# Acquiring 3D scene information from 2D images

Acquiring 3D scene information from 2D images

# Acquiring 3D scene information from 2D images

**PROEFSCHRIFT**

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 31 oktober 2011 om 14.00 uur

door

Ping Li

geboren te Wuyi, China

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. P.H.N. de With

Copromotor:
dr.ir. P. Vandewalle

---

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Ping Li

Acquiring 3D scene information from 2D images / by Ping Li. - Eindhoven : Technische Universiteit Eindhoven, 2011.
A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-2739-7
NUR 959
Subject headings: computer vision / 3D reconstruction / 3DTV / depth estimation / feature point matching / critical configuration / image processing

# Acquiring 3D scene information from 2D images

**Ping Li**

Committee:

| | |
|---|---|
| prof.dr.ir. P.H.N. de With | Eindhoven University of Technology, The Netherlands |
| dr.ir. P. Vandewalle | Philips Research |
| prof.dr.ir. R.L. Lagendijk | Delft University of Technology, The Netherlands |
| prof.dr.ir. C.H. Slump | University of Twente, The Netherlands |
| prof.dr.ir. J.J. Lukkien | Eindhoven University of Technology, The Netherlands |
| prof.dr.ir. G. de Haan | Eindhoven University of Technology, The Netherlands |

The publication of this work has been kindly sponsored by:



The work described in this thesis has been supported by the Dutch Freeband I-Share project in the BSIK framework on sharing resources in virtual communities for storage, communications, and processing of multimedia data.

Cover design: Paul Verspage
Cover illustration: Paul Verspage

# Summary

In recent years, people are becoming increasingly acquainted with 3D technologies such as 3DTV, 3D movies and 3D virtual navigation of city environments in their daily life. Commercial 3D movies are now commonly available for consumers. Virtual navigation through our living environment has become a reality on computers, enabled by well-known web-based geographic applications based on advanced imaging technologies. To enable such 3D applications, many technological challenges such as 3D content creation, 3D displaying technology and 3D content transmission need to be addressed, developed and deployed at low cost. This thesis concentrates on the reconstruction of 3D scene information from multiple 2D images, aiming at an automatic and low-cost production of 3D visual content.

In this thesis, two multiple-view 3D reconstruction systems are proposed: a *3D modeling system* for reconstructing the sparse 3D scene model from long video sequences captured with a hand-held consumer camcorder, and a *depth reconstruction system* for creating depth maps from multiple-view videos taken by multiple synchronized cameras. Both systems are designed to compute the 3D scene information in an automated way with minimum human interventions, in order to reduce the production cost of 3D content. Experimental results on real videos of hundreds and thousands frames have shown that the two systems are able to accurately and automatically reconstruct the 3D scene information from 2D image data. The findings of this research are useful for emerging 3D applications such as 3D games, 3D visualization and 3D content production.

Apart from the design and implementation of the two proposed systems, we have developed three key scientific contributions to execute the two 3D reconstruction systems. The first contribution is that we have designed a novel feature point matching algorithm, only based on a smoothness constraint for matching the points. The constraint states that neighboring feature points in images tend to move with similar directions and magnitudes. The employed smoothness assumption is not only valid but also robust for most images with limited image motion, regardless of the camera motion and scene structure. As a result, the algorithm obtains two major advantages. First, it is robust to illumination changes, as the employed smoothness constraint does not rely on any texture information. Second, the algorithm has a good capability to handle the drift of the feature points over time, since the drift can hardly lead to a violation of the smoothness constraint. This leads to a large number of feature points matched and tracked by the proposed algorithm, which significantly helps the subsequent 3D modeling process. Our feature point matching algorithm is specifically designed

for matching and tracking feature points in image/video sequences where the image motion is limited. Our extensive experimental results show that the proposed algorithm is able to track at least 2.5 times the amount of feature points as produced by state-of-the-art algorithms, with a comparable or higher accuracy. This contributes significantly to the robustness of the 3D reconstruction process.

The second contribution is that we have developed algorithms to detect critical configurations where the factorization-based 3D reconstruction degenerates. Based on the detection, we have proposed a sequence-partitioning algorithm to divide a long sequence into subsequences, such that successful 3D reconstructions can be performed on individual subsequences with a high confidence. The partial reconstructions are merged later to obtain the 3D model of the complete scene. In the critical configuration detection algorithm, four critical configurations are detected: (1) coplanar 3D scene points, (2) pure camera rotation, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. The configurations in cases (1), (2) and (4) will affect the rank of the Scaled Measurement Matrix (SMM). The number of camera centers in case (3) will affect the number of independent rows within the SMM. By examining the rank and the row space of the SMM, the above-mentioned critical configurations are detected. Based on the detection results, the proposed sequence-partitioning algorithm divides a long sequence into subsequences, such that each subsequence is free of the four critical configurations, in order to obtain successful 3D reconstructions on individual subsequences. Experimental results on both synthetic and real sequences have demonstrated that the above four critical configurations are robustly detected, and a long sequence of thousands frames is automatically divided into subsequences, yielding successful 3D reconstructions. Experiments have shown that both critical configuration detection and sequence-partitioning algorithms have been found essential for an automatical 3D reconstruction on long sequences.

The third contribution is that we have proposed a coarse-to-fine multiple-view depth labeling algorithm to compute depth maps from multiple-view videos, where the accuracy of the resulting depth maps is gradually refined in multiple optimization passes. In the proposed algorithm, multiple-view depth reconstruction is formulated as an image-based labeling problem, using the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF). The MAP-MRF framework allows the combination of various objective and heuristic depth cues to define the local penalty and the interaction energies, which provides a straightforward and computationally tractable formulation. Furthermore, the global optimal MAP solution to depth labeling can be found by minimizing the local energies, using existing MRF optimization algorithms. The proposed algorithm contains the following three key contributions. First, a graph construction algorithm is proposed to create triangular meshes on over-segmentation maps, in order to exploit the color and the texture information for depth labeling. Second, multiple depth cues are combined to define the local energies. Furthermore, the local energies are adapted to the local image content, in order to consider the varying nature of the image content for an accurate depth labeling. Third, both the density of the graph nodes and the intervals of the depth labels are gradually refined in multiple labeling passes. By doing so, both the computational efficiency and the robustness of the depth labeling process are improved. The experimental results on real multiple-view videos show that

the depth maps of selected reference views are accurately reconstructed. Depth discontinuities are quite well preserved, so that the geometric reconstruction on the edges of objects is improved perceptually.

# Samenvatting

In de afgelopen jaren zijn mensen in hun dagelijkse leven geleidelijk vertrouwd geraakt met 3D-technologieën zoals 3D-TV, 3D-films en 3D-virtuele-navigatie door stedelijke gebieden. Commerciële 3D-films zijn nu algemeen beschikbaar voor de consument. Virtuele navigatie door onze leefomgeving is nu beschikbaar op computers met behulp van gangbare Internet gebaseerde geografische toepassingen, waarbij gebruik wordt gemaakt van geavanceerde imaging technologieën. Voor deze 3D-toepassingen moeten veel technologische uitdagingen worden opgelost, ontwikkeld en met lage kosten worden gebruikt, zoals creatie van 3D-content, 3D-displays en transmissie van 3D-informatie. Dit proefschrift concentreert zich op de reconstructie van 3D-scène informatie uit verscheidene 2D-beelden (multi-view) met als doelstelling een automatische en kostenefficiënte productie van 3D-beelddata.

In dit proefschrift worden twee multi-view 3D-reconstructiesystemen gepresenteerd: een *3D-modelleringssysteem* voor de reconstructie van een uit schaarse kernpunten gebouwd 3D-scène-model (sparse) van lange videosequenties die zijn opgenomen met een digitale videocamera, en een *diepte-reconstructiesysteem* voor het creëren van dieptebeelden van multi-view video's opgenomen door verscheidene gesynchroniseerde camera's. Beide systemen zijn ontworpen om de 3D-scne-informatie automatisch te berekenen met minimale menselijke interventie, om de productiekosten van 3D-inhoud te reduceren. Experimentele resultaten met normale videosequenties van honderden en duizenden beelden hebben aangetoond dat de twee systemen nauwkeurig en automatisch de 3D-scène-informatie uit 2D-beelden kunnen reconstrueren. Dit onderzoeksresultaat is nuttig voor opkomende 3D-toepassingen zoals 3D-games, 3D-visualisatie en productie van 3D-beeldmateriaal.

Naast het ontwerp van de twee reconstructiesystemen, beschrijft het proefschrift drie belangrijke wetenschappelijke bijdragen voor de executie van de twee reconstructiesystemen. De eerste bijdrage is het ontwerp van een nieuw feature-punt matching-algoritme, gebaseerd op slechts een beperking in de bewegingsuniformiteit (smoothness). De beperking gebruikt de aanname dat naburige feature-punten in beelden de neiging hebben om met vergelijkbare richtingen en amplitudes te bewegen. De smoothness-aanname is niet alleen geldig maar ook robuust voor de meeste beelden met een beperkte beweging, onafhankelijk van de camerabeweging en scènestructuur. Het algoritme verkrijgt hierdoor twee grote voordelen. Ten eerste is het robuust voor intensiteitsveranderingen omdat de gebruikte smoothness-aanname niet afhankelijk is van textuurinformatie. Ten tweede kan het algoritme omgaan met de drift van de feature- punten die na verloop van tijd optreedt, omdat de drift vrijwel altijd voldoet aan de smoothness-aanname. Het algoritme zorgt er dus voor dat een groot aantal feature-punten

correspondenties hebben die daarnaast gevolgd kunnen worden in de tijd, hetgeen aanzienlijk helpt bij het daaropvolgende 3D-modelleringsproces. Het feature-punt matching-algoritme is speciaal ontworpen voor het vinden van correspondenties en bijhouden van feature-punten in beeld- en videosequenties met beperkte beeldbeweging. De uitgebreide experimentele resultaten tonen aan dat het ontworpen algoritme in staat is om op zijn minst 2,5 keer het aantal feature-punten te volgen vergeleken met het aantal feature-punten gevonden door de huidige alternatieve algoritmen, met een vergelijkbare of hogere nauwkeurigheid. Het hoge aantal draagt aanzienlijk bij aan de robuustheid van het 3D-reconstructieproces.

De tweede bijdrage is de ontwikkeling van algoritmes om kritieke cameraconfiguraties te detecteren, waar de factorisatie-gebaseerde 3D-reconstructie degenereert tot een onnauwkeurig resultaat. Op basis van deze detectie gebruiken we een sequentie-partitionerings algoritme om een lange videosequentie te verdelen in deelsequenties, zodat succesvolle 3D-reconstructies uitgevoerd kunnen worden voor de individuele deelsequenties met een grote betrouwbaarheid. De partiële 3D-reconstructies worden later samengevoegd tot het 3D-model van de volledige scènesequentie. Het detectie-algoritme identificeert vier kritieke configuraties: (1) coplanaire 3D-scènepunten, (2) zuivere camerarotatie, (3) rotatie om twee cameracentra, en (4) de aanwezigheid van extreme ruis en uitschieters in de metingen. De kritieke configuraties in de gevallen (1), (2) en (4) zullen de rang van de Scaled Measurement Matrix (SMM) beïnvloeden. Het aantal cameracentra in geval (3) is van invloed op het aantal onafhankelijke rijen van de SMM. Door inspectie van de rang en de rij-onafhankelijkheid van de SMM, worden de bovengenoemde kritieke configuraties gedetecteerd. Op basis van deze detectieresultaten zal het ontworpen sequentie-partitioneringsalgoritme een lange sequentie in deelsequenties verdelen, waardoor elke deelsequentie niet gebaseerd is op de vier kritieke configuraties, teneinde succesvolle 3D-reconstructies te verkrijgen voor de individuele deelsequenties. Experimenten met zowel synthetische en natuurlijke videosequenties hebben aangetoond dat de bovenstaande vier kritieke configuraties robuust worden gedetecteerd en een lange sequentie van duizenden beelden automatisch wordt verdeeld in deelsequenties, waardoor een succesvolle 3D-reconstructie resulteert. Experimenten hebben aangetoond dat zowel de detectie van kritieke configuraties als sequentie-partitioneringsalgoritmen van essentieel belang zijn voor een automatische 3D-reconstructie met lange sequenties.

De derde bijdrage omvat een grof-naar-fijn multi-view diepte-labelingsalgoritme om dieptebeelden te berekenen uit verschillende video's, waarbij de nauwkeurigheid van de resulterende dieptebeelden geleidelijk wordt verfijnd in verscheidene optimalisatiestappen. In dit labelingsalgoritme is multi-view diepte reconstructie geformuleerd als een beeld-gebaseerd labelingsprobleem dat gebruik maakt van het Maximale A Posterior (MAP) raamwerk afgebeeld op zogenaamde Markov Random Fields (MRF). Het MAP-MRF kader staat toe om een combinatie van verschillende objectieve en heuristische diepteaanwijzingen te gebruiken voor het definiëren van de lokale penalty en de interactie-energieën. Dit maakt de probleemformulatie eenvoudig en behandelbaar met betrekking tot de berekeningswijze. Bovendien kan de globale optimale MAP-oplossing voor de diepte-labeling worden gevonden door het minimaliseren van de lokale energieën, daarbij gebruikmakend van bestaande algoritmen voor MRF-optimalisatie. Het voorgestelde algoritme bevat de volgende drie kernbijdragen. Eerst wordt een graafconstructie-algoritme gepresenteerd om driehoekige mazen te maken

in overgesegmenteerde beelden, om de kleur- en textuurinformatie voor de diepte-labeling te kunnen gebruiken. Ten tweede worden verschillende diepte-aanwijzingen gecombineerd voor de definitie van de lokale energieën. Voor een nauwkeurige diepte-labeling worden bovendien de lokale energieën adaptief gemaakt aan de lokale beeldinhoud in verband met de variatie in beeldinhoud. Ten derde worden zowel de dichtheid van de knooppunten van de graaf als de intervallen van de dieptelabels geleidelijk verfijnd in verscheidene diepte-labeling stappen. Hierdoor worden zowel de berekeningsefficiëntie als de robuustheid van het diepte-labelingsproces verbeterd. De experimenten met natuurlijke multi-view videosequenties resulteren in een nauwkeurige reconstructie van de dieptebeelden op de geselecteerde referentiepunten voor de camera's. Discontinuïteiten in de dieptebeelden blijven goed bewaard, zodat de geometrische reconstructie op de randen van objecten perceptueel is verbeterd.

# Abbreviation list

| | |
|---|---|
| SaM | Structure and Motion |
| DBIR | Depth-Based Image Rendering |
| SMM | Scaled Measurement Matrix |
| MVV | Multiple-View Video |
| MVF | Multiple-View Frame |
| SAD | Sum of Absolute Difference |
| SVD | Singular Value Decomposition |
| CV | Correspondence Vector |
| MV | Matching Vector |
| TIFM | Texture-Independent Feature point Matching |
| MAP | Maximum A Posterior |
| MRF | Markov Random Fields |

# Contents

CHAPTER 1

# Introduction

*The creation of 3D scene information is a major recurring challenge in many 3D applications. This thesis attempts to acquire 3D scene information from multiple-view 2D images in an automated way while exploring two fundamental technologies. One is aiming at reconstructing a sparse 3D scene model from a long video sequence, and the other is aiming at creating depth maps from multiple-view video. This chapter first gives a broad introduction to 3D reconstruction and associate problems. Then we discuss the production of 3D contents in the case of free-viewpoint 3DTV, followed by a discussion on another possible application of a 3D virtual visualization of our living environment. After that, we present our research objectives and existing methods for computing the essential 3D scene information like geometry models or depth information. The chapter finalizes with stating the research questions and research contributions, and ends with the thesis outline and publication history.*

## 1.1 Acquiring 3D scene information from 2D images

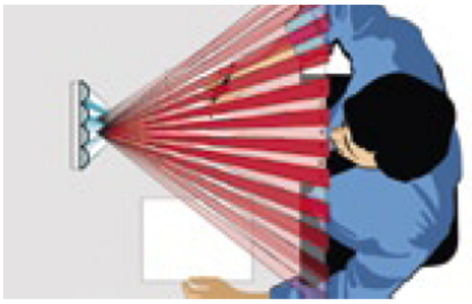
High-quality 3D video is regarded by experts and general public as a clearly enhanced viewing experience, provided that the quality is high enough to avoid viewing fatigue and depth appears in a natural way. Though the principle of 3D viewing has been well understood and the first inception of a 3D viewing device has appeared as early as 1838, wide commercialization of 3D video technologies was simply not possible. It is only recently that commercial 3D films and 3DTV are available and gradually accepted by consumers. However, today's 3D technology is still in its early stage. Similar to the development of color television, many technological and computational problems in scene acquisition, scene reconstruction, and scene displaying need to be solved. Especially, it remains a challenging problem to create 3D contents of natural environments in an efficient and cost-effective way, particularly to create 3D contents from existing or new 2D image data.

3D scene information can be obtained using various methods. For example, by using the time-of-the-light approach, the distance between scene objects and the camera can be measured, based on the traveling time of the light between objects and the camera. By using multiple-view images, 3D geometry models can be reconstructed using triangulation techniques. Acquiring 3D scene information from multiple-view 2D images is attractive due to its flexibility and potential low cost, which has been extensively studied in the area of computer vision in the past decades. However, reconstruction of high-quality 3D information of natural scenes from 2D images is inherently an ill-posed problem. Many technological challenges need to be addressed. This thesis attempts to improve the automation of the 3D reconstruction process from multiple-view images. The two main applications of the explored reconstruction technologies are 3D content creation for free-viewpoint 3DTV and virtual visualization of our living environment.

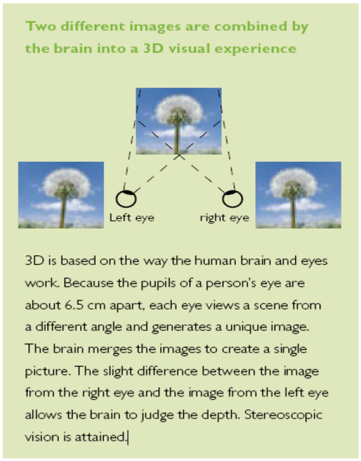
## 1.2 3D applications

### 1.2.1 Free-viewpoint 3DTV

3DTV is regarded as the next revolution in the television history. It will not only fundamentally change the way we watch the video, but also will have a deep impact on our daily life. The principle of 3DTV is well understood. In contrast with the conventional TV where a same view of a 3D scene is perceived by both of our two eyes, 3DTV is able to provide two slightly-displaced views for our left and right eyes. Similar to a real-life situation where our two eyes always receive two slightly-displaced views of a 3D scene, 3DTV can thus provide a more vivid 3D perception than the conventional 2DTV.



(a) human eyes always receive two slightly displaced views from 3D monitor

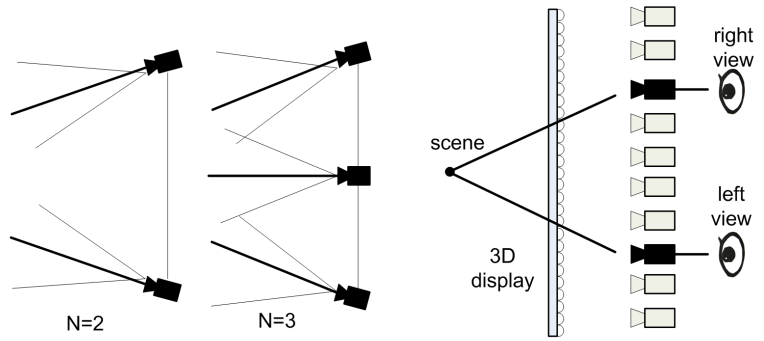



(b) the left and right views integrated by human brain to generate the 3D effect

**Figure 1.1:** *Principle of an auto-stereoscopic 3D monitor.*

The principle of an auto-stereoscopic 3D monitor is illustrated in Fig. 1.1(a). As seen from the figure, the auto-stereoscopic monitor directs the lights from the display to different

angles in 3D space such that our two eyes always perceive two different views. The perceived left and right views are thereafter integrated by the human visual system for a vivid 3D perception, as illustrated in Fig. 1.1(b).




**(a)** multiple-camera scene capture: N is the number of cameras.

**(b)** two virtual views perceived by left and right eyes.

**Figure 1.2:** *Concept of multi-view scene capture and free-viewpoint 3DTV.*

In order to render the left and right views, a 3D scene has to be captured and represented in a suitable data format. For this, there are three main data representation formats: (1) representation using classical 2D images (image-based approach), (2) representation using texture plus depth map that is represented by per-pixel values describing the distances between the object and the camera (depth-based image rendering approach), and (3) representation using a 3D geometry model (model-based approach). In the image-based approach, multiple video streams from different viewpoints are directly coded and transmitted to the receiver, where two appropriate streams are decoded and displayed. To enable a wide-viewpoint viewing, a huge amount of video data will need to be encoded and transmitted. In the model-based approach, a 3D geometry model is first computed at the acquisition side. At the receiver side, the left and right views are rendered by projecting the 3D model onto two virtual cameras. The benefit of the model-based approach is its flexibility. One geometry model plus one texture stream and possibly added occlusion information will be able to render a wide range of viewpoints. The disadvantage is the difficulty in reconstructing the geometry model, because obtaining an accurate 3D scene model from multiple-view images is still a challenging problem. The Depth-Based Image Rendering (DBIR) approach achieves a tradeoff between the model-based and the image-based approaches, and is used in this thesis[1].

Fig. 1.3 depicts a DBIR-based 3DTV system that comprises of four main parts, i.e., 3D acquisition (including 3D content production), transmission, rendering and displaying. Generally speaking, the technologies for transmission, rendering and displaying are relatively mature compared with 3D acquisition. For example, commercial stereoscopic 3D monitors are readily available in the consumer market. The technological challenge of realizing such a 3DTV system lies mainly in 3D content production, in this case, the depth maps. This thesis

[1]Currently, the image-based approach is used by most 3D films, where two video streams from two fixed viewpoints are directly decoded and displayed. The reason is its simplicity in acquiring and representing only two video streams from two fixed viewpoints.

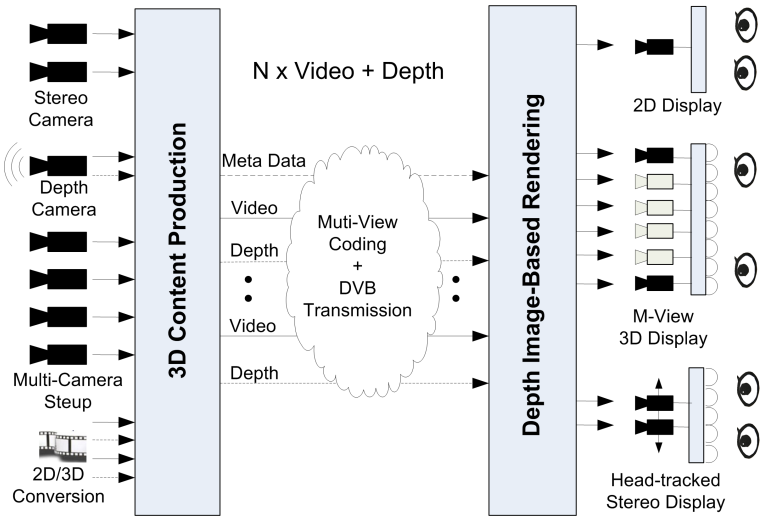



**Figure 1.3:** *A 3DTV system using the texture-plus-depth data format. Depth information is created from stereo, multiple-view or 2D videos at the content-production side and then transmitted to the receiver side, where the selected left and right views are generated for 3D perception on different displays according to display configuration. The rightmost symbols represent human viewpoints as in the previous figure.*

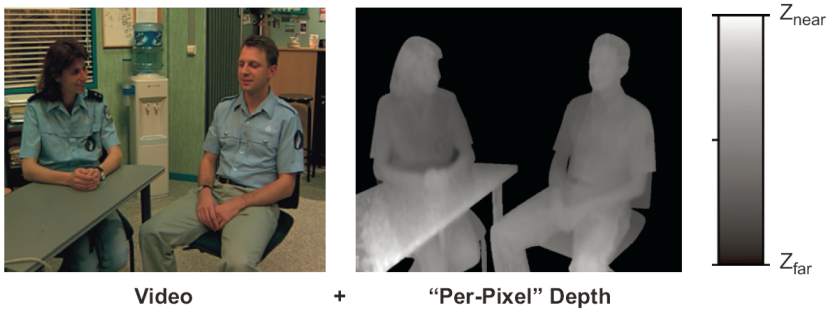


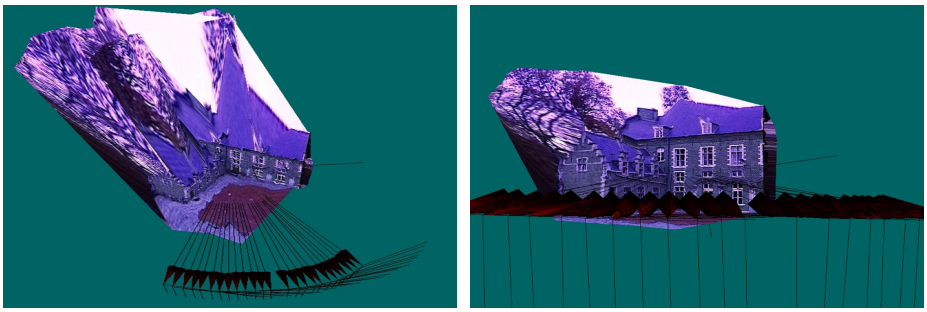

**Figure 1.4:** *Video-plus-depth data representation. Figure extracted from [36].*

makes some attempts to create the high-quality depth maps from multiple-view video.

The video-plus-depth data representation shown in Fig. 1.4 offers good backward compatibility with today's 2D television. Besides, it also has the advantages of good scalability with respect to different viewing conditions and receiver complexity, since varieties of left and right views at different viewpoints can be rendered using the same depth information [13], as illustrated by Fig. 1.3.

### 1.2.2 Visualization of living environment

Depth estimation from MVV is closely related with the problem of 3D modeling from multiple-view images. For example, if the 3D geometry model of a scene is available, then the left and right views can be rendered easily by projecting the 3D model onto the two virtual left and right cameras. In this aspect, we can say that 3D modeling is a 'super-problem' of depth estimation. In this thesis, besides presenting a system for creating depth map from MVVs, we also present a 3D modeling system that automatically recovers the sparse 3D shape of a scene from a long video taken by a moving hand-held consumer camcorder.



(a) top-front view of a house. (b) front view of a house.

**Figure 1.5:** *Visualization of a 3D house reconstructed from the* castle *sequence [3].*

An automated reconstruction of highly detailed 3D models of large-scale outdoor scenes has important applications for scene visualization and analysis. As can be observed from the existing web-based earth representation applications (e.g. Google Earth and Microsoft Virtual Earth) in delivering an effective visualization of large-scale scenes based on aerial and satellite images, it is expected that a realistic visualization of our living environment from ground-based imagery will become a reality in the near future. Due to the rapid development of computer vision and computer graphics technologies, the fast penetration of broadband internet, and the popularity of high-resolution consumer cameras, a realistic visualization of our living environment from ground imagery captured using a hand-held consumer camera is highly desired. For example, a potential house buyer will be able to freely navigate and choose his viewpoint around a virtual house if an accurate 3D representation is available, as illustrated in Fig. 1.5.

Despite the recent accomplishments in both the understanding of the problem and the toolboxes available for the researchers, an automatic scene reconstruction from long image sequences remains as a challenging problem. Fig. 1.6 depicts an envisioned 3D modeling system, where 3D scene information is reconstructed from a video captured using a hand-held consumer camcorder. For such a system to work, many technical challenges need to be solved. For example, how to handle the massive amount of video data? How to accurately estimate the positions and orientations of the large number of cameras? How to handle the varying content of the scene? How to handle the degeneracy of the scene or camera configuration? This thesis addresses some of the above questions.

## 1.3 Research objectives

This thesis proposes algorithms for acquiring 3D scene information from multiple-view images and describes two 3D reconstruction systems to acquire the 3D scene information.

### A. 3D modeling from long video sequences

The 3D modeling system aims at reconstructing the sparse 3D shape of a large-scale static natural scene. The video is captured using a hand-held consumer camcorder. The obtained 3D scene model can be used for visualization of the natural scene environment. As illustrated in Fig. 2.4, a static scene is captured using a hand-held camcorder, which provides the input for the reconstruction system. As a result, the sparse 3D geometry model of the scene is recovered, which can be used for both visualization and video content analysis.

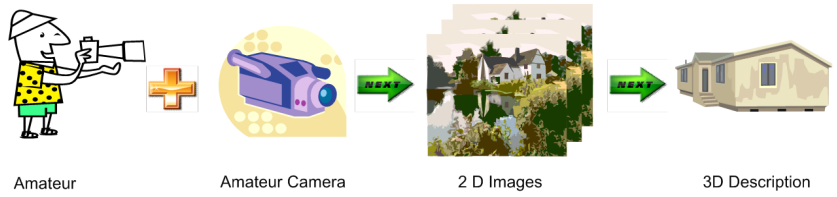



**Figure 1.6:** *3D modeling from video: a video of a large-scale scene is taken by a hand-held camcorder. The video is input to the scene reconstruction system and the 3D shape of the scene is reconstructed.*

With the proposed 3D modeling system, we pursue that a non-professional user can reconstruct a large-scale outdoor scene using a consumer camera.

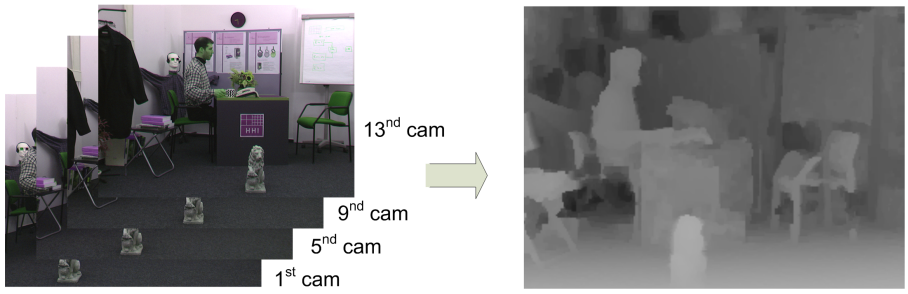
### B. Depth estimation from Multiple-View Video (MVV)

The system aims at creating depth maps from MVV for free-viewpoint 3DTV. As depicted in Fig. 1.6, multiple video streams taken by multiple-synchronized cameras at different viewpoints, form an input to the depth-estimation system. As an output, depth maps for the selected viewpoints are created, which can be used to render the left and right views of the scene for a 3DTV system.

With the proposed depth estimation system, we expect that high-quality depth maps can be automatically created from MVV such that the cost of 3D content production can be significantly reduced.

## 1.4 Research methods

For both systems described in the above section, substantial research has been reported in literature. This section introduces existing research methods that are widely used.




(a) Input: multiple video streams taken by synchronized cameras

(b) Output: depth maps for the reference camera.

**Figure 1.7:** *Depth estimation from multiple-view videos: thirteen video streams taken by thirteen synchronized cameras at different viewpoints are input to the depth reconstruction system. As the output, the depth map for each frame of the reference camera is created.*

### 1.4.1 Methods for 3D modeling from long video

Acquiring 3D scene information has been an active research topic in computer vision for many years. As will be introduced in Chapter 2, various approaches can be used for recovering the 3D scene information. Among those, acquiring 3D information from 2D images is becoming increasingly attractive because of the increasing computational power of personal computing devices and the popularity of high-resolution consumer cameras and broadband networks. Let us first introduce two state-of-the-art 3D-from-2D-image approaches.

#### A. Merging method

The merging method is well-known and widely used for 3D scene reconstruction [65]. In this method, key frames are first selected based on the measurement of camera disparity between two frames from different views. With the selected key frames, the fundamental matrix between the first two key frames is computed and an initial projective shape of the scene is recovered. The projective motion and shape of every subsequent key frame are then computed based on the 3D-2D correspondences between the reconstructed 3D points and the 2D feature points in the new key frame, as illustrated by Fig. 1.8. After all key frames are merged, the Euclidean scene shape and motion are recovered by applying metric constraints to the internal camera parameters.

Generally, the merging method is susceptible to the drift of feature points over long image sequences. Bundle adjustment is often used to refine the reconstruction results, in order to obtain a maximum likelihood estimation of the structure.

#### B. Factorization method

The factorization method was first introduced by Tomasi and Kanade [79] for orthographic views, and extended by Poelman and Kanade [62] for para-perspective views. Then it was

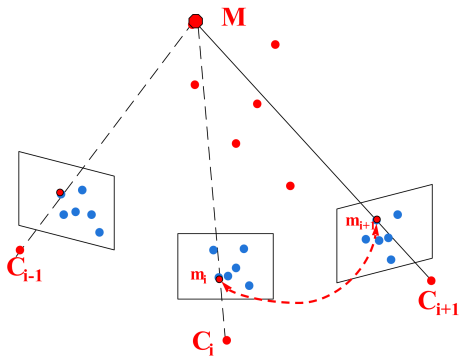



**Figure 1.8:** *Adding a new key frame to the reconstructed structure in the merging method. The 3D point* $\mathbf{M}$ *is reconstructed from images* $i-1$ *and* $i$*. The 3D point* $\mathbf{M}$ *is projected into* $\mathbf{m}_i$ *in image* $i$*. The point correspondence* $<\mathbf{m}_i,\mathbf{m}_{i+1}>$ *provides a 3D-2D correspondence* $<\mathbf{M},\mathbf{m}_{i+1}>$*. The projection matrix of camera* $i+1$ *can be computed given a sufficient number of such 3D-2D correspondences.*

further extended by Han and Kanade [20] for perspective views. The method begins by identifying salient feature points and tracking them from each image to the next. The positions of those points in each image are then collected into a large measurement matrix, which is factorized into projective shape and motion, using Singular Value Decomposition (SVD). The projective shape and motion are thereafter upgraded to Euclidean shape and motion by enforcing metric constraints on camera parameters.

Compared to the merging method, factorization achieves its robustness and accuracy by applying a well-conditioned numerical computation to highly redundant data. The information from a large number of images and feature points is uniformly exploited and thus the influence of the errors in individual images and feature points is considerably reduced, which improves the robustness. Furthermore, factorization is also simpler for implementation, since it does not need key frame selection. Besides, it computes the parameters of all cameras (not only for the selected key frames such as in the merging method), which will be useful for 3D applications such as image-based rendering, where images between two close viewpoints may be required. The major drawback of the factorization method is that it is difficult to be applied to long sequences, where insufficient feature points can be tracked along the whole sequence. Additionally, it also fails when sequences contain so-called critical motions and critical surfaces.

### 1.4.2 Methods for depth estimation from multiple-view images

The point cloud reconstructed by the proposed 3D modeling system comprises of hundreds or thousands of points. Such a sparse 3D model is not sufficient for applications such as 3DTV, where per-pixel depth is required. Consequently, the sparse density of the point cloud has to be filled in order to obtain a per-pixel depth map. In the field of computer graphics, the process of reconstructing the 3D surface of a scene from a point cloud is called *surface*

*reconstruction*. It has been actively studied and many algorithms have been proposed. Unfortunately, due to the sparse and uneven distribution of the reconstructed 3D points obtained during scene reconstruction, the surface reconstruction algorithms from computer graphics area cannot be applied directly to the reconstructed point cloud [89]. Surface reconstruction from point clouds has to be solved by multi-view reconstruction methods.

Multiple-view depth estimation can be formulated as image-based depth labeling, where each pixel of an image is assigned a discrete depth value. If we consider each pixel to be a node in a Markov Random Field (MRF) and define an appropriate neighborhood system, image-based depth labeling can be solved via energy minimization over that MRF.

One major advantage of the energy-minimization approach is that it provides a straightforward and computationally-tractable formulation, where various constraints and prior information about the scene can be used to determine the optimal labeling. The global Maximum A Posteriori (MAP) solution can be found by minimizing the local energies using MRF-optimization algorithms, such as graph cut [67, 8] and belief propagation [75]. More discussion on the motivation of the MAP-MRF framework and its optimization can be found in Chapter 6.

## 1.5 Research challenges

The previous section introduces the existing methods for acquiring 3D scene information from multiple-view images. This section points out a few challenges to realize the two proposed 3D reconstruction systems.

### 1.5.1 3D modeling from long video

Due to its simplicity and robustness, the factorization method introduced in Section 1.4.1 is used in this thesis to reconstruct the 3D scene model from long video sequences captured by a hand-held camcorder. A video sequence can easily contain thousands of frames. To handle such a large amount of data, automated processing is highly desired to reduce the production cost of 3D content. A number of challenges need to be addressed to realize such a 3D reconstruction system.

- *Matching and tracking a large number feature points along a long sequence of images*. The factorization method requires that all feature points must occur in all frames. Tracking a large number of feature points along a long sequence of frames is critical for an automatic reconstruction of the scene shape and the camera motion, using the factorization method. For a video footage of a natural scene captured using a hand-held consumer camcorder, the content, contrast and motion between consecutive images may vary significantly. It is important to design a feature-point matching algorithm that can robustly match feature points between two images for tracking a large number of feature points along a long sequence of images.

- *Handling critical configurations where 3D reconstruction degenerates*. In a long video taken by a non-static positioned camcorder, scene contents vary over time. In certain situations, the configuration of a scene or the multiple capturing positions of the camera

may lead to a failure of the 3D reconstruction process. For an automated 3D modeling from a long video sequence, such critical configurations need to be appropriately detected and handled. A long sequence has to be carefully split into multiple sequences for individual reconstruction. In conclusion, we need an algorithm to detect critical configurations and based on that algorithm, to split a long sequence in such way that partial reconstructions on individual subsequences are possible.

### 1.5.2 Depth estimation from multiple-view video

The energy-minimization approach introduced in Section 1.4.2 is used in this thesis to recover the depth of a scene from multiple-view images. To use the energy-minimization approach, we need to (1) construct an appropriate representation of a Markov Random Field (MRF), which represents a graph, and (2) define appropriate local energies. Despite the wide use of the energy-minimization approach for depth labeling in literature, many challenges remain to be solved. Our research concentrates on the following two aspects.

- *Preserving depth discontinuities between objects.* An image may contain multiple objects. For rendering the left and right views of a scene, the object boundaries have to be accurately preserved. To achieve this, multiple issues have to be addressed. (1) Various depth cues and prior knowledge can be used for depth labeling. We need to find a way to convert them into quantitative data and smooth costs to define the local penalty and interaction energies. (2) The contents of an image can vary significantly in the spatial dimensions. For example, an image may contain object boundaries, smooth areas and high-contrast areas at the same time. The local energies need to be adapted to the local image content in order to precisely describe the nature of the image content. (3) We need to construct an appropriate graph (sites and cliques) for MRF optimization, which is able to facilitate the localization of the penalty and interaction energies. The commonly-used regular lattice does not work well. Especially when the density of the nodes is low, the regular lattice does not align well with the object boundaries, which degrades the resulting depth map. In conclusion, we aim at an algorithm to construct a graph, facilitating a precise definition of the local energies using various depth cues.

- *Efficiency and robustness of energy-minimization process.* Energy minimization is solved using a graph cut algorithm in this thesis, which is computationally expensive. Besides, the optimization result can converge to local minima, especially when the number of graph vertices and the number of depth planes are large. In that case, depth labeling using graph cut will become slow and unstable. We need to find an appropriate optimization approach to improve both the speed and robustness of the energy-minimization process.

## 1.6 Thesis contributions

### 1.6.1 Contributions to 3D modeling from long video

Corresponding to the research challenges as pointed out in Section 1.5.1, the following major contributions are proposed to realize the 3D modeling system that recovers the 3D scene

shape from a long sequence captured with a hand-held camcorder.

- *Matching and tracking a large number of feature points along a long sequence of images*

  To track a large number of feature points along a long sequence of images, a novel texture-independent feature-point matching algorithm is designed. The proposed algorithm uses only a smoothness constraint, which states that neighboring feature points in images tend to move with similar directions and magnitudes. The employed smoothness assumption is not only valid but also robust for most images with limited image motion, regardless of the camera motion and scene structure. Because of this, the algorithm obtains two major advantages. First, the algorithm is robust to illumination changes, as the employed smoothness constraint does not rely on any texture information. Second, the algorithm has a good capability to handle the drift of the feature points over time, as the drift can hardly lead to a violation of the smoothness constraint. This leads to the large number of feature points matched and tracked by the proposed algorithm, which significantly helps the subsequent 3D modeling process.

- *Splitting a long sequence into multiple subsequences and handling critical motions and surfaces*

  In order to split a long image sequence, we have designed an algorithm to detect critical configurations where the factorization method degenerates. The following four critical configurations are detected: (1) coplanar 3D scene points, (2) pure camera rotation, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. The configurations in cases (1), (2) and (4) will affect the rank of the Scaled Measurement Matrix (SMM). The number of camera centers in case (3) will affect the number of independent rows of the SMM. By examining the rank and the row space of the SMM, the above-mentioned critical configurations are detected. Based on the analysis of the singular values and linear dependency of the row space of the SMM, the proposed algorithm provides a simple but effective criterion to detect the above four critical configurations. Based on the critical configuration-detection algorithm, a sequence-dividing algorithm is designed to split a long sequence into subsequences such that a successful 3D reconstruction can be performed on each subsequence with a high confidence.

### 1.6.2 Contributions to depth estimation from MVV

Corresponding to the research problems pointed out in Section 1.5.2, we have achieved the following contributions to realize the depth-estimation system that computes the depth of a scene from multiple-view images.

- *Improved accuracy of the depth maps*.

  Section 1.5.2 indicates the necessity for an algorithm that constructs a graph facilitating a precise definition of the local energies, using various depth cues. In this thesis, we

have designed a segmentation-driven graph-generation algorithm that constructs 2D triangular meshes on over-segmented image maps. With this algorithm, the edges of the resulting triangular meshes align well with the object boundaries, which improves the depth accuracy. Besides this aspect, the segmentation-based process enables a convenient use of various depth cues and prior knowledge for defining the local energies and adaptation of them to the local image content. This also improves the depth accuracy.

- *Increased efficiency and robustness of the energy-minimization process.*

  As stated in Section 1.5.2, energy minimization becomes inefficient and unstable when the number of vertices of a graph and the number of the depth labels are large. To address this issue, a coarse-to-fine optimization scheme has been designed to improve the efficiency and robustness of the energy-minimization process. In this scheme, both the density of the vertices and the interval between two depth planes are gradually refined in multiple optimization passes. The labeling results obtained in the previous optimization pass are used as an initial input for the current pass. Because of this, the number of vertices and the number of depth labels for every subsequent optimization pass are significantly reduced, which increases both the efficiency and the robustness of the energy-minimization process.

## 1.7 Thesis outline and publication history

This thesis presents the two 3D reconstruction systems to obtain 3D scene information from multiple-view 2D images, with the aim to automate the 3D reconstruction process based on the framework of Structure and Motion (SaM). The first reconstruction system is based on SaM, and aims at automating reconstruction of a 3D scene using long video sequences. The system is discussed in detail in Chapter 3 and two related major contributions on feature point matching and dividing long video sequences are presented in Chapters 4 and 5, respectively. The second reconstruction system attempts to automatically reconstruct depth maps from multiple-view videos taken by multiple synchronized cameras. Chapter 6 presents this system and novel algorithms to improve the quality and robustness of the depth reconstruction system. This section outlines the contents of the remaining chapters of this thesis.

**Chapter 2: System Overview and Related Work**
This chapter presents the overview of the two 3D reconstruction systems studied in this thesis, i.e., the system for reconstructing sparse 3D geometry of a large-scale scene from a long video sequence, and the system for creating depth maps from multiple-view videos. A survey of existing 3D acquisition methods is presented, where special attention is paid to the "3D-from-image" approach used in this thesis, where 3D scene information is computed from normal 2D imagery. Subsequently, the design objectives and requirements of the two systems are described. Afterwards, the principal modules of the two systems are presented and common processing modules of the two systems are identified. Prior work related to the two main processing functions of both systems is presented.

**Chapter 3: Factorization-based Scene Reconstruction from Long Sequences**

The first reconstruction system is based on SaM, and aims at automating reconstruction of a 3D scene using long video sequences. This chapter gives a detailed presentation of the 3D modeling system, starting with an overview of the mathematical background of SaM and then presenting multiple minor contributions to the system. We commence with the mathematical formulation of multiple-view scene reconstruction. After that, we add a number of improvements to this framework. First, *blur-and-abrupt-frame removal* removes blur frames and frames with abrupt image motion in order to track more feature points. Second, a *Harris corner detector with content-adaptive thresholds* makes the detected feature points more evenly distributed over the frames, which improves the robustness of the reconstruction process. Third, a *hierarchical triangulation scheme* maximizes the number of 3D points while minimizing the redundant triangulations to enhance the quality of the reconstructed point cloud. Fourth, a *scheme merges partial reconstructions* from individual subsequences in order to obtain a 3D model of a complete scene from a long video sequence. Finally, experimental results using two video sequences are presented to demonstrate the effectiveness of the proposed 3D modeling system for an automatic scene reconstruction from long sequences.

**Chapter 4: Texture-Independent Feature Point Matching**
As pointed out in Chapter 3, tracking a large number of feature points along a long sequence of frames is critical for an automatic SaM on a long sequence using the factorization method. For 3D reconstruction on a long video sequence captured with a hand-held camcorder, some special factors need to be considered when matching and tracking the feature points. For example, the motion between two frames of a video sequence is generally small. Therefore, this chapter introduces a feature point matching algorithm which is specifically designed for matching and tracking a large number of feature points over successive frames where the image motion is limited. The algorithm is based only on a smoothness constraint and does not use any image texture for matching, which leads to an improved robustness against illumination changes and a large number of tracked feature points. In the algorithm, the correspondences of feature points in a neighborhood are collectively determined in a way such that the smoothness of the local motion field is maximized. Experimental results show that the proposed method outperforms existing methods for feature-point tracking in image sequences. The algorithm forms one of the major contributions of this thesis. The initial result of this work have been published in the proceeding of 2007 IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP) [42]. The paper has been selected out of around 2000 papers as the finalist for the best paper award. The version with extended experimental results and algorithm validation has been published in the 8th Asian Conference on Computer Vision (ACCV) [43].

**Chapter 5: Dividing long sequences for factorization-based structure and motion**
Chapter 2 points out that a long sequence has to be divided into subsequences such that sufficient feature points can be tracked for the factorization-based SaM on individual subsequences. An automatic division of a long sequence into subsequences is essential for the proposed SaM system. This chapter proposes algorithms for dividing long sequences with the consideration of so-called critical configurations, where the factorization method fails. First, we introduce the projective reconstruction and camera calibration algorithms that are used in this thesis, and are needed for presenting the proposed dividing algorithm. Second,

we present algorithms to detect the following critical configurations where the factorization method is not possible: (1) coplanar 3D points, (2) pure rotation of the camera, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. We have observed that the configurations in cases of (1), (2) and (4) will affect the rank of the scaled measurement matrix (SMM). We have also observed that the number of camera centers in case of (3) will affect the number of independent rows of the SMM. Therefore, we propose to examine the rank and the row space of the SMM, in order to detect the above-mentioned critical configurations. The third part in the chapter proposes a sequence-dividing algorithm to automatically divide a long sequence into subsequences such that a successful SaM can be obtained on individual subsequences with a high confidence. Finally, experimental results for both synthetic and real sequences are presented, to demonstrate the effectiveness of the proposed algorithm for dividing a long sequence and creating an automatic 3D reconstruction. This work has been published in the proceedings of the 9th Asian Conference on Computer Vision (ACCV) [46].

**Chapter 6: Estimating depth map from multiple-view video**
This chapter deals with the second reconstruction system for creating depth maps from multiple-view videos. This system involves a sparse reconstruction of the 3D scene using SaM, and subsequently upgrading of the sparse reconstruction to obtain the per-pixel depth maps. Chapter 3 presents the algorithm for scene reconstruction from long video sequences, where a sparse set of 3D points (point cloud) can be reconstructed. For many applications such as 3DTV, the density of the obtained point cloud is not sufficient for rendering high-quality left and right views, and the point cloud needs to be converted to per-pixel depth maps. The density of the reconstructed points shows holes and this insufficiency has to be filled. This chapter presents the system for creating depth maps from multiple-view videos (MVV) taken with multiple synchronized cameras, which is typically used for the production of 3D video material. The proposed system is presented in two parts: (1) *sparse reconstruction* to calibrate the cameras and to reconstruct the point cloud, and (2) *depth reconstruction* to upgrade the point cloud to a 3D surface such that the per-pixel depth map can be created. The initial result of this work has been published in the proceedings of the 29th Int. Symp. Information Theory in the Benelux [44]. More elaborated results and algorithm descriptions have been published in the 2008 conference on Advanced Concepts for Intelligent Vision Systems (ACIVS) [45].

**Chapter 7: Conclusion**
This chapter evaluates the values of the two reconstruction systems presented in this thesis. The SaM framework is a complex system involving many processing modules. This thesis designed and implemented two complete systems for 3D reconstruction with a high degree of automation, which clearly helps in the production of 3D video content. Besides the system construction, a number of novel algorithms such as texture-independent feature point matching, dividing long video sequences, and coarse-to-fine depth labeling using an energy-minimization framework, have been proposed to enhance the system performance.

CHAPTER 2

# System overview and related work

*This chapter presents the overview of the two 3D reconstruction systems studied in this thesis, i.e., the system for reconstructing sparse 3D geometry of a large-scale scene from a long image sequence, and the system for creating depth maps from multiple-view videos. This chapter starts with a survey of the existing 3D acquisition methods, where a special discussion is addressed to the 3D-from-image approach that is used in this thesis. Subsequently, the design objectives and requirements of the two systems are described. Afterwards, the major modules of the two systems are presented and common processing modules of the two systems are identified. Prior work related to the two main concepts is discussed. Finally, this chapter ends with a discussion and conclusion.*

## 2.1 Taxonomy of 3D acquisition methods

Obtaining 3D scene information has been an active research topic in computer vision for a long time for which many techniques have been proposed. As shown in Fig. 2.1, 3D acquisition methods can be classified into multiple categories. In this section, we briefly introduce each category of the 3D acquisition methods as well as their advantages and drawbacks.

### 2.1.1 Active and passive acquisition

As illustrated in Fig. 2.1, 3D acquisition techniques can be broadly classified into two categories: *active* and *passive* techniques. The active techniques usually rely on controlled light sources for acquiring the 3D information. Examples of this category include the structured light approach and the time-of-flight approach. In the *structured light approach*, a controlled light source projects a special pattern on the scene, which is captured by cameras and used for computing the 3D scene geometry. The special pattern provides extra information that emphasizes the borders of scene objects and their geometry. In the *time-of-light approach*,

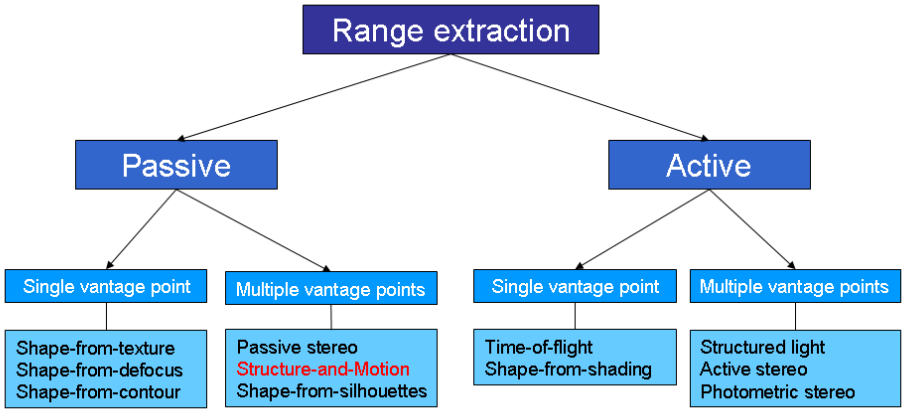



**Figure 2.1:** *Taxonomy of 3D acquisition methods (figure extracted from [54]).*

the traveling time of the controlled light between scene objects and cameras is measured in order to calculate the depth of the scene. The advantage of the active techniques is the robustness and efficiency, because the special illumination significantly simplifies many challenging tasks, such as feature point matching and camera calibration. The limitation is that these techniques are typically applicable to indoor environments only. Furthermore, the use of the controllable light source also increases the cost of the acquisition system, which indicates that they are mostly suitable for studio production.

The passive techniques rely on 2D images for recovering the 3D scene information. This category has three main benefits: (1) it has a low requirement on the acquisition equipment, (2) it allows a flexible scene size and is applicable to both indoor and outdoor scenes, and (3) it is generally easy to operate during acquisition. For example, a camera can be mounted on top of a car or even held by a hand. Due to the increasing computational power of personal devices and the popularity of digital cameras, this approach becomes increasingly attractive because of the inherent low cost and high flexibility. The major drawback is that the 3D-from-image process is generally an ill-posed problem. For example, it fails for certain scene and camera configurations such as a rotation-only camera, coplanar feature points and texture-less scenes. Many technological challenges need to be handled to achieve a robust 3D-from-image system.

### 2.1.2 Multiple-view and single-view acquisition

The 3D acquisition methods can also be classified according to the number of viewpoints from where the scene is captured, i.e., the single-vantage approach and the multi-vantage approach. For example, for the passive approach, possibilities for the single-vantage methods include shape-from-texture, shape-from-shading, shape-from-gravity, shape-from-focus, etc. The possibilities for the multi-vantage methods include multiple-view 3D modeling, depth

from stereo, etc.

The single-view approach is considered to be part of the *recognition school* in computer vision, where the 3D information is obtained by deriving the high-level semantic description of the image content, based on heuristic depth cues such as shading, texture, blur, gravity, occlusion, etc. This approach has the advantage that it is widely applicable to varieties of scenes, including scenes with moving and deformable objects. The drawback of this approach is the difficult modeling of the high-level heuristic cues from an image. Scene interpretation based on a single 2D image remains as a very challenging problem [41], if not fundamentally impossible.

Compared with the single-view approach, the multiple-view approach is usually classified within the *reconstruction school* in computer vision. In this approach, the physical relation between the image motion, the camera motion and the 3D scene geometry is mathematically described using theories of projective and multiple-view geometry. Assuming certain configurations of the scene and the cameras, the 3D scene information can be computed in a well-formulated way. However, one major drawback is its limited applicability. It cannot handle particular configurations (e.g. rotation-only, coplanar scene and texture-scarce scene), where 3D reconstruction degenerates [61].

### 2.1.3 Approach used in this thesis

This thesis attempts to recover the 3D scene information from multiple 2D images. The use of multiple 2D images for 3D acquisition is motivated by a number of beneficial aspects.

- No special hardware is required. Only consumer cameras and camcorders are used in our experiments.
- Variable scene sizes can be used for reconstruction, which applies to both indoor and outdoor situations.
- The acquisition process is simple, and does not require special training of the operator.
- This approach becomes increasingly attractive over time due to the growing computational power of personal computing devices and the popularity of high-resolution cameras.

As already indicated in Section 2.1.2, the major drawback is the difficulty in handling critical camera and scene configurations where 3D reconstruction degenerates. Compared with the active approach where active sensors such as a laser scanner can be used, this difficulty makes the design of an automated 3D-from-image system challenging. This thesis contributes in increasing the automation of multiple-view 3D reconstruction process by providing algorithms to handle some of the above critical configurations.

## 2.2 Objectives and requirements of two explored systems

As pointed out in Section 1.3 of Chapter 1, our research aims at (1) automatically reconstructing the sparse geometry models of a static 3D scene from video sequences captured with a hand-held camcorder (3D modeling from long video), and (2) creating depth maps

from multiple-view videos (depth estimation from MVV). This section elaborates the design objectives and requirements of the two explored systems.

**Table 2.1:** *Objectives and requirements of 2 explored 3D reconstruction systems*

| | 3D modeling from long video | depth estimation from MVV |
|---|---|---|
| Input | video captured by a hand-held consumer camcorder | multiple-view videos acquired by multiple synchronized cameras |
| Output | sparse 3D geometry of a large-scale scene | depth maps for the selected cameras |
| Scene | static scene | dynamic scene |
| Processing | automated off-line processing | automated off-line processing |

The design objectives and requirements of the two systems are summarized in Table 2.1. With the 3D modeling system, we pursue that a non-professional user will be able to reconstruct the 3D scene geometry of a large-scale outdoor scene using consumer cameras. The reconstructed 3D geometry can be used for visualization, gaming, etc. With the proposed depth estimation system (right side), we strives for a high-quality depth map that is automatically created from MVVs, in order to reduce the cost of 3D-content production such as for 3DTV applications.

## 2.3 Overview of the proposed 3D modeling system

### 2.3.1 System block diagram

Fig. 2.2 depicts the block diagram of the proposed 3D modeling system, where we observe that the system is comprised of five major modules and some of them are further divided into several blocks. For example, the first module *feature point detection and matching* is divided into four blocks, i.e., *Harris-corner detection*, *blur-frame removal*, *abrupt-frame removal* and *feature point matching*. In this thesis, we implement the complete 3D modeling system as shown in this figure, and propose a number of novel improvements to individual processing modules. The functionalities of the five system modules as shown in Fig. 2.2 are briefly introduced as below.

1. *Feature point detection and matching:*

   In this module, feature points from individual frames and the feature point correspondences between every two successive frames are detected. For a long video sequence taken by a hand-held video camcorder, blur images and images with abrupt image motions are detected and rejected for feature point tracking, in order to track more feature points along more frames.

2. *Splitting long video sequence:*

   In this module, a long video sequence is automatically partitioned into short subsequences in order to track sufficient feature points for the factorization-based SaM. To

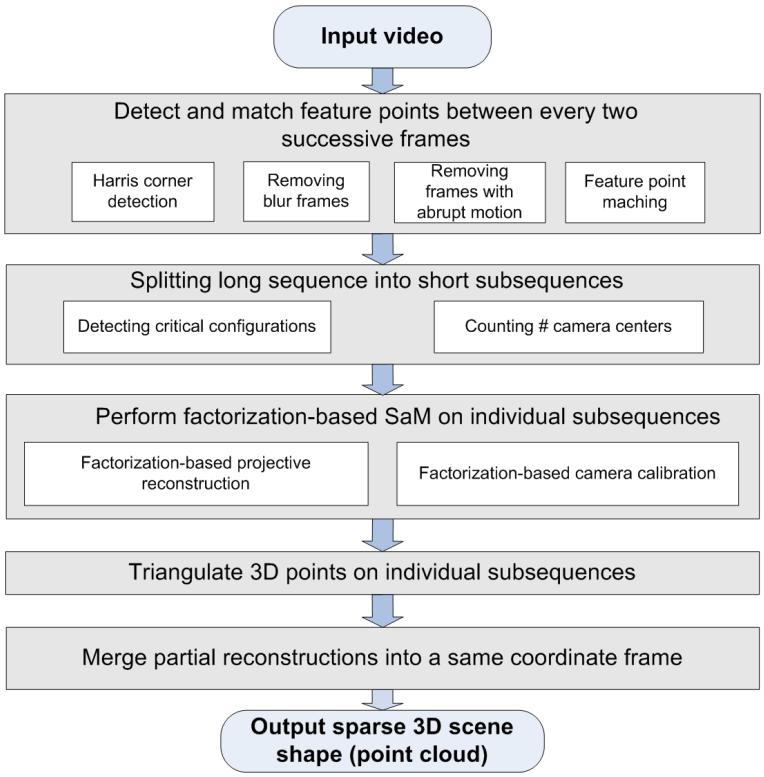



**Figure 2.2:** *Block diagram of the proposed 3D modeling system.*

ensure an accurate SaM on individual subsequences, a number of critical configurations (e.g. coplanar feature points and rotation-only cameras) are detected, where the factorization-based SaM degenerates.

3. *Factorization-based SaM:*

   In this module, the factorization method is applied to individual subsequences to reconstruct the 3D scene shapes and camera positions and orientations for individual subsequences.

4. *Triangulation:*

   In this module, 3D points are triangulated from the available feature point tracks. Redundant triangulations are minimized by avoiding projection of multiple 3D points onto the same 2D feature point.

5. *Merging partial reconstructions:*

   The reconstructed point clouds from individual subsequences are located in separate coordinate frames. In this module, partial reconstructions from individual subsequences are merged into the same coordinate frame to obtain the complete scene geometry.

Note that if the input video is short enough such that sufficient feature points can be tracked along the whole sequence, then no splitting and merging of subsequences are required, so that the modules of *splitting a long sequence* and *merging partial reconstructions* can be skipped. The blocks of *removing blur frames* and *removing frames with abrupt motion* are also not required if the sequence is taken by a digital camera, when almost no frame contains blur and only a few frame pairs will have abrupt camera motion.

### 2.3.2 Video capturing for 3D modeling from long sequences

Some of the video material used for experimenting in this thesis is captured by the author and some other material is derived from a standardized set of experimental sequences used for exchanging results between research institutes. The reason for this is as follows. The amount of cameras and capturing devices in consumer equipment is growing steadily. This will lead to a situation where a significant amount of video content is going to be produced by consumers. Simultaneously, 3D content is emerging for various applications. This motivates why we are investigating the use of consumer cameras for 3D reconstruction. In consumer video capturing, the camera is recording extensively so that the use of long video sequences is a common case. For this reason, we have used video content that is either downloaded from the Internet, or taken with hand-held consumer cameras. The content from the Internet has enabled us to identify sequences of multiple-view video of a dynamic scene taken by multiple synchronized cameras, which will be used for our depth estimation system.

Below, we describe the important aspects concerning image sequence capturing for 3D reconstruction based on the presented framework for 3D modeling.

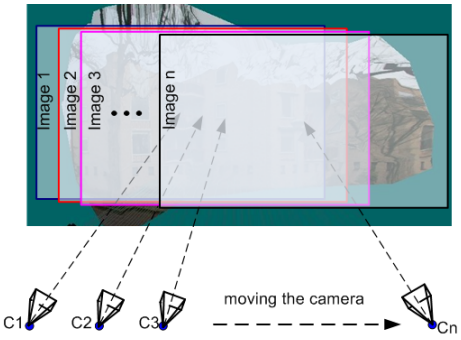



**Figure 2.3:** *3D scene captured by smoothly moving a camera around the scene. Two consecutive images have a significant portion of overlap, so that the recorded image sequence contains a large amount of image redundancy.*

As illustrated in Fig. 2.3, the image sequence or video of a scene is captured by smoothly moving a hand-held camera around the scene, which we refer to as *sliding-window capturing scheme*. Literature [39] states that the use of redundant images in 3D reconstruction provides several clear advantages, which we cite here literally: *"(a) better geometric accuracy; (b) improved automation with fewer catastrophic failures and fewer manual interventions; (c) reduced occlusions of urban surfaces; (d) fewer gross errors in the automated computations."*.

Our sliding-window approach also generates video with large redundancy. Furthermore, this also offers the following benefits.

- *Easy operation*: no special training and equipment is required for scene capturing by users.
- *Large redundancy*: because two consecutive frames significantly overlap each other, the image sequence contains a large amount of image redundancy, which helps to increase the system robustness, as indicated previously. This leads to a decreased cost and an improved quality of the produced 3D contents.
- *Improved feature point tracking*: the small camera motion between two consecutive frames is helpful for matching and tracking feature points between two neighboring frames.

Apart from the above benefits, the large amount of image redundancy is also challenging for the processing. A video of a large-scale scene (e.g. captured in a resolution of $1280 \times 720$ at a frame rate of 30 Hz[1]) contains thousands of frames. To process this large amount of data, automated processing is highly desired.

### 2.3.3 Related work

Substantial effort on modeling of the large-scale 3D scenes has been performed recently. A special review [95] of the recent 3D reconstruction research in the computer vision community discusses the modeling and representations of large-scale 3D scenes. Ref. [66] reports a system for real-time reconstruction of the 3D models of outdoor scenes from multiple videos by car-mounted cameras and GPS/INS data, using the structure-and-motion approach. In [12], the author presents a city modeling framework that integrates both recognition and reconstruction methods for reconstructing the urban city from ground imagery captured by car-mounted cameras. In this work, the detection of the pedestrians and cars and the reconstruction of the 3D environment are tightly integrated such that both detection and reconstruction can benefit from each other's continuous input. Ref. [92] presents a method for estimating the depth information of the straight or mildly-curved urban streets by analyzing the stationary blur in route panoramas. An advantage of this method is that it avoids the feature matching process, which is error-prone in complex street scenes. Concerning the complete design of 3D reconstruction system, closely related work can be found in [33, 55, 64, 19, 40]. Furthermore, recent work on tree modeling [77] and non-rigid object modeling [26] makes a realistic modeling of the city environment possible. These advances in 3D acquisition and 3D modeling technologies have resulted in emerging 3D-visualization applications, such as geo-referenced country and city imaging[2], which enables the detailed viewing of city environments even at home.

Besides 3D modeling from images, many 3D modeling systems use explicit depth sensors. For example, the 'Bayon Digital Archival Project' [1] models the entire Bayon temple using laser scanners. Although initially explored at universities, laser scanning finds its way increasingly in society and is applied by engineering companies for detailed measurements

[1] We describe frame rate by using "frames per second", fps or Hz in this thesis.

[2] Well-known examples of this development are Google Earth, Microsoft Bing, etc.

and modeling of specific areas that are crucial in a country's infrastructure. As a result of this increased use, the cost of laser scanning equipment is dropping considerably, which likely will further boost the use of this technology.

## 2.4 Overview of the proposed depth estimation system

### 2.4.1 System block diagram

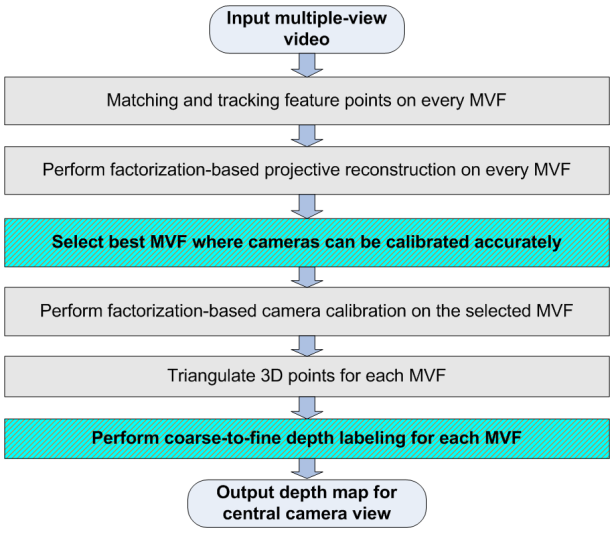
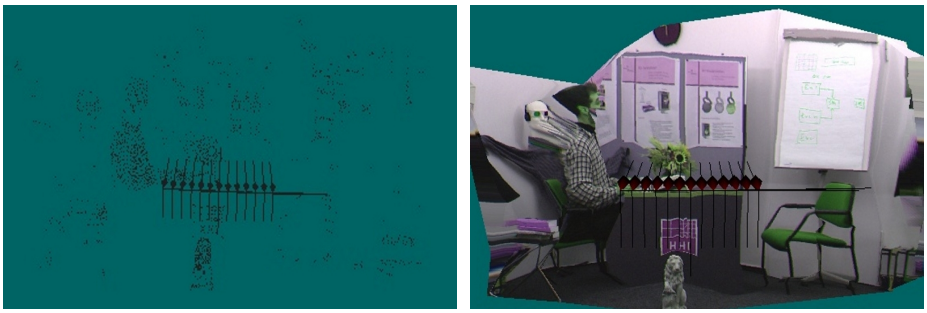



**Figure 2.4:** *Block diagram of the proposed depth estimation system. In the figure, MVF refers to Multiple-View Frame. Only the two shaded modules are new to this system. The other three modules are the same as those in the discussed 3D modeling system depicted in Fig. 2.2.*

Fig. 2.4 depicts the major steps of the studied depth estimation system. In the figure, the MVF refers to *Multiple-View Frame*, which is defined as a set of images that are taken by multiple-synchronized cameras at the same point in time. For example, Fig. 2.5 shows an example of an MVF that comprises 13 images. From the figure, we observe that the system takes the MVV data as the input and creates the depth map for the selected camera view as the output (an example depth map is shown in Fig. 2.7(b)). Going back to Fig. 2.4, we further note that the system comprises of five major modules, where only the two shaded modules, i.e., *camera calibration* and *depth labeling*, are newly introduced (see the figure caption). With the assumption that camera parameters cannot be changed during capturing, the first new module computes both the intrinsic and extrinsic parameters of the multiple-synchronized cameras, as illustrated in Figs. 2.6(a) and 2.6(b). Knowing the camera parameters and reconstructed point cloud, the second new module assigns a discrete depth value to each pixel of the selected image, in order to create the depth map as illustrated in Fig. 2.7(b).

As an example, Fig. 2.5 shows four out of thirteen multiple-view video streams captured


**Figure 2.5:** *Thirteen multiple-view video streams captured by thirteen synchronized cameras as the input to the discussed depth estimation system.*



(a) Point cloud without texture mapping. (b) Point cloud with texture mapping.

**Figure 2.6:** *Reconstructed sparse 3D scene geometry and camera placements prior to depth labeling.*

(a) Selected ($7^{th}$) camera view of a MVF. (b) Depth map for the selected camera view.

**Figure 2.7:** *Selected camera view of an MVF and the created depth map.*

by thirteen synchronized cameras from different viewpoints, which are input to the system. Fig. 2.6 shows the sparse point cloud and the camera placements reconstructed for an MVF, and Fig. 2.7 shows the depth map for the selected camera view.

### 2.4.2 Related work

The explored depth estimation system computes the depth maps from multiple-view image data. Concerning the implementation of a complete Depth-Image-Based-Rendering (DIBR) 3DTV system, the European project ATTEST [13] is one of the earliest projects that investigated the video-plus-depth 3DTV for broadcasting systems. In this project, the entire processing chain of a 3DTV system including 3D acquisition, coding, transmission and rendering is investigated. The project has resulted in the conclusion that the generic video-plus-depth layered coding syntax is flexible, backward-compatible with 2D images and commercially feasible for 3DTV in broadcasting. The work of [36] proposes an advanced approach for 3DTV services. The paper discusses particularly the aspects of inter-operability and multiple-view adaptation in a 3DTV system, where a wide range of production formats and presentation techniques are available. The author indicates that a single video-plus-depth stream is not sufficient to meet the wide range of formats and displaying approaches, and therefore proposes to use multiple video-plus-depth streams to represent the data, which can be encoded efficiently using the multiple-view video coding technique currently under standardization by MPEG [57]. The number of input streams can be converted to a varying number of presentation views at the decoder side, using the DIBR technology. An algorithm for depth map estimation for arbitrary multi-baseline camera system is described.

Multiple-view depth estimation is often formulated as Image-Based Depth-Labeling (IBDL), where each pixel of an image is assigned a discrete depth value. The research in this thesis is based on the IBDL approach. The problem of IBDL can be elegantly expressed using the Markov Random Field (MRF) framework, and the optimal solution can be found by energy minimization using approximation techniques, such as graph cuts [8] and belief propagation [75]. At this point, we summarize some related work concerning energy minimization. In [7], the theoretical background of the energy-minimization framework and its application to vision problems are presented. In [67], an algorithm based on energy minimization is proposed to solve the multiple-view correspondence problem. In [76], a comparative study of energy minimization techniques for MRF is presented, where the solution quality and execution time of a few commonly-used energy minimization algorithms are compared.

## 2.5 Common processing of two systems

As shown in Fig. 2.2 and Fig. 2.4, the two discussed systems for 3D modeling and depth estimation share a large part of common processing. This section indicates the commonalities between the two systems, and motivates the background for those commonalities.

As discussed above, the depth estimation is performed on MVFs that contain multiple images taken at the same time. From the information theory point of view, there is no essential difference between MVFs by multiple-synchronized cameras and image sequences of a static scene taken from different viewpoints with a camcorder. This leads to the following four common processing blocks.

- *Feature point matching:* this is essential for multiple-view 3D reconstruction, because it finds the corresponding points between consecutive or neighboring images of the same scene. This correspondence information is required by both systems to perform the subsequent factorization-based 3D reconstruction.

- *Factorization-based projective reconstruction:* this is required by both systems to recover the projective scene geometry. The images are basically projections of 3D scenes so that projective geometry is the starting point for recovering the Euclidean scene geometry, given the limited knowledge about the camera parameters.
- *Factorization-based camera calibration:* The knowledge of the camera parameters enables us to rectify the projective scene dimensions to Euclidean distances. This is required by both systems to recover the Euclidean scene geometry. The motivation to use the factorization method has been discussed in Chapter 1.
- *Triangulation:* this is required by both systems to increase the density of the reconstructed 3D points.

## 2.6 Conclusion

In this chapter, we have briefly presented the overview of the two explored systems. First, we have given an introduction to the existing 3D-acquisition methods for acquiring 3D scene information. The benefits and drawbacks of the recognition and reconstruction schools in computer vision as well as the individual 3D-acquisition methods from those two categories have been discussed. The multiple-view 3D-from-image approach is adopted in this thesis due to its high flexibility and low cost. This is a passive approach, which brings the following advantages. Variable scene sizes can be used for reconstruction, and the technique is applicable to both indoor and outdoor situations. Moreover, the acquisition process is simple, and can be done with consumer cameras without requiring special training of the operator. We have also indicated that the major drawback of the adopted approach is the difficulty of handling the critical configurations where 3D reconstruction degenerates.

After that, the design targets and requirements of the two proposed systems have been described. The key aspects of both systems are that we need to retrieve some form of 3D geometry information from multiple-view data, and do this in an automated processing way. In the 3D modeling system, we continuously capture the surroundings with a video camera, so that a high amount of samples is available for reconstruction. Alternatively, the depth reconstruction system computes the depth signal from multiple videos captured by multiple-synchronized cameras. The depth signal represents a simplified form of geometry information.

The major modules of each system have been briefly introduced. Our research objectives are mostly satisfied by providing specific contributions to individual processing blocks. In the following chapter, we concentrate on the details of the individual processing modules of the 3D modeling system. To this end, we propose to improve corner detection and triangulation techniques for better 3D modeling from long video sequences. Results on real, natural video sequences will be reported.

CHAPTER 3

# Factorization-based scene reconstruction from long sequences

*An automated reconstruction of highly detailed 3D models of large-scale outdoor scenes has been an active research subject in computer vision, which has diverse applications such as 3D visualization and video content analysis. Despite the recent accomplishments in both the understanding of the problem and the availability of the algorithmic tools, an automatic recovery of the scene structure from long image sequences remains as a challenging problem. This chapter gives a detailed presentation of the 3D modeling system which has been briefly introduced in the previous chapter. First, we introduce the mathematical formulation of the problem of multiple-view scene reconstruction. After that, we present a number of improvements that are made to the individual components of the proposed 3D modeling system. Finally, experimental results using two video sequences are presented to demonstrate the effectiveness of the proposed 3D modeling system for an automatic scene reconstruction from long sequences.*

## 3.1 Introduction

As discussed in Section 1.4.1, the factorization method has the advantages of simplicity and robustness. However, there are a number of important system aspects that need to be addressed while designing the proposed 3D modeling system. The primary aspects are identified below.

- The factorization method requires that all feature points should occur in all images. Thus, matching and tracking a large number of feature points along a long sequence

of images is essential to achieve a robust 3D reconstruction. This implies the use of a sensitive and robust feature point detector to provide a high number of evenly distributed feature points and a feature point matcher that is able to match and track a large number of feature points along a long sequence of images.

- For 3D reconstruction on a long video captured using a hand-held moving camcorder, blurred frames and frames with abrupt image motion need to be removed in order to sustain the feature point tracking over time.
- The factorization method fails on so-called *critical motions* and *critical surfaces* [74, 20]. Hence, such critical configurations need to be detected in the proposed 3D modeling system.
- The number of the tracked feature points decreases with the number of images. For a long sequence comprising hundreds of images, we have to divide the long sequence into subsequences for partial reconstructions. Consequently, we need a partitioning algorithm to divide a long sequence.
- Since each partial reconstruction on subsequences using the factorization method has a separate coordinate frame and different scale, partial reconstructions have to be merged into a single coordinate frame in order to obtain a 3D model of the complete scene.
- After the cameras are calibrated, 3D points can be computed via triangulation. However, noise or outliers within the measurements lead to inaccurate triangulations, which have to be detected and removed. Further, since the same 3D point can be triangulated from different feature point tracks, redundant triangulations have to be minimized in order to improve the reconstruction accuracy.

The above list basically provides a set of requirements that need to be satisfied for obtaining a robust and automated 3D modeling system. Since a number of these requirements refer to particular details of the processing steps in the system, we have to incorporate those requirements in the design of our system. For example, the requirements need to be considered with respect to the factorization method that is used in this thesis. This chapter addresses some of the above-mentioned issues, in particular, blurred image detection, corner detection, triangulation and merging partial reconstructions. Other essential processing modules such as feature point matching and sequence partitioning will be presented separately in the following chapters.

## 3.2 Mathematical formulation

Recovering 3D scene structure from multiple-view images is a classical computer vision problem and has been extensively studied in literature. This section introduces the mathematical formulation underlying the multiple-view 3D reconstruction.

### 3.2.1 Projective geometry

The mapping between the 3D world and a 2D image of commonly-used consumer cameras can be described using the pinhole camera model. As illustrated in Fig. 3.1, the pinhole

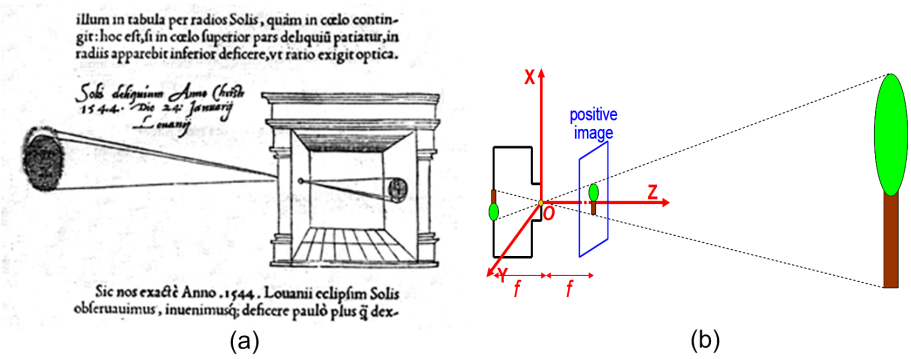



**Figure 3.1:** *Perspective projection by pinhole camera model. (a) Historic drawing of a pinhole camera. (b) Pinhole camera projection.*

camera model is widely used and has been well established in literature [23]. Other camera models also exist, such as the affine camera model that is introduced in the same reference.

As depicted in Fig. 3.2, $n$ 3D points $\{\mathbf{X}_j, j = 1, \cdots, n\}$ are projected onto $m$ perspective cameras $\{\mathbf{P}^i, i = 1, \cdots, m\}$ at different viewpoints $\{\mathbf{C}_i, i = 1, \cdots, m\}$[1]. In this definition, $\mathbf{X}_j = \beta_j(X_j, Y_j, Z_j, 1)^T$ are the homogenous coordinates of 3D point $j$, $\beta_j$ is an arbitrary none-zero scale factor, and $\mathbf{P}^i$ is the projection matrix of camera $i$, which can be represented by

$$\mathbf{P}^i = \alpha_i \mathbf{K}_i \mathbf{R}_i [\mathbf{I} \ -\mathbf{C}_i]. \tag{3.1}$$

In the above equation, $\mathbf{K}_i$ is the $3 \times 3$ intrinsic matrix that contains all internal parameters of a camera such as focal length, aspect ratio, skew and principal point. Parameter $\mathbf{R}_i$ is the $3 \times 3$ camera orientation matrix that specifies the orientation of the camera, and vector $\mathbf{C}_i$ denotes the homogenous coordinate of the camera center. Matrix $\mathbf{R}_i$ and vector $\mathbf{C}_i$ specify the camera orientation and position (hereafter referred to *placement*) and are called the extrinsic camera parameters.

With the defined notations, the mapping between the 3D points and their corresponding 2D projections can be mathematically described as a linear projection, assuming that the radial lens distortion is negligible. The 2D projection of 3D point $\mathbf{X}_j$ in image $i$ can be computed by

$$\lambda_j^i \mathbf{x}_j^i = \lambda_j^i (u_j^i, v_j^i, 1)^T = \mathbf{P}^i \mathbf{X}_j, \tag{3.2}$$

where $\mathbf{x}_j^i = (u_j^i, v_j^i, 1)^T$ are the homogeneous coordinates of the 2D projection, $\lambda_j^i$ is called the *projective depth*. Rewriting Eq. (3.2) into matrix form, we obtain

$$\mathbf{W}_s = \begin{pmatrix} \lambda_1^1 \mathbf{x}_1^1 & \dots & \lambda_n^1 \mathbf{x}_n^1 \\ \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{x}_1^m & \dots & \lambda_n^m \mathbf{x}_n^m \end{pmatrix} = \begin{pmatrix} \mathbf{P}^1 \\ \vdots \\ \mathbf{P}^m \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 \dots & \mathbf{X}_n \end{pmatrix} = \mathbf{P}\mathbf{X}, \tag{3.3}$$

where $\mathbf{W}_s \in \Re^{3m \times n}$ is called the *Scaled Measurement Matrix* (SMM), $\mathbf{P} \in \Re^{3m \times 4}$ is called the *Euclidean motion matrix* that contains the intrinsic and extrinsic parameters of all

[1] In this thesis, we follow the conventional notation where 3D points are denoted with capitals $\mathbf{X}$ and 2D points with lower case $\mathbf{x}$.

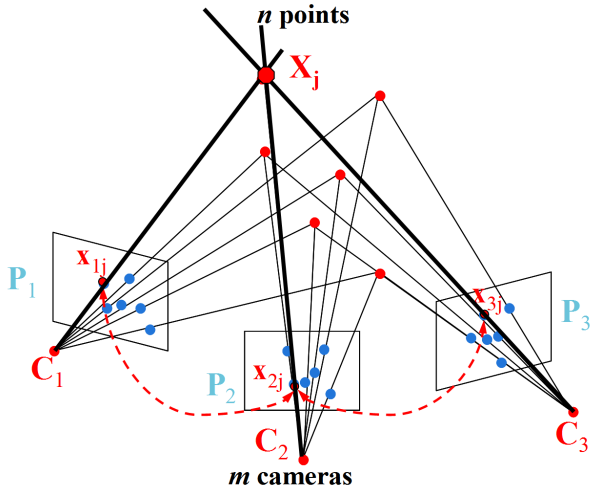



**Figure 3.2:** *$n$ 3D points are projected onto $m$ cameras.*

cameras, and $\mathbf{X} \in \Re^{4\times n}$ is called the *Euclidean shape matrix* that contains the coordinates of all 3D points.

Let us assume an element-wise matrix product between $\lambda$ and 2D point $\mathbf{x}$, hence

$$\mathbf{L} = \begin{pmatrix} \lambda_1^1 & \dots & \lambda_n^1 \\ \vdots & \ddots & \vdots \\ \lambda_1^m & \dots & \lambda_n^m \end{pmatrix} \text{ and } \mathbf{W} = \begin{pmatrix} \mathbf{x}_1^1 & \dots & \mathbf{x}_n^1 \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^m & \dots & \mathbf{x}_n^m \end{pmatrix}, \tag{3.4}$$

then Eq. (3.3) can be rewritten into a condensed form:

$$\mathbf{L} \cdot \mathbf{W} = \mathbf{P}\mathbf{X}. \tag{3.5}$$

Here, $\mathbf{L}$ is called the projective depth matrix, $\mathbf{W}$ is called the Unscaled Measurement Matrix (UMM), and '$\cdot$' denotes the element-wise matrix product. Actually, the UMM $\mathbf{W}$ contains the correspondence information of all tracked feature points in all images, which is usually the only information that we can directly compute from images taken by non-calibrated cameras.

### 3.2.2 Conventional SaM steps

From Eq. (3.5), we can deduce that multiple-view 3D reconstruction can be formulated as the problem of recovering $\mathbf{P}$ and $\mathbf{X}$ given $\mathbf{W}$. As discussed above, the UMM $\mathbf{W}$ contains the correspondence information of all tracked feature points, and is the only information that we can directly obtain from image sequences. The following text summarizes the major steps of a conventional SaM algorithm.

Step 1 *Feature point detection*: detect feature points $\mathbf{x}_j^i$ in individual frames. In this step, the well-known algorithm such as Harris corner detector is used to find salient feature points in individual frames. This detector is discussed in Section 3.3.2.

Step 2 *Feature point matching*: match feature points between every two consecutive frames and then link the two-frame correspondences to obtain multiple-frame correspondences $\mathbf{W}$. In this step, we propose a novel texture-independent feature point matching algorithm to match a large number of feature points between two frames. The details are presented in Chapter 4.

Step 3 *Projective reconstruction*: compute the projective motion $\hat{\mathbf{P}}$ and the projective shape $\hat{\mathbf{X}}$ from $\mathbf{W}$ for the complete set of frames of the same scene. For this computation, the factorization approach is used, which will be introduced in Section 3.2.3.

Step 4 *Euclidean reconstruction*: compute the Euclidean motion $\mathbf{P}$ and the Euclidean shape $\mathbf{X}$ from $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ for the complete scene (see Eq. (3.3)). This step is also referred to as camera calibration, since the Euclidean scene shape can be computed from the projective scene shape as long as the camera parameters are known. This thesis uses a known factorization-based camera calibration algorithm, which is introduced in Section 3.2.4.

Step 1 is further discussed in this chapter, whereas Step 2 will be presented in Chapter 4. In the next subsections, we will discuss Steps 3 and 4 in more detail.

### 3.2.3 Projective reconstruction

Given a set of available image points $\mathbf{x}_j^i$, solving Eq. (3.3) without metric restriction on the camera and scene will yield a $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ that differ from the true Euclidean reconstruction by an unknown projective transformation $\mathbf{H}$. Mathematically, it can be shown that

$$\lambda_j^i \mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j = (\mathbf{P}^i \mathbf{H})(\mathbf{H}^{-1} \mathbf{X}_j) = \hat{\mathbf{P}}^i \hat{\mathbf{X}}_j, \tag{3.6}$$

where $\mathbf{H}$ is an arbitrary $4 \times 4$ non-singular matrix. If image measurements contain noise, point $\hat{\mathbf{X}}_j$ will not project exactly to $\mathbf{x}_j^i$. The difference between the exact point and the re-projected point is called the *re-projection error*. The task of factorization-based projective reconstruction is to find a good $\hat{\mathbf{P}}^i$ and $\hat{\mathbf{X}}_j$, such that the re-projection error between the re-projected point and $\mathbf{x}_j^i$ is minimized. The re-projection error is defined as

$$E_{proj} = \sum_{ij} d(\hat{\mathbf{P}}^i \hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2, \tag{3.7}$$

where $d(\cdot, \cdot)$ denotes the Euclidean distance between the two 2D points. Projective motion and shape can be recovered by, e.g., the iterative factorization method, which will be discussed in Section 5.2.3 of Chapter 5.

Assuming the measurement matrix is Gaussian, the Maximum Likelihood (ML) solution to $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ can be found by minimizing the re-projection error in Eq. (3.7) using *bundle adjustment* [47]. This involves a non-linear minimization process based on a tailored Levenberg-Marquardt algorithm. Bundle adjustment provides a theoretically optimal solution to projective reconstruction, since it provides a true ML estimate while allowing missing data. However, in practice it is difficult to implement. For example, some difficulties are (1)

it requires a good initialization, and (2) it can be very slow due to the large number of parameters for optimization. Because of this, it is generally used as a final step of an algorithm to refine the initial reconstruction.

**Existing work on projective reconstruction**

Many projective reconstruction algorithms have been proposed to estimate the projective depths $\lambda_j^i$ of multiple views. Some algorithms use nonlinear least squares to minimize the re-projection error, while a few use linear least squares [10, 29]. Also, certain algorithms are based on factorization of the SMM [73, 82, 27, 51], a few rely on direct minimization of the re-projection error [10, 29]. Another class of algorithms use the subspace constraint [27, 51], while a few employ the bilinear epipolar constraint [73, 82]. Generally, the factorization method is able to provide a stable solution due to the fact that all data from all images are uniformly utilized. A disadvantage is that it cannot handle missing feature points. The least squares method is better suited for handling missing data. However, a good initialization of the projective depths is usually required. A good discussion of existing projective reconstruction algorithms is provided in [29]. Below, we summarize a few well-known algorithms.

Sturm and Triggs [73] have proposed an algorithm for estimating the projective depths by means of the epipolar constraints between two views, which has the advantage of being non-iterative. However, the method requires the estimation of the fundamental matrix, which is susceptible to data degeneracy. Triggs [82] extends the method to refine the projective depths by iteratively factorizing the SMM to update the projective depths. Heyden [27] has proposed a method for estimating the projective depths using the subspace constraint. Its advantage is that the result is independent of the coordinate representation of image points. Another reconstruction algorithm, based on the subspace constraint is reported by Mahamud and Herbert [51]. Convergence of the iterative algorithms is an important issue, though few publications clearly address the convergence of their proposed algorithms. Discussions of the convergence of the iterative factorization algorithms are reported in [50, 59]. In [29], Huang and Tang propose a least squares method that is guaranteed to converge while it does not require an initialization. This thesis uses the algorithm proposed by Chen and Medioni [10], where the re-projection error is minimized by iteratively performing intersection for individual points, followed by resection for individual cameras. The advantages of Chen's algorithm include its capability of handling missing data and the significantly reduced size of the linear equation system, as well as the good convergence, as demonstrated in his paper. This algorithm is referred to Weighted Iterative Eigen (WIE) algorithm and will be described in Section 5.2.3 of Chapter 5.

### 3.2.4 Euclidean reconstruction

It is shown in [24] that the structure and motion can be recovered from 2D images at most up to a collineation in 3D space, i.e., an unknown 3D projective transformation. To convert the projective reconstruction to Euclidean reconstruction, cameras need to be calibrated, which is called *camera calibration*. For the factorization method, calibrating the cameras is equivalent to computing the unknown $4 \times 4$ matrix $\mathbf{H}$, which is defined in Eq. (3.6). Given $\mathbf{H}$, the Euclidean motion and shape can be directly computed from the projective motion and shape using Eq. (3.6).

Two types of camera parameters need to be recovered, i.e., (1) intrinsic parameters that describe the internal geometry and optical characteristics of the image sensor, and (2) extrinsic parameters that specify the position and orientation of the camera. Both types of camera parameters can be recovered by imposing metric constraints on the camera parameters. The algorithm for recovering $\mathbf{H}$ by imposing metric constraints on the camera parameters will be introduced in Section 5.2.4 of Chapter 5.

**Existing work on camera calibration**

Camera calibration has been well studied in both photogrammetry and computer vision communities, and it is currently considered a 'solved' problem. Various calibration methods have been reported. According to whether external calibration objects are required for the calibration process, calibration algorithms can be classified into automatic approaches [17, 52, 83, 63, 20, 9], or non-automatic approaches [84, 91].

The automatic approach, also called self-calibration or auto-calibration in literature, attempts to find the intrinsic camera parameters by imposing metric constraints on the intrinsic parameters themselves, without assuming properties of the scene structure or the camera motion. This approach provides major advantages in systems like 3D city modeling from image sequences captured by a camera with varying parameters, where it is impossible to calibrate every camera. However, the disadvantage is that self-calibration fails for certain types of camera motion and scene structures (e.g. pure rotation and planar scene), which are called critical motion and surfaces [32, 74]. Auto-calibration methods can be further classified into direct approaches [83, 20, 9], where the Euclidean structure is recovered by directly computing the so-called absolute dual quadric [2], and stratified approaches [63], where the projective structure is first upgraded to an affine structure and then upgraded to an Euclidean structure.

Non-automatic approaches, usually called photogrammetric calibration, attempt to recover the intrinsic camera parameters by observing calibration objects (e.g. calibration checkerboard) whose 2D or 3D geometries are predefined. Unfortunately, within this class, different algorithms require a variable amount of knowledge of the calibration objects. For example, Zhang's approach [91] simply requires the camera to observe a planar pattern in a few unknown orientations, while a few other methods require the coordinates of 3D points in high precision. The advantage of the non-automatic approach is its capability of calibrating all cameras with varying parameters, though its applicability is limited to applications that require online calibrations. A review of a few non-automatic calibration methods is reported in [68], and a survey of a few auto-calibration methods is found in [25].

Since this thesis deals with 3D reconstruction from video captured by non-calibrated consumer cameras, auto-calibration is preferred. For doing this, we have adopted the auto-calibration method of Han [20], where the metric constraints on the internal camera parameters are naturally imposed to compute the unknown projective transformation directly. Han's method is computationally equivalent to Triggs' method [83], which relies on the absolute quadric for camera calibration. The primary reason for using Han's method is its elegant mathematical formulation and its ability to involve all data from all images for calibration.

[2] The method proposed in [20] is not about computing the absolute dual quadric, but it is computationally equivalent to recovering the absolute dual quadric.

## 3.3 Proposed improvements

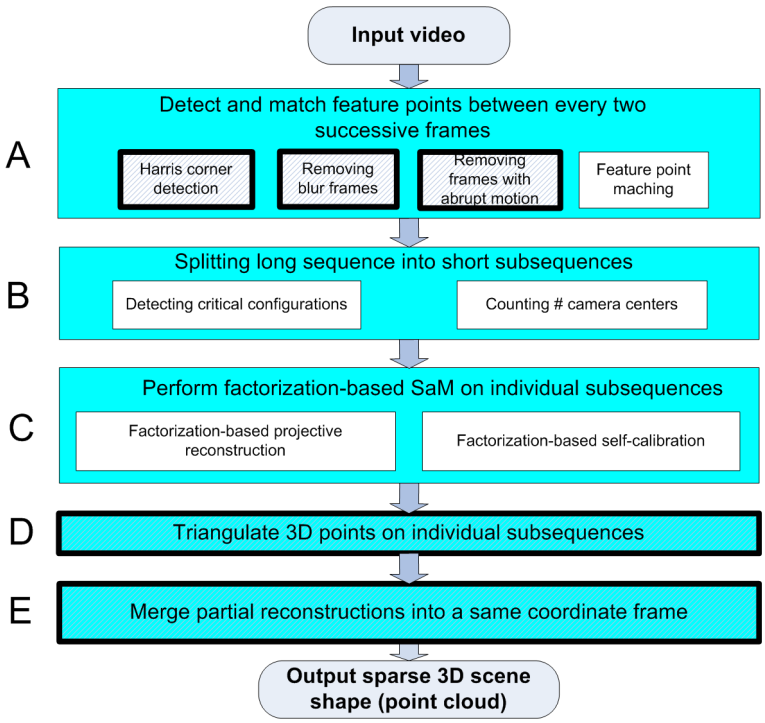



**Figure 3.3:** *Block diagram of the proposed 3D modeling system.*

As discussed in Chapter 2, Fig. 3.3 depicts the block diagram of the proposed 3D modeling system. While an overview of the system has been given in Chapter 2, this section presents our individual contributions to those components that are highlighted by bold rectangles in Fig. 3.3 in Block A, and Blocks D and E. Contributions to other processing components such as *feature point matching* (the remaining step in Block A) and *splitting long sequence* (Block B) will be presented separately in Chapters 3 and 4, respectively.

### 3.3.1 Blur-and-abrupt-frame removal

Given a long sequence captured by a hand-held consumer camcorder, some frames will inevitably contain motion blur or abrupt image motion due to hand movements, which make it difficult to detect and match feature points. Both of them need to be detected prior to subsequent processing. The process to remove blur frames (frames with motion blur) and frames with abrupt motion is referred to as *blur-and-abrupt-frame removal* in this thesis.

This section presents two small supportive algorithms to improve the quality of the feature point matching process and complete the system, as explained in Subsections A and B below. These algorithms are not essential for the complete framework, so that they are not further elaborated.

### A. Removing blur frames

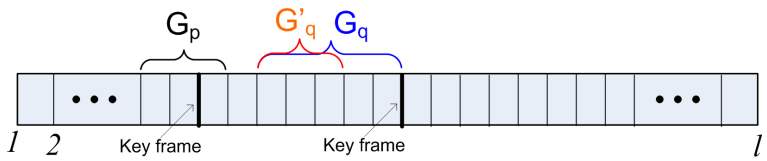



**Figure 3.4:** *Removal of blur frames by selecting key frames from groups of consecutive frames $G_p$.*

Our idea of removing blur frames[3] is to select only one key frame from several neighboring frames, such that the selected frame has the maximum number of feature points. Fig. 3.4 illustrates the selection process. The steps for blur-frame removal are summarized in Alg. 3.3.1, and further elaborated in the succeeding explanation.

---

**Algorithm 3.3.1:** REMOVEBLURIMAGE()

---

**comment:** Remove blur frames with small number of feature points.

Step 1. Divide frames into individual groups of a fixed length of 4 frames ($G_p$ and $G'_q$ in Fig. 3.4).

Step 2. Pick the frame with the maximum number of feature points in current group $G'_q$.

Step 3. If the current maxima of $G'_q$ is lower than the previous maxima of $G_p$ by a threshold, increase the length of frames in current group to have a new group $G_q$ and re-pick the frame.

Step 4. A good frame is picked, if the current maxima is larger than the previous maxima by a threshold, or the maximum length of frames in current group is reached.

---

The above algorithm is visually explained in Fig. 3.4. In the algorithm, a video sequence is divided into groups of 4 frames (at 60 Hz, or 2 at 30 Hz). Only the frame with the maximum number of feature points in a group is selected as the key frame (non-maxima suppression), as indicated in the figure. All remaining frames in the group are removed from the sequence. If the maximum number of the feature points of a group is below a certain amount of points compared with the previous group, the number of frames in the group is increased until the maximum number of the feature points reaches the threshold, or a maximum length of the group is reached. In the figure, the length of group $G_q$ is increased from initially 4 to 6 until a key frame is selected.

[3] In this thesis, we will denote individual pictures extracted from a video sequence as a frame, whereas the term images is applied for pictures taken by a digital camera.

**B. Removing frames with abrupt image motion**

Frames with abrupt image motion due to sudden hand movements degrade the feature point matching performance, so that they need to be removed as well. Our idea of removing frames with abrupt image motion is as follows: whenever we find that the number of detected correspondences between the *previous frame* and the *current frame* is below a threshold, then we perform feature point matching again between the *previous frame* and the *next frame*. If the resulting number of correspondences is larger, the current frame is removed from the sequence. The algorithm is summarized in Alg. 3.3.2.

**Algorithm 3.3.2:** REMOVEABRUPTMOTIONFRAME()

**comment:** $I_p$ = previous frame, $I_c$ = current frame, $I_n$ = next frame.

Step 1. Match points between $I_p$ and $I_c$. If the resulting number of matches is lower than that of the matching between $I_p$ and its preceding frame, match points between $I_p$ and $I_n$, and go to Step 2.

Step 2. If the resulting number of matches between $I_p$ and $I_n$ is larger than that of between $I_p$ and $I_c$, remove $I_c$. Otherwise, keep $I_c$.

The rationale of the above algorithm is that the abrupt image motion caused by the shaking of the hand-held camcorder makes it difficult to match feature points. Thus, by looking into the number of correspondences, abrupt-motion frames can be detected and rejected. The benefit is that the subsequent feature point *tracking* algorithm will be able to track points over such frames, which is critical for the factorization method.

### 3.3.2 Harris corner detector with content-adaptive threshold

For a robust operation of SaM, it is important to detect and match a large number of feature points. Many feature detection and matching algorithms have been proposed in literature. The Harris corner detector [21] is used in our work to detect the feature points, due to its proven performance. However, our experiments have revealed that the original detector does not perform well for frames where the texture varies significantly across the whole frame. In such a case, the detected corners tend to cluster in a small image area. This may lead to the failure of the subsequent processing such as feature point matching and fundamental matrix estimation. This section proposes a simple but effective scheme to make the detected feature points evenly distributed over an image.

For the sake of completeness, we commence with presenting the original Harris corner detector. Subsequently, we adapt the detector and optimize its performance for our purpose.

**A. Harris corner detector**

Feature points should be located in an area of a frame with large intensity variations, such that they can be easily matched and tracked. As illustrated in Fig. 3.5, these points typically

correspond to corners of objects in a frame. The Harris corner detector (also known as Plessy detector) is specifically designed to detect such corners.

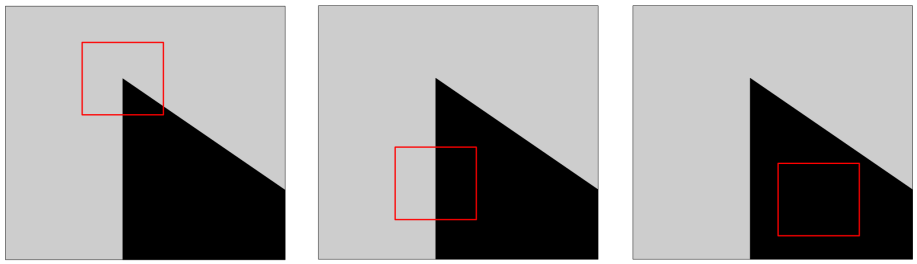



**Figure 3.5:** *Large and small changes of intensity caused by shifting the window in various directions. (a) Shifting around a corner; (b) shifting around an edge; (c) shifting within a flat area.*

The Harris corner detector relies on measuring the local changes of the intensity with patches shifted by a small amount in different directions (the original paper calls this computing the auto-correlation function). Given a pixel shift $(\Delta x, \Delta y)$ of a window $W$ around a pixel $(x, y)$, the intensity change in all directions $E(\Delta x, \Delta y)$ is computed as

$$E(\Delta x, \Delta y) = \sum_{(x_k, y_k) \in W} w(x_k, y_k)[I(x_k + \Delta x, y_k + \Delta y) - I(x_k, y_k)]^2, \tag{3.8}$$

where $I(\cdot, \cdot)$ denotes the intensity function of the frame, and $w(\cdot, \cdot)$ is a window function (usually a Gaussian function), which assigns larger weights to pixels that are closer to the window center. Its purpose is to make the detected corner stay closer to the center of the window [16]. Using the first order of a Taylor expansion of $I(x_k + \Delta x, y_k + \Delta y)$, we need the partial derivatives $\delta I(x_k, y_k)/\delta x$ and $\delta I(x_k, y_k)/\delta y$ of the intensity at pixel $(x_k, y_k)$, which will be denoted in an abbreviated form as $I_x$ and $I_y$, respectively. With this notation, Eq. (3.8) is re-written into

$$\begin{aligned} E(\Delta x, \Delta y) &= \sum_{(x_k, y_k) \in W} w(x_k, y_k)[I_x \Delta x + I_y \Delta y + O^2(\Delta x, \Delta y)]^2 \\ &\approx \sum_{(x_k, y_k) \in W} w(x_k, y_k)[I_x \Delta x + I_y \Delta y]^2 \\ &= \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} \mathbf{M} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \end{aligned} \tag{3.9}$$

with

$$\mathbf{M} = \begin{bmatrix} \sum w I_x^2 & \sum w I_x I_y \\ \sum w I_x I_y & \sum w I_y^2 \end{bmatrix}. \tag{3.10}$$

In the above equations, $w(x_k, y_k)$ is the weighting factor at pixel $(x_k, y_k)$, and the summation is performed over all pixels within $W$. The $2 \times 2$ matrix $\mathbf{M}$, which is called auto-correlation matrix in the original paper, contains the information about the intensity variation within the local neighborhood.

With Eqs. (3.9) and (3.10), the intensity changes $E(\Delta x, \Delta y)$ in all directions can be computed based on the two Eigenvalues $\lambda_1$ and $\lambda_2$ of $\mathbf{M}$. Using $\lambda_1$ and $\lambda_2$, window $W$ can be classified into three categories, as depicted in Fig. 3.5. The classification algorithm is as follows:

1. If both Eigenvalues are large, the intensity variation will be large in all directions. This indicates a corner, as shown in Fig. 3.5(a).
2. If both Eigenvalues are small, the intensity variation will be small in all directions. This indicates a flat region, as shown in Fig. 3.5(c).
3. If one Eigenvalue is large and the other is small, then the intensity variation is large in only one direction. This indicates an edge, as shown in Fig. 3.5(b).

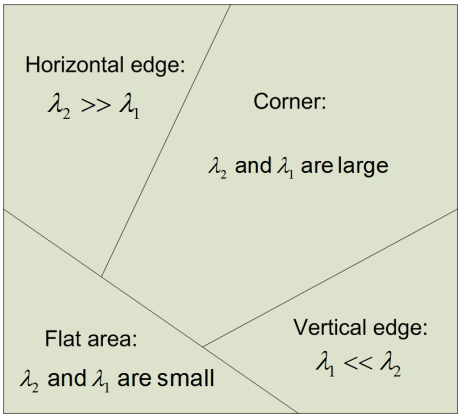



**Figure 3.6:** *Categorizing image point according to the eigenvalues of the auto-correlation matrix* $\mathbf{M}$ *computed by Eq. (3.9).*

To have a quantitative criterion for classification into the above three cases, the following metric is used in our implementation, as is suggested in [56], giving

$$R = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\det \mathbf{M}}{\text{trace } \mathbf{M}}. \tag{3.11}$$

With the above metric, a corner is detected if $R$ is above a certain threshold $T_R$, as illustrated in Fig. 3.6. Metric $R$ is referred to as the *Harris response* and threshold $T_R$ is called the *Harris threshold* in this thesis.

To detect salient feature points and reject points that are less salient, the number of Harris corners are suppressed using so-called non-maxima suppression, which is illustrated in Fig. 3.7. As can be observed from Fig. 3.7, at most one point in a Harris window can be detected as feature point.

### B. Adapting Harris threshold to local image content

Assuming the same Harris threshold and Harris window size for the complete image, the density of the resulting Harris corners varies with the local image content. Image areas with

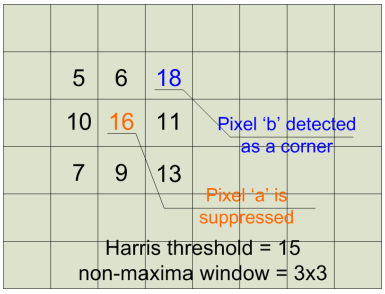



**Figure 3.7:** *Non-maxima suppression of the Harris corners using a* $3 \times 3$ *pixel window. Pixel 'a' is suppressed even though its response is above the Harris threshold, because it is not the maximum within the* $3 \times 3$ *pixel window.*

complex textures may have very dense feature points, while the texture-scarce areas may have only a few. This may pose problem for the subsequent 3D reconstruction process. For example, condensation of feature points in a small image area will increase the chance for the feature points to be located on a planar surface, which will lead to the failure of the factorization-based 3D reconstruction.

The density of the feature points depends on the following three factors: (1) the Harris threshold $T_R$, (2) the size of the Harris window, and (3) the local image content. In our implementation, the Harris window is empirically fixed at $7 \times 7$ pixels. Thus, the only choice to obtain an even distribution of the feature points is to adjust the Harris threshold based on the local image content. The idea is to divide an image into many small blocks and then adapt the Harris threshold for each block, such that we obtain an equal number of feature points in every block within a given range. The proposed Harris corner detector is summarized as follows.

**Algorithm 3.3.3:** HARRISCORNERDETECTOR()

**comment:** Detect Harris corners in an image

Step 1. Specify: (1) the number of points that are expected to be detected for an image, (2) the maximum Harris threshold $T_{max}$, (3) the minimum Harris threshold $T_{min}$, (4) the size $S$ of the Harris window for non-maxima suppression, (5) the total number of blocks $n$ that are contained in the complete image.

Step 2. Compute the Harris response for every pixel using Eqs. (3.10) and (3.11).

Step 3. Perform non-maxima suppression using window size $S$.

Step 4. Divide the image into $n$ blocks. For each block, adjust the Harris threshold within the range of $[T_{min}, T_{max}]$ such that the number of points for each block remains as equal as possible.

For the sake of completeness, we indicate here that, e.g., other interest point detectors can also be used. They have been extensively studied and reported in literature. A detailed review of state-of-the-art interest point detectors can be found in [48, 53] and Chapter 3 of [16].

### 3.3.3 Triangulation

Knowing the camera parameters, we wish to reconstruct as many 3D points as possible from the available feature point correspondences. Theoretically, the coordinates of a 3D point can be computed by intersecting two back-projected rays that pass through the camera centers and the 2D projections in the image plane, as illustrated in Fig. 3.11. In practice, intersection of only two back-projected rays is not reliable, especially when the distance between the two cameras is small or when there is noise in the measurements.

This section proposes a hierarchical scheme for triangulating 3D points with the consideration of the following three factors: (1) measurements may contain noise or outliers, so that bad triangulations have to be removed; (2) long feature point tracks are preferred over short feature point tracks, since they give a more reliable intersection; (3) feature points may disappear and reappear in frames. For a long sequence, redundant triangulations decrease the accuracy and therefore redundant 3D points have to be minimized.

#### A. Linear Iterative Eigen Algorithm

Fig. 3.11 illustrates the triangulation process, where the positions of 3D points are determined by intersecting multiple back-projected rays from several camera centers. In this thesis, the Iterative-Eigen Algorithm (IEA) [22] is used for triangulation, since it works with any number of frames and achieves very good results. The IEA algorithm relies on iterative Eigenvalue decomposition to solve the 3D point that intersects all back-projected rays. This algorithm is briefly introduced below.

From Eq. (3.2), by eliminating the scaling factor $\lambda_j^i$, we obtain

$$u_j^i - \frac{\mathbf{P}_{i(1)}^T \mathbf{X}_j}{\mathbf{P}_{i(3)}^T \mathbf{X}_j} = 0 \text{ and } v_j^i - \frac{\mathbf{P}_{i(2)}^T \mathbf{X}_j}{\mathbf{P}_{i(3)}^T \mathbf{X}_j} = 0, \tag{3.12}$$

where $\mathbf{P}_{i(1)}^T$ is the first row of $\mathbf{P}_i$, and so on. Let $(\mathbf{P}_p, \mathbf{P}_{p+1}, \ldots, \mathbf{P}_q)$ be the reconstructed projection matrices of $l$ (with $l = q - p + 1$) number of cameras, and $(\mathbf{x}_{p,j}, \mathbf{x}_{p+1,j}, \ldots, \mathbf{x}_{q,j})$ be the 2D projections of 3D point $\mathbf{X}_j$ in the $l$ cameras. From Eq. (3.12), we have:

$$\begin{bmatrix} (u_{pj}\mathbf{P}_{p(3)}^T - \mathbf{P}_{p(1)}^T)/w_{pj} \\ (v_{pj}\mathbf{P}_{p(3)}^T - \mathbf{P}_{p(2)}^T)/w_{pj} \\ \vdots \\ (u_{qj}\mathbf{P}_{q(3)}^T - \mathbf{P}_{q(1)}^T)/w_{qj} \\ (v_{qj}\mathbf{P}_{q(3)}^T - \mathbf{P}_{q(2)}^T)/w_{qj} \end{bmatrix} \mathbf{X}_j = \mathbf{A}_j \mathbf{X}_j = 0, \tag{3.13}$$

where $w_{ij} = \mathbf{P}_{i(3)}^T \mathbf{X}_j$ is a weighting factor that makes the residue $\epsilon_j = ||\mathbf{A}_j \mathbf{X}_j||$ correspond to the re-projection error as defined in Eq. (3.12). One way to solve Eq. (3.13) is to find an

$\mathbf{X}_j$ such that $\epsilon_j$ is minimized subject to $||\mathbf{X}_j|| = 1$. The solution is the unit eigenvector corresponding to the smallest eigenvalue of matrix $\mathbf{A}_j^T\mathbf{A}_j$, which can be obtained using singular value decomposition. After we obtain the first $\mathbf{X}_j$, weight $w_{ij}$ can be updated, and iterations are repeated until the change of $w_{ij}$ between two successive iterations is sufficiently small, or a maximum number of iterations have been executed.

Each camera provides two constraints for the solution, i.e., two rows in Eq. (3.13). The two rows of $\mathbf{A}_j$ for camera $\mathbf{P}^i$ define the back-projected ray passing through the camera center of $\mathbf{P}^i$ and the 2D projection $(u_j^i, v_j^i)$ in the image plane. The geometric meaning of solving Eq. (3.13) is equivalent to determining the intersecting point of the $l$ back-projected rays from $l$ cameras, by minimizing the re-projection error.

If measurements are error-free, the $l$ back-projected rays will intersect at an exact point. In that case, the re-projection error will be zero, and $\mathbf{A}_j$ will be of rank three. However, due to the noise in the measurements and the inaccuracy of camera calibration, as well as the degeneracy of the configurations (e.g. camera undergoes pure rotation), the back-projected rays will not intersect at a single point, which leads to a non-zero re-projection error. The bad triangulations can be rejected if the resulting re-projection error is above a given threshold.

### B. Hierarchical triangulation scheme

Feature points can be triangulated from feature point tracks with varying lengths. Long feature point tracks are preferred over short feature point tracks for a better triangulation accuracy. This section proposes a hierarchical triangulation scheme that triangulates 3D points with decreasing lengths of the feature point tracks.

As shown in Fig. 3.8, 3D points are triangulated in multiple Triangulation Passes (TPs) with decreasing lengths of the feature point tracks. In the figure, $\text{TP}_1$ has the longest length and $\text{TP}_n$ is with the shortest length. The length of the feature point track is decremented by one frame between two consecutive passes. In the same pass, the two consecutive feature point tracks overlap with each other by one frame, in order to reduce redundant triangulations. The steps of each individual triangulation pass are presented in Alg. 3.3.4.

**Algorithm 3.3.4:** TRIANGULATION()

**comment:** Triangulate 3D points in a specific Triangulation Pass

Step 1. Suppose the length of the feature point track used in current TP is $l_{TP_i}$. Starting from frame 1, track feature points from frame 1 to frame $l_{TP_i}$, and then perform the triangulation using the IEA algorithm presented in the previous section.

Step 2. Starting from frame $l_{TP_i}$, track feature points from frame $l_{TP_i}$ to frame $2l_{TP_i} - 1$, and then perform the triangulation using the IEA algorithm. Note that during feature point tracking, the feature points that have been tracked in Step 1 are excluded in this step, in order to reduce redundant triangulations. This process is repeated until the end of the sequence is reached.

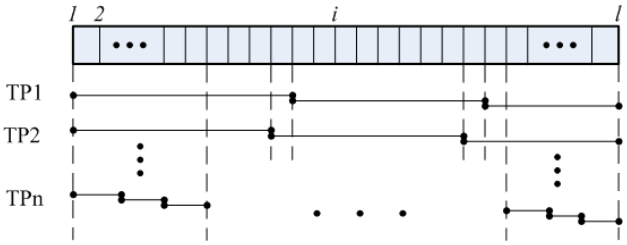



**Figure 3.8:** *Hierarchical triangulation scheme; $TP_1$ - $TP_n$ denote the triangulation passes using different lengths of feature point tracks. The numbers at the top refer to the frame numbers. The index of TP indicates the execution sequence of the pass.*

Using the hierarchical triangulation scheme, 3D points will be triangulated with a decreasing accuracy until a required number of 3D points is obtained. To reduce redundant triangulations, within a same TP, the sequence is divided into multiple sequences that overlap by only one frame. In this way, redundant triangulations can be minimized by ensuring that no 3D point is projected onto the same feature point in the overlapped frame.

To summarize, the proposed hierarchical triangulation scheme provides two benefits: (1) the algorithm starts with long feature point tracks giving the highest triangulation accuracy, followed by short tracks; (2) redundant triangulations are minimized by excluding feature point tracks that pass through the same feature points in the overlapped frame.

### C. Removing redundant triangulation

For a long sequence, feature points may disappear and reappear in frames. As a consequence, the same 3D point may be triangulated from multiple feature point tracks. Such redundant triangulations decrease the accuracy of the reconstructed 3D points, since positions of the same 3D point triangulated from different feature point tracks will not be exactly identical. They are slightly displaced in 3D space. This subsection describes the proposed algorithm for removing the redundant triangulations.

As shown in Fig. 3.9, the visibility information of all reconstructed 3D points is represented with a so-called *visibility matrix* $\mathbf{V}$, of which row $j$ corresponds to 3D point $\mathbf{X}_j$ and column $i$ corresponds to frame $i$, and element $\mathbf{V}(i,j)$ records the index of the feature point in frame $i$ where 3D point $\mathbf{X}_j$ projects onto. Note that $\mathbf{V}(i,j)$ is set to $-1$ if point $\mathbf{X}_j$ is not visible in frame $i$.

Two equally valued elements in the same column of the visibility matrix indicate that two 3D points are projected onto the same feature point in that frame. Therefore, one of the two 3D points is redundant. By detecting such equally valued elements in each column, redundant 3D points are detected. As illustrated in Fig. 3.9, both two 3D points indexed by 2 and $n-1$ are projected onto feature point with 256 in frame frm_3.

After redundant 3D points are detected, they can be removed according to multiple criteria, for example, by keeping the 3D point with the smallest re-projection error, or by keeping

the point with the longest feature point track, or by averaging the coordinates of all redundant 3D points. In the current implementation, we have chosen to keep the 3D point with the longest feature point track, since, as explained above, long feature point tracks give a higher triangulation accuracy.

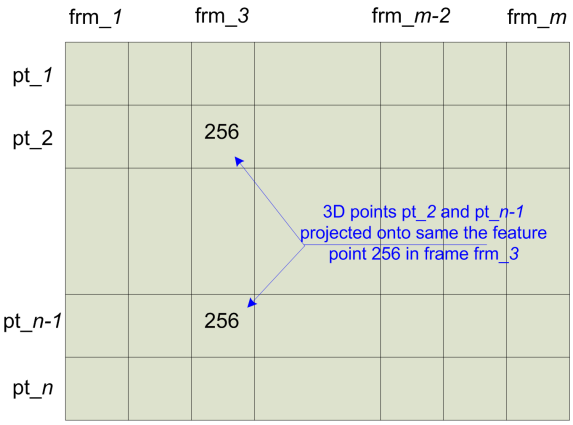



**Figure 3.9:** *Visibility matrix that contains the visibility information of all reconstructed 3D points (indicated at row positions) in all frames (indicated at column positions). The matrix involves $n$ 3D points and $m$ frames. 3D points 2 and $n-1$ are detected as redundant triangulations, since they are projected onto same feature point 256 in frame frm_3.*

It should be noted that the above algorithm cannot remove all redundant 3D points. For example, if a feature point disappears at some frames and re-appears at later frames, the proposed algorithm will not be able to detect it, since the reconstructed 3D points do not project into a common frame.

### 3.3.4 Merging partial reconstructions

Using the factorization method, scene shapes reconstructed from individual subsequences are located in different coordinate systems. To obtain a complete 3D model of the whole scene, individual partial reconstructions have to be merged into a single coordinate system. This section presents the proposed algorithm for merging such partial reconstructions.

Given two different reconstructions $(\mathbf{P}^{i1}, \mathbf{X}_{j1})$ and $(\mathbf{P}^{i2}, \mathbf{X}_{j2})$ of the same 3D point $\mathbf{X}_j$ and the same camera $\mathbf{P}^i$, it is known that the two reconstructions are related by an unknown 3D projective transformation $\mathbf{H}$ that, according to Eq. (3.2), satisfies

$$\mu_i \mathbf{P}^{i2} = \mathbf{P}^{i1}\mathbf{H}^{-1} \text{ and } \nu_j \mathbf{X}_{j2} = \mathbf{H}\mathbf{X}_{j1}, \tag{3.14}$$

where $\mu_i$ and $\nu_j$ are two non-zero scaling factors. Thus, if $\mathbf{H}$ is known, the two reconstructions can be aligned into the same coordinate system.

From Eq. (3.14), we see that $\mathbf{H}$ can be computed either from the camera-to-camera correspondences $(\mathbf{P}^{i1} \leftrightarrow \mathbf{P}^{i2})$, or from the 3D point-to-point correspondences $(\mathbf{X}_{j1} \leftrightarrow$

$\mathbf{X}_{j2}$), as shown in Fig. 3.11. In our proposal, we have chosen to use the 3D point-to-point correspondences, since the 3D points are generally more evenly distributed in 3D space than the camera positions, yielding a more stable solution to the computation of $\mathbf{H}$.

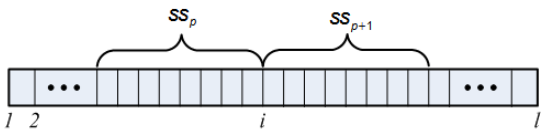



**Figure 3.10:** *Division of a long image sequence containing $l$ frames, into multiple short subsequences. Frame $i$ belongs to both two consecutive subsequences $ss_p$ and $ss_{p+1}$.*

To find a sufficient number of 3D point-to-point correspondences for computing $\mathbf{H}$, a long sequence is divided into consecutive subsequences that overlap with each other by a number of frames. As shown in Fig. 3.10, the two consecutive subsequences $ss_p$ and $ss_{p+1}$ overlap by one frame at frame $i$. In this way, the 3D point-to-point correspondences can be easily found by checking whether 3D points are projected onto the same feature points in the common frame. As illustrated in Fig. 3.11, the 3D point-to-point correspondence between $\mathbf{X}$ and $\mathbf{X}'$ is easily found, because $\mathbf{X}$ and $\mathbf{X}'$ are triangulated from two feature point tracks that contain the same feature point $\mathbf{x}_3$ in the overlapped frame with camera center $\mathbf{C}_3$.

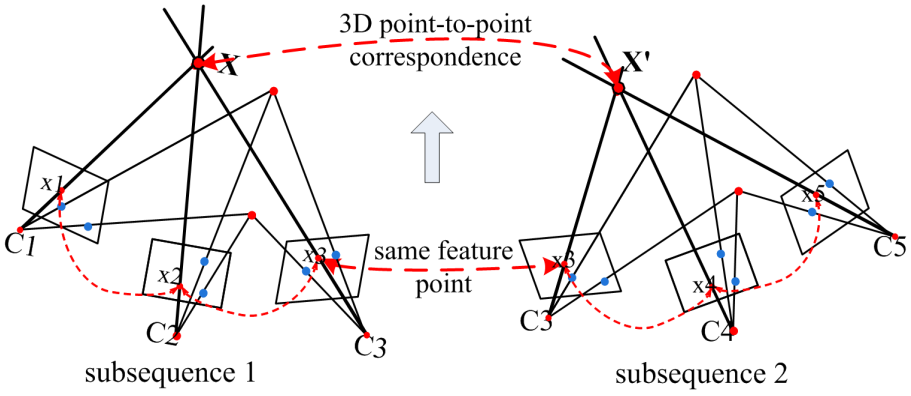



**Figure 3.11:** *The 3D point $\mathbf{X}$ is triangulated from the feature point track $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ in subsequence 1, and the 3D point $\mathbf{X}'$ is triangulated from the feature point track $(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5)$ in subsequence 2. The two subsequences overlap at the common frame with camera center $\mathbf{C}_3$.*

## 3.4 Algorithm steps and experimental results

### 3.4.1 Test sequences

The performance of the proposed system is evaluated for two sequences that were captured using a consumer hand-held camcorder[4]. Table 3.1 lists the two sequences tested in this section. Among the two sequences, *vca* is an indoor sequence of which the scene dimensions can be easily measured. Thus, *vca* is mainly used to evaluate the accuracy of the reconstructed point cloud. The *campus* is an outdoor sequence that contains more than 3,000 frames. It is mainly used for demonstrating the effectiveness of the proposed algorithm for an automatic SaM for a long sequence. Both sequences are recorded in mp4 format and need to be decoded into the raw YUV format. For *campus*, the resolution is first downsized to $1280 \times 720$ pixels prior to further processing. Two example frames of the *vca* and *campus* sequences are shown in Figs. 3.12 and 3.15(a), respectively.

**Table 3.1:** Two video test sequences: *vca* and *campus* captured with the same camera.

| sequence | frame rate (fps) | #frames | resolution (pixels) | example frame |
|---|---|---|---|---|
| *vca* | 30 | 321 | $1024 \times 768$ | Fig. 3.12 |
| *campus* | 60 | 3523 | $1920 \times 1080$ | Fig. 3.15(a) |

### 3.4.2 Reconstruction results for short sequences

Depending on lengths of the sequences, the implementation steps of the proposed scene-reconstruction algorithm are different. For a short sequence such as *vca*, where a sufficient number of feature points can be tracked along the whole sequence, processing steps such as *dividing long sequence* and *merging partial reconstructions* are skipped. This leads to the Alg. 3.4.1 for scene reconstruction for short sequences.

It should be noted that in the above steps, Steps 2 and 3 can be skipped for sequences taken using a digital camera, because in that case, the sequence will contain almost no blur frames. Furthermore, removing abrupt-motion frames may not be necessary, since the sequence may contain large motion between every two consecutive frames.

As shown in Table 3.1, sequence *vca* contains 321 frames. After blur-and-abrupt-frame removal in Steps 2 and 3, only 141 frames remain. Reconstruction is performed on the 141 frames, and 152 points are reconstructed. The reconstruction results are depicted in Figs. 3.13 and 3.14. Note that the Harris window size is $7 \times 7$ and the Harris threshold is set to $50$ for the Harris corner detection in Step 1.

[4]This camera is commercially available with type number VPC-FH1 from Sanyo.

**Algorithm 3.4.1:** SCENERECONSTRUCTIONFROMSHORTSEQUENCE()

**comment:** Algorithm steps for SaM for short sequences.

Step 1. Detect feature points in each frame using the algorithm described in Section 3.3.2.

Step 2. Remove blur frames using the algorithm, as described in Section 3.3.1.

Step 3. Remove frames with abrupt image motion using the algorithm that is described in Section 3.3.1.

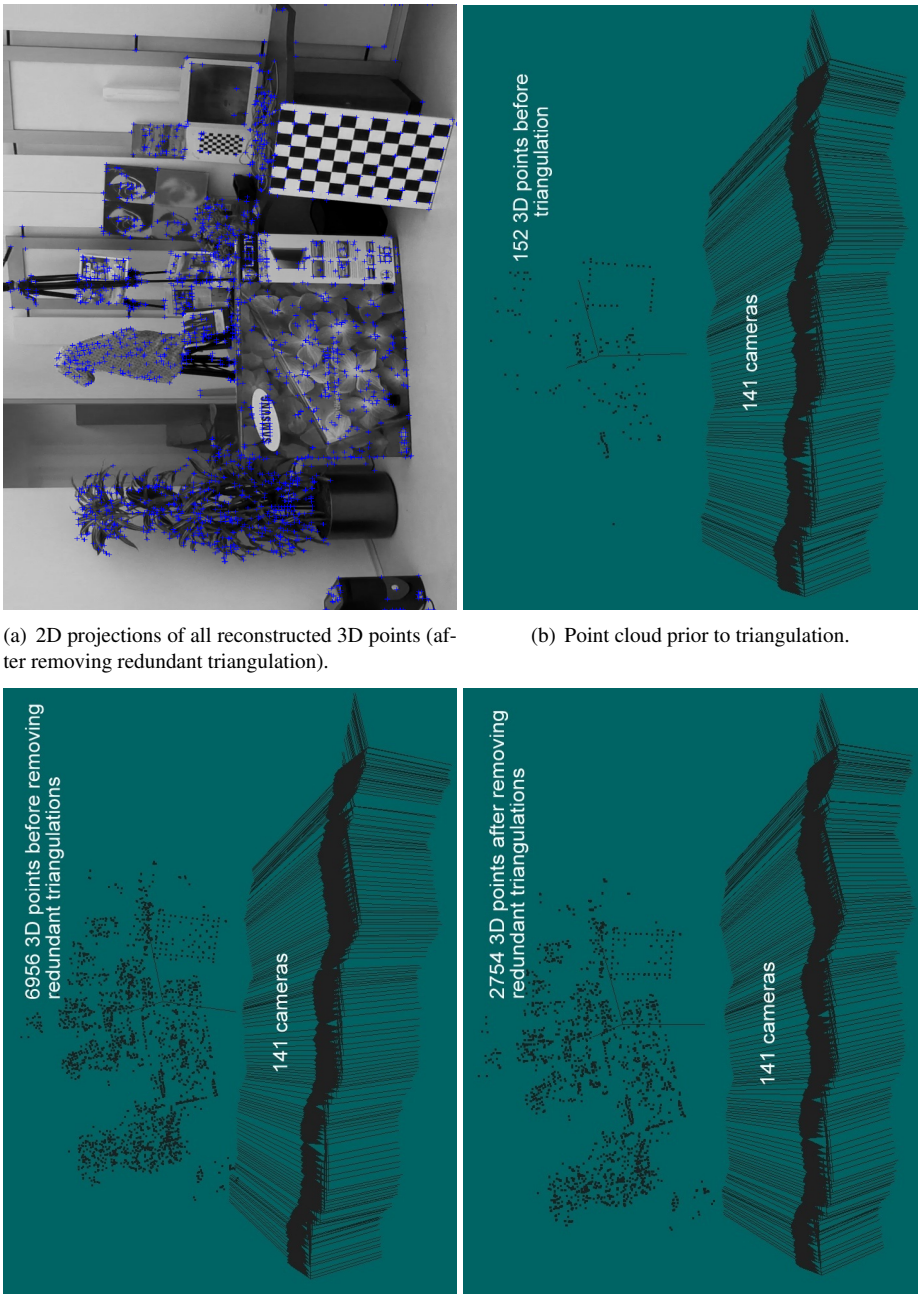
Step 4. Match feature points between every two successive frames using the algorithm that will be presented in Chapter 4. Link the two-frame correspondences to obtain feature point tracks along the whole sequence.

Step 5. Carry out the projective reconstruction to recover the projective motion and shape.

Step 6. Upgrade the projective shape to Euclidean shape via self-calibration of the camera.

Step 7. Triangulate 3D points using the algorithm, as presented in Section 3.3.3.


**Figure 3.12:** *Eight points* $(a, b, \cdots, h)$ *are selected for measurement. The lengths of the eight line segments* $(l_{ab}, l_{bc}, \cdots, l_{he})$ *and the eight corresponding orthogonal angles* $(\theta_{ad\_ab}, \theta_{ba\_bc}, \cdots, \theta_{hg\_he})$ *are measured for evaluating the accuracy of the reconstructed point cloud.*



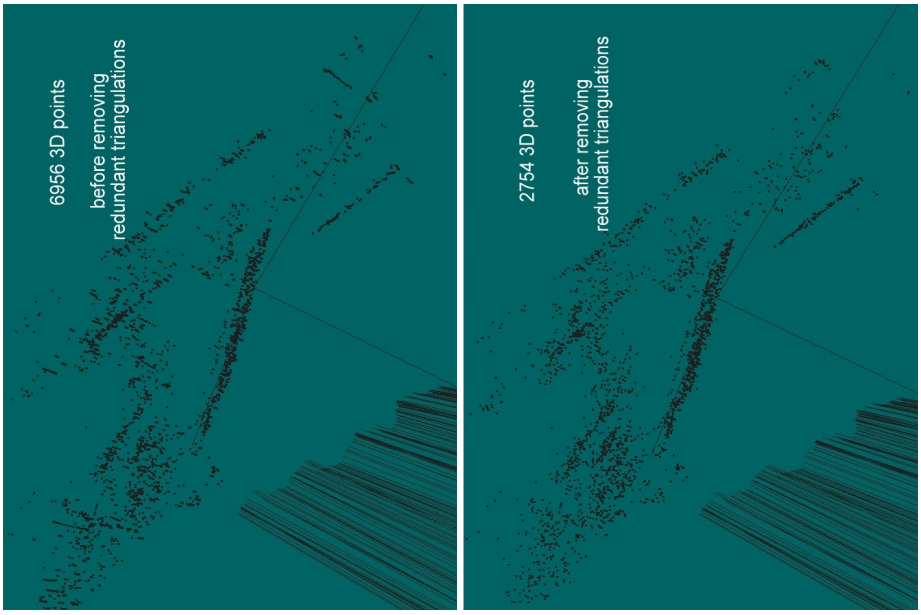

(a) 2D projections of all reconstructed 3D points (after removing redundant triangulation).

(b) Point cloud prior to triangulation.

(c) Point cloud after triangulation but prior to removing redundant triangulations.

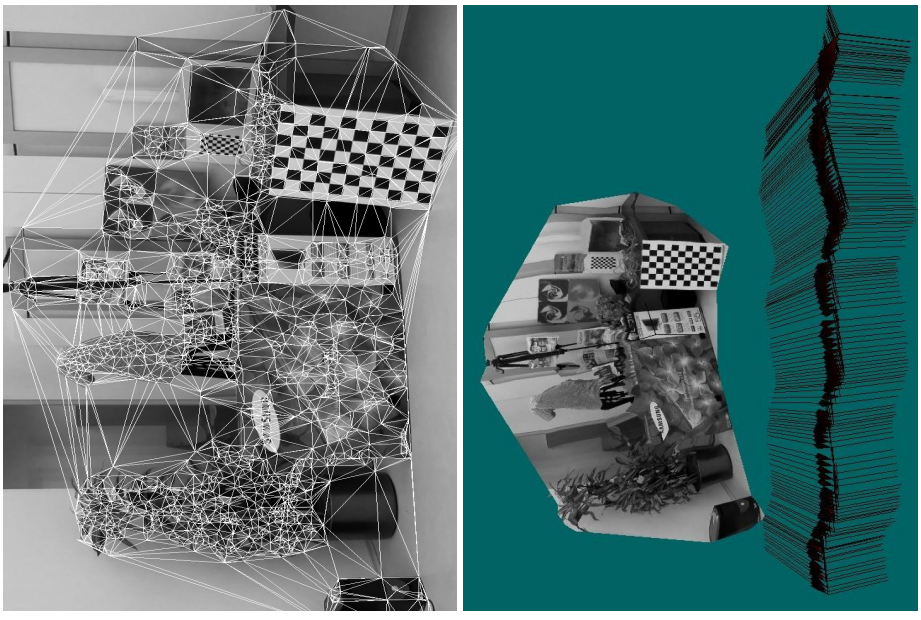
(d) Point cloud after removing redundant triangulations.

**Figure 3.13:** Reconstruction results for the *vca* sequence - I.




(a) Prior to removing redundant triangulation, many redundant 3D points are observed, most of which are located on the back-projected rays from camera centers.

(b) After removing redundant triangulation, almost no redundant 3D points are observed.



(c) Triangular mesh on the 2D projections for texture mapping.

(d) Texture-mapped point cloud.

**Figure 3.14:** Reconstruction results for the *vca* sequence - II.

**Table 3.2:** *Coordinates of the 8 selected feature points (a-h) and the 8 corresponding reconstructed 3D points. Note that the factorization method automatically puts the coordinate system to the geometry center of the scene (explaining the negative coordinates of the 3D points)*

| | 2D coordinates (pixels) | | 3D coordinates | | |
|---|---|---|---|---|---|
| a | 422.171 | 879.926 | 1.30841 | 15.2401 | 3.3054 |
| b | 441.605 | 1069.45 | -1.56019 | 29.0688 | 8.14803 |
| c | 712.224 | 1035.27 | -20.8327 | 22.7806 | 12.9316 |
| d | 675.236 | 846.348 | -18.1444 | 9.18434 | 7.8855 |
| e | 406.882 | 692.569 | 4.3271 | -0.406416 | -1.39986 |
| f | 404.615 | 793.48 | 2.83071 | 8.16546 | -3.3924 |
| g | 615.42 | 786.628 | -15.7692 | 4.42722 | -2.16584 |
| h | 620.353 | 688.369 | -14.456 | -3.94053 | -0.0990409 |

**Table 3.3:** *Comparison between the computed and measured lengths of the 8 line segments defined by the 8 reference points in Fig. 3.12. The computed lengths are multiplied by the same scale factor such that the computed length of $l_{ab}$ equals its measured length.*

| | $l_{ab}$ | $l_{bc}$ | $l_{cd}$ | $l_{da}$ | $l_{ef}$ | $l_{fg}$ | $l_{gh}$ | $l_{he}$ |
|---|---|---|---|---|---|---|---|---|
| computed | 60.10 | 83.84 | 59.37 | 84.06 | 35.93 | 76.53 | 35.10 | 77.11 |
| measured | 60.10 | 80.10 | 60.05 | 80.10 | 35.85 | 74.40 | 35.90 | 74.40 |
| delta (%) | 0.0 | 4.7 | -1.1 | 4.9 | 0.2 | 2.9 | -2.2 | 3.6 |

To show the effect of the triangulation scheme as proposed in Section 3.3.3, Figs. 3.13(b), 3.13(c) and 3.13(d) show the reconstructed point cloud prior to triangulation, the point cloud after triangulation but prior to removing redundant triangulations, and the point cloud after triangulation and after removing redundant triangulations. As can be observed from the figures, the number of 3D reconstructed points is significantly increased using triangulation (from 152 to 6,956 points). Evidently, after removing redundant triangulations, most redundant 3D points are removed, and only 2,754 3D points are left, as depicted in Fig. 3.13(d).

Without the knowledge about the scene dimensions and orientations with respect to a 3D coordinate system, a scene can be reconstructed at best up to the model of similarity transformation (with an unknown scale and orientation) from image measurements [23]. With a similarity reconstruction, only the relative lengths and angles can be measured. If the scale of the scene is known, then the scene can be determined up to a Euclidean transformation, with unknown rotation and translation. In that case, the absolute length can also be measured. Consequently, the accuracy of the Euclidean reconstruction obtained by the proposed algorithm[5] can be evaluated by measuring the lengths and angles of some known structures in the scene.

As shown in Fig. 3.12, eight reference points (a-h) are selected for measurement. The coordinates of the eight 2D feature points and the coordinates of the eight corresponding 3D points are listed in Table 3.2. The comparison between the measured and reconstructed

[5]The scale of the scene can be determined once we are able to measure a single distance between any two scene points. For *vca*, this can be easily done, and thus Euclidean reconstruction is achieved.

**Table 3.4:** *Comparison between the computed and measured values of the 8 orthogonal angles defined in Fig. 3.12.*

| | $\theta_{ad\_ab}$ | $\theta_{ba\_bc}$ | $\theta_{cb\_cd}$ | $\theta_{dc\_da}$ | $\theta_{eh\_ef}$ | $\theta_{fe\_fg}$ | $\theta_{gf\_gh}$ | $\theta_{hg\_he}$ |
|---|---|---|---|---|---|---|---|---|
| computed | 91.06 | 88.43 | 91.78 | 88.71 | 91.60 | 87.75 | 93.25 | 87.39 |
| measured | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 |
| delta (%) | 1.2 | -1.7 | 2.0 | -1.4 | 1.8 | -2.5 | 3.6 | -2.9 |

lengths and angles are presented in Tables 3.3 and 3.4, respectively. As we can derive from the two tables, the lengths and angles are reconstructed with a high accuracy. The maximum error of the length and angle between the reconstructed and measured parameters is $3.6\%$. One of the possible sources for errors is the inaccurate positioning of the reference points. For example, the eight selected points in Fig. 3.12 are not exactly positioned in the eight corners of the scene objects. However, our measurement assumes that they are exactly positioned. If we take this into account, the actual accuracy of the reconstruction would even be higher. If we look into Fig. 3.12 more closely, feature points appear to be more closely located to the horizontal edges compared to the spacing to the vertical edges. This also explains the observation that the reconstructed vertical dimensions in Tables 3.3 and 3.4 appear less accurate than the horizontal dimensions.

### 3.4.3 Reconstruction results for long sequences

For long sequences such as *campus*, all processing steps shown in Fig. 3.3 are required. The implementation steps for reconstruction of long sequences are summarized in Alg. 3.4.2.

Alg. 3.4.2 is applied to the *campus* sequence. After blur-and-abrupt frame removal in Steps 2 and 3, 1,561 frames remain in the sequence. Step 4 matches feature points between every two successive frames. The detected feature points and correspondences are saved to files for subsequent processing. The feature point matching algorithm will be presented in detail in Chapter 4.

**Table 3.5:** *Eight subsequences obtained by dividing the* campus *sequence comprising 1,561 frames.*

| subsequence | frame range | rproj err4 (pixels) | rank4 (%) | rproj err3 (pixels) | rank3 (%) |
|---|---|---|---|---|---|
| sub1 | 0 →186 | 0.27 | 99.5 | 12.6 | 80.7 |
| sub2 | 186 →372 | 0.23 | 98.7 | 2.9 | 96.2 |
| sub3 | 372 →561 | 0.22 | 99.2 | 6.0 | 93.7 |
| sub4 | 489 →693 | 0.25 | 99.5 | 9.9 | 88.6 |
| sub5 | 693 →913 | 0.21 | 99.2 | 3.6 | 95.2 |
| sub6 | 810 →1083 | 0.23 | 99.3 | 11.3 | 92.0 |
| sub7 | 1083→1305 | 0.28 | 98.3 | 7.5 | 94.3 |
| sub8 | 1305→1560 | 0.30 | 95.6 | 5.6 | 94.8 |

**Algorithm 3.4.2:** SCENERECONSTRUCTIONFROMLONGSEQUENCE()

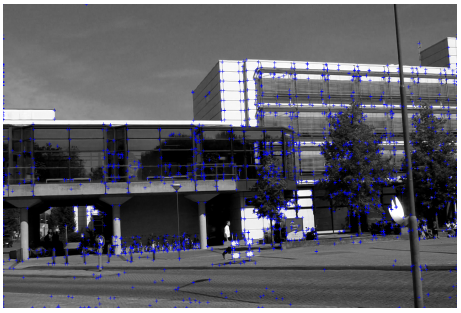
**comment:** Algorithm steps of SaM for long sequences.

Step 1. Detect feature points in each frame using the algorithm, as described in Section 3.3.2.
Step 2. Remove blur frames using the algorithm as described in Section 3.3.1.
Step 3. Remove frames with abrupt image motion using the algorithm, as described in Section 3.3.1.
Step 4. Match feature points between every two consecutive frames using the algorithm that will be presented in Section 4.3.
Step 5. Divide the long sequence into multiple subsequences, using the algorithm that will be presented in Section 5.3. During this step, projective reconstruction is carried out on each subsequence, and the projective motion and shape of each subsequence are reconstructed.
Step 6. Upgrade the projective shape to Euclidean shape via self-calibration of the camera for individual subsequences.
Step 7. Triangulate 3D points for individual subsequences using the algorithm, as presented in Section 3.3.3.
Step 8. Merge subsequences into a single coordinate system using the algorithm, as described in Section 3.3.4.

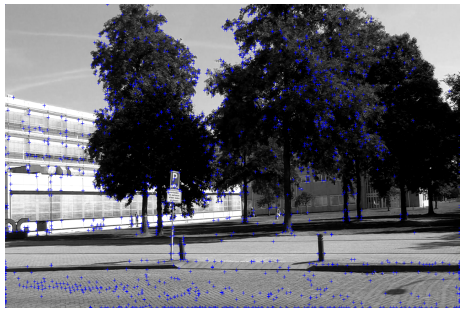
Step 5 divides *campus* into eight subsequences which are listed in Table 3.5. The meaning of the table will be discussed in Chapter 5. The reconstruction results from *sub1* and *sub3* are shown in Fig. 3.15, and the reconstruction results from *sub6* and *sub8* are shown in Fig. 3.16. The merged point cloud of all subsequences are shown in Fig. 3.17. In this figure, the point clouds reconstructed from eight subsequences are merged into the same coordinate system using the algorithm from Section 3.3.4.

Since we do not know the dimensions of the scene, the accuracy of the reconstructed 3D points of the individual subsequences cannot be evaluated. However, the reconstruction accuracy can be derived from the results for the *vca* sequence, since the same reconstruction algorithm is used. This experiment is used to illustrate that the proposed algorithm is capable to divide a long sequence into subsequences and merge partial reconstructions of those subsequences into a complete reconstruction of the scene. This experiment also shows that by dividing a long sequence into subsequences, a repetitive reconstruction is performed on individual subsequences, so that the testing of the algorithm becomes more elaborate. Visual inspection of the reconstructed point clouds shows that the 3D scene is accurately reconstructed. Relative lengths and orthogonality are well recovered in this experiment.

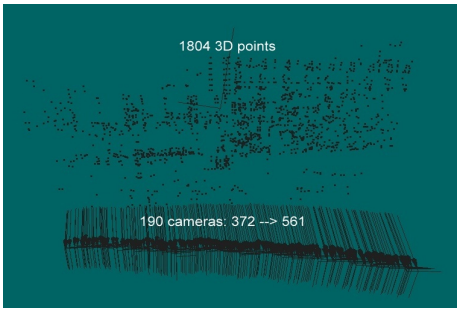
If we more closely inspect the merged results shown in Fig. 3.17, we observe that the merging algorithm proposed in Section 3.3.4 does not merge the subsequences perfectly. Although the orientations and positions of the cameras are generally well aligned, we can clearly observe gaps between subsequences, for example, between *sub2* and *sub3*. Further



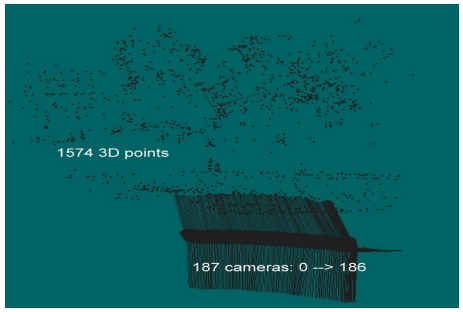
(a) 2D projections of all reconstructed 3D points for *sub3*.



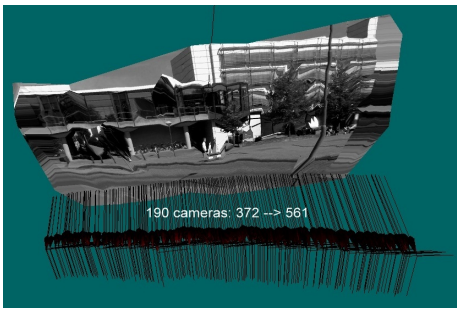
(b) 2D projections of all reconstructed 3D points for *sub1*.



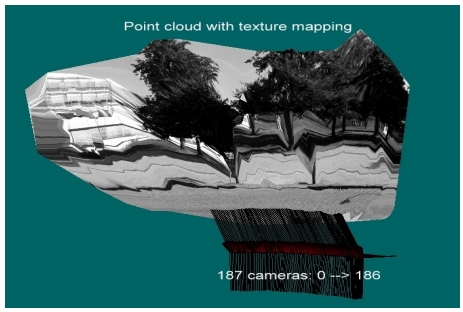

(c) Reconstructed point cloud for *sub3*.




(d) Reconstructed point cloud for *sub1*.



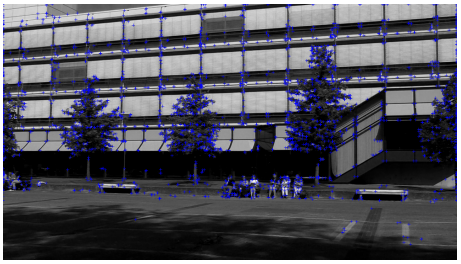

(e) Texture-mapped point cloud for *sub3*.




(f) Texture-mapped point cloud for *sub1*.

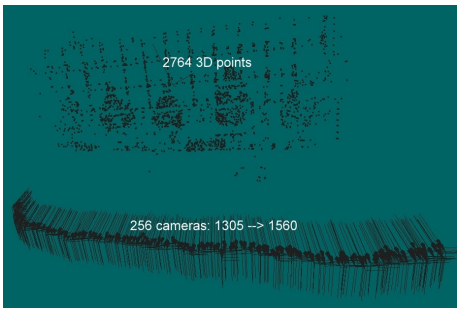
**Figure 3.15:** Reconstruction results from subsequences *sub3* and *sub1*.

processing such as bundle adjustment, is expected to improve the quality significantly. Bundle adjustment is not implemented in this thesis partly due to time limitation and because it is not the focal point of our contribution. The results show that the proposed algorithm is able to reconstruct the scene of a long sequence comprising thousands of frames automatically.

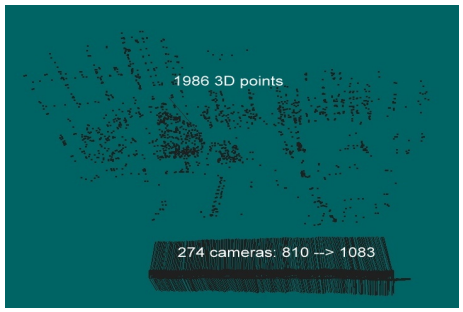


(a) 2D projections of all reconstructed 3D points for *sub8*.

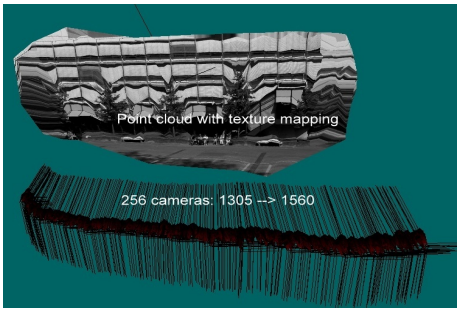
(b) 2D projections of all reconstructed 3D points for *sub6*.



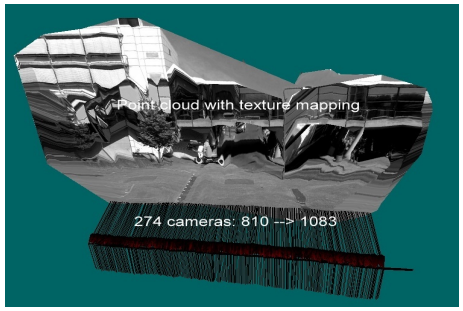

(c) Reconstructed point cloud for *sub8*.




(d) Reconstructed point cloud for *sub6*.




(e) Texture-mapped point cloud for *sub8*.




(f) Texture-mapped point cloud for *sub6*.

**Figure 3.16:** Reconstruction results from subsequences *sub8* and *sub6*.

## 3.5 Conclusion

This chapter has presented a system for reconstructing a sparse 3D model for long video sequences captured with a hand-held camcorder. An automatic 3D reconstruction for long video sequences is complicated by several factors, of which a selection is addressed in this chapter. More specifically, we have addressed (1) robust feature point tracking, where the abrupt motion and blur in the frames that cause difficulties in tracking need to be detected, (2) the condensation of feature points in some images with varying texture making it difficult to obtain a robust 3D reconstruction, (3) the quality and density of the reconstructed 3D points,

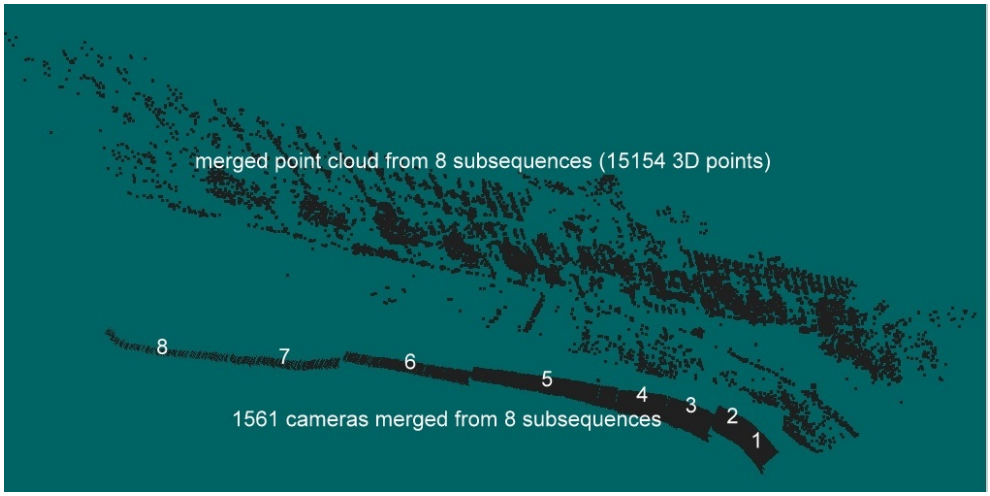



**Figure 3.17:** *Point cloud merged from eight subsequences.*

and (4) the merging of partial reconstructions to constitute a 3D model for the complete scene of the long video sequence.

With respect to the robust feature point tracking, we have designed an algorithm for detecting and removing the blur frames and frames with abrupt image motion, based on intelligently controlling the number of feature points or the number of correspondences over the frames. By requiring that the number of feature points in successive frames does not vary larger than a certain threshold, the blur frames giving less feature points can be removed. By imposing a continuity constraint requiring that the number of feature point correspondences between two successive frames should not vary larger than a given threshold, the frames with abrupt motion yielding less feature point correspondences can be removed. Both measures contribute to a more robust feature point matching and tracking, which is crucial for factorization-based 3D reconstruction for long sequences.

With respect to the condensation of feature points in some frames, an adaptive-threshold control has been proposed to improve the Harris corner detector such that the detected feature points are more uniformly distributed in a frame. This measure further contributes to the robustness of the factorization-based 3D reconstruction.

With respect to the density and quality of the reconstructed 3D points, a hierarchical triangulation scheme has been constructed that triangulates a large number of 3D points, where 3D points are reconstructed from feature point tracks with decreasing lengths until a certain length giving a acceptable triangulation accuracy. During this hierarchical triangulation process, the redundant triangulations are minimized by avoiding triangulation on feature point tracks containing feature points that are already used in previous triangulations.

Finally, a 3D merging scheme has been elaborated for combining partial reconstructions into a single coordinate system. This leads to overall scene reconstruction for the complete long sequence. The proposed scheme aligns the two point clouds from two successive subsequences with independent coordinate systems, such that the same 3D points in two different

coordinate systems are coinciding in the selected coordinate system. This is implemented by computing the unknown $3 \times 3$ projective transformation that maps the two point clouds from one coordinate system to the other, or vice versa.

We have applied the proposed algorithm to a short and a long video sequence to demonstrate performance of the system. The experimental results from the short sequence show that the proposed system is able to accurately reconstruct the 3D scene from multiple-view data. The accuracy evaluation on the short sequence reveals that the maximum error of angles and lengths is $3.6\%$ and $4.7\%$, respectively. The key result of the long sequence reconstruction is that, with our algorithm, it is now possible to automatically divide a long sequence into subsequences, and then merge the partial reconstructions into the same coordinate system.

The first contribution of this chapter is the introduction of the overall 3D reconstruction framework, providing an automatic reconstruction for long video sequences. The algorithm contributions in this chapter are refining additions to the introduced framework of which the major algorithms will be addressed in following two chapters, i.e., feature point matching and dividing long video sequences. Although the presented algorithms have resulted in an improvement in the robustness of feature point tracking and reconstruction, the evaluation has not been performed within the framework of the system. This means that we do not know how much the presented algorithms contribute to the overall results that we observed on the two test sequences. This detailed analysis will be useful for future work. Besides, the proposed framework can be further improved by adding more processing blocks. For example, by including the bundle adjustment of the reconstruction results into our framework, the merged point cloud can be refined to a high quality.

CHAPTER 4

# Texture-independent feature-point matching

*As pointed out in Chapter 3, tracking a large number of feature points along a long sequence of frames is critical for an automatic SaM on a long sequence using the factorization method. For 3D reconstruction on a long video sequence captured with a hand-held camcorder, some special factors need to be considered when matching and tracking the feature points. For example, the motion between two frames of a video sequence is generally small. This chapter introduces a feature point matching algorithm which is specially designed for matching and tracking a large number of points over successive frames where the image motion is limited. This step is essential for the proposed framework for 3D reconstruction on long video sequences, and the feature point algorithm that is presented in this chapter forms one of the major contributions of this thesis.*

## 4.1 Introduction

### 4.1.1 Positioning and summary of this work

Feature point tracking is essential and the first step for the proposed 3D reconstruction framework. Successive stages are splitting the long sequence into subsequences and performing factorization-based 3D reconstruction. This tracking should be able to obtain a large number of feature points along a long sequence of frames for the following three reasons. First, the factorization method requires that all feature points should occur in all frames. To be able to perform factorization-based reconstruction, sufficient feature points should be tracked along a sufficient number of frames, in order to obtain a robust reconstruction. Further discussion can be found in Chapter 5. Second, tracking a large number of feature points improves the

density of the reconstructed point cloud. Third, to merge partial reconstructions from individual subsequences, a large number of 3D points is key to robustly compute the unknown $3 \times 3$ transformation matrix, as discussed in Section 3.3.4. Summarizing, a good feature point tracking determines both the robustness of the reconstruction process and the quality of the reconstructed results.

Since in our framework the multiple-frame feature point tracks are obtained by linking the two-frame correspondences that are obtained by the feature point *matching* process, the performance of the feature point *matching* algorithm is key for the proposed 3D reconstruction system. As depicted in Fig. 4.1, this chapter concentrates on *matching* feature points between two successive frames.

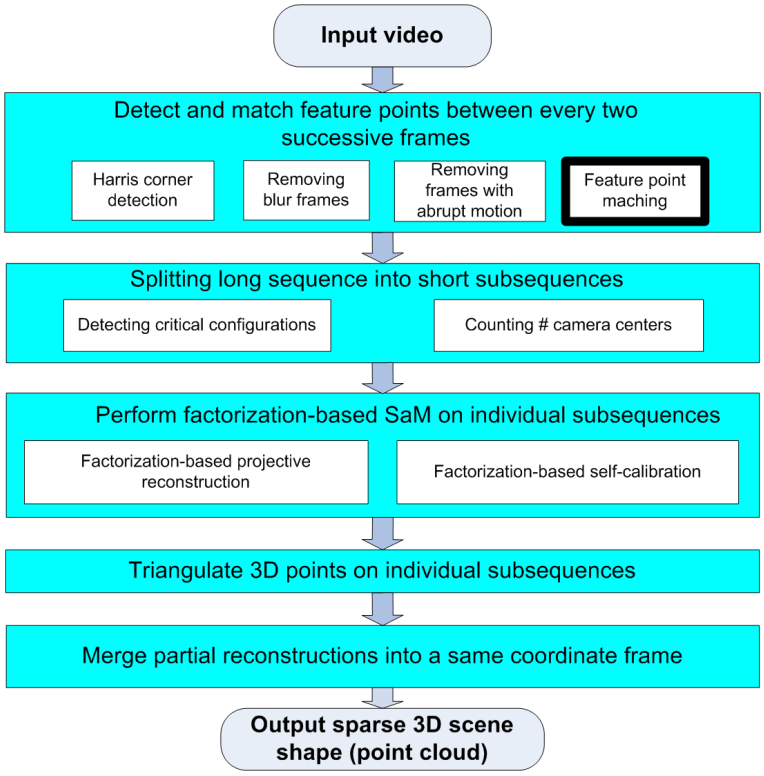



**Figure 4.1:** *Block diagram of the proposed 3D modeling system. This chapter presents our contribution on feature point matching.*

The proposed algorithm uses only a smoothness constraint, which states that neighboring feature points in images tend to move with similar directions and magnitudes. In the algorithm, the heuristic smoothness constraint is converted into a quantitative metric for determining the coherent matching vectors. With this metric, the true correspondences of all points in individual local neighborhoods are jointly determined by maximizing the local motion smoothness, using a RANSAC process. The employed smoothness assumption is valid for most images with limited image motion, regardless of the camera motion and scene struc-

ture. By building our algorithm solely on such a widely applicable smoothness constraint, the algorithm obtains two major advantages. First, it is robust to illumination changes, as the employed smoothness constraint does not rely on any texture information and remains valid for most images with limited motion. Second, it yields a high performance when tracking feature points in sequences. The algorithm can smoothly handle the drift of the feature points over time, because the drift can hardly lead to a violation of the smoothness constraint. This aspect contributes to the large number of feature points matched and tracked by the proposed algorithm. Our extensive experimental results show that the proposed algorithm is able to track at least twice as many points as the state-of-art algorithms, with a comparable or higher accuracy.

### 4.1.2 Motivation of algorithm design

The motivation of our feature matching algorithm is based on two aspects: the use of video sequences and the underlying constraints for matching feature points. Let us address both aspects briefly here.

In general, the use of video sequences implies that the image motion between successive frames for most image parts is limited. This means that the majority of existing feature point matching algorithm which are designed for matching feature points between two images taken at clearly different camera positions are generally not appropriate for our case. The limited motion aspect is a key feature of a video sequence, which should be exploited to improve the matching performance.

**Discussion:** Many algorithms have been proposed for matching points between images with large baselines[1] (for applications such as object recognition, image retrieval and texture classification). In contrast, few techniques have been proposed for tracking correspondences over successive images where the camera baseline and image motion are limited[2], as in 3D scene reconstruction from video. Although many wide-baseline algorithms can well match the points between images with scale or view distortions, the number of tracked points may be limited. The reliance of these algorithms on the intensity similarities limits their capability to match more drifted feature points, as will be discussed in Section 4.5.3. The existing narrow-baseline algorithms, which are typically based on the local image correlation for matching, are generally less robust to image motion and light changes. **end**

The second aspect of the motivation deals with the underlying constraint for feature point matching. Geometry similarity and image similarity are two fundamental constraints that are used in most matching algorithms. Image similarity typically assumes that the intensities of two images are constant. However, a constant intensity is difficult to maintain in practice. Even if we assume that the camera hardware is identical, the amount of light entering the two cameras can be different for slightly different points of view, causing dynamic adjustments

[1] Baseline is defined as the distance between two cameras.

[2] Limited image motion usually corresponds to limited camera baselines. But this is not always the case, since small image motion may also be associated with large camera baseline. Since the only condition for the proposed algorithm to work is that the image motion should be limited, the proposed algorithm actually also works for wide camera-baseline scenarios as long as the image motion is limited. In this thesis, we do not distinguish between limited motion and limited baseline.

of intrinsic camera parameters such as aperture, exposure and gain [58]. The geometry similarity is more fundamental and stable than the image similarity, as intensity is more prone to changes [28]. To obtain a robust matching, it is favorable to establish feature correspondences based on geometric constraints only, though it is commonly believed that this concept is computationally intractable [88].

Therefore, this chapter proposes a feature point matching algorithm that uses only a smoothness constraint, which states that neighboring points in an image usually move with similar magnitudes and directions. In the proposed algorithm, the best matching vectors for all points within individual local neighborhoods are jointly determined such that the local motion smoothness is maximized. The smoothness constraint does not rely on any texture information and is valid for most sequences with limited image motion, whatever the origin of those sequences. This makes the proposed algorithm robust and capable to track a large number of points, as will be discussed in Section 4.5.3.

### 4.1.3 Related work

This section gives a brief review of existing feature point matching algorithms. Since there is little literature about algorithms designed for limited-motion sequences, most algorithms reviewed here are for general point matching.

Feature point matching algorithms can be classified into multiple categories according to the applied matching criteria. Example algorithms are, the image-correlation approach [78], the region-descriptor approach [48, 4, 69, 53, 5], the geometric-constraint approach [71, 6] and the combined approach [28, 35, 71, 94, 11, 35]. Feature point matching algorithms can also be classified according to whether the points are matched locally (points are matched without considering the neighboring points) or globally, i.e., the local approach [78] and the global-optimization approach [71, 94, 11, 35, 49]. In the following, we summarize a few representative algorithms from these categories.

**Correlation-based algorithms.** In [28], a matching algorithm using both the image correlation and geometric constraints is proposed. Feature correspondences are first detected using window-based image correlation. The outliers are thereafter rejected by a few subsequent heuristic tests involving geometry, rigidity, and disparity. As discussed above, algorithms relying on image similarity do not work well for images with large light changes. In [78], the Kanade-Lucas-Tomasi (KLT) feature tracker is proposed for tracking feature points in image sequences. In KLT, feature points are first detected in image areas containing sufficient texture variation, and then tracked by determining the displacement vectors according to image gradients. The disadvantage of KLT is that it works only for images with very small motion and light changes. Also, the spatial localization accuracy of the tracked feature points is generally lower than the matching-based algorithms. In [6], a feature point matching algorithm that uses only geometric constraints is presented. The algorithm assumes that the scene surface is piecewise planar, and thus it uses the homography and epipolar geometry to iteratively fit the curved scenes for removing the matching ambiguity. One drawback of the algorithm is that the piecewise assumption is not always valid for complex scenes. In [11], a global optimization algorithm is proposed to match points by preserving the local geometric consistency, using the homography-induced pairwise constraint. As pointed out in the article, one major drawback is its high computational complexity to compute the pairwise homography

terms.

**Descriptor-based algorithms.** Photometric region descriptors have recently been widely used for feature point matching. In this approach, local image regions are described using image measurements such as the histogram of the pixel intensity, distribution of the intensity gradients [48], image derivatives [4, 69], etc. Below, we summarize some well-known descriptors from this category. A review of state-of-the-art region descriptors can be found in [53, 90]. Lowe [48] has proposed a Scale-Invariant Feature Transform (SIFT) algorithm for feature point matching, which combines a scale-invariant region detector and a gradient-distribution-based descriptor. The descriptor is represented by a 128-dimensional vector, capturing the distribution of the gradient orientations in 16 location grids (sub-sampled into 8 orientations and weighted by gradient magnitudes). Features are matched if two descriptors show a small difference. The Gradient Location and Orientation Histogram (GLOH) algorithm proposed by Mikolajczyk and Schmid [53] extends the SIFT to consider more regions for computing the histogram, and was shown to outperform the SIFT. Recently, Bay *et al.* have proposed a new rotation- and scale-invariant interest point detector and descriptor, called SURF (Speeded Up Robust Features) [5]. It is based on sums of 2D-Haar wavelet responses and makes an efficient use of integral images. The algorithm is claimed to have comparable or better performance, while obtaining a much faster execution than previously proposed schemes. For tracking correspondences over successive images with limited image motion, one disadvantage of many descriptor-based algorithms is that the reliance of these algorithms on the appearance similarity decreases their capability to match feature points that drift over successive images. This drift can be caused by either the viewpoint change or illumination changes. This decreases the number of points that can be matched and tracked over images. Furthermore, the high computational complexity of most algorithms due to the large number of elements in these descriptors is also a concern for correspondence tracking in limited-motion sequences, such as originated from video.

**Tracking.** The algorithms described above (except KLT) are designed for matching feature points between two images. To track feature points over images using these algorithms, correspondences between two images are first established, and then linked to obtain multiple-frame correspondences. In the following text, we describe several tracking algorithms that are designed for directly tracking feature points over images. In [30], Jenkin has proposed a tracking algorithm based on a general smoothness constraint, that assumes that the locations, velocities and moving directions of feature points remain relatively unchanged from one frame to the next. In [87], Yao *et al.* have proposed an algorithm that tracks feature points by estimating the motion vectors between two images based on a localized appearance constraint. The motion information of the previous frame is used when estimating the motion vectors for the current frame. The algorithm achieves the benefits of both the two-frame-based tracking approach and the long-sequence-based approach[3]. In [93], Zheng *et al.* have proposed a feature point tracking algorithm containing two major steps: (1) image registration to compensate the camera motion, and (2) feature point matching based on the intensity correlation to determine the best matching point within the neighborhood of anticipated lo-

[3]The two-frame-based approach attempts to track feature points by first matching them independently between every two successive images, and then linking the correspondences to obtain multiple-frame tracks. The long-sequence-based approach attempts to track feature points by employing the temporal motion information across multiple frames in a sequence, with the assumption that the motion of an object does not change abruptly [87].

cations. In [85], a motion tracking algorithm is proposed to track points in multiple images using so-called multi-objective optimization, which relies heavily on the motion coherence constraint to track feature points, in order to avoid the situation that the image appearance may not be reliable as compared to the geometry similarity. In the algorithm, feature points are tracked in a sequential heuristic search algorithm, where the search tree is adequately pruned based on second-order motion characteristics optimized over several frames.

The aforementioned algorithms either use multiple constraints, or require that image motion should be smooth across multiple frames. Compared with these algorithms, the proposed algorithm only requires that the motion should be smooth between *two* consecutive frames. It does not limit image motion variation across more than two frames, and it does not restrict image intensity variations over time. By constraining the design of our algorithm to such a broadly applicable smoothness assumption, we expect to obtain a robust and good matching performance.

### 4.1.4 Proposed approach

Unlike most existing algorithms where the smoothness constraint is either used as a pairwise smoothing term, or as a complementary matching constraint for feature point matching, the proposed algorithm uses only a smoothness constraint for feature point matching. Therefore, the proposed algorithm is referred to as Texture-Independent Feature Matching (TIFM), thereby emphasizing that no texture information is required for the matching process. The texture independence is explicitly pursued in the design of TIFM to increase its robustness to light changes and non-Lambertian surfaces.

We consider the used smoothness constraint as a geometric constraint, since it is related with the geometry rigidity. For example, a group of points on the surface of a rigid object typically move with similar speeds, which leads to a smooth motion in images. By building our algorithm on the smoothness constraint only, TIFM gains two major advantages: (1) the robustness to light changes, and (2) the good capability to handle the drift of feature points, and thus to match and track a large number of feature points.

One important condition for the proposed algorithm is that the image motion, especially the non-translational motion (e.g. rotation and scaling), between two images should be limited. For example, for two images of $1024 \times 768$ pixels, it is preferred that the pixel displacement between two images is smaller than 90 pixels (a larger pixel displacement is allowed if more information such as the fundamental matrix is available). We refer to such sequences as *limited-motion sequences*. It should be noted that for most image sequences such as those tested in Section 4.5, which are captured for 3D scene reconstruction, the translation, scaling and rotation between two images are not very large, and thus they can be treated as limited-motion sequences, where TIFM works well.

The remainder of this chapter is organized as follows. In Section 4.2, the notations that are used throughout this chapter are introduced, and the problem that this chapter addresses is defined in more detail. Section 4.3 presents the proposed algorithm. First, the heuristic smoothness constraint is converted into a quantitative smoothness metric. After that, the detailed steps of the proposed algorithm are presented. Finally, the rationale of the algorithm is further explained. Section 4.3.8 discusses a number of important algorithm parameters, which have a large impact on the performance of the proposed algorithm. Section 4.4 evaluates the

correctness of TIFM using synthetic data. Section 4.5 presents the experimental results on real data. The performance of TIFM is compared against three state-of-the-art algorithms for feature point matching and tracking. Finally, Section 4.6 concludes this chapter.

## 4.2 Notations and problem formulation

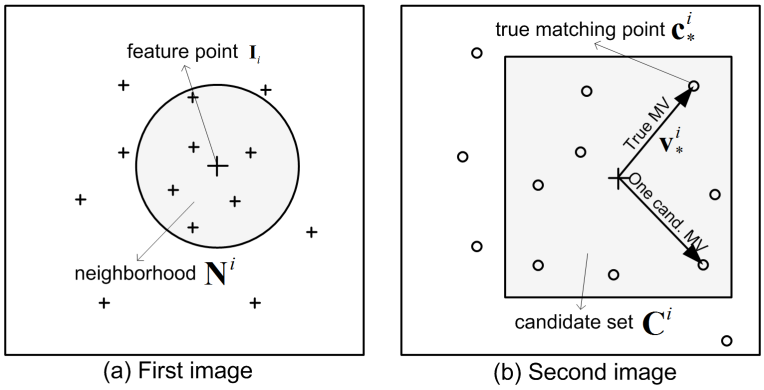



**Figure 4.2:** *Each feature point $\mathbf{I}_i$ is associated with: (1) a set of neighboring points $\mathbf{N}^i$ in the first image, and (2) a set of candidate matching points $\mathbf{C}^i$ in the second image. Candidate point $\mathbf{c}_*^i$ is the true matching point to $\mathbf{I}_i$ and $\mathbf{v}_*^i$ is the true matching vector. Points in the first image are represented by '+' and in the second by 'o'.*

This section defines the notations which are used in the remaining sections to describe the problem statement and solution.

Let $\mathbf{I} = \{\mathbf{I}_1, \mathbf{I}_2, \cdots, \mathbf{I}_M\}$ and $\mathbf{J} = \{\mathbf{J}_1, \mathbf{J}_2, \cdots, \mathbf{J}_N\}$ be the two sets of feature points in two related images, containing $M$ and $N$ feature points, respectively. Let $(x_i, y_i)^T$ be the coordinates of feature point $\mathbf{I}_i$ (to simplify later equations we have chosen the vector notation for coordinate representation). For each feature point $\mathbf{I}_i \in \mathbf{I}$, we define the following notations.

$\mathbf{N}^i = \{\mathbf{n}_1^i, \mathbf{n}_2^i, \cdots, \mathbf{n}_n^i\}$, set of $n$ neighboring points in a circular neighborhood around $\mathbf{I}_i$, as shown in Fig. 4.2(a).

$\mathbf{C}^i = \{\mathbf{c}_1^i, \mathbf{c}_2^i, \cdots, \mathbf{c}_m^i\}$, set of $m$ candidate matching points in a co-located rectangle[4] in the second image, as shown in Fig. 4.2(b). The true matching point is denoted as $\mathbf{c}_*^i$.

[4]The shape of the candidate set can be a circle or any other shape depending on what information is available. For example, if the fundamental matrix is already available, the shape of the set will be a line. For feature point matching, a smallest possible candidate set is desired in order to reduce the matching ambiguity. The rectangle is used in this thesis for the ease of illustration.

$\mathbf{V}^i = \{\mathbf{v}_1^i, \mathbf{v}_2^i, \cdots, \mathbf{v}_m^i\}$, set of $m$ candidate Matching Vectors (MV) arising from $\mathbf{C}^i$. Each candidate matching point gives one candidate MV. The true MV is denoted as $\mathbf{v}_*^i$.

For each point $\mathbf{I}_i \in \mathbf{I}$, we want to find its matching point $\mathbf{c}_*^i \in \mathbf{C}^i$. This is equivalent to finding the true MV $\mathbf{v}_*^i \in \mathbf{V}^i$. In the above notations, index $i$ indicates that the defined sets are associated with the feature point $\mathbf{I}_i$. In the following discussions, we omit the index $i$ for simplicity when there is no ambiguity.

## 4.3 Proposed algorithm

In this section, the smoothness constraint is translated into a quantitative metric for finding coherent matching vectors. Afterwards, the steps of the proposed matching algorithm are described and the rationale of the algorithm is further explained.

### 4.3.1 Coherence metric

Suggested by the motion smoothness theory [88], we assume that the true MVs within a small neighborhood have similar directions and magnitudes, which is referred to as the *local-translational-motion* (LTM) assumption/constraint in this thesis. MVs that satisfy this constraint are called Coherent Vectors (CVs).

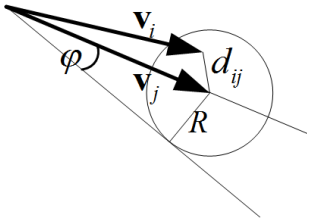



**Figure 4.3:** *Two coherent vectors within a bounding coherent circle with radius R.*

Given two CVs $\mathbf{v}_i$ and $\mathbf{v}_j$, describing two point-to-point matches between two images, we require that the Euclidian distance $d_{ij}$ between them is smaller than a threshold, as illustrated in Fig. 4.3. This can be achieved by imposing the following coherence criterion:

$$d_{ij} = ||\mathbf{v}_i - \mathbf{v}_j|| < ||\mathbf{v}_j|| \times sin(\varphi) = R, \tag{4.1}$$

where $\varphi$ is the allowed angle of deviation between two coherent vectors, and $R$ is the threshold, which is computed based on the magnitude of the reference vector $\mathbf{v}_j$ and $\varphi$. In our implementation, $R$ is set to a minimum value of 2 pixels. The circle with a radius $R$ in Fig. 4.3 is referred to as *coherent circle*. Geometrically, $\mathbf{v}_i$ is the *coherent* with $\mathbf{v}_j$ when it points to a pixel within the bounding coherent circle associated with $\mathbf{v}_j$. Deviation $\varphi$ specifies how close two MVs should be in order to satisfy the coherence criterion. The determination of the value of $\varphi$ is discussed in Section 4.3.8B.

### 4.3.2 Algorithm overview

The proposed algorithm determines the best matching vector for a feature point by maximizing the motion smoothness within the local neighborhood around the feature point. In order to maximize the motion smoothness, we need to determine the coherent vectors for any candidate matching vector. After that, the motion smoothness associated with the candidate matching vector can be computed by counting the number of the coherent vectors. The best matching vector is then determined as the candidate matching vector with the highest smoothness.

Fig. 4.4 depicts the flowchart of the proposed algorithm. In the following text, each of the major steps in the depicted flowchart will be introduced.

### 4.3.3 Determining coherent vectors

With the coherence criteria specified by Eq. (4.1), we can determine the CV (coherent with matching vector $\mathbf{v} \in \mathbf{V}$) for any neighboring point $\mathbf{n}_k \in \mathbf{N}$. The key steps for finding the coherent vectors are as follows.

1. Given a MV with $\mathbf{v} \in \mathbf{V}$, find the closest matching vector $\mathbf{v}_{\triangle}^k(\mathbf{v}) \in \mathbf{V}^k$ for every $\mathbf{n}_k \in \mathbf{N}$ according to
$$\mathbf{v}_{\triangle}^k(\mathbf{v}) = \arg \min_{\mathbf{v}_i^k \in \mathbf{V}^k} ||\mathbf{v} - \mathbf{v}_i^k||. \tag{4.2}$$

2. If $\mathbf{v}_{\triangle}^k(\mathbf{v})$ satisfies the coherence metric Eq. (4.1), it is considered coherent with $\mathbf{v}$. Then we define the binary function $f_{\mathbf{n}_k}(\mathbf{v})$ indicating whether we can find a CV (coherent with $\mathbf{v}$) for $\mathbf{n}_k$, as follows:
$$f_{\mathbf{n}_k}(\mathbf{v}) = \begin{cases} 1, & \text{when } ||\mathbf{v} - \mathbf{v}_{\triangle}^k(\mathbf{v})|| \leq R; \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

It should be noted in the first step that finding the closest MV is performed for each neighboring point, and that the considered set of candidate matching vectors for each point is different, as defined in Section 4.2. In the second step, the use of binary function not only indicates which set has a CV but also the number of points having a CV can be easily counted to evaluate the local motion smoothness.

With the above steps, we can determine the CVs for all points in a neighborhood for any candidate MV with $\mathbf{v} \in \mathbf{V}$. Based on the obtained CVs and binary function, the smoothness of the local motion field associated with any $\mathbf{v} \in \mathbf{V}$ can be computed, as will be presented in the next subsection.

### 4.3.4 Matching points by maximizing local motion smoothness

Using steps described in Section 4.3.3, we can count the number of CVs in neighborhood $\mathbf{N}$ for any candidate MV with $\mathbf{v} \in \mathbf{V}$. Knowing the number of CVs, the smoothness $S(\mathbf{v})$ of the local motion field associated with $\mathbf{v}$, can be computed by
$$S(\mathbf{v}) = \frac{\sum_{\mathbf{n}_k \in \mathbf{N}} f_{\mathbf{n}_k}(\mathbf{v})}{n} \times 100\%. \tag{4.4}$$

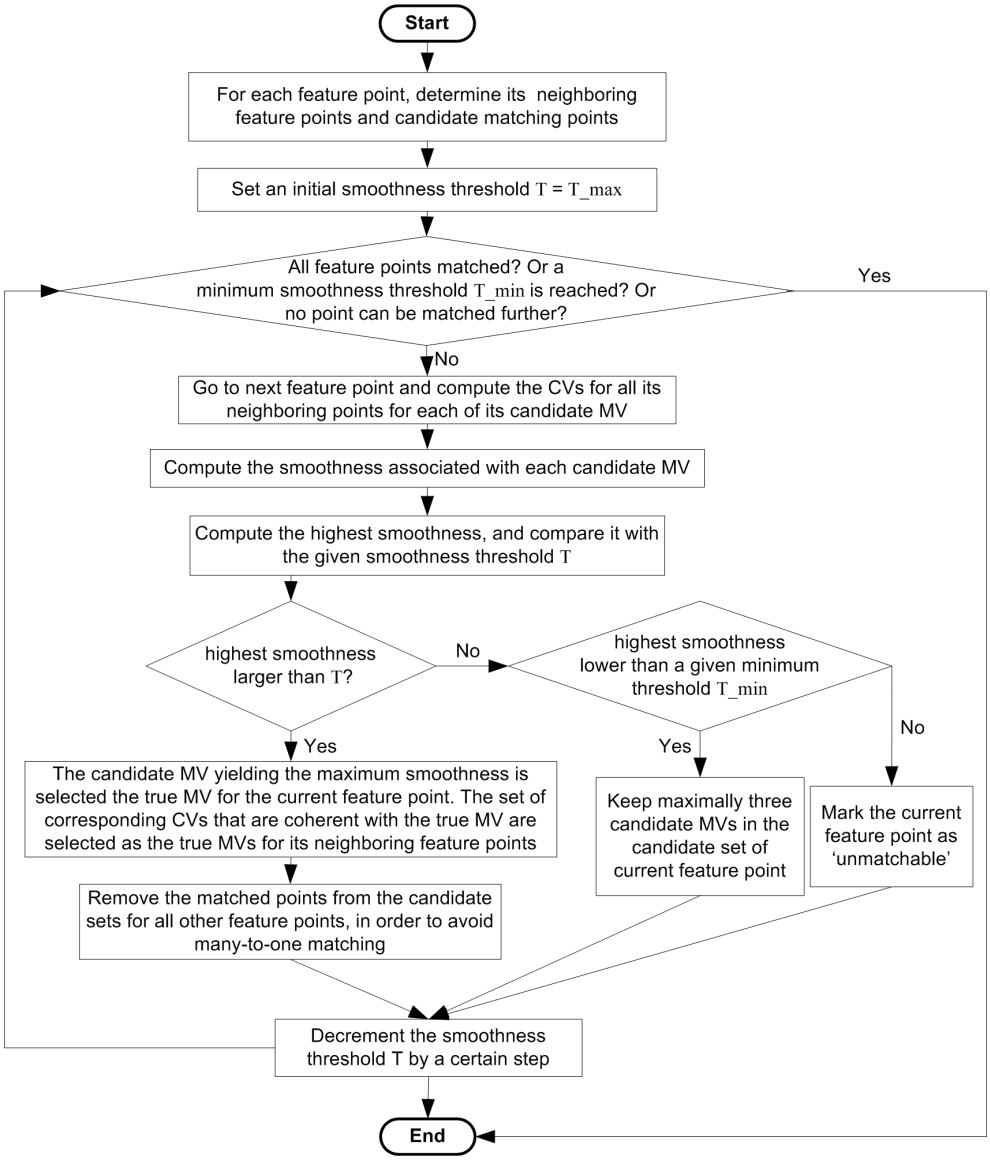



**Figure 4.4:** *Flowchart of the proposed algorithm.*

In the equation, $n$ is the number of points in the neighborhood, and $f_{\mathbf{n}_k}(\mathbf{v})$ is a binary function defined in Eq. (4.2). Given the motion smoothness for all candidate MVs, the maximum smoothness $S_{max}$ is then found by

$$S_{max} = \max_{\mathbf{v} \in \mathbf{V}} S(\mathbf{v}). \tag{4.5}$$

The LTM constraint suggests that true MVs within a neighborhood should have similar

directions and magnitudes. This implies that $S(\mathbf{v}_*)$ should be as large as possible in order to obtain a smooth motion field. Therefore, the candidate MV with the largest smoothness will most likely correspond to the true MV. This assumption will be discussed and validated in Sections 4.3.7 and 4.4. At this point, we assume that it is valid. Therefore, the true MV denoted by $\mathbf{v}_*$ can be determined using:

$$\mathbf{v}_* = \begin{cases} \arg\max_{\mathbf{v}\in\mathbf{V}} S(\mathbf{v}), & \text{when } S_{max} \geq T_s; \\ \varnothing \text{ (empty)}, & \text{otherwise.} \end{cases} \tag{4.6}$$

In the above formula, $T_s$ is a given threshold for the local smoothness. If $S_{max} < T_s$, we consider the smoothness to be too low and unreliable, so that correspondences should not be assigned.

Knowing $\mathbf{v}_*$ for $\mathbf{I}_i$, the best matching vectors $\mathbf{v}_*^k$ for all neighboring points $\mathbf{n}_k \in \mathbf{N}$ are selected as the set of closest matching vectors which are coherent with $\mathbf{v}_*$. More formally, this step is as follows: the $\mathbf{v}_*^k$ is computed as the closest vector $\mathbf{v}_\Delta^k(\mathbf{v}_*)$, if $\mathbf{v}_\Delta^k(\mathbf{v}_*)$ is coherent with $\mathbf{v}_*$, i.e., if $f_{\mathbf{n}_k}(\mathbf{v}_*) = 1$. If $f_{\mathbf{n}_k}(\mathbf{v}_*) = 0$, no correspondence will be assigned for $\mathbf{n}_k$.

### 4.3.5 Steps to match all feature points within a neighborhood

Let us now summarize the steps to compute the correspondences for all feature points within neighborhood $\mathbf{N}$:

1. For every $\mathbf{v} \in \mathbf{V}$, determine the CV for every point $\mathbf{n}_k \in \mathbf{N}$ using the steps described in Section 4.3.3. After that, compute the smoothness $S(\mathbf{v})$ using Eq. (4.4).

2. Compute the maximum smoothness $S_{max}$ using Eq. (4.5). The true matching vector $\mathbf{v}_*$ of $\mathbf{I}_i$ is found if $S_{max} \geq T_s$, according to Eq. (4.6). The MVs for all neighboring points are computed as the set of CVs that are coherent with $\mathbf{v}_*$.

The above steps select the candidate with the highest number of CVs as the true MV. By doing this, the local motion smoothness within the neighborhood is maximized. As illustrated in Fig. 4.5, $n$ feature points within a local neighborhood are jointly matched using the proposed algorithm. As indicated earlier, we assume that true MVs within a local neighborhood are coherent with each other. By the proposed algorithm, the true MVs denoted in boldface are detected because they give the maximum smoothness, measured by the number of the CVs.

**Relation with RANSAC.** It should be noted that the proposed algorithm to determine the best MV based on the number of supporting CVs is similar to the RANSAC voting process, where the best model parameters are determined based on their support from other samples. However, compared with RANSAC, one important difference is that, in our case, there is no guarantee that the set of candidate MVs will contain the true MV. In contrast, in RANSAC, it is guaranteed that the assumed model will be applicable to the given data, and the correctness of the model parameters can be guaranteed by increasing the number of iterations. Thus in our case, the optimality of obtained MVs cannot be guaranteed by increasing the number of iterations. For this reason, we propose to use a comparison of the motion smoothness with the

smoothness threshold $T_s$ as a performance criterion to measure the correctness of the obtained MVs, as shown in Step *2*. As will be explained in Section 4.3.8 C, a high confidence on the correctness of the MVs is obtained if the motion smoothness is above a given threshold.

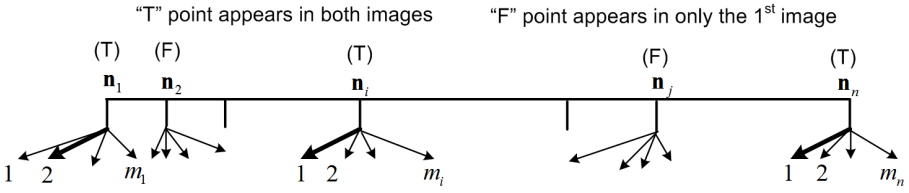



**Figure 4.5:** *Matching process to match $n$ feature points within a local neighborhood. Each of the $n$ points in the neighborhood of point $\mathbf{I}_i = \mathbf{n}_i$ has a varying number of candidate MVs (indicated by vectors), with the true MVs marked in bold. Only points labeled as "T", called repeated points, that appear in both images have true MVs.*

### 4.3.6 Algorithm steps and input parameters for matching all feature points in one image

In the above section, we have presented the steps to match feature points within a local neighborhood. This section extends the algorithm to a complete image. The overall steps to match all feature points in one image can be summarized as follows:

*1* Compute $\mathbf{N}^i$ and $\mathbf{V}^i$ for every $\mathbf{I}_i \in \mathbf{I}$. The computation of $\mathbf{N}^i$ is described in Section 4.3.8 D. In order to determine $\mathbf{V}^i$, we need to supply the maximum pixel displacement between two images, which will be discussed in Section 4.3.8 E.

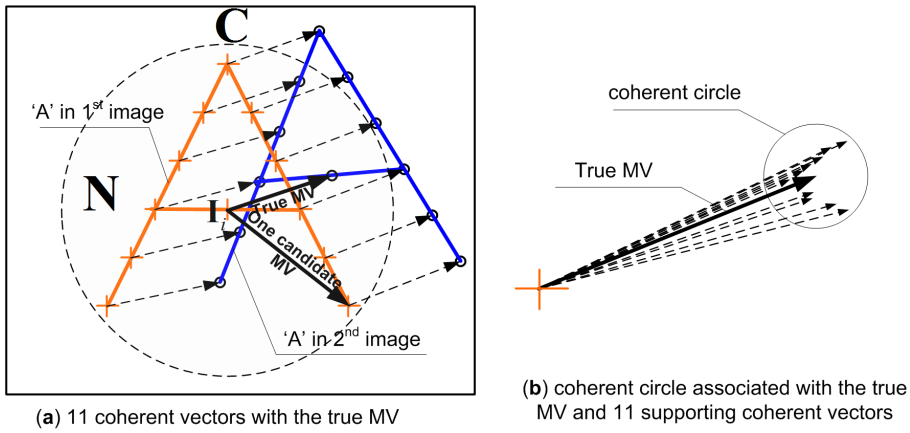
*2* For every $\mathbf{N}^i$, match the points using the steps described in Section 4.3.5. To ensure a good matching, we need to specify the smoothness threshold $T_s$, and to determine the CVs, we need to specify the maximum allowed angle of deviation $\varphi$.

*3* Decrease $T_s$ by a fixed value and repeat Step *2* until a minimum $T_s$ is reached. This means that feature points in local neighborhoods with a higher smoothness are matched prior to others. By doing this, the previously matched points with a high confidence help to constrain the matching process of other feature points.

In our implementation, $\varphi$ is fixed at $2^o$. The determination of $\varphi$ is given in Section 4.3.8 B. The maximum $T_s$ is fixed at 0.8 and the minimum $T_s$ is set to 0.4. After each iteration, $T_s$ is decreased by 0.1. More discussion on parameter setting is given in Section 4.3.8 C. The only parameter that needs to be manually set by the user is the pixel displacement between two successive images.as will be further discussed in Section 4.3.8 E. Note that due to the

RANSAC-like process, smoothing over object boundaries is avoided, since points will not be matched if the number of CVs is not sufficient. For example, this typically occurs in neighborhoods with crossing object boundaries, where the number of CVs may not be sufficient due to differences in motion.

One example of the matching process of the proposed algorithm is depicted in Fig. 4.6. As can be observed, the true MV generates 11 CVs while the other candidate MV has only a few. Therefore, the true MV gives the highest the smoothness in the local neighborhood, and thus the MV with the highest number of CVs is selected as the true MV for point $\mathbf{I}_i$.




(**a**) 11 coherent vectors with the true MV

(**b**) coherent circle associated with the true MV and 11 supporting coherent vectors

**Figure 4.6:** *Letter 'A' is translated and rotated in two images. A maximum smoothness of* $100\%$ *(11 coherent vectors in dotted lines) is found with the true MV. The smoothness is low along any other candidate MV.*

### 4.3.7 Rationale of the algorithm

In the beginning of this chapter, our algorithm has been motivated by the robustness of the smoothness constraint, and in the previous section, we have demonstrated that the smoothness can be maximized using a RANSAC-like process. In this section, we will explain that such a smoothness maximization will lead to a correct determination of the MVs, using a probabilistic framework.

**Intuitive explanation.** According to the Local Translational Motion (LTM) assumption, the $(n \times \alpha)$ number of true MVs in Fig. 4.5 are coherent and compose a *smooth* motion field with a smoothness of $\alpha$, where $\alpha$ is the repetition ratio of the feature points within the neighborhood. Repetition ratio $\alpha$ is defined as the percentage of the feature points in the first image that appears also in the second image. Due to a randomness of the texture, feature points appear randomly along any other candidate MV. The probability to find another set of CVs giving a smoothness higher than $\alpha$ is small. Thus, the true MV resulting from the repetition of a feature point will be elected as giving the maximum number of CVs in most cases. Once

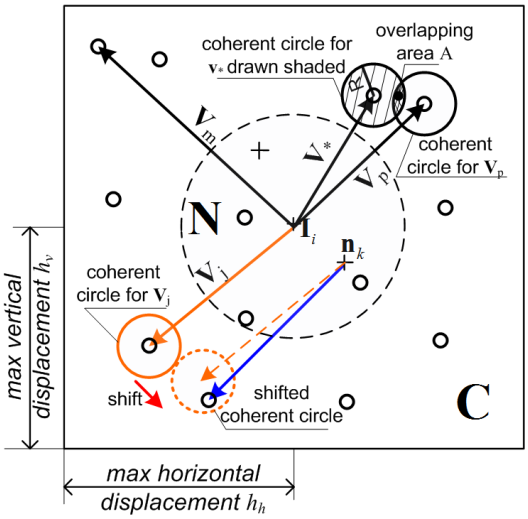



**Figure 4.7:** *Finding coherent vectors for neighboring feature points. Each candidate MV of point* $\mathbf{I}_i$ *is associated with a coherent circle. The MV (the vector at the bottom) for point* $\mathbf{n}_k$*, which is coherent with* $\mathbf{v}_j$*, should point to a pixel within the shifted coherent circle. In other words, there has to be at least one feature point in the shifted coherent circle in order to find a coherent vector for* $\mathbf{n}_k$*.*

the maximum smoothness is detected, the true MV is found, as illustrated by Fig. 4.6.

**Probabilistic explanation.** We further motivate the correctness of the algorithm using probability theory. As shown by Fig. 4.7, the set $\mathbf{V}$ contains $m$ candidate MVs ($\mathbf{v}_*$,$\mathbf{v}_p$,$\mathbf{v}_j$,$\cdots$,$\mathbf{v}_m$) where each candidate is associated with a bounding *coherent circle*, of which the radius is computed by Eq. (4.1). The coherent circle associated with the true matching vector $\mathbf{v}_*$ is shaded in Fig. 4.7. Since any point in a shifted coherent circle for $\mathbf{n}_k$ provides a vector coherent with $\mathbf{v}_j$, *the probability of finding a coherent vector (coherent with* $\mathbf{v}_j$*) for any point* $\mathbf{n}_k \in \mathbf{N}$ *is equal to the probability of finding a point within the shifted coherent circle*, as illustrated in Fig. 4.7. Fig. 4.7 depicts three types of coherent circles, i.e., (1) the shaded circle for $\mathbf{v}_*$, (2) the circle for $\mathbf{v}_p$ that overlaps with the shaded circle, and (3) the circle for $\mathbf{v}_j$ that does not overlap with the shaded circle. In the following discussion, each of these three types of circles is investigated, and then the confidence of the detected correspondences is analytically computed.

Due to the repetition of the feature points, the probability $p'$ to find a point in the shaded circle in Fig. 4.7 can be approximated by the repetition ratio $\alpha$ of the feature points in the neighborhood, under the condition that the radius $R$ of the coherent circle is larger than the maximum deviation $d_{max}$ between two true MVs, as illustrated in Fig. 4.8. If $R < d_{max}$, probability $p'$ will be smaller than $\alpha$, since some repeated points cannot be detected. Thus,

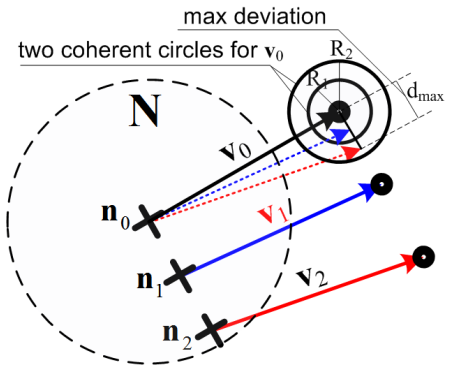



**Figure 4.8:** *Deviation between coherent vectors. The MVs of 3 points within a neighborhood deviate from each other. The deviation is usually larger between two points with a larger pixel distance. For example, the deviation between* $\mathbf{v}_2$ *and* $\mathbf{v}_0$ *is larger than the deviation between* $\mathbf{v}_1$ *and* $\mathbf{v}_0$*. The radius* $R$ *of the coherent circle should be large enough to detect both* $\mathbf{v}_1$ *and* $\mathbf{v}_2$*. In the figure,* $R_1$ *is too small, since it can only detect* $\mathbf{v}_1$*.*

probability $p'$ can be estimated using the following relation:

$$p' : \begin{cases} p' = \alpha, & \text{when } R \geq d_{max}, \\ p' < \alpha, & \text{when } R < d_{max}. \end{cases} \tag{4.7}$$

Assuming a uniform distribution of the feature points, the probability $p''$ to find a feature point within a coherent circle that does not overlap with the shaded circle can be estimated by:

$$p'' = \frac{m}{4h_h h_v} \times \pi R^2 = \rho \pi R^2. \tag{4.8}$$

Parameters $h_h$ an $h_v$ are the maximum horizontal and vertical pixel displacements, and $m$ is the number of candidate MVs for feature point $I_i$, as shown in Fig. 4.7. Parameter $\rho$ represents the density of the feature points. Eq. (4.8) suggests that $p''$ is proportional to the density of the feature points and to the surface of the coherent circle. It should be noted that $p''$ is bounded to unity if Eq. (4.8) gives a value larger than unity.

Suppose the coherent circle for matching vector $\mathbf{v}_j$ overlaps with the shaded circle by a fraction $\lambda_j$ (e.g. the circle for $\mathbf{v}_p$ in Fig. 4.7), then *the probability* $p_j$ *to find a coherent vector (coherent with* $\mathbf{v}_j$*) for any point* can be computed as:

$$p_j = \lambda_j p' + (1 - \lambda_j) p'', \ \forall j \in [1, \cdots, m], \tag{4.9}$$

where $\lambda_j \in [0, 1]$ is the fraction of the overlapping area covering the circular surface. When $\lambda_j = 0$, this means there is no overlap, and $\lambda_j = 1$ stands for a complete overlap, which corresponds to the shaded circle associated with $\mathbf{v}_*$ in Fig. 4.7.

With Eq. (4.9), the probability to find $l$ ($l \leq n$) CVs (coherent with $\mathbf{v}_j$) can be computed as $P_j = C_l^n \times (p_j)^l$, where $C_l^n$ is the number of $l$-combinations out of $n$ points. To compute

the *confidence* $F$ that these $l$ CVs are true MVs, we define *Event A* as: '$l$ CVs can be found in the neighborhood due to any reason', and *Event B* as: '$l$ CVs are found due to the repetition of the feature points'. The confidence $F$ in this case can be computed as the conditional probability of *Event B* given *Event A*, which is

$$F = P(B|A) = \frac{P_*}{\Sigma_{j=1}^{m} P_j} = \frac{(p')^l}{\Sigma_{j=1}^{m} [\lambda_j p' + (1-\lambda_j)p'']^l}. \tag{4.10}$$

Assuming no coherent circle overlaps with the shaded circle ($\lambda_j = 0, \forall j \neq *$), we obtain:

$$F = \frac{(p')^l}{(p')^l + (m-1)(p'')^l} = \frac{1}{1 + (m-1)(p''/p')^l}. \tag{4.11}$$

Assuming $R \geq d_{max}$ in Eq. (4.7), we obtain $p' = \alpha$, and Eq. (4.11) can be rewritten into:

$$F = \frac{1}{1 + (m-1)(p''/\alpha)^l} \tag{4.12}$$

From Eq. (4.12), we observe that for a reasonably high repetition ratio $\alpha$ and a sufficient number $l$ of CVs, we can obtain true correspondences with a high confidence. For example, assuming $m = 100$, $n = 15$, $l = 6$, $h_h = h_v = 70$, $||\mathbf{v}_*|| = 70$, $\varphi = 2^o$ and $\alpha = 0.5$, we obtain $R = ||\mathbf{v}_*|| \times sin(\varphi) = 2.44$ and $p'' = 0.095$. The computed value of $F$ in this case is $98.2\%$. Note that the parameter values in this example are already close to the worst-case scenario of tracking correspondences in limited-motion sequences, i.e., very dense feature points and very large motion.

The derivation of Eq. (4.12) is based on a number of assumptions which are

a1. All repeated points appear within the corresponding shifted coherent circles, i.e., $R \geq d_{max}$ in Eq. (4.7);

a2. Feature points are distributed uniformly;

a3. No coherent circle overlaps with the shaded coherent circle.

These assumptions are not exactly valid in practice, and thus Eq. (4.12) cannot give an accurate confidence computation. However, it does provide a good explanation of the rationale of the proposed algorithm, since these three assumptions remain valid for most local circular neighborhoods.

### 4.3.8 Discussion on algorithm parameters

From Eqs. (4.11) and (4.12), we discern four important parameters: $m$, $l$, $p'$ and $p''$. The parameter $m$ is the number of candidates. The more candidate matching vectors, the higher the matching ambiguity, and thus the lower the obtained confidence. Probability $p'$ can be considered as the smoothness contributing factor arising from repeated points. Probability $p''$ is representing another form of smoothness arising from texture randomness. It should be noted that both forms lead to a certain smoothness computed using Eq. (4.4). The matching process tends to favor one of those forms over the other, which gives a higher motion

smoothness. In most cases, the exponent $l$ gives favor to $p'$ such that the true MVs will win over other candidates, since $p' = \alpha > p''$. In the following, we will discuss some important algorithm parameters that we can use to control $m$, $l$, $p'$ and $p''$.

**A. Sensitivity to the density of feature points**

The Harris corner detector [21] is used in this work for feature point detection, where two parameters control the density $\rho$ of the detected feature points. These parameters are the Harris window size and the Harris threshold. In Harris detection, only the maximum Harris response in a window of selected size, larger than the Harris threshold, is selected as a feature point. Evidently, a large window size and a large threshold will decrease the density and increase the repetition ratio, since the detected feature points are more distinct. In TIFM, $\rho$ affects multiple parameters including $m$ and $p''$. When using Eq. (4.8) and Eq. (4.12), a large $\rho$ increases $p''$ and $m$ and leads to a low confidence $F$. For example, if every pixel is detected as a feature point, $p''$ will be equal to unity, and consequently, the true MV will never prevail over other candidates according to Eq. (4.12). In our Harris corner detector, we allow at most one feature point in a $7 \times 7$ window. Combining with an appropriate Harris threshold, the density $\rho$ will mostly be smaller than $0.0057$ point per square pixel ($4,500$ feature points in a $1024 \times 768$ image). Note that with a larger Harris window and a larger Harris threshold, the repetition ratio of the feature points is also increased because the feature points detected in this case are generally more salient and tend to repeat more often.

**B. Sensitivity to the allowed angle of deviation**

As can be derived from Fig. 4.3 and Eq. (4.1), $\varphi$ determines the radius $R$ of the coherent circle, and specifies how similar two MVs should be, in order to satisfy the coherence criterion. It can be deduced from Fig. 4.8, Eq. (4.7) and Eq. (4.8), that the radius $R$ and thus $\varphi$ affect $p'$ and $p''$ directly. First, $\varphi$ has to be sufficiently large such that $R \geq d_{max}$ in order to detect all repeated points (to make sure $p' = \alpha$). This implies that a larger $\varphi$ is required for images with large non-translational motion (large $d_{max}$). On the other hand, according to Eq. (4.8), a large $R$ increases $p''$ and therefore decreases the confidence by Eq. (4.12). Thus, the value of $\varphi$ should be limited, especially when the density $\rho$ is high. Due to this reason, TIFM does not work for images with very large non-translational motion $d_{max}$ and very dense feature points, because in that case, a large $R > d_{max}$ will be required, which leads to a large $p''$ and thus low confidence $F$. In our implementation, we empirically fix $\varphi$ at $2^o$, and the value of $R$ computed by Eq. (4.1) is clipped to 2 if it is smaller than 2. Our experiments demonstrate that this value works well for the tested sequences.

**C. Minimum smoothness required for a neighborhood**

According to Eq. (4.12), we need to find a sufficient number $l$ of CVs to obtain an unambiguous voting for the true MV. This implies that the maximum smoothness $S_{max}$ of a local neighborhood should be above a given smoothness threshold $T_s$, since $l = n \times T_s$. A large $T_s$ and thus a larger $l$ gives a higher confidence of the detected correspondences. Depending on the balance between $p'$ and $p''$ in individual neighborhoods, $T_s$ should be selected adaptively. In TIFM, points are matched in multiple matching passes with decreasing $T_s$, starting from 0.8 down to 0.3. In each pass, if the maximum smoothness $S_{max} > T_s$, we consider

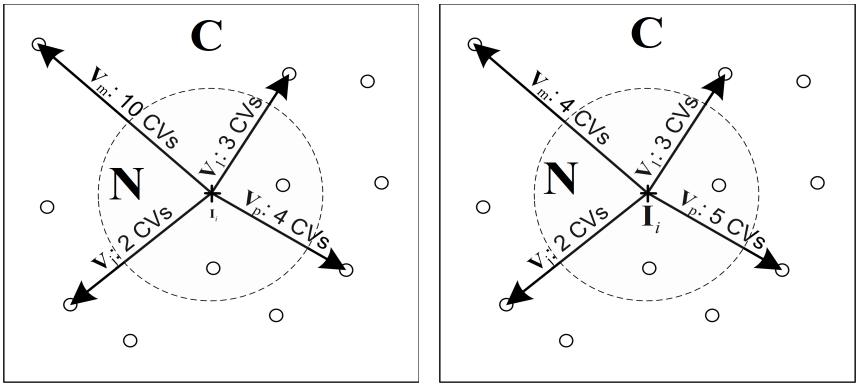
that points are matched unambiguously, as illustrated in Fig. 4.9(a), where $\mathbf{v}_m$ clearly wins over all other candidate MVs with a smoothness of $66\%$. If $S_{max} < T_s$, we consider that the matching is ambiguous. In this case, we keep only the 3 candidate MVs with the highest smoothness ($\mathbf{v}_1$, $\mathbf{v}_p$ and $\mathbf{v}_m$ in Fig. 4.9(b)). In this way, the number of the candidates is significantly reduced from originally $m$ to at most 3 in all subsequent passes. This largely reduces the density $\rho$, thereby reducing both the matching ambiguity and the computation time (discussed in Section 4.5.3). Because of the reduction of candidates in the matching pass, a smaller $T_s$ is allowed in subsequent passes to obtain a sufficient confidence.

**D. Determining the number of feature points in a neighborhood**

The size of a circular neighborhood depends on the number $n$ and the density $\rho$ of the feature points within the neighborhood. The number $n$ should be set with the following two considerations: (1) the number should be large enough to obtain a large $l$ in Eq. (4.12), (2) the neighborhood should be small enough such that the LTM assumption remains valid for the neighborhood. In our implementation, the number $n$ is empirically determined based on the density of the feature points, using the following relation:

$$n = \text{clip}[(12 + \frac{\rho - \rho_1}{\rho_2 - \rho_1} \times 2), 12, 18]. \tag{4.13}$$

In the above equation, $\rho_1 = 1000/(1024 \times 768)$ is the density of 1,000 points in a $1024 \times 768$ image, $\rho_2 = 2000/(1024 \times 768)$, and 'clip' means that $n$ is clipped to $[12 \leq n \leq 18]$. With an appropriate smoothness threshold $T_s$, the number $l = n \times T_s$ of coherent vectors will be larger than 6 in most cases, which will lead to a sufficient confidence by Eq. (4.12).




(a) Unambiguous voting in high-repetition-ratio neighborhoods: 4 candidate MVs where $\mathbf{v}_m$ clearly wins over other candidates in terms of #CVs. After voting, only one candidate ($\mathbf{v}_m$) is kept in the candidate set $\mathbf{C}$.

(b) Ambiguous voting in low-repetition-ratio neighborhoods: 4 candidate MVs where no one clearly wins over others. After voting, only the 3 candidates ($\mathbf{v}_1$,$\mathbf{v}_p$ and $\mathbf{v}_m$) with the highest #CVs are kept in the candidate set $\mathbf{C}$.

**Figure 4.9:** *The voting process of the proposed algorithm in neighborhoods with different repetition ratios.* **(a)** *Unambiguous voting;* **(b)** *Ambiguous voting.*

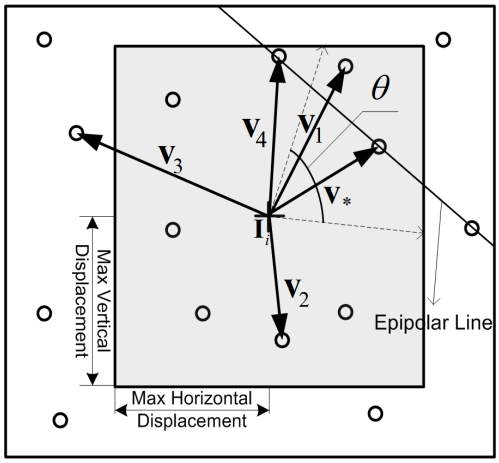



**Figure 4.10:** *Reducing matching ambiguity by imposing multiple constraints. Ambiguous voting among* $\mathbf{v}_1$*,* $\mathbf{v}_2$*,* $\mathbf{v}_3$*,* $\mathbf{v}_4$ *and* $\mathbf{v}_*$*. Angle* $\theta$ *denotes the cone in which the true MV* $\mathbf{v}_*$ *should be located, which can be predicted from neighboring points that have been matched.*

By fixing the number of feature points in each neighborhood using Eq. (4.13), the size of the neighborhood will be automatically adapted to local feature point densities. In image areas with rich texture and thus dense feature points, the neighborhood will be smaller. This is helpful for an accurate feature point matching, since the LTM assumption is 'more' valid in a small neighborhood. On the other hand, in areas with little texture, the size of the neighborhood will be larger, which improves the robustness of the proposed algorithm because a larger neighborhood imposes a stronger smoothing over feature points.

**E. Determining the search range for finding candidate matching points**

The maximum displacements $h_h$ and $h_v$ in Fig. 4.7, i.e., the horizontal and vertical search ranges for candidate matching points, depend on the actual motion between two images, and the availability of additional information about the two images. For example, if the fundamental matrix is known, the search area will simply be a line. If the fundamental matrix is not known, assuming that the search area is a rectangle or circle is a natural choice. As discussed in Section 4.3.8 F, less candidates give less matching ambiguity. Thus, our task is to determine a search range that satisfies the following two conditions: (1) it is large enough such that the true candidate matching point will be in the candidate set $\mathbf{C}$, and (2) it is as small as possible such that the candidate set contains the lowest number of points. This section proposes the following steps to determine the best search range:

1. Specify a minimum search range $h_{min}$ and a maximum search range $h_{max}$.
2. Match the points using $h_{min}$. If the resulting $precision$ (percentage of correctly matched points as defined in Section 4.4.1) is smaller than a given threshold $T_{precision}$, then increment the search range and go to Step 3.

3. Match the points using the new search range. If the resulting gain in $precision$ is larger than a threshold, then increment the search range further, until no further gain in $precision$ can be obtained, or the $precision$ is above $T_{precision}$, or the maximum search range is reached.

In the above steps, the $precision$ can be computed in a simplified way as the ratio between the number of detected correspondences and the number of points in the first image, instead of using Eq. (4.16). The initial value of $T_{precision}$ is assigned as an estimated value of the repetition ratio ($60\%$ in our implementation). For image sequences, $T_{precision}$ is updated after every successive image pair.

As can be derived from the above steps, TIFM is not sensitive to the actual motion in a sequence. As long as the actual motion is smaller than $h_{max}$, TIFM is able to automatically adjust the search range to achieve the best results. For image sequences, it is important that $h_{min}$ is set to the smallest possible value that is larger than the actual motion between most successive images, and $h_{max}$ has a value that is larger than the largest motion. If the image motion within a sequence has a small variation, a single search range with $h_{min} = h_{max}$ will lead to the same results. The above steps are mainly useful for sequences where image motion between most successive images is small, while some occasional image pairs may have very large image motion. In that case, a single search range increases the computation time and decreases the matching performance, because of the higher number of matching candidates.

**F. Using additional matching constraints**

In image areas with low repetition ratio, $p'$ cannot easily win over $p''$, and consequently, it is difficult to correctly match the feature points. As seen from Eq. (4.12), a possible solution is either to reduce the number of candidates $m$, or to reduce $p''$. From Eq. (4.8), we observe that reducing $m$ also reduces $p''$. Thus, the only solution is to reduce the number of candidates $m$.

To reduce the number of matching candidates $m$ and thus the matching ambiguity, more constraints such as the epipolar constraint, photo consistency, uniqueness constraint, etc., can be used. As shown in Fig. 4.10, by limiting the maximum displacement of the feature point, $\mathbf{v}_3$ is rejected. By incorporating the epipolar constraint, $\mathbf{v}_1$ and $\mathbf{v}_2$ are rejected. When predicting the motion direction from neighboring MVs, $\mathbf{v}_4$ is rejected. Hence, the ambiguity among the several promising candidates is gradually removed and the true correspondence $\mathbf{v}_*$ is found. In our implementation, we only use the uniqueness constraint that ensures one-to-one correspondences: once a feature point in the second image is found corresponding to a feature point in the first image, it is removed immediately from the candidate sets of other feature points. In this way, many-to-one correspondences are avoided. To avoid one-to-many matching, once a point in the first image finds its correspondence, all other candidates are removed from its candidate set immediately.

The above candidate removal procedure to ensure one-to-one matching is not optimal, and will lead to the following consequence: if the first matching is false, the feature point in the first image can never be correctly matched again, since all other candidates are removed. Furthermore, it also impacts the matching process for neighboring points, since the matching point in the second image is removed from the candidate sets of all the neighboring points.

A better strategy would be: first allow the many-to-many matching and then select the best matching, using global optimization techniques. However, this is not implemented in our algorithm for the following two reasons. (1) The probability for the first matching to be false is low. As seen from our experimental results, the recall is above $95\%$ in most cases, so that the impact of the above suboptimal candidate-removal strategy is limited. (2) It is not straightforward to come up with an efficient global optimization algorithm for optimizing the matching process.

## 4.4 Evaluating the correctness of TIFM using synthetic data

We test the proposed TIFM algorithm on synthetic data to show its correctness. First, we introduce three criteria for evaluating the performance of TIFM. Second, we explain the generation of synthetic images with controlled noise levels, rotations and translations. After that, TIFM is applied to match the feature points in the synthetic images.

### 4.4.1 Evaluation criteria for feature point matching

In general, only a portion of the detected points can be matched, from which only a percentage of the detected matches are correct. For feature point matching, the results are presented with the parameters $\#$*CorrectMatches*, *recall* and *precision*, as will be introduced in the sequel. A correct match is determined based on its conformity to, either the homography matrix $H$, or the fundamental matrix $F$ that is computed using the RANSAC algorithm. The metric for evaluating the correctness of a match is that the associated residual error $d_r$ should be smaller than a given *matching threshold*. The residual error is computed by

$$d_r = \begin{cases} [d(x', Fx) + d(x, F^T x')]/2, & \text{when } F \text{ is given, or,} \\ [d(x', Hx) + d(x, H^{-1} x')]/2, & \text{when } H \text{ is given,} \end{cases} \tag{4.14}$$

where $(x, x')$ is a pair of matched feature points, $d(\cdot, \cdot)$ is the Euclidian distance between the feature point and the epipolar line when $F$ is given, or $d(\cdot, \cdot)$ represents the Euclidian distance between the two feature points when $H$ is given. The *recall* and *precision* are computed by

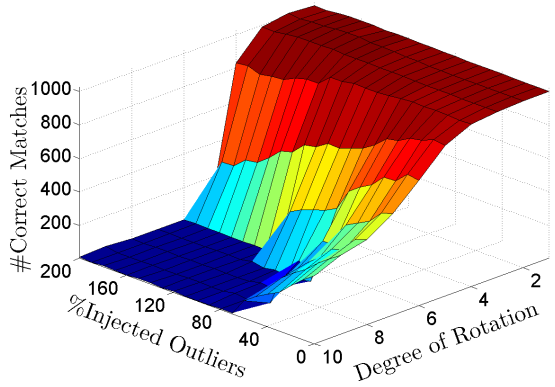
$$recall = \frac{\#CorrectMatches}{\#DetectedMatches}, \tag{4.15}$$

$$precision = \frac{\#CorrectMatches}{\#PointsIn1stImage}. \tag{4.16}$$

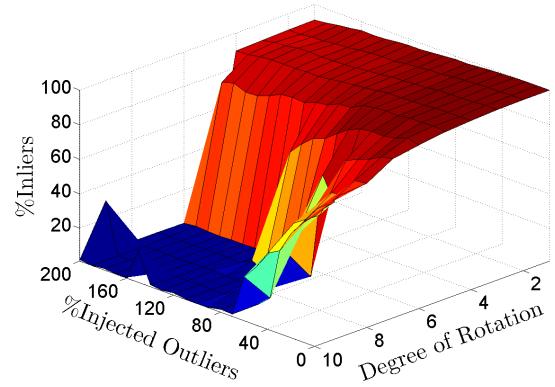
Parameter *recall* is the percentage of the correct matches among the total detected matches, which measures the quality of the detected correspondences. Parameter *precision* is the percentage of all detected feature points that are correctly matched, which indicates the efficiency of an algorithm.

### 4.4.2 Results on synthetic images

First, we generate an $800 \times 600$ image with $1,000$ randomly-distributed points, where the distance between any two points is not smaller than 3 pixels. Second, the $1,000$ feature points




(a) *#CorrectMatches*.

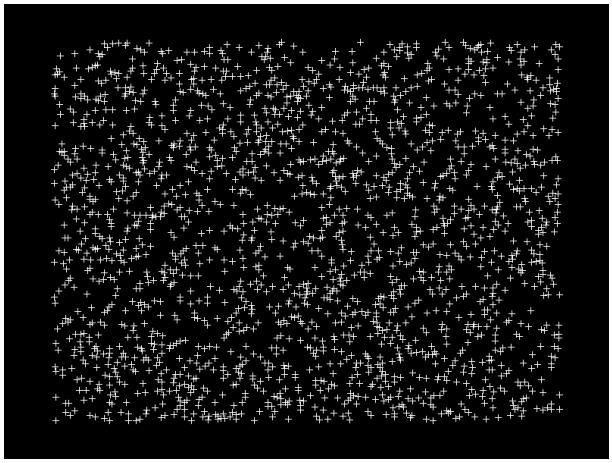



(b) *recall*.

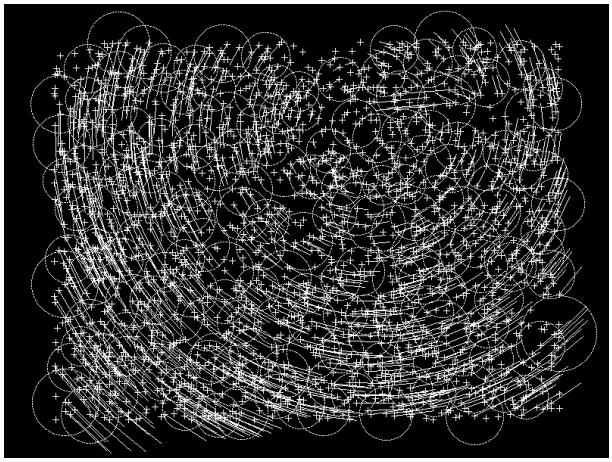
**Figure 4.11:** *Feature point matching results by TIFM on synthetic images with different settings of Degrees of Rotation (DoR) and Percentages of Injected Outliers (PIO).*

are rotated and translated with controlled rotation and translation parameters to generate the second image. Third, an equal number of randomly-distributed outliers are added to both images, where it is ensured that the distance between any two points is not smaller than 3 pixels. TIFM is then applied to the two images to match the points. The homography is then computed on the obtained correspondences for performance evaluation. The matching threshold is equal to 1 pixel.

Many experiments have been conducted to test the proposed algorithm under different settings of *Degrees of Rotation (DoR)* and *Percentages of Injected Outliers (PIO)*. Parameter *DoR* is the angle that the image rotates about its image center, which controls the strength of non-translational motion. Parameter *PIO* is the ratio between the number of outliers added



(a) The first synthetic image with 1000 correct points and 1000 noisy points.



(b) The detected MVs and associated neighborhoods displayed on top of the first image with noisy points.

**Figure 4.12:** *Results on one simulated image pair when $DoR = 4^o$ and $PIO = 100\%$. A total of 1,016 correspondences are detected, among which $94.9\%$ are conforming to the homography.*

to the images, and the number of the repeated points yielding true correspondences (1,000 points in our experiments). When setting a value for the $PIO$, the repetition ratio $\alpha$ can be computed as $\alpha = 1/(1 + PIO)$.

Figs. 4.11(a) and 4.11(b) show the results obtained by TIFM on synthetic images. It can be observed that the *DoR* value increases from $0$ to $10$ degrees (from pure translation to significant non-translation), and the *PIO* value grows from $0\%$ to $200\%$ (repetition ratio

decreases from $100\%$ to $33\%$). In all experiments, the translation between two images is kept constant at $(5, 10)$, i.e., vertical translation is 5 pixels and horizontal shift is 10 pixels.

From Figs. 4.11(a) and 4.11(b), we deduce the following findings.

(1) TIFM is able to reliably detect the correspondences, even when the image contains many outliers and evident rotations. For example, when $PIO = 100\%$ and $DoR = 4^o$, we have found $964$ correct matches. Furthermore, $94.9\%$ of the $1,016$ detected correspondences align with the homography. The obtained MVs are shown in Fig. 4.12(b), where a large rotation is clearly visible.

(2) The $\#CorrectMatches$ and $recall$ drop when the rotation increases above a certain level. As can be observed from Figs. 4.11(a) and 4.11(b) when $DoR$ increases above $5^o$, the performance drops significantly. As we discussed in Section 4.3.8 B, TIFM assumes that the local motion field is 'translational'. The large deviation between two MVs at a large rotation angle leads to a violation of the LTM assumption. Consequently, the performance of TIFM deteriorates.

(3) The noise (density of points) has a minor influence on the performance when the rotation is small, but has a large impact when the rotation is large. As stated in Sections 4.3.8 A and 4.3.8 B, the reason is that a large non-translational motion $d_{max}$ and dense feature points $\rho$ increase the values of $m$ and $p''$ and decrease $p'$. Therefore, it is difficult to correctly match feature points with a sufficient confidence.

## 4.5 Experimental results

We test the proposed TIFM algorithm against three other well-known algorithms for both feature point matching and feature point tracking. The three algorithms are SIFT, KLT and the Block Matching (BM) algorithm[5]. For the BM algorithm, the sum of the absolute differences between the two $7 \times 7$ pixel windows is computed. A correspondence is established if the difference is minimal among candidates and below a given threshold.

Tables 4.1 and 4.2 list all the test image sequences and image pairs, which are depicted in Fig. 4.13. The search ranges (maximum displacements) discussed in Section 4.3.8 E, for individual sequences are also shown in Table 4.1. In our experiments, the same maximum displacements are used for TIFM, SIFT and BM. For TIFM, besides the maximum displacements, minimum displacements are also specified. The matching threshold is set to one pixel unless stated otherwise. In the following discussion, we first present the results of feature point matching and then the results of feature point tracking. To track feature points across frames, the two-frame correspondences obtained by TIFM are linked to obtain multiple-frame correspondences.

### 4.5.1 Results of feature point matching

Fig. 4.14 shows the feature point matching results obtained by TIFM, SIFT and BM for individual images pairs in Table 4.2. From the figure, we can see that TIFM obtains the largest *#CorrectMatches* for 5 out of 6 image pairs. The *recall* of TIFM is comparable to

[5]Executables or source codes of SIFT and KLT are available from `http://www.cs.ubc.ca/~lowe/keypoints/` and `http://www.ces.clemson.edu/~stb/klt/`, respectively.

**Table 4.1:** *Test sequences.*

| Seq (#frm) | Description | max pixel displacement |
|---|---|---|
| *castle* (26) | Fig. 4.13(a); from `www.cs.unc.edu/~marc/` | 80 (moderate motion) |
| *house* (16) | Fig. 4.13(b); by hand-held camera | 80 (moderate motion) |
| *church* (25) | Fig. 4.13(c); by hand-held camera | 60 (moderate motion) |
| *kspoort* (22) | Fig. 4.13(d); by hand-held camera | 60 (moderate motion) |
| *lab* (150) | Fig. 4.13(e); by hand-held camcorder | 30 (small motion) |
| *medusa* (194) | Fig. 4.13(f); from `www.cs.unc.edu/~marc/` | 50 (small motion) |
| *leuven* (6) | Fig. 4.13(h); from `www.robots.ox.ac.uk/~vgg/research/affine`; significant brightness change | 30 (small motion) |
| *campus* (2000) | Fig. 4.13(i); by hand-held moving camcorder; with disappearing and appearing scene contents | 50 (small motion) |

**Table 4.2:** *Test image pairs.*

| ImagePair | Description |
|---|---|
| IP1 | Fig. 4.13(a); extracted from *castle* |
| IP2 | Fig. 4.13(b); extracted from *house*; evident light change |
| IP3 | Fig. 4.13(f); extracted from *medusa* |
| IP4 | Fig. 4.13(g); by hand-held camera; containing large reflecting objects |
| L01 | Fig. 4.13(h); the two brightest images from *leuven*; large brightness change |
| L05 | Fig. 4.13(h); the brightest and darkest images from *leuven*; significant brightness change |

that of SIFT, while *precision* is much higher than when using SIFT and BM. This implies that TIFM is accurate and more efficient. A large number of correspondences can be obtained with a high accuracy without the need to detect many feature points. Note that TIFM and BM use identical Harris corner detectors[6], while SIFT uses another detector. As a result, the number of feature points detected by TIFM and SIFT can be different from each other. In most cases, SIFT detects more points than TIFM. However, the $\#$*CorrectMatches* detected by SIFT is not higher than with TIFM. As will be further discussed in Section 4.5.3, the high $precision$ of TIFM is due to the use of the smoothness constraint alone, since the drift of the feature points can hardly break the smoothness constraint and thus more points can be matched.

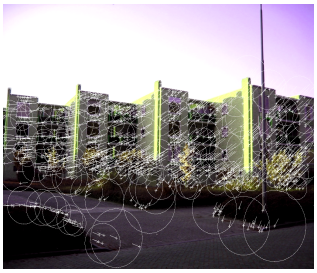
The results on the image pairs L01, L05 and IP2 in Table 4.2, which contain clear light changes, demonstrate the robustness of TIFM to light changes. Results on IP4 show the potential of the TIFM for images containing non-Lambertian objects, due to the *texture independence* of the algorithm. It is not surprising that BM works only for IP1 and IP3 with small light changes. We have applied TIFM to every two successive images of the first six test sequences as listed in Table 4.1. The results are shown in Fig. 4.15, where similar conclusions can be made.

### 4.5.2 Results of feature point tracking

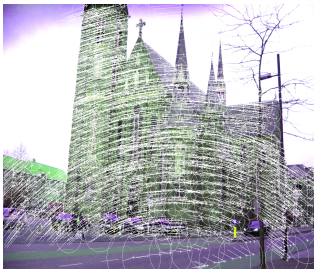
This section presents our experimental results on seven image sequences that are listed in Table 4.1 (all except the *leuven* sequence). The performance of the proposed TIFM algorithm

[6]We attempt to detect approximately 3,000 points in every image by adjusting the threshold for the Harris corner detector. However, the number of actually detected points can differ from 3,000, depending on the image contents.

(a) *castle* (IP1) with MVs.

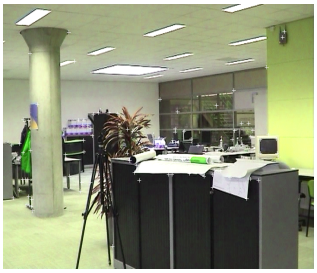


(b) *house* (IP2) with MVs.



(c) *church* with MVs.

(d) *kspoort* with 579 points tracked along 22 frames.



(e) *lab* with 97 points tracked along 51 frames.

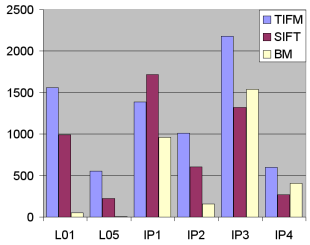
(f) *medusa* (IP3) with MVs.

(g) 1st image of IP4 with MVs.

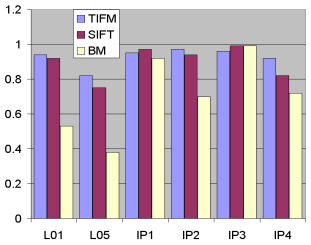
(h) *leuven* (L01 and L05) with MVs.

(i) frame 180 of *campus*

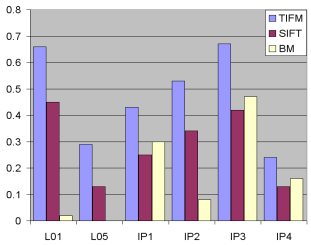
**Figure 4.13:** *Test sequences and image pairs superimposed with detected MVs or tracked feature points by TIFM.*




(a) *#CorrectMatches*.

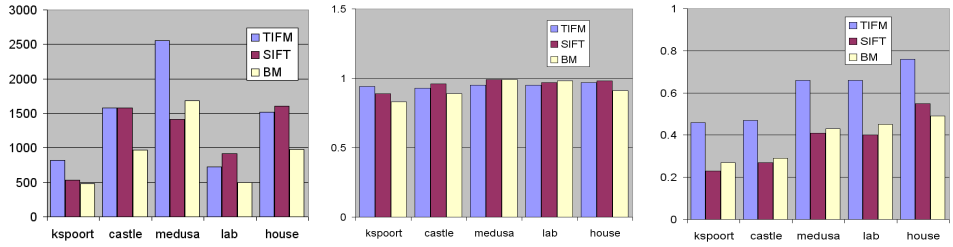



(b) *recall*.




(c) *precision*.

**Figure 4.14:** *Matching results by the TIFM, SIFT, and BM algorithms for individual image pairs.*




(a) *#CorrectMatches*. (b) *recall*. (c) *precision*.

**Figure 4.15:** *Averaged matching results by TIFM, SIFT and BM on successive image pairs of the test sequences.*

is compared against SIFT, KLT and BM using three criteria, which will be introduced next.

### A. Evaluation criteria for feature point tracking

Three metrics are used to evaluate the performance of feature point tracking: (1) the *number of the tracked feature points*, (2) the *mean and standard deviation* of the residual errors of the point correspondences between the first and the last images of the feature point track, and (3) the *failure or success* of the factorization-based 3D reconstruction [20]. The factorization-based 3D reconstruction first collects all tracked feature points into a so-called measurement matrix. The measurement matrix is then factorized into the projective shape matrix and projective motion matrix using iterative methods. For a robust and accurate factorization-based 3D reconstruction, it is crucial to track a large number of correspondences along a long sequence of images with a high accuracy.

To compute the mean and standard deviation of the residual errors, the fundamental matrix between the first and the last images of the feature point tracks is computed, and the residual errors are thereafter computed using Eq. (4.14). The motivation for using the results of 3D reconstruction as a metric for feature point tracking is that 3D reconstruction is sensitive to the number, spatial distribution and localization accuracy of the tracked feature points. Thus, as long as the tracked correspondences successfully render a 3D reconstruction, they can be considered of high quality.

### B. Results on seven image sequences

To track points across frames in image sequence, correspondences between every two consecutive frames are first computed, and then linked to obtain multiple-frame feature point correspondences. For all tested algorithms, the erroneous correspondences are not rejected prior to 3D reconstruction.

Table 4.3 shows the tracking results obtained by the four tested algorithms on the *medusa* sequence, where we see that TIFM performs much better than other algorithms in terms of both the number and the quality of the tracked feature points. Among the six results for tracking over 6 frames up to 181 frames, TIFM outperforms SIFT, KLT and BM by successfully tracking at least 2.5 times the amount of feature points tracked by the other algorithms. Furthermore, only the first $(0 \rightarrow 5)$ and the last $(0 \rightarrow 180)$ tracking intervals fail

**Table 4.3:** *Tracking results on* medusa*: #*TrackedPoints*, the* Success (S) or Failure (F) *of the 3D reconstruction, as well as the* mean *and* standard deviation *of the residual errors of the correspondences between the first and the last images. For example, '*$0 \rightarrow 180$*' means that points are tracked from frame 0 to frame 180; the term '1922-F' means that 1922 points are tracked and the 3D reconstruction is a failure; the term '(0.29,0.32)' denotes that the mean of the residual errors is 0.29 pixels and the standard deviation is 0.32 pixels.*

| track | $0 \rightarrow 5$ | $0 \rightarrow 10$ | $0 \rightarrow 50$ | $0 \rightarrow 150$ | $0 \rightarrow 160$ | $0 \rightarrow 180$ |
|---|---|---|---|---|---|---|
| TIFM | 1922-F | 1412-S | 473-S | 63-S | 49-S | 32-F |
| | (0.29,0.32) | (0.30,0.33) | (0.38,0.35) | (0.61,0.58) | (0.69,0.63) | (0.83,0.67) |
| SIFT | 726-F | 438-S | 57-S | 3-F | 1-F | 0-F |
| | (0.20,0.23) | (0.27,0.27) | (0.71,0.76) | - | - | - |
| KLT | 869-F | 619-F | 190-F | 19-F | 15-F | 10-F |
| | (0.34,1.7) | (0.70,3.5) | (2.6,8.1) | (7.0,6.7) | (5.6,8.7) | (7.5,8.4) |
| BM | 132-F | 5-F | 0-F | 0-F | 0-F | 0-F |
| | (0.15,0.13) | - | - | - | - | - |

**Table 4.4:** *Tracking results on* kspoort *and* castle.

| track | $0 \rightarrow 5$ | $0 \rightarrow 15$ | $0 \rightarrow 21$ |
|---|---|---|---|
| TIFM | 919-S | 640-S | 579-S |
| | (0.48,0.55) | (0.40,0.42) | (0.37,0.50) |
| SIFT | 137-F | 56-F | 46-F |
| | (1.0,3.6) | (0.94,1.1) | (1.1,1.5) |
| KLT | 400-F | 69-F | 61-F |
| | (27,35) | (16,25) | (19,26) |
| BM | 22-F | 0-F | 0-F |
| | (5.6,14) | - | - |

**(a)** *kspoort*

| track | $0 \rightarrow 5$ | $0 \rightarrow 15$ | $0 \rightarrow 25$ |
|---|---|---|---|
| TIFM | 912-S | 430-S | 280-S |
| | (0.36,0.43) | (0.49,0.53) | (0.62,0.69) |
| SIFT | 443-S | 106-S | 33-S |
| | (0.48,1.7) | (0.47,0.46) | (1.2,1.4) |
| KLT | 88-F | 5-F | 0-F |
| | (8.6,13) | - | - |
| BM | 57-F | 0-F | 0-F |
| | (1.8,7.6) | - | - |

**(b)** *castle*

**Table 4.5:** *Tracking results on* house *and* lab.

| track | $0 \rightarrow 5$ | $0 \rightarrow 10$ | $0 \rightarrow 14$ |
|---|---|---|---|
| TIFM | 437-F | 340-S | 314-S |
| | (0.49,2.4) | (0.57,2.3) | (0.35,0.45) |
| SIFT | 216-F | 124-F | 105-F |
| | (0.57,1.5) | (1.5,7.8) | (0.56,0.99) |
| KLT | 285-F | 53-F | 17-F |
| | (25,31) | (21,27) | (27,37) |
| BM | 18-F | 0-F | 0-F |
| | (3.4,14) | - | - |

**(a)** *house*

| track | $0 \rightarrow 10$ | $0 \rightarrow 50$ | $0 \rightarrow 100$ | $0 \rightarrow 150$ |
|---|---|---|---|---|
| TIFM | 272-S | 97-F | 33-F | 24-F |
| | (0.50,0.69) | (0.64,1.0) | (1.1,0.74) | (2.8,8.6) |
| SIFT | 106-S | 26-F | 7-F | 4-F |
| | (0.7,1.1) | (1.3,1.8) | - | - |
| KLT | 542-F | 186-F | 93-F | 62-F |
| | (4.6,8.7) | (2.4,3.9) | (3.5,5.7) | (2.4,3.5) |
| BM | 0-F | 0-F | 0-F | 0-F |
| | - | - | - | - |

**(b)** *lab*

for the factorization-based 3D reconstruction for TIFM. The failure of the 6-frame tracking is due to insufficient disparity between the 6 frames because of the small camera baseline. Failure for the 181-frame tracking could be due to the insufficient number of the tracked feature points. The mean and standard deviation obtained by TIFM are comparable to or better than those obtained by SIFT.

Tables 4.4, 4.5 and 4.6 list the tracking results on *kspoort*, *castle*, *house*, *lab*, *church* and *campus*, where similar observations can be made. That is, TIFM is able to track more points

**Table 4.6:** *Tracking results on* church *and* campus.

| track | $0 \rightarrow 5$ | $0 \rightarrow 10$ | $0 \rightarrow 20$ | $0 \rightarrow 24$ |
|---|---|---|---|---|
| TIFM | 1405-S | 1065-S | 782-S | 662-S |
| | (0.29, 0.41) | (0.29, 0.47) | (0.44, 0.71) | (0.73, 4.4) |
| SIFT | 661-S | 356-S | 185-S | 152-S |
| | (0.46, 1.7) | (0.51, 1.7) | (0.38, 0.59) | (0.38, 0.63) |

**(a)** *church*

| track | $0 \rightarrow 50$ | $0 \rightarrow 150$ | $0 \rightarrow 250$ | $0 \rightarrow 300$ |
|---|---|---|---|---|
| TIFM | 1030-S | 191-S | 42-S | 22-F |
| | (0.3,0.34) | (0.55,0.63) | (0.43,0.39) | (0.74,0.66) |
| SIFT | 363-F | 40-S | 11-F | 0-F |
| | (0.4,0.63) | (1.6,3.9) | (2.6,3.6) | - |

**(b)** *campus*

(at least twice) along more images and with a higher quality (measured by the factorization-based 3D reconstruction) than when using SIFT, KLT and BM. Table 4.6(b) shows the tracking results on the *campus* sequence, which is captured by a hand-held camcorder with disappearing and appearing scene contents, where we see that TIFM also performs much better than SIFT. For the *lab* sequence that is captured using a hand-held camcorder at a frame rate of 30 fps, KLT tracks the largest number of points. However, a large percentage of outliers occurs in the tracked feature points, which leads to the failure of the 3D reconstruction. From the tables, we observe that KLT and BM fail for all 3D reconstructions. Outlier rejection is necessary for KLT and BM. KLT and BM rely on image texture correlation for feature matching. It is not surprising that they work only the *medusa* sequence which contains small image motion, small light change, and rich textures.

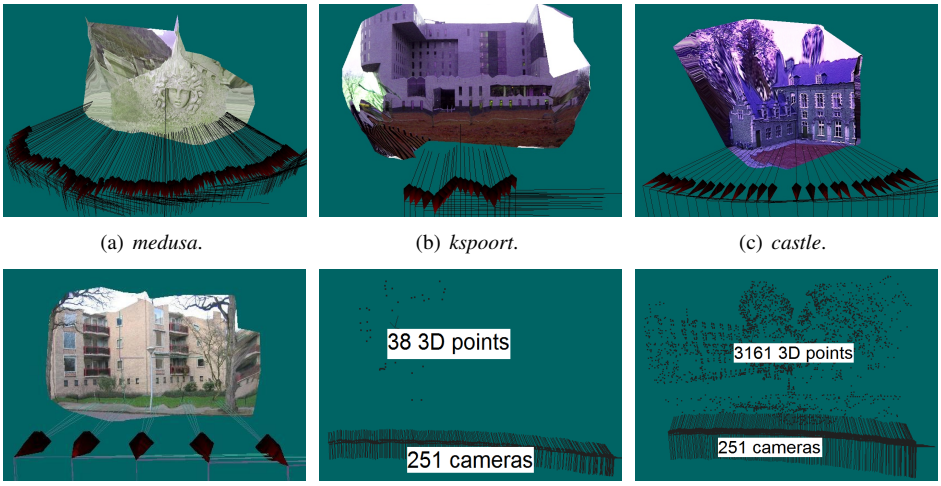
Fig. 4.16 provides an example of five sparse 3D models that are reconstructed using the correspondences tracked by TIFM. Note that when the camera baseline is small (e.g. *medusa* and *campus*), a large number of accurate correspondences will be required for 3D reconstruction. As we see from Tables 4.3 and 4.6(b), TIFM provides an excellent solution for such small-motion scenarios. Even for sequences with moderate motion (e.g *castle*), TIFM also tracks a significantly larger number of feature points than KLT and SIFT, as seen from Table 4.4(b).

### 4.5.3 Discussion on the results

Our experimental results show that TIFM is able to track a large number of feature points along a long sequence of images with a high accuracy. This section discusses the reasons for the algorithm performance. The computational complexity of TIFM is discussed as well.

#### A. Accuracy of the detected correspondences

As can be seen from Figs. 4.12(b), 4.14(b) and 4.15(b), the accuracy of the correspondences detected by TIFM is very high. The obtained *recall* is in most cases above $95\%$, with a matching threshold of one pixel.




(a) *medusa*. (b) *kspoort*. (c) *castle*.

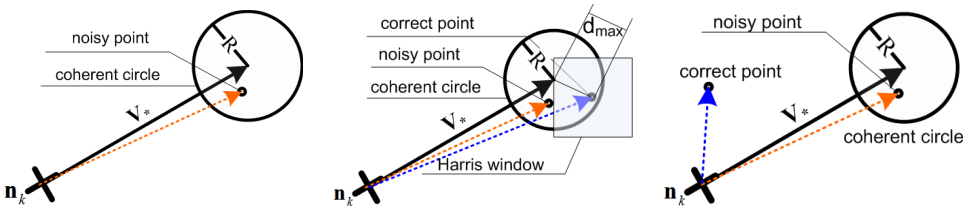
(d) *rotation*. (e) Reconstructed point cloud for *campus*: before triangulation. (f) Reconstructed point cloud for *campus*: after triangulation.

**Figure 4.16:** *Five sparse 3D models obtained by the factorization method using the tracked correspondences by TIFM.*

In TIFM, an erroneous matching occurs in the following two situations. (1) The voting process selects a wrong MV for a neighborhood. In this case, all correspondences in the neighborhood are wrong. We refer to such a neighborhood as *erroneous neighborhood*. (2) The voting process selects the correct MV for $\mathbf{I}_i$. However, noisy points occur within the shifted coherent circle for point $\mathbf{n}_k \in \mathbf{N}$, which leads to an erroneous matching for $\mathbf{n}_k$. This situation can be further divided into three scenarios, which are illustrated in Figs. 4.17(a),(b) and (c). This type of neighborhood is referred to as *correct neighborhood*.

Assuming a uniform distribution of the feature points, the probability that the erroneous matching in Fig. 4.17(a) occurs is equal to the probability to find a noisy point in the circle, which can be approximately computed as $\rho\pi R^2$ using Eq. (4.8) (note that the density $\rho$ in this case is the density of the noisy points). For most images with limited motion, the probability that such erroneous matching occurs is very low. For example, with 5,000 points in a $1024 \times 768$ image, $\varphi = 2^o$, $\alpha = 40\%$ and $||\mathbf{v}_*|| = 70$, the probability computed by Eq. (4.8) is $7.1\%$. For the case of Fig. 4.17(b), the probability that the erroneous matching occurs is even smaller, since the image area where the noisy point has to be located is smaller. For Fig. 4.17(c), the probability of erroneous matching is the lowest, since in practice, very few neighborhoods will contain object boundaries. Note that when considering tracking feature points in limited-motion sequences, the assumed values for $d_{max}$, $\rho$ and $\alpha$ in the above calculation are already close to the worst-case scenarios.

From the above discussion, we conclude that the probability for the erroneous matching as shown in the cases of Fig. 4.17 is very low. Furthermore, even when the erroneous matching in Figs. 4.17(a) and (b) occurs, the false MV will stay close to the true MV, i.e., the dotted and solid MVs in Figs. 4.17(a) and (b) will stay close to each other, due to the use of the smoothness constraint. The exception is depicted in Fig. 4.17(c), when the neighbor-




(a) A feature point does not repeat itself within the coherent circle and a noisy point is found within the coherent circle. The probability to find such a noisy point depends on the radius of the coherent circle $R$ and the density of the noisy points $\rho$. As discussed in Section 4.3.8B, $R$ should be larger than the maximum deviation $d_{max}$ between two true MVs. Thus, the probability depends on $\rho$ and $d_{max}$.

(b) A feature point repeats itself within the coherent circle. But a noisy point is also found within the circle, with a closer distance to the circle center. The probability to find such a noisy point depends on $d_{max}$, $\rho$, and the size of the Harris window, since the noisy point has to be located outside the Harris window and closer to the circle center.

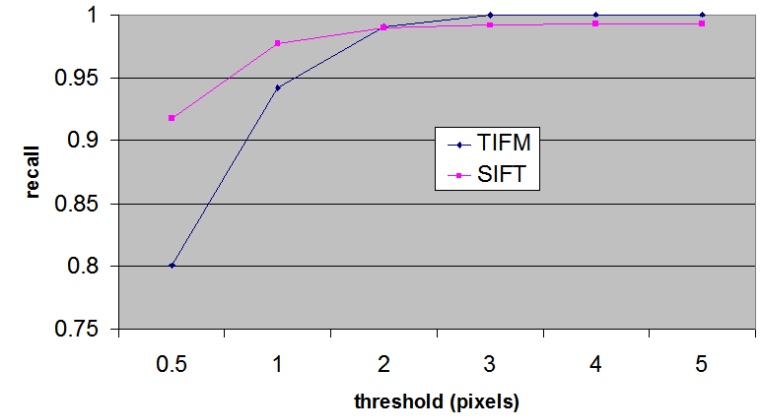
(c) A noisy point is found within the coherent circle, which is far away from the correct point. This scenario occurs mostly when the neighborhood contains object boundaries and thus the smoothness assumption is invalid. As a result, for point $\mathbf{n}_k$ located on the other side of the object boundary, the true MV will differ significantly from $\mathbf{v}_*$.

**Figure 4.17:** *Three erroneous matching scenarios for* correct *neighborhoods, where the true MV $\mathbf{v}_*$ is successfully detected. However, erroneous matching occurs for feature point $\mathbf{n}_k$. Note that the MV of $\mathbf{n}_k$ is detected as the most-similar vector to $\mathbf{v}_*$ using Eq. (4.2).*

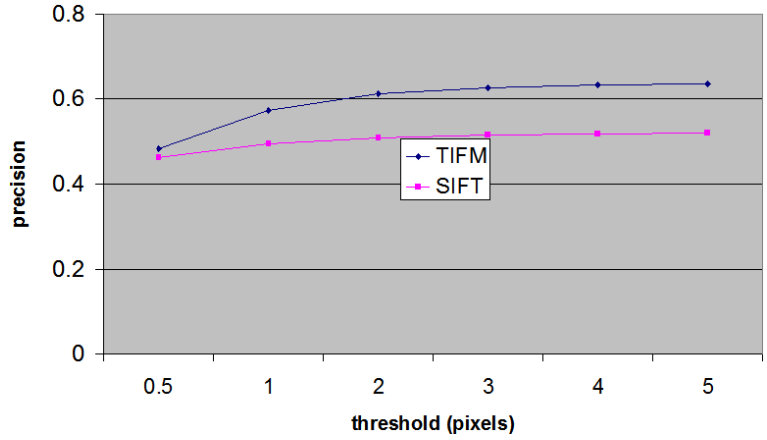
hood contains object boundaries and the smoothness assumption is valid. In that case, the dotted and solid MVs can be far apart. However, since few neighborhoods will contain object boundaries in practice, the probability that the situation in Fig. 4.17(c) occurs is very low. Therefore, we conclude that, even though the false MVs in correct neighborhoods may exceed the matching threshold, they can be considered correct when a slightly larger threshold is used. This contributes to the large number of the tracked feature points, since more drifting points can be tracked.

To quantitatively evaluate how close the erroneous matching obtained by TIFM comes to the true matching, we perform the feature point matching for every image pair of the *rotation* sequence as depicted in Fig. 4.16(d). Since every 5 successive images in *rotation* have the same camera center, the homography is computed for performance evaluation[7]. The matching results are evaluated using different matching thresholds, and the obtained *recall* and *precision* are shown in Fig. 4.18. As can be noticed from the figure, the *recall* and *precision* for TIFM increase clearly when the matching threshold becomes larger, and the *recall* reaches nearly $100\%$ when the threshold is above 3 pixels. For SIFT, the increase of *recall* and *precision* is not as pronounced as for TIFM. This validates our observation that many erroneous correspondences in TIFM are actually approximately correct.

[7]Since homography is not applicable to two images of a non-planar scene with different camera centers, the results between images with different centers are excluded for performance evaluation.




(a) Average $recall$.




(b) Average $precision$.

**Figure 4.18:** *Average $recall$ and $precision$ for the* rotation *sequence under different matching thresholds. The homography is used for performance evaluation in this case.*

### B. Tracking capability of the proposed algorithm

The experimental results of the proposed algorithm in Section 4.5.2 show a very good tracking capability. The large number of the tracked feature points are obtained due to the use of the smoothness constraint. The smoothness constraint is robust to the drift (see Section 6.3.1) of feature points. This is because the image motion stays smooth in local neighborhoods despite the drift of feature points. As a result, more 'drifted' feature points can be matched by the proposed algorithm. This contributes to the large number of feature points tracked by the proposed algorithm. Below, we give a further analysis of the tracking capability of the proposed algorithm.

To track a feature point, the feature point should repeat in consecutive frames and should

be correctly matched in those frames. The number of tracked points is dependent on: (1) the number $n_1$ of feature points in the first image; (2) the repetition ratio $\alpha$ of the feature points in the first image; (3) the repetition ratio $\beta$ of the tracked points[8] in all remaining images, and (4) the percentage $r$ of the tracked feature points that are correctly matched in every frame. For a relative comparison between two algorithms, $r$ can be estimated as the *precision*, which is defined as the percentage of the correctly-matched feature points. Thus, for a relative comparison, the number $n_l$ of the feature points that can be tracked along $l$ frames can be computed as

$$n_l = n_1(\alpha r)(\beta r)\cdots(\beta r) = n_1\alpha\beta^{l-1}(precision)^l. \tag{4.17}$$

Based on the *precision* obtained by the tested algorithms, as shown in Fig. 4.15, and assuming the same repetition ratios for all algorithms, it can be deduced that TIFM is able to track the largest number of feature points. The intuitive understanding of this is that, by using the smoothness constraint alone, any feature point can be matched as long as it follows the smooth 'point flow' in an image sequence. Thus, more feature points can be matched and tracked despite that they may drift over time.

**C. Discussion on using the appearance constraint**

Appearance similarity is an important constraint that is used by almost every feature matching algorithm. However, it is not employed in the proposed algorithm, because the smoothness assumption is already sufficiently constraining the feature matching process due to our new design presented in Section 4.3. Using the appearance constraint will not bring any performance improvement. Instead, it may adversely affect the algorithm performance. This section gives a further analysis on this phenomenon.

As discussed in Section 4.3.8 C, additional constraints can be used in the proposed algorithm to reduce the number of candidates in $\mathbf{C}$. To demonstrate the effectiveness of the appearance constraint on the proposed algorithm, the following steps have been implemented for determining the candidate matching points.

1. For each point $\mathbf{n} \in \mathbf{N}$, compute the normalized cross correlation (NCC) between two $7 \times 7$ pixel windows around point $\mathbf{n}$ and every candidate matching point $\mathbf{c} \in \mathbf{C}$.

2. Based on the computed NCCs, we keep at most three candidates with the highest NCCs in the candidate set $\mathbf{C}$.

Based on the small-sized candidate sets, feature points are matched by maximizing the local motion smoothness using the same process as described in Section 4.3.6. We have performed the experiment on the *castle* sequence and the results are presented in Table 4.7. From the table, we observe that using the appearance constraint improves the *recall* while it decreases the *precision*. The number of the points that are tracked is also significantly

[8]The repetition ratio $\beta$ of the tracked points should be higher than the repetition ratio $\alpha$, since $\beta$ is evaluated only on the points that have repeated and have been matched in the previous frame. This implies that these points are most probably located in areas with sufficient intensity variations and tend to repeat more often than others.

**Table 4.7:** *Matching results on* castle *with/without using the appearance constraint. The numbers in the table are the average results over all image pairs of the sequence. The number of tracked points is obtained by tracking points from frame 0 to frame 20.*

| | *#CorrectMatches* | *recall* | *precision* | # tracked points |
|---|---|---|---|---|
| with | 1331 | 0.971 | 0.413 | 138 |
| without | 1605 | 0.921 | 0.498 | 338 |

reduced. As can be seen, from frame 0 to frame 20, the number of tracked points is reduced from 338 down to 138 points. From the perspective of feature point tracking, the use of the appearance constraint significantly degrades the algorithm performance.

Summarizing, employing the appearance constraint has two consequences: (1) on one hand, it poses an additional constraint on the matching process and thereby reduces the number of false matches; (2) on the other hand, it also falsely rejects some correct correspondences that otherwise can be successfully matched. It is our observation that a matching algorithm should use as few constraints as possible, as long as the imposed constraints are able to sufficiently remove the matching ambiguity. More constraints increase the probability of ruling out some possible correct matches, and consequently lead to less tracked feature points. For example, the locations of feature points may drift over time due to viewpoint change, illumination change, camera motion, etc [93]. Such drift may easily violate the appearance constraint, while it can hardly affect the smoothness constraint as discussed earlier. Thus, by avoiding the appearance constraint, the proposed algorithm is able to match more of such drifting points, as can be observed from Table 4.7. This increases the *precision* and thus the number of the tracked feature points.

**D. Computational complexity of TIFM**

When observing Fig. 4.5 and assuming ($m_1 = m_2 = \cdots = m_n = m$), we have ($n \times m$) candidate MVs for $n$ points in a neighborhood, among which only ($n \times \alpha$) MVs are true MVs. To detect these true MVs, we compare each of the $m$ candidate MVs for point $\mathbf{n}_i$ with other $[(n-1) \times m]$ candidate MVs to determine the true MV. After the true MV for $\mathbf{n}_i$ is determined, the true MVs for all other repeated points in the neighborhood are determined as well. Thus, the computational complexity for matching one point can be computed with a complexity in the order of $O(m^2)$, which suggests that the processing speed of TIFM depends on the number of the candidate matching points. In practice, the number $m$ is usually limited in limited-motion sequences. Consequently, the computation of TIFM is fast in most cases. Furthermore, during a real matching process, some neighboring feature points of a feature point may have already been correctly matched in the matching processes for other neighborhoods, and thus these points have only one candidate. As a result, the actual average number of candidates will be smaller than $m$. Computation time can be reduced by incorporating more constraints. For example, if the fundamental matrix is known, the number of candidates $m$ can be significantly reduced. This not only reduces the computation time, but also improves the matching performance because of a reduced matching ambiguity. To give an indication about the execution speed, we have detected and matched the feature points for

the first 21 frames of the *medusa* sequence. TIFM spends a total of 213 seconds for detecting and matching feature points, while SIFT needs 285 seconds.

## 4.6 Conclusion

This chapter has presented a novel Texture-Independent Feature point Matching (TIFM) algorithm for tracking feature points over successive frames in a video or image sequence with limited image motion or pixel shifts. Our TIFM algorithm uses only a smoothness constraint for feature point matching. This constraint considers that all feature points in a local neighborhoods move with similar directions and magnitudes. By using this constraint, all feature points within a local neighborhood are collectively matched by maximizing the local motion smoothness, using a RANSAC-like process.

This chapter has also shown that the heuristic smoothness constraint can be converted into a quantitative criterion for feature point matching. This has enabled us to construct a concept for feature point matching based solely on the above smoothness constraint. This concept has not been presented in literature. A smoothness constraint is commonly used by other feature point matching algorithms. However, it is always applied in conjunction with other matching constraints such as the appearance constraint. The extensive experimental results have shown a significantly improved performance when matching and tracking feature points in video sequences with limited motion. Experimental results demonstrate that TIFM is able to track at least twice the number of points compared with SIFT, KLT and BM, with a comparable or higher accuracy.

Due to the single use of the smoothness constraint, TIFM provides two major advantages. The first advantage is that TIFM is robust to illumination changes because of the following four reasons. (1) TIFM is texture-independent, which gives the robustness to light changes. (2) The smoothness assumption is valid for most local neighborhoods, providing the robustness to camera motion and scene structure. (3) The RANSAC-like process prevents the smoothing over object boundaries. (4) TIFM achieves a good balance between the local and global matching. Neighborhood-based matching better constrains the matching process for ambiguous feature points when compared with the local algorithms, while it allows more complex global image motion when compared with the global algorithms. The second advantage is that TIFM is able to track a large number of feature points along a long sequence of frames, due to its good capability to handle the drift of the feature points.

The smoothness constraint assumes that feature points in a local neighborhood move with similar directions and magnitudes. However, this is a heuristic constraint that is only valid under specific conditions. For example, if a sequence contains very large non-translational motion, or the repetition ratio of the feature points is very low, the smoothness constraint is no longer valid, so that the proposed algorithm cannot work. However, as demonstrated by our experiments, TIFM works very well for all tested limited-motion sequences, which are frequently used in practice. It provides an excellent solution to feature point tracking in applications such as 3D reconstruction from video.

The proposed algorithm forms one of our major contributions of this thesis. With the algorithm, a large number of feature points can be tracked over a long sequence of frames, which is critical for an automated SaM on a long sequence. In the next chapter, we will

present another important contribution of this thesis, that is, partitioning a long sequence into subsequences by detecting critical configurations where the factorization-based 3D reconstruction degenerates.

CHAPTER 5

# Dividing long sequence for factorization-based SaM

*In Chapter 2, we have pointed out that a long sequence has to be divided into subsequences such that a sufficient number of feature points can be tracked for factorization-based SaM on individual subsequences. An automatic division of a long sequence into subsequences is essential for the proposed SaM system. This chapter proposes algorithms for dividing long sequences with the consideration of so-called critical configurations where the factorization method degenerates. First, we introduce the projective reconstruction and camera calibration algorithms that are used in this thesis. Second, we propose algorithms to detect the following critical configurations where the factorization method is not possible: (1) coplanar 3D points, (2) pure rotation of the camera, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. Third, a sequence-dividing algorithm is proposed to automatically divide a long sequence into subsequences such that a successful SaM can be obtained on each subsequence with a high confidence. Finally, experimental results on both synthetic and real sequences are presented to demonstrate the effectiveness of the proposed dividing algorithm for an automatic 3D reconstruction.*

## 5.1 Introduction

A long video sequence may comprise of thousands of frames, and has to be divided into subsequences such that a sufficient number of feature points can be tracked in individual subsequences for the factorization-based SaM on individual subsequences. As depicted in Fig. 5.1, this chapter presents our contribution on partitioning a long sequence into multiple subsequences for an automatic 3D reconstruction of long sequences using the factorization method. A number of factors need to be taken into account when dividing a long sequence.

In the following text, we discuss these factors and elaborate further on the approach followed in this chapter.

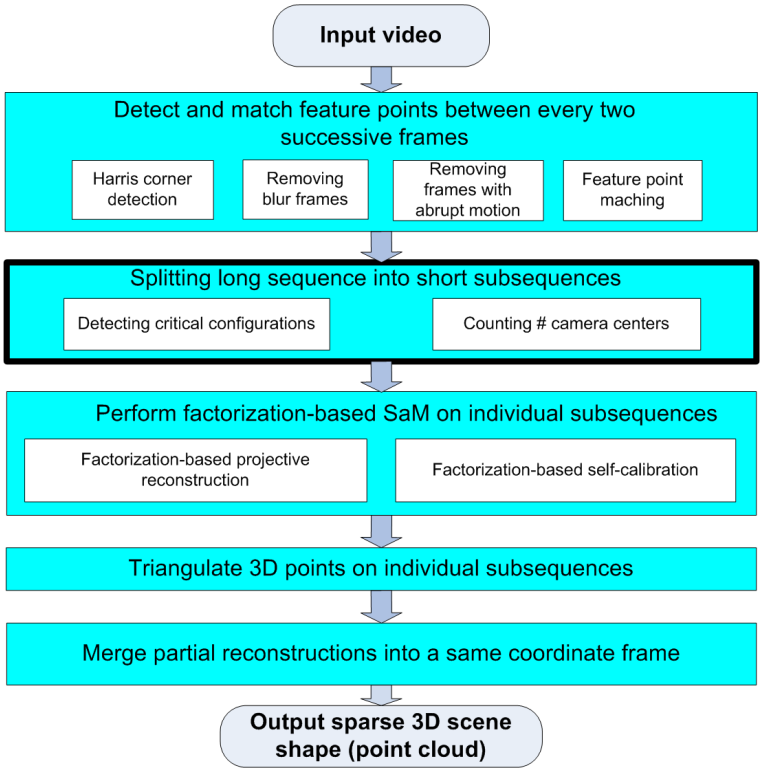



**Figure 5.1:** *Block diagram of the proposed 3D modeling system. This chapter presents our contribution on dividing a long sequence into multiple short sequences.*

The factorization-based 3D reconstruction method [20] used in this work requires that all feature points should occur in all frames of an image sequence. This is complicating in long sequences where scene content is changing over time and feature points disappear and emerge simultaneously. In order to track a sufficient number of feature points in the subsequence for 3D reconstruction, a long sequence has to be divided into multiple short subsequences. However, although short sequences usually have sufficient feature points, the camera disparity may be insufficient due to the limited number of images (e.g. the camera may undergo pure rotation), which may lead to the failure of the factorization-based 3D reconstruction. In contrast, long sequences usually have sufficient disparity. However, the number of tracked feature points is usually limited, which may also lead to a failure. Thus, a tradeoff has to be made between the number of the tracked feature points against the length of the individual subsequences.

Apart from the above tradeoff, there are other technical complications for applying the factorization-based 3D reconstruction to a long video sequence. The factorization method has other fundamental difficulties in handling special camera and scene configurations. More

specifically, Euclidean scene information can never be recovered by any algorithm under so-called *critical motions* and *critical surfaces* [20, 74]. For example, if all detected feature points are coplanar, or all cameras have the same center, projective reconstruction using the factorization method will not be possible. Such critical configurations where Euclidean reconstruction degenerates need to be appropriately handled while partitioning a long sequence.

Furthermore, for general applicability of our framework, the algorithm should be robust to varying scene contents and picture qualities. It has been found that 3D reconstruction is sensitive to noise and outliers, especially when the configuration is 'close' to a critical configuration [32]. The presence of one single outlier may deteriorate the whole SaM process. To enhance robustness, it is important that the impact of the noise and outliers on the 3D reconstruction process is measured and appropriately handled. This chapter aims at designing an algorithm for dividing long image sequences while considering the above-mentioned three main issues when applying the factorization method to long sequences, i.e., the tradeoff on the number of feature points, the critical configurations, and the impact of the noise and outliers.

The three issues have not been equally well discussed in literature. For dividing long sequences for factorization-based projective reconstruction, we have found related work [31], where a quantitative measure is proposed for dividing a long sequence based on measuring the number of the feature points, the homography error, and the distribution of the feature points. We have not found other research concerning dividing long sequences. Critical motions and surfaces have been thoroughly investigated in literature. Assuming that the focal lengths are the only unknown parameters, a complete categorization of critical motions is given in [74]. Ref. [32] extended this work by relaxing the constraints on the intrinsic parameters. The critical motions under different calibration constraints (zero skew, unit aspect ratio, vanishing principal point) are derived. Some particular critical configurations that frequently occur in practice are discussed. Despite the extensive literature study on the critical configurations, we found little work on detecting them. Related work on detection of the degenerate configurations for estimating the fundamental matrix is reported in [81]. With respect to the impact of the noise and outliers, we did not find any publication fitting in the framework of applying the factorization-based 3D reconstruction to a long video sequence.

The approach that we follow in this chapter is to jointly consider all three issues in the algorithm design for dividing long sequences. Moreover, our approach will be based on detecting critical configurations in advance so that the algorithm can handle them appropriately. Two aspects in our approach are different from existing literature. First, algorithms are proposed to detect the critical configurations resulting from (1) pure rotation of the camera, (2) coplanar 3D points, and (3) rotation around two camera centers, and (4) presence of outliers or excessive noise. Our detection is possible because the configurations in cases of (1), (2) and (4) will affect the rank of the scaled measurement matrix (SMM). Besides this, the number of camera centers in case of (3) will affect the number of independent rows of the SMM. Hence, by examining the rank and the row space of the SMM, we detect the above-mentioned critical configurations. The second contribution is that we propose an algorithm to divide a long image sequence, which considers the above-mentioned critical configurations and balances the number of the feature points and the length of the subsequences for a successful factorization-based SaM.

This chapter is divided as follows. Section 5.2 introduces the algorithms for factorization-

based SaM. Section 5.3 presents the proposed algorithms for detecting critical configurations and dividing long sequences. Section 5.4 presents the experimental results on synthetical and real sequences, and Section 5.5 concludes this chapter.

## 5.2 Factorization-based SaM

This section introduces the algorithm for factorization-based projective reconstruction. Using the notations introduced in Chapter 3, first, the algorithm for matrix factorization is introduced. Second, the projective reconstruction algorithm that is used in this framework is presented. Finally, the self-calibration algorithm for calibrating the parameters of cameras is addressed. The algorithms presented in this section are the fundamental algorithms for the framework. Our proposed algorithms for detecting critical configurations and dividing long sequences are based on these fundamental algorithms and exploit specific features of those to design our contributions.

### 5.2.1 Notations

This section recalls the notations that are introduced in Section 3.2.1. Some important notations about projective geometry from Section 3.2.1 are summarized as follows.

**Projective equation:**

$$\lambda_j^i \mathbf{x}_j^i = \lambda_j^i (u_j^i, v_j^i, 1)^T = \mathbf{P}^i \mathbf{X}_j, \tag{5.1}$$

where $\mathbf{X}_j = \beta_j (X_j, Y_j, Z_j, 1)^T$ are the homogenous coordinates of 3D point $j$, vector $\mathbf{x}_j^i = (u_j^i, v_j^i, 1)^T$ are the homogeneous coordinates of the 2D projection of 3D point $j$ in camera $i$, and $\mathbf{P}^i$ is the projection matrix of camera $i$, which can be represented by[1]

$$\mathbf{P}^i = \alpha_i \mathbf{K}_i \mathbf{R}_i [\mathbf{I} \ -\mathbf{C}_i]. \tag{5.2}$$

In the above equations, $\lambda_j^i$, $\alpha_i$ and $\beta_j$ are three scaling factors used by the homogenous presentation to make the left and right hand terms of the equation equal. The reader is referred to Section 3.2.1 for the definition of other parameters in the equations.

**Projective equation in matrix form:**

$$\mathbf{W}_s = \begin{pmatrix} \lambda_1^1 \mathbf{x}_1^1 & \dots & \lambda_n^1 \mathbf{x}_n^1 \\ \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{x}_1^m & \dots & \lambda_n^m \mathbf{x}_n^m \end{pmatrix} = \begin{pmatrix} \mathbf{P}^1 \\ \vdots \\ \mathbf{P}^m \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 \dots & \mathbf{X}_n \end{pmatrix} = \mathbf{P}\mathbf{X}, \tag{5.3}$$

where $\mathbf{W}_s \in \Re^{3m \times n}$ is called the Scaled Measurement Matrix (SMM), $\mathbf{P} \in \Re^{3m \times 4}$ is called the *Euclidean motion matrix* that contains the internal and external parameters of all cameras, and $\mathbf{X} \in \Re^{4 \times n}$ is called the *Euclidean shape matrix* that contains the coordinates of all 3D points.

[1] Unfortunately, we cannot use superscripts in all equations in this chapter, since we will use superscripts for iterations. For example, both $\mathbf{P}^i$ and $\mathbf{P}_i$ are used to represent the projection matrix of camera $i$.

As pointed out in Section 3.2.3, given a set of available image points $\mathbf{x}_j^i$, solving Eq. (5.3) without metric restriction on the camera and scene will yield a $\hat{\mathbf{P}}$ and a $\hat{\mathbf{X}}$ that differ from the Euclidean motion and shape matrices by an unknown projective transformation $\mathbf{H}$, satisfying

$$\lambda_j^i \mathbf{x}_j^i = \hat{\mathbf{P}}^i \hat{\mathbf{X}}_j = (\hat{\mathbf{P}}^i \mathbf{H}^{-1})(\mathbf{H}\hat{\mathbf{X}}_j) = \mathbf{P}^i \mathbf{X}_j. \tag{5.4}$$

The process to recover $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ is called projective reconstruction, and the process to recover $\mathbf{H}$ is referred to as Euclidean reconstruction. As discussed in Section 3.2.3, the accuracy of projective reconstruction can be measured by the re-projection error defined in Eq. (3.7). From Eqs. (3.7) and (5.1), we can compute the average re-projection error as

$$E_{proj} = \sqrt{\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left\{\left(u_j^i - \frac{\hat{\mathbf{P}}_{i(1)}^T \hat{\mathbf{X}}_j}{\hat{\mathbf{P}}_{i(3)}^T \hat{\mathbf{X}}_j}\right)^2 + \left(v_j^i - \frac{\hat{\mathbf{P}}_{i(2)}^T \hat{\mathbf{X}}_j}{\hat{\mathbf{P}}_{i(3)}^T \hat{\mathbf{X}}_j}\right)^2\right\}}{2 \times m \times n}}. \tag{5.5}$$

In the equation, $\hat{\mathbf{P}}_{i(1)}^T$, $\hat{\mathbf{P}}_{i(2)}^T$ and $\hat{\mathbf{P}}_{i(3)}^T$ are the row vectors of the $i$-th camera matrix $\hat{\mathbf{P}}_i$, and $\hat{\mathbf{X}}_j$ is the $j$-th 3D point.

### 5.2.2 Matrix rank-$r$ factorization

The factorization method relies on matrix factorization to reconstruct the projective motion matrix $\hat{\mathbf{P}}$ and the projective shape matrix $\hat{\mathbf{X}}$, which are related with the $\mathbf{P}$ and $\mathbf{X}$ by an unknown $4 \times 4$ projective transformation $\mathbf{H}$, as in Eq. (5.4). This subsection introduces the mathematical factorization method that factorizes an arbitrary matrix $\mathbf{W}$ into two matrices $\hat{\mathbf{P}}_r$ and $\hat{\mathbf{W}}_r$ of a rank $r$.

Let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ be the singular value decomposition of a matrix $\mathbf{W}$. The best approximating matrix $\hat{\mathbf{W}}_r$ to $\mathbf{W}$ under the Frobenius norm[2] with a rank $\leq r$ is computed by $\hat{\mathbf{W}}_r = \mathbf{U}\mathbf{\Sigma}_r\mathbf{V}^T$, where $\mathbf{\Sigma}_r$ is obtained from the diagonal matrix $\mathbf{\Sigma}$ by keeping the first $r$ largest singular values [74]. Furthermore, it is possible to split $\mathbf{\Sigma}_r$ into two diagonal matrices of rank $r$ that satisfy $\mathbf{\Sigma}_r = \mathbf{\Sigma}_r^{'}\mathbf{\Sigma}_r^{''}$. Thus, $\mathbf{W}$ can be approximated using the product of two rank-$r$ matrices as $\mathbf{W} \approx \hat{\mathbf{W}}_r = \hat{\mathbf{P}}_r\hat{\mathbf{X}}_r$, where $\hat{\mathbf{P}}_r = \mathbf{U}\mathbf{\Sigma}_r^{'}$ and $\hat{\mathbf{X}}_r = \mathbf{\Sigma}_r^{''}\mathbf{V}^T$. The above process of approximating $\mathbf{W}$ by two rank-$r$ matrices is referred to as **rank-$r$ factorization** in this thesis. Obviously, if the rank of $\mathbf{W}$ is larger than $r$, some nonzero singular values are discarded during factorization, which represents the inaccuracy of the approximation. Conversely, the accuracy of the rank-$r$ factorization can be measured using the 'rank-$r$-ness' of $\mathbf{W}$, which we define here as

$$\kappa_r = (1 - \sigma_{r+1}/\sigma_r) \times 100\%, \tag{5.6}$$

where $\sigma_r$ is the $r$-th largest singular value. In most cases, a large $\kappa_r$ implies that the rank of $\mathbf{W}$ is close to $r$, since $\sigma_{r+1}$ is negligible in this case.

In our proposal, $\kappa_r$ together with the re-projection error, which will be defined in the next subsection, are used to determine the rank of the SMM. Note that $\kappa_r$ alone does not give much confidence about the rank of the SMM. For example, when the rank of $\mathbf{W}$ is smaller than $r$, $\kappa_r$ may have a large value even though both $\sigma_{r+1}$ and $\sigma_r$ are very small.

[2]The Frobenius norm or the also called Hilbert-Schmidt norm of a $m \times n$ matrix $\mathbf{A}$ can be computed as $||\mathbf{A}|| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$.

### 5.2.3 Projective reconstruction using iterative minimization

The projection reconstruction algorithm used in this thesis attempts to recover $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ by minimizing the re-projection error defined in Eq. (5.5). This section introduces the used algorithm for projective reconstruction. From Eq. (5.5), we obtain the following two constraining equations for every $i, j$ combination to minimize the re-projection error, stating that

$$u_j^i - \frac{\hat{\mathbf{P}}_{i(1)}^T \hat{\mathbf{X}}_j}{\hat{\mathbf{P}}_{i(3)}^T \hat{\mathbf{X}}_j} = 0 \quad \text{and} \quad v_j^i - \frac{\hat{\mathbf{P}}_{i(2)}^T \hat{\mathbf{X}}_j}{\hat{\mathbf{P}}_{i(3)}^T \hat{\mathbf{X}}_j} = 0. \tag{5.7}$$

Based on the above two equations, we can formulate linear equations to solve for $\hat{\mathbf{X}}_j$ and $\hat{\mathbf{P}}_i$ by alternatively holding $\hat{\mathbf{P}}$ and $\hat{\mathbf{X}}$ constant, as proposed in the Weighted Iterative Eigen (WIE) algorithm [10]. The steps of the WIE algorithm are as follows.

S1: Assuming $\lambda_{ij} = 1$, factorize $\mathbf{W}_s$ into two rank-4 matrices $\hat{\mathbf{P}}^{(0)}$ and $\hat{\mathbf{X}}^{(0)}$ (the superscript $(0)$ denotes the iteration count) using rank-4 factorization.

S2: Given $\hat{\mathbf{P}}^{(k)}$ and $\hat{\mathbf{X}}^{(k)}$ (the superscript $(k)$ denotes the iteration count), update $\hat{\mathbf{X}}_j^{(k+1)}$ $(\forall j = 1, \ldots, n)$ by minimizing the re-projection error of the $2m$ number of linear equations derived from Eq. (5.7).

S3: Given $\hat{\mathbf{P}}^{(k)}$ and $\hat{\mathbf{X}}^{(k+1)}$, update $\hat{\mathbf{P}}_i^{(k+1)} (\forall i = 1, \ldots, m)$ by minimizing the re-projection error of the $2n$ number of linear equations derived from Eq. (5.7).

S4: Iterate Steps S2 and S3 until the projective depths $\lambda_{ij}^{(k+1)}$converge[3] or the maximum number of iterations is reached.

The derivation of the linear equations from Eq. (5.5) in Steps S2 and S3 can be found in [10]. In the above algorithm, $\mathbf{W}_s$ is factorized into two rank-4 matrices. However, as discussed in Section 5.2.2, we can factorize $\mathbf{W}_s$ into two matrices of an arbitrary rank $r$, which is smaller than or equal to the rank of $\mathbf{W}_s$. In theory, the rank of $\mathbf{W}_s$ is maximally 4. However in practice, the rank can be larger due to noise and outliers. In that case, we obtain a $3m \times r$ matrix $\hat{\mathbf{P}}^{(0)}$ and a $r \times n$ matrix $\hat{\mathbf{X}}^{(0)}$ in Step S1, which is refined later in Steps S2, S3 and S4. We refer to such process as **rank-$r$ iteration**. We have observed that the rank-$r$ iteration converges quickly if $\mathbf{W}_s$ is indeed of rank $r$, while the iteration may not converge if the rank of $\mathbf{W}_s$ differs from $r$. This property is exploited in our proposed algorithm to determine the rank of the $\mathbf{W}_s$, as will be discussed in Section 5.3.

### 5.2.4 Factorization-based self-calibration

Given the projective motion and shape ($\hat{\mathbf{P}}_i$, $\hat{\mathbf{X}}_j$) obtained in the previous section, we need to compute the Euclidean motion and shape ($\mathbf{P}_i$, $\mathbf{X}_j$). To do this, the $4 \times 4$ transformation matrix $\mathbf{H}$ that relates the projective motion and shape matrices to the Euclidean motion and shape matrices needs to be computed from

$$\mathbf{P}_i = \hat{\mathbf{P}}_i \mathbf{H} \quad \text{and} \quad \mathbf{X}_j = \mathbf{H}^{-1} \hat{\mathbf{X}}_j. \tag{5.8}$$

[3]The projective depths $\lambda_{ij}^{(k+1)} = \hat{\mathbf{P}}_{i(3)}^{T(k+1)} \hat{\mathbf{X}}_j^{(k+1)}$ are considered converged if the relative change of the projective depths between two consecutive iterations is smaller than a pre-determined threshold.

This section briefly introduces the factorization-based self-calibration technique proposed in [20] to compute the matrix $\mathbf{H}$.

Let us now present the algorithm. We represent $\mathbf{P}_i$ by

$$\mathbf{P}_i = [\mathbf{M}_i \ \mathbf{T}_i], \tag{5.9}$$

where

$$\mathbf{M}_i = \alpha_i \mathbf{K}_i \mathbf{R}_i = [\mathbf{m}_{xi} \ \mathbf{m}_{yi} \ \mathbf{m}_{zi}]^T \ \text{ and } \ \mathbf{T}_i = -\alpha_i \mathbf{K}_i \mathbf{R}_i \mathbf{C}_i = [T_{xi}, T_{yi}, T_{zi}]^T. \tag{5.10}$$

Knowing $\hat{\mathbf{P}}$, $\hat{\mathbf{X}}$ and $\mathbf{W}_s$, the purpose of self-calibration is to determine $\mathbf{H}$ by imposing metric constraints on $\mathbf{R}_i$ and $\mathbf{K}_i$ (e.g. orthogonality of camera axes). With the assumption that the principal point is at the origin, the aspect ratio equals unity and the skew equals zero, it can be verified (Appendix A) that

$$|\mathbf{m}_{xi}|^2 = |\mathbf{m}_{yi}|^2 \quad \text{and} \quad \mathbf{m}_{xi}^T \mathbf{m}_{yi} = \mathbf{m}_{xi}^T \mathbf{m}_{zi} = \mathbf{m}_{yi}^T \mathbf{m}_{zi} = 0. \tag{5.11}$$

Thus, we obtain 4 linear constraining equations from each camera on the elements of $\mathbf{M}_i \mathbf{M}_i^T$, which are referred to as *calibration constraints*. The matrix product $\mathbf{M}_i \mathbf{M}_i^T$ is introduced as a construct to solve for $\mathbf{H}$, which will be reworked to a set of equations to be introduced below.

We will show now how the calibration constraints are used to solve for $\mathbf{H}$. Let $\mathbf{H} = [\mathbf{A} \ \mathbf{B}]$, where $\mathbf{A}$ is a $4 \times 3$ matrix and $\mathbf{B}$ is a 4-element vector, we obtain $\mathbf{M}_i = \hat{\mathbf{P}}_i \mathbf{A}$ and $\mathbf{T}_i = \hat{\mathbf{P}}_i \mathbf{B}$ from Eq. (5.8). Consequently, for the construct $\mathbf{M}_i \mathbf{M}_i^T$, we obtain

$$\mathbf{M}_i \mathbf{M}_i^T = \hat{\mathbf{P}}_i \mathbf{A} \mathbf{A}^T \hat{\mathbf{P}}_i^T = \hat{\mathbf{P}}_i \mathbf{Q} \hat{\mathbf{P}}_i^T. \tag{5.12}$$

As can be observed from Eq. (5.12), the 4 linear constraints from Eq. (5.11) are transferred to 4 linear constraints on the 10 elements of the $4 \times 4$ symmetric matrix $\mathbf{Q}$. For $m$ cameras, the linear least squares solution of $\mathbf{Q}$ can be computed from $4m + 1$ linear equations[4]. Matrix $\mathbf{A}$ can then be computed by a rank-3 decomposition of $\mathbf{Q}$. We refer to [20] for details on how $\mathbf{B}$ is solved. The summary of this algorithm is found in Appendix A.

Obviously, at least 9 independent linear equations are required for solving the 10 elements of $\mathbf{Q}$ (with one element fixed to unity). This means that we require at least 3 distinct cameras for factorization-based self-calibration under the above-mentioned calibration assumptions, as will be shown below.

*Proof:* From Eqs. (5.8), (5.9) and (5.10), we can derive $\hat{\mathbf{P}}_i = \mathbf{M}_i[\mathbf{I} \ -\mathbf{C}_i]\mathbf{H}^{-1}$. Suppose cameras $\mathbf{P}_p$ and $\mathbf{P}_q$ have the same center, i.e., $\mathbf{C}_p = \mathbf{C}_q$, it can be verified that

$$\hat{\mathbf{P}}_p = \mathbf{M}_p \mathbf{M}_q^{-1} \hat{\mathbf{P}}_q. \tag{5.13}$$

When substituting Eq. (5.13) into $\mathbf{M}_p \mathbf{M}_p^T = \hat{\mathbf{P}}_p \mathbf{Q} \hat{\mathbf{P}}_p^T$, we obtain $\mathbf{M}_q \mathbf{M}_q^T = \hat{\mathbf{P}}_q \mathbf{Q} \hat{\mathbf{P}}_q^T$. Thus, $\mathbf{P}_p$ and $\mathbf{P}_q$ actually provide the same set of calibration constraints. □

In practice, due to the inaccuracy of the calibration constraints (e.g. the principal point is not exactly located at the origin), errors in the measurements and the degeneracy of the configurations, more cameras are usually required for self calibration. One contribution of this chapter is that an algorithm is proposed to count the number of distinct camera centers to ensure sufficient calibration constraints. This counting algorithm is necessary for robustly dividing a long sequence, and will be presented in Section 5.3.

[4] One extra equation is obtained by requiring that the scale factor of the first camera $\alpha_1$ equals unity, which means $\mathbf{m}_{z1} \cdot \mathbf{m}_{z1} = 1$.

## 5.3 Proposed algorithm

This section presents the proposed dividing algorithm for partitioning a long sequence into multiple subsequences for partial reconstruction using the factorization method. The main steps of the proposed algorithm are depicted in Fig. 5.2. From the figure, we observe that the proposed dividing algorithm recursively divides a long sequence of $l$ frames into subsequences. The dividing algorithm is based on three sub-algorithms, i.e., forward dividing, backward dividing and bi-directional dividing, to determine the best subsequences. The key aspect of all three sub-algorithms is that the four critical configurations (discussed in Section 5.1) are detected during dividing, such that a successful reconstruction on individual subsequences can be obtained with a high confidence. The only difference is in the directions how feature points are tracked. The three sub-algorithms for forward, backward and bi-directional dividing are equally important in the sense that we attempt to make the division robust for reconstruction. When dividing fails on one direction, it is still possible to divide the long sequence from other directions, so that the robustness is improved. The forward dividing algorithm will be presented in Section 5.3.5.

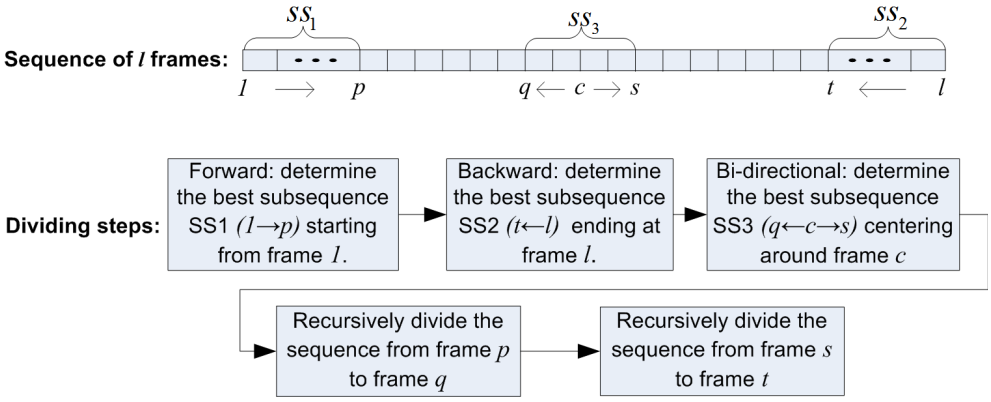



**Figure 5.2:** *Algorithm steps to recursively divide a long sequence into subsequences. The dividing is based on three sub-algorithms, i.e., forward dividing, backward dividing and bi-directional dividing. The only difference between the three sub-algorithms is that feature points are tracked in different directions.*

### 5.3.1 Discussion on four critical configurations to be detected

In the following, each of the 4 critical configurations presented in Section 5.1 is discussed briefly. They are referred to as C1-C4.

C1. *Coplanar 3D points*: From Eq. (5.3), we derive that $\mathbf{W}_s$ is of rank 3, since there are only 3 independent columns in $\mathbf{X}$, and consequently in $\mathbf{W}_s$.

C2. *Pure rotation*: $\mathbf{W}_s$ is of rank 3, since there are only 3 independent rows in $\mathbf{W}_s$, as shown from the following proposition.

**Proposition 1:** Let $\mathbf{v}_i = [\lambda_{i1}\mathbf{x}_{i1}, \ldots, \lambda_{in}\mathbf{x}_{in}]$ be the partial SMM for camera $i$, the 3 row vectors of $\mathbf{v}_i$ are linearly dependent on the 3 row vectors of $\mathbf{v}_j$ iff cameras $i$ and $j$ have the same center.

*Proof*: From Eq. (5.3), we have:

$$\begin{aligned} \mathbf{v}_i &= \alpha_i \mathbf{K}_i \mathbf{R}_i [\beta_1(\mathbf{X}_1 - \mathbf{C}_i), \ldots, \beta_n(\mathbf{X}_n - \mathbf{C}_i)], \\ \mathbf{v}_j &= \alpha_j \mathbf{K}_j \mathbf{R}_j [\beta_1(\mathbf{X}_1 - \mathbf{C}_j), \ldots, \beta_n(\mathbf{X}_n - \mathbf{C}_j)]. \end{aligned}$$

If $\mathbf{C}_i = \mathbf{C}_j$, then $\mathbf{v}_j = (\mathbf{K}_j \mathbf{R}_j \mathbf{R}_i^T \mathbf{K}_i^{-1} \alpha_j / \alpha_i)\mathbf{v}_i = \mathbf{M}\mathbf{v}_i$, where $\mathbf{M}$ is a $3 \times 3$ non-singular matrix. Hence, the three rows of $\mathbf{v}_j$ are linear combinations of the three rows of $\mathbf{v}_i$. If $\mathbf{C}_i \neq \mathbf{C}_j$, representing $\mathbf{v}_j$ by $\mathbf{M}\mathbf{v}_i$ is generally not possible. □

C3. *Rotation around two camera centers*: The $3m \times 4$ $\mathbf{W}_s$ is of rank 4 in this case, because from Proposition 1 we know that there will be more than three independent rows in $\mathbf{W}_s$. For a successful factorization-based self-calibration, we need at least 3 distinct camera centers, as discussed in Section 5.2.4. Since the rank of $\mathbf{W}_s$ stays at 4 in this case, we cannot detect this critical configuration using rank analysis.

C4. In the presence of excessive noise or outliers, the rank of the SMM will deviate from 3 or 4, and will be generally larger than 4. Consequently, large re-projection errors and low rank-$r$-ness are expected for both the rank-3 and the rank-4 iterations.

Counting of the number of distinct camera centers will be discussed in Section 5.3.2, while the dection of configurations C1, C2, C3 and C4 will be discussed in Section 5.3.3.

### 5.3.2 Algorithm for Counting distinct Camera Centers (ACCC)

As discussed in Section 5.2.4, at least three distinct cameras are required for the factorization-based self-calibration (C3 in Section 5.3.1). Depending on the scene configuration, two cameras that are positioned close to each other may not give independent constraints for self-calibration. Therefore, we propose to measure the 'closeness' between two cameras, using the normalized distance between the camera and object as a metric. The metric $D$ is defined as

$$D = \frac{\|\mathbf{X}_a - (\mathbf{C}_i + \mathbf{C}_j)/2\|}{\|\mathbf{C}_i - \mathbf{C}_j\|}, \tag{5.14}$$

where $\mathbf{C}_i$ and $\mathbf{C}_j$ are the two camera centers, $\mathbf{X}_a$ is the geometry center of the set of 3D points. Apparently, the smaller the $D$, the closer the object with respect to the cameras, and thus the more distinct the cameras are located. Thus, for a successful factorization-based Euclidean reconstruction, we need to make sure that the closeness metric values $D$ for at least 3 camera pairs related to at least 3 camera centers, are below a certain threshold. However, the problem is that $D$ cannot be computed prior to the Euclidean reconstruction. As a solution, Proposition 1 suggests that the closeness between two cameras $\mathbf{C}_i$ and $\mathbf{C}_j$ can be measured by the difference between $\mathbf{v}_i$ and $\mathbf{v}_j$, which leads to the following ACCC algorithm. The steps of the proposed ACCC algorithm are as follows.

S1. Randomly pick a $\mathbf{v}_i = \alpha_i \mathbf{K}_i \mathbf{R}_i [\beta_1(\mathbf{X}_1 - \mathbf{C}_i), \ldots, \beta_n(\mathbf{X}_n - \mathbf{C}_i)]$ from $\mathbf{W}_s$.

S2. Check if the row space of $\mathbf{v}_j(\forall j \neq i)$, denoted $RS(\mathbf{v}_j)$, can be spanned by $RS(\mathbf{v}_i)$. This can be done by checking if the maximum angle deviation between the individual row vectors of $\mathbf{v}_j$ and their corresponding projections in $RS(\mathbf{v}_i)$ is below a threshold. The maximum deviation $d$ is found by

$$d = \arcsin(\max_{k=1}^{3}(\|\mathbf{v}_{j(k)}^T - \mathbf{J}\mathbf{v}_{j(k)}^T\|/\|\mathbf{v}_{j(k)}^T\|)) \times 180/\pi, \tag{5.15}$$

where $\mathbf{J} = [\mathbf{v}_i^T(\mathbf{v}_i\mathbf{v}_i^T)^{-1}\mathbf{v}_i]$ is the projection matrix that projects a vector into $RS(\mathbf{v}_i)$, and $\mathbf{v}_{j(k)}^T$ is the $k$-th row vector of $\mathbf{v}_j$.

S3. If $d$ is smaller than a given threshold $T_d$, cameras $i$ and $j$ are considered to have the same center.

S4. Repeat the above steps until all cameras are clustered with distinct camera centers.

### 5.3.3 Algorithm for Detecting Critical Configurations (ADCC)

In Section 5.3.1, we have analyzed each of the four critical configurations and have derived that the rank and the row space of the SMM are constrained by the four configurations. This implies that an *a-priori* analysis of the rank and the row spaces of the SMM can lead to the detection of those configurations. As discussed in Section 5.2.2, the rank of the SMM can be estimated using the rank-$r$-ness of the SMM as defined in Eq. (5.6), together with the re-projection error as defined in Eq. (3.7). The row space of the SMM can be analyzed using the ACCC algorithm for counting distinct camera centers. Below, the steps of the ADCC algorithm for detecting critical configurations are presented.

S1. Perform both the *rank-3* and the *rank-4* iterations to factorize the SMM into matrices of rank 3 and 4, as described in Section 5.2.3. If both iterations fail to converge considering both the rank-$r$-ness and the re-projection error, we conclude that the the rank of the SMM is larger than 4 and the measurements contain either outliers or excessive noise. If both iterations converge, or only the rank-3 iteration converges, we conclude that either the 3D points are coplanar or the camera undergoes pure rotation. If only the rank-4 iteration converges, we proceed to Step S2.

S2. Count the number of camera centers using the ACCC algorithm. If there are more than two camera centers, we proceed to further processing.

The essence of the above algorithm is that we require a high rank-4-ness of the SMM, more than two camera centers, and a small re-projection error for a successful factorization-based 3D reconstruction. The configurations that do not satisfy these necessary conditions are considered degenerate and should be omitted for subsequent processing.

### 5.3.4 Discussion on the ADCC algorithm

The proposed ADCC algorithm is based on matrix rank analysis for critical configuration detection. In the scientific community, rank estimation is considered not sufficiently robust. We have already discussed that noise and outliers may change the rank of the SMM. In specific cases, it will be difficult to distinguish the real causes for a specific rank of the SMM,

which may lead to false detections by the proposed ADCC algorithm. This section provides an analysis on the possible false detections of the proposed ADCC algorithm. Afterwards, it will be argued that the lack of the theoretical validation of the algorithm is compensated by considering that the possible false detections occur only rarely. Furthermore, we compare the ADCC algorithm with other degeneracy detection algorithms in handling such situations.

There are mainly two flaws that may lead to a false detection by ADCC.

(1) As pointed out in [59], there is no theoretical justification that the iterative projective reconstruction methods such as those in [27, 51] will converge to sensible results, even when the data is free of noise. This also holds for the WIE algorithm that is used in this thesis, where the iterations may not converge to useful results with or without noise. Consequently, relying on the results of WIE iterations for determining the rank of the SMM may not be trustworthy.

(2) The presence of the noise or outliers may completely mask the degeneracy. For example, when outliers are present in coplanar 3D points, the 'coplanar points' may become non-coplanar, so that the critical configuration cannot be detected.

Due to the above two limitations, ADCC may result in the following two false detections.

(a) The SMM is considered as being 'noisy' (SMM is detected to have a rank$> 4$). However, the actual rank is 3 or 4, when WIE does not converge (Flaw 1).

(b) The rank of the SMM is detected as 4. However, the actual rank is 3, but is raised to 4 due to the presence of noise or outliers (Flaw 2).

.

The above two cases clearly indicate the limitations of the proposed ADCC algorithm. Fortunately, we have found that the occurrence of those limitations is also limited.

Despite the lack of a theoretical justification, we argue that the probability that Case (a) occurs is low because of the good convergence of the WIE algorithm, as will be demonstrated by the experimental results in Section 5.4. For the probability that the rank of the SMM increases from 3 to 4 as in Case (b), we do not have a statistical measurement. However, no existing method from literature is able to avoid such a false detection, because degeneracy detection metrics in most publications are presented with a scoring percentage lower than $100\%$. The following two aspects provide a justification of the ADCC algorithm. First, similar to curve fitting where a high-order curve cannot be well fit by a low-order curve, it is not possible to well approximate a high-rank SMM using a lower-rank matrix. Thus, as long as the rank-$r$ iteration produces a small re-projection error, we can safely conclude that the rank of the SMM is maximally $r$. Second, a high rank-$r$-ness $\kappa_r$ together with a small re-projection error further increases the confidence that the rank of the SMM is truly $r$, since $\sigma_{r+1}$ is negligible compared with $\sigma_r$. In the cases that both iterations converge to small re-projection errors, the lower rank is selected, as indicated in Step S1 of Section 5.3.3.

Let us now briefly discuss the relation between the proposed algorithm and other degeneracy detection metrics from literature. As discussed in [34, 80], degeneracy can be detected using a model-checking approach, where the best geometric model is selected using some

scoring criteria considering both the *goodness of the fit* and the *model complexity*. A good model should not only produce a small residual error, but also should have a low complexity. In the following, the relation between ADCC and the model-checking approach is discussed.

In our experiments, the G-AIC [34] of Kanatani is used for performance evaluation. The G-AIC is used for selecting the best model that fits to a given data, such that the selected model not only produces a small residue, but also should have a low complexity. The G-AIC is computed by

$$\text{G-AIC} = \hat{J} + 2(Nd + p)\varepsilon^2, \quad \text{with} \quad \varepsilon^2 = \frac{\hat{J}}{cN - p}, \tag{5.16}$$

where $\hat{J}$ is the re-projection error, $N$ is the number of data measurements (e.g. the number of feature points), $d$ is the dimension of the model, $p$ is the number of the parameters of the parameterized model, $c$ is the number of the constraints provided by one observed data sample, and $\varepsilon^2$ is the estimated noise level. Eq. (5.16) implies that a good model should not only produce small residual $\hat{J}$, but also should have a low complexity ($d$ and $p$). Note that $d$ and $p$ are the model parameters we can control during the design, whereas $\varepsilon^2$ is the noise level that is inherent in the data.

Comparing with ADCC, $\hat{J}$ is equivalent to the re-projection error $E$, which describes how well the model fits to the data. However, the second term of the right-hand side of Eq. (5.16), which describes the model complexity, is simply represented by the rank of the SMM in ADCC. Furthermore, instead of defining a score for evaluating the goodness of the fit and the model complexity, ADCC employs hard thresholds for evaluating the goodness and the complexity. That is, *a low-rank approximation is preferred as long as the resulting re-projection error is below a given threshold*. There are two reasons that the geometric model selection criteria such as G-AIC cannot be directly used in our problem. (1) The WIE algorithm does not guarantee that the rank-4 iteration will always produce a smaller re-projection error than the rank-3 iteration. (2) Though it was demonstrated that G-AIC works well for curve fitting, the weighting between the residual and the model complexity in G-AIC may not be optimal for our problem. Experimental results will be presented in Section 5.4.

### 5.3.5 Algorithm for Dividing Long image Sequence (ADLS)

As illustrated in Fig. 5.2, we propose an ADLS algorithm to recursively divide a long sequence into multiple subsequences, based on three sub-algorithms to determine the best subsequence from different directions. As has also been indicated, the three sub-algorithms are for determining: (1) the best subsequence starting from Frame $i$ (forward dividing), (2) the best subsequence ending at Frame $i$ (backward dividing), and (3) the best subsequence centering around Frame $i$ (bi-directional dividing). Although three sub-algorithms differ in the directions of feature point tracking, their principle is the same, i.e., to detect critical configurations during dividing such that a successful factorization-based reconstruction can be obtained on individual subsequences with a high confidence. This section presents only the forward-dividing sub-algorithm. The other two sub-algorithms for the cases (2) and (3) are basically identical, and are not discussed further.

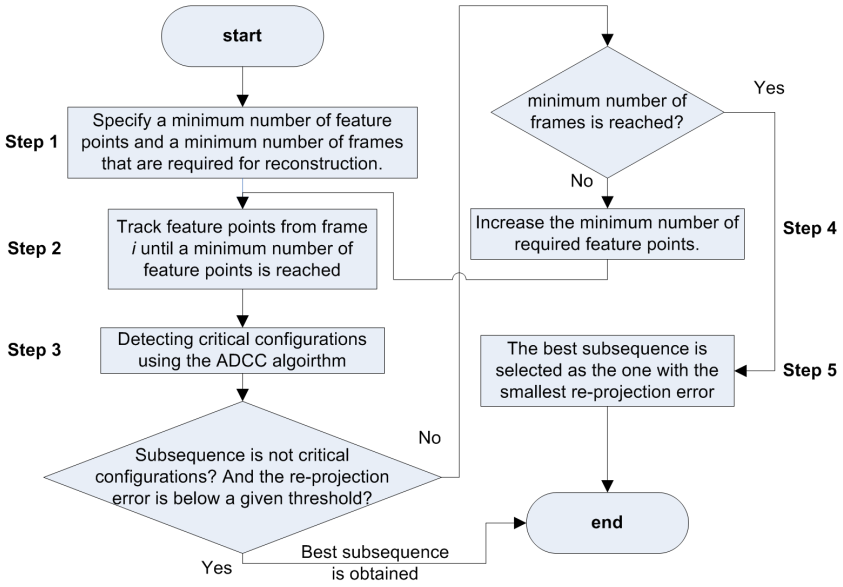



**Figure 5.3:** *Steps of the forward dividing algorithm to determine the best subsequence starting from Frame $i$.*

---

**Algorithm 5.3.1:** FORWARDDETERMINEBESTSUBSEQ()

---

**comment:** Determining best subsequence starting from Frame $i$.

Step 1. Specify a minimum number of feature points and a minimum number of frames that are required for reconstruction.

Step 2. Track feature points from Frame $i$ until a minimum number of feature points is reached (the longer the tracking, the fewer the feature points).

Step 3. Perform the projective reconstruction as described in Section 5.2.3 and compute parameters $\kappa_3$, $\kappa_4$, $E_3$, $E_4$ and $\#cc$. If $E_4$ is smaller than $E_3$ and smaller than a given threshold, $\kappa_4$ is larger than $\kappa_3$ and larger than a given threshold, and $\#cc$ is larger than 3, the best subsequence is obtained. Otherwise, go to Step 4.

Step 4. Increase the minimum number of required feature points, and repeat Steps 2 and 3, until the best subsequence is determined, or the minimum number of frames is reached.

Step 5. If no best subsequence is selected, select the subsequence with the minimum $E_4$ as the best subsequence.

---

As discussed in Section 4.1, 3D reconstruction is sensitive to noise and inaccuracy, es-

pecially when the configuration is close to a 'critical' case. For example, if cameras are positioned with small distances in between, there will be ambiguities for determining the absolute quadric [74]. In that case, we will need more feature points and more frames (camera disparity) to constrain the solution. The challenge here is that we cannot obtain a large number of feature points and a large number of images at the same time, since the number of the tracked points decrease with the number of frames. The numbers of required feature points and images depend on a number of factors such as scene shape, camera placement and noise level, which is referred to as 'criticalness' in this thesis.

From the proposed ADCC algorithm, it is observed that the criticalness of a configuration can be measured using metrics $\kappa_3$, $\kappa_4$, $E_3$, $E_4$ and $\#cc$, where $\#cc$ is the number of distinct camera centers. This leads to the proposed forward-dividing algorithm that is shown in Fig. 5.3. The steps of the proposed algorithm are summarized in Alg. 5.3.1.

It is important to note that the proposed ADLS algorithm provides only necessary conditions for Euclidean reconstruction, because the proposed ADCC algorithm is not able to detect other critical configurations than C1-C4 in Section 5.3.1. Consequently, not all obtained subsequences can yield a successful Euclidean reconstruction using the factorization method.

## 5.4 Experimental results

We have tested the ADCC algorithm on both the synthetic and the real sequences shown in Fig. 5.4. In the experiments, we assume that the principal point is at the origin, the aspect ratio equals unity and the skew is zero. Only the focal lengths of the cameras can differ from each other.

### 5.4.1 Detecting pure rotation and coplanar 3D points

Fig. 5.4(a) shows a synthetic house that is used for our experiments, where we observe 15 3D points in general positions and 10 cameras pointing towards the house. Knowing the coordinates of all 15 points and the parameters of all 10 cameras, we project them onto the 10 cameras. Afterwards, Gaussian noise is added to each coordinate of the 2D projections. More specifically, Gaussian noise of 4 different magnitudes is added to each coordinate of the 2D projections. The 4 standard deviations $v$ of the Gaussian noise are set to $0.002w$, $0.001w$, $0.0005w$ and $0.0002w$, where $w = 14$ is the width of the synthetic image. To simulate the influence of the outliers, we assume that $1\%$ of the feature points are outliers, of which the magnitude is computed as $0.05w$.

Euclidean reconstruction is then performed on the noisy 2D projections. The accuracy $A$ of the reconstruction is measured by the relative difference between the reconstructed focal lengths and the preset focal lengths, and is computed by

$$A = 1/m \sum_{i=1}^{m} (|f^i - s \times f_r^i| / f^i) \times 100\%, \tag{5.17}$$

where $f^i$ is the preset focal length for the $i$-th camera, $f_r^i$ is the reconstructed focal length, and $s$ is the scale factor computed as $s = (\sum_{i=1}^{m} f^i)/(\sum_{i=1}^{m} f_r^i)$. Note that the computation

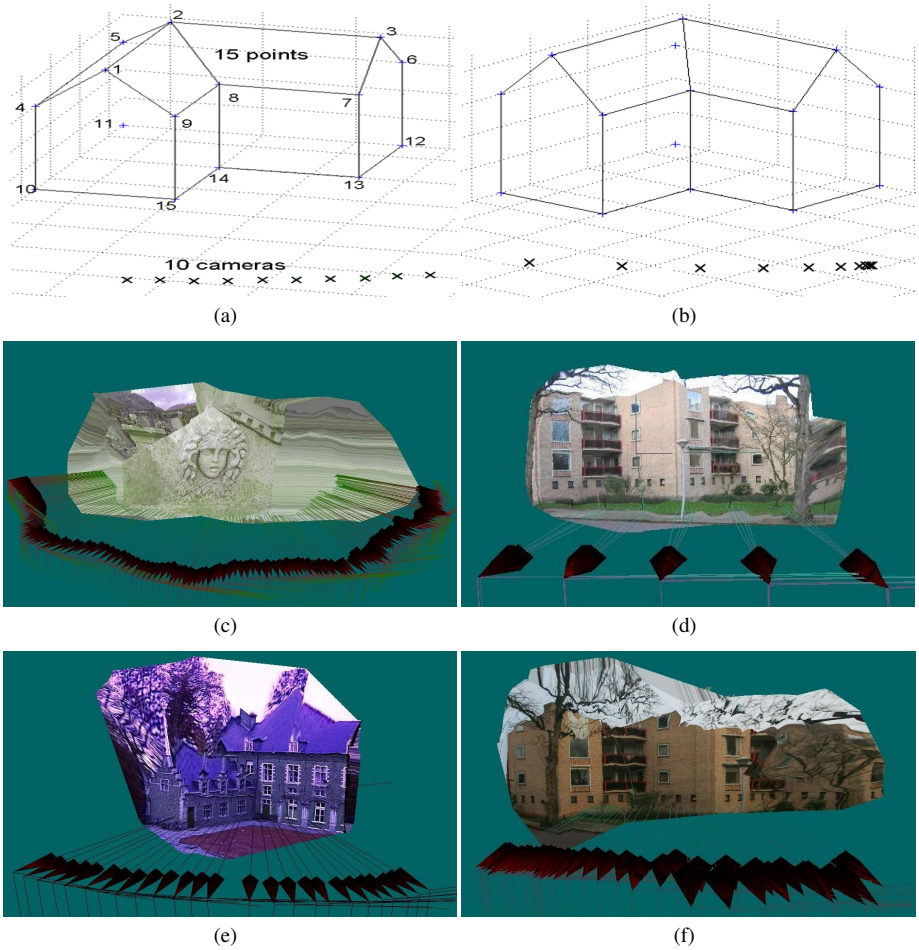



**Figure 5.4:** *Test sequences:* ***(a)*** *synthetic house with 10 cameras equally spaced;* ***(b)*** *synthetic house with 10 cameras positioned with varying distances;* ***(c)*** medusa *with 195 images from [3];* ***(d)*** house1 *with 20 images taken from 5 viewpoints (4 images for each viewpoint);* ***(e)*** castle *with 26 images from [3];* ***(f)*** house2 *with 38 images taken from slightly changing viewpoints.*

of $A$ is only possible for synthetic sequences where the actual focal lengths of the cameras are given, and when the 3D reconstruction converges. If reconstruction does not converge, the accuracy $A$ is assigned one of the following values: 'D' when the configuration is detected as 'degenerate', 'N' when the configuration is detected as 'noisy'. For real sequences, we cannot compute the accuracy using Eq. (5.17), since we do not know the actual focal lengths of the cameras. Thus, when the configuration is not detected as degenerate, Success (S) or Fail (F) of the 3D reconstruction (judged by visual inspection of the reconstruction results) is assigned for the accuracy.

Table 5.1 shows the experimental results of the ADCC algorithm for both the synthetic and real sequences. The 'X' in the table means that the value is not defined, since the configuration is detected as either noisy or degenerate.

Rows *b0-b_o* and *c0-c_o* of Table 5.1(a) show the results of the detection of pure rotation and coplanar 3D points in the presence of noise and outliers. As discussed in Section 5.3, if all 3D points are coplanar, or all cameras have the same center, the SMM will be of rank 3. From rows *b0-b4* and *c0-c4* of Table 5.1(a), we observe that the rank-3 iteration converges with a small $E_3$ and a large $\kappa_3$ in all experiments. This is as we expected. Thus, the proposed ADCC algorithm successfully detects all degenerate configurations, resulting from pure rotation and coplanar 3D points in the presence of noise of varying levels.

Rows *b_o* and *c_o* represent the cases where outliers occur. In experiment *b_o*, the rank of the SMM remains at 3, despite the presence of outliers, which leads to the convergence of the rank-3 iteration. Row *c_o* represents a case of false detection, where the rank of the SMM increases from 3 to 4 due to the presence of outliers, which leads to the convergence of the rank-4 iteration. Thus, outliers hide the degeneracy and lead to the failure of the proposed algorithm. Fortunately, as discussed in C4 in Section 5.3, outliers can be easily removed using, for example, the epipolar constraint. Thus, such false detection will rarely occur in practice. Rows *h4, m3* and *m4* in Table 5.1(b) correspond to the cases of pure rotation for real sequences, which are also successfully detected by the ADCC algorithm.

Table 5.1(b) also shows the G-AIC scores that are computed using Eq. (5.16) with the following parameters: $N = nm$ ($n$ points tracked along $m$ images), $d = r$ (rank of the SMM), $p = m + n$ (number of projective depths) and $c = 1$ (one point provides only one constraint). As observed from Table 5.1(b), the G-AIC scores for rank-4 iterations are consistently smaller than the rank-3 iterations, which should not be the case for *h3, h4, m3* and *m4* in Table 5.1(b). The weight between the residual error and the model complexity in G-AIC needs to be optimized for our case. Furthermore, as shown in rows *b3-b_o* and *c1-c4* in Table 5.1(a), the rank-4 iteration does not always produce smaller re-projection errors than the rank-3 iteration. This prevents a direct use of G-AIC in our case.

Summarizing, the experimental results show that the proposed ADCC algorithm is able to successfully detect all degenerate configurations for synthetic sequences, resulting from pure rotation and coplanar 3D points in the presence of noise of varying levels. Furthermore, the proposed algorithm also applies well to the real sequences for degeneracy detection. False detection occurs once for the synthetic sequence when the configuration is degenerate and outliers are present. However, as indicated above, outliers can be easily removed using constraints such as the epipolar constraint. Thus, the false detection by the ADCC algorithm rarely occurs in practice, as observed from the experimental results for the real sequences.

### 5.4.2 Counting distinct camera centers

The ACCC algorithm proposed in Section 5.3.2 is based on the Proposition 1, which suggests that the maximum angle deviation $d$ (defined in Eq. (5.15)) between $\mathbf{v}_i$ and $\mathbf{v}_j$ will be small if the camera centers $\mathbf{C}_i$ and $\mathbf{C}_j$ are close (with a large closeness $D$ defined in Eq. (5.14)). In this section, we first verify the validity of this proposition, and then apply ACCC to count the camera centers.

Figs. 5.5 shows the $d$-$D$ curves obtained in our experiments for both synthetic and real

**Table 5.1:** *Results of detecting critical configurations by ADCC.* ***(a)****: results on synthetic data.* ***(b)****: results on real data. Labeling in the table is as follows. a0-a_o: results on counting the number of camera centers; b0-b_o: results on detection of pure rotation; c0-c_o: results on detection of coplanar 3D points; '0' in 'a0' refers to noise-free data, '1' in 'a1' means that the std. dev. $\upsilon$ of the noise $\upsilon = 0.0002w$, '2' means $\upsilon = 0.0005w$, '3' means $\upsilon = 0.001w$, '4' means $\upsilon = 0.002w$. The 'o' in 'a_o' means that outliers are present. The above labeling also holds for b0-b_o and c0-c_o.*

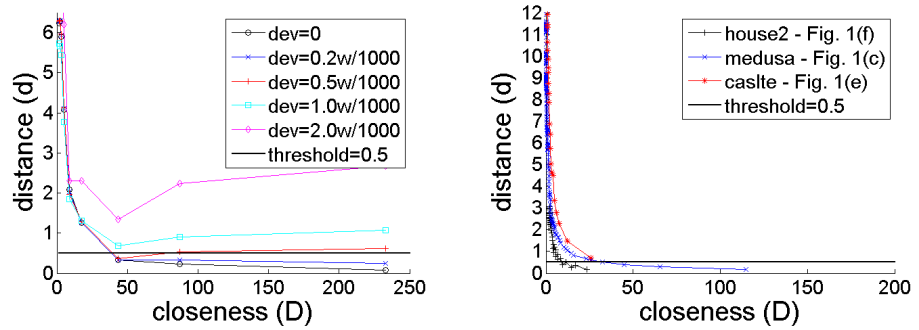
| exp | rank-3 ite. | | rank-4 ite. | | $\#cc$ | $A$ |
|---|---|---|---|---|---|---|
| | $E_3$ | $\kappa_3$ (%) | $E_4$ | $\kappa_4$ (%) | | |
| *a0* | 0.1 | 89.7 | 3e-5 | 100 | 7 | 0.05 |
| *a1* | 0.1 | 89.7 | 0.003 | 99.5 | 7 | 0.5 |
| *a2* | 0.1 | 89.6 | 0.007 | 99.2 | 9 | 1.8 |
| *a3* | 20 | 38.1 | 0.01 | 98.9 | 9 | 2.6 |
| *a4* | 0.1 | 85.9 | 0.03 | 98.9 | 10 | 7.9 |
| *a_o* | 70 | 95.8 | 4 | 21.4 | X | N |
| *b0* | 3e-7 | 100 | 4e-7 | 70 | X | D |
| *b1* | 0.003 | 99.9 | 0.002 | 99.7 | X | D |
| *b2* | 0.008 | 99.7 | 0.006 | 99.1 | X | D |
| *b3* | 0.02 | 99.4 | 6 | 23.3 | X | D |
| *b4* | 0.03 | 98.9 | 8 | 34.4 | X | D |
| *b_o* | 0.04 | 98.2 | 0.1 | 89.5 | X | D |
| *c0* | 2e-6 | 100 | 2e-7 | 72.2 | X | D |
| *c1* | 0.003 | 99.6 | 0.003 | 99.6 | X | D |
| *c2* | 0.007 | 99.4 | 5 | 61.9 | X | D |
| *c3* | 0.01 | 99.1 | 7 | 48.3 | X | D |
| *c4* | 0.03 | 98.3 | 0.06 | 88.4 | X | D |
| *c_o* | 0.1 | 96.2 | 0.03 | 97.8 | 5 | 24 |

**(a)** Results on synthetic sequence (Fig. 5.4(b))

| exp | #fm | rank-3 ite. | | | rank-4 ite. | | | #cc | $A$ |
|---|---|---|---|---|---|---|---|---|---|
| | | $E_3$ (pix) | $\kappa_3$ (%) | GAIC3 ($\times 10^3$) | $E_4$ (pix) | $\kappa_4$ (%) | GAIC4 ($\times 10^3$) | | |
| *h1* | 20 | 2.8 | 96.1 | 350 | 0.23 | 96.9 | 1.9 | 5 | S |
| *h2* | 12 | 2.5 | 95.8 | 480 | 0.27 | 95.4 | 3.2 | 3 | S |
| *h3* | 8 | 3.3 | 96.4 | 450 | 0.24 | 96.7 | 2.2 | 2 | D |
| *h4* | 4 | 0.25 | 99.9 | 4 | 0.19 | 65.7 | 3.2 | X | D |
| *m1* | 181 | 3.8 | 95.7 | 580 | 0.31 | 96.5 | 4.2 | 18 | S |
| *m2* | 16 | 1.5 | 98 | 810 | 0.26 | 96.6 | 12 | 5 | S |
| *m3* | 11 | 0.54 | 99.4 | 74 | 0.25 | 88.7 | 9.6 | X | D |
| *m4* | 8 | 0.29 | 99.8 | 10 | 0.25 | 84 | 8.1 | X | D |
| *g1* | 24 | 6.6 | 93.4 | 2400 | 0.27 | 98.9 | 4.6 | 24 | S |
| *g2* | 12 | 6.1 | 91.8 | 2300 | 0.26 | 98.4 | 4.9 | 12 | S |
| *g3* | 8 | 4.9 | 92.6 | 1500 | 0.29 | 94.1 | 14 | 8 | S |
| *g4* | 4 | 3 | 95.3 | 420 | 0.24 | 96.1 | 3.3 | 4 | F |

**(b)** Results on three real sequences. Labeling is as follows. *h1-h4*: results on *house1* (Fig. 5.4(d)); *m1-m4*: results on *medusa* (Fig. 5.4(c)); *g1-g4*: results on *castle* (Fig. 5.4(e)).

data, where we observe that $d$ monotonously decreases with $D$ for all three real sequences and for the synthetic sequence when the std. dev. of the injected noise is $\upsilon < 0.0005w$. This justifies the use of deviation $d$ for inferring the closeness $D$ in the proposed ACCC algorithm. Note that $d$ and $D$ are computed using Eq. (5.15) and Eq. (5.14), respectively.




(a) $d-D$ curves for the synthetic sequence with varying noise levels.

(b) $d-D$ curves for three real sequences.

**Figure 5.5:** *$d-D$ curves for synthetic and real sequences ($d$ = angle deviation and $D$ = closeness).*

The threshold $T_d$ for grouping the close cameras in the ACCC algorithm has been empirically set to $T_d = 0.5$ for all experiments. From Figs. 5.5(a) and 5.5(b) we note that $D < 40$ if $d > 0.5$, for all real sequences and for the synthetic sequence when $\upsilon < 0.0005w$. With the chosen $T_d$, our experimental results indicate that the ACCC algorithm is able to detect the $\#cc$ robustly. Column $\#cc$ of rows *a0-a4* in Table 5.1(a) shows the detected $\#cc$ for the synthetic sequence depicted in Fig. 5.4(b), where 10 cameras are positioned with varying distances between each other. With the rightmost camera taken as the reference camera, the closeness $D$ between each neighboring camera and the reference camera is shown in Fig. 5.5(a) as the $x$-coordinates of the data samples. From rows *a0-a2* of Table 5.1(a), we observe that 7 camera centers are counted for the first two experiments where $\upsilon < 0.0005w$. In these two experiments, the 3 rightmost cameras in Fig. 5.4(c) are considered to have the same center. This is what we expected, since their corresponding $D$ is larger than 40, as seen from Fig. 5.5(a). The detected $\#cc$ is not correct under large noise levels (Exps. *a2-a4*), because in this case ACCC is no longer able to distinguish the real cause of the large value of $d$, i.e., whether it stems from the 'distinctness' of the cameras, or from the large noise in the measurements. Fortunately, for the real data, the noise levels, which can be measured by the value of the re-projection error $E_4$, are mostly smaller than $0.0005w$ (equivalently around 0.5 pixels for $1024 \times 768$ images), as can be observed from our experimental results on real sequences as shown in Table 5.1(b) and Fig. 5.5(b).

Rows *h1-h4* in Table 5.1(b) show the results for the *house1* sequence. In the table, '#fm' denotes the number of images. As we see from the Exp. *h1* in Table 5.1(b), the rank-3 iteration fails to converge while the rank-4 iteration converges, and the $\#cc$ is correctly counted as $\#cc = 5$, as depicted in Fig. 5.4(b). Exp. *h2* corresponds to a case with 3 camera centers. Exp. *h3* represents a case of rotation around only 2 camera centers, and Exp. *h4*

points to a case of pure rotation. The ADCC algorithm has correctly counted the number of camera centers in all experiments.

Rows *m1-m4* show the experimental results for the *medusa* sequence. In contrast with the *house1* sequence where the camera centers are widely positioned (with a closeness $D < 30$), some adjacent cameras in *medusa* are very close to each other. Using the threshold $T_d = 0.5$, some neighboring cameras are grouped with the same center, even though they have slightly different positions. This explains why the 181 cameras depicted in Fig. 5.4(c) are counted as only 18 distinct camera centers in Exp. *m1*. In Exp. *m2*, 16 images are used and 5 camera centers are counted, which lead to a successful Euclidean reconstruction. Exp. *m3* corresponds to a degenerate case of rotation around 2 camera centers, and Exp. *m4* corresponds to a case of pure rotation. The ADCC algorithm successfully detects the critical configurations resulting from insufficient camera disparity. Experiments for the *castle* sequence (rows *g1-g4* in Table 5.1(b)) show similar results.

### 5.4.3 Dividing long image sequences

This section presents results for dividing long sequences using the ADLS algorithm. Tables 5.2(a) and 5.2(b) show the dividing results for the *castle* (Fig. 5.4(e)) and *medusa* (Fig. 5.4(b)) sequences. From the tables, we observe that the *castle* sequence is automatically divided into 2 subsequences and the *medusa* is automatically divided into 4 subsequences. Factorization-based SaM is successful for all subsequences, judged by our visual inspection. The reconstructed 3D models of the subsequences *t1* and *s1* and the merged 3D models for the complete *castle* and *medusa* sequences are depicted in Fig. 5.6. Note that direct factorization-based SaM for *medusa* is not possible, since feature points cannot be tracked along the complete sequence (from Frame 0 to Frame 193). For the *castle* sequence, 134 feature points are tracked along the complete sequence (from Frame 0 to Frame 25). However, the ADCC algorithm detects that outliers are present. Using the ADLS algorithm, both sequences are automatically divided into multiple subsequences for partial reconstructions, which are thereafter merged into a common coordinate system.

**Table 5.2:** *Results on dividing* castle *and* medusa *by the ADLS algorithm.*

| sseq | frames | $E_3$ | $\kappa_3$ | $E_4$ | $\kappa_4$ | #cc |
|---|---|---|---|---|---|---|
| *t1* | 0-15 | 20.8 | 96.0 | 0.43 | 99.0 | 16 |
| *t2* | 15-25 | 9.2 | 96.6 | 0.30 | 98.4 | 11 |

**(a)** Results for *castle*

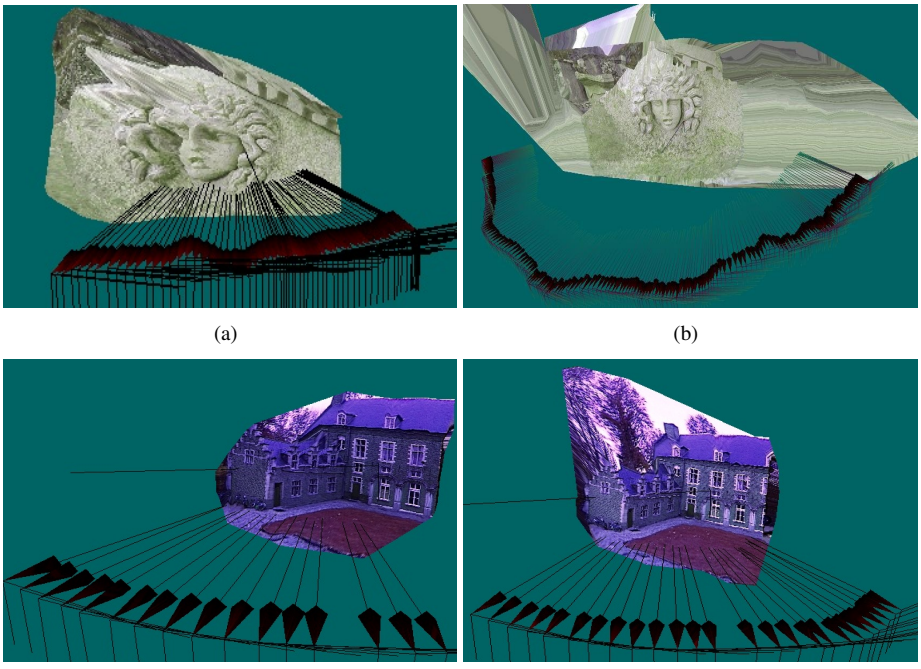
| sseq | frames | $E_3$ | $\kappa_3$ | $E_4$ | $\kappa_4$ | #cc |
|---|---|---|---|---|---|---|
| *s1* | 0-60 | 6.3 | 97.8 | 0.39 | 97.6 | 6 |
| *s2* | 60-124 | 3.6 | 98.2 | 0.34 | 98.2 | 5 |
| *s3* | 124-171 | 12.6 | 96.1 | 0.46 | 98.2 | 7 |
| *s4* | 171-193 | 26.7 | 96.5 | 0.36 | 99.3 | 8 |

**(b)** Results for *medusa*

As has been presented in Chapter 3, we have also applied the ADLS algorithm to the *campus* sequence which comprises of 1,561 frames. As shown in Table 5.3, the sequence is

**Table 5.3:** *The* campus *sequence with 1,561 frames is divided into 8 subsequences using the proposed ADLS algorithm.*

| subsequence | frame range | rproj err4 (pixels) | rank4 (%) | rproj err3 (pixels) | rank3 (%) |
|---|---|---|---|---|---|
| sub1 | 0 →186 | 0.27 | 99.5 | 12.6 | 80.7 |
| sub2 | 186 →372 | 0.23 | 98.7 | 2.9 | 96.2 |
| sub3 | 372 →561 | 0.22 | 99.2 | 6.0 | 93.7 |
| sub4 | 489 →693 | 0.25 | 99.5 | 9.9 | 88.6 |
| sub5 | 693 →913 | 0.21 | 99.2 | 3.6 | 95.2 |
| sub6 | 810 →1083 | 0.23 | 99.3 | 11.3 | 92.0 |
| sub7 | 1083→1305 | 0.28 | 98.3 | 7.5 | 94.3 |
| sub8 | 1305→1560 | 0.30 | 95.6 | 5.6 | 94.8 |



(a) (b)

(c) (d)

**Figure 5.6:** *Examples of reconstructed sparse 3D models:* ***(a)*** *3D model reconstructed from sseq* s1 *in Table 5.2(b);* ***(b)*** *merged 3D model for the complete* medusa *sequence;* ***(c)*** *3D model reconstructed from sseq* t1 *in Table 5.2(a);* ***(d)*** *merged 3D model for the complete* castle *sequence.*

automatically divided into 8 subsequences by the ADLS algorithm. The merged point cloud from the 8 subsequences is shown in Fig. 3.17. The results demonstrate that the proposed dividing algorithm is able to automatically divide a long sequence into subsequences, such that the factorization-based SaM can be successfully performed on individual subsequences.

## 5.5 Conclusion

This chapter has presented the proposed Algorithm for Detecting four Critical Configurations (ADCC) where the factorization-based SaM degenerates. Based on the ADCC algorithm, an algorithm is proposed to divide long sequences into multiple subsequences, such that a successful factorization-based reconstruction can be performed on individual subsequences with a high confidence.

First we have introduced the matrix factorization technique for decomposing a matrix into two matrices with a specified rank $r$. This matrix factorization technique is used in our iterative projective reconstruction algorithm to recover the camera motion and scene shape matrices. Based on these fundamental algorithms, we have designed two quantitative metrics for estimating the rank of the SMM, i.e., the rank-$r$-ness of the SMM and the re-projection error of the projective reconstruction. The rank estimation algorithm forms the key for our algorithm to *a-priori* detect the four critical configurations. These configurations are: (1) coplanar 3D points, (2) pure rotation of camera, (3) only two distinct camera centers, and (4) presence of excessive noise and outliers. Our detection is possible because the configurations in cases of (1), (2) and (4) will affect the rank of the scaled measurement matrix (SMM). Besides this, the number of camera centers in case (3) will affect the number of independent rows of the SMM. Hence, by examining the rank of the SMM, configurations (1), (2) and (4) are detected. Further, by analyzing the row space of the SMM, the number of distinct camera centers are counted, which solves detecting case (3). These two core steps form the backbone of the ADCC algorithm. Our experimental results on both synthetic and real sequences have demonstrated that the proposed ADCC algorithm is very effective in detecting the four critical configurations. In the experiments, the G-AIC metric from existing literature which is mainly used for model checking, has also been computed. However, the results show that G-AIC cannot be applied here for the detection of the four configurations.

Based on the ADCC algorithm, we have constructed an Algorithm for Dividing Long Sequences (ADLS) into multiple subsequences. We have employed the ADCC algorithm during dividing to ensure that all resulting subsequences are free from the four critical configurations. The proposed ADLS algorithm recursively divides a long sequence into short sequences, using three sub-algorithms for: (1) forward dividing to determine the best subsequence starting from Frame $i$, (2) backward dividing to determine the best subsequence ending at Frame $i$, and (3) bi-directional dividing to determine the best subsequence centering around Frame $i$. The three sub-algorithms are basically identical except for the directions of feature point tracking. For an automatic 3D reconstruction on a long sequence, such a dividing algorithm is essential for a robust reconstruction for long sequences, since we cannot track sufficient feature points along a long sequence of images. The benefit of using three directions for dividing is that it significantly increases the robustness of the reconstruction for a long sequence. Even if some subsequences are poorly divided, we are still able to obtain a good reconstruction from neighboring subsequences. This is much more robust than with the sequential dividing where subsequences are divided one by one from the start to the end, and one poor dividing can stop the dividing procedure.

Experimental results on real sequences comprising hundreds and more than thousand frames, have shown that the proposed dividing algorithm is able to yield subsequences where factorization-based 3D reconstruction can be successfully performed. However, as discussed

in Section 5.3.4, the proposed critical configuration algorithm has its inherent limitations. First, when the data contain noise and outliers, the proposed rank analysis may not be able to distinguish the real cause of the degeneracy of the configuration. Second, the rank analysis using the WIE algorithm does not always converge to sensible results, which may lead to a false detection. Furthermore, the proposed algorithm is designed to detect the only four critical configurations. If a sequence contains other critical configurations, the dividing algorithm will fail to provide a good dividing. Luckily, these situations occur rarely in practice. As demonstrated by the experimental results on both synthetic and real data, the rank-3-ness, rank-4-ness and their corresponding re-projection errors, and the counting of the number of distinct camera centers, provide a good set of metrics for a detection of the critical configurations and a robust dividing of long sequences.

For an automatic 3D reconstruction for long sequences, the algorithms presented in this chapter are crucial. With these algorithms, we can robustly divide a long sequence into short subsequences where a successful factorization-based 3D reconstruction can be obtained. The successful partial reconstructions are thereafter merged, such that the 3D model of the complete scene can be created. With all contributions from Chapters 3, 4 and 5, it is possible to reconstruct a sparse 3D scene model from long video sequences taken with a low-cost hand-held camcorder. This chapter completes our research on 3D reconstruction.

CHAPTER 6

# Estimating depth maps from multiple-view video

*The scene reconstruction algorithm presented in Chapters 3, 4 and 5 enables us to reconstruct a sparse point cloud from multiple-view image data. For applications such as 3DTV where dense per-pixel depth maps are needed for rendering the left and right views of a scene, the sparse point cloud is not sufficient. The density insufficiency and the depth holes in the image have to be filled. This chapter presents the proposed system for automatically creating depth maps from Multiple-View Videos (MVV). The proposed depth reconstruction system relies on the 3D modeling algorithms presented in previous chapters for sparse reconstruction and camera calibration, based on which depth maps are thereafter reconstructed using the technique of depth labeling via energy minimization. The main intended application of the obtained depth maps is for 3D content production in 3DTV.*

## 6.1 Introduction and overview

As illustrated in Fig. 1.3, the success of 3DTV in the market will rely on several factors, including 3D content production, content distribution and affordable display technology. Of those factors, distribution and display technology are relatively mature, while an efficient 3D content production is still at its early stage, and thus essential. This chapter aims at creating depth maps for a 3DTV system based on the depth-based imaging rendering approach, where an automatic creation of accurate depth maps is important. For good quality, the created depth maps should have the same resolution as the visual information, and the depth signal should be accurate. Furthermore, an automatic creation is desired in order to lower the cost of the 3D content production.

To obtain a vivid 3D visualization of a scene in 3DTV, the scene is usually captured using multiple camera systems. After that, the left and right views of the scene are rendered using

different approaches such as the depth-based approach, as introduced in Section 1.2.1. This thesis uses the depth-based image rendering approach, where per-pixel depths are required to render the left and right views. To create the depth maps, this chapter presents the proposed depth estimation system that creates depth maps from multiple-view videos. The system is depicted in Fig. 6.1 and has been briefly discussed in Chapter 2.

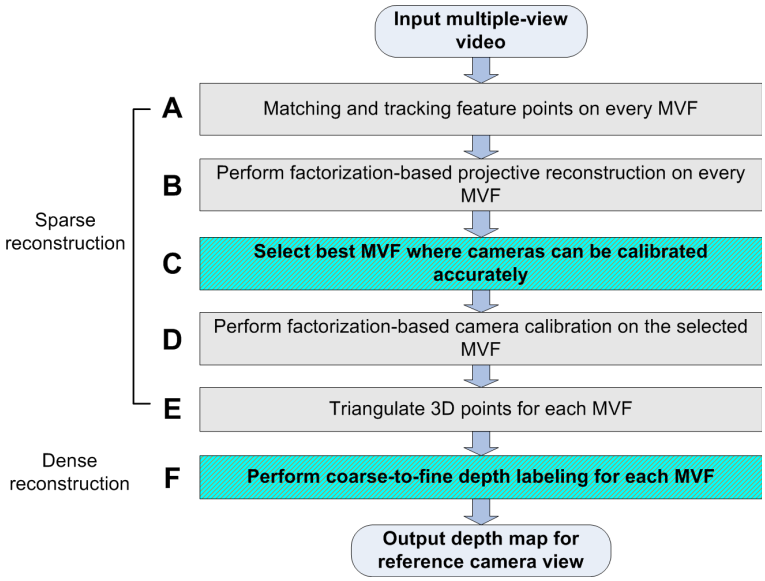



**Figure 6.1:** *Block diagram of the proposed depth estimation system.*

The 3D modeling system presented in the previous chapters enables us to reconstruct a sparse 3D model from multiple-view data. Despite the advances in this thesis, the obtained *sparse* 3D model of a scene is not sufficient for an accurate rendering of the left and right views, because per-pixel depth maps are required. The depth insufficiency has to be filled. This chapter presents a system for creating per-pixel depth maps from Multiple-View Video (MVV), and is presented in two separate parts: (1) *sparse reconstruction* to calibrate the cameras and to reconstruct the sparse point cloud for each MVF, and (2) *dense reconstruction* to create depth maps from the reconstructed point cloud for each MVF. MVF refers to Multiple-View Frame, which is defined as a set of images that are taken by the multiple-synchronized cameras at the same time instant. For example, all images shown in Fig. 2.5 comprise an MVF.

This chapter concentrates on our contributions on the only two shaded blocks (Blocks C and F) in Fig. 6.1. The algorithms for other processing blocks (Block A, B, D and E) are exactly the same as those used in the 3D modeling system, which has been presented in previous chapters. We briefly summarize them here. Block A is about feature point matching and tracking, which has been presented in Chapter 4. Block B deals with reconstructing the projective motion and shape, as described in Section 5.2.3. Block D addresses camera self-calibration using the factorization method, as presented in Section 5.2.4. Block E involves triangulating 3D points given feature point correspondences and camera parameters, as dis-

cussed in Section 3.3.3. The objective of this chapter is to find algorithms for Block C and F, dealing with selecting the best MVF for camera calibration, and coarse-to-fine depth labeling.

The remainder of the chapter is as follows. Section 6.2 introduces the sparse reconstruction algorithm, while our focus is on selecting the best MVF for camera calibration, after a brief introduction of the multiple-camera capturing system. Section 6.3 introduces the dense reconstruction algorithm, i.e., an algorithm for creating dense depth maps using the energy minimization approach. This section discusses a number of supporting algorithm steps contributing to depth reconstruction in terms of graph construction and cost definition. Section 6.4 presents the coarse-to-fine labeling algorithm to create an accurate depth map in a hierarchical way, which forms one of our major contributions of this chapter. Section 6.5 presents our experimental results for several multiple-view sequences. Finally, Section 6.6 concludes the chapter.

## 6.2 Sparse reconstruction

### 6.2.1 Background and context

As shown in Fig. 6.1, the proposed depth system comprises of five processing blocks. The first four blocks can be clustered into one processing module, which we define then as *sparse reconstruction*. In this module, the camera array is calibrated and the sparse 3D scene model for each MVF is reconstructed. The last block in Fig. 6.1 creates per-pixel depth maps by assigning discrete depth labels to individual pixels of an image, which is referred to as *dense reconstruction*. This section presents the sparse reconstruction algorithms as illustrated in Fig. 6.1. More specifically, we focus on Block C in Fig. 6.1, where the best MVF is selected to calibrate the cameras using the factorization-based 3D reconstruction algorithm.

Let us briefly discuss some work on multiple-camera systems for capturing dynamic scenes. Since the pioneering work of Kanade [33] about capturing dynamic 3D events using multiple synchronized cameras, 3D video capturing and rendering have been extensively studied in the past years. The main intended applications driving this research include free-viewpoint video and 3DTV. In [14], a survey is given on several time-varying scene capturing technologies. Among the available dynamic scene capturing approaches, the multiple-camera capturing is attractive, due to the continuously decreasing costs of image sensors and computing power.

As an example, Fig. 6.2 shows an experimental multiple-camera array [38] used for capturing dynamic 3D scenes. The design of a multiple-camera system is beyond the scope of this thesis. Therefore, all test MVVs are obtained either from publicly available websites, or from project collaboration partners [1].

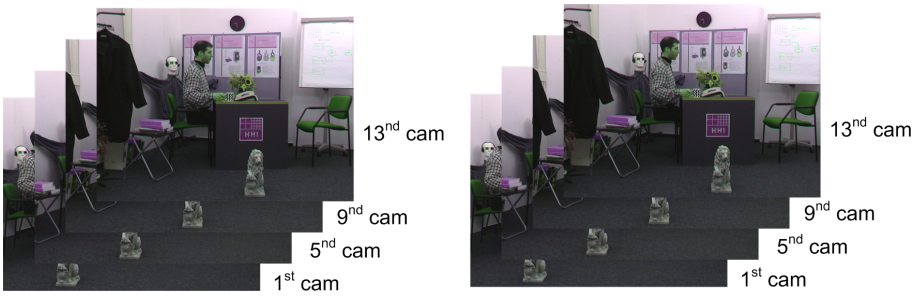
Using multiple synchronized cameras, multiple images of a 3D scene are obtained from *different* viewpoints. As discussed in Chapter 2 and as illustrated in Fig. 6.3, if we collect all images taken at the same time instant by the multiple cameras, we obtain a sequence of images of a scene from different viewpoints, which is referred to as Multiple-View Frame

[1] In the BSIK program I-Share, we gratefully acknowledge the cooperation with Philips Research Eindhoven.

(a) (b)

**Figure 6.2:** *Multiple-camera array [38] for capturing dynamic scenes.*

(MVF) in this thesis. The work in this chapter aims at creating depth maps for the selected reference cameras for each MVF.




(a) MVF *#1* taken at time instant *1* (b) MVF *#i* taken at time instant *i*

**Figure 6.3:** *All 13 images taken at the same time instant by different cameras are arranged into an image sequence, which is referred to Multiple-View Frame (MVF).*

### 6.2.2 Select the best MVF for camera calibration

As discussed in the previous section, an MVF comprises of a sequence of images taken by multiple synchronized cameras from different viewpoints. Conceptually speaking, an MVF is not different from a video sequence of a static scene taken by a moving camcorder, which is used in previous chapters for 3D reconstruction. As a consequence, the 3D modeling algorithm Alg. 3.4.1 can be directly applied to individual MVFs for sparse reconstruction. The main difference is that, due to the limited number of images in an MVF, camera calibration may become less robust, especially when the configuration of an MVF is near-critical (e.g. when feature points are located on the same 3D plane). This section presents our algorithm to select the best MVF for camera calibration, such that the cameras can be accurately calibrated.

Assuming that the parameters of the camera array do not change during scene capturing, we can optimize the camera calibration as follows. First, we *select* the 'best' MVF where

cameras can be calibrated accurately, and then we apply the calibration algorithm discussed in Section 5.2.4 to calibrate the parameters of the multiple cameras. With the calibrated cameras, 3D points can thereafter be triangulated for all MVFs, using the triangulation algorithm presented in Section 3.3.3. These last two steps (calibration and triangulation) are Blocks D and E in Fig. 6.1, which will not be further discussed here. In the following text, we concentrate on the first step, i.e., Block C in Fig. 6.1, which aims at selecting the best MVF. The involved algorithm steps are as follows.

We assume that projective reconstruction has been performed for MVFs and parameters $E_4$, $E_3$, $\kappa_4$ and $\kappa_3$ as discussed in Chapter 5 are available for each MVF. The steps for determining the best MVF are now summarized below.

---

**Algorithm 6.2.1:** SELECTBESTMVF()

---

**comment:** Select the best MVF from an MVV for camera calibration.

Step 1. Perform factorization-based projective reconstruction on all MVFs.

Step 2. Reject all MVFs where $E_4$ is larger than a pre-determined threshold.

Step 3. Reject all MVFs where $E_4$ is larger than $E_3$, or $\kappa_4$ is smaller than $\kappa_3$.

Step 4. Select the best MVF having the largest $\kappa_4$ from the MVFs that remain after the previous two steps.

---

The underlying principle of the above selection algorithm is the same as the principle of Alg. 5.3.1 for determining the best subsequence, i.e., the MVF with the largest rank-4-ness and the smallest re-projection error, is considered as the best MVF.

## 6.3 Dense reconstruction

The sparse reconstruction from the previous section reconstructs a sparse set of 3D points for individual MVFs. This section deals with the creation of the per-pixel depth maps from the reconstructed 3D points, where the quality of the depth maps is further increased. In this section, we first introduce the problem of multiple-view depth reconstruction, followed by a survey of related work in literature. After that, the problem of multiple-view depth reconstruction is mathematically formulated as an image-based labeling problem, using the theory of probabilistic inference within the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF). Subsequently, algorithms are presented for constructing a graph (sites and edges) for depth labeling and defining local energies, which are essential for energy minimization.

### 6.3.1 Introduction, motivation and related work

Reconstructing a dense 3D object model from a cloud of 3D points is usually formulated as a problem of surface reconstruction in the field of computer graphics, where surfaces are

created from the point cloud. Many surface reconstruction algorithms have been proposed for computer graphics. Unfortunately, due to the sparse and uneven distribution of the reconstructed 3D points by sparse reconstruction, most of these algorithms cannot be applied to such sparse 3D data and also to not to our case. The dense reconstruction to create per-pixel depth maps is typically solved by multiple-view reconstruction methods [89].

Our literature survey has revealed that the existing automatic depth-creation algorithms can be coarsely classified into two categories. One is the SaM approach, and the other is the Depth From Cues (DFC) technique that creates the depth from various depth cues such as the gravity, focus/defocus, occlusion, texture, etc. The SaM approach exploits the physical relation between the motion in the image, motion of the camera, and the motion of the object in the 3D space. One major advantage of this method is that this relation can be well modeled using the pin-hole camera model, epipolar geometry, etc. However, the deficiency is that it cannot handle scenarios containing critical configurations [61], which was discussed in the previous chapter. DFC uses heuristic depth cues for depth estimation, and is capable of analyzing all kinds of scenes, giving some inherent robustness. However, a significant drawback of DFC is that the heuristic depth cues are hard to model due to the complexity of the scene interpretation. Obtaining an accurate depth map is usually difficult. In this section, we concentrate on multiple-view depth reconstruction, where the initial creation of depth maps is obtained from SaM and then refined by DFC.

One major difficulty of the DFC approach is to combine and jointly optimize various depth cues that are based on very different heuristic and objective cues. This motivates our choice to adopt a generic framework with a generic metric for optimization. In this thesis, multiple-view depth reconstruction is formulated as an image-based labeling problem, using the theory of probabilistic inference within the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF). The MAP-MRF theory allows the problem to be formulated to generic data fields, also e.g. a pixel-based depth map. Furthermore, the solution of the probabilistic inference within the MAP-MRF framework can be solved using energy minimization. One major advantage of the energy-minimization approach is that it provides a straightforward and computationally-tractable formulation, where various depth cues and prior information about the scene can be combined to determine the optimal depth labeling. The global MAP solution can be found by minimizing the local energies using existing MRF-optimization algorithms, such as graph cut [67, 8] and belief propagation [75].

Let us now introduce related work from literature about multiple-view reconstruction. There are two classes of multiple-view reconstruction techniques in terms of how a 3D scene is represented: *volumetric* and *image-based* representations. The voxel coloring [72] and space carving [60] approaches belong to the first category. The advantage of the volumetric technique is its ability to preserve the depth discontinuity and handle occlusion. The drawback is that it can be easily trapped in local minima because of the lack of spatial smoothing. Especially, for image areas with little texture or having object boundaries, the photo consistency [2] does not provide a good constraint. Another drawback is that the visibility of each voxel needs to be known to compute the consistency cost. A base shape or visual hull is typically required, which complicates the problem. Recently, a few volumetric graph cuts [86] have been proposed, where the smoothing between neighboring voxels is applied.

[2] 3D points should be projected in multiple views of a scene onto the same corresponding positions.

For the image-based techniques, an image is selected as the reference image. Each pixel in the reference image is then assigned a discrete depth value, which is often formulated as a problem of energy minimization on an MRF network. Fast optimization algorithms such as graph cuts [67, 8] or belief propagation [75] can be used. One advantage of this approach is that it provides a clean and computationally-tractable formulation, where various depth cues or prior information can be integrated. A review of some existing image-based methods can be found in [70].

Our work aims at reconstructing depth maps from multiple-view images taken by multiple synchronized cameras. The occlusion is not an issue, since most pixels in the reference view are also visible in neighboring views. Thus, the image-based approach is suitable for solving our problem and is used in this thesis.

The remainder of this section is organized as follows. After problem formulation in the next subsection, an algorithm overview is provided in Section 6.3.3. The definition of the graph is introduced in Section 6.3.4. After that, the depth planes are defined to assign discrete depth values to pixels in Section 6.3.5. Finally, Sections 6.3.6 till 6.3.8 define the data penalty energy and interaction energy for energy minimization.

### 6.3.2 Problem formulation

Given a set of images $\{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n\}$ taken by cameras at different viewpoints, we want to find the depth value for every pixel $p \in \mathcal{P}$, where $\mathcal{P}$ is the reference image.

Based on the reconstructed 3D points of an MVF, we can compute the maximum depth $d_{max}$ and minimum depth $d_{min}$ of the 3D scene. From the depth range [$d_{min}$,$d_{max}$], a number of discrete depth planes $L \in \mathcal{L}$ can be sampled, which are thereafter assigned to individual pixels. Each 3D plane is defined to be parallel to the principal plane of the reference camera. The sampling process will be described later in this section. Given discrete depth planes, our aim is to assign a depth label to every pixel $p \in \mathcal{P}$ of the reference image. This is equivalent to finding a labeling function $f : \mathcal{P} \rightarrow \mathcal{L}$ to assign a depth label to each pixel. This assignment is a typical combinatorial optimization problem and can be solved using the max-flow algorithm [37].

To use energy minimization within the MAP-MRF framework for depth labeling, we need to construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$, where $\mathcal{V}$ is a set containing all sites (nodes) of the graph and $\mathcal{N}$ is a set containing all edges of the graph. Following the notations from [7], the energy $E$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$ can be computed as:

$$E(L) = \sum_{p \in \mathcal{V}} D_p(L_p) + \sum_{(p,q) \in \mathcal{N}} V_{p,q}(L_p, L_q), \tag{6.1}$$

where $L = \{L_p | p \in \mathcal{V}\}$ is a labeling of all nodes $\mathcal{V}$, $D_p(.)$ is the local *data penalty energy* that penalizes labeling $L_p$ at node $p$, $V_{p,q}$ is the local *interaction energy* that penalizes the depth changes between two neighboring nodes. Note that if every pixel of the reference image is treated as a node of the graph, then $\mathcal{V} = \mathcal{P}$ and the $\mathcal{N}$ can be defined as a set that contains all four connecting orthogonal edges of a pixel. As will be discussed later, this thesis uses a triangular mesh or regular lattice as the graph instead of the original image pixels. After the depths of the graph nodes are determined, depths of all pixels are linearly interpolated.

### 6.3.3 Algorithm overview

From the above problem formulation, we observe that three design factors are important for an accurate depth labeling using energy minimization. These three factors are as follows.

(1) We need to construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{N})$, where sites and edges of the graph are defined.

(2) We need to define the local data penalty energy *dCost(pixel, label)* for every node, and to define the local interaction energy *sCost(pixel1, label1, pixel2, label2)* for every edge.

(3) We need to design a robust and efficient optimization scheme to minimize the global energy.

Corresponding to the above three design factors, the proposed algorithm has three major steps, as illustrated in Fig. 6.4. In this section, we address the first two steps. The last step will be motivated and addressed in Section 6.4.

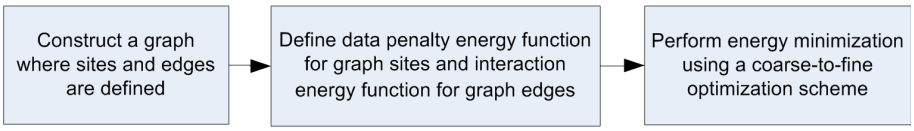



**Figure 6.4:** *Block diagram of the proposed algorithm for depth reconstruction using energy minimization. The algorithm is executed for each MVF to create the depth map.*

### 6.3.4 Constructing graph

This section involves the first block of Fig. 6.4. In the proposed depth-labeling algorithm, two types of graphs are used: (1) triangular mesh, and (2) regular lattice. The triangular mesh is depicted in Fig. 6.6, and the regular lattice is shown in Fig. 6.8. Let us now evaluate the benefits and drawbacks of both types of graphs.

The triangular mesh has the advantage of aligning well with the object boundaries, while the regular lattice has the benefit of being efficient for energy minimization due to its regularity. Especially when the graph nodes are coarse, the triangular mesh significantly improves the accuracy of the resulting depth maps, because object boundaries can be better preserved. When the graph nodes are dense, the advantage of the triangular mesh diminishes, because the regular lattice can also align well with the object boundaries in that case. For example, in an extreme case, if we treat every pixel as a node, and define graph edges as all orthogonal edges linking two neighboring pixels, we obtain a regular lattice that aligns perfectly with the object boundaries.

As will be further discussed in Section 6.4, a coarse-to-fine optimization scheme is proposed for energy minimization. In that algorithm, the triangular mesh is used in the early optimization stage when the graph nodes are still coarse, while the regular lattice is employed at the later optimization stage when the graph nodes are dense. By doing so, both the efficiency and the robustness of the energy minimization process are improved.

In the following paragraphs, we will introduce how the triangular mesh (Paragraphs A and B) and the regular lattice (Paragraph C) are constructed. Motivation of the construction is also explained.

(a) Reference image with reconstructed points.

(b) Segmentation map.

(c) Segmentation boundaries.

(d) Triangular mesh by Delaunay triangulation.

(e) Triangular mesh after splitting.

(f) Tetrahedral mesh from triangular mesh.

**Figure 6.5:** *Constructing triangular mesh on over-segmentation map.*

### A. Constructing triangular mesh

In the proposed graph construction algorithm, a triangular mesh is constructed based on the segmentation map of the reference image. Image segmentation is done by an existing segmentation algorithm from [18], which over-segments an image based on image texture and color. The algorithm parameters are set such that the algorithm tends to over-segment the image, in order prevent under-segmentation. Over-segmentation is used to ensure that each resulting triangle is located in the same segment with identical color and texture. The major steps of the proposed graph construction algorithm are illustrated in Fig. 6.5, and summarized below.

---

**Algorithm 6.3.1:** CONSTRUCTTRIANGULARMESH()

---

**comment:** Construct triangular mesh on segmentation maps.

Step 1. Segment the reference image - Fig. 6.5(b).

Step 2. Sample triangle vertices from segmentation boundaries - Fig. 6.5(c).

Step 3. Create triangular mesh using Delaunay triangulation - Fig. 6.5(d).

Step 4. Split large triangles - Figs. 6.5(e).

Step 4. Create tetrahedral mesh by connecting the centers of neighboring triangles - Fig. 6.5(f).

---

Let us now explain the rationale of the above algorithm. It is observed that neighboring pixels with similar color or texture are likely to be located on the same surface and will have similar depths. By constructing the triangular mesh on the segmentation map, all pixels in a triangle will be located in the same segment with same color and texture. Due to the over segmentation, the resulting triangles are well aligned with object boundaries, which contributes to the improved accuracy of the resulting depth maps. Furthermore, since all pixels in a triangle are likely to have similar depths, they can be labeled together, which contributes to both the robustness and efficiency of the depth-labeling process. Summarizing, by constructing a graph in this way, two important depth cues, i.e., the color and the texture information of an image, are automatically used during depth labeling.

Fig. 6.6 shows a tetrahedral mesh that is computed from a triangular mesh. As we observe from the figure, each node in a tetrahedral mesh represents a triangle in the corresponding triangular mesh, and each edge represents the common edge between two neighboring triangles in the corresponding triangular mesh. Using such a tetrahedral mesh, all pixels in a triangle are assigned a single depth value, which improves the robustness and efficiency of the depth labeling.

Step 4 splits large triangles into smaller sizes, such that the depth accuracy is refined. The splitting of large triangles is presented in the following Paragraph B.

Since triangular meshes and regular lattices are used in our case, the term ‘pixel’ in the following text does not refer to the original image pixels. Instead, it refers to a node in a graph. To cover this aspect, we do not discriminate the term ‘pixel’ from ‘node’ in the following discussion.

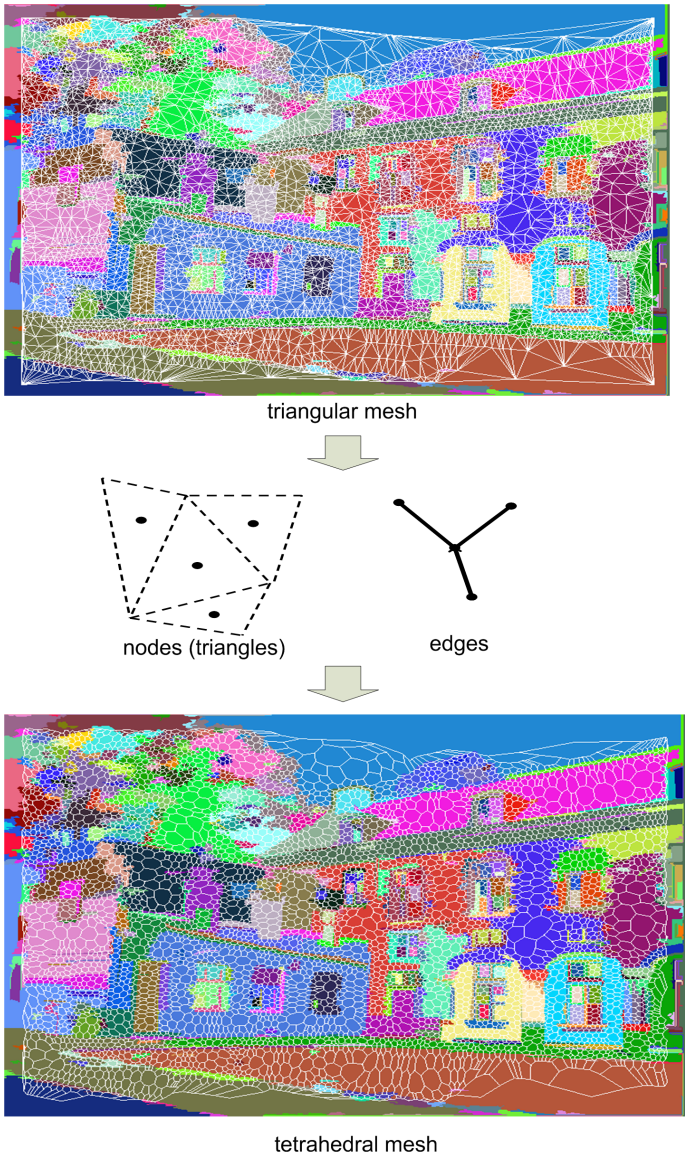



**Figure 6.6:** *Constructing tetrahedral meshes from triangular meshes. Each node represents a triangle and each edge represents the common edge of the two neighboring triangles where energy interacts.*

### B. Splitting large triangles

Step 4 of Alg. 6.3.1 splits large triangles into smaller ones. As illustrated in Fig. 6.7, a large triangle is split into three smaller triangles if the maximum distance between the three vertices and its centroid is larger than a given threshold $T_{graph}$. The threshold specifies how

large a triangle can be in a triangular mesh, which is referred to as the *graph granularity* in this chapter. Apparently, the smaller the $T_{graph}$, the finer the graph and the more accurate the resulting depth maps. However, a fine granularity also increases the complexity and decreases the robustness of the labeling process, since more nodes need to be labeled. Consequently, the labeling process is slower and gets easier trapped in local minima. The refinement of the triangulation is further discussed in Section 6.4.

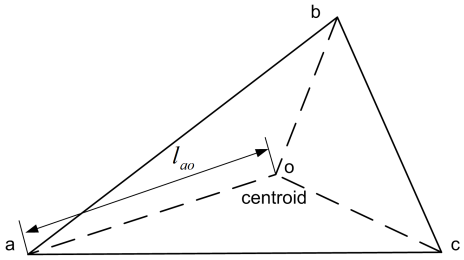



**Figure 6.7:** *Splitting large triangles: a large triangle is split into three smaller triangles if the maximum distance between its three vertices and its centroid (length $l_{ao}$ in the figure) is larger than a given threshold $T_{graph}$.*

### C. Constructing regular lattice

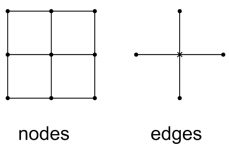




**Figure 6.8:** *Constructing regular lattice: an image is divided into regular grids with horizontal and vertical lines that are displaced with a equal distance $T_{graph}$. The intersections of the grid lines are defined as the nodes, and the four connecting edges of each vertex are defined as edges.*

As discussed in the beginning of this subsection, regular lattice is also used in our proposed depth-labeling algorithm when the graph nodes are dense. Constructing a regular lattice can be done easily. Given a granularity threshold $T_{graph}$, the regular lattice is constructed by dividing an image into grids, where the horizontal and vertical grid lines are positioned

with an equal distance $T_{graph}$ between each other. As illustrated in Fig. 6.8, the intersections of the grid lines are defined as the nodes, and the four connecting edges of each node are defined as edges.

This ends the discussion on graph construction. The next subsection deals with sampling depth planes for depth labeling.

### 6.3.5 Computing the set of depth planes for depth labeling

To label the nodes of a graph, we need a set of discrete depth labels. This subsection presents how the discrete depth labels are determined from the reconstructed point cloud of a scene. After that, these labels can be assigned to the individual nodes to reconstruct depth maps by the proposed labeling algorithm. The steps to compute depth labels from the reconstructed point cloud are summarized in Alg. 6.3.2.

In Alg. 6.3.2, threshold $T_{int}$ specifies the interval between two neighboring depth planes, which is referred to as the granularity of the depth planes or *depth granularity*. Similar to the graph granularity, depth granularity affects directly the computation time and the robustness of the labeling process. A fine depth granularity slows down the labeling process, and decreases the robustness of energy minimization, since more depth labels tend to make the optimization process more difficult to converge to the global minimum.

**Algorithm 6.3.2:** COMPUTEDEPTHPLANE()

**comment:** Sampling depth planes from the reconstructed point cloud.

Step 1. Shift the world-coordinate frame to the camera coordinate frame of the reference camera.

Step 2. Determine the maximum and minimum depths of the scene. The minimum depth can be computed as the $Z$-coordinate of the point that is located closest to the $Z = 0$ plane, and the maximum depth is computed as the $Z$-coordinate of the point that lies farthest away from the $Z = 0$ plane. The minimum and maximum depths can be multiplied by scale factors, to account for the fact that some scene points may be outside the reconstructed point cloud.

Step 3. Starting from the minimum depth, the next depth plane is determined by moving a 3D point along the principal axis from the current plane to the next plane, such that the shift of its 2D projected shift in a pre-selected neighboring image equals a given threshold $T_{int}$, as illustrated in Fig. 6.9.

Step 4. Repeat Step 3 until the maximum depth is reached. The resulting depth planes are depicted in Fig. 6.10.

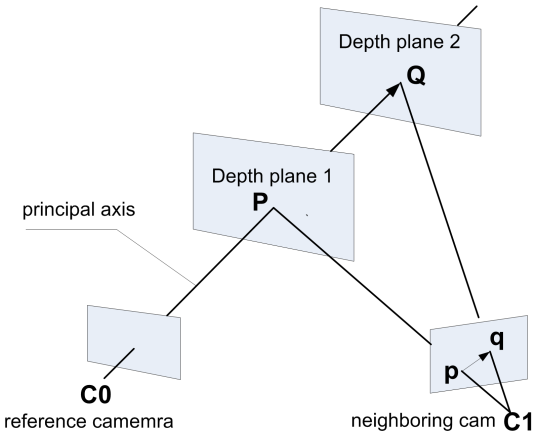



**Figure 6.9:** *Computing depth planes: moving a 3D point along the principal axis of the reference camera from plane 1 to plane 2, such that the 2D projected shift* $||\mathbf{p} \rightarrow \mathbf{q}||$ *in a neighboring image equals a given threshold.*

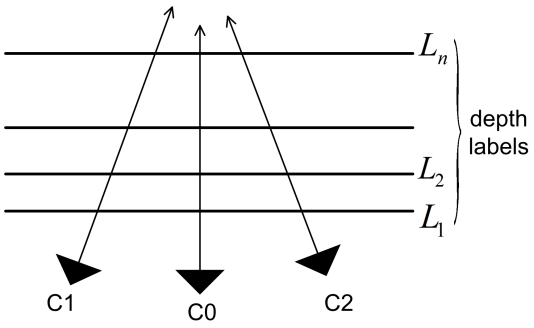



**Figure 6.10:** *Depth planes computed by Alg. 6.3.2.*

### 6.3.6 Defining local data penalty energy

Various depth cues can be used to determine the depths. In this section, we describe how multiple objective and heuristic depth cues are used to define the data penalty energy, which is also referred to as *data cost*. This energy is essential for energy minimization. The sequel of this subsection contains Paragraphs A through E computing the energy using different depth cues. A depth cue can also be a constraint, as indicated by the titles of the paragraphs.

#### A. Photo-consistency constraint

Given a depth $L$ for pixel $p$, the photo-consistency cost $E_p^{photo}(L)$ is computed as the average of the Sum of Absolute Differences (SAD) between image patches in the reference image and the neighboring images. A high SAD value means a low correlation, which implies that the depth $L$ is not appropriately set. Fig. 6.11 depicts a typical *SAD-Depth* curve, where the SAD

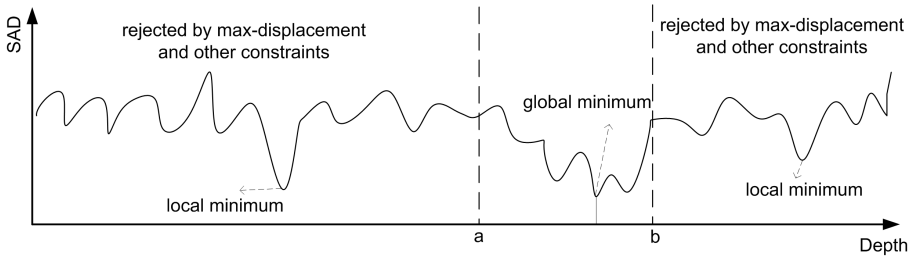



**Figure 6.11:** *Typical SAD-Depth curve.*

values vary with the depth values. From the figure, we observe that by using only the photo-consistency constraint, it is hard to determine the optimal solution because of the multiple local minima. This ambiguity has to be resolved using other depth cues. For example, if we can limit the depth range to $[a, b]$ in Fig. 6.11 using other depth cues, the probability to obtain the optimal solution will be much higher.

### B. Max-displacement constraint

A 3D point $(p, L)$ is projected onto multiple views. Knowing the camera parameters, the pixel displacement $d_p$ between the positions of its 2D projections in neighboring images can be computed. For multiple-view images, the pixel displacement between two neighboring images are typically limited. This can be exploited as a constraint for depth labeling, which is referred to here as the *max-displacement constraint*. Using this constraint, a large number of depth labels can be rejected by ensuring that the pixel displacement $d_p$ is smaller than a given threshold $d_{max}$. This reduces the depth range for individual nodes and thus reduces the ambiguity of depth labeling.

### C. Slanting constraint

For most outdoor scenes, it is plausible to assume that objects at the top of an image are farther away than the bottom objects, which can also be used to constrain the depth-labeling process. This is referred to as *slanting constraint* in this chapter [3]. To apply the slanting constraint, the following slanting cost is introduced, which is computed as

$$E_p^{slanting}(L) = \begin{cases} 3^{(dL-0.5)\cdot 10} & \text{for } dL > 0.5, \\ 0 & \text{elsewhere.} \end{cases} \tag{6.2}$$

In the above equation, $dL$ is the relative depth difference between the current label $L$ and the initial label $L_0$, which is computed as $dL = |d(L) - d(L_0)|/(d_{max} - d_{min})$, where $L_0$ is the initial depth label that is computed based purely on the vertical image coordinate of the node, and $d(L)$ denotes the depth value for depth label $L$.

The shape of the above cost function is shown in Fig. 6.12, where we observe that a node is assigned a large cost if its depth differs significantly from the initial depth $L_0$. Since

[3] The slanting constraint is sometimes also called gravity constraint in literature.

computing the initial depth based on the vertical image coordinate does not give a high confidence, the depth of a node should be allowed to differ from the initial depth. This explains the wide range of the zero cost in Fig. 6.12. This implies that the slanting constraint is a soft and heuristic depth cue that should be imposed with care.

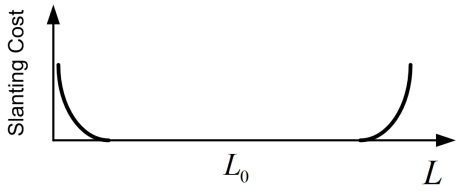



**Figure 6.12:** *Shape of the slanting cost function.*

### D. Surface-point constraint

The positions of a number of 3D points have been reconstructed during sparse reconstruction. To constrain the 3D surface to fit to these known 3D points, we introduce a *surface-point cost* $E_p^{point}$ in the data-cost definition. The steps to compute the surface-point cost are: (1) for each triangle $p$, find all 3D points that are projected onto that triangle, (2) the surface-point cost is then computed as the Euclidian distance between point $(p, L)$ and all 3D points found in the first step.

### E. Rejection of floating triangles

For most scenes, it is plausible to assume that there is no 'floating' triangle whose depth differs significantly from all its neighbors. To remove such floating triangles, we introduce the scaling factor $w_s$ in the data-cost definition Eq. (6.3) in order to give a penalty to floating triangles. The procedure is as follows: once we find that the depth of a triangle differs significantly from all its three neighbors, the data cost is multiplied by a weighting factor $w_s$ that is larger than unity.

### F. Definition of total data cost

With all the costs defined in Paragraphs A through E, the total data cost $D_p(L)$ for pixel $p$ is computed as:

$$D_p(L) = \begin{cases} w_s A_p \cdot [w_1 E_p^{photo}(L) + w_2 E_p^{slanting}(L) + w_3 E_p^{point}(L)] & \text{if } d_p < d_{max}, \\ \infty & \text{elsewhere.} \end{cases} \tag{6.3}$$

In the above, $A_p$ is the image area of the triangle $p$, parameter $d_p$ is the pixel displacement and $d_{max}$ is the maximum pixel displacement between neighboring images, as in Paragraph B. Parameters $w_1$, $w_2$ and $w_3$ are weighting factors used to adjust the contributions from individual depth cues. The higher the confidence, the larger the weight that we assign to that specific depth cue. For example, $w_1$ is a weight used to adjust the contribution of the photo

consistency. A large $w_1$ should be used if photo consistency gives a high confidence. Conversely, a small $w_1$ should be used if photo consistency does not provide a good constraint. This motivates why the following algorithm adapts the data cost to the local image content, as will be described in the next subsection.

### 6.3.7 Adapting penalty energy to local image content

Within an image, the content can vary significantly. Therefore, the *local* data cost should be adapted to the *local* image content, in order to use the local image properties. This subsection presents a method for adapting the photo-consistency cost to the local image content.

As we discussed in Paragraph E, photo consistency does not always provide a good constraint, so that it needs to be used with care. We have observed the following two situations where the photo consistency fails in providing a high confidence. First, for smooth image areas containing little texture, photo consistency will be consistently high, regardless of the depth labels that we assign to the node. Second, for image areas containing object boundaries, photo consistency will be consistently low for all depth labels. In both cases, the photo-consistency cost is either constantly small or is constantly large (small variation between photo-consistency costs), which cannot give a high confidence. Therefore, a small weighting factor $w_1$ should be used in these cases. In the following, two simple metrics are proposed to measure the *smoothness* of an image area, and the *variation* of the photo-consistency costs.

First, compute the intensity variance $\sigma_p^2$ of the $7 \times 7$ image patch around node $p$, for $\forall p \in \mathcal{V}$. We also compute the photo-consistency cost $E_p^{photo}(L)$ for node $p$, for $\forall L \in \mathcal{L}$. With the computed $\sigma_p^2$ and $E_p^{photo}(L)$, the following two metrics are computed

$$r_p^s = \sigma_p^2 / (\sum_{i \in \mathcal{V}} \sigma_i^2 / N), \tag{6.4}$$

where $N$ is the size of set $\mathcal{V}$, and

$$r_p^c = \frac{\min_{j \in \mathcal{L}} E_p^{photo}(L_j)}{\sum_{j \in \mathcal{L}} E_p^{photo}(L_j)/M}, \tag{6.5}$$

where $M$ is the size of set $\mathcal{L}$.

Ratio $r_p^s$ in Eq. (6.4) measures the smoothness of the image patch, and $r_p^c$ in Eq. (6.5) measures the variation of the photo-consistency costs. The smaller the $r_p^s$ and the larger the $r_p^c$, the lower the confidence of the photo-consistency cost, and thus a smaller weight $w_1$ should be used.

This subsection has defined the local penalty energy. The next subsection presents the definition of the local interaction energy.

### 6.3.8 Defining local interaction energy (smoothness cost)

The fundamental principle for depth interaction is the geometric *connectivity* and *discontinuity*, which leads to the widely-used *piecewise-smooth* assumption. In this section, the piecewise-smooth assumption is used to define the interaction energy, such that the resulting depth maps will be smooth while preserving the object boundaries. The interaction energy is

referred to as *smoothness cost* in the sequel for the sake of similarity with naming of other cost terms.

Assuming that two depth labels $L_p$ and $L_q$ are assigned to two neighboring pixels $p$ and $q$, the interaction energy $V_{p,q}(L_p, L_q)$ is defined as the distance between those two depth planes:

$$V_{p,q}(L_p, L_q) = |d(L_p) - d(L_q)|, \tag{6.6}$$

where $d(\cdot)$ denotes the depth value of the depth plane.

Similar to the data cost, the smoothness cost should be adapted to the local image content, in order to employ the local image properties. For example, smoothing should not be applied to neighboring nodes across object boundaries, while smoothing should be strongly applied to neighboring nodes located on the same object surface. Below, we describe the adaptation of the smoothness cost to the local image content.

The smoothness cost is adapted based on the detection of object boundaries. If an object boundary is detected between two neighboring nodes, a very small smoothness cost is then used. Because our graph is constructed on the over-segmentation map, we know the segment $S_p$ where each node $p$ is located. If two neighboring nodes are located in different segments and have a large depth difference, there is a high probability that an object boundary occurs between the two nodes. Therefore, the smoothness cost can be adjusted by checking the segments and depths of two neighboring nodes, as in the following equation:

$$V_{(p,q)}(L_p, L_q) = \begin{cases} 0, & |L_p - L_q| > T_h \text{ and } S_p \neq S_q, \\ |d(L_p) - d(L_q)|, & \text{elsewhere.} \end{cases} \tag{6.7}$$

In the above equation, $T_h$ is a threshold for the depth difference between two nodes. The equation implies that an object boundary between nodes $p$ and $q$ is detected if $|L_p - L_q| > T_h$ and $S_p \neq S_q$.

Up to here, we conclude Section 6.3 with the definition of graph and local energies, which involves the first two blocks in Fig. 6.4. With graph and local energies, we can start energy minimization to label the nodes, as will be presented in the next section.

## 6.4 Coarse-to-fine depth labeling

For minimizing the global graph energy, a robust and efficient optimization scheme is desired. This section presents a coarse-to-fine optimization scheme, where both the graph granularity and the depth granularity are refined gradually in multiple optimization passes. In the following paragraphs, we will first discuss the motivation to use such a coarse-to-fine optimization scheme, where three design factors are considered. After that, the steps of the proposed coarse-to-fine scheme are presented in a separate subsection.

### A. Impact of graph granularity

As discussed in Paragraph B of Section 6.3.4, the graph granularity $T_{graph}$ has a direct impact on the labeling process. On one hand, a fine granularity enforces a better alignment of the triangular mesh with the object boundaries and therefore increases the accuracy of the

resulting depth map [4]. On the other hand, a fine granularity also increases the complexity and decreases the robustness of the labeling process, because more nodes need to be labeled and minimization is easier trapped in local minima. Especially when the depth granularity is also fine, a large number of nodes together with a large number of depth labels will not only significantly slow down the labeling process, but also lead to the non-convergence of the optimization process.

## B. Impact of depth granularity

As discussed in Section 6.3.5, the impact of the depth granularity $T_{int}$ is similar to the graph granularity. There is a similar tradeoff as above. On one hand, a fine depth granularity decreases the intervals between two depth planes and thus increases the accuracy of the resulting depth maps. On the other hand, it also slows down and decreases the robustness of the labeling process, since more labels need to be assigned and the optimization can be more easily trapped in local minima.

## C. Triangular mesh versus regular lattice

As stated in Section 6.3.4, the triangular mesh gains the advantage over the regular lattice when the graph granularity is coarse, because it better aligns with object boundaries. However, this advantage diminishes when the graph granularity increases, because a fine regular lattice can also align well with the object boundaries. For example, for very thin objects in a scene, the regular lattice better describes the objects than the triangular mesh if the graph granularity is sufficiently fine. Therefore, this has motivated us to use the triangular mesh when the graph granularity is coarse, and employ the regular lattice when the graph granularity is fine.

### 6.4.1 Steps of coarse-to-fine depth labeling

Based on the above discussion, we propose a coarse-to-fine depth-labeling algorithm, where both the graph granularity and depth granularity are gradually refined in multiple optimization passes. Furthermore, the triangular mesh is used in early optimization when the graph nodes are sparse, and the regular lattice is used in later optimization when the graph nodes are dense. The labeling results of a preceding optimization pass are used as initial results in the current pass, in order to constrain the labeling process. By doing this, the number of depth labels that can be assigned to individual nodes is significantly reduced, which contributes to both the efficiency and robustness of the proposed algorithm.

Fig. 6.13 illustrates the proposed coarse-to-fine labeling process. The steps of the proposed coarse-to-fine labeling algorithm are summarized as follows.

[4] This applies also to the regular lattice.

**Algorithm 6.4.1:** COARSETOFINEREFINEMENTSCHEME()

**comment:** Coarse-to-fine depth labeling

Step 1. Given a graph granularity $T_{graph}$, create the triangular mesh or the regular lattice, using the algorithms described in Section 6.3.4. The use of the triangular mesh or the regular lattice is empirically determined based on the value of $T_{graph}$.

Step 2. Given a depth granularity $T_{int}$, compute a set of depth labels using Alg. 6.3.2.

Step 3. Determine the depth range and the set of allowed depth labels for each node, using the algorithm to be described in Section 6.4.2.

Step 4. Perform energy minimization via graph cut, assigning a depth to each node, where the label is taken from its own set of allowed labels (see Section 6.4.2). This is done using an existing MRF-optimization software package from [2, 15].

Step 5. Decrement $T_{graph}$ and $T_{int}$ and repeat the above steps until both the minimum graph granularity and the minimum depth granularity are reached.

The above coarse-to-fine optimization scheme brings two benefits. The first is the computational efficiency. The depth range of each node is hierarchically refined in multiple passes. This reduces the number of depth labels for individual nodes, thereby decreasing the processing time. The second benefit is the robustness. Because of the reduced depth labels for individual nodes, the solution is less easily trapped in local minima.

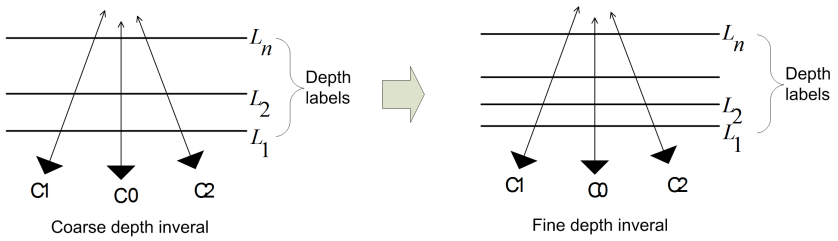
### 6.4.2 Determining allowed depth range of a node

With the labeling results from the preceding labeling pass, the depth range for each node in the current optimization pass is estimated. Therefore, the set of depth labels that can be assigned to each node is determined, which is referred to as *allowed label set*. Obviously, each node has its own allowed label set, from which each node receives its depth label. Below, we summarize the steps to determine the depth range for node $p$ in the current optimization pass.

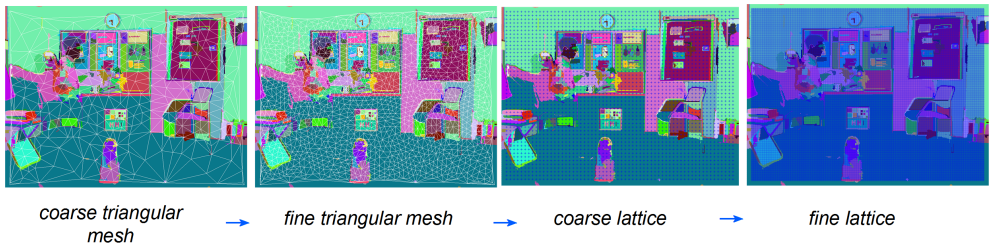
1. First optimization pass: the depth range is set to the global depth range that is determined in Step 2 of Alg. 6.3.2. Otherwise, go to the next step.

2. Current optimization pass (not the first pass): find the triangle or the grid in the graph where node $p$ is located, which is used in the previous pass.

3. Let $U$ denote the triangle or the grid where node $p$ is located, the allowed depth range for node $p$ is set to a range that contains the depths of all vertices of $U$, which are obtained from the previous pass. The depth range is enlarged to allow large depth changes across consecutive optimization passes.




(a) Coarse-to-fine refinement of depth intervals.




(b) Coarse-to-fine refinement of graphs.

**Figure 6.13:** *Coarse-to-fine depth labeling: a depth map is refined in multiple optimization passes. Coarse graphs and coarse depth intervals are used in the early optimization passes, while finer graphs and finer depth intervals are used in the later passes. Triangular mesh is used in early passes, while regular lattice is used in the later passes.*

As we observe from the above steps, the labeling results of the previous pass are used to determine the depth ranges of each node in the current optimization pass. This significantly narrows down the depth range of a node and decreases the number of allowed depth labels for a node, which increases both the efficiency and robustness of the labeling process.

## 6.5 Experimental results

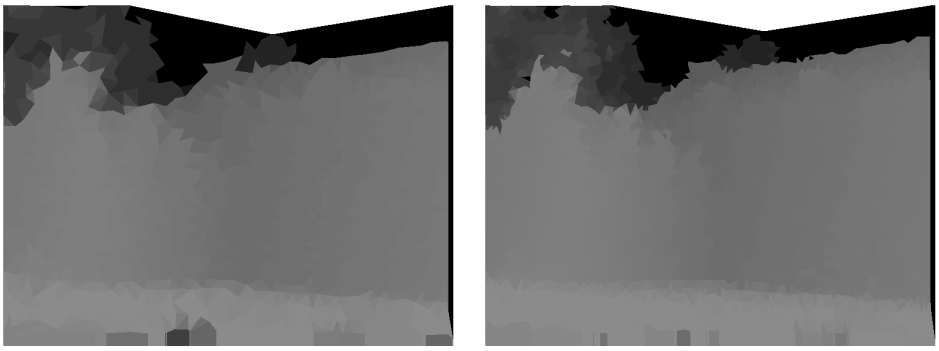
This section presents our experimental results on both multiple-view images and Multiple-View Videos (MVV). The only difference between experiments on multiple-view images and MVV is that selection of the best MVF is not required for multiple-view images, while it is required for MVV.

### 6.5.1 Results for multiple-view images

The proposed algorithm is tested on the *castle* sequence from [3]. As depicted in Fig. 6.5, the *castle* sequence comprises of 26 images taken from different viewpoints. In our experiments, a sparse shape (point cloud) of the scene is reconstructed using the sparse reconstruction algorithm Alg. 3.4.1 (note that blur-and-abrupt image removal is not required in this case).

The depth map is then created using the coarse-to-fine depth-labeling scheme Alg. 6.4.1.

As an example, Image 12 is selected as the reference image of which the depth map is created. Four neighboring images are used for depth labeling. Thus, Image 12 is the central image and the four neighboring images are Images 10, 11, 13, and 15. A $5 \times 5$ pixel window is used to compute the intensity variance $\sigma_p^2$ of the image patch about pixel $p \in \mathcal{P}$. A $17 \times 17$ pixel window is used to compute the SAD value $S_p(L)$ for all depth labels $L \in \mathcal{L}$. The SAD value is computed in all RGB channels and then averaged. If the projection of a 3D point is outside an image, the SAD value for that image is excluded from the averaging.



(a) TM with $T_{graph} = 16$ and $T_{int} = 4$.

(b) TM with $T_{graph} = 8$ and $T_{int} = 2$.

(c) RL with $T_{graph} = 4$ and $T_{int} = 2$.

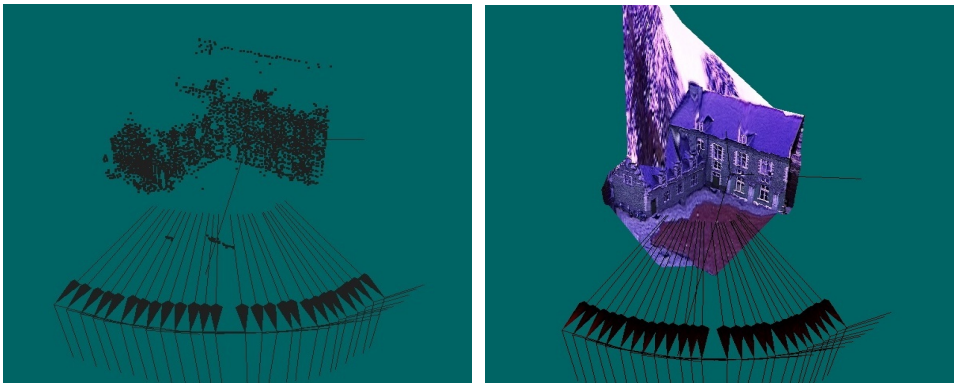
(d) RL with $T_{graph} = 2$ and $T_{int} = 1$.

**Figure 6.14:** *Depth maps for the reference image of the castle sequence obtained from multiple optimization passes with varying graph granularity $T_{graph}$ and depth granularity $T_{int}$, using Triangular Mesh (TM) or Regular Lattice (RL). The figure should be read in the order (a)-(b)-(c)-(d).*

During the hierarchical depth-labeling process, the starting graph granularity $T_{graph}$ is 16 pixels and is then refined to 2 pixels in the end. After every pass, $T_{graph}$ is halved. The starting depth granularity $T_{int}$ is 4 pixels and is refined to 1 pixel in the end. After every optimization pass, $T_{int}$ is halved. The triangular mesh is used for all passes where $T_{graph} > 4$ and the regular lattice is used when $T_{graph} \leq 4$. The four used optimization passes are as follows:

P1: Triangular mesh with $T_{graph} = 16$ and $T_{int} = 4$, see Fig. 6.14(a),
P2: Triangular mesh with $T_{graph} = 8$ and $T_{int} = 2$, see Fig. 6.14(b),
P3: Regular lattice with $T_{graph} = 4$ and $T_{int} = 2$, see Fig. 6.14(c),
P4: Regular lattice with $T_{graph} = 2$ and $T_{int} = 1$, see Fig. 6.14(d).

Fig. 6.5(a) shows the central image of *castle*, overlayed with 2D projections of all reconstructed 3D points. Fig. 6.5(e) shows the triangular mesh and Fig. 6.8 shows the regular lattice constructed on the over-segmentation map. The point cloud and the cameras reconstructed during sparse reconstruction are depicted in Fig. 6.15. Visual inspection of Fig. 6.15 reveals that the Euclidean scene shape of *castle* is accurately reconstructed. Fig. 6.14 shows the depth maps obtained from multiple optimization passes. As we see from the figure, the accuracy of the depth maps is gradually increased in multiple optimization passes. The depths of most pixels are accurately reconstructed. Depth discontinuities (e.g. object boundaries between the foreground house and the background tree) are well preserved, due to the adaptation of the local costs to the local image content.



(a) Point cloud. (b) Point cloud with texture mapping.

**Figure 6.15:** *Reconstructed point cloud and cameras for* castle.

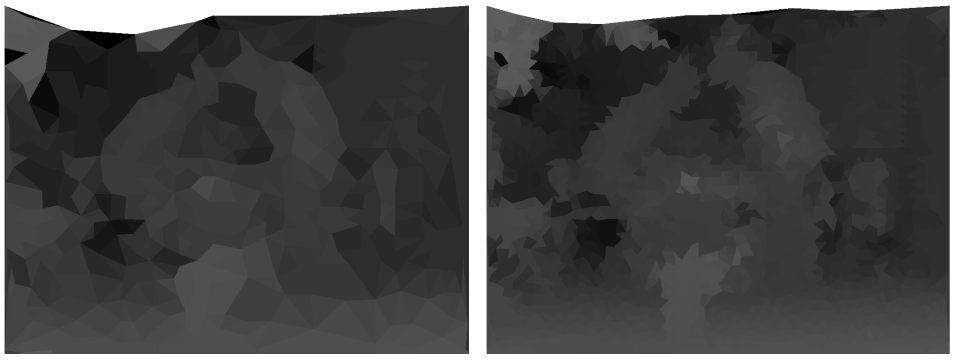
### 6.5.2 Results on multiple-view video

The proposed depth-labeling algorithm Alg. 6.4.1 has been applied to the *bookarchive* sequence to create depth maps for the central camera view (7th camera). The sequence is depicted in Fig. 6.3, where we see that 13 views are captured by 13 synchronized cameras. In our experiments, the depth maps for the first 100 MVFs are created.

Fig. 6.16 depicts the reconstructed point cloud and cameras of one MVF of the *bookarchive* sequence. Visual inspection of the figure and the 3D point cloud in visualization software shows that the Euclidean scene shape and camera positions are accurately reconstructed. Fig. 6.17 depicts the depth maps obtained by the proposed algorithm. As observed from the figure, the accuracy of the depth maps are gradually improved (reading from (a) to (d)) during each optimization pass. Visual inspection shows that the proposed algorithm is able to obtain an accurate depth map for every MVF.

(a) Point cloud. (b) Point cloud with texture mapping.

**Figure 6.16:** *Reconstructed point cloud and cameras with/without texture mapping of an MVF of the bookarchive sequence.*



(a) TM with $T_{graph} = 32$ and $T_{int} = 4$. (b) TM with $T_{graph} = 16$ and $T_{int} = 2$.

(c) RL with $T_{graph} = 16$ and $T_{int} = 2$. (d) RL with $T_{graph} = 4$ and $T_{int} = 0.5$.

**Figure 6.17:** *Depth maps obtained at varying graph granularity and depth granularity using Triangular Mesh (TM) or Regular Lattice (RL).*

## 6.6 Conclusion

This chapter has presented the proposed depth estimation system for creating depth maps from multiple-view videos, which consists of two major steps: (1) *sparse reconstruction* to calibrate cameras and reconstruct a sparse 3D scene model, and (2) *dense reconstruction* to create accurate depth maps for the selected view. The sparse reconstruction involves multiple steps like feature point matching and tracking, factorization-based projective and Euclidean reconstruction, and triangulation, which are common with the proposed 3D modeling that was presented in preceding chapters. The new step in sparse reconstruction is to select the best Multiple View Frame (MVF) composed of all views taken by multiple cameras. The purpose of selecting the best MVF is to ensure that the selected MVF is free of critical configurations and cameras can be accurately calibrated. To this end, we have presented Alg. 6.2.1 which exploits both the rank-ness of the scaled measurement matrix and the re-projection error to measure the 'criticalness' of the scene and camera configurations.

Based on the calibrated cameras and the reconstructed 3D points in the first step, the second step concentrates on creating dense per-pixel depth maps from multiple-view videos. This step uses the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF) to infer the best depth labeling, which is based on three algorithm steps. These algorithm steps are graph reconstruction, energy function definition and coarse-to-fine energy minimization. In *graph reconstruction*, we construct 2D triangular meshes on over-segmentation maps so that the constructed triangles are well aligned with object boundaries. In *energy function definition*, multiple depth cues are combined to define local penalty and interaction energies, which are thereafter adapted to the local image content. The use of multiple depth cues and the adaptation of local energies increase the accuracy of the resulting depth maps. In *energy minimization*, the depth accuracy of each node is refined in multiple optimization passes, where each pass uses the labeling results from the preceding pass as the initial labeling. This increases both the computation efficiency and robustness of the labeling process, because each node has less depth labels to choose from and therefore the optimization increasingly avoids trapping in local minima.

The proposed depth estimation system has been evaluated for both the multiple-view images and the multiple-view videos for depth estimation. The experimental results show that the proposed depth reconstruction algorithm is able to create the accurate per-pixel depth maps automatically. Given multiple-view videos, depth maps for all MVFs can be created automatically without any human intervention. With respect to the quality of the resulting depth maps, we have found that object boundaries are well preserved. Although depth maps are computed automatically and computation time has been reduced because of the coarse-to-fine depth labeling, we have noticed that labeling a large number of nodes from a large number of depth labels using energy minimization remains computationally expensive. The computational complexity is not evaluated in this chapter, since our focus is on the robustness and the automation of the process. Moreover, despite the improvement of the robustness by the hierarchical labeling algorithm, the accuracies of the depth maps for individual MVFs are not always at the same level. For some special MVFs, e.g. when the scene content changes abruptly with dynamic objects, the scene configuration may become more critical, which may deteriorate the accuracy of the resulting depth maps.

With the presented depth estimation system, this chapter completes our presentation about computing 3D scene information from multiple-view 2D images. The proposed depth estimation system uses several building blocks from the 3D modeling system introduced in earlier chapters, which is efficient for constructing a complete framework for computing the sparse 3D scene model and the dense depth information from multiple-view images. The obtained depth maps have been used for rendering the left and right views on an autostereoscopic 3DTV display. Visual inspection of the rendering images shows a better visual quality compared with the state-of-the-art depth estimation methods, especially with regard to the quality of object boundaries and quality stability (consistency) over time. One method for comparison is based on applying bilateral filtering, which potentially may give problems at object boundaries. The coarse-to-fine energy minimization using content-adaptive local energies based on multiple depth cues increases the accuracy and the stability of the resulting depth maps.

CHAPTER 7

# Conclusions

## 7.1 Brief summary of our work

Obtaining 3D scene information from multiple-view image data has a wide range of applications such as 3DTV, 3D visualization, 3D gaming and surveillance, which receives an increasing amount of attention from both academia and industry. Applications of this technology such as 3D movies, 3DTV and 3D games are now available in the consumer market. It is expected that more 3D applications will emerge in the near future. This thesis is directly related to this ongoing developments, and attempts to robustly automate the production of 3D content. This involves the computation of 3D scene information from multiple-view image data. For this computation, we have proposed the following two systems.

1. *A 3D modeling system for automatically reconstructing sparse 3D scene geometry using long image sequences.*

   The system automatically reconstructs a sparse 3D model of a complete scene from long video sequences captured with a hand-held consumer camcorder, which supports the low-cost production of the 3D content using personal computers. Experimental results on real sequences of hundreds and thousands of frames have demonstrated the effectiveness of the proposed system for scene reconstruction. The use of consumer devices supports the low-cost production for future 3D applications such as 3D visualization of our living environment, 3D games, etc.

2. *A depth estimation system for automatically creating depth maps from multiple-view videos.*

   The depth estimation system automatically computes per-pixel depth maps from multiple-view videos, which can be used to render the left and right views of a scene for 3DTV.

Visual inspection of the rendering images shows a better visual quality compared with the state-of-the-arts depth estimation methods, especially with regard to the quality of object boundaries and quality stability (consistency) over time.

Apart from designing and implementing the two proposed systems, we have also developed three key scientific contributions to enable the two proposed 3D reconstruction systems, which are summarized in Section 7.3. In the next suction, we will briefly recapitalize the content of each chapter.

## 7.2 Recapitalization of individual chapters

**Chapter 1: Introduction**

This chapter provides a technical background to 3D reconstruction and motivates the chosen direction of our research. The research objectives, challenges and contributions are presented. The objectives focus on creating of two 3D reconstruction systems presented in the previous section. The two major research challenges for the 3D modeling system are to match and track feature points and to handle critical configurations where the factorization-based reconstruction degenerates. The two major research challenges for the depth reconstruction system are to accurately reconstruct the object boundaries and to improve the robustness of the processing. Afterwards, research contributions corresponding to those challenges are indicated. The chapter ends with an outline of the thesis and scientific background of each chapter.

**Chapter 2: System overview and related work**

This chapter presents the overview of the two aforementioned 3D reconstruction systems studied in this thesis. A survey of existing 3D acquisition methods are presented, where a special discussion is devoted to the 3D-from-image approach that is adopted in this thesis. Subsequently, the design objectives and requirements of the two systems are described, which mainly aim at an automated reconstruction of 3D scene information from multiple-view images. Afterwards, the major modules of the two systems are presented such as feature point matching and tracking, factorization-based camera calibration and triangulation. Processing modules which are common between the two systems are identified. Prior work such as the merging method for structure and motion and the energy minimization approach for depth labeling are discussed.

**Chapter 3: Factorization-based scene reconstruction from long sequences**

This chapter gives a detailed presentation of the 3D modeling system which has been briefly introduced in the previous chapter. First, we introduce the mathematical formulation of the problem of multiple-view scene reconstruction. After that, we present a number of improvements that are made to the individual components of the proposed 3D modeling system. The improvements involve (1) removing blur and abrupt-motion frames for better feature point matching, (2) content-adaptive Harris corner detection to obtain more evenly distributed feature points, (3) a triangulation scheme to triangulate a large number of 3D points while minimizing redundant triangulations, and (4) merging partial reconstructions to obtain a complete

3D model. Finally, experimental results using two real video sequences are presented to demonstrate the effectiveness of the proposed 3D modeling system for an automatic scene reconstruction from long sequences. The experimental results on the short sequence show that the proposed system is able to accurately reconstruct the 3D scene from multiple-view data. The accuracy evaluation on the short sequence reveals that the maximum error of angles and lengths is $3.6\%$ and $4.7\%$, respectively. The key result of the long sequence reconstruction is that, with our algorithm, it is now possible to automatically divide a long sequence into subsequences, and then merge the partial reconstructions into the same coordinate system.

**Chapter 4: Texture-independent feature point matching**

This chapter has presented a novel Texture-Independent Feature point Matching (TIFM) algorithm for tracking feature points over successive frames in a video or image sequence with limited image motion or pixel shifts. Our TIFM algorithm uses only a smoothness constraint for feature point matching. By using this constraint, all feature points within a local neighborhood are collectively matched by maximizing the local motion smoothness, using a RANSAC-like process. Due to the single use of the smoothness constraint, TIFM provides two major advantages. The first advantage is that TIFM is robust to illumination changes because of several reasons. The smoothness assumption is valid for most local neighborhoods, providing the robustness to camera motion and scene structure. The second advantage is that TIFM is able to track a large number of feature points along a long sequence of frames. The extensive experimental results have shown a significantly improved performance when matching and tracking feature points in video sequences with limited motion. Experimental results demonstrate that TIFM is able to track at least twice the number of points compared with SIFT, KLT and BM, with a comparable or higher accuracy.

**Chapter 5: Dividing long sequences for factorization-based structure and motion**

This chapter proposes algorithms for dividing long sequences with the consideration of so-called critical configurations where the factorization method degenerates. First, we introduce the projective reconstruction and camera calibration algorithms that are used in this thesis. Second, we propose algorithms to detect the following critical configurations where the factorization method is not possible: (1) coplanar 3D points, (2) pure rotation, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. The configurations in cases of (1), (2) and (4) will affect the rank of the Scaled Measurement Matrix (SMM). The number of camera centers in case of (3) will affect the number of independent rows of the SMM. By examining the rank and the row space of the SMM, we detect the above-mentioned critical configurations. Third, a sequence-dividing algorithm is proposed to automatically divide a long sequence into subsequences such that a successful SaM can be obtained on each subsequence with a high confidence. Experimental results on both synthetic and real sequences are presented to demonstrate the effectiveness of the proposed algorithm for dividing a long sequence for an automatic 3D reconstruction. We have found in the experiments that the rank-3-ness, rank-4-ness and their corresponding reprojection errors, and the counting of the number of distinct camera centers, provide a good set of metrics for detecting the critical configurations and for dividing a long sequence.

**Chapter 6: Estimating depth map from multiple-view video**

This chapter presents the proposed depth estimation system for creating depth maps from multiple-view videos, which consists of two major steps: (1) *sparse reconstruction* to calibrate cameras and reconstruct a sparse 3D scene model, and (2) *dense reconstruction* to create accurate depth maps for the selected view. The sparse reconstruction involves multiple steps from earlier chapters. The new step is to select the best Multiple View Frame (MVF) composed of all views taken by multiple cameras, in order to ensure that the selected MVF is free of critical configurations and cameras can be accurately calibrated. To this end, we have presented an algorithm which exploits both the rank-ness and the re-projection error to measure the 'criticalness' of the scene and camera configurations. The second step concentrates on creating dense per-pixel depth maps from multiple-view videos. This step uses the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF) to infer the best depth labeling, which is based on three algorithm steps. These algorithm steps are graph reconstruction, energy function definition and coarse-to-fine energy minimization. The proposed depth estimation system has been evaluated on multiple-view data. The experimental results show that the proposed depth reconstruction algorithm is able to create the accurate per-pixel depth maps automatically. With respect to the quality of the resulting depth maps, we have found that object boundaries are well preserved. Although depth maps are computed automatically and computation time has been reduced because of the coarse-to-fine depth labeling, we have noticed that labeling a large number of nodes from a large number of depth labels using energy minimization remains computationally expensive.

## 7.3 Scientific contributions

In response to the research challenges pointed out in Sections 1.5.1 and 1.5.2, we have developed several algorithms and techniques for the proposed *3D modeling system* and the *depth estimation system*, and we have found the following three major contributions.

**A. Matching and tracking a large number of feature points for a long sequence of images**

This is the first and the most critical step of both of the two proposed systems, since finding feature point correspondences is the beginning of both two systems and it is a challenging task in itself, because matching 3D points based on only the 2D image signal is an ill-posed problem. For a robust and accurate camera calibration using the factorization method, it is crucial to match a large number of correspondences along a long sequence of images.

In Chapter 4, we have presented a novel Texture-Independent Feature point Matching (TIFM) algorithm that is designed to match and track feature points in image/video sequences where the image motion is limited. The employed smoothness assumption is not only valid but also robust for most images with limited image motion, regardless of the camera motion and scene structure. Because of this robust smoothness constraint, the TIFM algorithm obtains two major advantages. First, the algorithm is robust to illumination changes, as the employed smoothness constraint does not rely on any texture information. Second, the algorithm has a good capability to handle the drift of the feature points over time, as the drift can hardly lead to a violation of the smoothness constraint because the local motion field stays smooth even though feature points can drift over frames. These benefits lead to the large number of feature points matched and tracked by the proposed algorithm, which significantly

helps the subsequent 3D modeling process. Our extensive experimental results show that the proposed algorithm is able to track at least 2.5 times as many feature points compared to state-of-the-art algorithms, with a comparable or higher accuracy. This contributes significantly to the robustness of the overall 3D reconstruction process.

This thesis has also shown that the heuristic smoothness constraint can be converted into a quantitative criterion to maximize the local motion smoothness for feature point matching, which has enabled us to build the framework for feature point matching using only the smoothness constraint. Such a framework has not been found in literature.

Unfortunately, the smoothness constraint is a heuristic constraint of which the validity is limited to certain conditions. For example, if a sequence contains very large non-translational motion or the repetition ratios of feature points is very low, the TIFM will not work, since the smoothness assumption is no longer valid in these cases. However, as demonstrated by our extensive experiments, TIFM works very well for the limited-motion sequences, which are frequently used in practice.

**B. Splitting long sequence by detecting critical configurations**

Video sequences captured by a camcorder contain hundreds or even thousands of frames. Such long sequences have to be divided into short subsequences for partial reconstructions using the factorization method. After that, individual partial reconstructions have to be merged to obtain a complete 3D model of the scene. To divide a long sequence, we have developed algorithms to detect critical configurations where the factorization-based 3D reconstruction degenerates. Based on the degeneracy detection, a sequence-dividing algorithm has been developed to divide a long sequence into subsequences, such that successful 3D reconstructions can be performed on individual subsequences with a high confidence. The partial reconstructions are merged later to obtain the 3D model of the complete scene.

In the critical configuration detection algorithm, four critical configurations are detected: (1) coplanar 3D scene points, (2) pure camera rotation, (3) rotation around two camera centers, and (4) presence of excessive noise and outliers in the measurements. The configurations in cases (1), (2) and (4) will affect the rank of the Scaled Measurement Matrix (SMM). The number of camera centers in case (3) will affect the number of independent rows of the SMM. The elegance of our proposed solution is that, by examining the rank and the row space of the SMM, the four aforementioned critical configurations are detected. Based on the detection results, the proposed sequence-dividing algorithm partitions a long sequence into subsequences, such that each subsequence is free of the four critical configurations, so that successful 3D reconstructions for individual subsequences can be obtained. Experimental results on both synthetic and real sequences have demonstrated that the above four critical configurations are robustly detected, and a long sequence of thousands of frames is automatically divided into subsequences, yielding successful 3D reconstructions. The proposed critical configuration detection and sequence-dividing algorithms provide an essential processing block for an automatical 3D reconstruction on long sequences.

As discussed in Chapter 5, the proposed critical configuration algorithm has its inherent limitations. First, when the data contain noise and outliers, the proposed rank analysis technique may not be able to distinguish the real cause of the degeneracy of the configuration. Second, the rank analysis using the Weighted Iterative Eigen (WIE) algorithm does

not always converge to sensible results, which may lead to a false detection. Furthermore, the proposed algorithm is designed to detect only the four critical configurations. If a sequence contains other critical configurations, the dividing algorithm will fail to provide a good partitioning. Luckily, these situations occur rarely in practice. As demonstrated by the experimental results on both synthetic and real data, the rank-3-ness, rank-4-ness and their corresponding re-projection errors, and the counting of the number of distinct camera centers, provide a good set of metrics for detecting the critical configurations for dividing a long sequence.

### C. Preserving depth discontinuities and improving robustness of energy minimization

The point cloud obtained by the factorization method is not sufficient for applications such as 3DTV, where per-pixel depth maps are required. Therefore, the density insufficiency of the point cloud has to be addressed and depth holes have to be filled.

As a solution, in Chapter 6, we have proposed a coarse-to-fine depth labeling algorithm to compute depth maps from multiple-view videos, where the accuracy of resulting depth maps is gradually refined in multiple optimization passes. In the proposed algorithm, multiple-view depth reconstruction is formulated as an image-based labeling problem using the framework of Maximum A Posterior (MAP) on Markov Random Fields (MRF). The MAP-MRF framework allows the combination of various objective and heuristic depth cues to define the local penalty and the interaction energies, which provides a straightforward and computationally tractable formulation of depth labeling. Furthermore, the global optimal MAP solution to depth labeling can be found by minimizing the local energies, using existing MRF optimization algorithms.

The proposed algorithm contains the following three key features. (1) A graph construction algorithm to construct triangular meshes on over-segmentation maps, in order to exploit the color and the texture information for depth labeling. (2) Multiple depth cues are combined to define the local energies. Furthermore, the local energies are adapted to the local image content, in order to consider the varying nature of the image content for an accurate depth labeling. (3) Both the density of the graph nodes and the intervals of the depth labels are gradually refined in multiple labeling passes. By doing so, both the computational efficiency and the robustness of the depth labeling process are improved due to the coarse-to-fine depth labeling. The experimental results on real multiple-view videos show that the depth maps of the selected reference camera view are accurately reconstructed, and depth discontinuities are well preserved, because of the segmentation-driven graph construction and the combination of multiple depth cues for defining the image content adaptive local energies.

Although depth maps are computed automatically and computation time has been reduced because of the coarse-to-fine depth labeling, we have noticed that labeling a large number of nodes from a large number of depth labels using energy minimization remains computationally expensive. Moreover, the accuracies of the depth maps for individual Multiple View Frames (MVFs) sometimes deteriorate for some special MVFs, e.g. when the scene content changes abruptly with dynamic objects, so that the scene configuration may become more critical.

**D. Minor contributions**

Besides the above major contributions, we have proposed a number of other contributions to enable the two proposed multiple-view reconstruction systems. Some of the contributions are listed below.

1. *Harris corner detection with content-adaptive Harris threshold*

   Feature points detected by the original Harris corner detector are not evenly distributed over an image where the image texture varies significantly over the image. This makes the subsequent feature point matching and projective reconstruction difficult, since the reconstruction process is less constrained if 3D points are clustered in a small 3D space. To overcome this problem, a simple but effective algorithm has been proposed in Chapter 3 to make the detected feature points more evenly distributed over an image. The idea is simple: we first divide an image into a number of small blocks, and then try to detect an equal number of feature points in individual blocks by adjusting the threshold.

2. *Blur-and-abrupt frame removal*

   A long video taken by a hand-held consumer camcorder inevitably contains blur images and abrupt image motions. Such blur-and-abrupt frames increase the difficulty to match and track feature points across consecutive frames and need to removed. The proposed algorithms for blur-and-abrupt-frame removal employ the fact that the blur-and-abrupt frames will reduce the number of detected feature points and the number of matched feature points. Thus, by looking into the number of feature points detected in individual frames and by inspecting the number of matched feature points between two consecutive frames, blur-and-abrupt frames can be detected.

3. *Hierarchical triangulation scheme*

   After cameras are calibrated, we wish to triangulate as many 3D points as possible from the available feature point tracks, while at the same time minimize the redundant triangulations. Thus, a hierarchical triangulations scheme is proposed where 3D points are first triangulated from long feature point tracks and then from short feature point tracks. Redundant triangulations are removed by removing 3D points that are projected onto the same feature points in an image. By doing this, we are able to triangulate a large number of 3D points with a high accuracy, since redundant triangulations are reduced to a largest extent.

4. *Merging partial reconstructions*

   Partial reconstructions from individual subsequences by the factorization method are aligned with different coordinate frames. To obtain a complete model of a scene, partial reconstructions need to be aligned into a single coordinate frame. In the proposed algorithm, the reconstructed point clouds of two neighboring subsequences are registered based on the point-to-point correspondences between two sets of 3D points.

The above-listed contributions are also essential for the two proposed 3D reconstruction systems. However, since they are not the primary focus of this thesis, quantitative evaluation

of these individual algorithms are not presented. Their performance is taken into account in the overall results that have been presented in Chapters 3 and 6.

## 7.4 Future work

This section discusses a number of possible future extensions to improve the two proposed systems.

### A. Feature point matching

The Texture-Independent Feature point Matching (TIFM) algorithm proposed in Chapter 4 assumes that the local image motion is translational, which is valid for most images with limited image motion. However, for images with large rotation or scaling, the assumption of local translational motion is no longer valid. This leads to the failure of the proposed TIFM algorithm on images with large non-translational image motion. As a future work, we can look at how to explicitly consider the non-translational motion when we maximize the local motion smoothness, in order to improve the robustness of the TIFM algorithm.

### B. Detecting critical motion and surfaces

In Chapter 5, quantitative metrics have been proposed to detect several critical configurations where the factorization method degenerates. Despite this effort, still it cannot be guaranteed that the obtained subsequences will yield an successful Euclidean reconstruction. Other critical configurations than the discussed four configurations cannot be detected by the proposed algorithms. For example, if the camera undergoes a pure translation without rotation, Euclidean reconstruction will degenerate. However, the proposed metrics will not be able to detect it. It will be worthwhile to look into the detection of other critical configurations, in order to better divide a long sequence.

### C. Merging partial reconstructions

In Chapter 3, partial reconstructions are merged into a single coordinate frame by registrating 3D points that are common for two neighboring subsequences. As shown by Fig. 3.17 in Chapter 3, clear gaps between two groups of cameras can be observed for neighboring subsequences. The reason is that, although the factorization method is able to optimize the reconstruction on individual subsequences due to the uniform use of all feature points, the overall merged results are not optimized. Only a small set of local data (common 3D points) are used for the merging. Thus, global optimization techniques such as bundle adjustment is expected to improve the merged results significantly.

### D. Depth labeling via energy minimization

As pointed out in Chapter 6, one big advantage of the energy-minimization approach is that various objective and heuristic depth cues can be combined into a uniform energy-minimization framework. The optimized solution can be obtained via algorithms such as

graph cut. In our proposed algorithm, the slanting cue, no-floating-object assumption, photo-consistency constraint, etc. are used. Certainly more depth cues than the above-mentioned cues can be used, like the texture and color of an image. Since the triangular meshes are constructed on over-segmentation maps, it makes it easier to incorporate the image texture and color information into the energy-minimization framework. As was partly done in the proposed algorithm for adapting the local energies to local image content, it is worthwhile to look into how depth discontinuities can be better preserved by using more depth cues.

## E. Reducing computational complexity via parallel processing

Both of the two proposed 3D reconstruction systems are targeted for automatic off-line processing. Thus, little attention has been paid to improve the processing speed of the reconstruction process, even though the processing speed of many of the proposed algorithms can be improved using parallel processing techniques such as data partitioning. For example, there is a large amount of data parallelism in components such as feature point detection, feature point matching, sequence dividing, etc. Parallel implementation of these components is expected to significantly improve the processing speed.

The above points indicate that obtaining a perfect 3D reconstruction is a sliding target, of which the applicability still has to be validated. Some of the above extensions are more important than others. In particular, the author has the opinion that the merging of partial reconstructions is the most critical step for the successful deployment of the 3D modeling system for long sequences. However, in practice, it occurs often that the complex framework as described in this thesis can be solved by new technology breakthroughs. In this particular case, given the current rapid development of sensor technologies, it may well be that an economic depth sensing technique is found, so that the geometry reconstruction will be based on actual measurements.

APPENDIX A

# Appendix: factorization method

Using the notations introduced in Section 3.2.3, this appendix introduces the factorization method [20] to compute the $4\times4$ projective transformation $\mathbf{H}$, such that the projective motion $\hat{\mathbf{P}}_i$ and shape $\hat{\mathbf{X}}_j$ can be upgraded to the Euclidean motion $\mathbf{P}_i$ and shape $\mathbf{X}_j$, satisfying

$$\mathbf{P}_i = \hat{\mathbf{P}}_i\mathbf{H} \quad \text{and} \quad \mathbf{X}_j = \mathbf{H}^{-1}\hat{\mathbf{X}}_j. \tag{A.1}$$

Let us denote

$$\mathbf{H} = [\mathbf{A} \;\; \mathbf{B}], \tag{A.2}$$

where $\mathbf{A}$ is a $4 \times 3$ matrix and $\mathbf{B}$ is a $4 \times 1$ vector. The algorithm described here is to compute $\mathbf{A}$ and $\mathbf{B}$, such that the Euclidean reconstruction can be computed from the projective reconstruction.

## A.1 Solving B

We repeat the projection equation Eq. (3.2) here and write it into Eq. (A.3):

$$\lambda_{ij}\mathbf{x}_{ij} = \mathbf{P}_i\mathbf{X}_j, \tag{A.3}$$

where

$$\mathbf{P}_i = \alpha_i\mathbf{K}_i\mathbf{R}_i[\mathbf{I} \;\; -\mathbf{C}_i] = [\alpha_i\mathbf{K}_i\mathbf{R}_i \;\; -\alpha_i\mathbf{K}_i\mathbf{R}_i\mathbf{C}_i] = [\mathbf{M}_i \;\; \mathbf{T}_i], \tag{A.4}$$

and

$$\mathbf{X}_j = \beta_j(X_j, Y_j, Z_j, 1)^T = \beta_j(\overline{\mathbf{X}}_j^T, 1)^T. \tag{A.5}$$

and

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & u_{0i} \\ 0 & \upsilon_i f_i & v_{0i} \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}_i = \begin{bmatrix} \mathbf{i}_i^T \\ \mathbf{j}_i^T \\ \mathbf{k}_i^T \end{bmatrix} \quad \mathbf{C}_i = \begin{bmatrix} C_{xi} \\ C_{yi} \\ C_{zi} \end{bmatrix} \quad \overline{\mathbf{X}}_j = \begin{bmatrix} X_{xj} \\ Y_{yj} \\ Z_{zj} \end{bmatrix}. \tag{A.6}$$

In the above equations, matrix $\mathbf{K}_i$ is the internal camera matrix that contains all the internal parameters of camera $i$, where $f_i$ represents the focal length, coordinates $(u_{0i}, v_{0i})$ denote the image coordinates of the principal point, and $\upsilon_i$ represents the aspect ratio. Matrix $\mathbf{R}_i$ is the orientation matrix of camera $i$, with $\mathbf{i}_i, \mathbf{j}_i$ and $\mathbf{k}_i$ representing the orientations of the three camera axes in the world coordinate system. Vector $\mathbf{C}_i$ represents the coordinates of the camera center, and vector $\overline{\mathbf{X}}_j$ represents the coordinates of the 3D point $j$.

From Eq. (A.6) and Eq. (A.4), we derive

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{m}_{xi}^T \\ \mathbf{m}_{yi}^T \\ \mathbf{m}_{zi}^T \end{bmatrix} = \begin{bmatrix} \alpha_i f_i \mathbf{i}_i^T + \alpha_i u_{0i} \mathbf{k}_i^T \\ \alpha_i \upsilon_i f_i \mathbf{j}_i^T + \alpha_i v_{0i} \mathbf{k}_i^T \\ \alpha_i \mathbf{k}_i^T \end{bmatrix} \quad \text{and} \quad \mathbf{T}_i = \begin{bmatrix} T_{xi} \\ T_{yi} \\ T_{zi} \end{bmatrix} \tag{A.7}$$

Substituting Eq. (A.7) and Eq. (A.6) into Eq. (A.3), we obtain

$$\begin{bmatrix} \lambda_{ij} u_{ij} \\ \lambda_{ij} v_{ij} \\ \lambda_{ij} \end{bmatrix} = \beta_j \begin{bmatrix} \mathbf{m}_{xi}^T & T_{xi} \\ \mathbf{m}_{yi}^T & T_{yi} \\ \mathbf{m}_{zi}^T & T_{zi} \end{bmatrix} \begin{bmatrix} \overline{\mathbf{X}}_j \\ 1 \end{bmatrix} = \begin{bmatrix} \beta_j \mathbf{m}_{xi}^T \overline{\mathbf{X}}_j + \beta_j T_{xi} \\ \beta_j \mathbf{m}_{yi}^T \overline{\mathbf{X}}_j + \beta_j T_{yi} \\ \beta_j \mathbf{m}_{zi}^T \overline{\mathbf{X}}_j + \beta_j T_{zi} \end{bmatrix} \tag{A.8}$$

Assuming that the world coordinate system is positioned at the center of the gravity of all scaled 3D points, we have

$$\sum_{j=1}^{n} \beta_j \overline{\mathbf{X}}_j = 0. \tag{A.9}$$

With the above equation, we derive the following equation from Eq. (A.8).

$$\begin{bmatrix} \sum_{j=1}^n \lambda_{ij} u_{ij} \\ \sum_{j=1}^n \lambda_{ij} v_{ij} \\ \sum_{j=1}^n \lambda_{ij} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_{xi}^T \sum_{j=1}^n (\beta_j \overline{\mathbf{X}}_j) + \sum_{j=1}^n (\beta_j T_{xi}) \\ \mathbf{m}_{yi}^T \sum_{j=1}^n (\beta_j \overline{\mathbf{X}}_j) + \sum_{j=1}^n (\beta_j T_{yi}) \\ \mathbf{m}_{zi}^T \sum_{j=1}^n (\beta_j \overline{\mathbf{X}}_j) + \sum_{j=1}^n (\beta_j T_{zi}) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^n (\beta_j T_{xi}) \\ \sum_{j=1}^n (\beta_j T_{yi}) \\ \sum_{j=1}^n (\beta_j T_{zi}) \end{bmatrix} \tag{A.10}$$

Thus, we have

$$\frac{T_{xi}}{T_{zi}} = \frac{\sum_{j=1}^n \lambda_{ij} u_{ij}}{\sum_{j=1}^n \lambda_{ij}} \text{ and } \frac{T_{yi}}{T_{zi}} = \frac{\sum_{j=1}^n \lambda_{ij} v_{ij}}{\sum_{j=1}^n \lambda_{ij}} \tag{A.11}$$

Since $\mathbf{P}_i = \hat{\mathbf{P}}_i \mathbf{H}$, as in Eq. (A.1), we obtain

$$[\mathbf{M}_i \;\; \mathbf{T}_i] = \hat{\mathbf{P}}_i [\mathbf{A} \;\; \mathbf{B}] \tag{A.12}$$

and therefore

$$T_{xi} = \hat{\mathbf{P}}_{xi} \mathbf{B}, \quad T_{yi} = \hat{\mathbf{P}}_{yi} \mathbf{B}, \quad \text{and } T_{zi} = \hat{\mathbf{P}}_{zi} \mathbf{B}. \tag{A.13}$$

From Eqs. (A.11) and (A.13), we set up $2n$ linear equations to solve the 4 elements of vector $\mathbf{B}$. Thus, $\mathbf{B}$ can be solved using linear minimum squares. The solution of $\mathbf{A}$ is given in Section 5.2.4. In the following, we explain how to derive the calibration constraint in Eq. (5.11).

## A.2 Solving A

Assuming the rotation axes of a camera are orthogonal, we obtain the following constraints from Eq. (A.7).

$$\begin{aligned}
\mathbf{m}_{xi} \cdot \mathbf{m}_{xi} &= \alpha_i^2 f_i^2 + \alpha_i^2 u_{0i}^2 \\
\mathbf{m}_{yi} \cdot \mathbf{m}_{yi} &= \alpha_i^2 \upsilon_i^2 f_i^2 + \alpha_i^2 v_{0i}^2 \\
\mathbf{m}_{zi} \cdot \mathbf{m}_{zi} &= \alpha_i^2 \\
\mathbf{m}_{xi} \cdot \mathbf{m}_{yi} &= \alpha_i^2 u_{0i} v_{0i} \\
\mathbf{m}_{xi} \cdot \mathbf{m}_{zi} &= \alpha_i^2 u_{0i} \\
\mathbf{m}_{yi} \cdot \mathbf{m}_{zi} &= \alpha_i^2 v_{0i}
\end{aligned} \tag{A.14}$$

With the assumption that the principal point is at the origin $u_{0i} = v_{0i} = 0$, and the aspect ratio equals unity $\upsilon_i = 1$, it can be verified that

$$\mathbf{m}_{xi} \cdot \mathbf{m}_{xi} = \mathbf{m}_{yi} \cdot \mathbf{m}_{yi} \quad \text{and} \quad \mathbf{m}_{xi}^T \mathbf{m}_{yi} = \mathbf{m}_{xi}^T \mathbf{m}_{zi} = \mathbf{m}_{yi}^T \mathbf{m}_{zi} = 0. \tag{A.15}$$

From Eq. (A.12), we obtain $\mathbf{M}_i = \hat{\mathbf{P}}_i \mathbf{A}$. Thus, we have

$$\mathbf{M}_i \mathbf{M}_i^T = \hat{\mathbf{P}}_i \mathbf{A} \mathbf{A}^T \hat{\mathbf{P}}_i = \hat{\mathbf{P}}_i \mathbf{Q} \hat{\mathbf{P}}_i. \tag{A.16}$$

As can be observed, Eq. (A.15) provides four linear constraints on the 10 element of the symmetric matrix $\mathbf{Q}$. Knowing the projective motion matrix $\hat{\mathbf{P}}_i$, matrix $\mathbf{Q}$ can be solved if we have more than three distinct camera centers, as discussed in Section 5.2.4. After that, matrix $\mathbf{A}$ can be obtained using rank 3 matrix decomposition.

# References


[1] Bayon digital archival. In *http://www.cvl.iis.u-tokyo.ac.jp/research/bayon/*.

[2] Mrf software. In *http://vision.middlebury.edu/MRF/code/*.

[3] `castle` sequence. In *http://www.cs.unc.edu/~marc*.

[4] Adam Baumberg. Reliable feature matching across widely separated videws. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 774–781, 2000.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proc. 9th European Conf. Computer Vision*, May 2006.

[6] Houqin Bian and Jianbo Su. Feature matching based on geometric constraints in stereo views of curved scenes. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 313–318, 2005.

[7] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sep. 2004.

[8] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov. 2003.

[9] M.K. Chandraker, S. Agarwal, F. Kahl, D. Nister, and D.J. Kriegman. Autocalibration via rank-constrained estimation of the absolute quadric. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[10] Qian Chen and Gerard Medioni. Efficient iterative solution to m-view projective reconstruction problem. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 55–61, 1999.

[11] Ouk Choi and In So Kweon. Reducing ambiguity in feature point matching by preserving local geometric consistency. In *Proc. Int. Conf. Image Processing*, pages 293–296, Oct. 2008.

[12] Nico Cornelis, Bastian Leibe, Kurt Cornelis, and Luc Van Gool. 3d urban scene modeling integrating recognition and reconstruction. *Int. J. Computer Vision*, 78:121–141, 2008.

[13] Marc Op de Beeck, Etienne Fert, Christoph Fehn, and Peter Kauff. Broadcast requirements on 3d video coding. MPEG02/M8040, Mar. 2002.

[14] Elena Stoykova et. al. 3d time-varying scene capture technologies - a survey. *IEEE Trans. Circuits and Systems for Video Technology*, 17(11):1568–1586, Nov. 2007.

[15] Richard Szeliski et. al. A comparative study of energy minimization methods for markov random fields. In *Proc. European Conf. Computer Vision*, volume LNCS 3952, pages 16–29, 2006.

[16] Dirk Farin. *Automatic Video Segmentation Employing Object/Camera Modeling Techniques*. Technische Universiteit Eindhoven, ISBN: 90-386-2381-X, first edition, 2005.

[17] O. D. Faugeras, Q. T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. pages 321–334, 1992.

[18] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Computer Vision*, 59(2):167–181, Sep. 2004.

[19] Christian Fruh and Avideh Zakhor. An automated method for large-scale, ground-based city model acquisition. *IJCV*, 60(1):5–24, 2004.

[20] Mei Han and Takeo Kanade. A perspective factorization method for euclidean reconstruction with uncalibrated cameras. *J. Visual. Comput. Animat.*, 13(4):211–223, Sep. 2002.

[21] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conf.*, pages 147–151, 1988.

[22] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, Nov. 1997.

[23] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[24] Richard I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(10):1036–1041, 1994.

[25] Elsayed E. Hemayed. A survey of camera self-calibration. volume 0, page 351, 2003.

[26] Carlos Hernandez, George Vogiatzis, Gabriel J. Brostow, Bjorn Stenger, and Roberto Cipolla. Non-rigid photometric stereo with colored lights. In *Proc. Int. Conf. Computer Vision*, pages 1–8, 2007.

[27] A. Heyden, R. Berthilsson, and G. Sparr. An iterative factorization method for projective structure and motion from image sequences. *Image and Vision Computing*, 17:981–991, 1999.

[28] Xiaoping Hu and Narendra Ahuja. Matching point feature with ordered geometric rigidity, and disparity constraints. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(10):1041–1049, 1994.

[29] Y.S. Hung and W.K. Tand. Projective reconstruction from multiple views with minimization of 2d reprojection error. *Int. J. Computer Vision*, 66(3):305–317, 2006.

[30] M.R.M. Jenkin. Tracking three dimensional moving light displays. In *Proc. ACM SIGGRAPH/Sigart Interdisciplinary workshop on motion: representation and perception*, pages 66–70, 1983.

[31] Yoon-Yong Jung, Yong-Ho Hwang, and Hyun-Ki Hong. Frame grouping measure for factorization-based projective reconstruction. In *Proc. Int. Conf. Pattern Recogonition*, volume 4, pages 112–115, 2004.

[32] Fredrik Kahl. Critical motions and ambiguous eucldian reconstructions in autocalibration. In *Proc. Int. Conf. Computer Vision*, volume 1, pages 469–475, 1999.

[33] Takeo Kanade, Peter Rander, and P.J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *Immersive Telepresence*, pages 34–46, 1997.

[34] Kenichi Kanatani. Uncertainty modeling and model selection for geometric inference. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(10):1307–1319, 2004.

[35] Yasushi Kanazawa and Kenichi Kanatani. Robust image matching preserving global consistency. In *Proc. Austraila-Japan Advanced Workshop on Computer Vision*, pages 32–37, Sep. 2003.

[36] P. Kauff, N. Atzpadin, C. Fehn, M. Muller, O. Schreer, A. Smolic, and R. Tanger. Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability. *Signal Processing: Image Communication*, 22:217–234, 2007.

[37] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. European Conf. Computer Vision*, volume LNCS: 2352, pages 82–96, 2002.

[38] Stanford Computer Graphics Laboratory. Standford multiple-camera array. In `http://graphics.stanford.edu/projects/array/`.

[39] Franz Leberl and Michael Gruber. 3d-models of the human habitat for the internet. In *Proc. 5th Visigrapp*, volume IS, pages 7–15, 2009.

[40] Carlos Leung. *Efficient Methods for 3D Reconstruction from Multiple Images*. Ph.D. dissertation, University of Queensland, 2005.

[41] Ping Li, Dirk Farin, Rene Klein Gunnewiek, and Peter H. N. de With. On creating depth maps from monoscopic video using structure from motion. In *Proc. 27th Symposium on Information Theory in the Benelux*, volume 1, pages 508–515, Jun 8–9th, 2006.

[42] Ping Li, Dirk Farin, Rene Klein Gunnewiek, and Peter H. N. de With. Contrast-invariant feature point correspondence. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, volume 1, pages I–477–I–480, Apr. 2007.

[43] Ping Li, Dirk Farin, Rene Klein Gunnewiek, and Peter H. N. de With. Texture-independent feature-point matching (tifm) from motion coherence. In *Lecture Notes in Computer Science: 8th Asian Conference on Computer Vision*, volume 4843/2007, pages 789–799, Nov. 2007.

[44] Ping Li, Rene Klein Gunnewiek, and Peter H. N. de With. Scene reconstruction using mrf optimization with image content adaptive energy functions. In *Proc. 29th Symposium on Information Theory in the Benelux*, May 2008.

[45] Ping Li, Rene Klein Gunnewiek, and Peter H. N. de With. Scene reconstruction using mrf optimization with image content adaptive energy functions. In *Proc. Advanced Concepts for Intelligent Vision Systems*, volume 5259 of *Lecture Notes in Computer Science*, pages 872–882. Springer, Oct. 2008.

[46] Ping Li, Rene Klein Gunnewiek, and Peter H.N. de With. Detecting critical configurations for dividing long image sequences for factorization-based 3-d scene reconstruction. In *Lecture Notes in Computer Science: 9th Asian Conference on Computer Vision*, volume 5995/2010, pages 381–394, Sep. 2009.

[47] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[48] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[49] Joao Maciel and Joao P. Costeira. A global solution to sparse correspondence problems. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(2):187–199, Feb. 2003.

[50] S. Mahamud, M. Hebert, Y. Omori, and J. Ponce. Provably-convergent iterative methods for projective structure from motion. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 1018–1025, 2001.

[51] Shyjan Mahamud and Martial Hebert. Iterative projective reconstruction from multiple views. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume II, pages 430–437, 2000.

[52] S. Maybank and O. D. Faugeras. A theory of selfcalibration of a moving camera. *Int. J. Computer Vision*, 8(2):123–151, Aug. 1992.

[53] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(10):1615–1629, Oct. 2005.

[54] Theo Moons, Maarten Vergauwen, and Luc Van Gool. *3D reconstruction from multiple images*. Lecture material of International Computer Vision Summer School, 2008.

[55] David Nister. *Automatic Dense Reconstruction from Uncalibrated Video Sequences*. Ph.D. dissertation, Kungl Tekniska Hogskolan, 2001.

[56] Alison Noble. Descriptions of image surfaces. In *PhD thesis, Department of Engineering Science, Oxford University*. 1989.

[57] T. Oelbaum. Subjective test results for the cfp on multi-view video coding (mvc). ISO/IEC M13009, Jan. 2006.

[58] Abhijit S. Ogale and Yiannis Aloimonos. Robust constrast invariant stereo correspondence. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 819–824, Apr. 2005.

[59] John Oliensis and Senior Member. Iterative extensions of the sturm/triggs algorithm: Convergence and nonconvergence. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(12):2217–2233, 2007.

[60] Sylvain Paris, Francois X. Sillion, and Long Quan. A surface reconstruction method using global graph cut optimization. *Int. J. Computer Vision*, 66(2):141–161, Feb. 2006.

[61] Andrew W. Fitzgibbon Philips H. S. Torr and Andrew Zisserman. The problem of degeneracy in structure and motion recovery from uncalibrated image sequences. In *MSR-TR-99-03*, volume 2, pages 1529–1533, Mar. 1999.

[62] Conrad Poelman and Takeo Kanade. A paraperspective factorization method for shape and recovery. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(3):206–218, Mar. 1997.

[63] M. Pollefeys and L. Van Gool. Stratified self-calibration with the modulus constraint. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(8):707–724, Aug. 1999.

[64] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Computer Vision*, 59(3):207–232, 2004.

[65] Marc Pollefeys, Luc Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. Computer Vision*, 59(3):207–232, 2004.

[66] Marc Pollefeys, David Nister, and Jan-Michael Frahm et al. Detailed real-time urban 3d reconstruction from video. *Int. J. Computer Vision*, 78:143–167, 2008.

[67] Sebastien Roy and Ingemar J. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *Proc. Int. Conf. Computer Vision*, pages 492–499, Jan. 1998.

[68] Joaquim Salvi, Xavier Armangue, and Joan Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35:1617–1635, 2002.

[69] Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets. In *Proc. European Conf. Computer Vision*, pages 414–431, May 2002.

[70] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. of Computer Vision*, 47:7–42, 2002.

[71] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. In *Proc. of the Royal Soc, London*, volume B–244, pages 21–26, 1991.

[72] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 1067–1073, Jun. 1997.

[73] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proc. European Conf. Computer Vision*, pages 709–720, 1996.

[74] Peter F Sturm. Critical motion sequences for the self-calibration of cameras and stereo systems with variable focal length. *Image and Vision Computing*, 20(5–6):415–426, 2002.

[75] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25(7):787–800, Jul. 2003.

[76] Rick Szeliski, Ramin Zabih, Daniel Scharstein, Olga Veksler, Vladimir Kolmogorov, Aseem Agarwala, Marshall Tappen, and Carsten Rother. A comparative study of energy minimization methods for markov random fields. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2007.

[77] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Tran. on Graphics*, 26(3):87:1–7, Jul. 2007.

[78] Carlo Tomasi and Takeo Kanade. Detecting and tracking of point features. *Carnegie Mellon University Technical Report CMU-CS-91-132*, Apr. 1991.

[79] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *Int. J. Computer Vision*, 9(2):1573–1405, Nov. 1992.

[80] Philip Torr. An assessment of information criteria for motion model selection. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 47–52, 1997.

[81] P.H.S. Torr, A. Zisserman, and S. Maybank. Robust detection of degenerate configurations whilst estimating the fundamental matrix. *Int. J. Computer Vision*, 71(3):312–333, 1998.

[82] Bill Triggs. Factorization methods for projective structure and motion. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996.

[83] Bill Triggs. Autocalibration and the absolute quadric. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 609–614, 1997.

[84] Roger Y. Tsai. A versatile camera calibration techniaue for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE J. Robotics and Automation*, RA-3(4):323–344, Aug. 1987.

[85] C.J. Veenman, M.J.T. Reinders, and E. Backer. Motion tracking as a constrained optimization problem. 36:2049–2067, 2003.

[86] G. Vogiatzis, P.H.S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 391–398, Jun. 2005.

[87] Yi-Sheng Yao and Rama Chellappa. Tracking a dynamic set of feature points. *IEEE Tran. Image Processing*, 4(10):1382–1395, Oct. 1995.

[88] Alan Yuille and Norberto Grzywacz. A mathematical analysis of the motion coherence theory. 3:155–175, 1989.

[89] Gang Zeng, Sylvain Paris, Long Quan, and Maxime Lhuillier. Surface reconstruction by propagating 3d stereo data in multiple 2d images. In *Proc. European Conf. Computer Vision*, pages 163–174, 2004.

[90] Jianguo Zhang, Marcin Marszalek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. Jounal of Computer Vision*, 73(2):213–238, Jun. 2007.

[91] Zhenyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[92] Jiang Yu Zheng and Min Shi. Scanning depth of route panorama based on stationary blur. *Int. J. Computer Vision*, 78(2-3):169–186, 2008.

[93] Qinfen Zheng and Rama Chellappa. Automatic feature point extraction and tracking in image sequences for arbitrary camera motion. *Int. J. Computer Vision*, 15(1-2):1573–1405, May 1995.

[94] Yefeng Zheng and David Doermann. Robust point matching for nonrigid shapes by preserving local neighborhood structures. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4):643–649, Apr. 2006.

[95] Zhigang Zhu and Takeo Kanade. Modeling and representations of large-scale 3d scenes. *Int. J. Computer Vision*, 78(2-3):119–120, 2008.

# Acknowledgement

After 4.5 years of full-time PhD research in TU/e, and 2 years of part-time thesis writing during nights and weekends while I was working for ASML, finally I can now enjoy this moment to write some words to thank all those who have helped me and supported me in my life in Netherlands.

First, I would like to offer my deepest gratitude to Rene Klein Gunnewiek, my supervisor in Philips Research. I thank you for your though guidance of my thesis and the nice arrangement of my working environment in the Video Processing Group at Philips Research. Unfortunately, Rene passed away in 2009, which I still deeply regret. May you have rest in heaven.

I would like to offer my sincerest gratitude to my supervisor, Prof. Peter de With. I thank you for giving me this great opportunity to do my PhD research in your group in the field of computer vision. You have supported me throughout my thesis with your knowledge and patience whilst allowing me the room to work in my own way. This thesis would have not been possible without your support. What I learned from you is not only academic, but also personality to pay attention to small details and the altitude to always pursue the best quality.

I would like to show my gratitude to Dr. Patrik Vandewalle, for agreeing to be my co-promotor in the last phase of my thesis. Thank you for your careful review and inspirational comments about my thesis, which have greatly improved my thesis.

Words of appreciation go to Dr. Dirk Farin, a former member of the VCA group, for shedding the first light on computer vision to me. In the first 2 years of my thesis, you answered most of my questions regarding with 3D reconstruction and Linux programming. Without your help, I would not have been able to get myself in the right direction so quickly when I entered this field.

This thesis is funded by the Dutch Freeband I-Share project in the BSIK framework, led by Prof. Inald Lagendijk. Inald, I want to thank you for your insightful comments that you provided during our regular project meetings. I am grateful to have you in my PhD committee.

Further thanks go to Prof. Kees Slump from University of Twente, Prof. Johan Lukkien and Prof. Gerard de Haan from Eindhoven University of Technology. It is my honor to have you in my PhD committee. Thank you for the improvements you have made to my thesis.

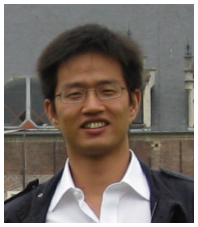
This thesis would have not been possible without the help from many of my colleagues and friends. Luc, thank you for translating the thesis summary into Dutch. Danqing, thank you for your advice on writing this acknowledgement. Ivo Liebregts, thank you for giving me the flexibility to use my holidays and after-work time to write my thesis, while I was working for ASML. Chenyang, JIANG Bin, GU Bing, SHU Feng, WU Yan, Wang Ting and Yide, thank you for your willingness to help me with my defense ceremony.

I spent 3-4 days per week in Philips Research Eindhoven for this thesis. For this, I have to thank Dr. Werner de Bruin, my office mate for around 4 years, for your help on reading so many of my Dutch letters, and for your patience and willingness to give me those scientific explanations about every bird that dropped by our office window. Your help and kindness have made my experience in Philips more memorable. I also want to thank my Chinese colleagues and friends in Philips, HU Hao, ZHAO Meng, SHAN Caifeng, ZUO Fei, WEI Xiuhong and SHAO Lin for all those interesting discussions during our lunches.

I spent 1-2 days every week in the TU/e VCA group for my thesis. For this, I have to thank my fellow colleagues from the VCA group for the nice working environment. Specially, I want to thank Dr. HAN Jungong, my table mate for 4.5 years, for numerous technical and social gossips in our mother language. I must say it cannot be more convenient to speak Chinese in the VCA group. Thanks also go to Goran Petrovic for those fruitful discussions during your smoking time, while I was drinking tea. Every time I went to the office on Saturday or Sunday afternoons, I expected you were there and you never disappointed me. Sjoerd, thank you for receiving me at the Eindhoven train station when I first arrived in the Netherlands in May 2005. Thanks also go to Ivo, Marijn, Rob, Lykele, Rob, Chrysi, Egor, Sveta, Weilun, Lingni, Bahman and Anja for your help and the happy time in the VCA group.

Last but not least, I want to thank my parents and sister for remotely supporting me throughout my life in the Netherlands. Your endless love and support are the source of the power for me to complete this thesis. I am proud of you.

# Curriculum Vitae Ping Li



Ping Li was born in Wuyi, Zhejiang, China. He received his B.Eng. and M.Eng. in Mechanical Engineering from the Xi'an Jiaotong University, China, in 1998 and 2000. He received his M.Eng. in Computer Engineering from the National University of Singapore in 2003. From March 2003 to July 2004, he was with the Institute for Infocomm Research, Singapore, working as a research engineer on H.264/AVC video coding. In October 2004, he joined the Philips Electronics Singapore as a software engineer, working on image content adaptive sharpness enhancement for LCD TV. Since May 2005, Ping Li was a Ph.D. student with the Video Coding and Architectures group at the Eindhoven University of Technology, the Netherlands, as well as the Video Processing Systems group at Philips Research, the Netherlands. His Ph.D. research project involves acquiring 3D scene information from 2D images. Since November 2009, Ping Li is with ASML, working on lithography printing.