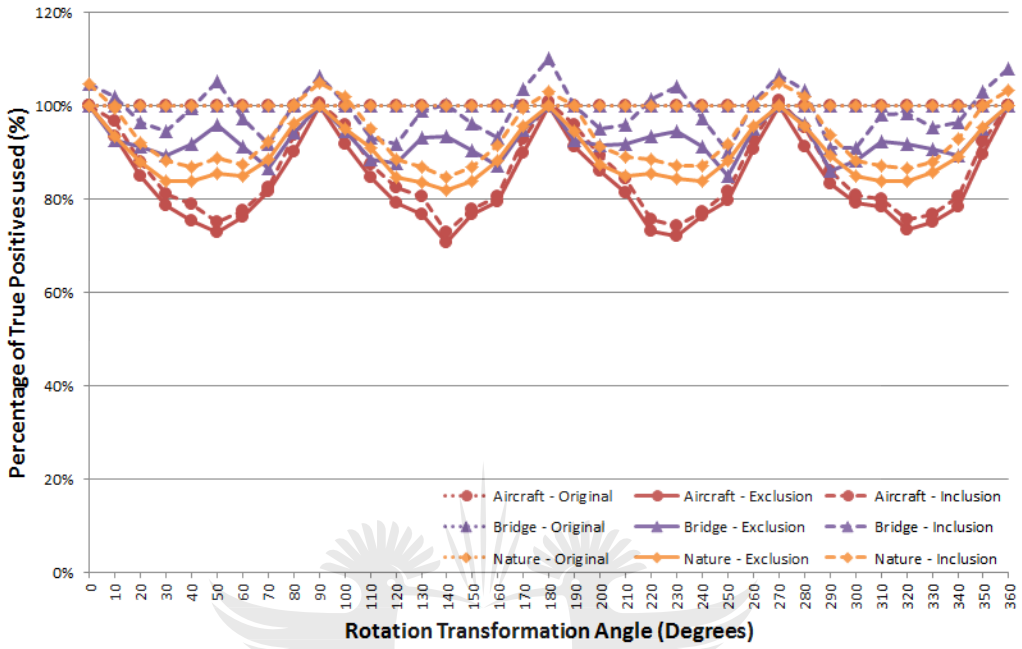


(a) Number of Features

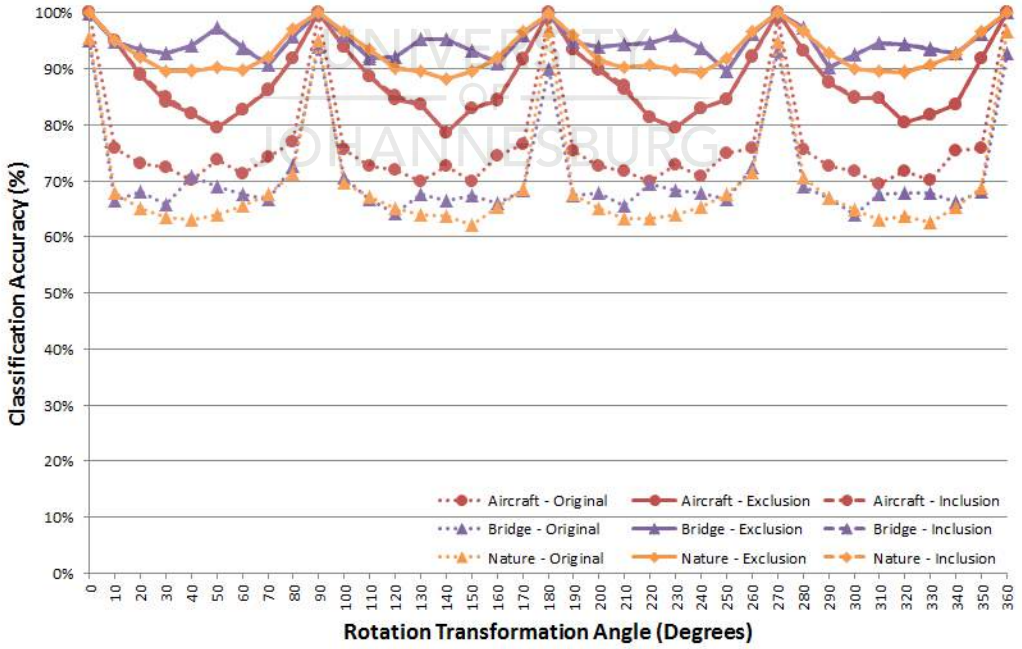


(b) Matching Accuracy

Figure 5.17: The number of features and matching accuracy results obtained by the MSER based Super-features algorithm for Rotation transformations



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.18: The percentage of True-positives used and classification accuracy results obtained by the MSER based Super-features algorithm for Rotation transformations

ance ability of MSER was slightly better than SURF but provided less reliable feature matches compared to SIFT. Consequently, this will have a positive effect on the Super-feature's classification accuracy compared to SURF, as less feature match outliers will be present and included in the estimation process. The original MSER features could be matched with an average matching accuracy of 70%, which was improved by the Super-feature algorithm by almost 30% for the majority of the tested rotation angles. As a result, both the Super-feature Exclusion and Inclusion algorithms based on MSER features produced matching accuracy results close to 100% as seen in Figure 5.17.b. Some True-positives were lost as seen in Figure 5.18.a which negatively impacted the classification accuracy provided in Figure 5.18.b. This can be attributed to the 30% outliers rate in the Original feature matches as well as the sparsity of the features detected by the MSER method. The Bridge and Nature image contained larger amounts of features which could improve performance, compared to the Aircraft image which was more sparse. The lack of features in the Aircraft test image reduced the classification accuracy by 10% compared to the other two test images. From this we can observed that in general, more feature matches provide better support which improves the Super-feature algorithm's ability to handle outliers and miss-matches. This will improve the classification and matching accuracy. From this we can see that the Super-features algorithm favours non-sparse feature sets, but do still work and provide an improved matching accuracy even on sparse datasets. Only on the Bridge test image which had the most detected features could the Super-feature algorithm with inclusion and correction sometimes match more True-positives than what was originally detected. The Super-feature algorithm with inclusion and correction still detected more feature matches than the Super-feature with Exclusion algorithm.

5.3.2 Scale transformation invariance

Scale invariance was tested by matching the features located in the original untransformed image to the features located in the scale transformed images ranging from half-size to double the original size images. Example frames of the image scale transformations applied to the Bridge test dataset image can be seen in Fig-

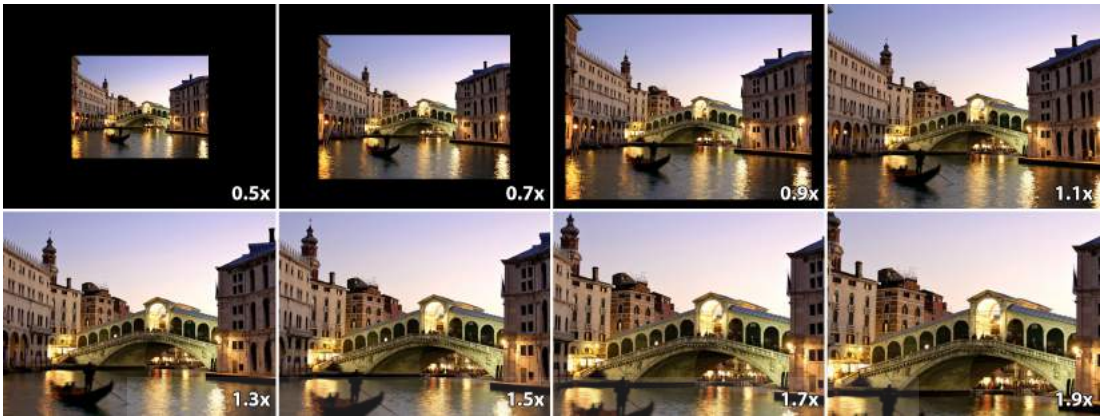
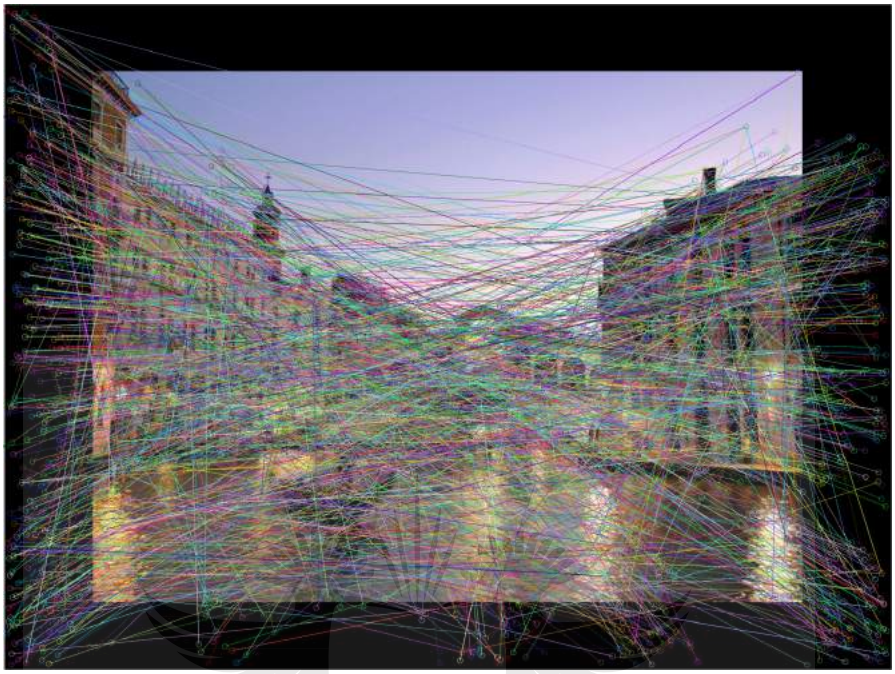


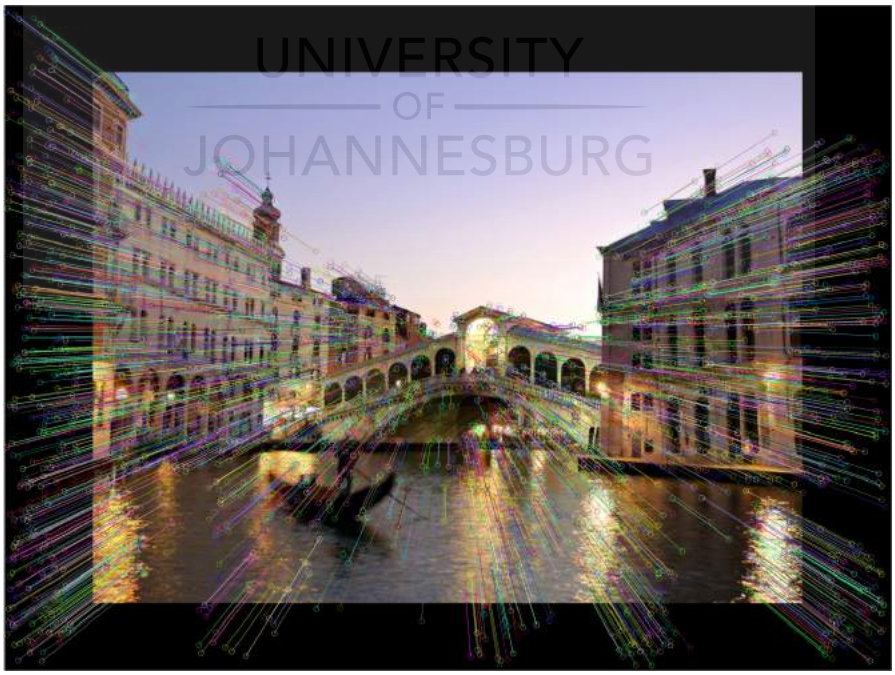
Figure 5.19: Example images of the Bridge scale dataset

ure 5.19. As can be observed, some of the image features are occluded and lost because they are transformed outside the image borders on the magnification tests. The three feature detectors SIFT, SURF and MSER and their Super-feature variants will be tested on the scale transformed versions of the three test images Nature, Bridge and Aircraft to determine the Super-feature's ability to handle scale transformations.

The Scale invariance test results obtained for the SIFT feature detector performed on the test images can be seen in Figure 5.21 and Figure 5.22. In Figure 5.20, an example of the feature matching results of the SIFT and SIFT based Super-feature algorithm can be seen. As the scale decreases the number of features detected by SIFT remains relatively similar but as the scale is increased the number of features decrease. This is because this synthetic dataset does not include new high frequency features as the scale is increased, some feature is also lost due to occlusion on the image borders since some image regions are transformed outside the usable image region. The original feature matching accuracy of SIFT was very similar between the three test images, SIFT struggled to handle smaller scales with the matching accuracy dropping rapidly to as low as 20% and 30% for matching of features to a half-size image. A scale increase was not as severe as scale decrease with the matching accuracy remaining at about 75%. The results improved dramatically after the Super-feature algorithm with Exclusion of poor features and Inclusion of corrected features was applied. As a result, the

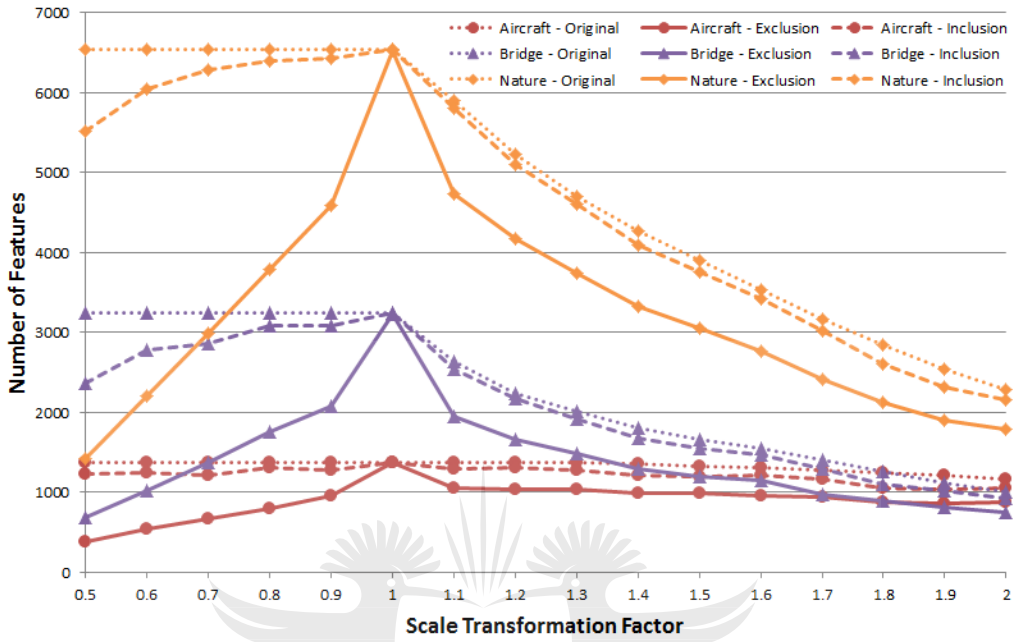


(a) Original SIFT matching results

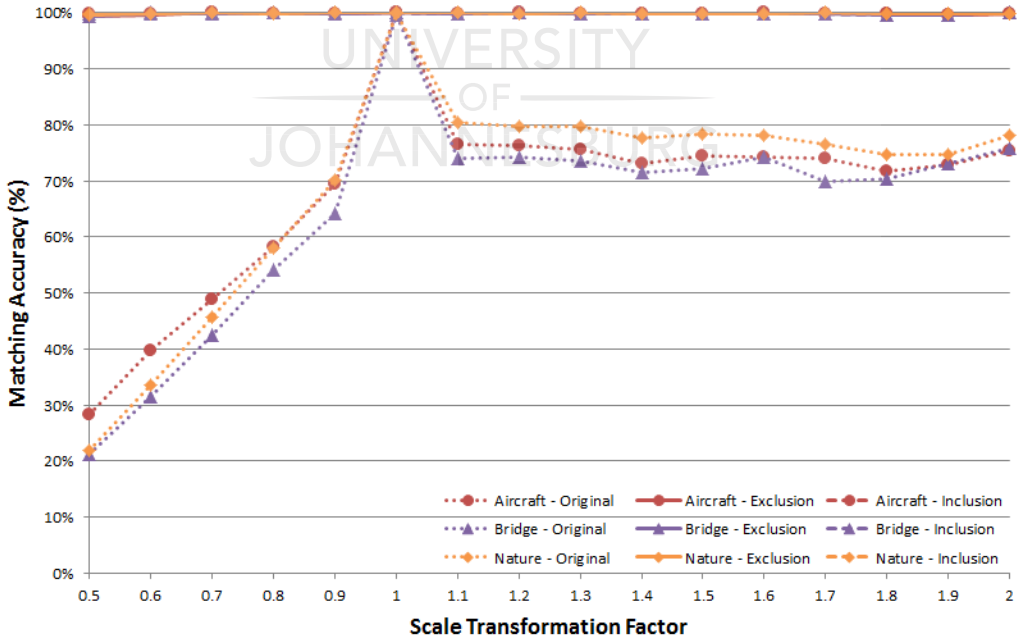


(b) SIFT based Super-feature matching results

Figure 5.20: A comparison of the feature matching results of 0.8x Scale transformed features by the SIFT and Super-feature algorithm

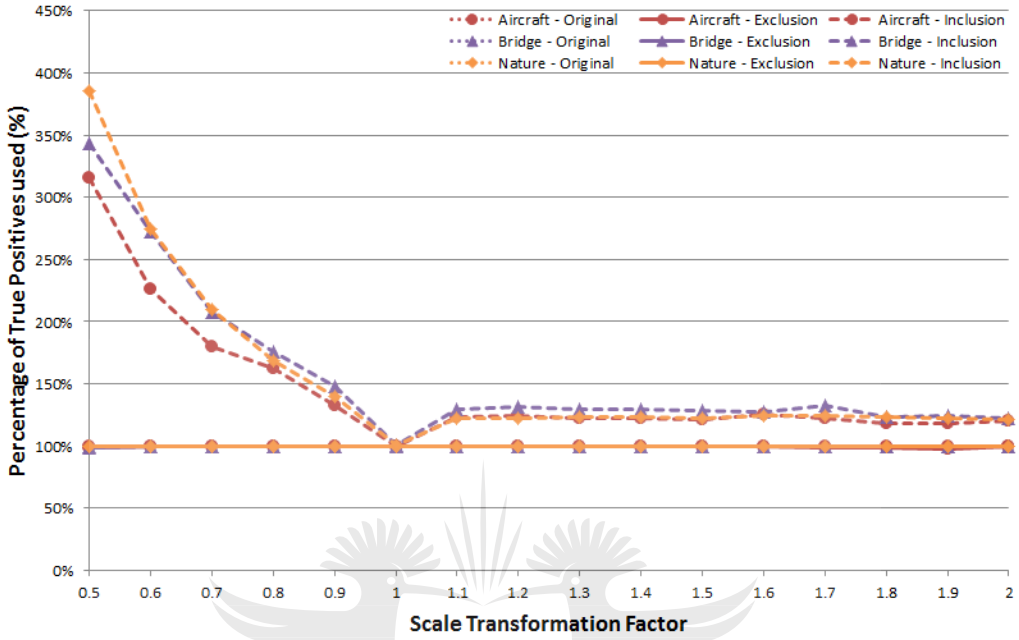


(a) Number of Features

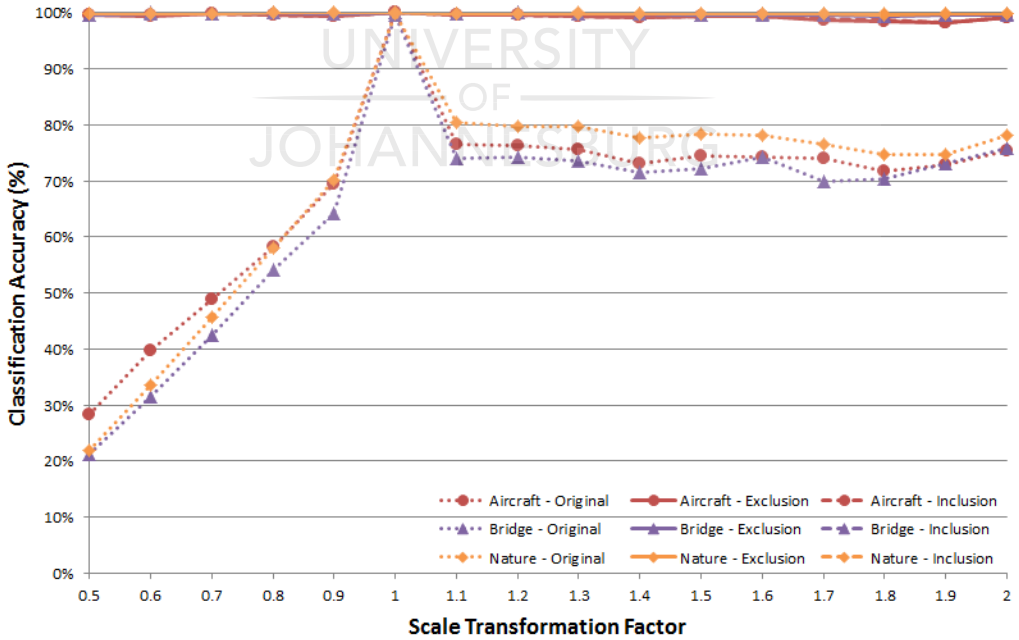


(b) Matching Accuracy

Figure 5.21: The number of features and matching accuracy results obtained by the SIFT based Super-features algorithm for Scale transformations



(a) Percentage of True-positives used



(b) Classification Accuracy

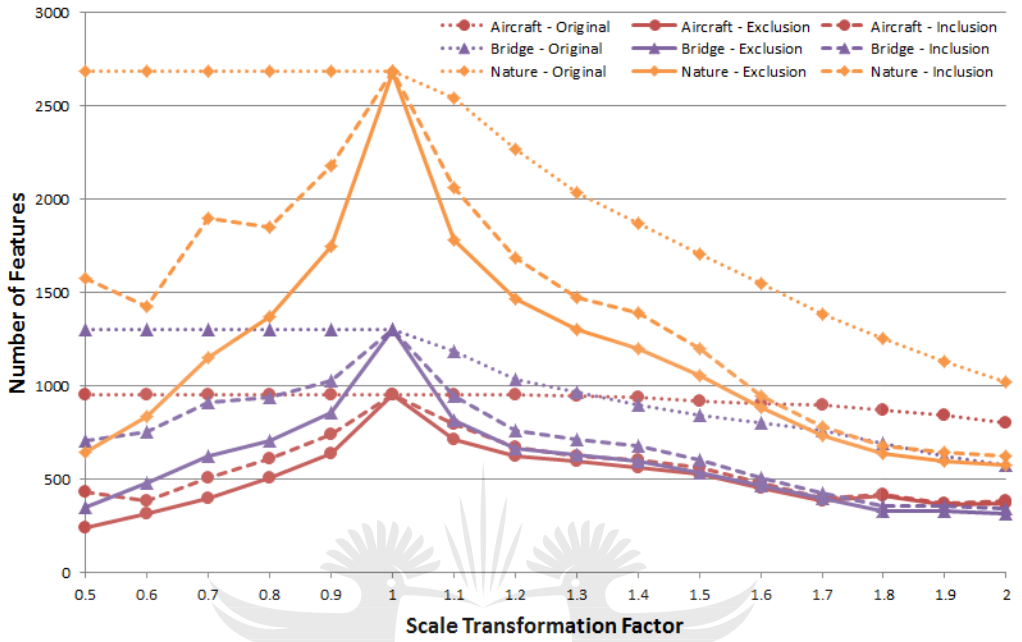
Figure 5.22: The percentage of True-positives used and classification accuracy results obtained by the SIFT based Super-features algorithm for Scale transformations

matching accuracy of both methods remained close to 100% for all tested scale transformations as seen in Figure 5.21.b.

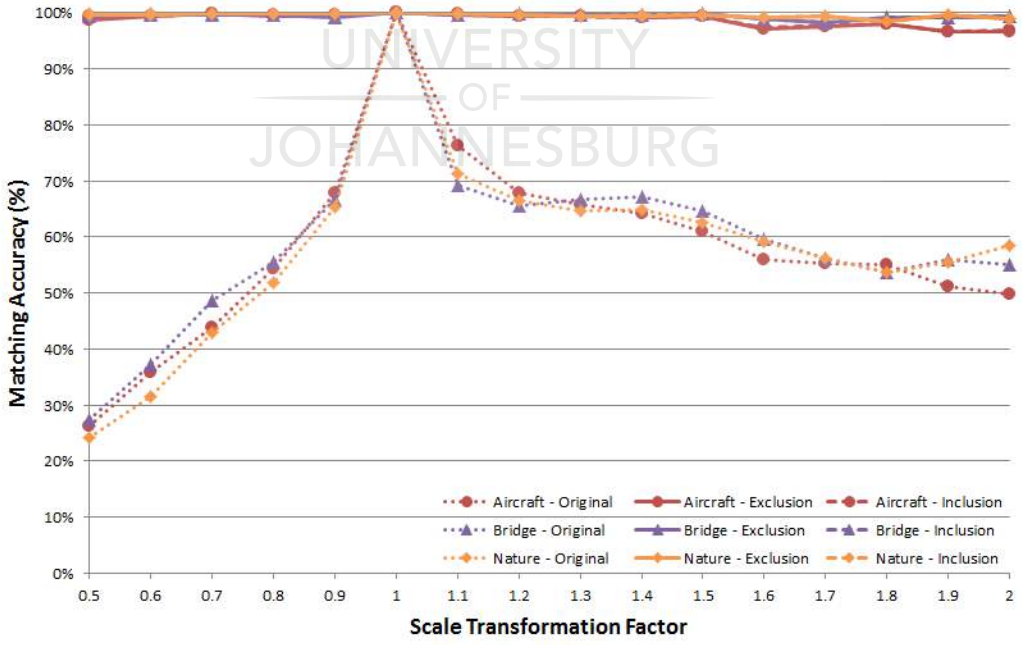
The Super-features Exclusion algorithm had almost a 100% classification accuracy and was able to detect almost all of the True-positive features as can be seen in Figure 5.22. Super-features Inclusion was able to detect 25% more features than the Super-feature Exclusion algorithm for larger scale increases. Matching on decreased scales suffered from poor matching accuracy, this allowed the Super-feature algorithm to correct a larger number of incorrectly matched features. It can be observed that there was a steady increase in the number of True-positives used by the SIFT based Super-features Inclusion algorithm as the scale was decreased. A good result that can be seen from these tests are that the classification accuracy remained constant over the whole range of tested scales.

The SURF based Super-features algorithm provided similar results to the the SIFT based feature detector. As previously observed in the Rotation transformation tests, the SURF feature detection algorithm did not provide feature matches that were as reliable as the features produced by SIFT. Consequently, more outliers were present during the estimation process of the SURF based Super-feature algorithm making it more difficult to classify and correct bad feature matches. The original SURF feature matches showed a steady decrease in matching accuracy similar to SIFT, as the scale was decreased, a matching accuracy of only 25% was obtained for matching feature to the half-resolution image as seen in Figure 5.23. An increase of image scale also resulted in a slight decrease of matching accuracy for the SURF algorithm, SIFT was better able to handle scale increases and its matching accuracy remained relatively constant.

The Super-feature Exclusion algorithm as-well as Super-features Inclusion algorithm improved the original matching accuracy by 35% to 75% for decreased scales and an improvement of 35% to 45% was observed for increased scales. The Super-features algorithm had more difficulty with larger scale increases when it was applied to SURF features, overall the maximum decrease of matching accuracy of only 3% was observed. This is because the SURF algorithm provided poor feature matches for these scales as well as sparse feature sets compared to the SIFT algorithm on similar scales. A dens set of features with a similar percentage of outliers, compared to a sparse set of features can provide more support to the

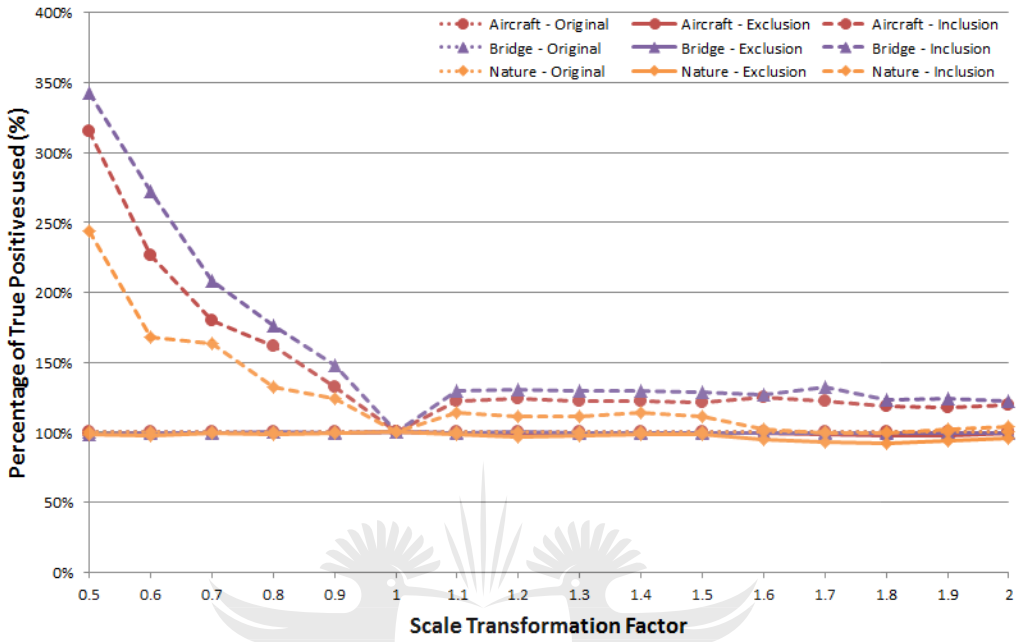


(a) Number of Features

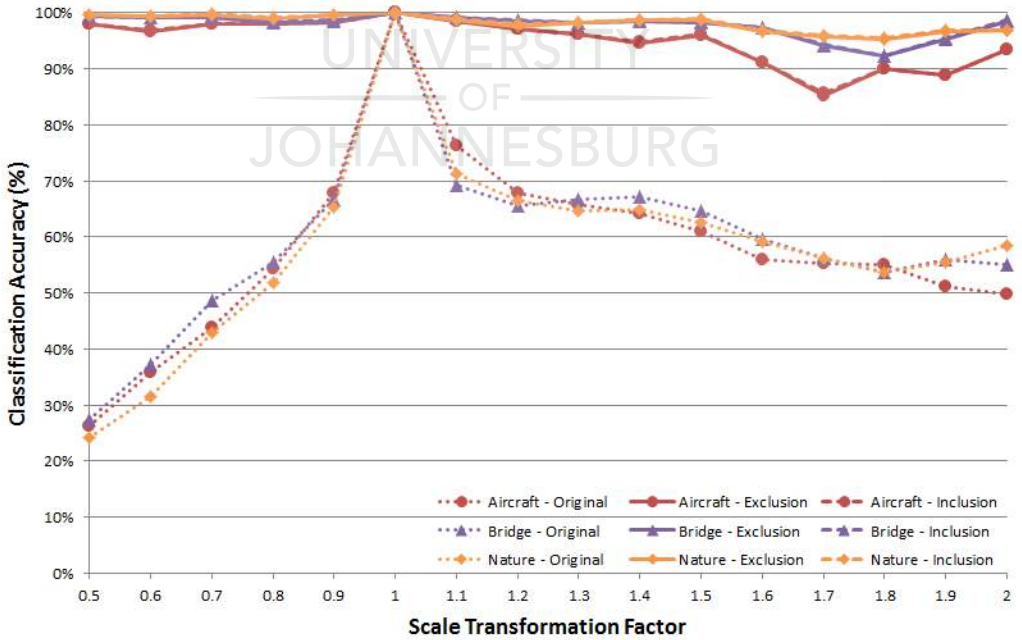


(b) Matching Accuracy

Figure 5.23: The number of features and matching accuracy results obtained by the SURF based Super-features algorithm for Scale transformations



(a) Percentage of True-positives used

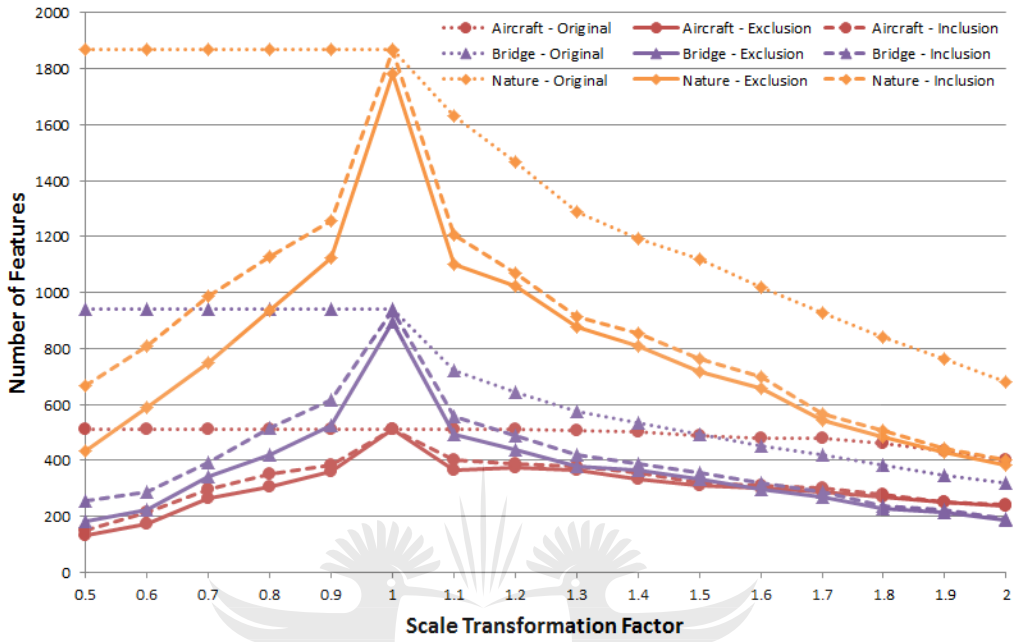


(b) Classification Accuracy

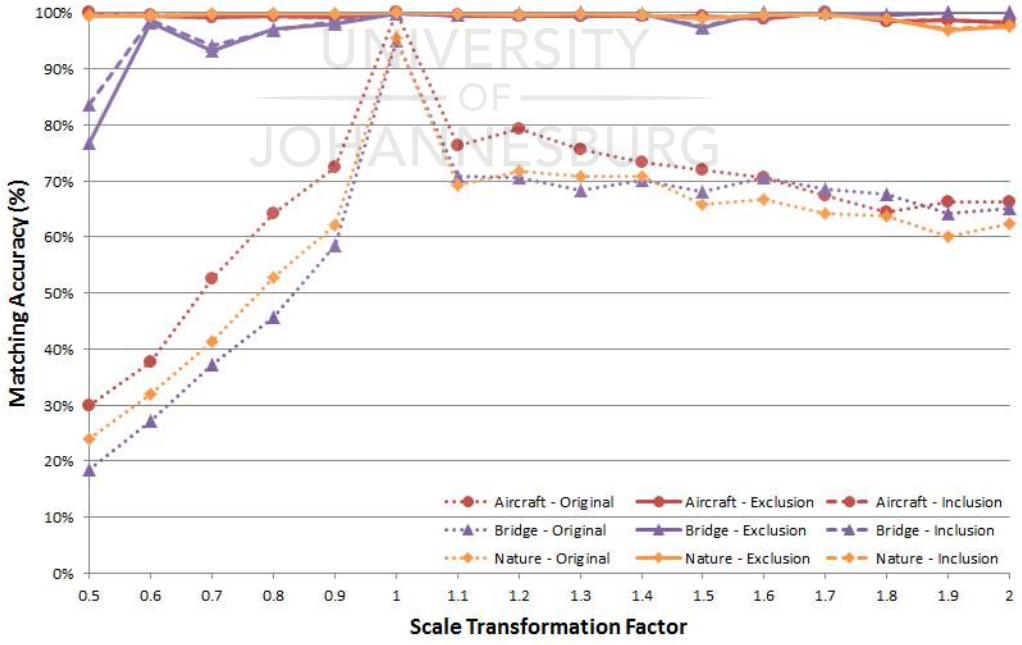
Figure 5.24: The percentage of True-positives used and classification accuracy results obtained by the SURF based Super-features algorithm for Scale transformations

features matches which will improve the classification accuracy. As seen in Figure 5.24.b the classification accuracy of the Super-feature algorithm on increased scales were negatively impacted be the reduced number of features detected and the increased number of mismatches present. The SURF based Super-features Inclusion algorithm was able to detect a large amount more True-positive features than the original SURF method for the different tested scale transformations as seen in Figure 5.24.a. It had some difficulty with the large scale increases but still provided superior results compared to the original SURF algorithm.

The MSER feature detector provided better matching accuracy results than SURF and it provided similar matching accuracy results to SIFT but with a reduction to the number of features detected. It detected about three times less features than the SIFT method, which could negatively affect the reliability of the MSER based Super-feature algorithm. The Super-feature algorithm favours a dense set of features compared to the sparse feature sets provided by the MSER feature detector on natural photographs. The MSER based Super-feature algorithm provided consistent results on the Aircraft and Nature test images over the entire range of tested image scale transformations. Invalid feature matches caused by duplicate feature clusters introduced some problems on the Bridge test image as can be observed by the decrease in matching accuracy observed on some scales as seen in Figure 5.25.b. A denser set of features could have provided some variation between the duplicate feature clusters allowing the Super-feature algorithm to distinguish better between them, resulting in the improvement of the matching results. Overall, the Super-feature Exclusion algorithm as well as the Super-features Inclusion algorithm provided a dramatic improvement over the traditional MSER algorithm by providing a classification accuracy above 90% for the majority of the tested scales as seen in Figure 5.26.b. Unfortunately, some True-positives were misclassified by the Super-feature algorithm due to the poor support provided by the sparse feature set as seen in Figure 5.26.a. Consequently, the Super-feature Exclusion algorithm missed some True-positives while the Super-feature Inclusion algorithm detected more True-positives compared to the Original algorithm for most of the tested image scale transformations, except for the largest increased scales.

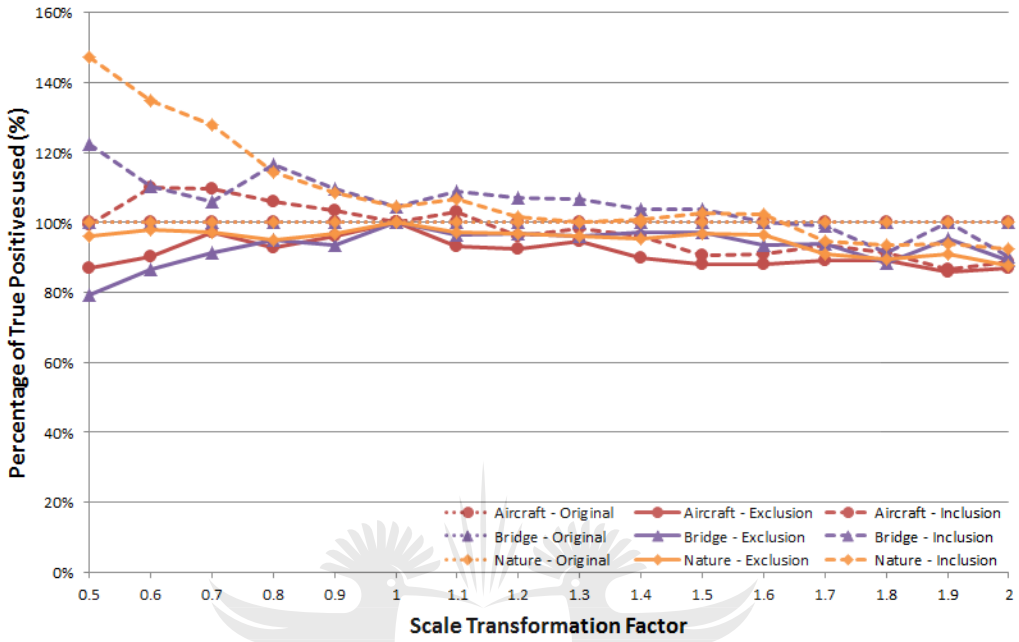


(a) Number of Features

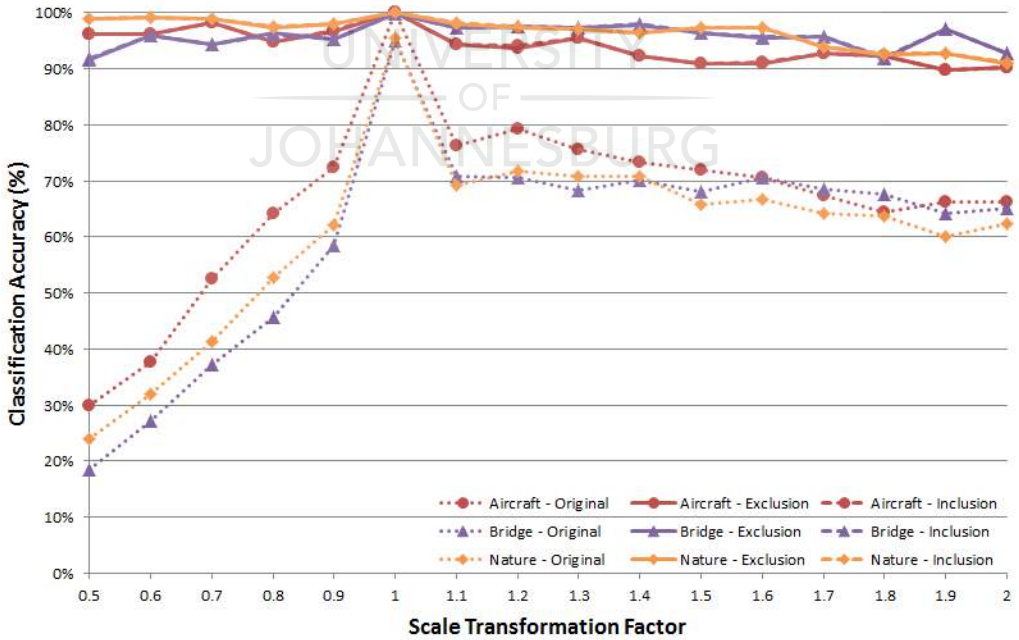


(b) Matching Accuracy

Figure 5.25: The number of features and matching accuracy results obtained by the MSER based Super-features algorithm for Scale transformations



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.26: The percentage of True-positives used and classification accuracy results obtained by the MSER based Super-features algorithm for Scale transformations

5.3.3 Affine transformation invariance

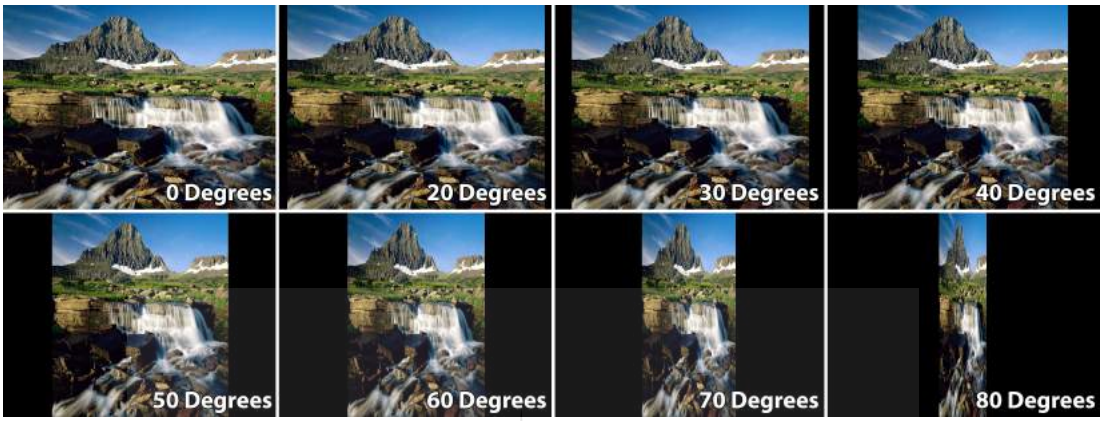
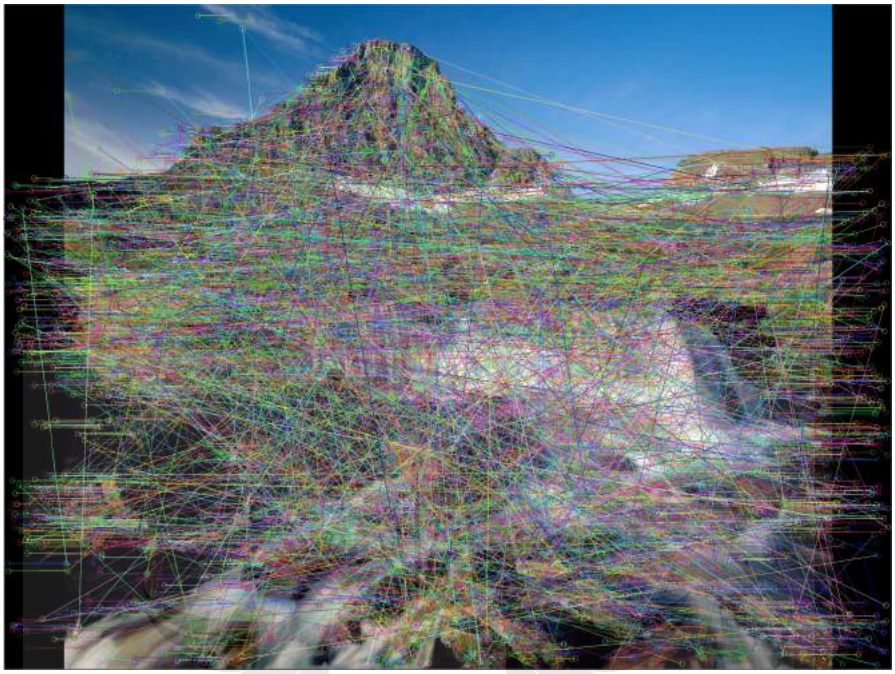


Figure 5.27: Example images of the Nature affine transformation dataset

Since a natural photo captured using a camera projects a 3-dimensional scene onto a 2-dimensional camera plane, affine transformations can be introduced in an image. Affine transformation can distort the local image region surrounding a feature, which makes feature description difficult. The state-of-the-art feature detectors have difficulty matching features when large affine transformations are present, most are limited to extreme affine transformation less then 30 degrees before the accuracy results decline dramatically. Example frames from the synthetic affine transformation Nature dataset is provided in Figure 5.27. Features will be detected in the original untransformed image, these features will then be matched to the features detected in each affine transformed version of the image. This will allows us to determine the Affine invariance capability of the tested feature detector as well as the Super-feature algorithm, and the potential improvement that can be gained by using Super-features in these situations.

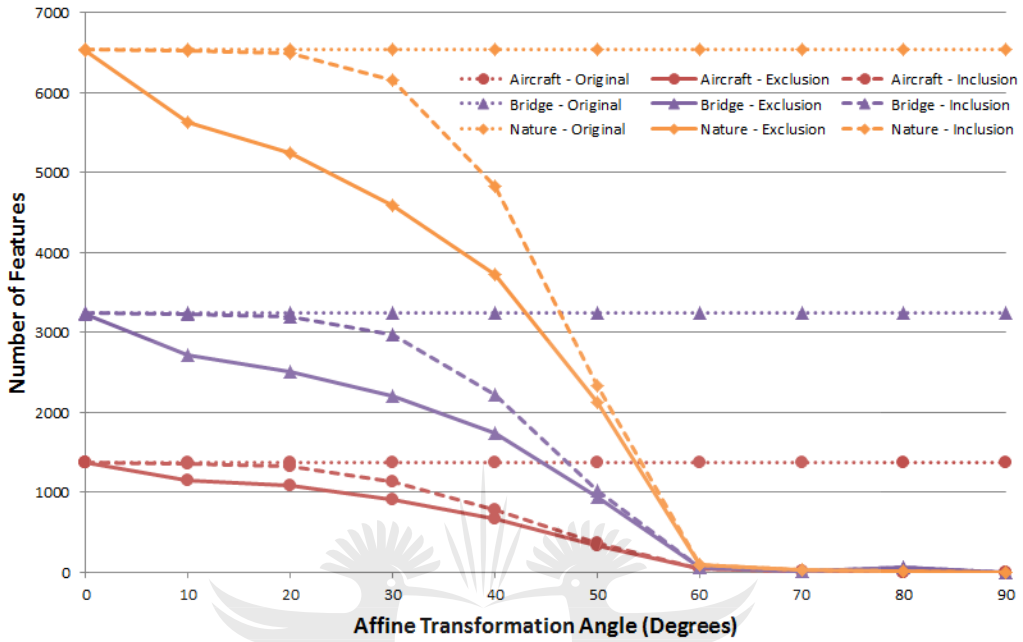
A comparison between the results obtained by SIFT and the Super-feature based SIFT algorithm is provided in Figure 5.28. As the amount of affine transformation is increased the number of detected features should decrease as the image information is compressed to fit into smaller regions, this can be seen in Figure 5.29.a. Very few of the Original features are still present above 60 degree affine transformations, after which the matching accuracy tends to drop quickly.



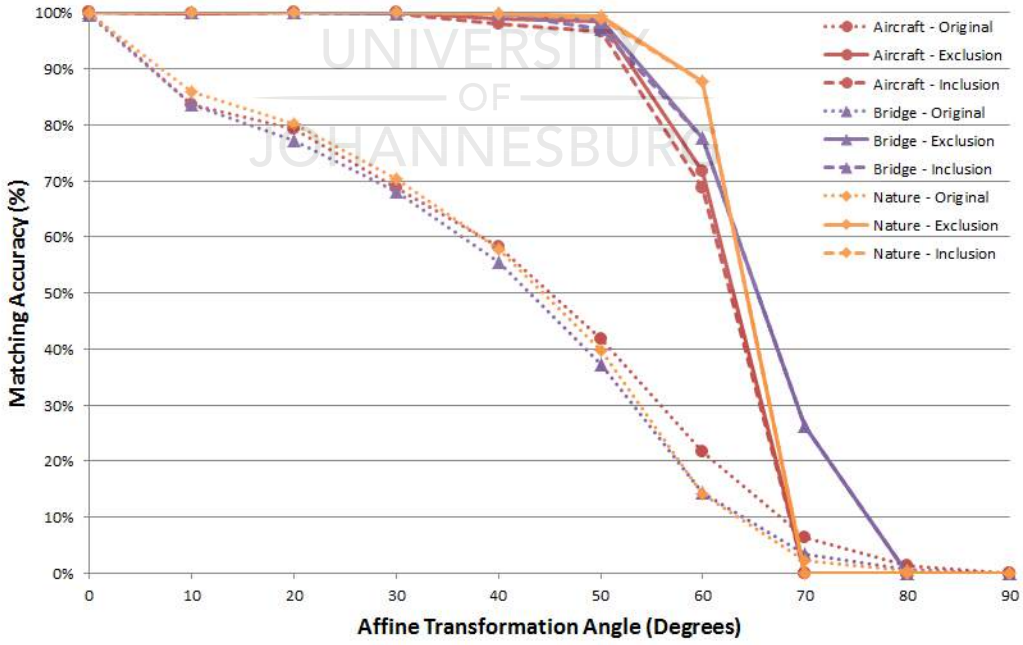
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.28: A comparison of the feature matching results of 30 degree Affine transformed features by the SIFT and Super-feature algorithm

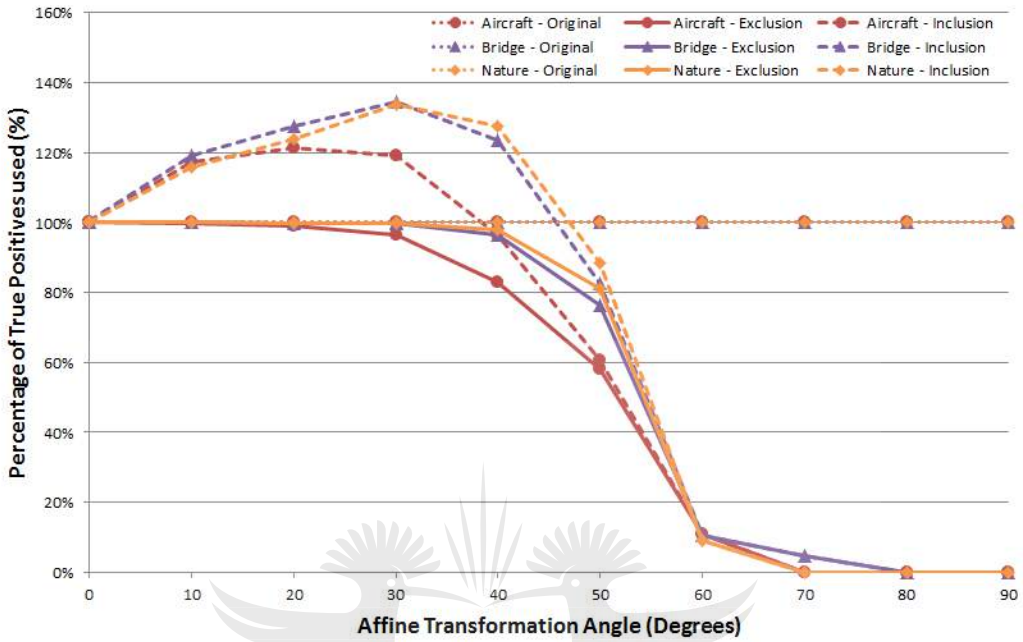


(a) Number of Features

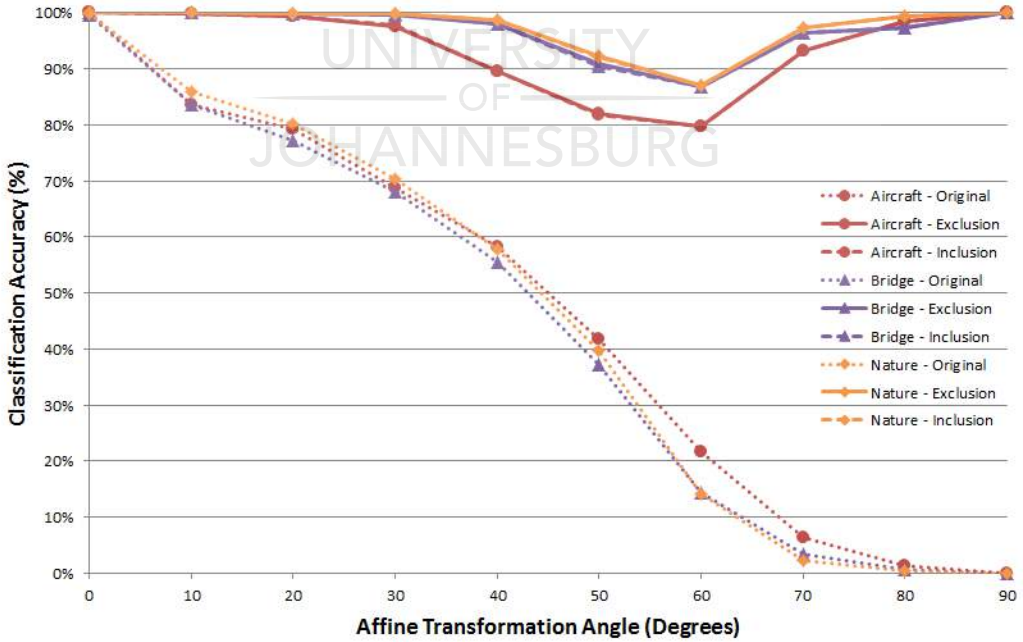


(b) Matching Accuracy

Figure 5.29: The number of features and matching accuracy results obtained by the SIFT based Super-features algorithm for Affine transformations



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.30: The percentage of True-positives used and classification accuracy results obtained by the SIFT based Super-features algorithm for Affine transformations

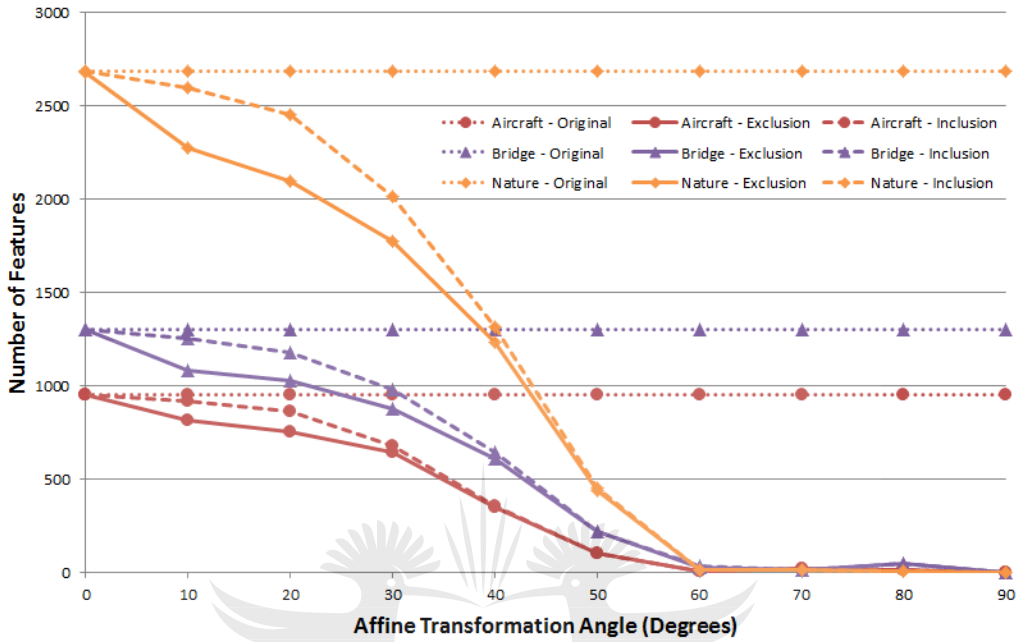
The matching accuracy results for SIFT is provided in Figure 5.29.b. The Original SIFT algorithm provided a very similar decrease in matching accuracy on the Aircraft, Bridge and Nature test dataset images as the affine transformation is increased.

Super-feature Exclusion as well as Inclusion of corrected features provided a matching accuracy of more than 95% for affine transformations of up to 50 degrees at which point the original SIFT algorithm only had a matching accuracy of about 40%, this can be observed for all three test dataset images. For affine transformations larger that 50 degrees, which can be considered as extreme transformation angles, the feature sets become very sparse which reduces the usefulness of applying Super-features. The SIFT based Super-feature algorithm allowed SIFT features to be matched more accurately even on feature sets with severe affine transformation of up to 55 degrees, which is 25 degrees more than what was originally capable by the SIFT algorithm. Almost all original True-positives can be found up to 40 degrees by the Super-features Exclusion algorithm, the Super-feature algorithm with Inclusion of corrected features can find 20% to 30% more features up to this angle, this can be seen in Figure 5.30.a. After 40% a large number of features are lost and the number of True-positives found decrease as well as the classification accuracy as seen in Figure 5.30.b.

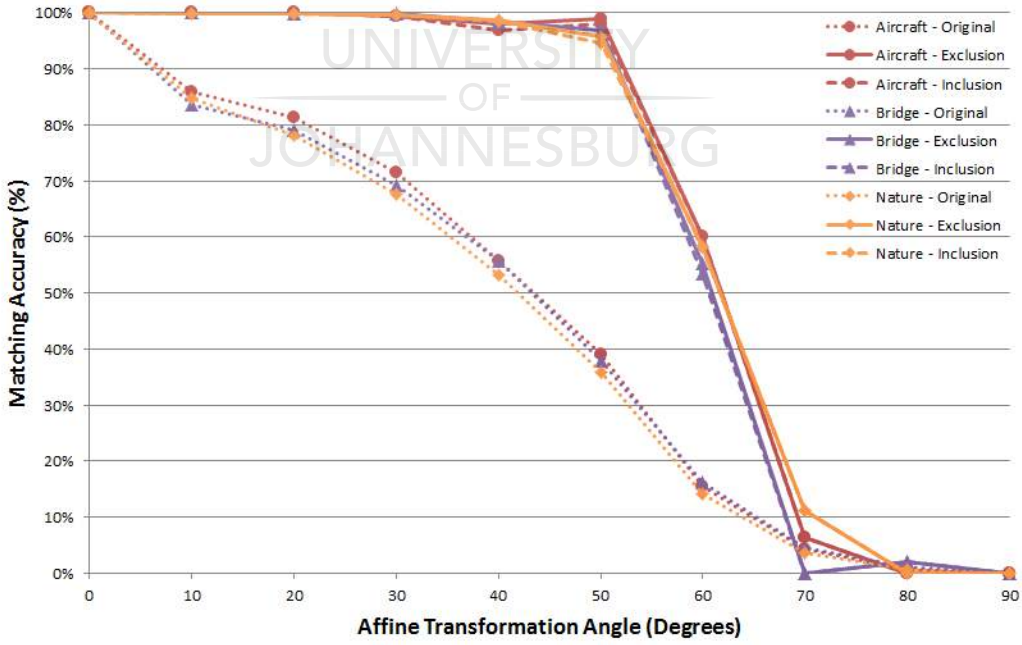
The affine transformation results of the SURF feature detector is provided in Figure 5.31 and Figure 5.32. The affine invariance properties of SURF was very similar to SIFT, there was a constant decrease in accuracy as the affine transformation angle was increased. The number of features detected also decrease with larger affine transformation angles.

Super-feature Exclusion and Inclusion based on SURF provided above 95% matching accuracy for affine transformation angles of up to 50 degrees where the original SURF matching algorithm was only able to maintain 35% to 40% matching accuracy on the three test images as seen in Figure 5.31.b. This is a big improvement which allows accurate matching of features even if extreme affine transformations have been applied. The matching accuracy obtained by SURF based Super-features, up to 50 degrees is very similar to what was obtained using SIFT features, even if less features was detected.

For angles up to 30 degrees a strong increase in the number of True-positive

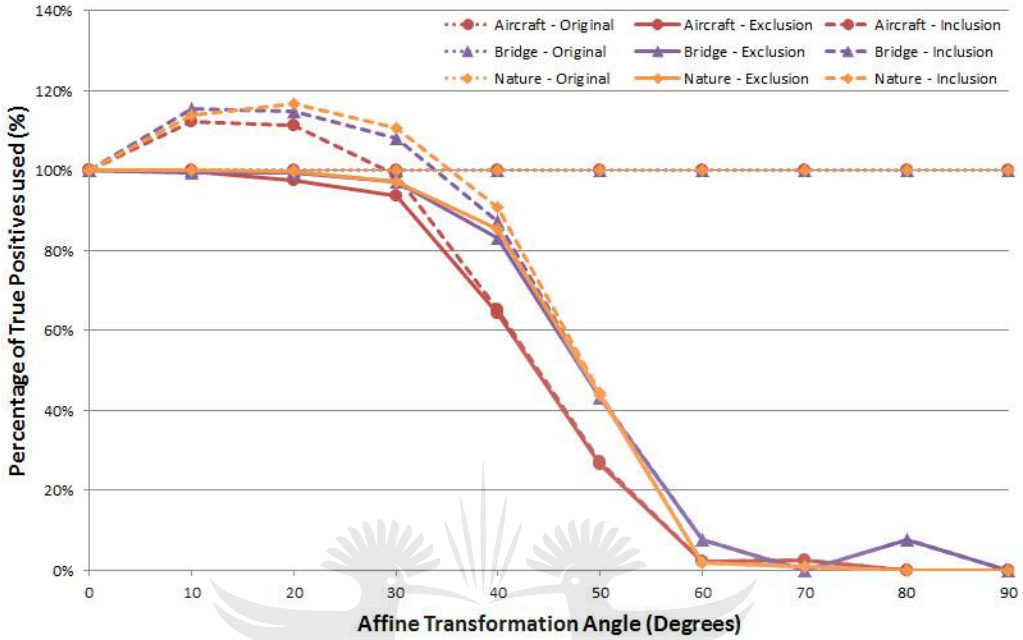


(a) Number of Features

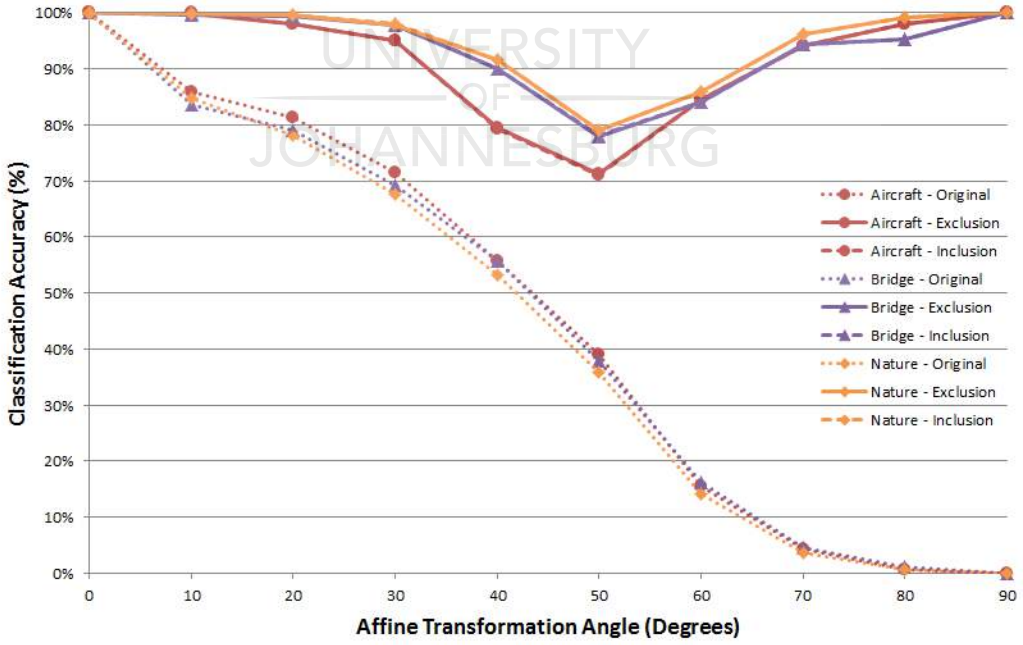


(b) Matching Accuracy

Figure 5.31: The number of features and matching accuracy results obtained by the SURF based Super-features algorithm for Affine transformations



(a) Percentage of True-positives used



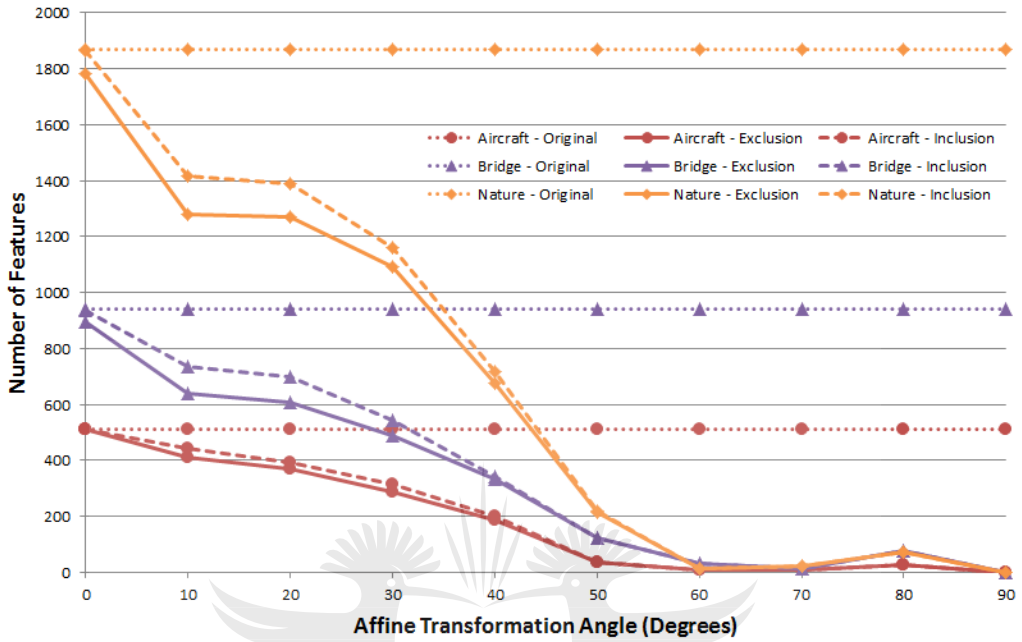
(b) Classification Accuracy

Figure 5.32: The percentage of True-positives used and classification accuracy results obtained by the SURF based Super-features algorithm for Affine transformations

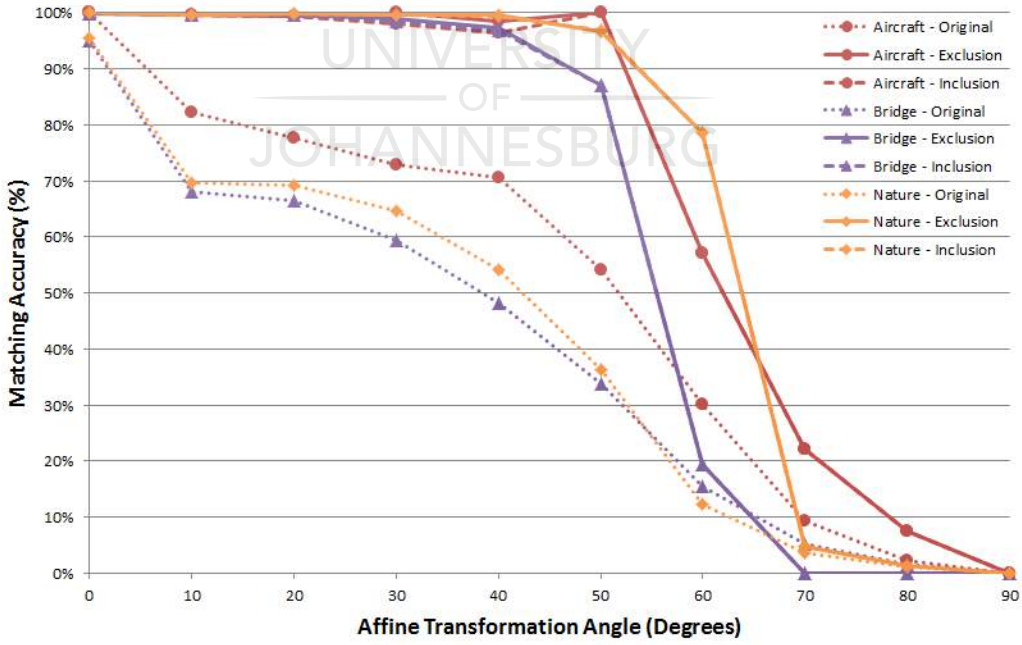
features can be observed for the Super-features with Inclusion algorithm as can be seen in Figure 5.32.a. Only a slight increase in True-positives are observed between 30 degrees and 40 degrees, after which the number of features decrease drastically. As the number of features decrease, it has an impact on the classification accuracy, since the matching accuracy remains high but the classification accuracy decreased for angles between 20 and 60 degrees as seen in Figure 5.32.b. This can be attributed to the fact that some True-positives were misclassified but that most True-negatives were correctly classified and discarded.

The results obtained for the MSER based Super-feature algorithm can be seen in Figure 5.33 and Figure 5.34. MSER features suffered from the same problem that SIFT and SURF struggle with, as the affine transformation angle increased, a large number of features were lost. Usually at about 50 degrees, less then 10% of the original features can still be detected, but these features are difficult to match accurately. At these extreme angles, large amounts of distortion would have been introduced in the feature descriptions which make them less reliable to match.

The affine invariance was increased to 50 degrees for the Aircraft and Nature image, the Bridge test image provided good results up to 40 degrees. The reason for the reduced accuracy exhibited in the Bridge image was due to duplicate feature regions supported by a sparse set of detected features. As seen in Figure 5.33.b, reliable feature matching with an accuracy of above 95% was possible for affine transformation of 40 degrees and smaller. Also, the Super-feature algorithm provided a matching accuracy improvement of between 20% and 30% for affine angles up to 40 degrees. Super-features with Inclusion was able to provide more True-positives compared to the original feature matches provided by MSER as seen in Figure 5.34.a. Due to the large loss of features at extreme affine transformation angles, the classification accuracy is affected negatively compared to the matching accuracy as seen in Figure 5.34.b. Some True-positives that the Super-feature algorithm was unsure of was discard to keep the matching accuracy high at the expense of classification accuracy

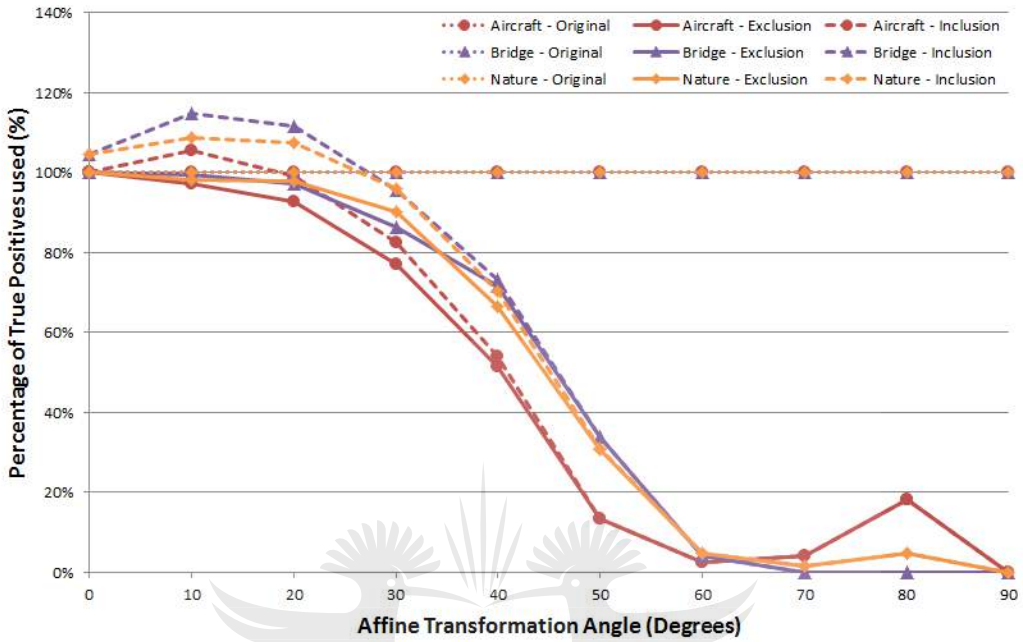


(a) Number of Features

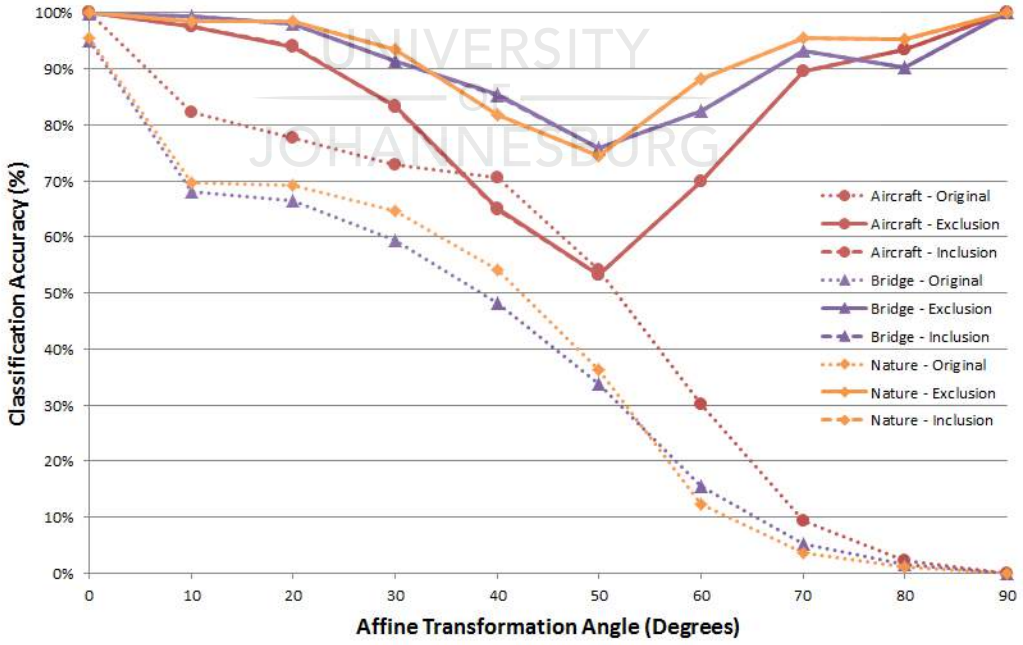


(b) Matching Accuracy

Figure 5.33: The number of features and matching accuracy results obtained by the MSER based Super-features algorithm for Affine transformations



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.34: The percentage of True-positives used and classification accuracy results obtained by the MSER based Super-features algorithm for Affine transformations

5.4 Complex global and local motion transformations

Figure 5.35: Example frames with ground truth motion from the MPI-Sintel dataset [94]

The MPI-Sintel optical flow dataset was used to test the ability of the traditional feature detectors and the Super-feature based detectors to handle complex motion transformations. This dataset is traditionally used to test different optical flow estimation algorithms, but provide sufficiently complex image data and ground truth motion data to allow the testing of feature detectors, example frames can be seen in Figure 5.35. Different feature detectors can be based on different feature detection theories and thus do not necessarily detect the same type of features, different types of features such as blobs, corners or interest points can be detected. This makes testing of these different feature detectors difficult using the same testing framework. An optical flow dataset provides ground truth information on how every local region moves between the different frames in a video sequence. Motion vectors for each pixel is provided allowing different feature detectors based on different feature types and techniques to be evaluated using the same framework.

The MPI-Sintel dataset is a synthetic dataset built from sequences from an animated feature film. Even though the images are synthetic, they exhibit the characteristics and complexity of natural scenes [101]. A rich set of features can be detected in the frames of the MPI-Sintel video sequences which are sufficiently difficult to match. Some of the characteristics, complexities and effects that might be present in the frames of the MPI-Sintel dataset are:

- Dynamic and morph-able objects
- Complex texturing
- Large and fast motion of objects
- Occlusions
- Specular reflections
- Dynamic shadows and illumination changes
- Motion blurring and defocus blur
- Atmospheric effects (blooming, smoke and fog)

The three feature detectors, SIFT, SURF and MSER including their Super-features counterparts was tested using this testing framework. Only feature description information will be used for feature matching, no tracking information was used to improve the matching results. The video sequences of the MPI-Sintel dataset was grouped into 4 categories, determined by the complexity of the feature matching problem for that specific video sequence. The results obtained, including an analysis of the results will be provided for some of the sequences tested in each category.

5.4.1 Dominant scene motion with possible occlusions

5.4.1.1 Mountain1 test sequence

The first set of test sequences does not contain any moving objects except dominant scene motion, some occlusion can occur between motion boundaries. These

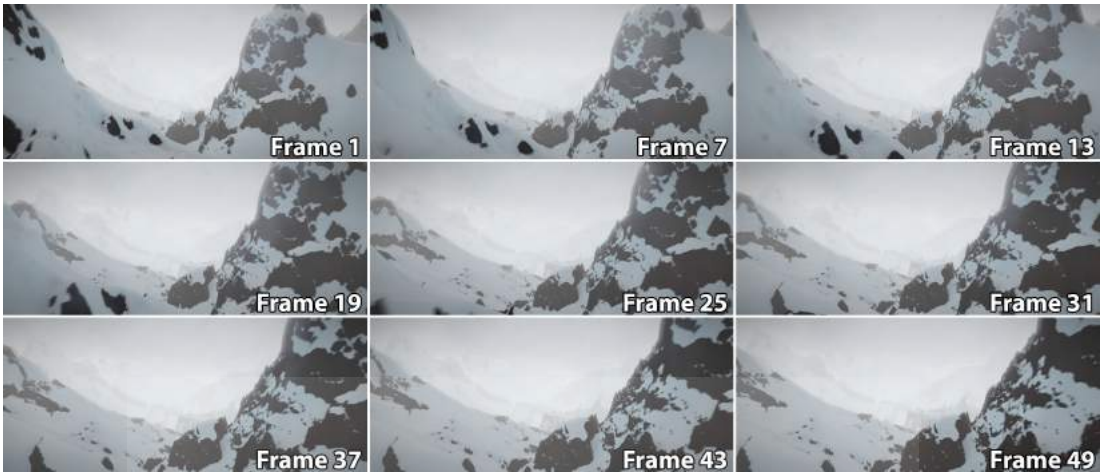
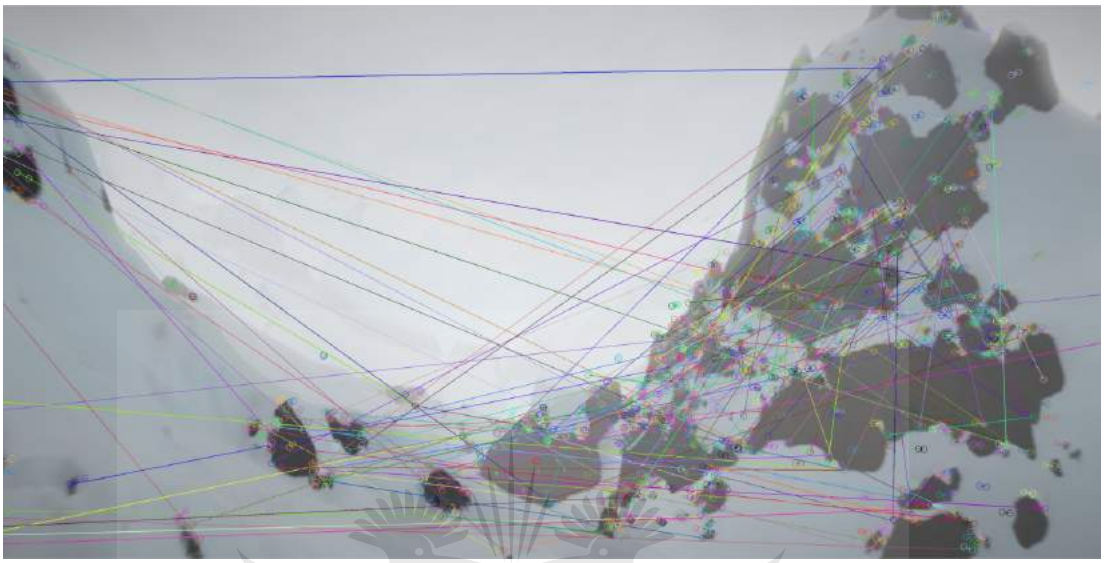


Figure 5.36: Example frames from the mountain1 video sequence

tested sequences will be the simplest of the tested MPI-Sintel sequences, the transformation model is constant over the entire image and can be determined by algorithms such as RANSAC. The Mountain1 sequence contains a large amount of forward motion, but the features are not very discriminant since no complex structures containing colour are present. Example frames extracted from the Mountain1 sequence can be seen in Figure 5.36, some motion blurring can be observed on nearby image regions.

Only a few features were detected by the tested feature detectors as seen in Figure 5.38.a, SIFT detected on average 1200 features, SURF detected 400 and MSER only detected 180. The feature sets were sparse and there were large empty regions located between feature dense regions.

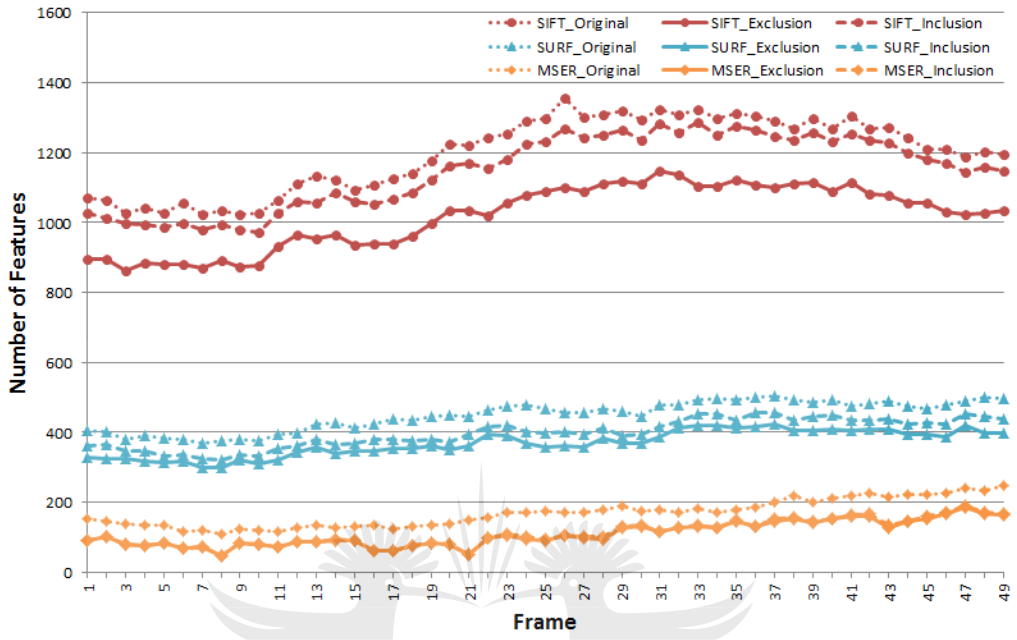
The matching accuracy was relatively constant between the different feature detection algorithms, except MSER had a slight reduction in accuracy at some points in the video sequence. This can be attributed to the small number of features detected by MSER, this will result in less support by neighbouring features used by the Super-feature algorithm. Feature support will also have to be enforced over longer distances which can be more difficult to do reliably. Matching results close to a 100% were obtained by the Super-feature algorithm for all tested feature detectors over the entire video sequence. Super-features provided a matching accuracy improvement of about 15% for SIFT and SURF and it pro-



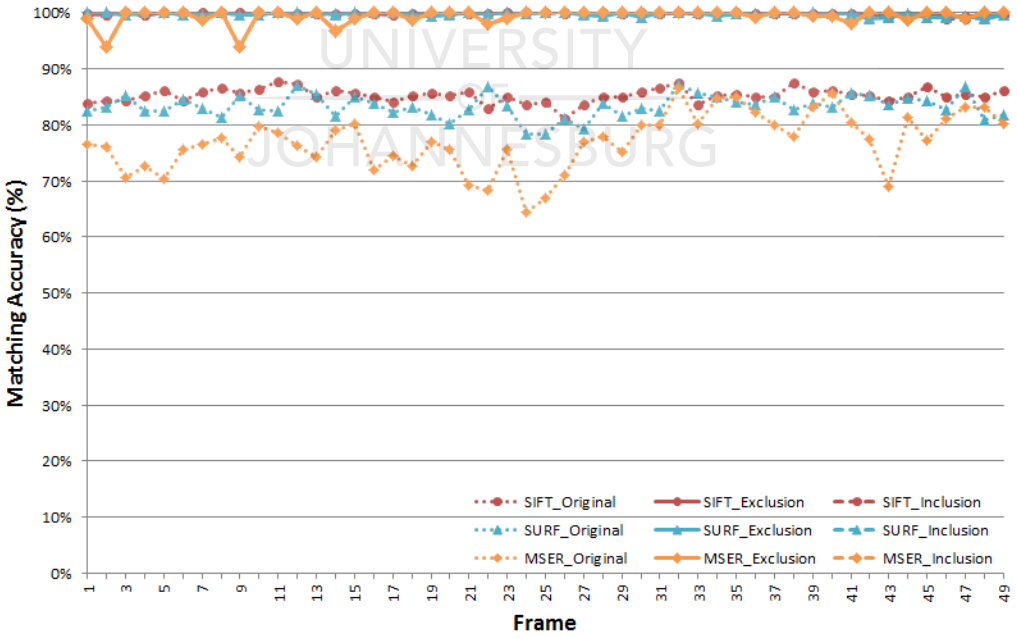
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.37: A comparison of the feature matching results obtained by the SIFT and Super-feature algorithm for frame 1 to 2 of the Mountain1 test sequence

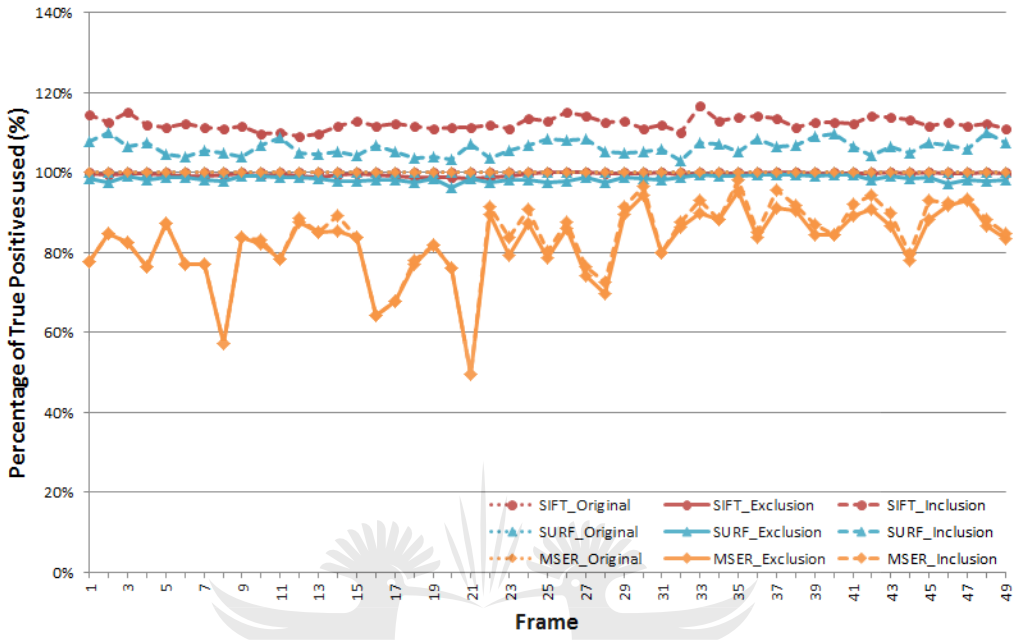


(a) Number of Features

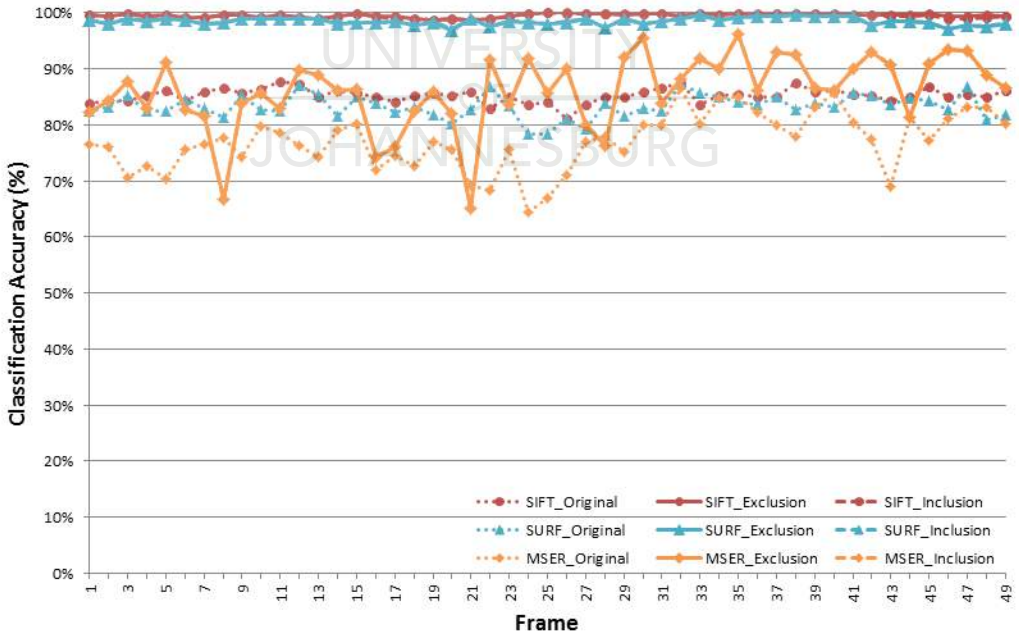


(b) Matching Accuracy

Figure 5.38: The number of features and matching accuracy results for the Mountain1 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.39: The percentage of True-positives used and classification accuracy results for the Mountain1 test sequence

vided a matching accuracy improvement of 25% for MSER, as can be seen in Figure 5.38.b.

SIFT and SURF was able to maintain a classification accuracy of 95% and higher, while MSER suffered due to its limited feature count as seen Figure 5.39.b. It was only able to maintain a classification accuracy of roughly 85% with some dips for some frames in the tested sequence. This was still an improvement over the original results obtained using SIFT, SURF and MSER. The Super-feature algorithm prefer denser sets of features, this can clearly be seen between the results obtained for SIFT and SURF compared to MSER. A comparison of the feature matching quality, between the original SIFT algorithm and the Super-feature based SIFT algorithm can be seen in Figure 5.37. Super-features based on SIFT and SURF provided consistently good results over the entire test sequence while MSER was less reliable. A large number of True-positive features were lost by the MSER based Super-features algorithm as is seen Figure 5.39.a.

5.4.1.2 Sleeping1 test sequence

Example frames extracted from the Sleeping1 dataset can be seen in Figure 5.40. The Sleeping1 test sequence contains a dense set of distinct features, transformed primarily by camera motion of a static scene. There are only small object motions

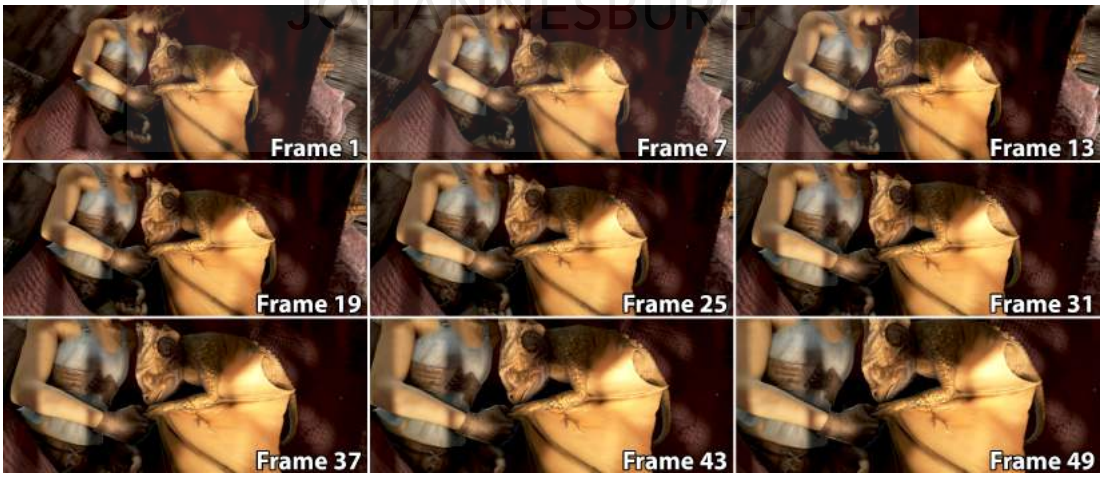


Figure 5.40: Example frames from the sleeping1 video sequence

present in this sequence, the dragon head, wing and arm moves slightly.

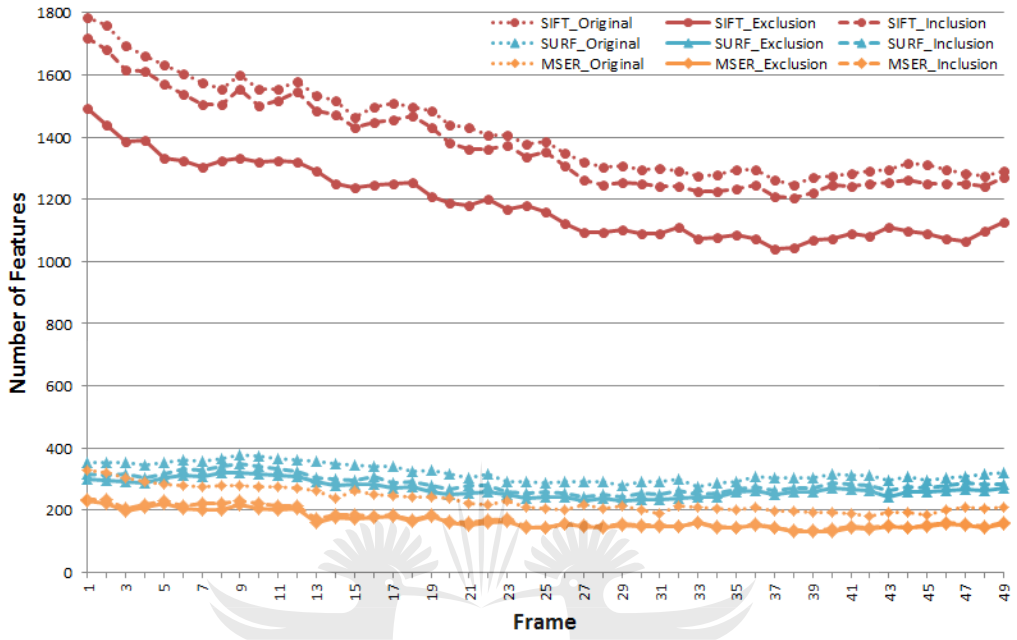
As the video sequence progresses the number of features becomes less as the camera zooms into the scene as can be seen in Figure 5.41.a. SIFT detected 3-6 times more features than SURF and MSER, the low number of features can negatively impact the performance of the Super-features algorithm based on SURF and MSER features.

The matching accuracy results presented in Figure 5.41.b. show that all three tested feature detectors were able to achieve a matching accuracy close to 100%, the MSER based Super-features algorithm was the only feature detector that dropped below 98% accuracy on some sequence frames. This was due to the sparse nature of the feature set detected by MSER. A matching accuracy improvement of 15%-20% was observed by adding the Super-features algorithm compared to the original matching scores. The Super-feature Inclusion algorithm based on SIFT and SURF provided some benefit in recovering mismatched features, more than 100% True-positives were detected. MSER on the other hand detected less than a 100% as seen in Figure 5.42.a.

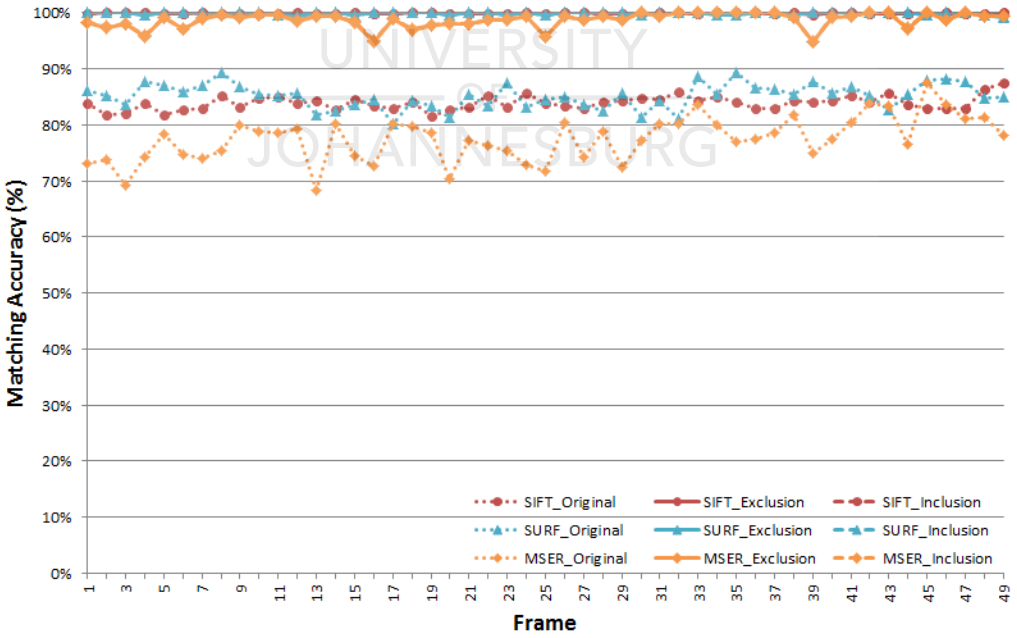
It can be observed in Figure 5.42.b that the more dense the feature set detected, the more likely it will be that the Super-feature algorithm's classification accuracy will be higher. SIFT performed the best performance since it had the most dense feature set while SURF was second and MSER had the worst results.

5.4.1.3 Sleeping2 test sequence

The Sleeping2 video sequence contained only scene motion, but large parts are occluded in this sequence as the foreground scene occludes the background, example frames are provided in Figure 5.43. This scene contained a large number of distinct features which were easily matched, the traditional Feature detection algorithms maintained a matching accuracy above 80% as is seen in Figure 5.44.b. The Super-features algorithm was able to provide some improvement on the matching and classification accuracy while improving the number of True-positives used as seen in Figure 5.45. The SIFT based Super-feature algorithm was able to detect 18% more feature while maintaining a classification accuracy close to 100%.

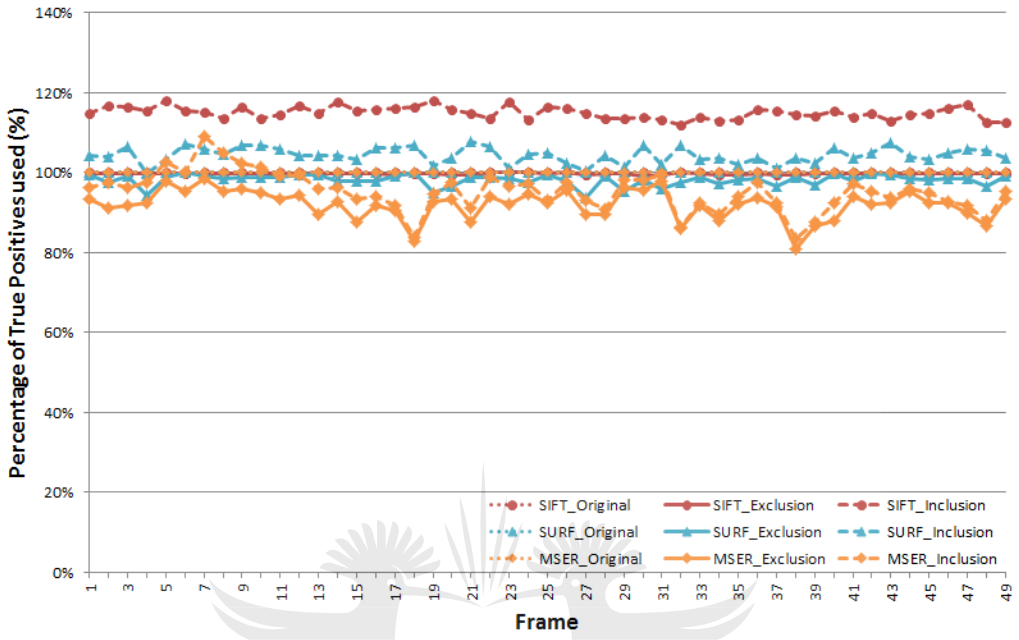


(a) Number of Features

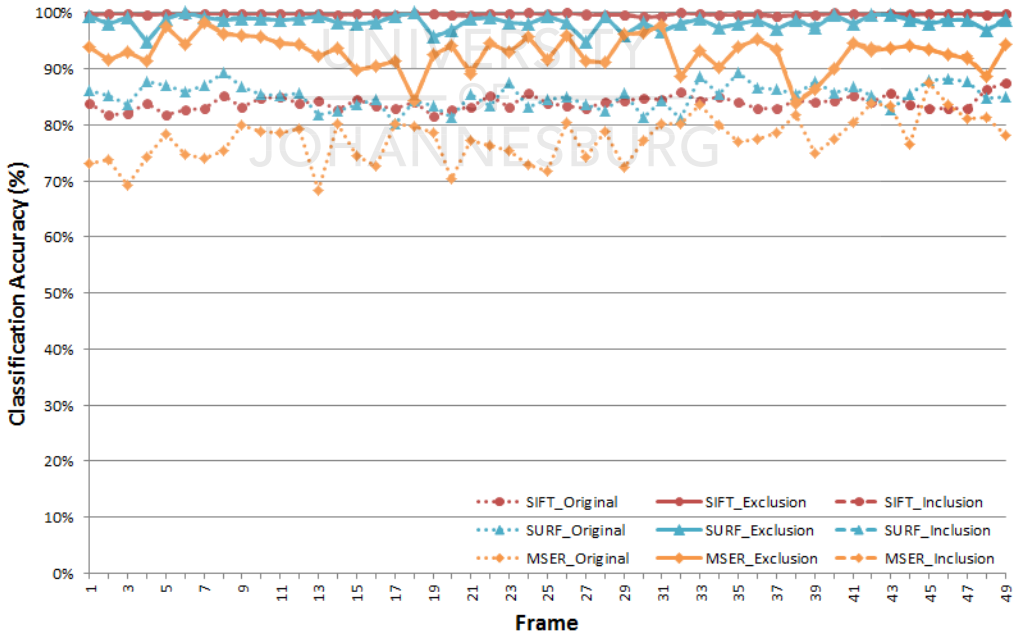


(b) Matching Accuracy

Figure 5.41: The number of features and matching accuracy results for the Sleeping1 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.42: The percentage of True-positives used and classification accuracy results for the Sleeping1 test sequence

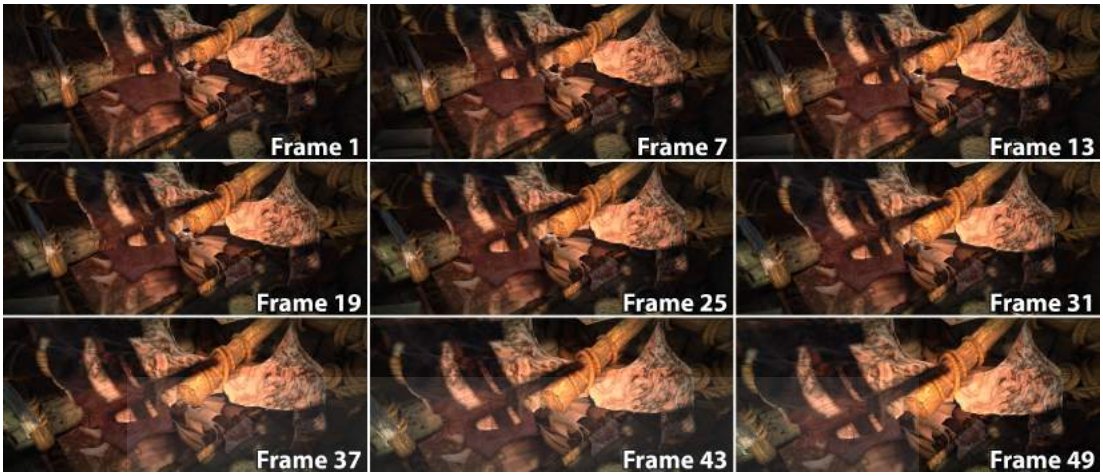
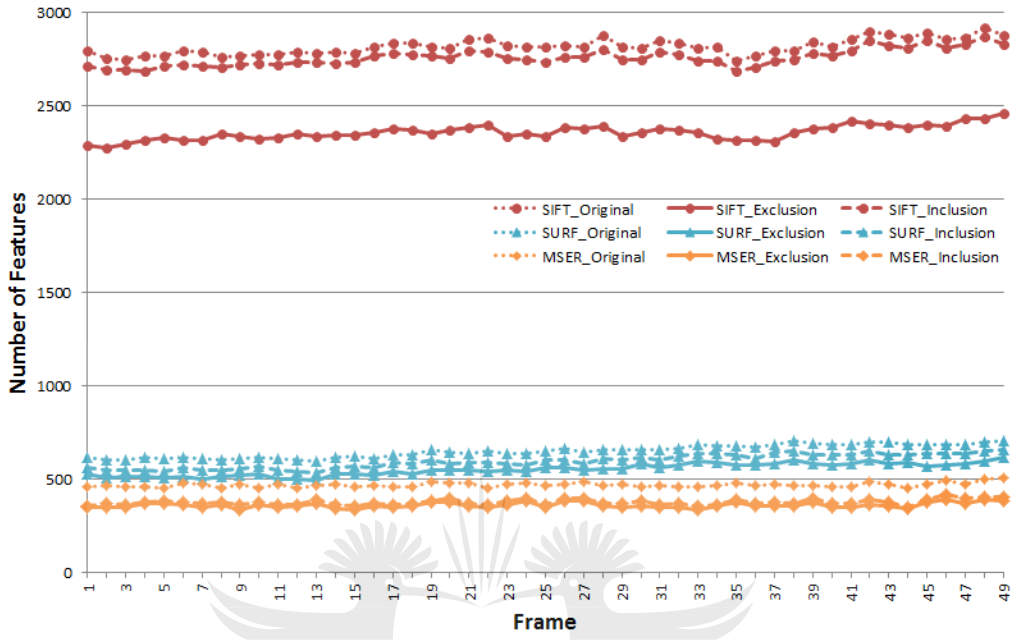


Figure 5.43: Example frames from the sleeping2 video sequence

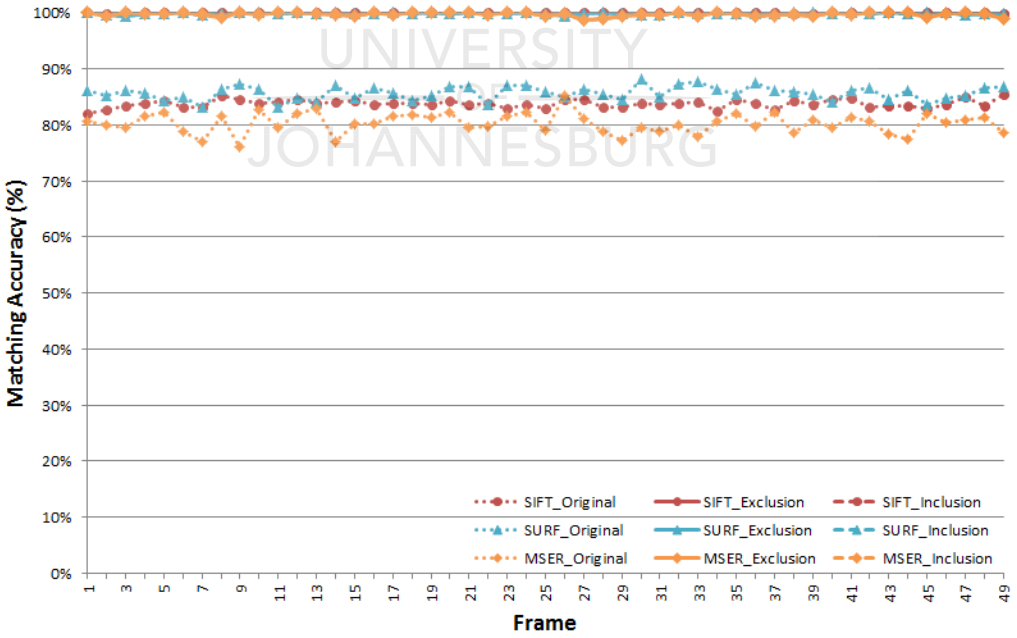
As seen from the results obtained by testing the Super-feature algorithm on the Mountain1, Sleeping1 and Sleeping2 test video sequences. The Super-features algorithm had no difficulty handling features detected in images with dominant scene motion and occlusions. It provided superior results for all three of the tested feature detectors, improving the matching accuracy when small motion was introduced into the video sequences.

5.4.2 Motion present on foreground as well as background

The next level of video sequences that will be used for testing, contain two or more dominant motions. Usually the scene is transformed while some moving objects are present as well. This is a difficult matching problem, as many features located on the borders between motion boundaries can become occluded. The description of these border features can also change as part of the feature's description region lies in an image region that will change due to a different motion being applied to it. The Super-feature algorithm can also potentially have some difficult matching and classifying features when more than one dominant motion is present, since only features detected in a single image are observed at a time by the feature detectors and no prior motion information is encoded. Neighbouring features surrounding a feature can exist on objects that are experiencing different motions

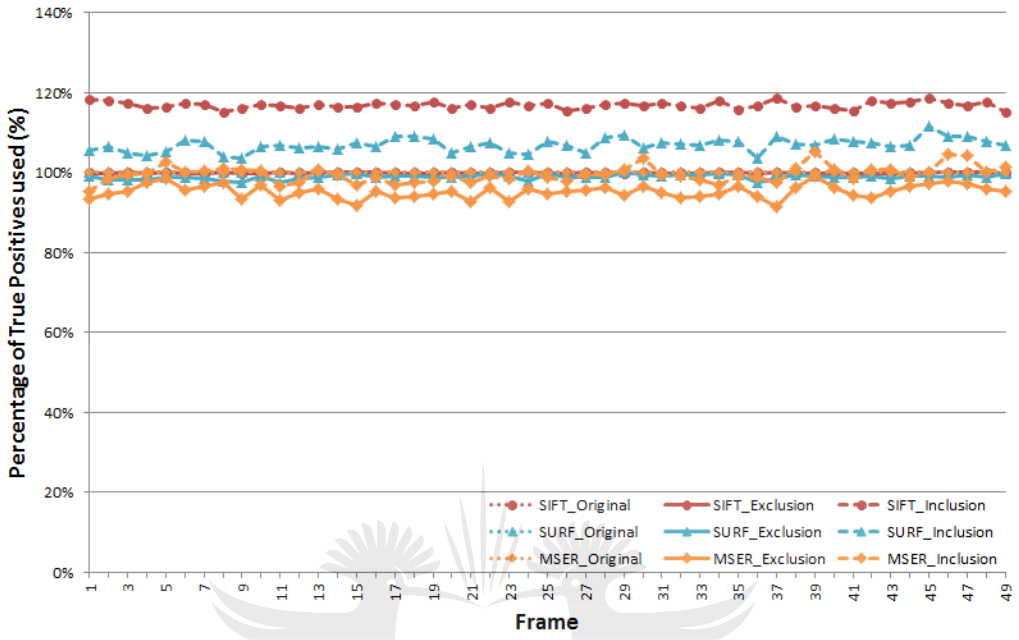


(a) Number of Features

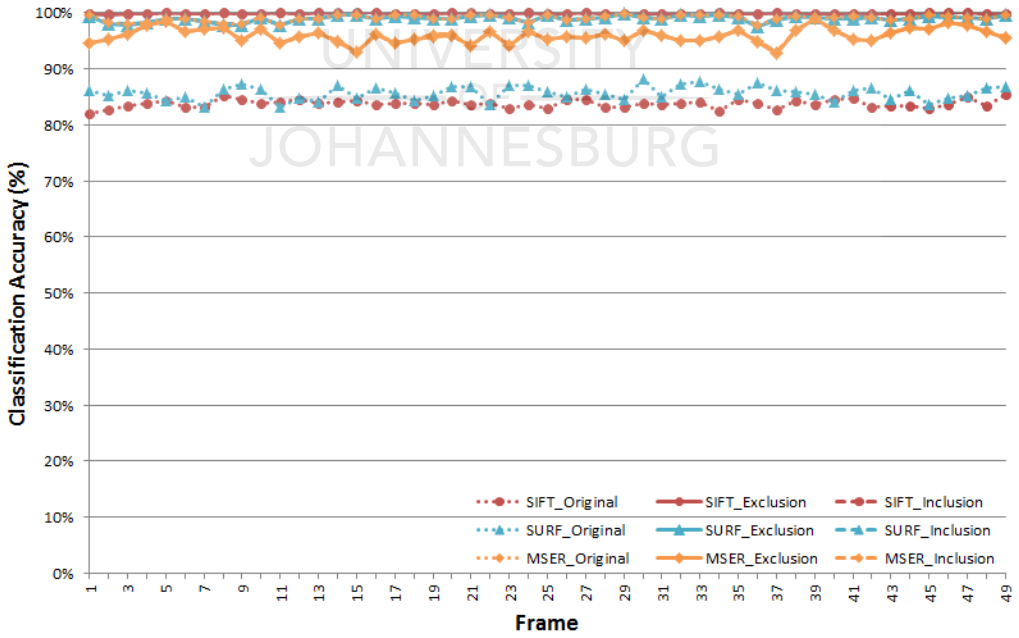


(b) Matching Accuracy

Figure 5.44: The number of features and matching accuracy results for the Sleeping2 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.45: The percentage of True-positives used and classification accuracy results for the Sleeping2 test sequence

and transformations. This will result in a multi-modal solution provided by the estimator.

5.4.2.1 Alley1 test sequence

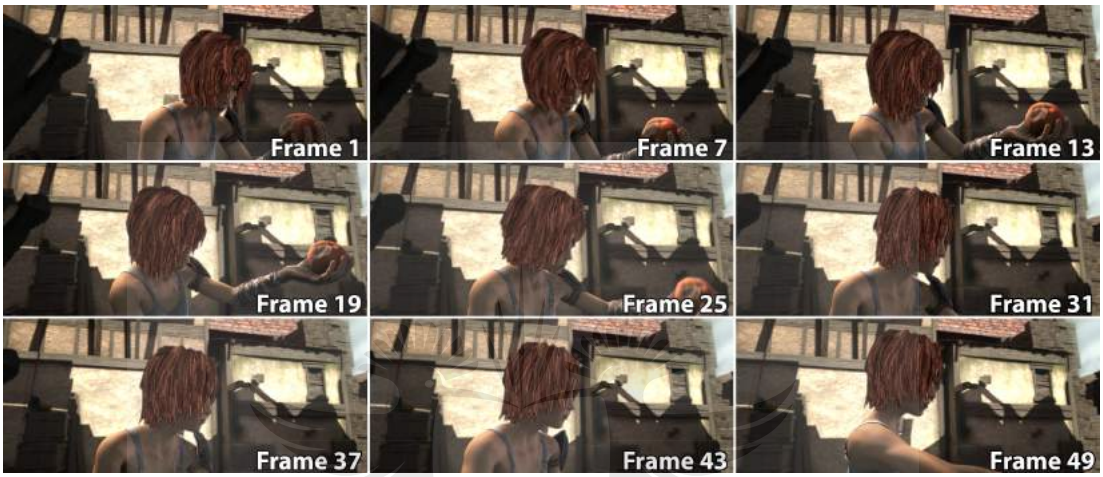
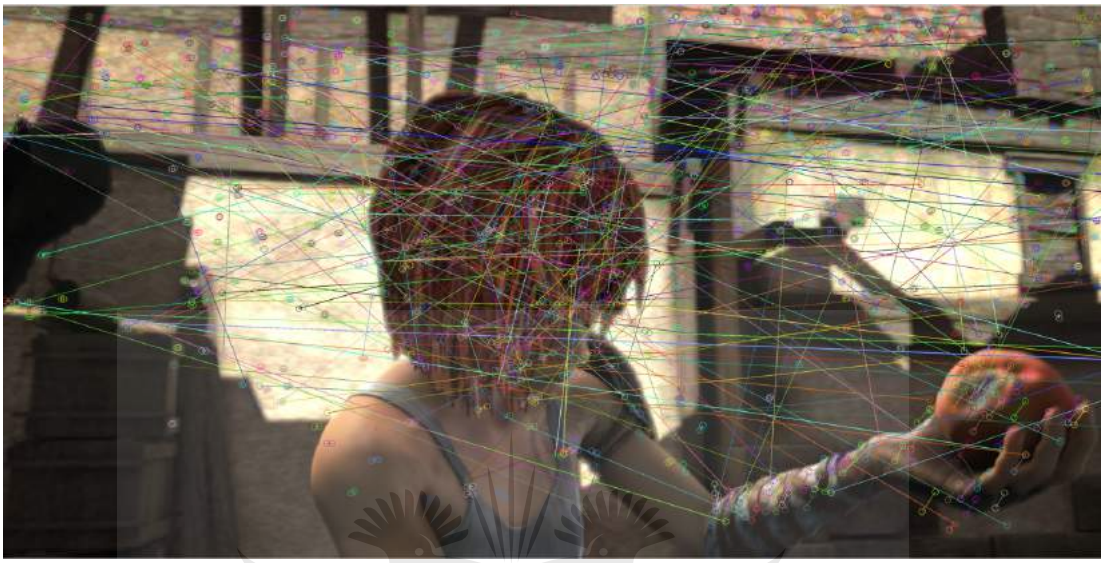


Figure 5.46: Example frames from the Alley1 video sequence

Example frames of the Alley1 video sequence are provided in Figure 5.46. This test sequence contains dominant horizontal motion on the background with some duplicate features caused by the man made structures. It has some dynamic foreground motion, as different parts of the foreground will experience different motions at the same time. Some blurring can also be observed as the motion speed is increased.

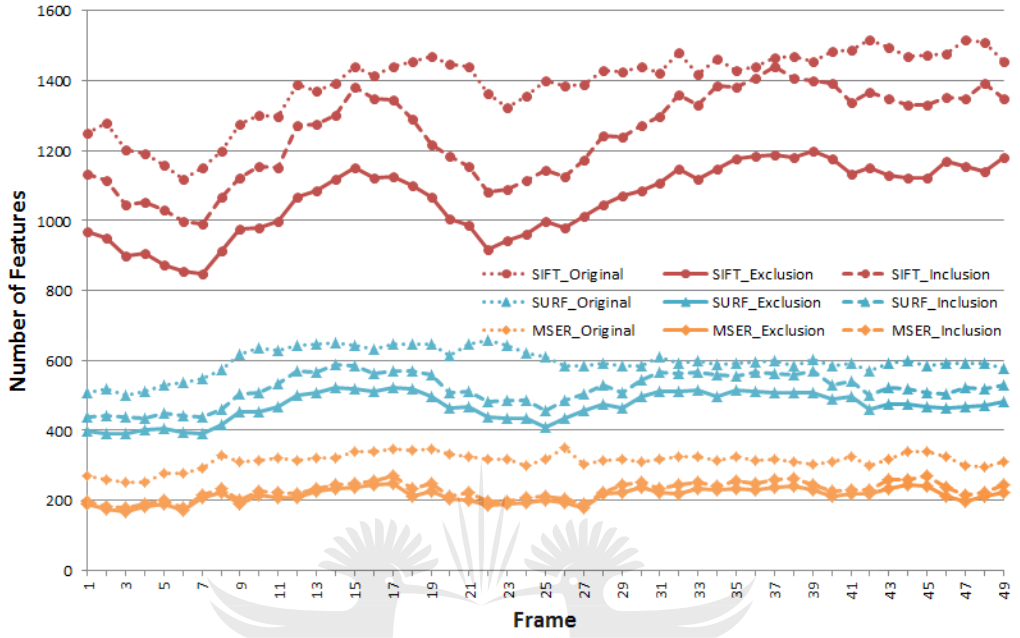
The number of features detected in this sequence varies over the frames of the sequence as different parts of the scene are occluded and move outside the image boundaries as can be seen in Figure 5.48.a. There are two dips in the frame count that can be observed in the region of frame 7 and 23. Some features will also be lost due to the blurring introduced by fast moving objects. An average matching accuracy of only 75% was achieved by the original SIFT, SURF and MSER feature detectors as can be seen in Figure 5.48.b. SURF achieved a 5% increase in matching accuracy compared to SIFT and MSER in the last half of the test sequence, but it was only able to detect a third of the features detected by



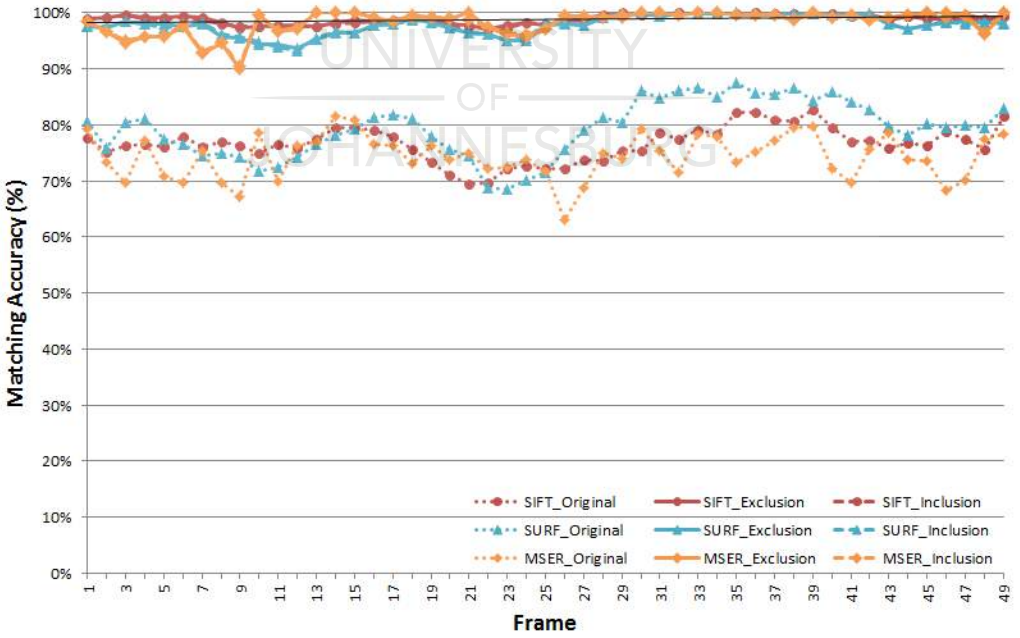
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.47: A comparison of the feature matching results obtained by the SIFT and Super-feature algorithm for frame 12 to 13 of the Alley1 test sequence

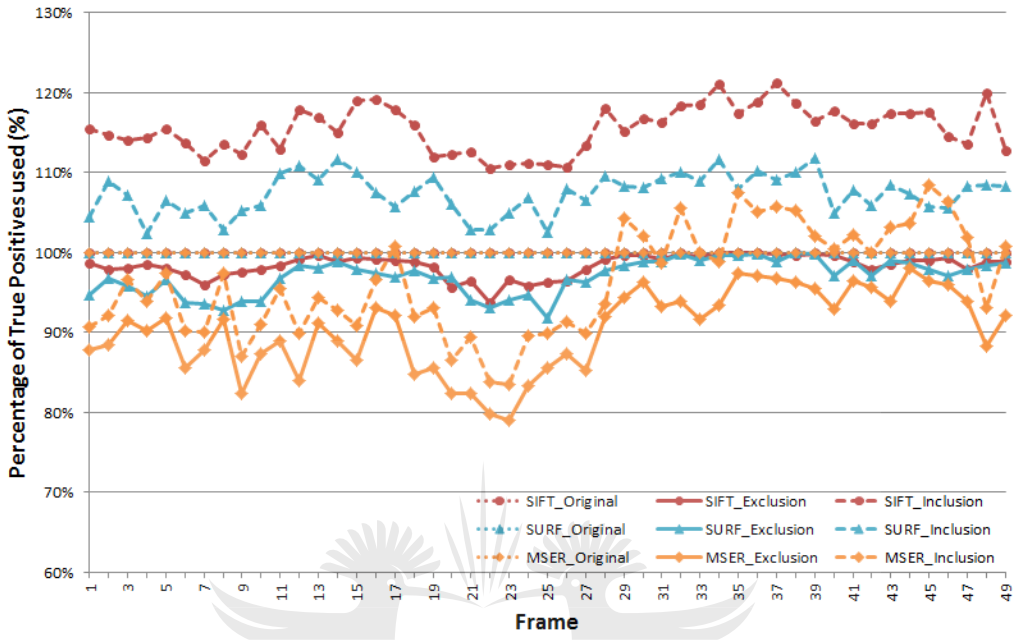


(a) Number of Features

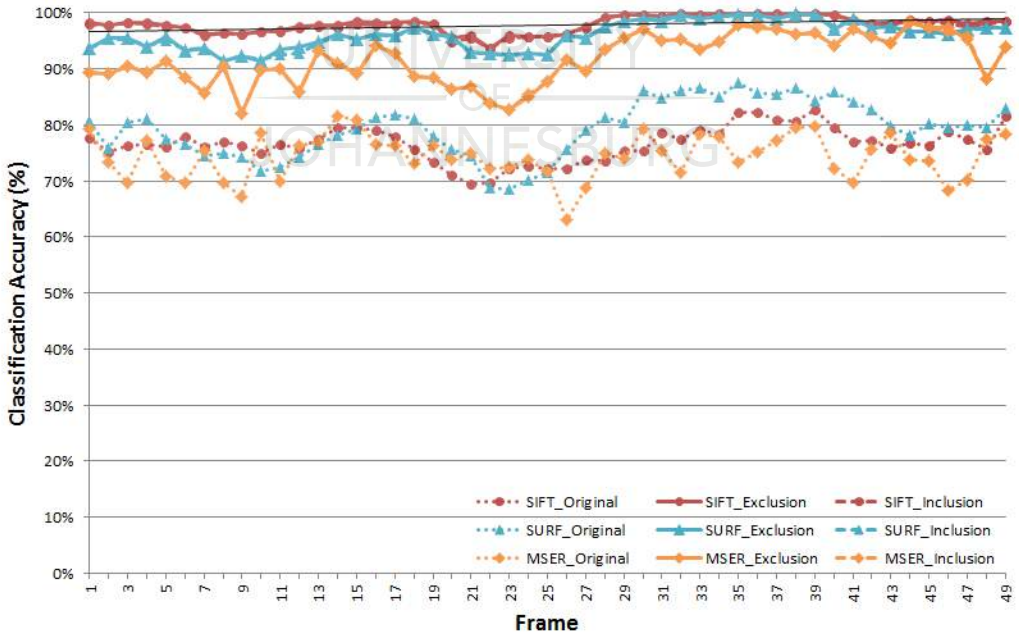


(b) Matching Accuracy

Figure 5.48: The number of features and matching accuracy results for the Alley1 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.49: The percentage of True-positives used and classification accuracy results for the Alley1 test sequence

SIFT. The detectors based on the Super-feature algorithm was able to improve the matching accuracy above 90% for all three tested feature detectors. A matching accuracy above 95% was obtained for SIFT and SURF for the majority of the frames in the sequence.

The number of True-positives used are presented in Figure 5.49.a, the Super-feature Inclusion algorithm based on SIFT and SURF was able to utilized the most number of True-positives. Integrating the Super-feature algorithm produced an improvement to the matching accuracy, a comparison between the original and Super-features method using SIFT features can be seen Figure 5.47. The MSER based Super-feature algorithm with Inclusion was not able to maintain a True-positive usage of more than 100%, it fluctuated between 80% when the least number of features were present to 108% when the most number of features were detected. Very sparse feature sets can drastically reduce the effectiveness of the Super-feature algorithm.

The classification accuracy results that can be seen in Figure 5.49.b show that the Super-feature algorithm provided an improvement over the traditional feature detection methods. The feature detectors that were able to provide a dense set of features were better able to utilize the Super-feature algorithm, these feature detectors such as SIFT and SURF maintained a classification accuracy of above 90% while MSER dropped and fluctuated between 82% and 98%.

5.4.2.2 Bamboo1 test sequence

Example frames from the Bamboo1 test video sequence can be seen in Figure 5.50. This test sequence contains a number of duplicate features due to the similarities present on the background. There are also a large number of thin moving objects which can cause occlusions, the features located on these moving objects can also suffer from changes to their description since regions used for description can be found on other moving objects. This can pose some problem for the feature detectors which will make accurate matching difficult. Some moving shadows are also present, which can cause darkening of features, which will test the lighting normalization ability of the different feature detectors.

There was a big difference between the number of features detected by the dif-

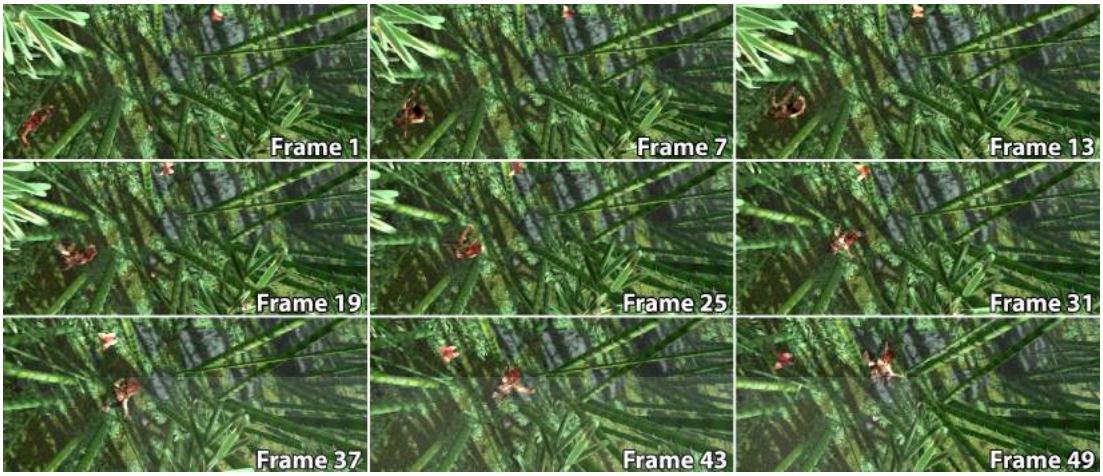
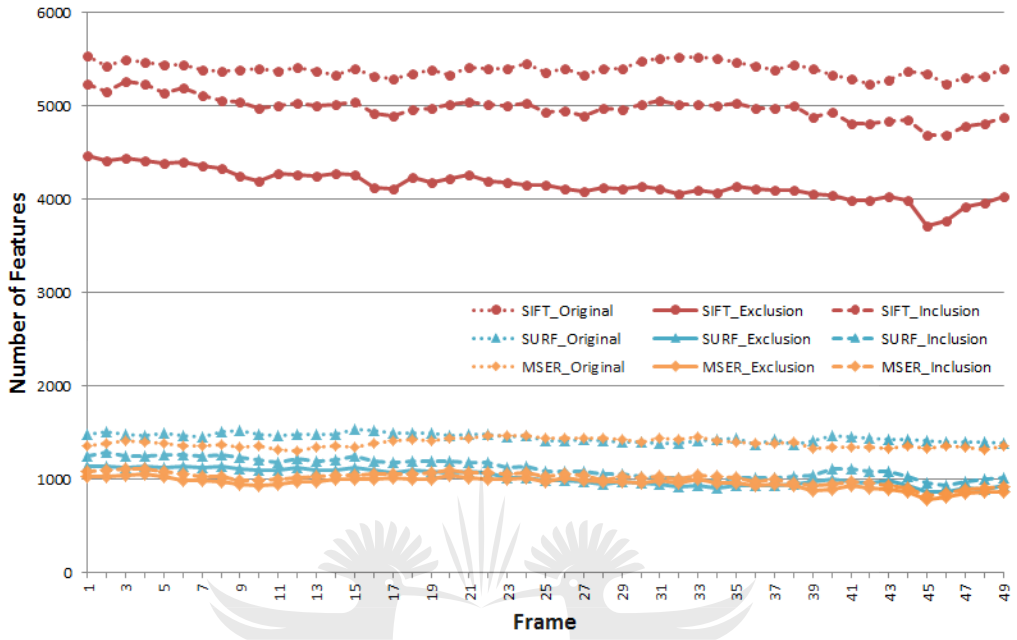


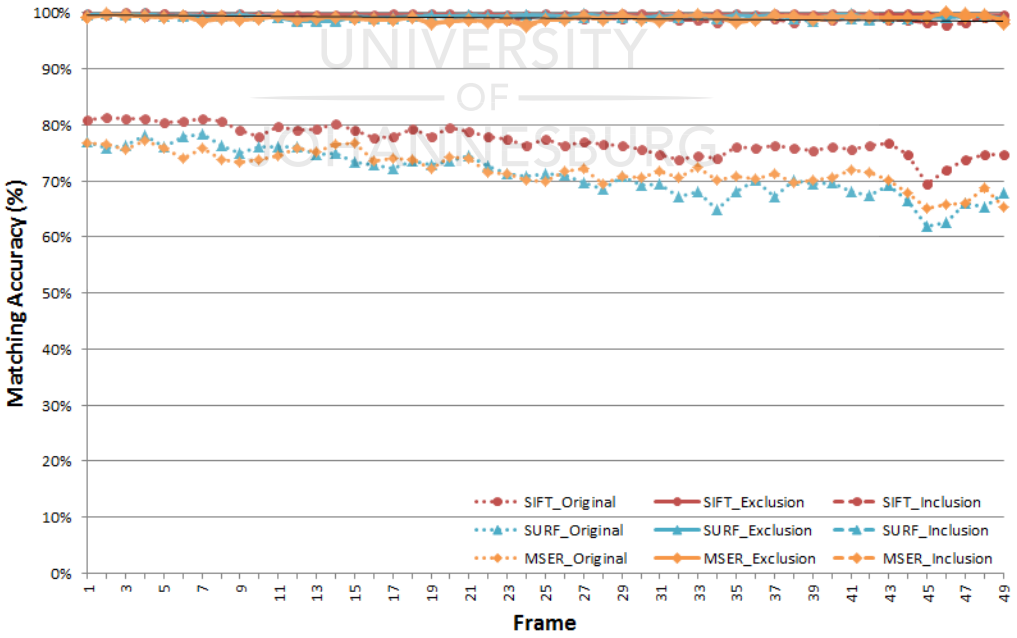
Figure 5.50: Example frames from the bamboo1 video sequence

ferent feature detection methods as can be seen in Figure 5.50. SURF and MSER only detected, on average 1400 features where SIFT detected 5500 features. The three original feature detection methods performed similarly over the sequence of video frames, there was only between 5% to 10% matching accuracy difference as can be seen in Figure 5.51.b. SIFT, SURF and MSER had an average matching accuracy of 75%. Incorporating the Super-feature algorithm into these feature detectors allowed for an improvement of 25% of the matching accuracy. Even though different amounts of features were detected between the three feature detection methods, above 98% matching accuracy was obtained consistently for the entire test sequence. The Super-feature algorithm relied less on a single feature's description information which might have been corrupted, but rather focused on a group of feature descriptions which enable more reliable matching which can be seen from these results. This produced a dramatic improvement over the traditional methods.

The classification accuracy for the different Super-feature algorithms provided in Figure 5.52.b was also maintained at a high level of accuracy. Better than 93% classification accuracy was produced by the MSER based Super-feature algorithm. The SIFT and SURF based Super-feature algorithm for inclusion as-well as exclusion of poor features, both produced classification accuracy results in access of 98%.

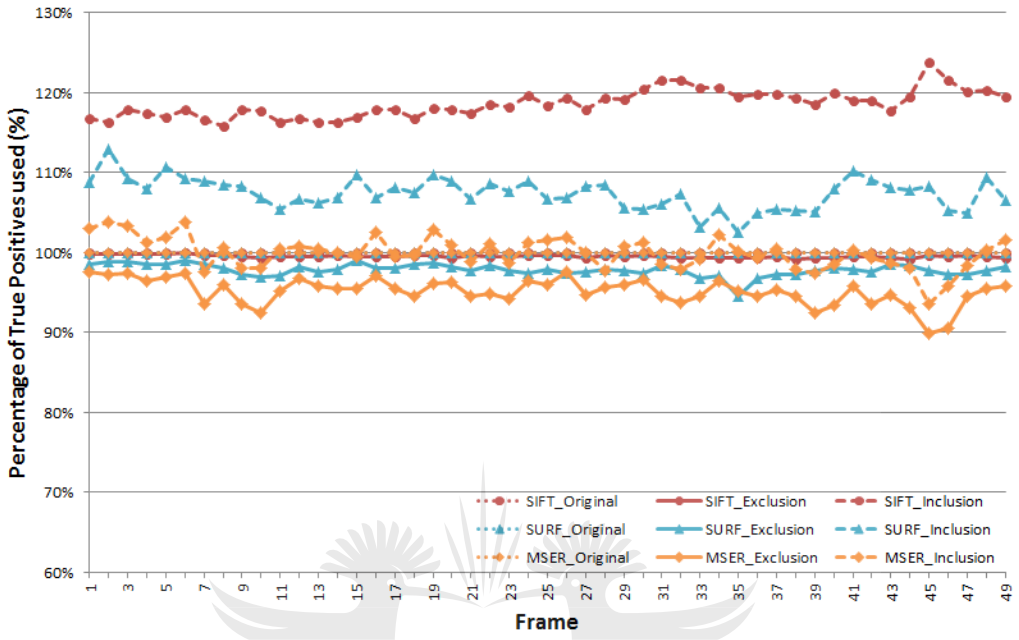


(a) Number of Features

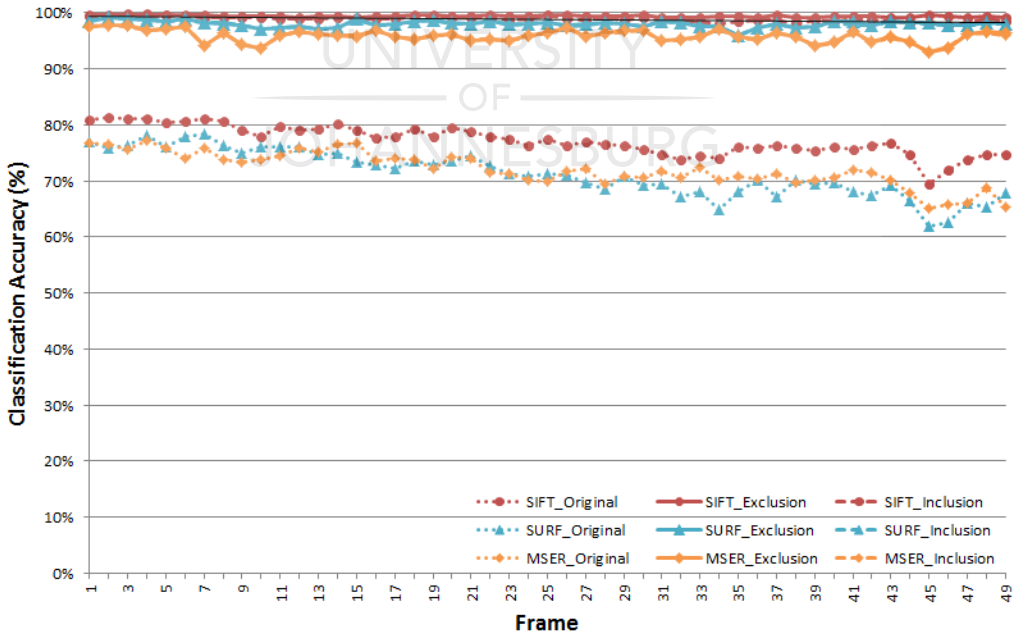


(b) Matching Accuracy

Figure 5.51: The number of features and matching accuracy results for the Bamboo1 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.52: The percentage of True-positives used and classification accuracy results for the Bamboo1 test sequence

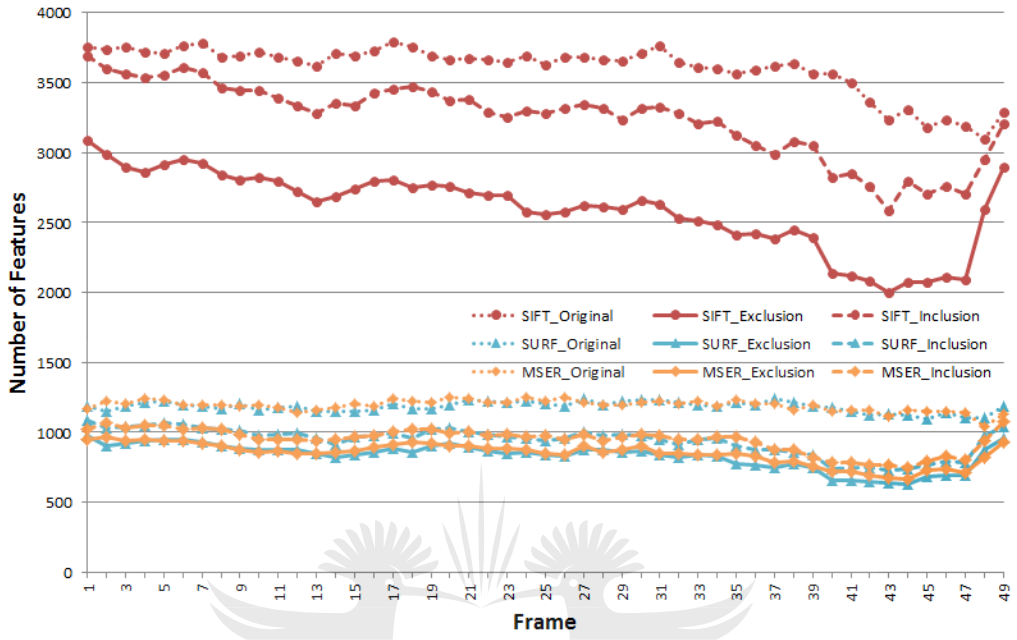
SIFT based Super-features with Exclusion of invalid features detected 99% of the original True-positives as can be seen in Figure 5.52.a, by correcting and including the corrected features, the True-positive usage was boosted to between 117% to 124%. The SURF based Super-feature algorithm was able to use 97% of the original True-positives which could be boosted to 107% True-positive usage if corrected features were included. MSER on the other hand struggled and was only able to achieve more than 100% True-positive usage for some of the tested sequence frames.

5.4.2.3 Bamboo2 test sequence

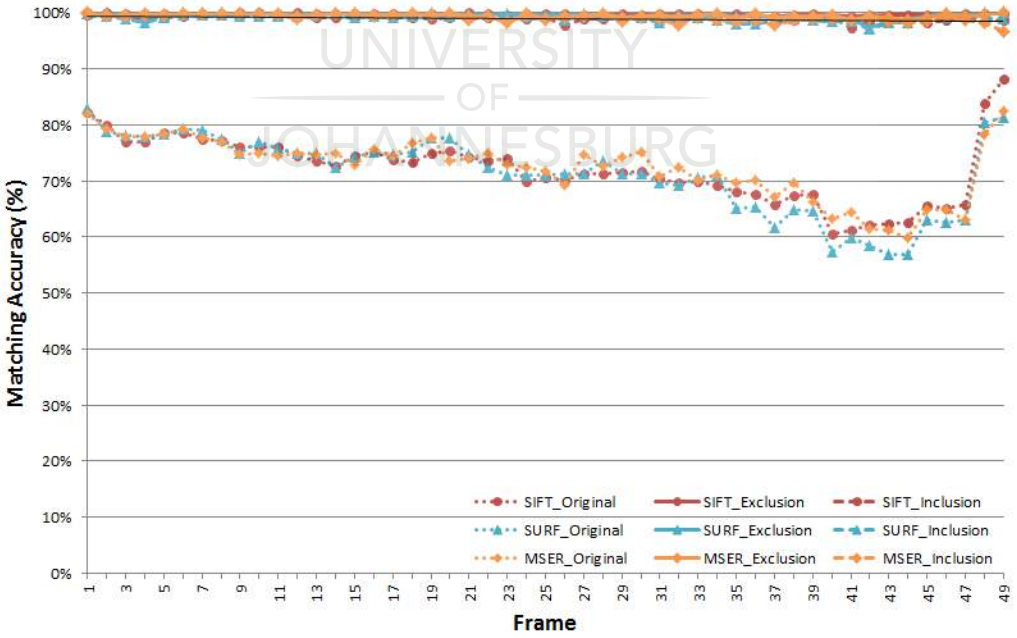
The Bamboo2 test sequence once again has many duplicate features due to the abundance of bamboo in the video frames. A large number of new features are introduced since new regions have become visible due to camera motion. Some feature will also be lost as regions move outside the camera frame. The features located on the foreground moving object will be difficult to match because of the large amounts of motion blur and the dynamic nature of the motion. Example frames of this sequence can be observed in Figure 5.53, note the large number of similar regions that can produce duplicate features.

As the video sequence progresses the number of features detected decrease

Figure 5.53: Example frames from the bamboo2 video sequence

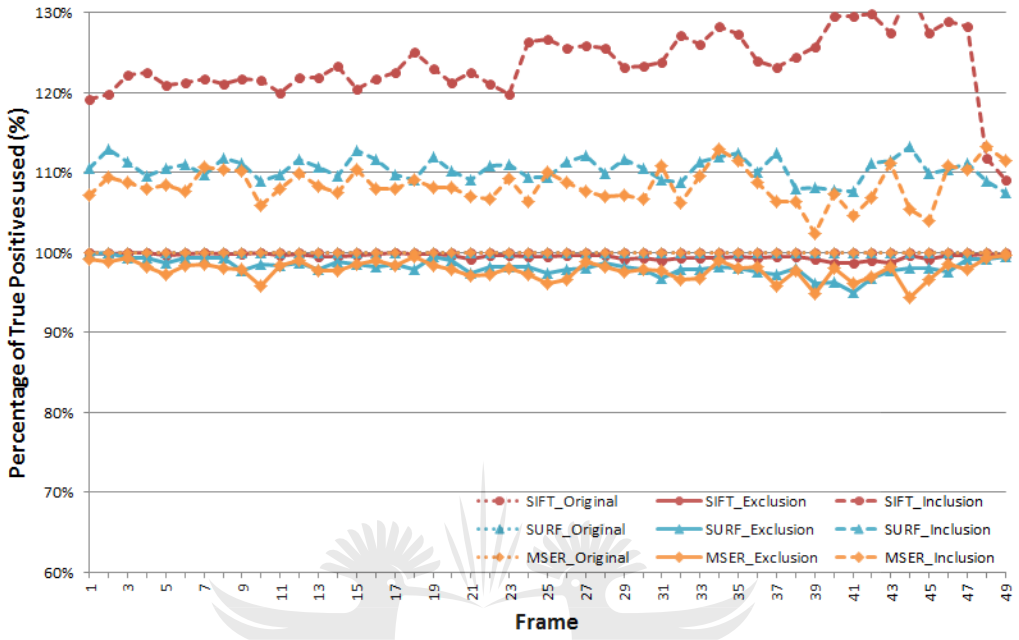


(a) Number of Features

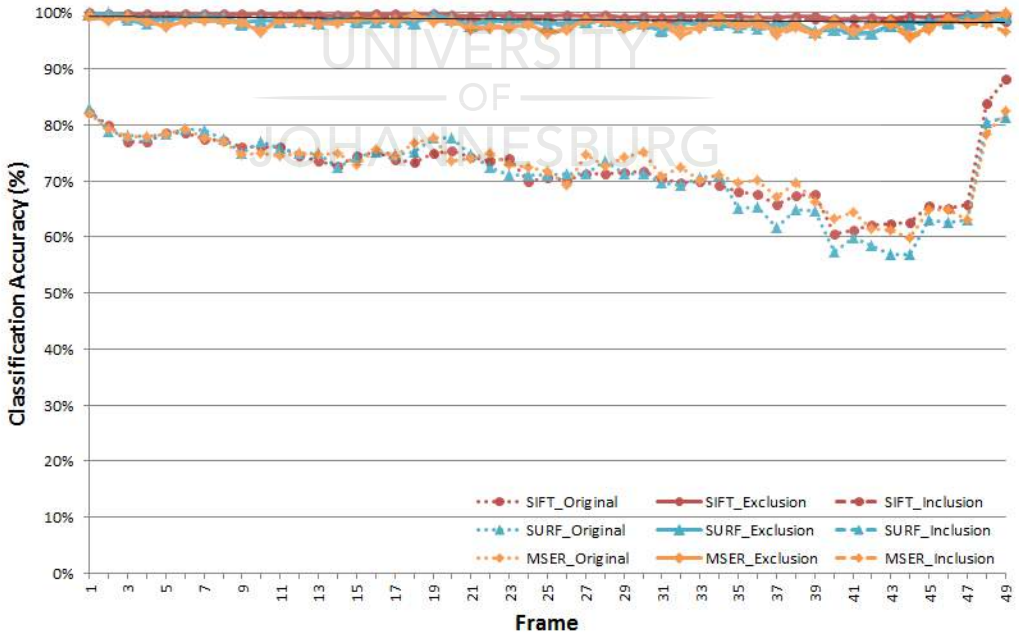


(b) Matching Accuracy

Figure 5.54: The number of features and matching accuracy results for the Bamboo2 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.55: The percentage of True-positives used and classification accuracy results for the Bamboo2 test sequence

because the foreground object affected by motion blur occupies more image space. This decrease in the number features also affected the matching accuracy as can be seen in Figure 5.54.b. As the number of features decrease the matching accuracy of the original feature detectors dropped from 80% down to 60%. The Super-feature based feature detectors did not suffer from this problem and a matching accuracy of above 97% was maintained for the entire test sequence. Features classified as incorrect was removed from the matching process, while some incorrect features that could be corrected was added back into the match list. This reduced the number of features, but improves the matching and classification accuracy.

The True-positive usage percentage of the different tested algorithms are provided in Figure 5.55.a. The Super-feature algorithm with Inclusion based on SIFT, SURF and MSER, all improved the number of True-positives used. An increase of between 4% to 33% True-positive usage was observed, with an average increase of 8% for MSER, 10% for SURF and 25% for SIFT.

Similar to the matching accuracy, the classification accuracy remained stable for the entire range of tested video sequence frames. Even though the classification accuracy of the original feature detectors drops down to as low as 60%, the Super-feature algorithm for both Inclusion as well as Exclusion achieved above 95% accuracy for all the tested feature detector. A slight accuracy increase was visible for SIFT compared to SURF and MSER which was able to maintain a classification accuracy of above 99%. The Super-feature algorithm provided reliable results for scenes with different foreground as well as background motion.

5.4.3 Dynamic, non-rigid and morph-able objects and environments

Since a global model of the transformations applied to image features are not required by the Super-feature algorithm, it has the ability to work on scenes that contain different local motion such as morph-able and dynamic objects. A number of tests that contain non-rigid moving objects will be performed to determine the Super-feature algorithm's ability to handle these situations.

5.4.3.1 Bandage1 test sequence

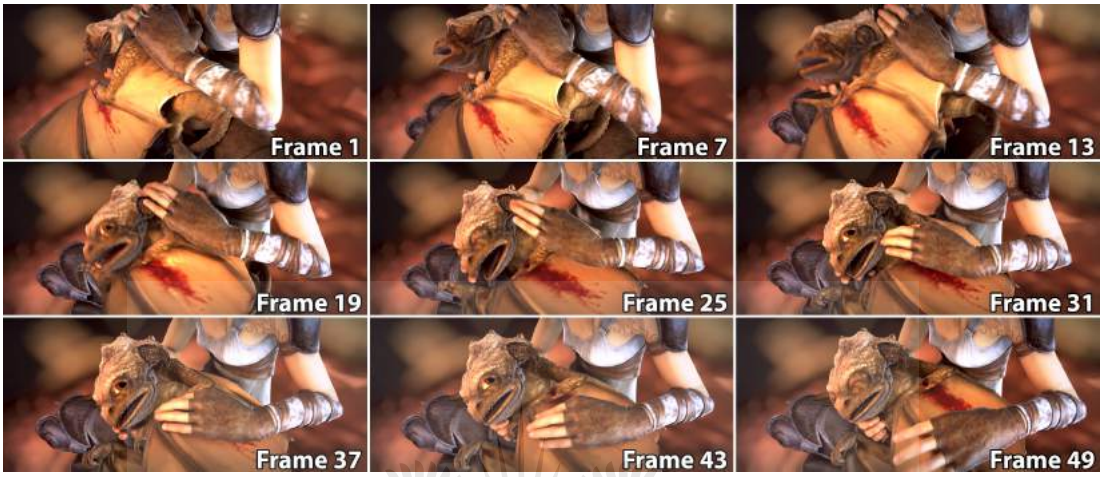


Figure 5.56: Example frames from the bandage1 video sequence

The bandage1 sequence will be the first sequence that will be tested, sample frames from the sequence can be seen in Figure 5.56. This test sequence consists primarily of dynamic moving objects, where every local region is exhibiting motion different from the neighbouring local regions. What makes this matching problem difficult is that the image regions described by a features description can become distorted and stretched due to the different local motions and transformations. A number of image regions can also become occluded, or may exhibit some motion blurring.

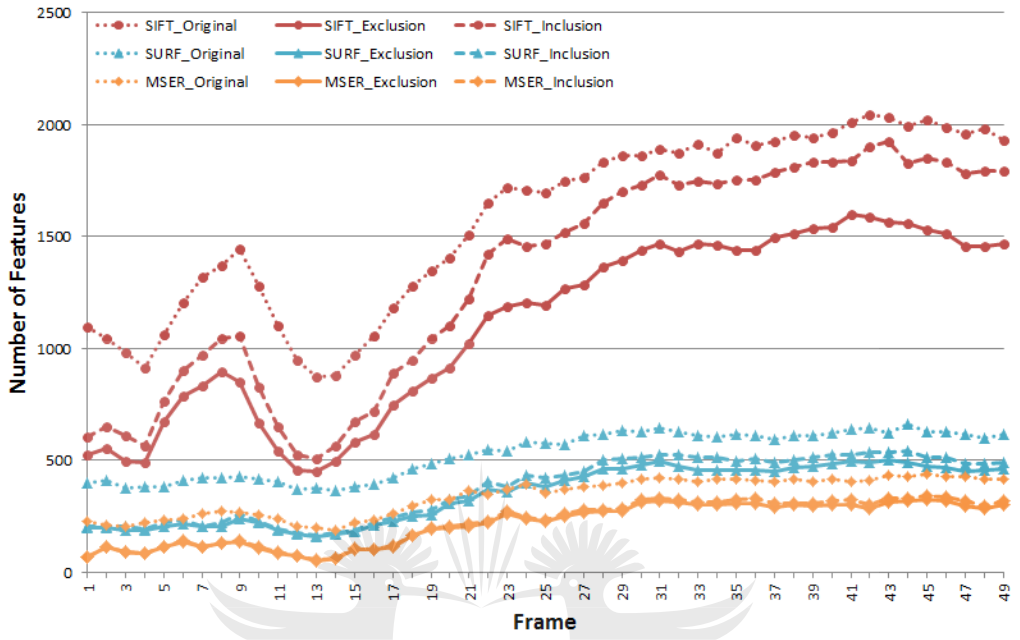
The number of features located in the test sequence frames increase toward the end of the sequence, this is visible in Figure 5.58.a. This can be attributed to the fast motion and blurring visible in the first half of the sequence, the motion of the dynamic objects becomes more stationary toward the end of the sequence and motion speed is reduced. The detection of low numbers of features can negatively affect the matching and classification accuracy of both the original feature detectors as well as the improved Super-feature algorithms.

The matching accuracy of the original feature detector is low for the initial frames, which contain fast motion, stretching of local regions and motion blurring. The different feature detectors had some difficult detecting features, only sparse

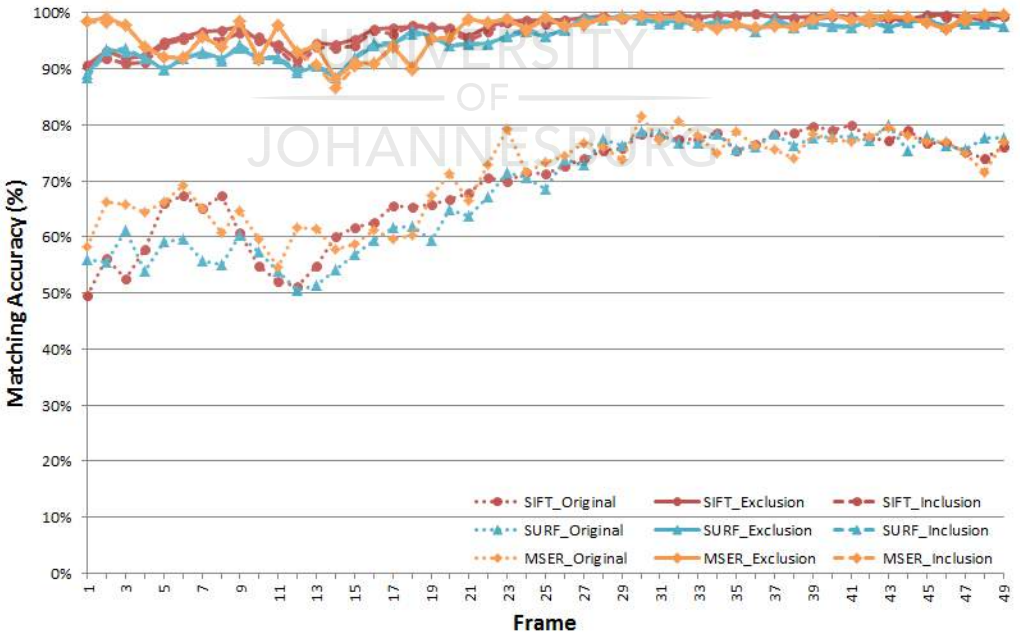
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.57: A comparison of the feature matching results obtained by the SIFT and Super-feature algorithm for frame 13 to 14 of the Bandage1 test sequence

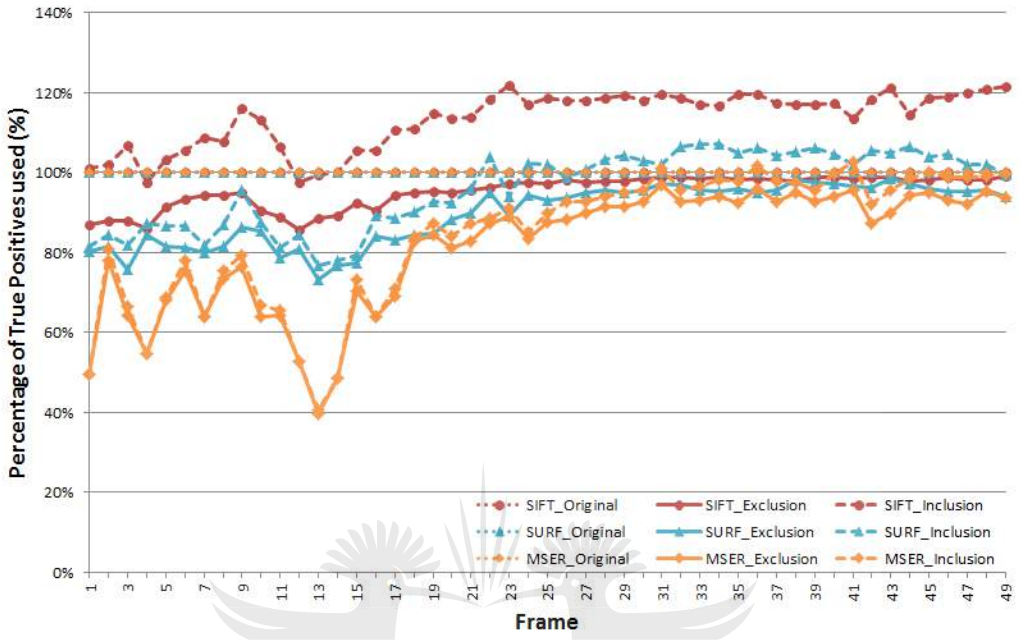


(a) Number of Features

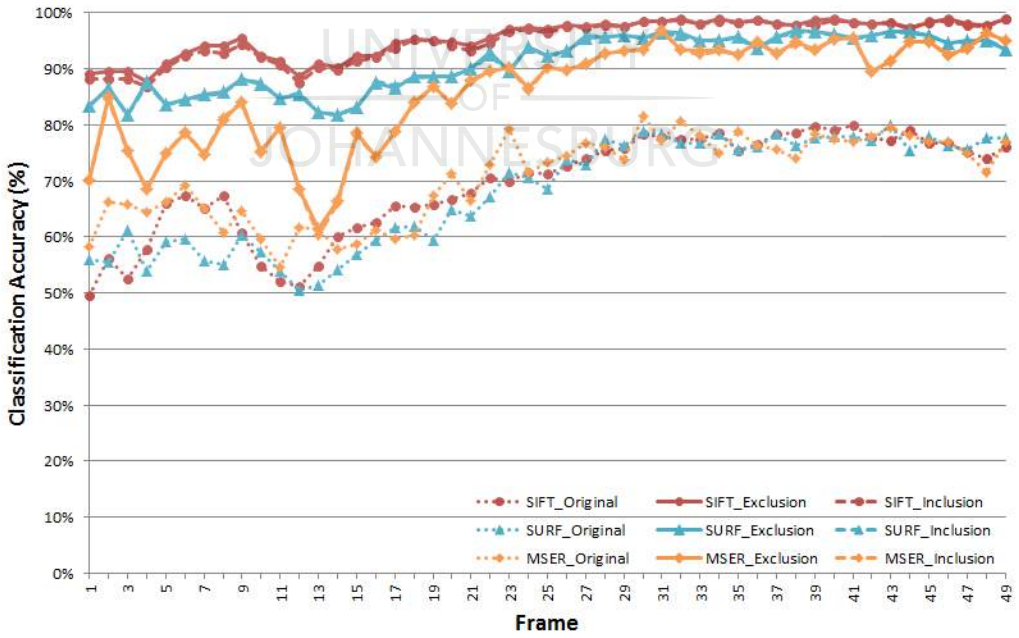


(b) Matching Accuracy

Figure 5.58: The number of features and matching accuracy results for the Bandage1 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.59: The percentage of True-positives used and classification accuracy results for the Bandage1 test sequence

sets of feature were detected by them. A demonstration of the feature matching difficulty experienced by the tradition feature detector and the Super-feature counter part of SIFT can be seen in Figure 5.57. The matching accuracy results are provided in Figure 5.58.b. The original SIFT, SURF and MSER matching accuracies dropped to close to 50% which is substantially less than any of the minimum matching accuracies obtained in the simpler complex transformation tests. By incorporating the Super-feature algorithm for SIFT, SURF and MSER features, the matching accuracy could be improved to higher that 88% for the difficult parts of the test sequence and above 95% for the simpler parts. Matching accuracy increases of up to 45% were observed in this test video sequence.

Only Super-feature Inclusion based on SIFT was able to make use of more than a 100% True-positives, it was able to reach True-positive usage of close to 120%. SURF and MSER struggled, their True-positive usage dropped substantially for the difficult first half of the sequence and increased in the second half to above 90%. The same decrease in accuracy for the first half of the sequence was also observed for the classification accuracy results provided in Figure 5.59.b. Super-features based on SIFT achieved the best results, it only dipped below 90% classification accuracy in the most extreme cases and provided results in the range of 90% to 100% for the majority of the tested frames in this sequence. The second best feature detector was Super-features based on SURF, which provided a classification accuracy of above 80% even on the difficult frames in the sequence.

5.4.3.2 Bandage2 test sequence

The Bandage2 test sequence is very similar to the Bandage1 sequence, a number of local transformations can be observed which can make feature matching difficult. Example frames from the Bandage2 sequence can be seen in Figure 5.60, note the occlusions caused by the motion of the dragon head as well as the hand which is affected by depth-of-field blurring.

MSER and SURF detected a similar number of features, they detected on average 400 features per frame as can be seen in Figure 5.61.a. SIFT on the other hand detected three times more features and was able to detect 1200 features per frame. There were some fluctuations in the number of features detected as

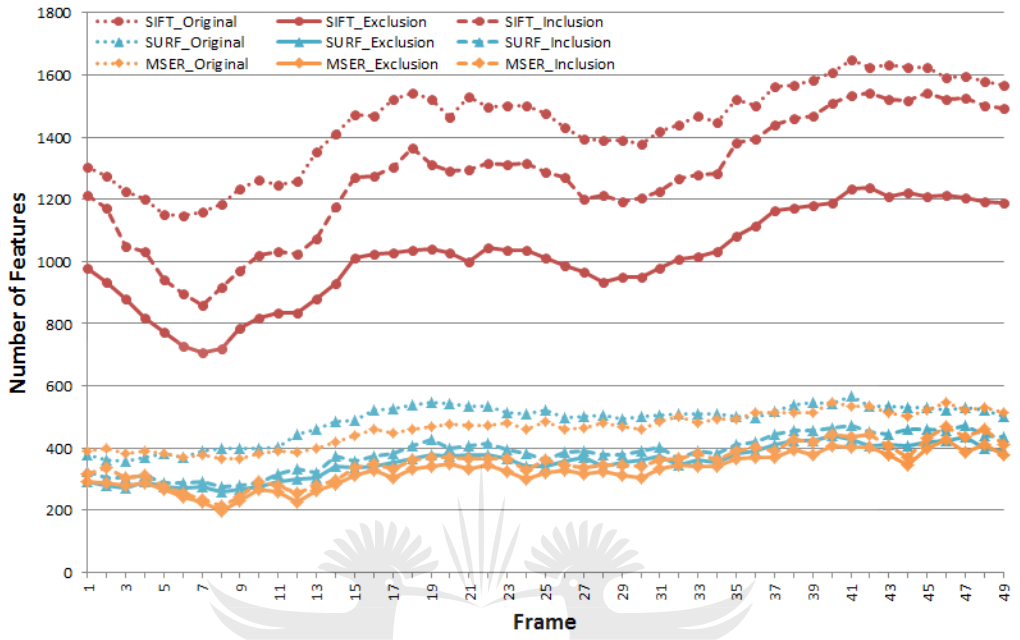
Figure 5.60: Example frames from the bandage2 video sequence

features became occluded and small features were lost due to blurring.

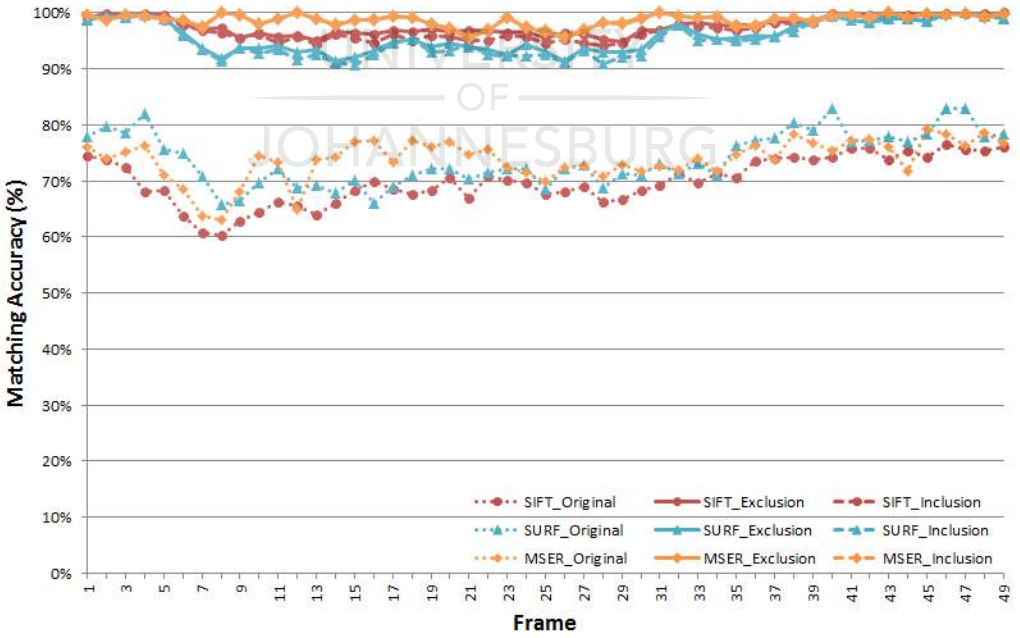
The original matching accuracy of SIFT, SURF and MSER was in the range of 60% to 83% and averaged at 72%. The matching accuracy results are provided in Figure 5.61.b. MSER performed slightly better than SIFT and SURF but at a reduced number of features. The Super-feature algorithm applied to the different feature detectors was able to improve the matching accuracy. Results above 90% was obtained for all the tested detectors with MSER in the lead, maintaining a matching accuracy above 95% for all tested frames. Super-features based on SURF were the worst performing of the Super-feature algorithms when the matching accuracy is considered.

The SIFT based Super-feature with Inclusion algorithm was the only Super-feature algorithm that was able to maintain a True-positive usage of more than a 100% at an average of 121%. The Super-feature algorithm with Inclusion of corrected features performed poorly on this test sequence as can be seen in Figure 5.62.a. It was able to correct some features which improved the feature usage of the Super-feature Exclusion algorithms.

Since fewer True-positives were used because they were misclassified, the classification accuracy was affected as seen in Figure 5.62.b. The classification accuracy of the Super-feature based algorithms remained high compared to the traditional feature detection methods. An average increase of 25% classification

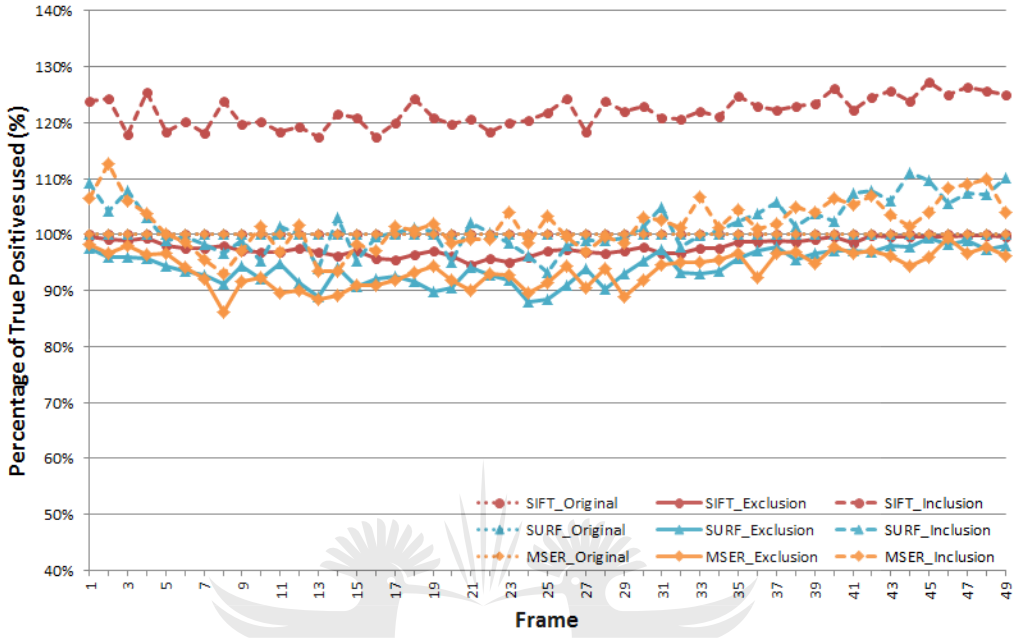


(a) Number of Features

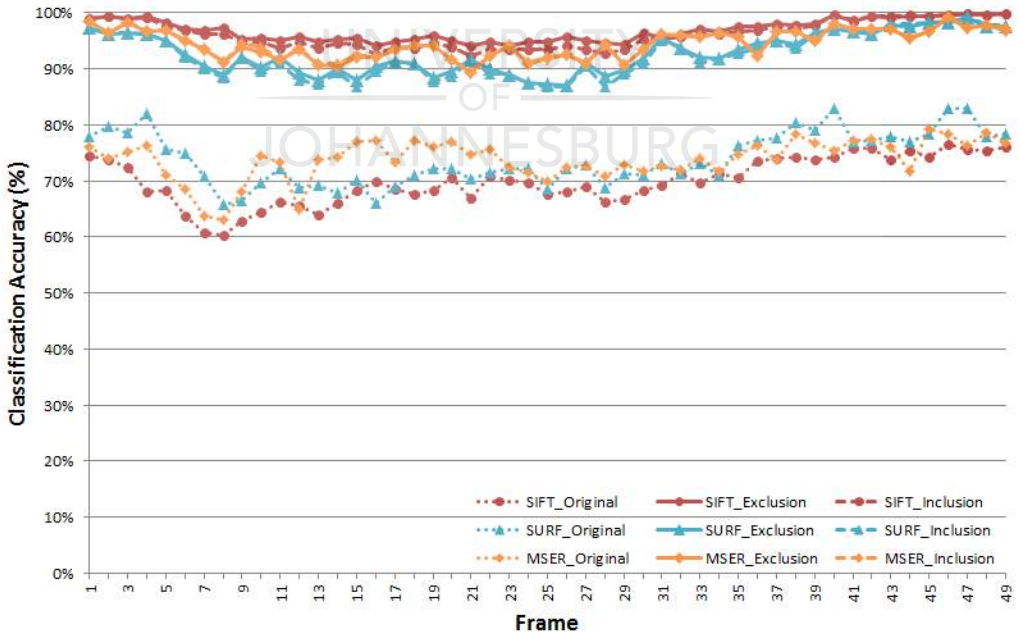


(b) Matching Accuracy

Figure 5.61: The number of features and matching accuracy results for the Bandage2 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.62: The percentage of True-positives used and classification accuracy results for the Bandage2 test sequence

accuracy was obtained by applying Super-features to this problem, which is still a big improvement over the original results.

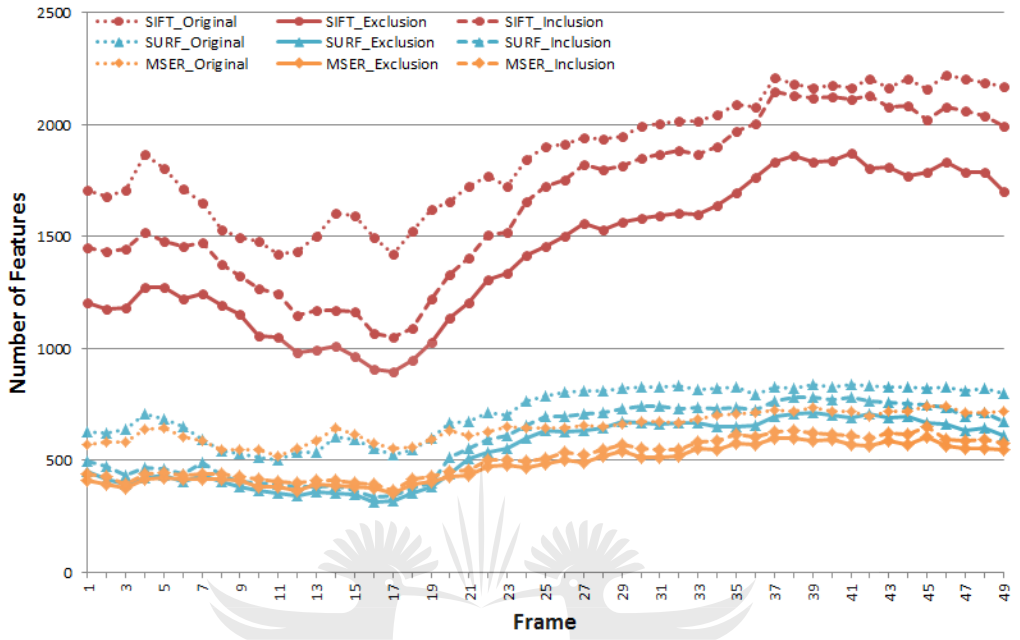
5.4.3.3 Market2 test sequence

The Market2 test sequence contains a number of dynamic objects that at times produce local occlusions as they pass behind and in front of each other. Motion and out of focus blur can also be observed as seen in the example frames provided in Figure 5.63.

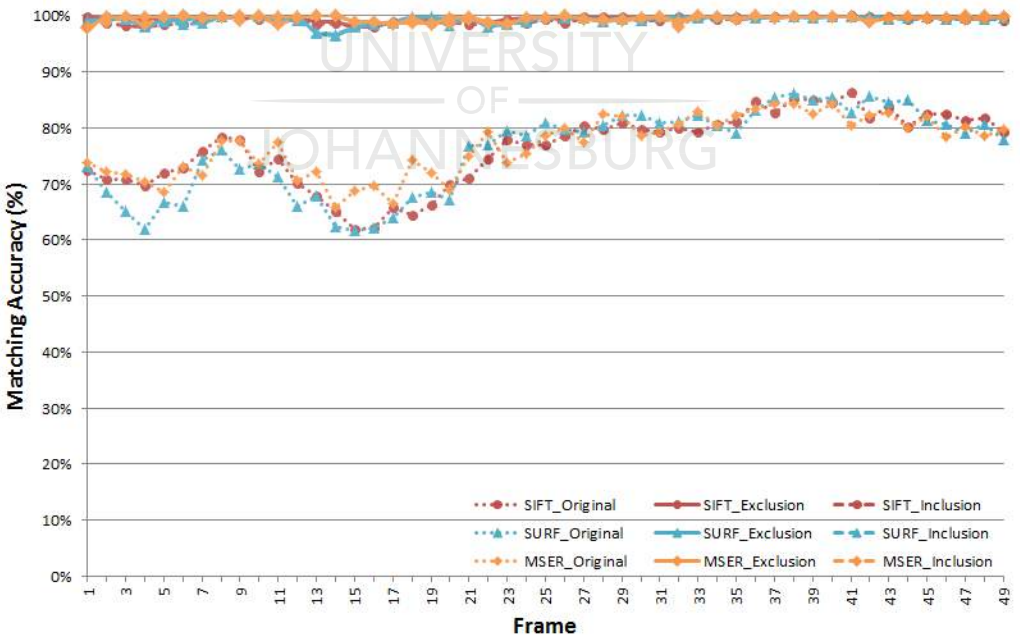
The number of features detected by the three different methods increased during the progression of the test video sequence as seen in Figure 5.64.a. MSER detected slightly more features compared to SURF in the first half of the sequence, after which SURF detected more features than MSER. Improved results can be expected for the test frames that contain more feature whereas a decreased matching and classification accuracy can be expected when the feature count is low.

All three feature detectors based on the Super-features algorithm with Exclusion as well as Inclusion shows a substantial improvement over their traditional counterparts. Only a small difference in matching accuracy was observed between SIFT, SURF and MSER after Super-features were applied as can be seen in Fig-

Figure 5.63: Example frames from the market2 video sequence

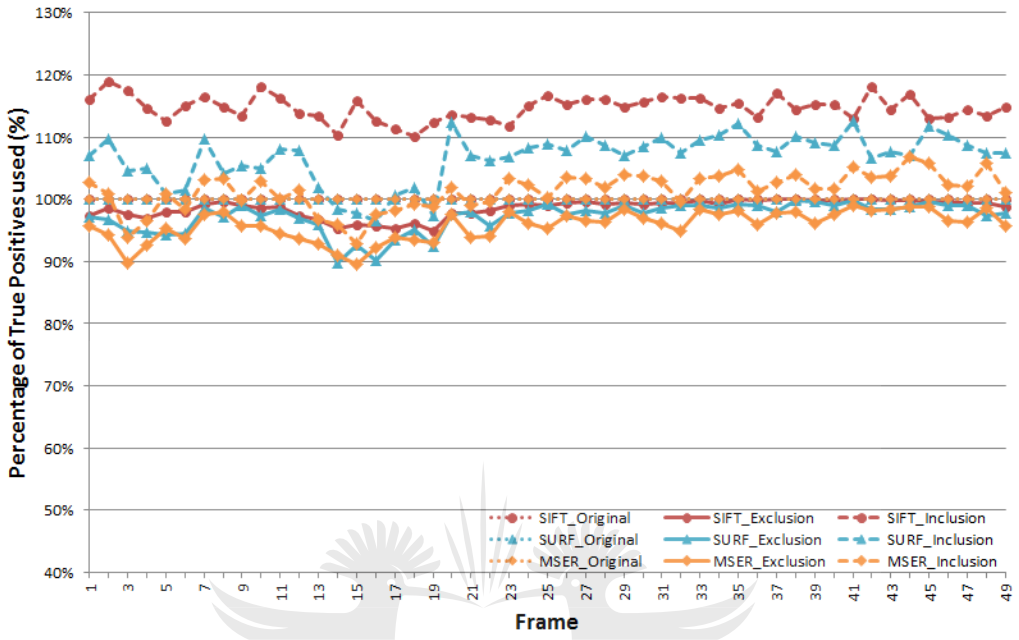


(a) Number of Features

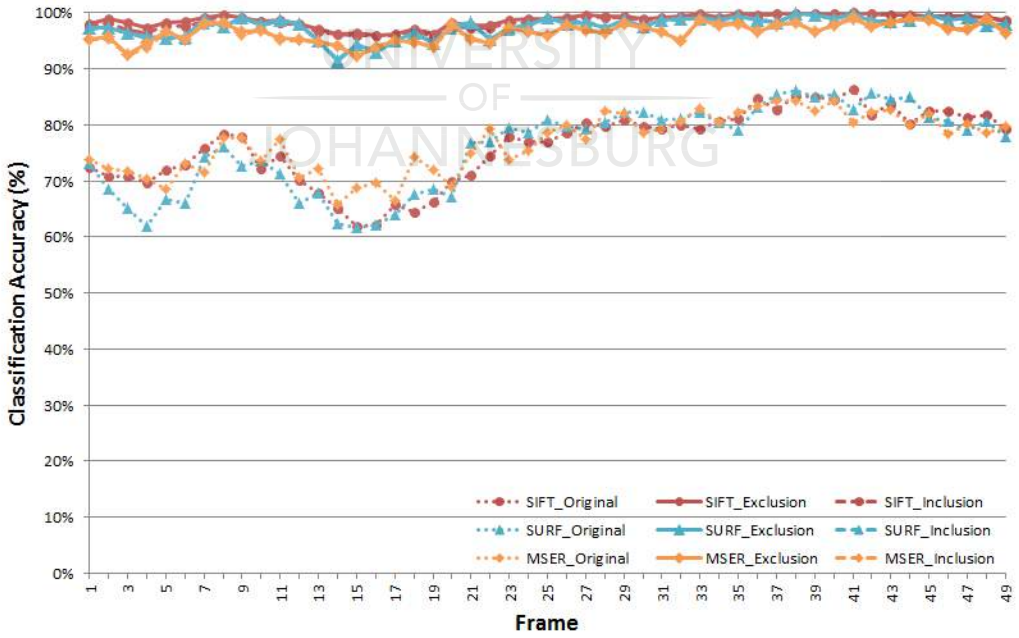


(b) Matching Accuracy

Figure 5.64: The number of features and matching accuracy results for the Market2 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.65: The percentage of True-positives used and classification accuracy results for the Market2 test sequence

ure 5.64.b. MSER obtained the highest matching accuracy of 99.38%, SURF produced the second highest score of 99.23% and SIFT was last with 99.19%. Even though SIFT produced the lowest matching accuracy, it was able to achieve this with an average increase of 15% True-positive usage as well as the detection of three times more features than SURF and MSER. The True-positive usage results are provided in Figure 5.65.a. It can be seen that SURF was only able to produce a 6% increase and MSER a 1% over their original feature detection counterparts.

A slight reduction of 4% in the classification accuracy can be observed in Figure 5.65.b for the Super-feature based algorithms in the first half of the test sequence. The traditional feature detectors had a more extreme decrease in classification accuracy of between 10% to 15%. This increase in match outliers was responsible for the slight drop in matching accuracy, observed for the Super-feature algorithms. The Super-feature algorithm provided a consistent average classification accuracy of 97% for the three different feature detector as well as never dropping below 91%.

These results show that Super-features work well for matching problems with morph-able, non-rigid objects with occlusion and blurring. Even if a number of dynamic moving objects are present, the Super-features algorithm can reliably increase the matching and classification accuracy.

5.4.4 Fast object and camera motion with motion blurring

The last set of tests will be performed to test the Super-feature algorithm's ability to handle feature matching in extreme cases. These test sequences contain fast motion of foreground objects as well as background objects, the majority of the foreground objects are dynamic, non-rigid and morph-able. Motion blurring and occlusions make feature matching difficult in these sequences.

5.4.4.1 Market5 test sequence

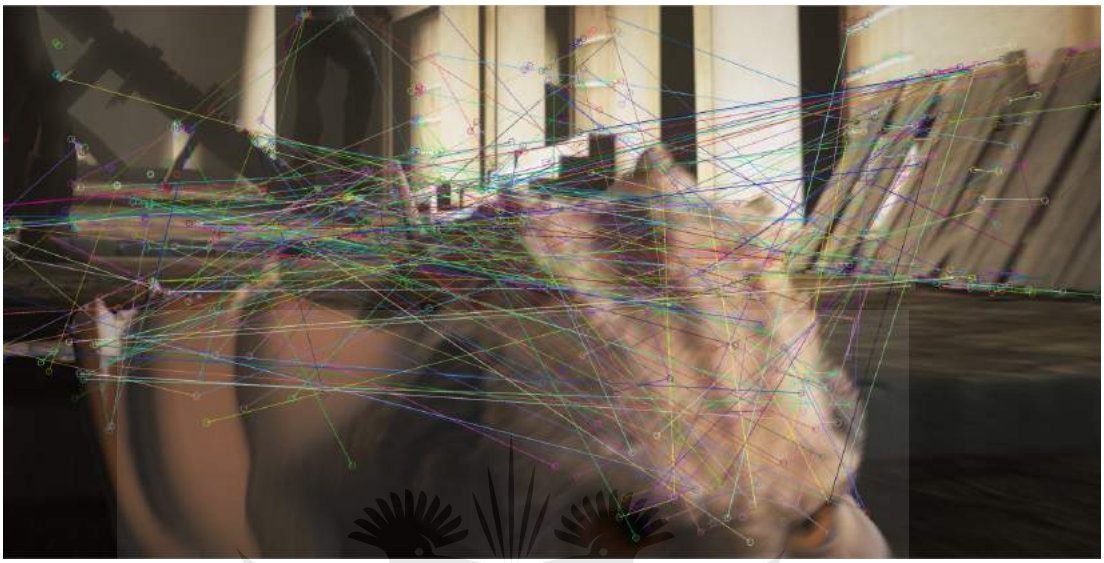
Example frames for the Market5 video sequence can be seen in Figure 5.66. Feature matching in this sequence is difficult due to the directional blurring effect produced by the fast motion and motion blurring that make feature description

Figure 5.66: Example frames from the market5 video sequence

difficult. Weak feature descriptions can negatively impact the matching performance, which will increase the number of outliers present during the estimation process of the Super-feature algorithm.

Only a sparse set of features were detected in the Market5 test sequence by the three feature detectors as seen in Figure 5.68.a. The original SIFT feature detector only detected 386 features on average, SURF detected 156 and MSER detected the least number of features with 147 features for the video sequence. The number of features detected, changed dramatically between the different frames of the sequence. This can be attributed to the extreme amounts of motion blur introduced by the fast motion, usually blurring as a side effect discards smaller high-frequency features.

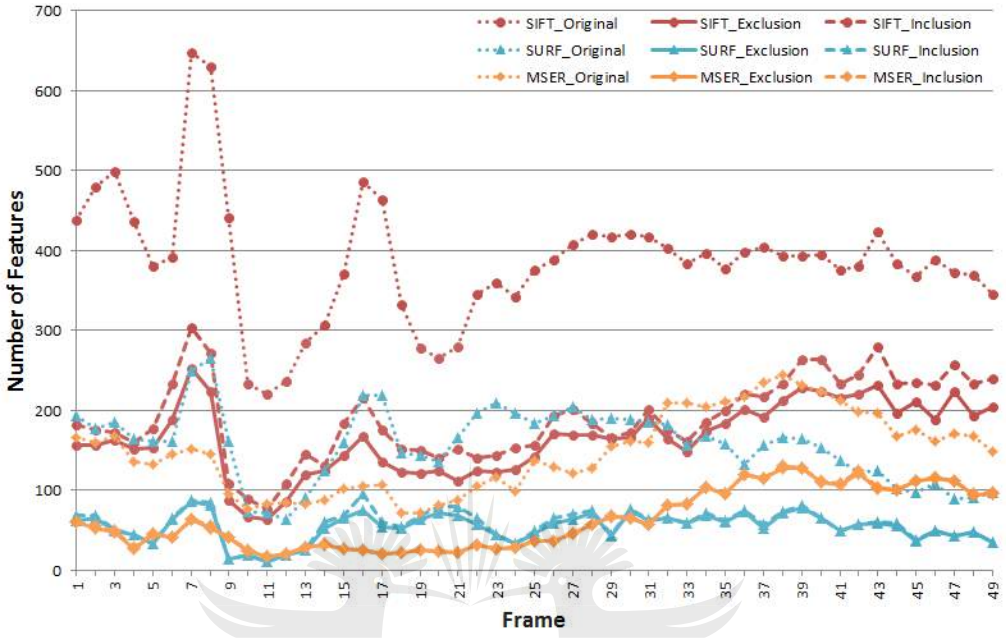
The matching accuracy results are provided in Figure 5.68.b and a visual demonstration of the the difference between SIFT and the Super-feature counterpart can be seen in Figure 5.67. The traditional feature detectors performed poorly on this sequence, matching accuracies as low as 20% was recorded with an average of 40% was common for the first half of the sequence. A slow increase in matching accuracy towards the end of the test sequence can be observed as the camera motion starts to stabilise and less motion is apparent on the background. The SIFT and SURF based Super-feature algorithms improved the matching accuracy to above 90%. The MSER based Super-feature algorithm's results on



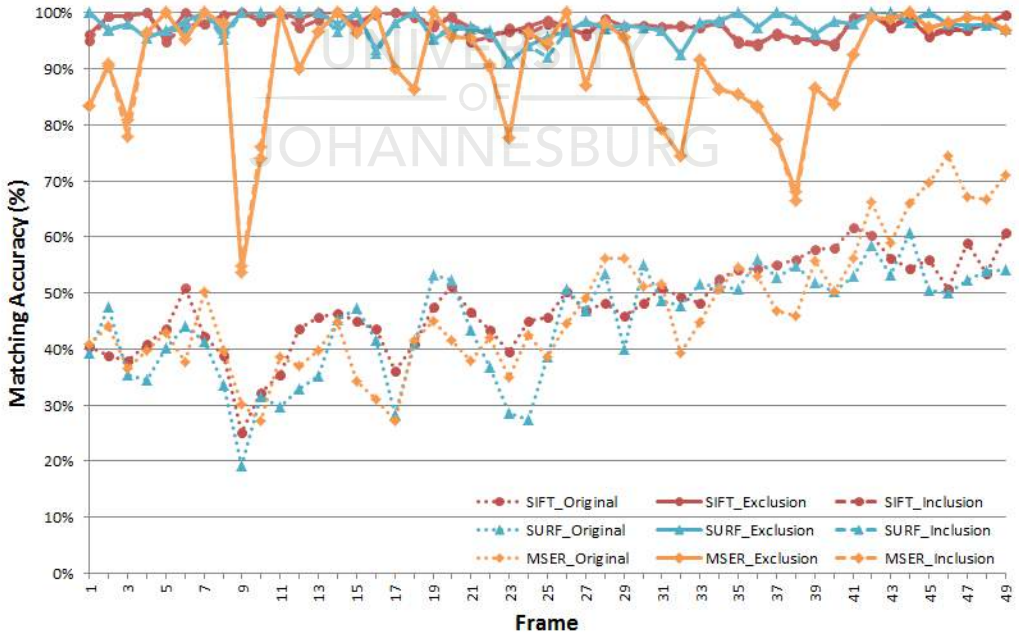
(a) Original SIFT matching results

(b) SIFT based Super-feature matching results

Figure 5.67: A comparison of the feature matching results obtained by the SIFT and Super-feature algorithm for frame 7 to 8 of the Market5 test sequence

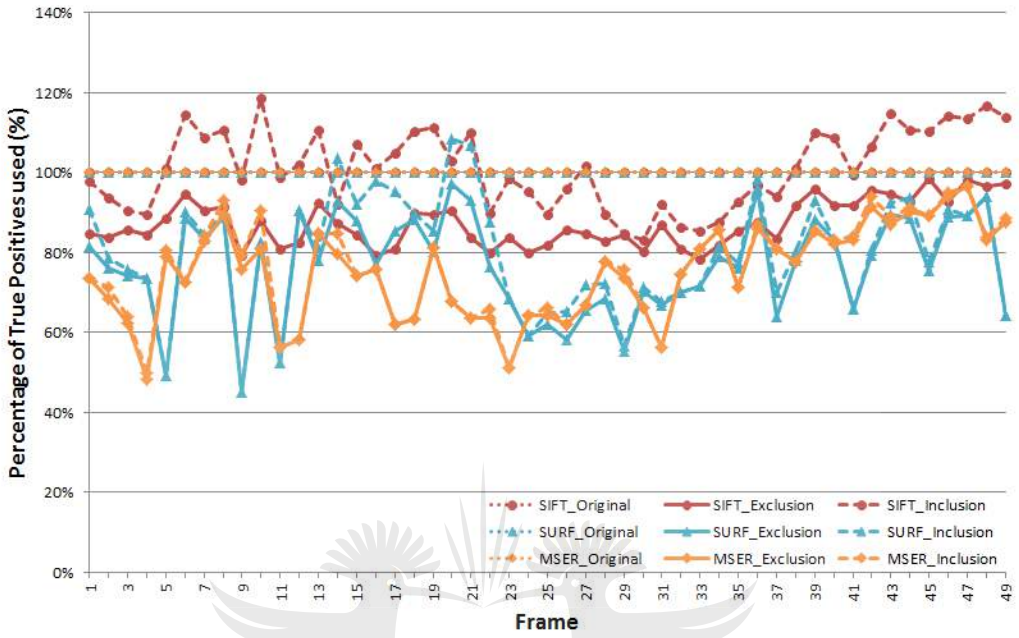


(a) Number of Features

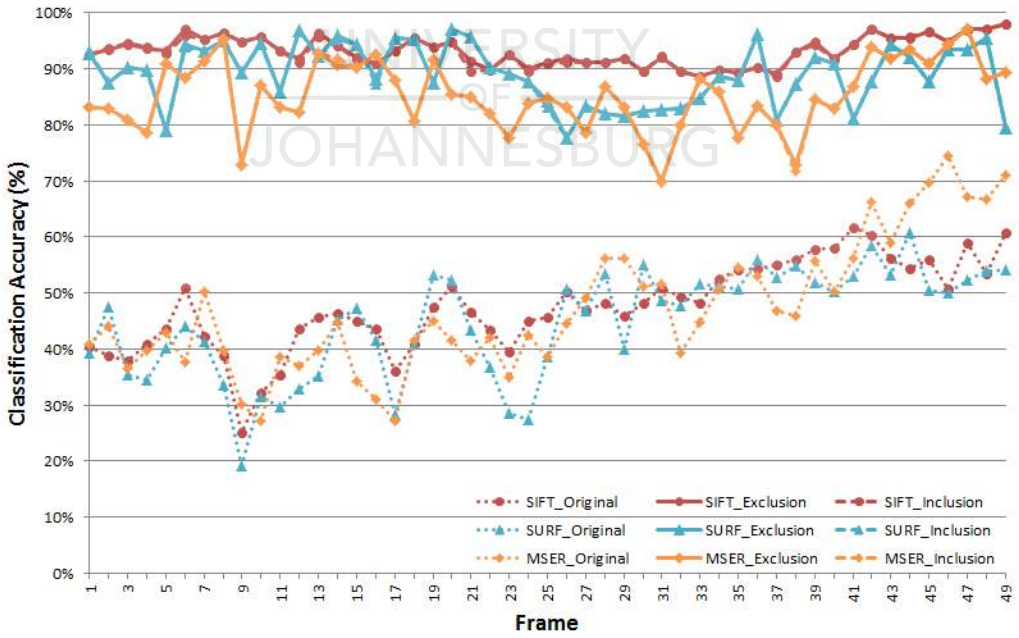


(b) Matching Accuracy

Figure 5.68: The number of features and matching accuracy results for the Market5 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.69: The percentage of True-positives used and classification accuracy results for the Market5 test sequence

the other hand fluctuated a lot, performing poorly on some tested frames. It did though, always provide an improvement over the original MSER method, increasing the matching accuracy between 22% and 50% for the entire tested video sequence.

The True-positive usage results can be seen in Figure 5.69.a, only Super-features Inclusion based on SIFT features was able to provide a slight improvement of more than 100% True-positive usage for some of the tested frames. A large number of features were discarded to increase the matching and classification accuracy, some features were misclassified due to the large number of match outliers.

Even though the feature matching accuracy of the original feature detectors were low, the classification accuracy of the Super-feature algorithm remained high as seen in Figure 5.69.b. An average of 88% was achieved by the three feature detectors, with SIFT performing the most reliable with a minimum obtained classification accuracy of 88%. This resulted in an improvement of between 15% to 55% when Super-features was applied to the feature matches of these feature detection methods.

5.4.4.2 Market6 test sequence

Figure 5.70: Example frames from the market6 video sequence

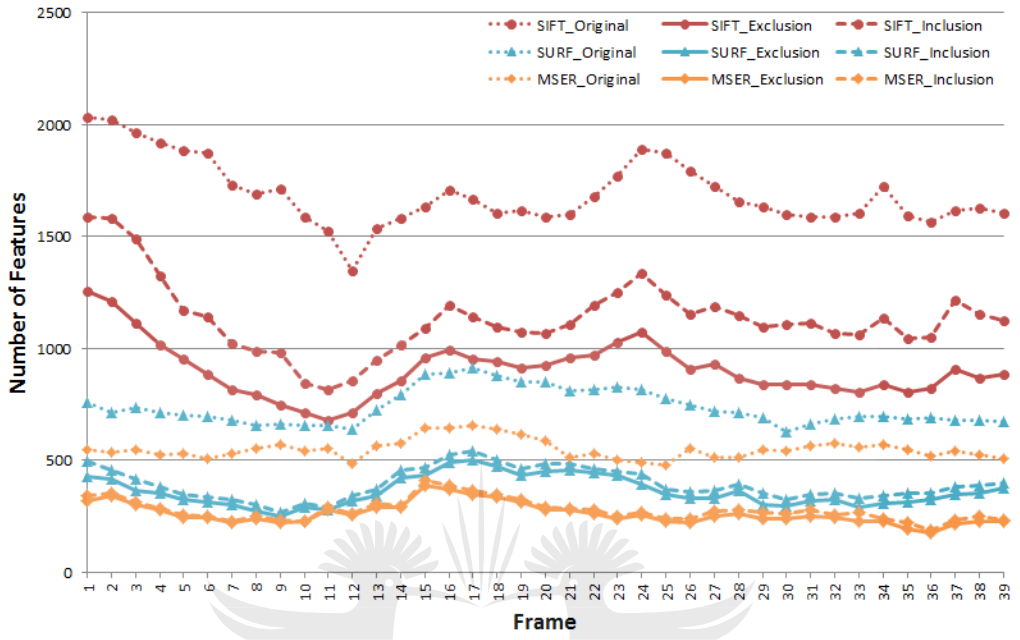
The Market6 test sequence contains fast forward motion, where objects that are far away contain less motion blurring, while object that are close contain large amounts of blurring. There is also a number of dynamic, non-rigid, rapid moving objects present, these objects can create occlusions of features. They also move into and out of shadows which could complicate the matching of the features located on these objects.

A dense set of features are located in this sequence, which should provide better support for the Super-features algorithm. The number of features detected by the different methods can be seen in Figure 5.71.a. There is some feature count variation between the different frames of the sequence for the different feature detection methods. MSER detected the least amount of features with 550 features on average, SURF detected slightly more with 736 features and SIFT provided the densest feature set with 1691 features on average.

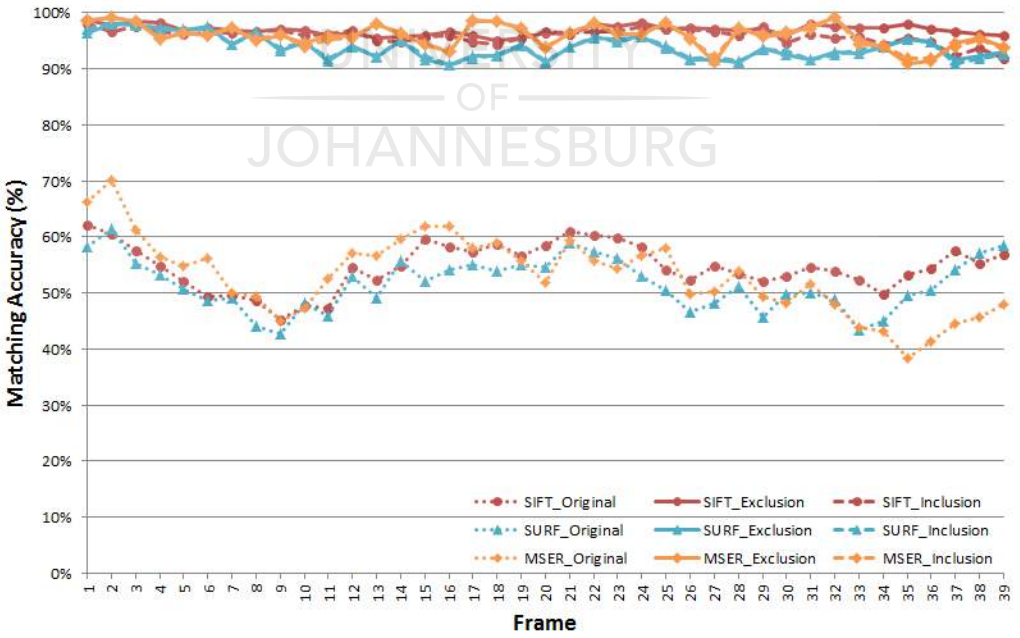
All three feature detection methods achieved a matching accuracy above 90%, after the Super-features algorithm was integrated, these results can be seen in Figure 5.71.b. The original feature detectors performed poorly and only had an average matching accuracy slightly above 50%. Due to the dense set of features provided by the original SIFT feature detector, it provided the most stable and consistent results between the tested feature detectors. MSER detected the least number of features, this resulted in a slight decrease in matching accuracy compared to SIFT and SURF. A large matching accuracy improvement can be observed when Super-features are used to improve the accuracy of SIFT, SURF and MSER for this extreme test sequence.

It can be seen from the True-positive usage results provided in Figure 5.72.a, that the density of the features used by the Super-feature algorithm affects the number of features that can be corrected and included in the matching process. SIFT which was the densest feature detector, combine with Super-features resulted in the maximum True-positive usage, ranging from 106% to 129%.

The classification accuracy results of the original feature detectors were between 23% and 48%, which was lower than their Super-feature counterparts. The classification results can be seen in Figure 5.72.b. The Super-feature based detectors provided consistent results between 85% and 98% even when the matching and classification accuracy of the original methods decreased. The SIFT based

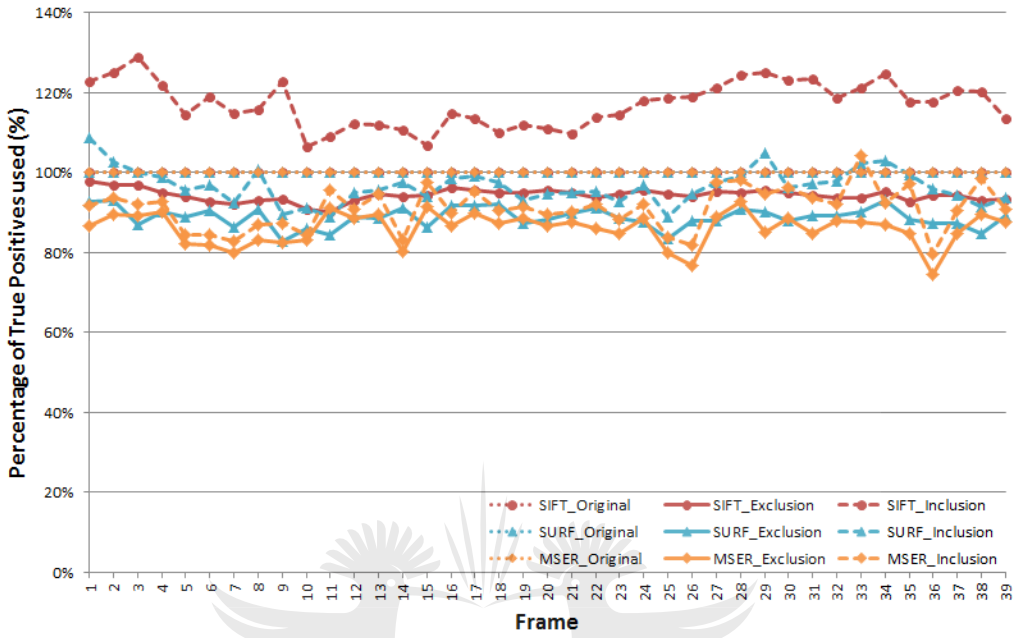


(a) Number of Features

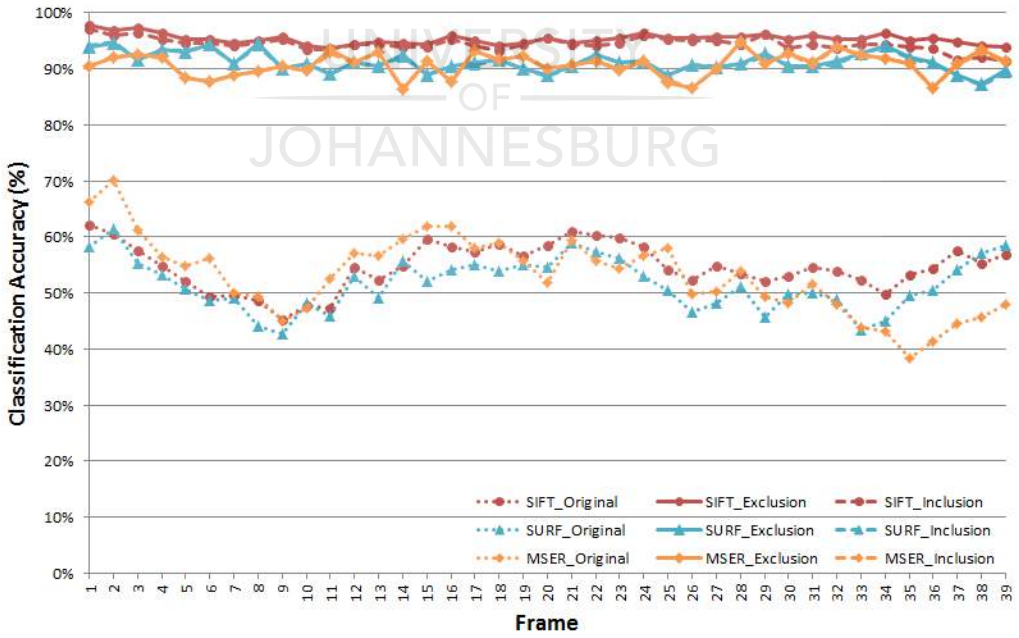


(b) Matching Accuracy

Figure 5.71: The number of features and matching accuracy results for the Market6 test sequence



(a) Percentage of True-positives used



(b) Classification Accuracy

Figure 5.72: The percentage of True-positives used and classification accuracy results for the Market6 test sequence

Super-feature algorithm provided the best classification results over the entire Market6 test sequence.

Chapter 6

Conclusion

The use of geometric consistency observed between different neighbouring features to improve feature matching, was investigated. It was believed that the geometric relationship between a selected primary feature and its neighbouring features in one coordinate frame can provide insight into the location of that feature in another, transformed coordinate frame. The estimated position determined by the geometric relationships allows the feature matching search space to be reduced, which will minimize the matching of invalid features produced by occlusions, duplicate features and poor feature detection repeatability. Further, the geometric relationships can also be used to correct previously mismatched features.

The Super-feature matching and correction framework was proposed, providing a robust probabilistic method for integrating feature appearance similarity and geometric consistency into the feature matching process in a translation, rotation, scale and affine invariant manner. Super-feature clusters were constructed for each feature using the geometric relationships provided by the neighbouring features of a selected primary feature. Each pair of neighbouring feature matches in a Super-feature and their corresponding transformed geometric relationships, provided a single solution to the primary feature's position in a new coordinate frame. These position estimates can be combined with kernel density estimation using Gaussian kernels to construct a probability density distribution. The modes with the highest probability responses in the probability density distribution are located using a Gaussian weighted Mean-shift algorithm. These detected position

estimates are selected as possible solutions to the primary feature's position in the new coordinate frame. A comparison between the dominant estimated positions and the matched position of the primary feature, allowed feature mismatches to be identified. Mismatched features can be corrected by finding the best feature match near the estimated position or creating a new feature to match to when a valid feature match is not available in the search region. Super-features can be used to improve feature matching even when a large number of features that form part of the Super-feature cluster have been matched incorrectly or are missing.

6.1 Overview of results

The ability of matching features using Super-features in the presence of rotation, scale and affine transformations was demonstrated for features detected using SIFT, SURF and MSER. The integration of the Super-feature algorithm produced a substantial improvement over the original feature detector counterparts, improving the matching results for increased and decreased scale and rotation changes. It also allowed features to be matched that were transformed by extreme affine transformations of up to 60 degrees. It was observed that the Super-feature algorithm preferred dense sets of features compared to sparse feature sets. A decrease in accuracy was observed when sparse sets of features were matched. Super-features from sparse feature sets still provided an accuracy improvement compared to the traditional feature detectors, but matching results were not as stable as when denser feature sets were used for testing.

A number of tests were performed to demonstrate the ability of Super-features to match features experiencing complex global as well as local transformations. Some of these feature sets contain multiple dynamic, non-rigid and morph-able objects where the motion model was unknown. Matching accuracies of close to 100% were achieved on the simpler matching problems such as scenes that contained dominant scene motion or motion on the foreground as well as the background. For feature sets with dynamic, non-rigid and morph-able objects and environments, a substantial increase of between 20% and 50% was observed for the Super-feature algorithms based on SIFT, SURF and MSER features. Fast ob-

ject and camera motion with motion blurring was the most difficult of the tested sequences. The blurring produced by the motion resulted in only sparse feature sets being detected. These features were also poorly described since the motion blurring affected the image regions used for describing the features. The blurring direction could also change between the different frames of the sequence, changing the feature description, which resulted in poor feature matches. Some improvement was observed compared to the original feature detection methods, but a large number of detected features were lost and could not be matched correctly. Overall, the results demonstrated that matching of features using Super-features produced an improvement to the matching accuracy and feature usage compared to the traditional algorithms over the entire range of test sequences, which ranged from simple to complex scenes.

6.2 Contributions

The contributions made in this work can be summarized as follows:

6.2.1 An efficient method of determining the geometric relationships between different features

An efficient method of constructing the geometric relationships observed between neighbouring features, was proposed using only feature position and rotation information. This method of calculating the geometric relationships was shown to be invariant to translation, rotation and scale changes and provided a large degree of affine invariance when multiple geometric relationships from neighbouring features were combined. Due to the limited feature information required to construct the geometric relationships, this method can be applied in conjunction with many state-of-the-art feature detectors that provide as a minimum, feature position and rotation information.

6.2.2 A reliable method of classifying valid and invalid feature matches without prior knowledge of the motion model

A method of determining and integrating the geometric relationships observed between neighbouring feature matches clustered into a Super-feature to enable the estimation of a selected feature's position in another transformed coordinate frame, was presented. This technique enabled feature matches to be corrected and classified as valid or invalid matches without prior knowledge of the motion models present in the feature sets, which allowed features to be matched using visual similarity as well as geometric consistency, enabling stronger and more reliable feature matches to be established. Since no prior motion model is required, this technique can be used to improve feature matching when global and local motion is present between multiple moving objects, as well as where some objects are dynamic, non-rigid and morph-able between the different feature sets.

6.2.3 A technique of improving invalid feature matches by estimating the true matching position

An efficient and robust technique of combining the weak position estimates produced by the observed geometric relationships in a Super-feature cluster, was provided. A probability density function of each feature's matching position was constructed allowing the locations with the highest probabilities to be used to improve invalid feature matches. Its successful application to SIFT, SURF and MSER features was demonstrated, allowing their matching accuracy to be improved in the presence of weak descriptions, poor feature detection repeatability, occlusions, duplicate features, illumination changes, shadows, image noise, atmospheric effects and distortions produced by extreme scale, rotation and affine transformations.

6.2.4 Accelerating the process of finding the extrema in a probability density distribution

Originally, a probability density function needed to be constructed for each feature that needed to be matched in a feature set, using the kernel density estimation technique. This was a computationally complex problem, as the probability distribution had to be sampled and discretized to a 2-dimensional grid that had to be fine-grained enough to allow the required position estimation accuracy to be achieved. A new probabilistic position estimation framework was introduced that made use of a Gaussian weighted Mean-shift algorithm to find the largest extrema of the underlying probability density function, without physically sampling the probability distribution. This enabled the geometric relationships observed in each Super-feature to be combined in an efficient and effective manner.

6.2.5 Using optical flow datasets for the evaluation and testing of feature detectors

Super-features can be used in conjunction with many different feature detectors, each based on different techniques of detecting interest points or features. This made it difficult to establish a ground truth feature dataset that could be used for testing the different feature detectors, since a common set of features between the different feature detectors was highly unlikely. An experimental setup was required to allow different feature detectors based on different feature detection methodologies to be compared and tested using the same framework. For this reason the use of optical flow datasets were proposed, since they provide the ground truth motion of each individual pixel between the frames in a video sequence. It allowed a common testing framework to be established.

6.3 Future work

Since each feature cluster is constructed and matched independently, the Super-feature algorithm can be accelerated using parallel processing architectures such as Graphics processing units (GPU) and multi-core processors. A parallel processor implementation of the Super-feature algorithm will enable reliable feature matching for real-time image processing applications such as required by visual aided navigation and control, object recognition for robotic platforms and 3-dimensional reconstruction.

The Super-feature algorithm provides a method of providing multi-mode position estimates. Multiple possible solutions can occur when a feature is located close to overlapping objects, each with their own motion. The neighbouring features located on each of these objects provide a different solution to the selected feature's position. When only a single estimate is available, the algorithm is able to select it as the best possible solution to the position estimation problem. On the other hand, when multiple position estimates are available, the Super-feature algorithm can then use the different solutions to classify a feature as either a valid or invalid match, but cannot correct the feature match, since determining which of the possible solutions is correct, is a difficult problem. A future investigation for correcting features with multi-mode solutions are required.

The current generation of feature detectors were developed without the availability of Super-features. These feature detectors use large feature descriptors and provide sparse feature sets in an attempt to improve feature matching reliability. A new computationally efficient feature detector can be designed and developed to work with and take advantage of the Super-feature algorithm's feature correction ability. The new feature detector can make use of less complex feature descriptors and should provide a dens set of features. Alternative methods of determining sub-pixel localization and feature orientation can be derived from the Super-feature cluster rather than the local image region surrounding a feature. This could improve the computational complexity and increase reliability, as information from a larger region is used to determine the feature's attributes.

Another possible future improvement is that the current Super-feature algorithm can only classify and correct the position of invalidly matched features.

Sometimes a valid feature could have been detected at the correct location but its attributes such as scale and rotation, which would influence the feature's description, could have been incorrectly calculated. More intelligent systems can be developed for the Super-feature cluster to enable it to correct the rotation and scale of features when a problem was encountered and a possible feature match was found at the correct location but a valid match could not be established.

Appendix

.1 Parameter selection results and videos

	SIFT
Original feature matches	Video
Super-feature exclusion matches	Video
Super-feature exclusion and inclusion matches	Video
Matlab figures and MAT-file of results	Folder

Table 1: Selection of the Gaussian sigma parameter

	SIFT
Original feature matches	Video
Super-feature exclusion matches	Video
Super-feature exclusion and inclusion matches	Video
Matlab figures and MAT-file of results	Folder

Table 2: Selection of the number of neighbours parameter

	SIFT
Original feature matches	Video
Super-feature exclusion matches	Video
Super-feature exclusion and inclusion matches	Video
Matlab figures and MAT-file of results	Folder

Table 3: Selection of the number of iterations parameter

	SIFT
Original feature matches	Video
Super-feature exclusion matches	Video
Super-feature exclusion and inclusion matches	Video
Matlab figures and MAT-file of results	Folder

Table 4: Selection of the weight threshold parameter

.2 Global image motion transformation results

.2.1 Rotation transformation invariance results and videos

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 5: Rotation transformation invariance tests for MSER based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 6: Rotation transformation invariance tests for SIFT based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 7: Rotation transformation invariance tests for SURF based Super-features

.2.2 Scale transformation invariance results and videos

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 8: Scale transformation invariance tests for MSER based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 9: Scale transformation invariance tests for SIFT based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 10: Scale transformation invariance tests for SURF based Super-features

.2.3 Affine transformation invariance results and videos

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 11: Affine transformation invariance tests for MSER based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 12: Affine transformation invariance tests for SIFT based Super-features

	Aircraft	Bridge	Nature
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 13: Affine transformation invariance tests for SURF based Super-features

.3 Complex global and local motion transformations results and videos

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 14: The results and videos for the Alley1 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 15: The results and videos for the Alley2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 16: The results and videos for the Ambush2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 17: The results and videos for the Ambush4 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 18: The results and videos for the Ambush5 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 19: The results and videos for the Ambush6 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 20: The results and videos for the Bamboo1 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 21: The results and videos for the Bamboo2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 22: The results and videos for the Bandage1 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 23: The results and videos for the Bandage2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 24: The results and videos for the Cave2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 25: The results and videos for the Cave4 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 26: The results and videos for the Market2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 27: The results and videos for the Market5 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 28: The results and videos for the Market6 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 29: The results and videos for the Mountain1 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 30: The results and videos for the Shaman2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 31: The results and videos for the Shaman3 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 32: The results and videos for the Sleeping1 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 33: The results and videos for the Sleeping2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 34: The results and videos for the Temple2 test sequence

	MSER	SIFT	SURF
Original feature matches	Video	Video	Video
Super-feature exclusion matches	Video	Video	Video
Super-feature exclusion and inclusion matches	Video	Video	Video
Matlab figures and MAT-file of results	Folder	Folder	Folder

Table 35: The results and videos for the Temple4 test sequence

References

[1] M.M. Nass and L.N. Cooper, "A Theory for the Development of Feature Detecting Cells in Visual Cortex", Biological Cybernetics, ISSN: 0340-1200, Vol 19, Iss 1, pp 1-18, 1975.

[2] A.S. Darmaillacq, L. Dickel and J. Mather, "Cephalopod Cognition", Cambridge University Press, ISBN-10: 1107015561, Edition 1, pp 10-19, 2014.

[3] S. Cannicci, L. Morino and M. Vannini, "Behavioural evidence for visual recognition of predators by the mangrove climbing crab Sesarma leptosoma", Animal behaviour, Vol 63, No 1, pp 77-83, 2002.

[4] I. Biederman, "Recognition-by-components: A theory of human image understanding", Psychological Review, Vol 2, No 94, pp 115-147, 1987.

[5] D.G. Lowe, "Object Recognition from Local Scale-Invariant Features", IEEE International Conference on Computer Vision (ICCV), Vol 2, pp 1150-1157, 1999.

[6] M. Kamath, D. Punjabi, T. Sabnis, D. Upadhyay and S. Shrawne, "Improving Content Based Image Retrieval using Scale Invariant Feature Transform", International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249-8958, Vol 1, Iss 5, 2012.

[7] N. Nagaraju, T.S. Savitri and C.A. Swamy, "Image Registration Using Scale Invariant Feature Transform", International Journal of Scientific Engineering and Technology (IJSET), ISSN: 2277-1581, Vol 2, Iss 7, pp 675-680, 2013.

[8] C. Liu, J. Yuen and A. Torralba, "SIFT Flow: Dense correspondence across scenes and its applications", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 33, Iss 5, pp 978-994, 2011.

[9] H. Joshi and K. Sinha, "Image Mosaicing using Harris, SIFT Feature Detection Algorithm", International Journal of Science, Engineering and Technology Research (IJSETR), ISSN: 2278-7798, Vol 2, Iss 11, 2013.

[10] M. Brown and D.G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features", International Journal of Computer Vision (IJCV), Vol 74, No 1, pp 59-77, 2007.

[11] N. Karlsson, E.D. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian and M.E. Munich, "The vSLAM Algorithm for Robust Localization and Mapping", IEEE International Conference on Robotics and Automation (ICRA), pp 24-29, 2005.

[12] S. Battiato, G. Gallo and G. Puglisi, "Fuzzy-Based Motion Estimation for Video Stabilization Using SIFT Interest Points", International Society for Optics and Photonics (SPIE), Vol 7250, 2009.

[13] E. Vural and A.A. Alatan, "Outlier Removal for Sparse 3D Reconstruction from Video", 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, pp 341-344, 2008.

[14] B. Clipp, C. Zach, J. Lim, J.M. Frahm and M. Pollefeys, "Adaptive, real-time visual simultaneous localization and mapping", IEEE Workshop on Applications in Computer Vision (WACV), ISSN: 1550-5790, pp 1-8, 2009.

[15] F.L. Wendt, S. Bres, B. Tellez and R. Laurini, "Markerless outdoor localisation based on SIFT descriptors for mobile applications", International Conference on Image and Signal Processing (ICISP), pp 439-446, 2008.

[16] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi and S. Tubaro, "Coding visual features extracted from video sequences", IEEE Transactions on Image Processing, Vol 23, No 5, pp 2262-2276, 2014.

[17] K. Lai, L. Bo, X. Ren and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset", IEEE International Conference on Robotics and Automation (ICRA), pp 1817-1824, 2011.

[18] D.G. Lowe, "Distinctive Image Features from Scale Invariant Key-points", International Journal of Computer Vision (IJCV), Vol 60, No 2, pp 91-110, 2004.

[19] H. Bay, A. Ess, T. Tuytelaars and L. van Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol 110, No 3, pp 346-359, 2008.

[20] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions", British Machine Vision Conference (BMVC), pp 384-396, 2002.

[21] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection", European Conference on Computer Vision (ECCV), Vol 1, pp 430-443, 2006.

[22] M. Calonder, V. Lepetit, C. Strecha and P. Fua, "BRIEF: Binary Robust Independent Elementary Features", European Conference on Computer Vision (ECCV), pp 778-792, 2010.

[23] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "ORB: an efficient alternative to SIFT or SURF", IEEE International Conference on Computer Vision (ICCV), pp 2564-2571, 2011.

[24] S. Leutenegger, M. Chli and R.Y. Siegwart,"BRISK: Binary Robust Invariant Scalable Keypoints", IEEE International Conference on Computer Vision (ICCV), ISSN: 1550-5499, pp 2548-2555, 2011.

[25] A. Morales, M.A. Ferrer, M. Diaz-Cabrera and E. Gonzalez, "Analysis of local descriptors features and its robustness applied to ear recognition", IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1-5, 2013.

[26] D.G. Lowe, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image", U.S. Patent 6,711,293, Issued March 23, 2004.

[27] S. Battiato, G. Gallo, G. Puglisi and S. Scellato, "SIFT features tracking for video stabilization. In Image Analysis and Processing", IEEE International Conference on Image Analysis and Processing (ICIAP), pp 825-830, 2007.

[28] A. Hevner and S. Chatterjee, "Design science research in information systems", Integrated Series in Information Systems, Springer, Vol 22, 2010.

[29] R.H. von Alan, S.T. March, J. Park and S. Ram, "Design science in information systems research", MIS quarterly, Vol 28, No 1, pp 75-105, 2004.

[30] S.T. March and G.F. Smith, "Design and natural science research on information technology", Decision support systems, Elsevier, Vol 15, No 4, pp 251-266, 1995.

[31] R.A. Hevner, "A three cycle view of design science research", Scandinavian journal of Information Systems, Vol 19, No 2, 2007.

[32] M. Nixon, "Feature Extraction and Image Processing for Computer Vision", Academic Press, ISBN-10: 0123965497, pp 152-167, 2008.

[33] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors", International Journal of Computer Vision (IJCV), Vol 60, No 1, pp 63-86, 2004.

[34] H. Zhou, Y. Yuan and C. Shi, "Object tracking using SIFT features and mean shift", Computer vision and image understanding, Vol 113, No 3, pp 345-352, 2009.

[35] N.Y. Khan, B. McCane and G. Wyvill, "SIFT and SURF performance evaluation against various image deformations on benchmark dataset", IEEE International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp 501-506, 2011.

[36] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 27, No 10, pp 1615-1630, 2005.

[37] J.Luo and O. Gwun, "A comparison of SIFT, PCA-SIFT and SURF", International Journal of Image Processing (IJIP), Vol 3, No 4, pp 143-152, 2009.

[38] J. Krizaj, V. Struc and N. Pavesic, "Adaptation of SIFT features for face recognition under varying illumination", International convention on information and communication technology, electronics and microelectronics (MIPRO), pp 691-694, 2010.

[75] M. Brown and D.G. Lowe, "Unsupervised 3D object recognition and reconstruction in unordered datasets", IEEE International Conference on 3-D Digital Imaging and Modeling (3DIM), pp 56-63, 2005.

[40] H. Ling, Y. Wu, E. Blasch, G. Chen, H. Lang and L. Bai, "Evaluation of visual tracking in extremely low frame rate wide area motion imagery", IEEE International Conference on Information Fusion (FUSION), pp 1-8, 2011.

[41] D. Marr and E. Hildreth, "Theory of Edge Detection", Proceedings of the Royal Society of London, Biological Sciences, Vol 207, No 1167, pp 187-217, 1980.

[42] J.F. Canny, "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 8, pp 679-698, 1986.

[43] H.P. Moravec, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover", Tech Report CMU-RI-TR-3, Carnegie-Mellon University, Robotics Institute, 1980.

[44] W. Forstner, "A feature based correspondence algorithm for image matching", The International Society for Photogrammetry and Remote Sensing (ISPRS), Vol 26, pp 150-166, 1986.

[45] C. Harris and M. Stephens, "A combined corner and edge detector", Alvey Vision Conference (AVC88), pp 147-151, 1988.

[46] Y.E. Peng and W. Yan, "An improved harris multi-scale corner detection", Computer Technology and Development, Vol 4, No 4, pp 58-61, 2010.

[47] J. Shi and C. Tomasi, "Good features to track", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp 593-600, 1994.

[48] S.M. Smith and J.M. Brady, "SUSAN - a new approach to low level image processing", International Journal of Computer Vision (IJCV), Vol 23, No 1, pp 45-78, 1997.

[49] Y. Xingfang, H. Yumei and L. Yan, "An improved SUSAN corner detection algorithm based on adaptive threshold", International Conference on Signal Processing Systems (ICSPS), Vol 2, pp 613-616, 2010.

[50] E. Rosten, R. Porter and T. Drummond, "Faster and better: A machine learning approach to corner detection", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 32, pp 105-119, 2010.

[51] E. Mair, G.D. Hager, D. Burschka, M. Suppa and G. Hirzinger, "Adaptive and generic corner detection based on the accelerated segment test", European Conference on Computer Vision (ECCV), pp 183-196, 2010.

[52] Y. Dufournaud, C. Schmid and R. Horaud, "Matching images with different resolutions", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 1, pp 612-618, 2000.

[53] T. Lindeberg, "Scale-Space Theory in Computer Vision", Springer, ISBN: 0-7923-9418-6, 1993.

[54] T. Lindeberg, "Feature Detection with Automatic Scale Selection", International Journal of Computer Vision (IJCV), Vol 30, No 2, pp 79-116, 1998.

[55] J.L. Crowley and A.C. Parker, "A representation for shape based on peaks and ridges in the difference of low pass transform", IEEE Transactions on

Pattern Analysis and Machine Intelligence (TPAMI), Vol 6, Iss 2, pp 156-170, 1984.

[56] Y. Ke and R. Sukthankar, "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 2, pp 506-513, 2004.

[57] J.M. Geusebroek, R. van den Boomgaard, A.W.M. Smeulders and H. Geerts, "Color invariance", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 23, Iss 12, pp 1338-1350, 2001.

[58] A.E. Abdel-Hakim and A.A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics", IEEE Computer Society Conference on Computer Vision (CVPR), Vol 2, pp 1978-1983, 2006.

[59] A. Bosch, A. Zisserman and X. Munoz, "Image classification using random forests and ferns", IEEE International Conference on Computer Vision (ICCV), pp 1-8, 2007.

[60] J.M. Morel and G. Yu, "ASIFT: a new framework for fully affine invariant image comparison", SIAM Journal on Imaging Sciences (SIIMS), Vol 2, No 2, pp 438-469, 2009.

[61] Z. Wang, B. Fan and F. Wu, "Local Intensity Order Pattern for Feature Description", IEEE International Conference on Computer Vision (ICCV), pp 603-610, 2011.

[62] Y. Roodt, A.L. Nel and W.A. Clarke, "NIFT: Noise Invariant Feature Transform", University of Johannesburg, 2012 .

[63] E. Tola, V. Lepetit and P. Fua, "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 32, Iss 5, pp 815-830, 2010.

[64] J. Cronje, "BFROST: binary features from robust orientation segment tests accelerated on the GPU", Annual Symposium of the Pattern Recognition Society of South Africa (PRASA), pp 25-30, 2011.

[65] A. Alahi, R. Ortiz and P. Vandergheynst, "FREAK: Fast Retina Keypoint", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 510-517, 2012.

[66] E.N. Mortensen, H. Deng and L. Shapiro, "A SIFT descriptor with global context", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 1, pp 184-190, 2005.

[67] W. Kai, C. Bo and T. Long, "An Improved SIFT Feature Matching Algorithm Based on Maximizing Minimum Distance Cluster", International Conference on Computer Science and Information Technology (ICCSIT), Vol 51, pp 255-259, 2012.

[68] O. Chum and J. Matas, "Two-View Geometry Estimation by Random Sample and Consensus", Center for Machine Perception, Czech Technical University in Prague, 2005.

[69] A. Hast, J. Nysjo and A Marchetti, "Optimal RANSAC Towards a Repeatable Algorithm for Finding the Optimal Set", Journal of WSCG, ISSN 1213-6972, Vol 21, No 1, pp 21-30, 2013.

[70] J.C. Bazin, H. Li, I.S. Kweon, C. Demonceaux, P. Vasseur and K. Ikeuchi, "A Branch-and-Bound Approach to Correspondence and Grouping Problems", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 35, Iss 7, pp 1565-1576, 2013.

[71] L. Torresani, V. Kolmogorov and C. Rother, "A Dual Decomposition Approach to Feature Correspondence", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 35, Iss 2, pp 259-271, 2012.

[72] H. Chen, Y. Lin, and B. Chen, "Robust Feature Matching with Alternate Hough and Inverted Hough Transforms", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2762-2769, 2013.

[73] H. Li, E. Kim, X. Huang and L. He, "Object matching with a locally affine-invariant constraint", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1641-1648, 2010.

[74] K. Grauman and T. Darrell, "Efficient image matching with distributions of local invariant features", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 2, pp 627-634, 2005.

[75] M. Brown and D.G. Lowe, "Invariant Features from Interest Point Groups", British Machine Vision Conference (BMVC), No. S 1, 2002.

[76] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition", IEEE Transactions on Information Theory, Vol 21, No 1, pp 32-40, 1975.

[77] R. Subbarao and P. Meer, "Nonlinear mean shift over Riemannian manifolds", International Journal of Computer Vision (IJCV), Vol 84, No 1, pp 1-20, 2009.

[78] M. Ciollaro, C. Genovese, J. Lei and L. Wasserman, "The functional mean-shift algorithm for mode hunting and clustering in infinite dimensions", Journal of Machine Learning Research (JMLR), 2014.

[79] G. Wu, X. Zhao, S. Luo and H. Shi, "Histological image segmentation using fast mean shift clustering method", Biomedical Engineering Online, Vol 14, No 1, 2015.

[80] J.M. Saragih, S. Lucey and J.F. Cohn, "Face alignment through subspace constrained mean-shifts", IEEE International Conference on Computer Vision (ICCV), pp 1034-1041, 2009.

[81] B. Georgescu, I. Shimshoni and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example", IEEE International Conference on Computer Vision (ICCV), pp 456-463, 2003.

[82] S. Paris and F. Durand, "A topological approach to hierarchical segmentation using mean shift", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1-8, 2007.

[83] M.A. Carreira-Perpinan, "Acceleration strategies for Gaussian mean-shift image segmentation", IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol 1, pp 1160-1167, 2006.

[84] A. Yilmaz, "Object tracking by asymmetric kernel mean shift with automatic scale and orientation selection", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1-6, 2007.

[85] C. Yang, R. Duraiswami and L. Davis, "Efficient mean-shift tracking via a new similarity measure", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 1, pp 176-183, 2005.

[86] Y. Cheng, "Mean shift, mode seeking, and clustering", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 17, No 8, pp 790-799, 1995.

[87] D. Comaniciu and P. Meer, "Mean shift analysis and applications", IEEE International Conference on Computer Vision (ICCV), Vol 2, pp 1197-1203, 1999.

[88] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), Vol 24, No 5, pp 603-619, 2002.

[89] W. Tao, H. Jin and Y. Zhang, "Color image segmentation based on mean shift and normalized cuts", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, Vol 37, No 5, pp 1382-1389, 2007.

[90] D. Comaniciu, R. Visvanathan and P. Meer, "Real-time tracking of non-rigid objects using mean shift", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol 2, pp 142-149, 2000.

[91] D.E. Knuth, "The Art of Computer Programming", Addison-Wesley Professional, ISBN-10: 0201896850, Vol 3, pp 138141, 1997.

[92] E.W. Weisstein, "Line-Line Intersection", Wolfram MathWorld, [Online]. Available: http://mathworld.wolfram.com/Line-LineIntersection.html accessed on 2015/04/20.

[93] F. Antonio, "Faster line segment intersection", Graphics Gems III, Academic Press Professional, ISBN-10: 0124096700, No 3, pp 199-202, 1992.

[94] D.J. Butler, J. Wulff, G.B. Stanley and M.J. Black, "A naturalistic open source movie for optical flow evaluation", European Conference on Computer Vision (ECCV), pp 611-625, 2012.

[95] "Detect SURF features and return SURFPoints object", The MathWorks, [Online]. Available: http://www.mathworks.com/help/vision/ref/detectsurffeatures.html accessed on 2013/03/17.

[96] "Detect MSER features and return MSERRegions object", The MathWorks, [Online]. Available: http://www.mathworks.com/help/vision/ref/detectmserfeatures.html accessed on 2013/03/17.

[97] D. Lowe, "Demo Software: SIFT Keypoint Detector", University of British Columbia, [Online]. Available: http://www.cs.ubc.ca/ lowe/keypoints/index.html accessed on 2015/03/14.

[98] Bridge test image, [Online]. Available: http://host2post.com/server13/photos/MyDE8tr_bmRCSM~/stunning-italy-pictures-experience.jpg accessed on 2015/03/23.

[99] Nature test image, [Online]. Available: http://www.youramazingplaces.com/wp-content/uploads/2013/03/glacier-national-park-1.jpg accessed on 2015/03/23.

[100] Aircraft test image, [Online]. Available: http://www.aircraftinformation.info/Images/Tornado_01.jpg accessed on 2015/03/23.

[101] J. Wulff, D.J. Butler, G.B. Stanley and M.J. Black, "Lessons and insights from creating a synthetic optical flow benchmark", European Conference on Computer Vision (ECCV), pp 168-177, 2012.