KEY FRAME EXTRACTION AND BROWSER-BASED VISUALIZATION FOR 3D RECONSTRUCTION FROM VIDEO STREAMS

by

Mirza Tahir Ahmed

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering

Examination Committee:	Dr. Jose Luis Landabaso (Telefonica I+D Barcelona)
	Assist. Prof. Dr. Matthew N. Dailey (AIT)
	Prof. Dr. Alicia Casals (UPC)

Nationality:	Pakistani
Previous Degree:	Bachelor of Science in Computer Science
	COMSATS Institute of Information Technology
	Islamabad, Pakistan
Scholarship Donor:	Higher Education Commission of Pakistan
	Telefonica I+D Barcelona, Spain

Universitat Politècnica de Catalunya (UPC) & Asian Institute of Technology (AIT)
Facultat d'Informàtica de Barcelona & School of Engineering and Technology
Spain & Thailand
July 2009

Acknowledgment

In the name of Allah, the Gracious, the Merciful, I am thankful to Almighty Allah who enabled me to complete this research study.

I express my gratitude to Dr. Jose Luis Lanadabaso for the completion of this research study. His very friendly behavior, always encouraging personality, and motivation are the reasons for the completion of this research study. The valuable suggestions and motivation from Dr. Matthew N. Dailey and Prof. Alicia Casals Galpi are highly admirable. I am also very thankful to the Telefonica I+D, Barcelona team (Nicolas Herrero, Guillermo Gallego, and Jose Carlos). They are very encouraging and supporting friends. I gratefully acknowledge the Higher Education Commission of Pakistan for provide me support to pursue higher studies at the Asian Institute of Technology and Universitat Politècnica de Catalunya. I express my love and gratitude to Jumat Ahmadiyya, my parents, my wife and my siblings for their prayers, support, and endless love throughout the duration of my studies. I am also thankful to all my friends for providing help and encouragement to successfully complete this research.

Abstract

In computer vision, the process of 3D reconstruction consists of rebuilding a 3D scene from a set of images. The frames (images) can be extracted from a video sequence to reconstruct the captured scene. This research study concentrates on finding the suitable frames for reconstruction and discarding the rest of the frames through the process of key frame extraction. The selection process based on the properties of the frames avoids degenerate cases, helps to improve performance, and enables the 3D reconstruction process to minimize compute time. Besides key frame extraction, I developed a browser-based 3D video player to allow users to view the captured video in three dimensions.

Table of Contents

Chapter	Title	Page
	Title Page	i
	Acknowledgment	ii
	Abstract	iii
	Table of Contents	iv
	List of Figures	v
	List of Tables	vii
	List of Abbreviations	viii
1	Introduction	1
	1.1 Problem Statement	1
	1.2 Objectives	2
	1.3 Assumptions and Limitations	2
	1.4 Environment	2
	1.5 Organization	3
2	Literature Review	4
	2.1 Overview	4
	2.2 Structure From Motion	4
	2.3 Key Frame Extraction	17
	2.4 Browser-Based 3D Video Player	19
3	Methodology	22
	3.1 Overview	22
	3.2 System Architecture	22
	3.3 Back End Processing	22
	3.4 Front End GUI	33
4	Experimental Results	37
	4.1 Overview	37
	4.2 Degenerate Cases	37
	4.3 Baseline Distance	38
	4.4 Methods	42
5	Conclusion	46
	5.1 Overview	46
	5.2 Contribution	46
	5.3 Future Enhancement	46
References		47

List of Figures

Figure	Title	Page
1.1	Microsoft Photosynth. The cloud of 3D reconstructed points and captured frames are shown. Reprinted from Microsoft (2009).	2
2.1	Harris Detector. A patch is shifted by a very small amount in different directions to measure local changes in auto correlation. Points, at which the correlation is large, are selected as interest points.	5
2.2	Harris corner detection method. (a) On flat surfaces the result will remain the same as there is no change in the background. (b) The result on edge surface will show small variation. (c) The corner surface will show large variation. Therefore corners can be detected using the variation of the correlation of small window over image.	5
2.3	SURF. (a) Features location with reproducible orientations. (b) The Haar-wavelets used to calculate the orientation. (c) Keypoints with oriented descriptor calculation windows. Reprinted from Bay and Tuytelaars (2006).	6
2.4	Feature Matching. Matches between the two images are shown along with outliers. The images are captured from a historical building called Sagarada Familia in Barcelona.	7
2.5	Feature Matching. Matching between the pair of images are shown with inliers using the fundamental model. The images are captured from a historical building called Sagarada Familia.	8
2.6	Point correspondences and corresponding epipolar lines. Reprinted from Hartley and Zisserman (2004).	9
2.7	Epipolar planes sharing a common axis. Reprinted from Hartley and Zisserman (2004).	9
2.8	Epipolar Geometry. Epipoles, epipolar planes and epipolar lines are displayed. Reprinted from Hartley and Zisserman (2004).	10
2.9	Camera Parameters. Camera rotation and translation with respect to world coordinate system with origin at O. Reprinted from Hartley and Zisserman (2004).	11
2.10	Second Camera. The four possible camera solutions for calibrated reconstruction from E. Reprinted from Hartley and Zisserman (2004).	12
2.11	Example projective reconstruction. Reprinted from Hartley and Zisserman (2004).	13
2.12	Example affine reconstruction. Reprinted from Hartley and Zisserman (2004).	13
2.13	Example metric reconstruction. Reprinted from Hartley and Zisserman (2004).	14
2.14	OpenGL embedded in FireFox Browser.	20
3.1	High-level system architecture. User inputs a video, which is processed in SFM. The parameters created by SFM are stored in the databased which are later utilized by the browser to display the video in browser-based 3D video player.	23

3.2 Relationship between baseline distance and triangulation uncertainty. With an appropriate camera motion direction, as the baseline distance increases the region of uncertainty decreases. 24
3.3 Ratio Variation. We captured Nico sequence with a typical hand recorded camera movement. The frames are multiple of 5 and the correspondence ratio decreases very slowly. 25
3.4 Fast Iteration Method. I searched the next key frames by jumping on the samples and then within the intervals with respect to a key frame. 25
3.5 Database Model. 31
3.6 Behind the camera mode. The camera is positioned slightly behind the current view. 33
3.7 Free Navigation mode. The user can move freely in the 3D world. The visualization starts from the perspective view of scene and first camera. 34
3.8 GUI Flow chart. The internal process of Graphical User Interface is shown. 36
4.1 Example degenerate motion in ground truth sequence. From frames 441 to 571, the camera rotates about its center without translation. 39
4.2 GRIC scores for homography and fundamental matrix model for the ground truth sequence. Red dash dot lines representing the GRIC score using the homography model and Green dash dot line represent the GRIC score using the fundamental model. To differentiate the selected model, a cyan dash dot line is placed which is set to 2500 if the fundamental model is selected else it is set to 0. The dots represents the occurrence of a frame. 40
4.3 Correspondence ratio for first frame and the subsequent frames for four different video sequences with different camera motion. 40
4.4 Imagenio sequence frames corresponding to the correspondence ratio data given in Figure 4.3. 41
4.5 The correspondence ratio upper threshold is selected when the model selection result changes from one model to the other. 42
4.6 Graphical Comparison. Weighted score *(KFWS)* is represented by cyan lines, GRIC difference *(KFGD)* is represented by blue lines and *uniformly sampled frame* is represented by red lines based on reprojection error. 44

List of Tables

Table	Title	Page
2.1	Feature extractor comparison based on computation time. Bay et al (2006) adjusted the threshold for each detector to exact a similar number of key points. Reprinted from Bay and Tuytelaars (2006).	7
4.1	Degenerate motion and structure detection.	38
4.2	Comparison based on reprojection error. Uniformly sampled frame method is shown as 'Fixed' and rest are same.	45

List of Abbreviations

SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
RANSAC	RANdom SAmple Consensus
SVD	Singular Value Decomposition
RE	Reprojection Error
OpenCV	Open source Computer Vision Library
OpenGL	Open Graphics Library
API	Application Programming Interface
GUI	Graphical User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WPF	Windows Presentation Foundation
GRIC	Geometric Robust Information Criterion
PELC	Point to Epipolar Line Cost
LMEDS	Least Median Squared
KFGD	Key Frames based on GRIC Difference
KFWS	Key Frames based on Weighted Score

Chapter 1
Introduction

The accuracy of the human vision system is far better than computer vision in some aspects. For instance, highly accurate analysis of location and depth of an object is a challenge for the research community in computer vision as compared to the human vision system. The use of two eyes to estimate depth in the human visual system can be extended to multiple cameras in a computer vision system. Calculating variations in depth in a structure using computer vision techniques requires a large amount of computational power. The process of image feature extraction, feature matching, projective reconstruction, metric reconstruction, and rendering all together need high performance computing. Recent advances in computing power and digital imaging technology have attracted researchers to find optimal methods for photo realistic reconstruction of 3D scenes from 2D images.

The goal of this research is to help out to find an optimal reconstruction of the structure of a 3D scene from a video sequence. The scenario is that anyone from anywhere, using a web browser, will upload a video to a server. The server will have the capability to convert the frames of video into a 3D world and store the parameters in a database. The user will be provided with a browser-based 3D video player to view the uploaded video, with the facility to freely view the world captured in that video. 'Freely view' means the user will be able to pause the video at any frame and move the virtual camera with respect to the 3D world points, as shown in Figure 1.1. This research is built on the latest findings in the structure-from-motion field (Pollefeys & Van Gool, 2002) which has inspired technical previews such as Microsoft Photosynth (Snavely, Seitz, & Szeliski, 2006; Noah, M., & Richard, 2008; Microsoft Cooperation, 2009), among others. This thesis is focused on 3D reconstruction from any arbitrary video sequence as opposed to still image-based reconstructions such as the ones by Microsoft Photosynth.

1.1 Problem Statement

This research is a part of a product which is in a phase of development in Telefonica I+D, Barcelona. The problem statement can be summarized as

- In order to reconstruct a 3D scene from video, it is necessary to lead the reconstruction by choosing a number of representative (key) frames. 3D reconstruction using images, taken from snapshot cameras, and frames, extracted manually from a video sequence, has been a focus of many researchers. A few researchers presented literature on key frame extraction. The main problem this thesis tries to solve is how to choose these key frames automatically. Consecutive key frames have to observe a common part of the scene. In addition, the relative positions of the camera from which each key frame was captured must be such that there exists a unique mathematical solution for the reconstruction and camera estimation problem.
- To visualize the video and the reconstructed scene, a new video playing paradigm must be developed. The problem of space-time visualization, i.e, to display the reconstructed scene as a cloud of points and to facilitate user to simultaneously play the captured video and view the reconstructed scene, is to be solved. While studying the existing systems, I did not find any system with such capabilities.

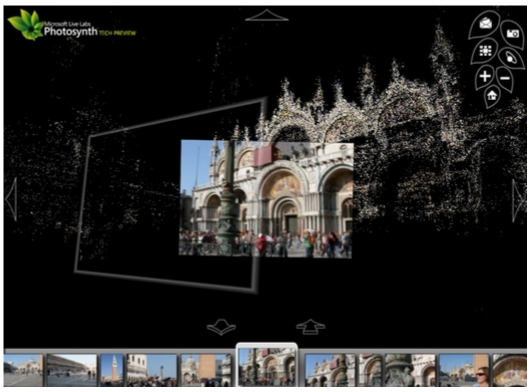


Figure 1.1: Microsoft Photosynth. The cloud of 3D reconstructed points and captured frames are shown. Reprinted from Microsoft (2009).

1.2 Objectives

The main objectives of this research are to optimize the structure from motion process through key frame extraction and to develop a browser-based 3D video player. Achieving these tasks, require the following steps:

- **Key Frame Extraction**: I must develop a key frame extractor to optimize reconstruction processing by using a minimal but appropriate set of frames instead of all frames. This method must be robust in cases where the epipolar geometry does not exist.
- **Front End: Construct a browser-based 3D video player**
  - **Render 3D Scene**: I must use Microsoft Silverlight to render 3D graphics in the web browser.
  - **Video Player**: I must develop a video player to support playback of videos in different modes.
  - **Free Viewer**: I must provide a viewing mode through which the user can move freely in the 3D world.

1.3 Assumptions and Limitations

There are a few assumptions and limitations.

- The camera intrinsic are predetermined.
- The environment is indoor.
- Only the sparse 3D structures consisting of a 3D point cloud, along with camera position, is required.

1.4 Environment

I carried out this research in an R&D company known as Telefonica I+D, Barcelona, Spain. An overview of the roles and conventions are given below.

- The team consists of 5 researchers, one supervisor, one software engineer, two Ph.D. students, and one master's student. The software engineer designed and implemented the over all architecture. Ph.D. students designed and implemented the high level 3D reconstruction algorithm, including the gold standard algorithm for fundamental matrix estimation, resectioning, and sparse bundle adjustment. I worked on key frame extraction, development of a browser-based 3D video player, and implementation of a basic reconstruction algorithm.
- The overall project was carried out by the whole team. Therefore, I used the following two conventions in this report:
  - "'I"' is used for parts of the research performed specifically by me.
  - "'We"' is used for the parts of the research performed by the team as a whole.

1.5 Organization

A brief review of the literature on 3D reconstruction, key frame extraction, and technology related to the embedded free viewer is given in Chapter 2. My methodology is described in Chapter 3. Chapter 4 discusses experiments and their results. Chapter 5 concludes the research work and highlights my possible future enhancements.

Chapter 2
Literature Review

2.1 Overview

This chapter includes material related to 3D reconstruction, key frame extraction, and browser-based 3D video player development. Each step of the research includes many algorithms, which are described in detail. As there are three major parts of the research, the review is divided accordingly. Classical 3D reconstruction methods are studied in Section 2.2, a brief introduction and the requirements for the key frame extractor are provided in Section 2.3, and our reasons for using Mircrosoft Silverlight as a tool in the browser-based 3D video player are given in Section 2.4.

2.2 Structure From Motion

Many researchers have been working on 3D reconstruction of models and scenes using images and videos. Humans are equipped with two vision sensors to capture a scene from two different angles. This help us to estimate relative depth or the points in the scene. Similar principles can be used to estimate depth from images. The basic steps explained in Pollefeys and Van Gool (2002) and (Hartley & Zisserman, 2004), are covered in the following subsections.

2.2.1 Feature Extraction

To estimate 3D structure we use a step by step process. Given matched features in two images, the 3D position of the points that generate these matches and the cameras that generate the images can be computed (Hartley & Zisserman, 2004). Therefore, the first important step in the process of 3D reconstruction is feature points extraction. Detailed literature is available about the features extraction techniques (Lowe, 2004; Bay, Tuytelaars, & Van Gool, 2006; Cha, Cofer, & Kozaitis, 2006). There are many techniques to detect features in images. They include those based on global features and local features.

Global features include those based on the area, perimeter, or symmetry of one object. But area and perimeter measures are sensitive to zoom and orientation.

Local feature extraction techniques have received more attention from researchers because they can be appropriate methods when image zoom, object orientation, shape, and image to image transformations are unknown. Some techniques for extracting local features include hole extraction, vertex extraction, edge extraction, and curvature extraction. Specific shapes, i.e, lines, circles and ellipses, can be extracted as features in images using the Hough transform (Cha et al., 2006). Arbitrary shapes can also be detected using the generalized the Hough transform. Other feature extraction methods, including blob extraction, template matching, and contour extraction, can also be found in the literature. These are general feature extraction methods.

In the literature, different practical approaches have been studied in the domain of feature extraction for 3D reconstruction. Pollefeys and Van Gool (2002) use a second-order approximation of dissimilarity between image windows with its neighboring windows. Noah et al. (2008) use the famous SIFT (Scale-Invariant Feature Transform) key point detector (Lowe, 2004) to detect features that are invariant to

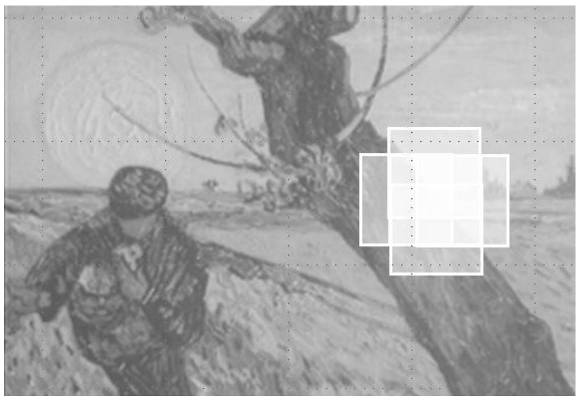
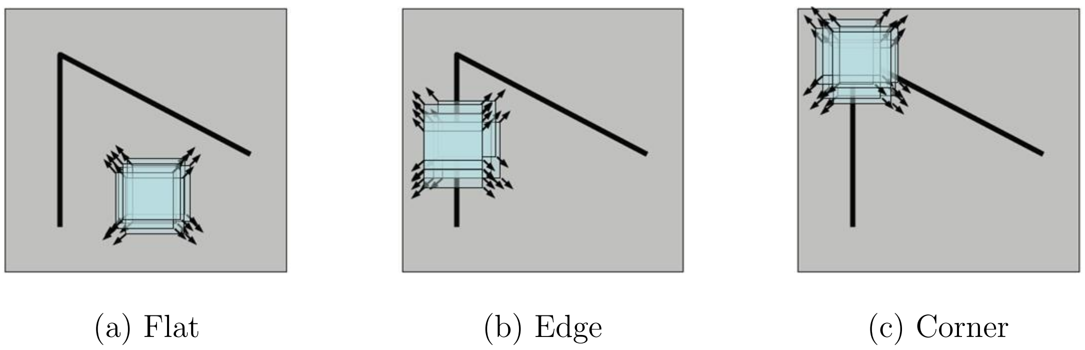


Figure 2.1: Harris Detector. A patch is shifted by a very small amount in different directions to measure local changes in auto correlation. Points, at which the correlation is large, are selected as interest points.



(a) Flat (b) Edge (c) Corner

Figure 2.2: Harris corner detection method. (a) On flat surfaces the result will remain the same as there is no change in the background. (b) The result on edge surface will show small variation. (c) The corner surface will show large variation. Therefore corners can be detected using the variation of the correlation of small window over image.

image transformations. A typical image contains several thousand SIFT key points. Seo, Kim, Doo, and Choi (2008) use a gradient matrix and extract feature using that matrix's eigenvalues to find key frames for 3D reconstruction.

The Harris detector (Harris & Stephens, 1988) is widely used. It is based on the idea of auto-correlation of image patches. A large difference auto-correlation in all directions indicates a good measure of an interest point, as shown in Figure 2.1. A patch is shifted a small amount in different directions to measure local changes in the auto-correlation function.

The auto-correlation for pixel (i, j) and shift $(\Delta i, \Delta j)$ can be calculated by

$$f(\delta i, \delta j) = \sum_{(i_k, j_k) \in \mathtt{W}(i,j)} (\mathtt{I}(i_k, j_k) - \mathtt{I}(i_k + \delta i, j_k + \delta j))^2. \tag{2.1}$$

If the function $f(\delta i, \delta j)$ is small in all directions, the region is essentially uniform. If it is large in one direction, then it is a contour, and if it is large in all directions, it is as a conner-like point. This concept is depicted in Figure 2.2. The Harris feature detector is invariant to geometric transformations such as translation and rotation. It

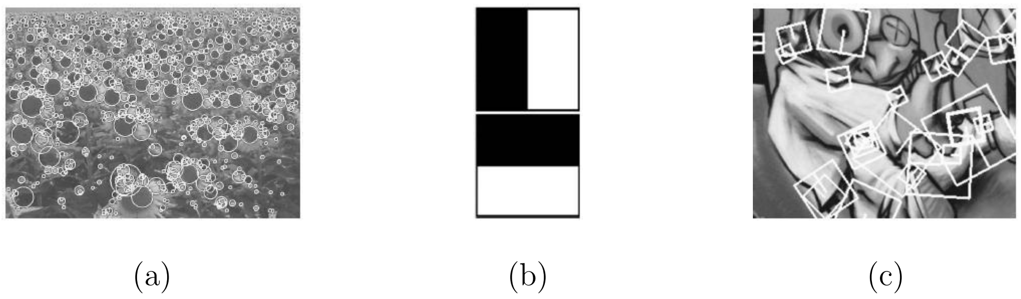


Figure 2.3: SURF. (a) Features location with reproducible orientations. (b) The Haar-wavelets used to calculate the orientation. (c) Keypoints with oriented descriptor calculation windows. Reprinted from Bay and Tuytelaars (2006).

is also invariant to some photometric transformations including intensity changes, but it is not scale invariant (Lowe, 2004)

Another method for local features extraction is SIFT (Lowe, 2004). SIFT is invariant to image scale and rotation. It is also robust to changes in illumination. Minor variations in viewpoint and image noise are also handled by SIFT. SIFT features are very distinctive and have low false positive detection probability. SIFT is also useful under partial occlusion as SIFT gives pose and location using very few features. The major stages of the algorithm are (Lowe, 2004):

- **Scale-space extrema detection:** This step searches over all scales and image locations to find points invariant to scale and orientation.
- **Keypoint localization:** The scale-space extrema are categorized as key points based on their stability. Stability can be determined by fitting a location and scale model.
- **Orientation assignment:** An orientation is assigned to each interest point based on the direction of the domain gradient. By doing so, the feature becomes invariant to scale, orientation, and location transformation.
- **Keypoint descriptor:** Gradients are measured within a small window around the key point and summarized in a descriptor vector so that changes in illumination and local shape distortion can be tolerated.

SURF (Speeded Up Robust Features) is another robust feature detection method. The SURF descriptor is similar to the SIFT descriptor but complexity is reduced even further. To make SURF rotation invariant, features with reproducible orientation are detected. To compute the orientation, Haar wavelet responses in the horizontal and vertical directions are combined to get the gradient vector. A square region aligned to the selected orientation is used to extract the descriptor, (see Figure 2.3). The details of this algorithm are given in Bay et al. (2006). A performance comparison is given in Table 2.1. The results show that the run time performance of SURF is much better than that of SIFT.

2.2.2 Feature Matching

SIFT and SURF provide features with orientation, scale, illumination, and location information. The next step is to find corresponding features in image pairs. The

	U-SURF	**SURF**	**SURF-128**	**SIFT**
Time (ms)	255	354	391	1036

Table 2.1: Feature extractor comparison based on computation time. Bay et al (2006) adjusted the threshold for each detector to exact a similar number of key points. Reprinted from Bay and Tuytelaars (2006).

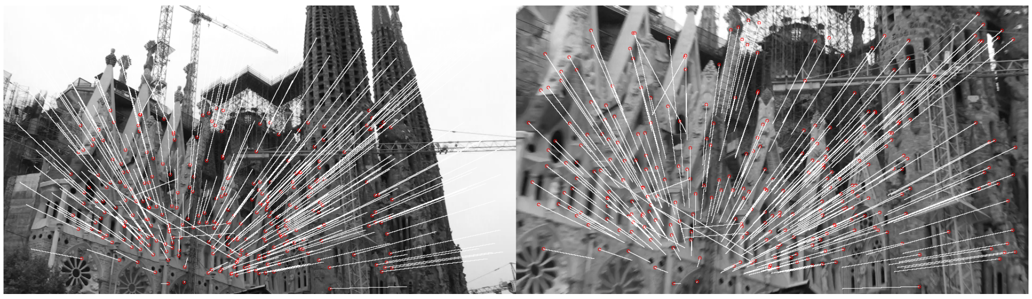


Figure 2.4: Feature Matching. Matches between the two images are shown along with outliers. The images are captured from a historical building called Sagarada Familia in Barcelona.

most popular technique was proposed by Lowe (2004). They match features with their nearest neighbors. For feature matching, each key point is matched with its nearest neighbor by minimizing the Euclidean distance between the keypoints descriptor vectors (Lowe, 2004). The results are not typically perfect, and many mismatches are classified as matches because of ambiguous local patterns. To discard such matches, Lowe (2004) only accepts matches where the ratio between the distances for the nearest and the second nearest match is above threshold. To speed up matching Lowe (2004) introduce the Best-Bin-First (BBF) approximation which returns the nearest neighbor of the descriptor with high probability. Figure 2.4 shows examples of matched features. Matched features in multiple images can be used for tracking. Each track at a point can be associated with a 3D point. The tracked points and 3D point relationship can be used to optimize the process of reconstruction.

2.2.3 Outliers Discarding

Mismatches always occur when features are matched. For example, there are some outliers in Figure 2.4. We might use an affine transformation to find and discard outliers since it would preserve collinearity and ratios of distances, but this measure is affected when 3D rotation of non planar objects takes place. A solution given in Hartley and Zisserman (2004) is to solve for the fundamental matrix. The fundamental matrix $\mathtt{F}$ is a 3×3 matrix of rank 2 that describes the geometry of two views of the same scene. A 3D point $\mathtt{X}$ is projected as 2D points $\mathtt{x}$ and $\mathtt{x}'$ in the first and second views, respectively, satisfying the relationship $\mathtt{x}'^{\mathtt{T}}\mathtt{F}\mathtt{x} = 0$. The fundamental matrix can be computed from matched features without requiring knowledge of camera pose or geometry. At least 7 point correspondences between two images are required to compute the fundamental matrix $\mathtt{F}$. In practice, more matches are required for an accurate estimate. Seo, Kim, Jho, and Hong (2003) compute a precise fundamental matrix from evenly distributed corresponding points over the entire image. Once the

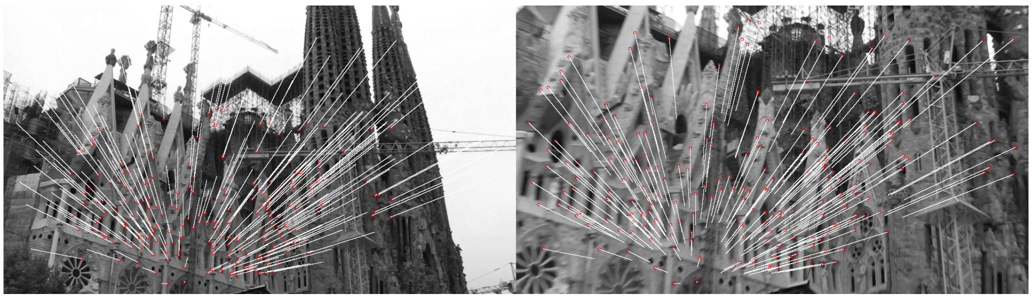


Figure 2.5: Feature Matching. Matching between the pair of images are shown with inliers using the fundamental model. The images are captured from a historical building called Sagarada Familia.

fundamental matrix is known, it can be used to discard outliers, because $\mathtt{Fx}$ and $\mathtt{F}^{\mathtt{T}}\mathtt{x}'$ are the epipolar lines corresponding to $\mathtt{x}$ and $\mathtt{x}'$ in the first and second image, respectively. In the presence of noise, the distance from a point to its epipolar line can be used as a measure to discard outliers. Details of epipolar geometry and the fundamental matrix are given in Section 2.2.4. There are a number of methods for optimally discarding outliers. The most common and recommended method is RANdom SAmple Consensus (RANSAC) (Hartley & Zisserman, 2004, Algorithm 11.4).

RANSAC is an iterative method to estimate outliers and inliers. The idea of RANSAC is very simple: it searches for the number of inliers within a threshold of a model fitted to a random sample. The robust sample is the one which is supported by the highest numbers of inliers. The decision as to whether a measurement is an inlier or outlier is as follows:

$$\begin{aligned} \mathit{Inlier} &: \; d_{\perp}^2 < t^2 \\ \mathit{Outlier} &: \; d_{\perp}^2 \leq t^2, \end{aligned} \tag{2.2}$$

where t is a threshold and $d_{\perp}^2$ is distance between the measurement and the model. Hartley and Zisserman (2004) describes two important cost functions for $\mathtt{F}$ estimation. The first is reprojection error, given by

$$\sum_i d(\mathtt{x}_i, \hat{\mathtt{x}}_i)^2 + d(\mathtt{x}_i^{'}, \hat{\mathtt{x}}_i^{'})^2, \tag{2.3}$$

where $\mathtt{x}_i \leftrightarrow \mathtt{x}_i^{'}$ are the measured correspondences, $\hat{\mathtt{x}}_i \leftrightarrow \hat{\mathtt{x}}_i^{'}$ are estimated true correspondences that satisfy $\hat{\mathtt{x}}_i^{'\mathtt{T}}\mathtt{F}\hat{\mathtt{x}}_i = 0$, and $\mathtt{F}$ is the estimated fundamental matrix. To calculate reprojection error, a pair of camera matrices $\mathtt{P} = [\mathtt{I}|0]$, and $\mathtt{P}^{'} = [\mathtt{M}|\mathtt{t}]$ are first estimated along with 3D points $\mathtt{X}_i$. Thus, $\hat{\mathtt{x}}_i = \mathtt{PX}_i$ and $\hat{\mathtt{x}}_i^{'} = \mathtt{P}^{'}\mathtt{X}_i$. Since given $\mathtt{P}$, $\mathtt{P}^{'}$ of this form $\mathtt{F}$ is $\mathtt{F} = [\mathtt{t}]_\times\mathtt{M}$, conversely, given $\mathtt{F}$, consistent camera matrices can be chosen as $\mathtt{P} = [\mathtt{I}|0]$ and $\mathtt{P}^{'} = [[\mathtt{e}']_\times\mathtt{F}|\mathtt{e}']$, where $\mathtt{e}'$ satisfies $\mathtt{F}^{\mathtt{T}}\mathtt{e}' = 0$. The second possible choice of cost the function is the symmetric epipolar distance:

$$\sum d(\mathtt{x}_i^{'}, \mathtt{Fx}_i)^2 + d(\mathtt{x}_i, \mathtt{F}^{\mathtt{T}}\mathtt{x}_i^{'})^2, \tag{2.4}$$

where $d(\mathtt{x}, \mathtt{l})$ is the distance of $\mathtt{x}$ to line $\mathtt{l}$, the point to epipolar line cost is

$$PELC = \sum (\mathtt{x}_i^{'}, \mathtt{Fx}_i)^2 \left(\frac{1}{(\mathtt{Fx}_i)_1^2 + (\mathtt{Fx}_i)_2^2} + \frac{1}{(\mathtt{F}^{\mathtt{T}}\mathtt{x}_i^{'})_1^2 + (\mathtt{F}^{\mathtt{T}}\mathtt{x}_i^{'})_2^2} \right). \tag{2.5}$$

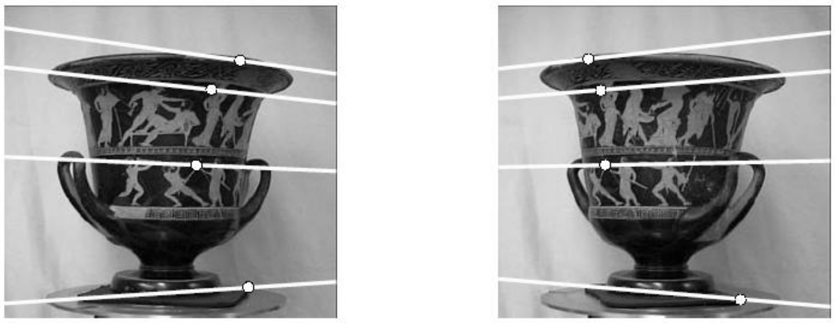


Figure 2.6: Point correspondences and corresponding epipolar lines. Reprinted from Hartley and Zisserman (2004).

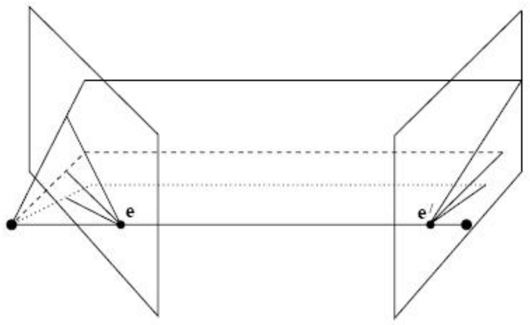


Figure 2.7: Epipolar planes sharing a common axis. Reprinted from Hartley and Zisserman (2004).

The above cost function is used in OpenCV (Bradski, Rost, Kaehler, & Adrian, 2008) for the calculation of the fundamental matrix and outlier discarding. The results of applying outlier discarding to the matches in Figure 2.4 are shown in Figure 2.2.3. It is apparent from the figure that many matches are discarded.

2.2.4 Two View Geometry

Point correspondences between two views can be used to estimate a fundamental matrix as described in the previous section. The fundamental matrix calculation using point correspondences is based on epipolar geometry. The geometry of the intersections of two image planes with the pencil of planes is the epipolar geometry. The set of planes share a common axis, the so called baseline, which is the line through the two camera centers.

There are some important concepts in epipolar geometry, given below and visualized in Figure 2.8.

- **Epipoles:** the points where the baseline intersects the image planes, represented by $\mathtt{e}$ and $\mathtt{e}'$.
- **Epipolar plane:** a plane containing the baseline and corresponding epipolar lines in the two image planes.
- **Epipolar line:** the lines formed by the intersection of epipolar plane with two image planes in a pair of images. The epipolar lines are represented by $\mathtt{l}$ and $\mathtt{l}'$ as shown in Figure 2.8.

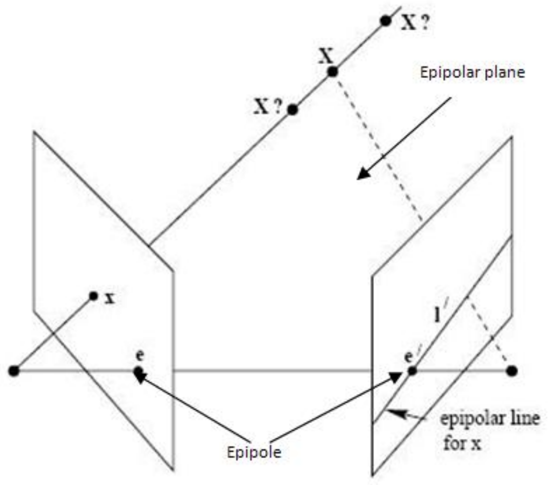


Figure 2.8: Epipolar Geometry. Epipoles, epipolar planes and epipolar lines are displayed. Reprinted from Hartley and Zisserman (2004).

Given point correspondences, we can estimate the epipolar geometry of a camera pair. Mismatched *outliers* can be discarded and additional matches can determined using the structure. Equation $\mathtt{x}'^{\mathtt{T}}\mathtt{F}\mathtt{x} = 0$ is an algebraic representation of the epipolar geometry. A single point correspondence is shown in Figure 2.8.

The point $\mathtt{x}$ in one image corresponds to an epipolar line $\mathtt{l}'$ in the other image, and the corresponding point $\mathtt{x}'$ must lie on $\mathtt{l}'$. Thus there is a mapping $\mathtt{x} \mapsto \mathtt{l}'$ from a point to its corresponding epipolar line in the other image. This projective mapping is represented by $\mathtt{F}$, the fundamental matrix. The line $\mathtt{l}'$ passes through point $\mathtt{x}'$ and the epipole $\mathtt{e}'$; as a result $\mathtt{l}'$ can be expressed as $\mathtt{l}' = \mathtt{e}' \times \mathtt{x}'$ and $\mathtt{x}'$ may be written as $\mathtt{x}' = \mathtt{H}_\pi \mathtt{x}$.

$$\mathtt{l}' = [\mathtt{e}']_\times \mathtt{H}_\pi \mathtt{x} = \mathtt{F}\mathtt{x},$$

where $\mathtt{F} = [\mathtt{e}']_\times \mathtt{H}_\pi$ is the fundamental matrix. The epipolar constraint $\mathtt{x}'^{\mathtt{T}}\mathtt{F}\mathtt{x} = 0$ is satisfied by the fundamental matrix and the corresponding points in two views, as point $\mathtt{x}'$ lies on epipolar line $\mathtt{l}' = \mathtt{F}\mathtt{x}$. To achieve a projective reconstruction, which is discussed in section 2.2.6, only correspondences and the fundamental matrix are required. But for metric reconstruction, camera caliberation or other constraints are required.

2.2.5 Camera Calibration

The fundamental matrix is the generalization of the essential matrix to the case of uncalibrated cameras. The essential matrix has fewer degrees of freedom and therefore additional constraints compared to the fundamental matrix. The camera calibration matrix and the fundamental matrix can be used to calculate the essential matrix. A general camera matrix $\mathtt{P}$ can be decomposed into intrinsic parameters $\mathtt{K}$ and extrinsic parameters $[\mathtt{R}|\mathtt{t}]$ as $\mathtt{P} = \mathtt{K}[\mathtt{R}|\mathtt{t}]$.

The intrinsic parameter matrix $\mathtt{K}$ is an upper triangular matrix that models the transformation of a standardized image plane, from camera coordinates to pixel coordinates. Camera $\mathtt{K}$ has five degrees of freedom: the focal length f, the principal point (x_0, y_0), the pixel aspect ratio and the coordinate axis skewness. These five degrees of freedom correspond to the five symbols in the matrix

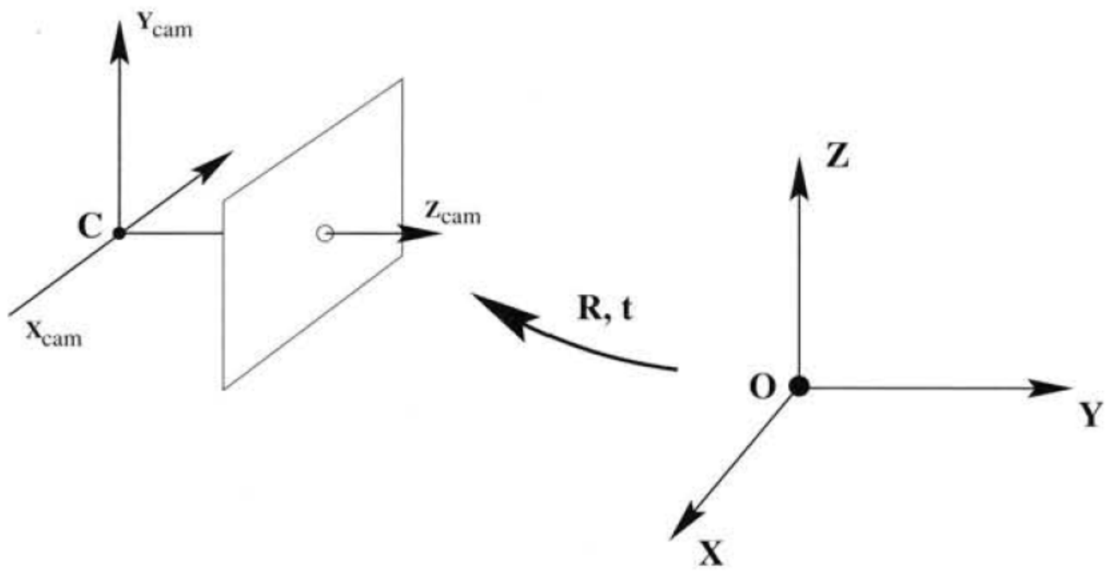


Figure 2.9: Camera Parameters. Camera rotation and translation with respect to world coordinate system with origin at $\mathtt{O}$. Reprinted from Hartley and Zisserman (2004).

$$\mathtt{K} = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.6}$$

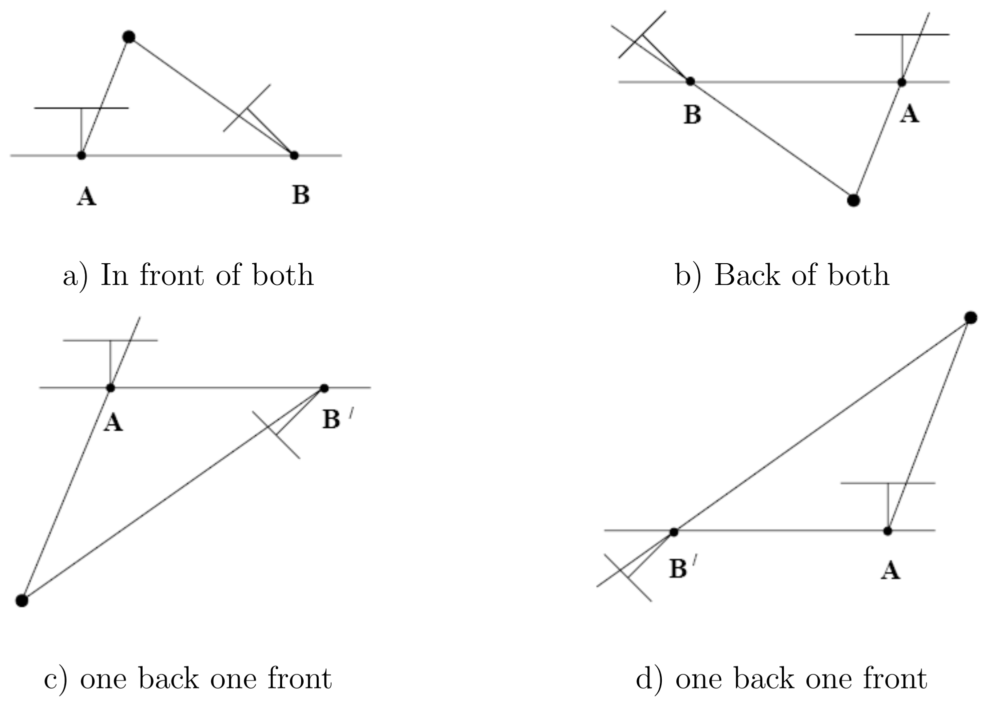
where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ are given in pixel units (the factors m_x and m_y relating the shape of the pixel are combined with the focal length of the camera), the parameter s is the skew, which is zero in most normal cameras.

The other two parts of the projection matrix are the rotation matrix $\mathtt{R}$ and the translation vector $\mathtt{t}$ with respect to a specified world coordinate system. Figure 2.9 shows a very clear picture of $\mathtt{R}$ and $\mathtt{t}$.

A 3D point $\mathtt{X}$ in world coordinates is projected to a 2D point $\mathtt{x}$ in the image by $\mathtt{x} = \mathtt{PX}$. If the camera intrinsics are known, then we can obtain a point $\hat{\mathtt{x}} = \mathtt{K}^{-1}\mathtt{x}$, which is the image point expressed in the normalized camera coordinate system. A normalized camera matrix $\mathtt{K}^{-1}\mathtt{P} = [\mathtt{R}|\mathtt{t}]$, has the effect of the known calibration matrix $\mathtt{K}$ removed. Normalized image coordinates $\hat{\mathtt{x}}$ satisfy the epipolar constraint with respect to the essential matrix $\hat{\mathtt{x}}'^{\mathtt{T}}\mathtt{E}\hat{\mathtt{x}} = 0$ and the essential matrix, which can be obtained from the fundamental matrix by

$$\mathtt{E} = \mathtt{K}'^{\mathtt{T}}\mathtt{FK}. \tag{2.7}$$

To calculate the essential matrix from the fundamental matrix, the camera's intrinsic parameters must be known. The essential matrix is a 3×3 matrix of rank 2 whose two nonzero singular values are equal (Hartley & Zisserman, 2004, Section 9.6). $\mathtt{E}$ can be decomposed as $[\mathtt{t}]_\times\mathtt{R} = \mathtt{SR}$ where $\mathtt{S}$ is skew-symmetric. Hartley and Zisserman



a) In front of both b) Back of both

c) one back one front d) one back one front

Figure 2.10: Second Camera. The four possible camera solutions for calibrated reconstruction from $\mathtt{E}$. Reprinted from Hartley and Zisserman (2004).

(2004) gives a method to compute $\mathtt{R}$ and $\mathtt{t}$ from the SVD of $\mathtt{E}$ by using matrices

$$\mathtt{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } \mathtt{Z} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where $\mathtt{W}$ is orthogonal and $\mathtt{Z}$ is skew-symmetric. The singular value decomposition of $\mathtt{E}$ is $\mathtt{U}\,\mathrm{diag}(1,1,0)\mathtt{V}^\mathtt{T}$. $\mathtt{S} = \mathtt{UZU}^\mathtt{T}$ and $\mathtt{R} = \mathtt{UWV}^\mathtt{T}$ can be computed using $\mathtt{Z}$ and $\mathtt{W}$ respectively.

Given the essential matrix $\mathtt{E}$, consistent normalized camera matrices can be chosen as $\mathtt{P} = [\mathtt{I}|0]$ and $\mathtt{P}' = [\mathtt{R}|\mathtt{t}]$, where there are four possible solutions for the second camera as shown in Figure 2.10. These four possible solutions can be written as

$$\mathtt{P}' = [\mathtt{UWV}^\mathtt{T}|\mathrm{u}_3],\ \mathtt{P}' = [\mathtt{UWV}^\mathtt{T}| - \mathrm{u}_3],\ \mathtt{P}' = [\mathtt{UW}^\mathtt{T}\mathtt{V}^\mathtt{T}|\mathrm{u}_3] \text{ or } \mathtt{P}' = [\mathtt{UW}^\mathtt{T}\mathtt{V}^\mathtt{T}| - \mathrm{u}_3]$$

where $\mathtt{U}$ and $\mathtt{V}$ are taken from $\mathrm{SVD}(\mathtt{E}) = \mathtt{U}\,\mathrm{diag}(1,1,0)\mathtt{V}^\mathtt{T}$ and $\mathrm{u}_3 = \mathtt{U}(0,0,1)^\mathtt{T}$, the last column of $\mathtt{U}$. One of the possible cameras is selected among these choices by triangulating a point known to be in front of both cameras and selecting the consistent camera matrix.

2.2.6 Reconstruction

Assume two points in two images that satisfy the epipolar constraint, $\mathtt{x}'^\mathtt{T}\mathtt{Fx} = 0$ and camera matrices $\mathtt{P}$ and $\mathtt{P}'$ are known. The epipolar constraint ensures the existence of the epipolar plane which contains the rays back-projecting from $\mathtt{x}$ and $\mathtt{x}'$. When two rays share a common plane, they intersect at some point $\mathtt{X}$. The projection of this

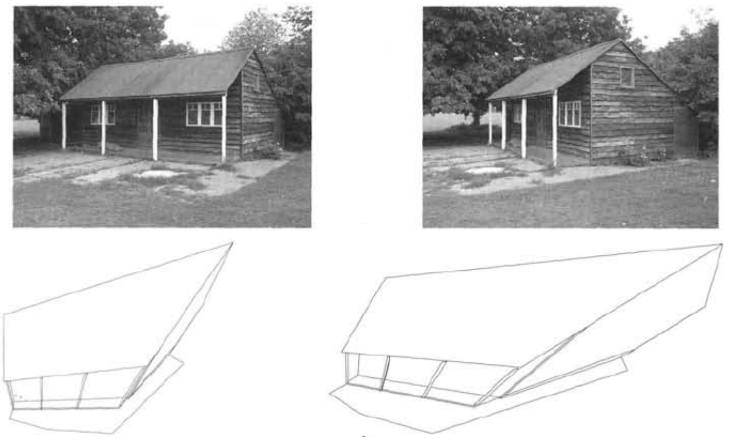


Figure 2.11: Example projective reconstruction. Reprinted from Hartley and Zisserman (2004).

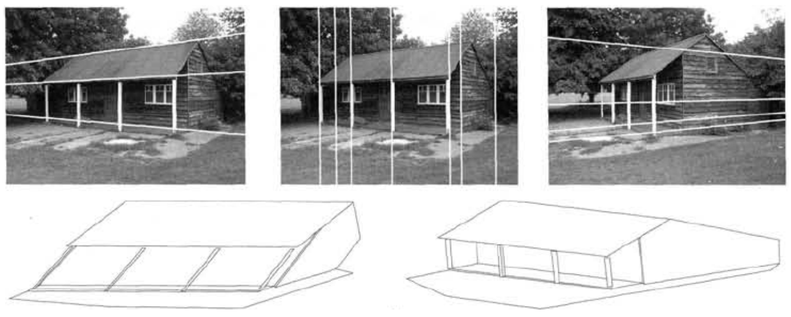


Figure 2.12: Example affine reconstruction. Reprinted from Hartley and Zisserman (2004).

point $\mathtt{X}$ using camera matrices $\mathtt{P}$ and $\mathtt{P}'$ results in $\mathtt{x}$ and $\mathtt{x}'$. Triangulation is a method to estimate 3D points from pair of points in two images.

Depending on the coordinate system in which $\mathtt{P}$ and $\mathtt{P}'$ are expressed, there are different types of reconstruction, each one with an associated ambiguity. For example, since the fundamental matrix can be uniquely determined from pairs of feature points, i.e. correspondences, and projective camera matrices may be reconstructed from these correspondences alone, the obtained reconstruction is defined up to a projective transformation (see Figure 2.11) and therefore it is called a *projective reconstruction.* A projective reconstruction is the first step towards metric reconstruction.

There is another approach for reconstruction known as the *stratified approach.* If $\mathtt{P}$ and $\mathtt{P}'$ are known in a projective coordinate system, and some self-calibration algorithm is used to express them with respect to an affine coordinate system, then the resulting reconstruction will be called an *affine reconstruction* (see Figure 2.12). Moreover, if on a second step, another self-calibration algorithm is used to express $\mathtt{P}$ and $\mathtt{P}'$ with respect to a Euclidean coordinate system, then the resulting reconstruction will be called a *metric reconstruction* (see Figure 2.13).

In a metric reconstruction, properties like angles between lines and ratios of lengths are preserved in the reconstructed model or scene. One way to transform a projective reconstruction in to a metric reconstruction involves finding the image of absolute conic in each image. There are three constraints on the image of the absolute conic given by Hartley and Zisserman (2004).

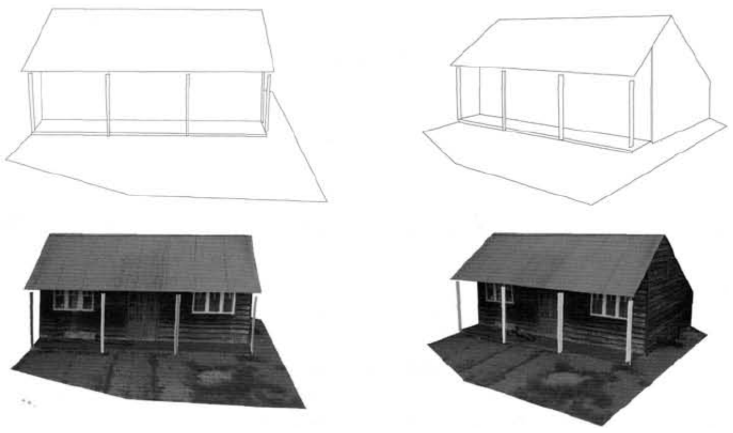


Figure 2.13: Example metric reconstruction. Reprinted from Hartley and Zisserman (2004).

- **Constraints arising from scene orthogonality**: A pair of vanishing points arising from orthogonal scene lines places a single linear constraint on the absolute conic. Similarly, an image plane containing metric information, such as a square grid, places two constraints on the absolute conic.
- **Constraints arising from known internal parameters**: If the calibration matrix $\mathtt{K}$ of the camera is known, it may be used as constraint in estimating an Euclidean camera matrix $\mathtt{P}$.
- **Constraints arising from the same cameras in all images**: When projecting the absolute conic, the projection depends only on the calibration matrix of of the camera. Given many images from the same camera with same internal parameters, a metric reconstruction can be obtained from an affine reconstruction. This is the second step of the stratified approach.

The affine reconstruction from Figure 2.12 can be upgraded to the metric reconstruction in Figure 2.13 by computing the absolute conic. (Hartley & Zisserman, 2004, Algorithm 10.1) provides a complete transformation from a projective reconstruction to a metric reconstruction.

If camera matrices $\mathtt{P}$ and $\mathtt{P}'$, and 2D feature points $\mathtt{x}$ and $\mathtt{x}'$, are given then a *Linear triangulation method* can be used to calculate 3D points. Linear triangulation method is a non iterative method for triangulation. The projection equations $\mathtt{x} = \mathtt{PX}$ and $\mathtt{x}' = \mathtt{P}'\mathtt{X}$ can be expressed as a linear system of equations $\mathtt{AX} = 0$; the homogeneous scale factor can be eliminated by a cross product, and each image point provides three equations, two of which are linearly independent. For instance $\mathtt{x} \times (\mathtt{PX}) = 0$ yields

$$\begin{aligned} x(\mathtt{p}^{3\mathsf{T}}\mathtt{X}) - (\mathtt{p}^{1\mathsf{T}}\mathtt{X}) &= 0 \\ y(\mathtt{p}^{3\mathsf{T}}\mathtt{X}) - (\mathtt{p}^{2\mathsf{T}}\mathtt{X}) &= 0 \\ x(\mathtt{p}^{2\mathsf{T}}\mathtt{X}) - y(\mathtt{p}^{1\mathsf{T}}\mathtt{X}) &= 0 \end{aligned} \tag{2.8}$$

where $\mathtt{p}^{i\mathsf{T}}$ are the rows of camera matrix $\mathtt{P}$. This can also be written as $[\mathtt{x}]_\times \mathtt{PX} = 0$. Now considering both projections, a system of the form $\mathtt{AX} = 0$ can be composed, where

$$\mathtt{A} = \begin{bmatrix} [\mathtt{x}]_\times \mathtt{P} \\ [\mathtt{x}']_\times \mathtt{P}' \end{bmatrix}. \tag{2.9}$$

The solution $\mathtt{X}$ is the right singular vector of $\mathtt{A}$ corresponding to the smallest singular value of $\mathtt{A}$. This triangulation algorithm can be extended to multiple views (by stacking more equations in $\mathtt{A}$) if the camera matrices are known in the same coordinate system as $\mathtt{P}$ and $\mathtt{P}'$.

2.2.7 Multiple Views

Given an initial reconstruction from two views, a camera matrix for the next view can be recovered using *resectioning.* Resectioning is the process of computing a camera matrix using 2D-3D point correspondences. We can use the 3D points computed from earlier frames in correspondence with the next frame to compute the camera matrix of the next frame. The Linear resection equations show the relationship between 3D points, 2D points, and the camera matrix.

$$\begin{bmatrix} \mathtt{0}^\mathsf{T} & -w_i\mathtt{X}_i^\mathsf{T} & y_i\mathtt{X}_i^\mathsf{T} \\ w_i\mathtt{X}_i^\mathsf{T} & \mathtt{0}^\mathsf{T} & -x_i\mathtt{X}_i^\mathsf{T} \\ -y_i\mathtt{X}_i^\mathsf{T} & x_i\mathtt{X}_i^\mathsf{T} & \mathtt{0}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \mathtt{p}^1 \\ \mathtt{p}^2 \\ \mathtt{p}^3 \end{bmatrix} = \mathtt{0}, \tag{2.10}$$

where $\mathtt{p}^{i\mathsf{T}}$ are the rows of camera matrix $\mathtt{P}$. Only two of the equations are linearly independent. Stacking up the equations of n point correspondences result in a $3n \times 12$ matrix $\mathtt{A}$. The projection matrix $\mathtt{P}$ is computed by solving the set of equations $\mathtt{Ap} = 0$ where $\mathtt{p}$ is the vector containing the entries of the matrix $\mathtt{P}$. At least six point correspondences are required to solve the system.

By combining the triangulation and resectioning algorithms explained before, a 3D reconstruction of multiple points and cameras can be computed. In practice, we use this strategy to compute the initial reconstruction that will be optimized using the sparse bundle adjustment technique (Hartley & Zisserman, 2004; Lourakis & Argyros, 2009).

2.2.8 Optimization Process

In the literature (Hartley & Zisserman, 2004), there are a few iterative methods for robust and efficient estimation to improve the results at different stages in the reconstruction process.

2.2.9 Iterative Estimation Methods

The most famous common iterative parameter estimation method is *Newton's iteration.* Newton's iteration estimates the parameters which locally minimize a general nonlinear function, in particular, a nonlinear least-squares function. The basic principle is explained here; for more details see (Hartley & Zisserman, 2004).

Newton's root finding method.

An initial estimate of parameter vector $\mathtt{P}_0$ is required to pass in the function $f(\mathtt{P}_0)$, where f is a nonlinear function. We have

$$\mathtt{X} = f(\mathtt{P}_0), \tag{2.11}$$

where $\mathtt{X}$ is a measurement vector. Let the error be $\epsilon_0 = f(\mathtt{P}_0) - \mathtt{X}$ and to approximate the function, we assume

$$f(\mathtt{P}_0 + \Delta) = f(\mathtt{P}_0) + \mathtt{J}\Delta, \tag{2.12}$$

where $\mathtt{J} = \frac{\partial f}{\partial \mathtt{P}}$. We search for $f(\mathtt{P}_1)$ by $\mathtt{P}_1 = \mathtt{P}_0 + \Delta$. So equation (2.12) is derived, with the measurement vector, as

$$f(\mathtt{P}_1) - \mathtt{X} = f(\mathtt{P}_0) + \mathtt{J}\Delta - \mathtt{X} = \epsilon_0 + \mathtt{J}\Delta, \tag{2.13}$$

Thus, we need to minimize the $\|\epsilon_0 + \mathtt{J}\Delta\|$ over Δ, which is a linear minimization problem whose solution satisfies the *Normal* equations (Hartley & Zisserman, 2004, A5.2):

$$\mathtt{J}^{\mathtt{T}}\mathtt{J}\Delta = -\mathtt{J}^{\mathtt{T}}\epsilon_0. \tag{2.14}$$

Equation (2.14) can be rewritten as $\Delta = -\mathtt{J}^{+}\epsilon_0$, where $\mathtt{J}^{+} = (\mathtt{J}^{\mathtt{T}}\mathtt{J})^{-1}\mathtt{J}^{\mathtt{T}}$ is the pseudo inverse of $\mathtt{J}$. We can generalize the above equations by $\mathtt{P}_{i+1} = \mathtt{P}_i + \Delta_i$ where Δ_i is the solution to the linear least-square problem $\mathtt{J}\Delta_i = -\epsilon_i$. The algorithm will converge to the least square solution but if the initial estimate $\mathtt{P}_0$ is close enough.

Newton's method for minimization of functions is a more general method to find minima of functions of many variables. We have an arbitrary scalar valued function $g(\mathtt{P})$ where $\mathtt{P}$ is a vector. We intend to minimize $g(\mathtt{P})$ over all values of $\mathtt{P}$. We assume that $\mathtt{P}_0$ is reasonably close to the minimum and function $g(\mathtt{P})$ has a well defined minimum value. The Taylor series for g

$$g(\mathtt{P}_0 + \Delta) = g + g_{\mathtt{P}}\Delta + \frac{1}{2}\Delta^{\mathtt{T}} g_{\mathtt{PP}}\Delta + \cdots, \tag{2.15}$$

where subscript $\mathtt{P}$ represents differentiation with respect to Equation (2.15). We intend to minimize with respect to Δ. To do so, we differentiate the quadratic approximation with respect to Δ and set it to zero. We get

$$g_{\mathtt{P}} + g_{\mathtt{PP}} = 0, \tag{2.16}$$

where $g_{\mathtt{PP}}$ and $g_{\mathtt{P}}$ are the Hessian and the gradient of g, respectively. In particular, if $g(\mathtt{P}) = \frac{1}{2}\|\epsilon(\mathtt{P})\|^2$, then $g_{\mathtt{PP}} = \epsilon_{\mathtt{P}}^{\mathtt{T}}\epsilon_{\mathtt{P}} + \epsilon_{\mathtt{PP}}^{T}\epsilon$ and $g_{\mathtt{P}} = \epsilon_{\mathtt{P}}^{\mathtt{T}}\epsilon$. *Newton's iteration* starts with initial parameter $\mathtt{P}_0$ and iteratively computes its increment Δ until convergence occurs. The disadvantage of this approach is that the computation of the Hessian may be difficult. Therefore, some alternatives to approximate the Hessian are proposed in the literature.

The *Guass-Newton method* assumes $\epsilon(\mathtt{P})$ is a linear function, thus, the second term of $g_{\mathtt{PP}}$ vanishes, leaving $g_{\mathtt{PP}} = \epsilon_{\mathtt{P}}^{\mathtt{T}}\epsilon_{\mathtt{P}} = \mathtt{J}^{\mathtt{T}}\mathtt{J}$. By substituting the gradient and the Hessian in equation (2.16) yields equation $\mathtt{J}^{\mathtt{T}}\mathtt{J}\Delta = -\mathtt{J}^{\mathtt{T}}\epsilon_0$, which is the same as equation (2.14). Therefore both equations are equal, assuming $\mathtt{J}^{\mathtt{T}}\mathtt{J}$ is an approximation for the Hessian of the function $g(\mathtt{P})$.

In the *Gradient Descent method*, the negative gradient vector $-g_{\mathtt{P}} = -\epsilon_{\mathtt{P}}^{\mathtt{T}}\epsilon$ locally defines the direction of maximum decrease of the cost function. Moving along this direction is the simplest method for minimization of g because it only requires information of the first derivatives (gradient), not the second derivatives (Hessian).

The *Levenberg-Marquardt* (LM) iteration method is a slight variation on the Guass-Newton iteration method (Ranganathan, 2004). The normal equations $\mathtt{J}^{\mathtt{T}}\mathtt{J}\Delta = -\mathtt{J}^{\mathtt{T}}\epsilon$ are replaced by the augmented normal equations $(\mathtt{J}^{\mathtt{T}}\mathtt{J} + \lambda\mathtt{I})\Delta = -\mathtt{J}^{\mathtt{T}}\epsilon$, where λ varies from iteration to iteration and $\mathtt{I}$ is the identity matrix. In our 3D reconstruction problem, the parameter vector is

$$\mathtt{P} = [\mathtt{a}_1^{\mathtt{T}}, \cdots, \mathtt{a}_n^{\mathtt{T}}, \mathtt{b}_1^{\mathtt{T}}, \cdots, \mathtt{b}_m^{\mathtt{T}}]^{\mathtt{T}}, \tag{2.17}$$

where parameters $\mathtt{a}_n$ are 3D points and parameters b_n describe the cameras. This leads to a sparse matrix in the system of normal equations. In the context of 3D reconstruction, an efficient implementation of the LM method that exploits the sparse structure of the matrix to solve the Normal Equations is known in the literature as *Sparse Bundle Adjustment* (SBA). The complete details and implementation of the LM method for 3D reconstruction using SBA is given in Lourakis and Argyros (2009).

2.2.10 Gold Standard Algorithm

- The gold standard algorithm for the fundamental matrix is given in Hartley and Zisserman (2004, Algorithm A14.1). The algorithm is used after n correspondences are found. The algorithm minimizes the reprojection error between two views. It provides an estimate of both camera matrices and 3D points in a projective coordinate system. A more robust algorithm is based on RANSAC (Hartley & Zisserman, 2004, Algorithm 11.4), and is implemented in OpenCV (Bradski et al., 2008), but the nonlinear refinement of the fundamental matrix is not included.
- There is also a gold standard algorithm for camera resection, if the intrinsics are known. OpenCV provides an implementation which finds the extrinsic camera parameters for a particular view in such case.

2.3 Key Frame Extraction

A brief literature review has been presented on 3D reconstruction of sparse structure using multiple views. When considering 3D reconstruction using video sequences, there are issues to be reviewed and techniques to be developed to ensure performance in terms of reconstruction and computational cost. 3D reconstruction using all the frames in a video sequence is very CPU demanding. There are many reasons given in literature for which some key frames, from all the frames in a video sequence, should be extracted.

2.3.1 Requirements

- **Performance**: If a camera with 25 frames per second rate is used to create one minute video, this results into 1500 frames. Processing all 1500 frames would be costly in term of computation and performance.
  - **Pose and Motion Analysis**: Estimation of the 3D camera pose and recovery of 3D scene geometry are two very expensive processes in 3D reconstruction if performed with all frames in a video sequence. If the frames are decimated then these processes become less expensive.

- **Baseline distance**: Consecutive frames may have very short baseline distances. Frames must have a long enough baseline for triangulation. The frame selection process must take this important factor in to account.
- **Degenerate Cases**: The fundamental matrix provides an accurate relationship between two images if there is general camera motion and general position for structure. When the generality assumptions for camera motion or structure do not hold, the method fails. These conditions of nongeneral camera motion and structure are known as degenerate cases. The degenerate cases can be described as

  1. **Motion degeneracy**: If the camera rotates about its center with no translation, the epipolar geometry is not defined. But if the 3D points are known then a camera matrix $\mathtt{P}$ can be computed from the 3D-2D correspondences as explained in section 2.2.7.
  2. **Structure degeneracy** : When all the 3D points in view are coplanar, the fundamental matrix cannot be uniquely determined from image correspondences alone.

Some preprocessing of the video may result in fewer frames for optimal reconstruction, with low computational cost, long enough baseline distance, known camera pose, known motion,and avoidance of degenerate cases.

2.3.2 Researchers Approach

Seo et al. (2003) considers three factors to extract key frames, (a) the ratio of the number of correspondences to the features number, (b) the homography error, and (c)the distribution of correspondences over the frames. The score function they propose is given as

$$\mathtt{S} \;=\; w_1 \quad (1 - \frac{N_c}{N_f}) + w_2 \quad \sigma_c + w_3 \quad \mathtt{H}_{err},$$

where N_c and N_f are the number of correspondences and features for correspondences respectively, σ_c is a standard deviation, and $\mathtt{H}_{err}$ is homography error. w_i $(i = 1, 2, 3)$ is the weight used to provide a relative importance. Symmetric transfer error (Hartley & Zisserman, 2004) is one measure of homography error. Hartley and Zisserman (2004) states that the homography error is small when there is little camera motion between frames. Homograhpy error is used a good proxy for the baseline distance between two views. They use evenly distributed correspondences over the entire image to obtain the fundamental matrix. They divide the image into subregion and the σ_c is calculated as

$$\sigma_c = \sqrt{\frac{1}{N_s} \sum_{i=1}^{N_s} \left(N_{ci} - \frac{N_c}{N_s} \right)^2},$$

where N_s and N_c are the number of subregions and correspondences. N_{ci} is the number of correspondences in i-th region. The authors consider the first frame to be a key frame and calculate the score with the consecutive frames until the ratio is below a certain threshold. The pair with the lowest score is selected as a key frame. But they do not discuss any measure for degenerate cases.

Pollefeys and Van Gool (2002) select key frames for structure and motion recovery based on a motion model selection mechanism (Torr, Fitzgibbon, & Zisserman, 1998) to select next key frame only once the epipolar geometry model explains the relationship between the pair of images better than the simpler homography model. The distinction between the homography and the fundamental is based on the Geometric Robust Information Criterion (GRIC) (Torr, 1998). They discard all frames based on degenerate cases.

Seo et al. (2008) also use the correspondence ratio. If the ratio R_c is close to one this means the images overlap too much and baseline distance will be small. Under these assumptions frame should not be selected as a key frame. The second measure is the reprojection error.

$$RE = \frac{\sum d(\mathtt{P}_i\mathtt{X}_j;\mathtt{x}_j^i)^2}{\mathtt{M}},$$

where $\mathtt{X}_i$ is a 3D point and $\mathtt{x}_i$ is the measured 2D point. The relationship between these two points is given by $\mathtt{x}_i = \mathtt{P}_j\mathtt{X}_i$ for all $\mathtt{x}_i$, where $\mathtt{P}_j$ is the 3×4 camera matrix. Seo et al. (2008) do not analyze degenerate cases.

2.4 Browser-Based 3D Video Player

The concept of the browser-based 3D video player is that a user can view a video in the player and at any stage can pause the video and change the mode to free view and move the viewpoint freely in the 3D scene. To fulfill our requirements, in this section, I review different possibilities for a 3D player to be embedded into a web browser. The following sections discuss the pros and cons of different plug-ins supporting 3D rendering in web browsers.

2.4.1 OpenGL

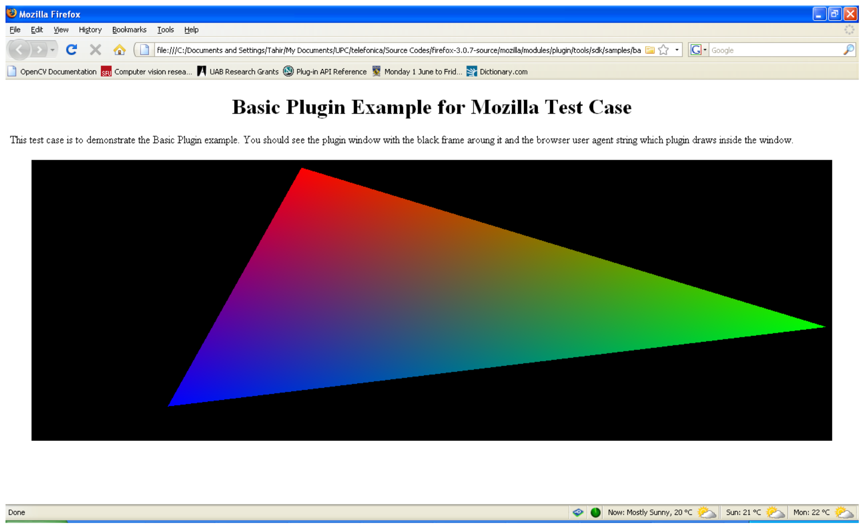
OpenGL is one of the most commonly used 3D rendering libraries. Initially our concept was to develop an engine to render OpenGL in the browser. In order to do so, we developed a plug-in for Gecko based browsers.

2.4.1.1 Development

Gecko based browsers provide an API for third part plug-ins. The web page, calling a plug-in completely controls the life cycle of the plug-in. On startup, Netscape Gecko checks for the plug-in modules in plug-ins folder for the browser application. The browser performs the following sequence when the user opens a page containing embedded data of a media type associated with the plug-in.

1. The plug-in is checked based on its MIME type.
2. The plug-in is loaded into memory.
3. The plug-in is initialized.
4. A new instance of the plug-in is created.

Multiple instances of the same plug-in on a single page or in several open windows can be loaded by Gecko at the same time. For example the browser will create as many instantiations of the RealPlayer plug-in as are needed when the user is browsing a page


Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Documents and Settings/Tahir/My Documents/UPC/telefonica/Source Codes/firefox-3.0.7-source/mozilla/modules/plugin/tools/sdk/samples/ba
OpenCV Documentation
Computer vision resea...
UAB Research Grants
Plug-in API Reference
Monday 1 June to Frid...
Dictionary.com
Done
Now: Mostly Sunny, 20 °C
Sun: 21 °C
Mon: 22 °C

Basic Plugin Example for Mozilla Test Case

This test case is to demonstrate the Basic Plugin example. You should see the plugin window with the black frame aroung it and the browser user agent string which plugin draws inside the window.

Figure 2.14: OpenGL embedded in FireFox Browser.

that clips. The plug-in instance is deleted when the user leaves the page or closes the window. The plug-in code is unloaded from memory when the last instance of a plug-in is deleted. When the plug-in is not loaded, it only consumes disk space and no other resources. Complete details are given in DevEdge (2001-2003). We used this API to develop a test plug-in for embedded OpenGL. Figure 2.14 shows a triangle which is coded in OpenGL and rendered in the browser.

2.4.1.2 Limitations

There are a number of limitations to the approach discussed above.

- It is browser specific. The 3D scene can only be rendered in Netscape and Mozilla browsers.
- Operating system specific coding is required.
- User interaction classes need to be programmed.

Due to these limitations, we considered a cross plate form alternative Microsoft Silverlight.

2.4.2 Silverlight 2.0

In order to achieve cross plate form support for all the major browser and operating system, and to provide high definition video and a rich interactive applications, we switched from Gecko to Microsoft Silverlight. Some of the Silverlight features are given below in summarized form; details can be seen in ScottGu (2007).

- Silverlight can efficiently download and play media from any web server.
- Silverlight supports built-in streaming is also optionally supported by Silverlight.
- Silverlight enables rich UI, animations, and vector graphics.
- Silverlight also makes it easy to build a rich, interactive video player.

Silverlight itself is not a 3D engine but it provides another API KIT3D which can be used for 3D development. KIT3D is a 3D graphics engine written for Microsoft Silverlight. It can be used to render and allow free viewing of 3D scenes in the browser. KIT3D supports cameras and 3D viewport3D similarly to OpenGL. 3D points, 3D vertors, and 3D matrices are used for geometric transformations. The API also supports 3D translation and rotations. Hence, it provides all of the basic tools needed for 3D geometry. We found Silverlight and KIT3D sufficient for our requirements for the front end.

2.4.3 Silverlight 3.0 Beta

While we were working on Silverlight version 2, Microsoft made the beta of Silverlight 3.0 available on March 18, 2009. We adapted our code and functionality to the latest version. Details are given in section 3.4. A brief description about some of the features of Silverlight 3.0 beta is given here.

- The graphics support is improved, including 3D graphics support, animation features, and hardware acceleration.
- The new version supports higher quality video and audio.
- It has now perspective 3D graphics by applying contents to a 3D plane.
- Silverlight 3 has pixel shading effects, including blur and drop shadow.
- It now detects the internet connectivity.

Chapter 3
Methodology

3.1 Overview

The methodology i used for the development of the project is described in this chapter. The methodology is divided into the following sections. Section 3.2 gives a brief overview of the architecture of the project. Section 3.3 explains the methodology used in the back end reconstruction process. The back end process includes a database, a key frame extractor and a 3D reconstruction algorithm. Section 3.4 discuss the systems, a graphical user interface (GUI). The browser-based 3D video player is the main component of the front end.

3.2 System Architecture

This research study's focus is to contribute toward the development of a product based on structure from motion whose aim is to visualize a video sequence. The tools used for the development are C with the Open Computer Vision (OpenCV) Library for the back end processes and Microsoft Visual Studio for the front end. A block diagram of system architecture is given in Figure 3.1. Figure 3.1 shows the foreseeable system architecture for the final product.

3.3 Back End Processing

There are three major components in the back end: key frame extraction, structure from motion (SFM), and a database to store the information created through SFM. The information in the database is utilized in the front end. The details of each component are given below.

3.3.1 Key Frames Extraction

Key frames extraction is the focus of my research. The need for key frame extraction was explained in Section 2.3. A detailed method for key frame extraction is given below, and a brief description of each step in our SFM process is explained in Section 3.3.2. There are three main criteria to classify a frame as key frame:

1. The cameras must have a *baseline* long enough for triangulation.
2. There are two conditions for non-general camera motion and non-general position of structure known as *degenerate cases* when epipolar geometry calculation fails.

3.3.1.1 Baseline Distance

To ensure a long enough baseline distance, understanding of camera movement is required. Baseline distance is the distance between the two camera centers. The baseline distance affects the region of uncertainty in the reconstruction, as shown in Figure 3.2. Seo et al. (2003) use homography error as a proxy for baseline distance. Homography error can be computed using symmetric transfer error (Hartley & Zisserman, 2004).

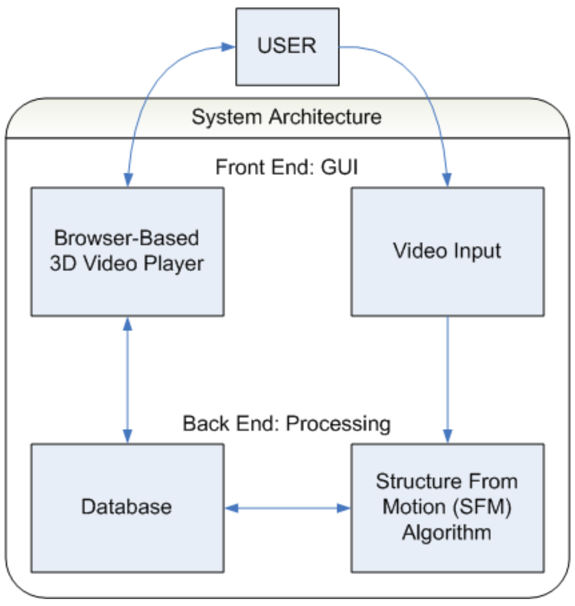


Figure 3.1: High-level system architecture. User inputs a video, which is processed in SFM. The parameters created by SFM are stored in the databased which are later utilized by the browser to display the video in browser-based 3D video player.

Seo et al. (2003) report that homography error is small when there is a little camera motion between frames. This assumption seems to be correct in some cases but not for all. One case may be when there is a structure degeneracy then homography error will be small but the camera motion may not be little (see section 2.3 for degeneracy in details). The authors use the correspondences ratio, which is the ratio of number of true correspondences with respect to total features for correspondence Seo et al. (2008). The Correspondence ratio is given as

$$R_c = \frac{T_c}{T_f}, \tag{3.1}$$

where R_c is correspondence ratio, T_c is true correspondences and T_f is total features for correspondences. The value of R_c is inversely correlated to the camera motion. As a simple experiment, I calculated R_c from a single frame to the rest of forthcoming frames and observed that the ratio is nearly equal to one for early frames and starts decreasing as the distance between the frames increase based on the camera motion. When there is no camera movement this value remains close to 1. Therefore, R_c is a good measure for approximating camera motion. This ratio is our first key frame selection criterion.

The first test selects a set of frames based on thresholds T_1 and T_2. Although we want a long baseline, a very long distance may cause poor reconstruction. My selection process for thresholds T_1 and T_2 is given in Section 4.3.

3.3.1.2 Fast Iteration Method

In the initial experiments with real video sequences, I observed that the correspondence ratio decreases very rapidly after the first few frames to the next few frames. It then decreases very slowly for the following frames and approaches zero when no further

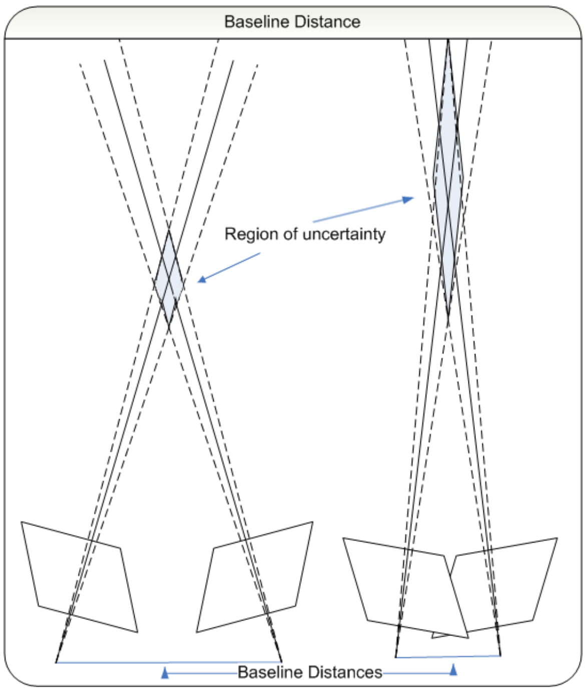


Figure 3.2: Relationship between baseline distance and triangulation uncertainty. With an appropriate camera motion direction, as the baseline distance increases the region of uncertainty decreases.

features are matched between the pair of images (see figure 3.3). I developed a fast iteration method for extracting a set of candidate key frames based on these observations. The early frames have small baselines and will be rejected by threshold T_1. For the following frames, the ratio typically decreases very slowly, therefore I uniformly sampled the sequence of frames with an interval I. There may be a chance that a suitable frame exists within an interval. I worked out to search for that suitable frame using the fast iteration method.

Let us consider the specific case shown in Figure 3.4. The first frame is a key frame. The first iteration finds the correspondence ratio between frame 1 and 5. If the ratio is within the thresholds T_1 and T_2, it will be selected for the set and frame 10 will be the next candidate frame. If the ratio is greater than T_1 as shown in Figure 3.4. It will jump to the next sample i.e. frame 10. If the ratio of frame 10 is less than T_2, the method will search for the frame within the interval 5-10. There may be a chance that frame 10 is an outlier due to blur image. To overcome that case the method will search in the next interval as well. Thus no frame within the limits of the ratio will lost and the frame within the range of threshold will be selected as candidate frames. One frame from the candidate frames selected within the interval is selected as key frame through further processing. Above explained processing is then carried out with the selected key frame and so forth. In our experiments ratio analysis with different camera movement is shown (see section 4.3).

3.3.1.3 Degenerate Case

The fundamental matrix `F` is used to obtain the position of one camera with respect to another one. However, camera positions may not be obtained in case of degenerate situations. The most common degenerate situations are motion degeneracy and struc-

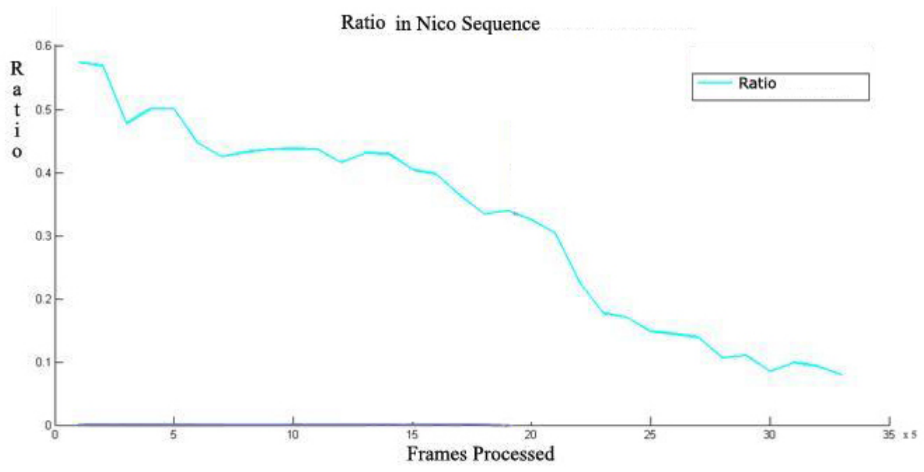


Figure 3.3: Ratio Variation. We captured Nico sequence with a typical hand recorded camera movement. The frames are multiple of 5 and the correspondence ratio decreases very slowly.

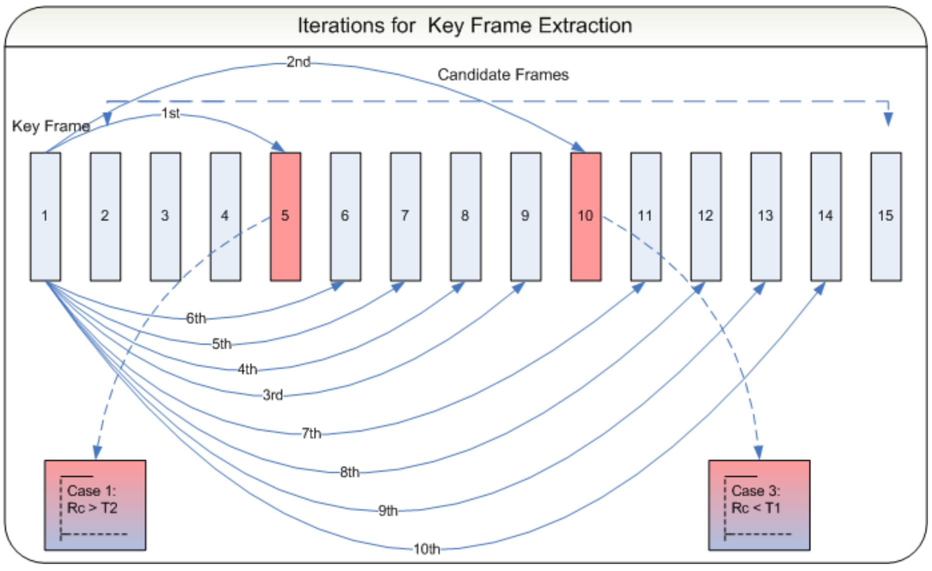


Figure 3.4: Fast Iteration Method. I searched the next key frames by jumping on the samples and then within the intervals with respect to a key frame.

ture degeneracy. Motion degeneracy occurs when the camera rotates around its center with no translation. The epipolar geometry is not defined for this case although the motion can be described by a homography and a camera matrix `P` can be estimated from 3D-2D correspondences. Structure degeneracy occurs when all the points in the structure are coplanar. In this case the epipolar geometry is not defined and it cannot be uniquely determined from image correspondences alone. But this situation can also be modeled with a homography (Torr et al., 1998).

The relationship between a pair of images with general camera motion and general structure is appropriately defined by a fundamental matrix, whereas non-general camera motion is more fittingly defined by a homography matrix. Thus, we can fit and compare two models, i.e, the fundamental matrix and a homography model, to distinguish above stated situations. To accomplish this, there must be some selection criterion for the fundamental matrix or homography model. Akaike's information criterion (AIC) is a useful statistic for model identification and evaluation (Bozdogan, 1987). AIC is based on a measure of the goodness of fit. But Torr (1998) reports that AIC fails in the presence of outliers. Another models selection criterion, the *Geometric Robust Information Criterion (GRIC)*, is given by Torr (1998). GRIC is not only based on goodness of fit but also the parsimony of the models. GRIC is a robust model selection criterion for the fundamental matrix and homography model. It provides a score for both models and the model with lowest score can be considered the one that defines the relationship between a pair of images the best. I use GRIC to identify degenerate cases. The score is

$$GRIC = \sum_i \rho(e_i^2)_i + \lambda_1 dn + \lambda_2 k, \tag{3.2}$$

where n is the total number of features matched in two frames, d is the number of dimensions modeled ($d = 3$ for `F`, $d = 2$ for `H`), k is the number of parameters of the model ($k = 7$ for `F` and $k = 8$ for `H`) and $\rho(e_i^2)$ is a robust function of the residual e_i. Parameters λ_1 and λ_2 are explained below in detail. GRIC can be further divided into two main parts, i.e, the goodness of fit and the parsimony of the model.

- **Goodness of Fit**: The residuals measure the goodness of fit:

  $$\rho(e_i^2) = \min(\frac{e^2}{\sigma^2}, \lambda_3(r-d)), \tag{3.3}$$

  where r is the dimension of the data (for two frames $r = 4$, one for each coordinate x, y, x', y'), σ^2 is the variance of the error and d is same as in equation (3.2). The residuals are calculated for each model considering the possible correspondences. To calculate residual for H, we used symmetric transfer error given in Hartley and Zisserman (2004) and for F we used point to epipolar line cost (see equation (2.5)). There is a possibility of very huge error which may probably affect the goodness, thus, it is discarded using $\rho(e_i^2)$. λ_3 is a weight to limit the residual error which is set to 2 in our case.

- **Parsimony**: The remaining term penalize the model complexity. $\lambda_1 dn + \lambda_2 k$ is constant for each model and each frame. If we discard the goodness to fit, the value of the GRIC score for the homography would always low because the parsimony of the homography model is less than the fundamental model. Therefore if the fundamental model does not return a small residual cost, the homography model is the only choice.

  - $\lambda_1 dn$ is penalty term for structure and $\lambda_1 = \ln(r)$.

– $\lambda_2 k$ is a penalty term for model and $\lambda_2 = \ln(rn)$.

Our experiments with GRIC indicate that it reliably distinguishes degenerate cases so we use this technique in our algorithm (see section 4.2 for sample results on distinguishing degenerate cases).

3.3.1.4 Frame Selection

Up to this stage, considering the first frame as a key frame, we have a set of frames selected using the upper and lower thresholds on the correspondence ratio R_c. Next, the relationship of each frame in the set to the key frame is represented by the best model, F or H, given by GIRC. We consider two techniques for key frame selection:

- **Key frame extraction based on GRIC difference (KFGD)** The first technique is based on GRIC score difference. If the scores of the fundamental matrix and homography models are very different, according to GRIC the relationship between frames is much better represented by the model with the lowest score. To use the GRIC difference to identify key frames, we let

$$\mathtt{K}_{i+1} = \text{argmax}_{j \in \phi(K_i)}(f(\mathtt{K}_i, j)), \tag{3.4}$$

where $\mathtt{K}_i$ is the index of the i-th key frame in the video sequence, $\phi(K_i)$ is the set of frames for which the correspondence ratio with frame K_i is within threshold. The function f is given as

$$f(i,j) = \frac{|g_{\mathtt{F}}(i,j) - g_{\mathtt{H}}(i,j)|}{g_{\mathtt{H}}(i,j)}, \tag{3.5}$$

where function g is given by Equation (3.2), subscript F and H shows the model used to calculate the score, and i and j are the indexes of the frames. KFDG is a good measure to select key frames because it provides very low variation in reprojection error as compared to uniformly sampled frames. This is demonstrated in my experiments in chapter 4.

- **Key frame extraction based on Weighted Score (KFWS)**

The KFGD method may be used to stabilize variation in reprojection error, but as we shall see in my experiments, it has a little effect on the mean error compared to uniformly sampled frames. High reprojection error is usually due to outliers in matching or because of a loose threshold set for inliers. From experiments, I analyzed the point to epipolar line cost and KFGD over many frames, and observed some frames in which the variation in the GRIC difference was very small but the variation in the point to epipolar line cost was very high. In such cases, the selection criterion should be based on both the GRIC difference and the point to epipolar line cost with high and relatively low weights, respectively. I therefore propose the weighted key frame score

$$S = w_1\,(f) + w_2(\sigma - PELC), \tag{3.6}$$

where function f is given by Equation (3.5), σ is standard deviation of the error and $PELC$ is given in equation (2.5). The weights w_1 and w_2 given in equation (3.6) are set to 0.7 and 0.3, respectively, because the model representing the relationship between two frame is more important than the point to epipolar line cost which is based on outliers. The frame with the maximum score is selected as a key frame with known model type.

Frames are extracted from the video sequence and passed to the key frame extraction process. The first frame is always considered as a key frame, and we search for the next key frame based on the selection criteria just discussed. An Algorithm for key frame extractor is given below.

Algorithm 1 KeyFrameExtractor

```
Require: A video stream with $n$ frames.
Ensure: Key Frames $\phi(K)$
1: $keyframe = 1$
2: while $keyframe \leq n$ AND $keyframe \neq$ NULL do
3:     $\phi(K) = keyframe$
4:     FeatureExtraction($keyframe$)
5:     for $frame = i + l$ to $n$ do
6:         FeatureExtraction($frame$)
7:         FeatureMatching($keyframe$, $frame$)
8:         DiscardOutliers($keyframe$, $frame$)
9:         $R_c = \frac{\text{Total Correspondences}}{\text{Total Features}}$
10:        if $T_{max} \geq R_c \geq T_{min}$ then
11:            $candidateframe[index] = frame$
12:        end if
13:        if Total Correspondences $\leq$ Threshold then
14:            $selectedkeyframe = \text{KFWS}_{max}(candidateframe)$
15:            ModelSelection($keyframe$,$selectedkeyframe$)
16:            $keyframe = selectedkeyframe$
17:            Goto step 2
18:        end if
19:    end for
20: end while
```

3.3.2 3D Reconstruction

A detailed literature review of 3D reconstruction is given in Chapter 2 Section 2.2. Here I explain the process in an algorithm which we developed during the research.

Algorithm 2 RECONSTRUCTION

```
Require: A video stream with $n$ frames, increment $l$ and camera intrinsics $K$ for each
    frame.
Ensure: Parameters for database.
 1: for $i = 1$ to $n$ do
 2:     for $j = i + l$ to $n$ do
 3:         $i \leftarrow$ KeyFrameExtractor$(i, j)$
 4:         $\phi(K_{(k)}) \leftarrow i$
 5:     end for
 6: end for
 7: set $Flag$ to True
 8: for $\phi(K_{(k)})$ where $k$ is the indexes of KeyFrames do
 9:     if Model is $\mathtt{F}$ AND $Flag$ is True then
10:         Gold standard to estimate the fundamental matrix $\mathtt{F}$. It will also provide
            3D points using 2D points feature correspondence in the pair of images.
11:         Use $\mathtt{K}$ and $\mathtt{F}$ to estimate $\mathtt{E} = \mathtt{F}^{\mathtt{T}}\mathtt{KF}$
12:         Set $\mathtt{P}_1 = [\mathtt{I}|0]$ and Extract $\mathtt{P}_2$ using $\mathtt{E}$
13:         set $Flag$ to False
14:     end if
15:     if $Flag$ is False then
16:         Resectioning next Camera matrix $\mathtt{P}_i$ using Equation (2.10)
17:         Triangulate to get more 3D points $X$ using the extracted camera matrix
            $\mathtt{P}_i$ using Equation (2.9)
18:         if no. of 3D points less than Threshold then
19:             Go to 16
20:         end if
21:     end if
22: end for
23: Sparse Bundle Adjustment to optimize reconstruction
24: Store the 3D reconstruction information in the database
```

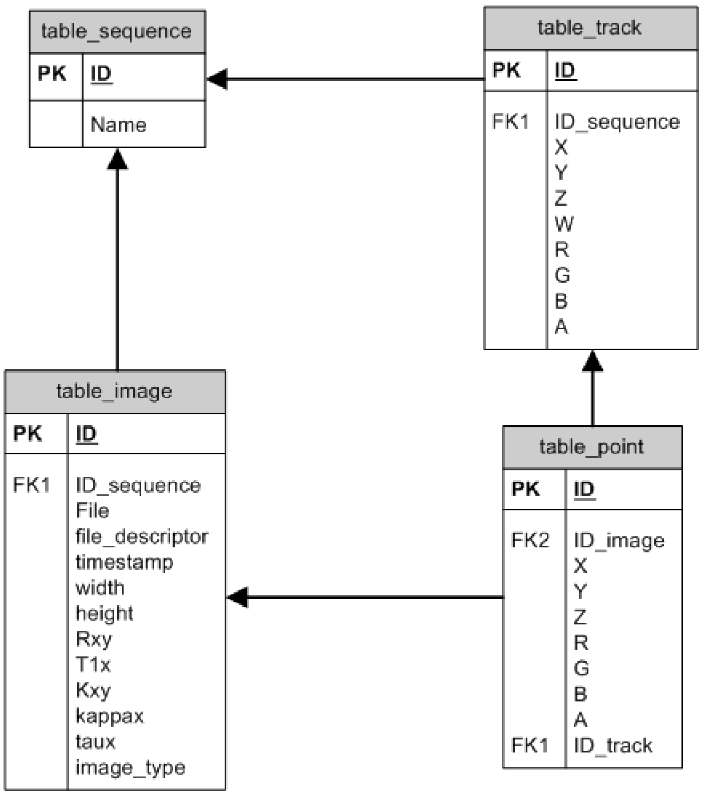


Figure 3.5: Database Model.

3.3.3 Database

As we shall see in Section 3.4, several kinds of information need to be stored in a database in order to develop a browser-based 3D video player. In this section, I briefly describe the data model (see Figure 3.5). The input from the user is a video of a scene; in order to display the scene in 3D, information about the sequence of frames, cameras, tracks, and the sequence itself has to be stored in a database. We extracted the data via the reconstruction algorithm just described. Each video is given a unique id. Each sequence is composed of several images. The 3D reconstruction algorithm processes the images and creates feature points and 3D points. A brief description of each data component is given below.

- **Sequence**: the table `table_sequence` represents the video sequence.
  - *ID*: A unique ID identifying a sequence. The web client's requests contain the ID of the sequence to be used.
  - *Name*: The path of the video file.
- **Images**: Each video sequence consists of images and each image is associated with data about its orientation, width, height, camera parameters, physical location, and image type. The table `table_image` is used to store the information about single images.
  - *ID*: A unique ID for the image.
  - *Sequence ID*: A reference to a particular sequence.
  - *File*: The URL of image file.

  - *Time Stamp*: Time at which the frame was captured.
  - *Width*: Image width.
  - *Height*: Image height.
  - *R Matrix*: The estimated rotation matrix.
  - *T Vector*: The estimated translation vector.
  - *K Matrix*: The estimated intrinsic parameter matrix to allow processing of videos with dynamic zoom in future.
  - *kappax*: Lens distortion parameter to allow processing of video of same scene with different cameras and varying intrinsics.
  - *taux*: Lens distortion parameter to allow processing of video of same scene with different cameras and varying intrinsics.
  - *Image Type*: Images can be categorized as ordinary frames, key frames with the fundamental matrix model and key frames with the homography model, see details in section 3.3.1.

- **Tracks**: Tracks correspond to the the 3D points that are the result of the reconstruction process. The cloud of points is represented by these tracks. The table `table_tracks` is used to store information on tracks.
  - *ID*: A unique ID for each 3D point.
  - *Sequence ID*: Reference to the containing sequence.
  - X: X coordinate of 3D point.
  - Y: Y coordinate of 3D point.
  - Z: Z coordinate of 3D point.
  - W: W to represent the 3D point in homogeneous coordinates.
  - R: Red color component of the 2D pixel in the pair of images.
  - G: Green color component of the 2D pixel in the pair of images.
  - B: Blue color component of the 2D pixel in the pair of images.
  - A: Alpha component of the 2D pixel in the pair of images.

- **Points**: Each frame contains 2D points obtained during matching. Resectioning requires correspondences between feature points and tracks (3D points). The table `table_track` stores the information required to do this.
  - *ID*: Unique ID for the point.
  - *Image ID*: Reference to the containing image.
  - *Track ID*: Reference to the assumed corresponding 3D point.
  - X: X coordinate of 2D point.
  - Y: Y coordinate of 2D point.
  - Z: Z to represent the 2D point in homogeneous coordinates.
  - R: Red color component.
  - G: Green color component.
  - B: Blue color component.
  - A: Alpha component.

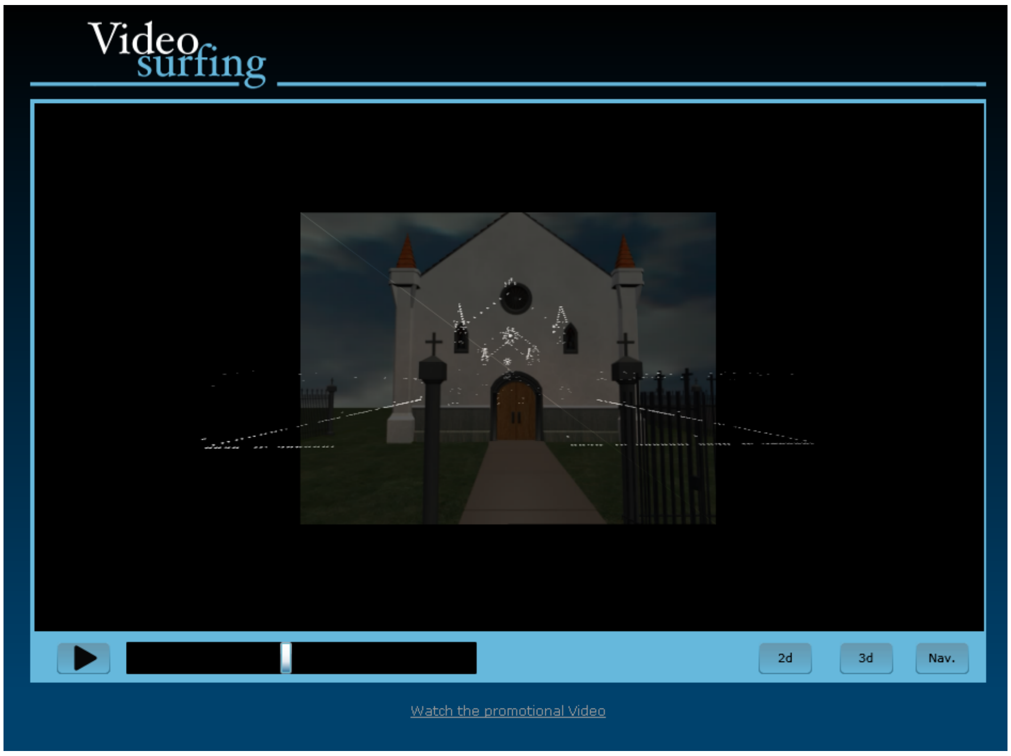


Figure 3.6: Behind the camera mode. The camera is positioned slightly behind the current view.

3.4 Front End GUI

I developed the player in Microsoft Visual Studio 2008 using Microsoft Silverlight. The user is provided a Web interface. The GUI is composed of two major interfaces: one to upload a video of any arbitrary scene, and one to view the video in different modes. The user has three possible choices of display mode:

- **Traditional**: The traditional mode allows viewing the video as in any standard video player. I gave the user standard functionality including play, pause, and move forward and backward using a scroll bar.
- **Behind-the-camera**: The behind-the-camera mode displays the video as if the user is standing behind the camera. The current frame is displayed with opacity less than one 1 to make it semi-transparent. See Figure 3.6 for an example scene.
- **Free Navigation**: This mode displays the 3D scene in a perspective view that is composed of the camera's path (the optical center of each view), the current visible frame, and the 3D reconstructed cloud of points of the scene. This mode has not only typical video player facilities, but it is also equipped with navigational options that allow movement in the 3D scene using keystrokes and mouse movements. The user can move the viewpoint freely in the reconstructed scene, while the camera and the current frame are continuously updated. See Figure 3.7 for an example scene.

A flow chart of the GUI is shown in the Figure 3.8. I explain the architecture of the viewer in the next section.

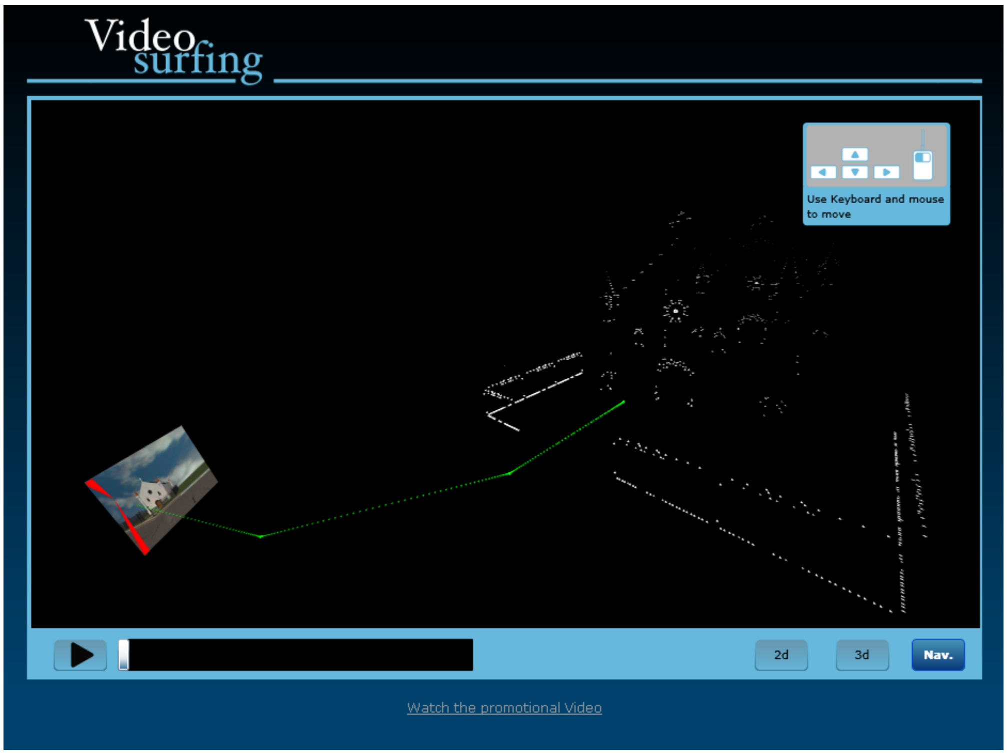


Figure 3.7: Free Navigation mode. The user can move freely in the 3D world. The visualization starts from the perspective view of scene and first camera.

3.4.1 Viewer

A graphical rendering engine is required to display the video, and a 3D rendering engine is required for the different modes explained in Section 3.4. Silverlight offers a subset of the functionality provided by Windows Presentation Foundation (WPF). The viewer has components to display the video, to project 3D objects in a perspective view, to render 3D graphics, and to transform objects in 3D. The components, I use are briefly explained blow.

- **ViewPort3D**: Viewport3D is a component of the WPF. It is actually a container of 3D objects to be displayed and rendered in scene. It also provides properties like clipping, height and width, and mouse events. It needs a Camera, a 3D geometry model (mesh), and a light object to display the scene. I use viewport 3D to show the video player, 3D movement, multiple images, camera movement, and the 3D reconstructed scene.
- **Camera**: The camera component specifies a projection of a 3D world to a 2D visual surface. The projection includes perspective foreshortening. The perspective camera is a frustum whose sides converge toward a point on the horizon to the camera center. Objects closer to the camera appear larger, and objects farther from the camera appear smaller. The camera is very important in our player. Most of the user-actions depend on the camera. In free navigation mode, the camera moves into the scene under control of the user. In the behind-the-camera mode, the orientation of camera depends on the orientation of the view. Therefore we need the plane for camera transformation and orientation of each frame in the video sequence.
- **Video3D**: The video3D component is used to display the video in 3D and to

keep the transformation information of each frame. The camera movement is visible in two modes, i.e, the behind-the-camera mode and the free navigation mode. Video3D has a sub component called Media Element which is a rectangular region that can display a video on its surface. It supports the functionality of play, pause, stop, and scroll. I use Video3D to transform that rectangular region into the 3D world so as to allow the user to see the frames displayed in their original orientation in the free navigation mode.

- **Mesh**: The Mesh component is the 3D geometry of the world. I use it to present the cloud of tracks (3D points) and the key frames. Any geometrical transformations applied to the tracks and key frames are managed by the Mesh.

To access the data for the above explained components, I equip the viewer with the database interface using a Silverlight component called Web client.

3.4.1.1 Web Client

The web client connects to the database to the viewer when a user makes requests. The Web client is a component of Silverlight. As shown in the Figure 3.8, the Web client provides the interface between the database and the viewer. The actual data resides in the database, and the Web client extracts the data it needs, following is a list of data items managed by the Web client:

- **Video**: The Web client extracts the path or URL of the video file from the database when requested by a user and hands this information to Video3D. Video3D buffers the video stream to display it.
- **Images**: The URIs of all frames are stored in the database. The Web client requests frame URIs and provides this information to Video3D. In free navigation and behind-the-camera modes, the camera movement is displayed using frames with opacity less than 1 (opacity lies in the range of 0 to 1). Therefore the position of each frame in 3D coordinates is required.
- **Tracks**: Tracks are 3D points. When these points are displayed in the 3D view port, the reconstructed scene is presented as a point cloud. The Web client fetches this information from the database and passes it to the Mesh.
- **Key frames**: Key frames are those frames used to compute the 3D structure of the scene. This information is also stored in the database. The web client extracts this information and provides it to the Mesh.
- **Ground Plane**: I define as the ground plane the plane best fitting the camera centers. The ground plane data item stores this information and is used as a key for the 3D points transformation. See Section 3.4.2 for plane fitting algorithm.

The web client provides asynchronous upload and download operations that are useful for multiple user access.

3.4.2 Plane Fitting

For arbitrary videos, we need to resolve the unavoidable world coordinate system ambiguity for SFM. I fit a plane to the optical centers assuming the best fitting plane will be approximately equal to the ground plane. I use the algorithm by Weingarten, Gruener, and Siegwart (2004). I estimated the plane $Ax+By+Cz+D=0$ minimizing

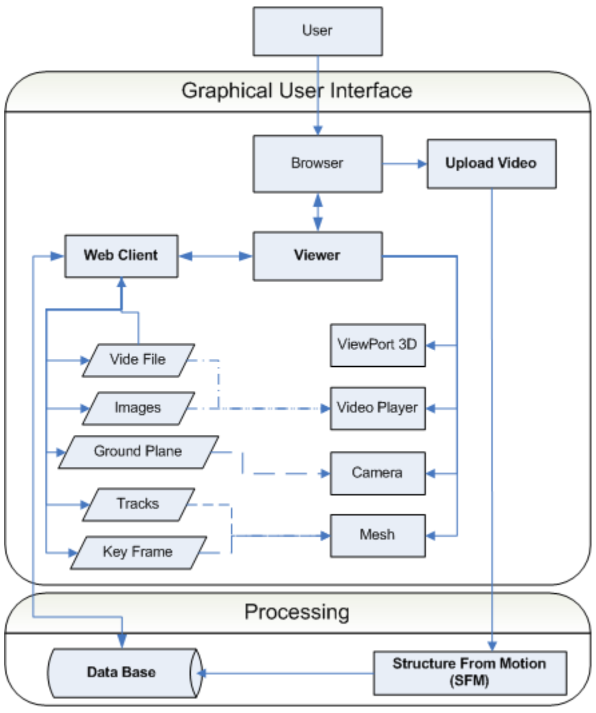


Figure 3.8: GUI Flow chart. The internal process of Graphical User Interface is shown.

the distance of the camera optical centers to the plane. In our experiments, we assume that at least 50% optical centers are inliers. Given any three camera centers, we calculate the plane containing them therefore we can calculate all possible planes from the combinations of the optical centers. The distance of all the remaining cameras optical centers to each plane is a measure of error. The median based on the distance for all the planes and it is used to distinguish inliers planes:

$$\pi = \underset{\pi_i}{\operatorname{argmedian}}(d(\mathbf{x}_j^i, \pi_i)),$$

where, j is index of all the remaining camera optical centers, i is the index of the planes, and π_i is the best plane. Finally, we processed all the planes, which are inliers, to find the optimal plane. The equation is

$$\begin{pmatrix} x_0 & y_0 & z_0 & 1 \\ x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{n-1} & y_{n-1} & z_{n-1} & 1 \\ x_n & y_n & z_n & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Leftrightarrow \quad \mathtt{A}\pi = 0 \qquad (3.7)$$

The plane with minimum distance can be obtained by taking singular value decomposition of $\mathtt{A}$: $\mathrm{SVD}(\mathtt{A}) = \mathtt{UDV}^{\mathtt{T}}$. The plane $\vec{\pi}$ is a vector associated with the last column of $\mathtt{V}$, which is the preferred plane.

Chapter 4
Experimental Results

4.1 Overview

This chapter describes experiments concerning degenerate cases, ratio thresholds, a measure of accuracy and, finally the reconstructed cloud of points with key frames in the browser. I perform experiments with both a synthetic sequence and real data. We generated the synthetic sequence with Blender for benchmarking the proposed algorithm, the sequence contains degenerate cases and provides ground truth depth for every 2D point (Blender Artists, 2000). Information about the synthetic sequence and the real sequences is given below.

- **Synthetic Sequence**
  - **Ground Truth (CHURCH) Sequence**: We use a 3D model of a church from Blender Artists (2000). We generated an animation of the model consisting of 930 frames. The animation is outdoors with sky in the background. We extracted the 3D points, projected 2D points, and camera projection matrices using a Python script. The coordinate system of the sequence has its origin in the center of the scene. We inserted degenerate cases of both types by rotating the camera view point about its center or zooming in on planar surfaces.
- **Indoor Sequences** These sequences were captured inside Telefonica R & D, Barcelona with a Sony HDR camera. We performed manual calibration of the camera intrinsics using a chessboard pattern.
  - **Library Sequence**: A real sequence of the library at Telefonica R & D, Barcelona. There are 1500 frames in the sequence.
  - **Lunch Room Sequence**: A real sequence of the Lunch Room at Telefonica R & D, Barcelona. There are 1500 frames in the sequence.
  - **Imagenio Sequence**: A real sequence of the Imagenio room at Telefonica R & D, Barcelona. There are 1500 frames in the sequence.
  - **Nico Sequence**: A real sequence of a person (still) sitting on a chair. There are 1000 frames in the sequence.
  - **Photocopy Machine Sequence**: A real sequence of photocopy machine. There are 1200 frames in the sequence.

Section 4.2 covers the experiments related to finding degenerate cases. Experiments for calculating thresholds for ratio are described in Section 4.3. A comparison of uniformly sampled key frames, key frames extracted based on the GRIC difference (KDGD) and key frames extracted based on the weighted score (KFWS) is described in Section 4.4.

4.2 Degenerate Cases

We used The *GRIC* score to distinguish degenerate cases from general motion and structure.

Sequence	False Positive	True Positive
Church Sequence	3	26

Table 4.1: Degenerate motion and structure detection.

I tested degenerate case identification using the ground truth sequence. I processed every 10th frame and manually identified 26 degenerate frames. Figure 4.1 shows some of these frames. The frames from 441-571 correspond to a camera rotating around its center. The GRIC scores for both `H` & `F` models over the sequence are shown in Figure 4.2.

Table 4.1 shows the overall results of the GRIC score-based method for degeneracy detection. My algorithm detected all 26 degenerate frames as degenerate cases along with 3 false detections.

4.3 Baseline Distance

The baseline between a key frame and a subsequent frame is the basic criterion for selecting a set of frames for further processing. Since the actual baseline cannot be determined from video alone, a good proxy is the the ratio of correspondences to the total numbers of features for two frames. A ratio close to 1 indicates that the two frames are almost overlapping. In a video sequence, consecutive frames almost always have very small baseline. To understand how to select key frames with long enough baseline distance, we computed the ratio in difference video sequences with different camera motion. Figure 4.3 shows correspondence ratio data for four different video sequences. These sequences include the Library, Imagenio, Nico, and Church sequences. The Library and Nico sequences were captured with very slow camera motion, and the Imagenio and Church sequences were captured with relatively fast camera motion. We fix the first frame as a key frame then calculate the correspondence ratio between the first frame and subsequent frames until we get very few correspondences. See Figure 4.5. In the Library sequence a sudden decrease in the correspondence ratio at 8th frame may be due to noise or blurred image. The Imagenio sequence frames for which the data in Figure 4.4 are plotted. The camera motion is very small in first nine frames and a relatively comparable camera motion in last eleven frames. The correspondence ratio results for these frames (see blue line in Figure 4.3) show similar variation.

A similar pattern of ratio with respect to camera motion is occurring in other sequences. We can conclude that slow camera motion results in slowly decreasing correspondence ratio and fast camera motion results in quickly decreasing correspondence ratio. Next we analyze GRIC variation to fix a lower bound on the correspondence ratio threshold. I find that near by frames are more precisely represented by a homography than the fundamental model, and these frames result in high correspondence ratios as shown in Figure 4.5 and the baseline distance is very small. Therefor an upper threshold for correspondence ratio is adjusted where the two motion models interchange using the GRIC scores, i.e, homography model to fundamental matrix model. The lower bound of the ratio is based on the number of inliers required to compute a precise fundamental matrix. This is only a selection criterion for a frame to be processed further. I use these lower and upper bounds to find a set of candidate frames. I then further process these frames, choosing only one as the next key frame.

In the Library sequence, the camera motion is slow, so the ratio decreases slowly. When the model selection results from one model to other, the correspondence ratio is

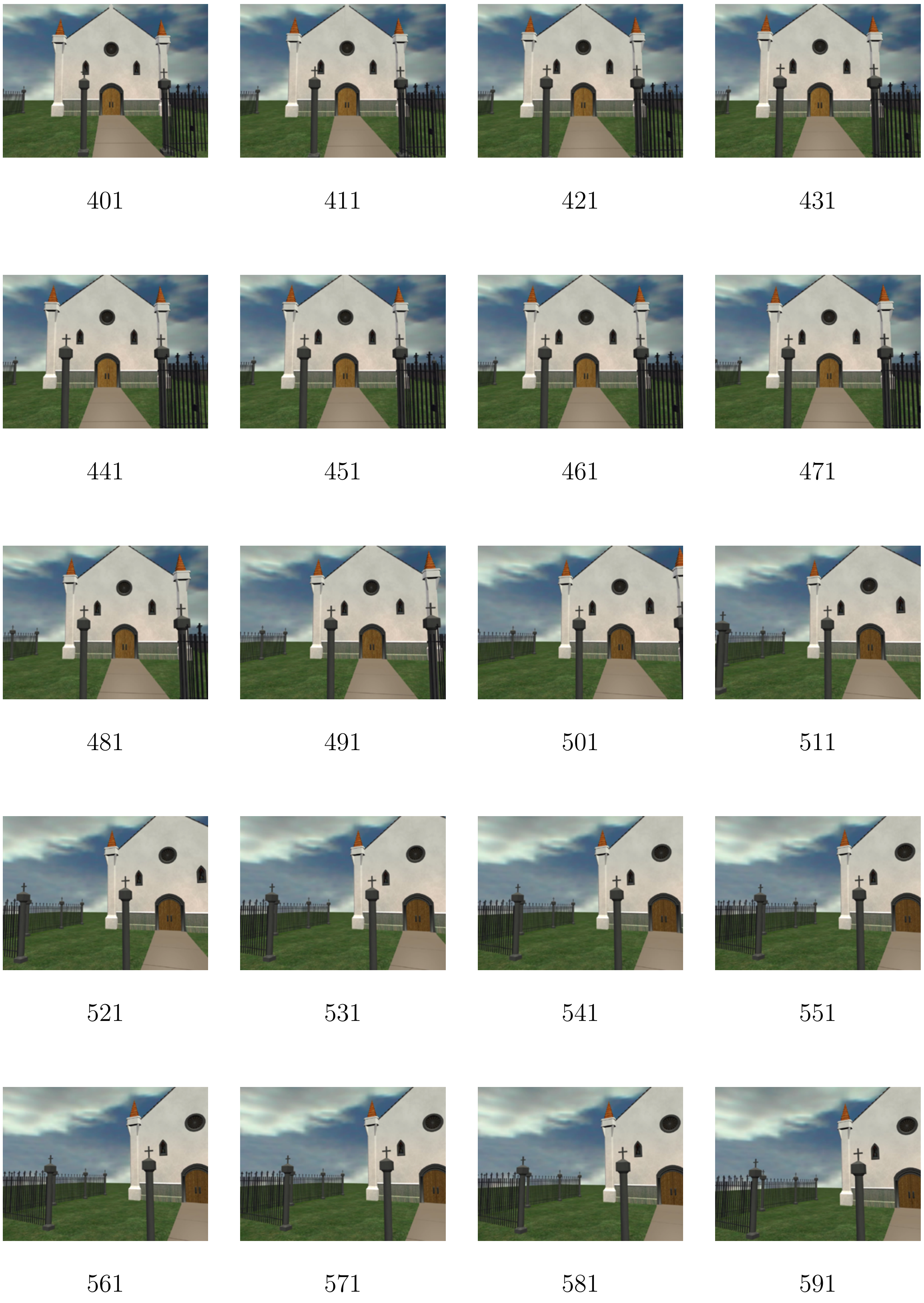


Figure 4.1: Example degenerate motion in ground truth sequence. From frames 441 to 571, the camera rotates about its center without translation.

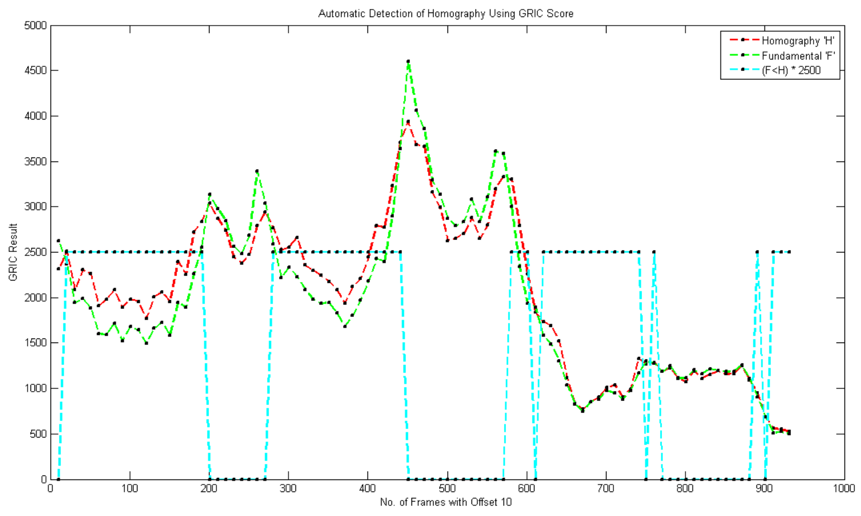


Figure 4.2: GRIC scores for homography and fundamental matrix model for the ground truth sequence. Red dash dot lines representing the GRIC score using the homography model and Green dash dot line represent the GRIC score using the fundamental model. To differentiate the selected model, a cyan dash dot line is placed which is set to 2500 if the fundamental model is selected else it is set to 0. The dots represents the occurrence of a frame.

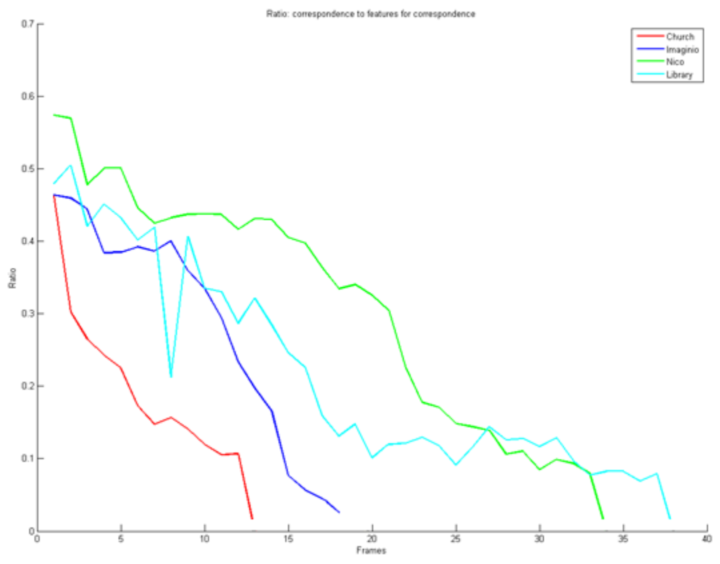


Figure 4.3: Correspondence ratio for first frame and the subsequent frames for four different video sequences with different camera motion.

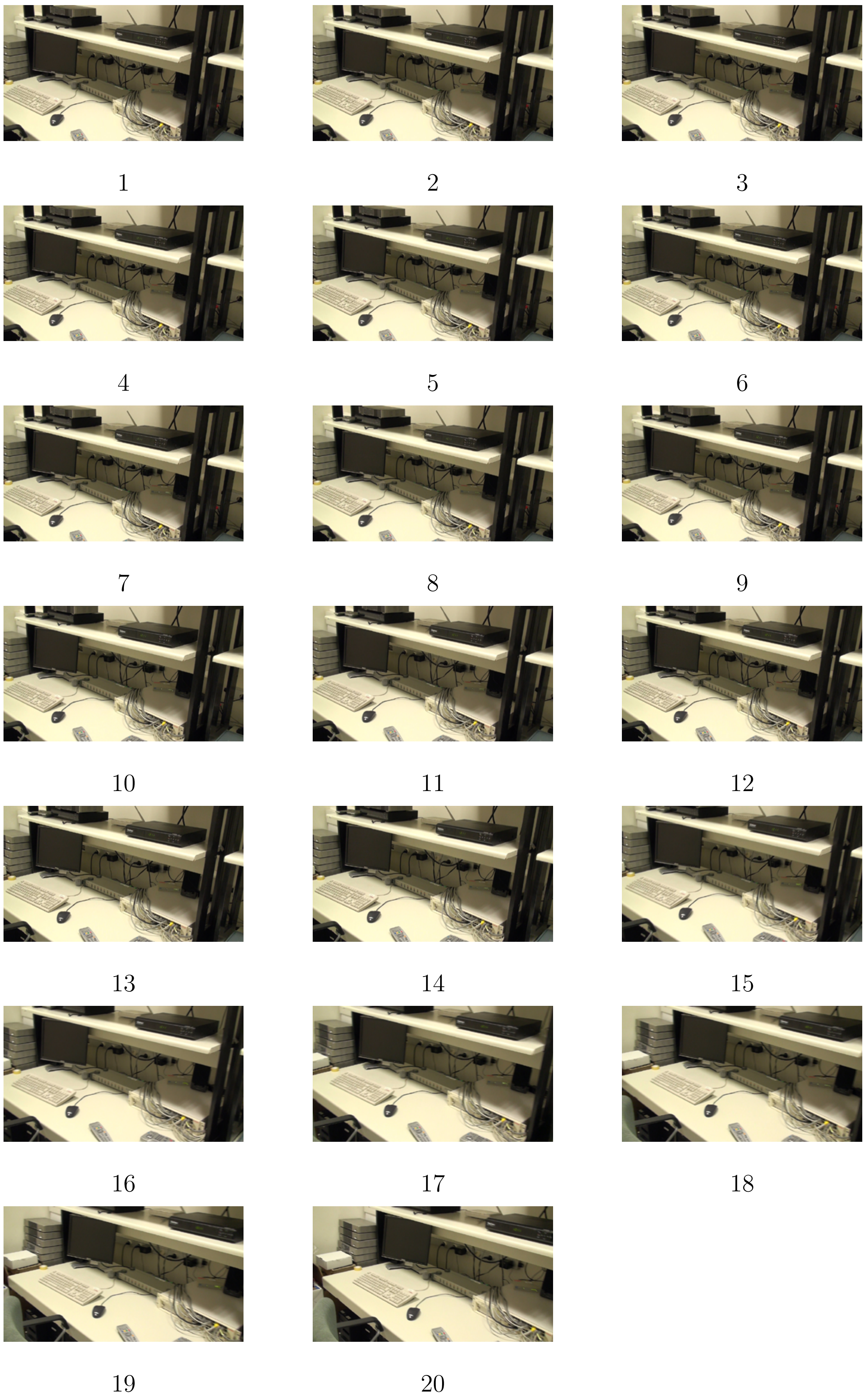
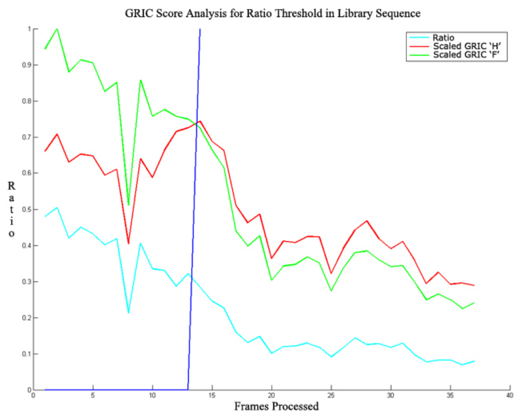
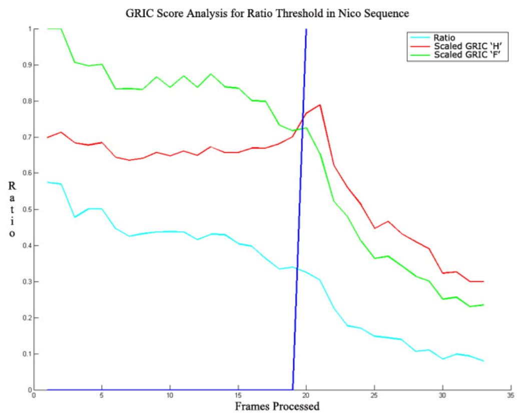


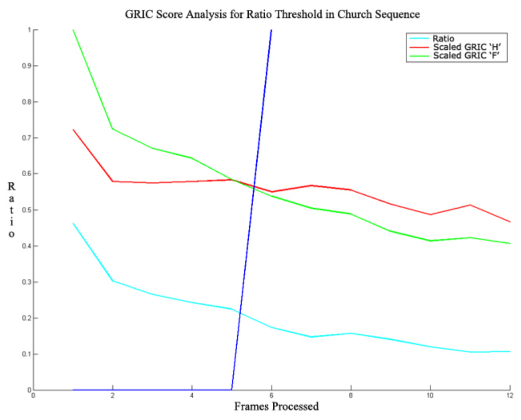
Figure 4.4: Imagenio sequence frames corresponding to the correspondence ratio data given in Figure 4.3.



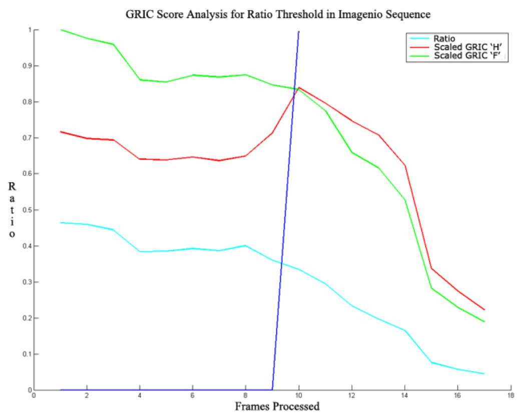
Library Sequence



Nico Sequence



Church Sequence



Imagenio Sequence

Figure 4.5: The correspondence ratio upper threshold is selected when the model selection result changes from one model to the other.

0.284. Similarly 0.334, 0.224 and 0.325 are the correspondence ratios for Nico, Church and Imagenio sequences shown in figures 4.5. In each figure, the cyan line shows the correspondence ratio, the green line represents the scaled GRIC score with the fundamental model, the red line represents the scaled GRIC score with the homography model, and the blue line denotes the cutoff where the GRIC score of two models cut each other.

4.4 Methods

I compare three methods for key frames extraction. In the first, I uniformly sample frames using a fixed offset and calculate the reprojection error. I do the same with KFGD and KFWS. The uniformly sampled frames are described in Section 4.4.1, the result for KFGD are described in Section 4.4.2 and the results for KFWS are described in Section 4.4.3. A tabular comparison is given in the Section 4.4.4.

4.4.1 Uniform Sampled Frames (Fixed Offset)

We sample frames with a fixed offset, assuming each sampled frame is a key frame. We then performed 3D reconstruction according to the method in Section 3.3.2 and plotted the reprojection error for each key frame. The results are plotted in Figure 4.6. The method produces moderately large variance in reprojection error. To compare this method with other methods we set the offset for uniform sampling based on the number of frames extracted by KFGD and KFWS.

4.4.2 Key frame selection based on GRIC difference (KFGD)

The method developed for key frame based on GRIC difference using (3.4) is experimented. In this section, I will just provide a brief comparison between uniformly sampled frames and key frame selection based on GRIC difference from Figure 4.6.

In Imagenio sequence the high peaks of reprojection error are visible for both uniformly sample frame and KFGD, therefore the mean and variance of both methods are very close. I cannot find a good improvement using KFDG in this sequence. In Photocopy sequence, there are couple of very high peaks for uniformly sampled frame but the mean reprojection error is less than the mean reprojection error using KFDG method but the variance is higher than KFDG. Similarly in Nico and Library sequences the mean is close to the uniformly sample frame method but variance is lower.

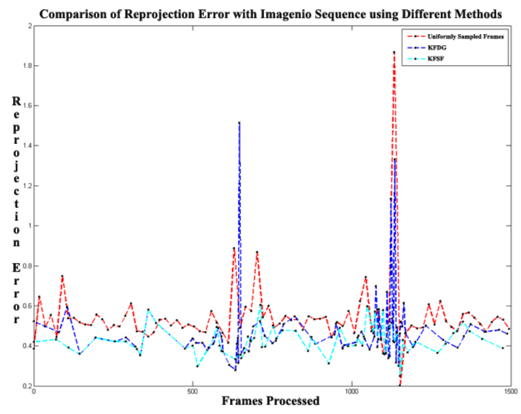
From the given sequences we can say that KFDG performs well because of the low variance of reprojection error but we performed some more experiments to improve our results which are shown in next section (section 4.4.3).

4.4.3 Key frame selection based on weighted score (KFWS)

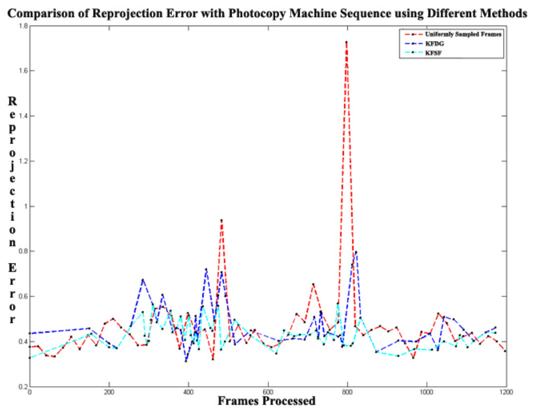
My second method for key frame extraction is based on the weighted score including GRIC difference and point to epipolar line cost given in Equation (3.6). are show in figures 4.6. There are four different video sequences processed with three methods. In Imagenio and Library sequences, the curve of KFWS is very stable in term of variance of reprojection error. It can be considered as a reliable measure and mean is also relatively low as compared to other two measures. KFGD's mean is very similar to the means using uniformly sampled frames. The variance is also not comparable to KFWS. Uniformly sampled frames not only has high variance but also is not able to give any idea of degenerate cases. The numerical differences of each method with five video sequences is given in section 4.4.4.

4.4.4 Comparison

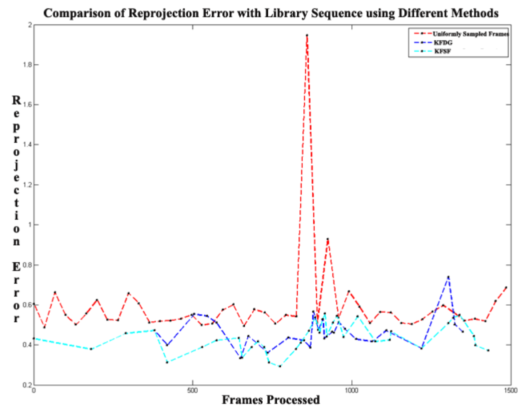
A numerical comparison of the three key frame selection methods is shown in Table 4.2. KFDG has a very low variance in all the video sequences as compared to uniformly distributed frames. This method provides a good measure of the relationship (goodness of fit) between the pair of frames and model selection as well. The only reason for a more robust method is that the mean reprojection error for KFGD is not much better than that of uniform sampling, due to a few outliers. Therefore we introduce the KFWS method based on KFDG. Although there is little improvement on our synthetic sequence, with all of the real sequences, KFWS has lower variance, lower mean and lower maximum reprojection error.



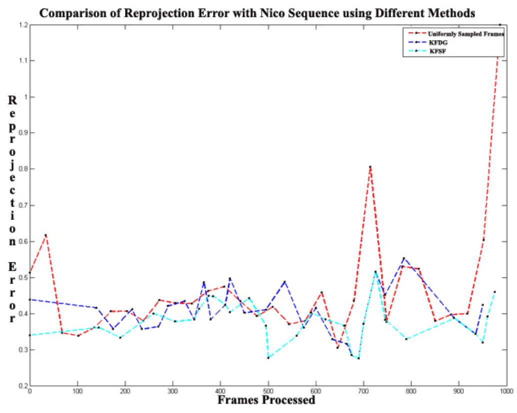
Imagenio Sequence



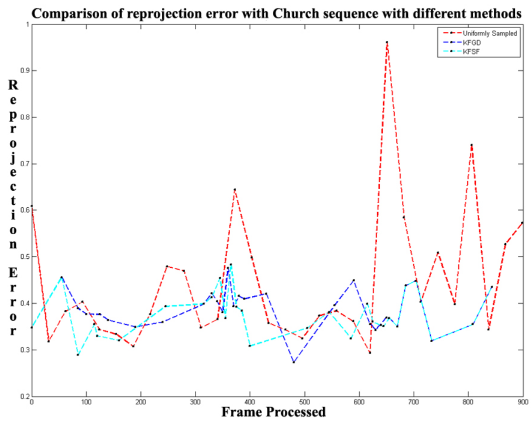
Photocopy Machine Sequence



Library Sequence



Nico Sequence



Church Sequence

Figure 4.6: Graphical Comparison. Weighted score *(KFWS)* is represented by cyan lines, GRIC difference *(KFGD)* is represented by blue lines and *uniformly sampled frame* is represented by red lines based on reprojection error.

Sequence	**Method**	**KF**	**F**	**H**	**Reprojection Error**			
					Min	**Max**	**Mean**	σ
Nico Sequence	Fixed	30			0.3052	1.2125	0.4668	0.0290
	KFGD	29	23	6	0.2760	0.5533	0.4032	0.0043
	KFWS	29	21	8	0.2760	**0.5155**	**0.3780**	**0.0031**
Photocopy Sequence	Fixed	51			0.3210	1.7264	0.4649	0.0368
	KFGD	51	36	15	0.3210	0.7955	0.4707	0.0116
	KFWS	58	45	13	0.3274	**0.5682**	**0.4324**	**0.0036**
Imagenio Sequence	Fixed	84			0.2016	1.8660	0.5415	0.0294
	KFGD	79	46	33	0.2786	1.5150	0.4772	0.0363
	KFWS	65	47	18	**0.2493**	**0.6042**	**0.4275**	**0.0049**
Library Sequence	Fixed	43			0.4681	1.9595	0.6108	0.1434
	KFGD	34	22	12	0.3362	0.7222	0.4738	0.0046
	KFWS	37	24	13	**0.2519**	**0.4852**	**0.3971**	**0.0022**
Church Sequence	Fixed	30			0.2929	0.9608	0.4451	0.0221
	KFGD	37	23	14	0.2627	0.5498	0.3878	0.0029
	KFWS	29	20	9	0.2752	**0.5164**	0.3748	**0.0026**

Table 4.2: Comparison based on reprojection error. Uniformly sampled frame method is shown as 'Fixed' and rest are same.

Chapter 5
Conclusion

5.1 Overview

3D reconstruction is the process of recovering 3D structure from a number of 2D images. A huge amount of literature exists in this area. We set out to provide a facility for 3D reconstruction for anyone anywhere to capture a video and view the reconstructed scene in 3D and to be able to move freely in the reconstructed world. The process is to extract all the key frames from the video sequence and perform projective reconstruction. Projective reconstruction can be achieved by matching corresponding features and finding the epipolar constraints between a pair of images. Camera calibration is required for Euclidean reconstruction. The resection algorithm computes camera matrices for the next frame using the 3D points and features computed from previous frames. Triangulation between a new camera and previous frames increases the number of 3D points using camera matrices computed by resectioning. The resection and triangulation processes performed, iteratively, result in a sparse 3D reconstruction. Sparse bundle adjustment optimizes the camera matrices and 3D points.

5.2 Contribution

This research study concentrated on finding suitable frames for reconstruction and discarding unnecessary frames through key frame extraction. Key frame extraction is a process to improve performance and serve the 3D reconstruction process by minimizing compute time. Sections 2.3 and 3.3.2 illustrate that all frames cannot be used for reconstruction simultaneously. Therefore, it is a prerequisite for the process of 3D reconstruction to have suitable frames for an initial reconstruction. More specifically the following are my contributions:

- An effective procedure for identification of degenerate cases in a video sequence.
- An effective method for classification of suitable frames for 3D reconstruction.
- Development of a web browser based 3D video player.

5.3 Future Enhancement

The current research work can be enhanced for outdoor environments and arbitrary videos.

- Camera auto-calibration is mandatory to make the method work with arbitrary video sequences.
- Motion detection is required to compute reconstruction of the scene and to avoid moving objects in the video. Usually in outdoor environment, there are moving objects in the scene that may be discarded to reconstruct the scene.

References

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up Robust Features. In *European conference on computer vision* (pp. 404–417).

Blender Artists. (2000). *3D models and animation.*

Bozdogan, H. (1987). Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrika*, *52*(3), 345-370.

Bradski, Rost, G., Kaehler, & Adrian. (2008). *Learning OpenCV, 1st edition.* O'Reilly Media, Inc.

Cha, J., Cofer, R., & Kozaitis, S. (2006). Extended Hough transform for linear feature detection. *Pattern Recognition*, *39*(6), 1034–1043.

DevEdge. (2001-2003). *DevEdge* [web page]. World Wide Web electronic publication.

Harris, C., & Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of the fourth Alvey vision conference* (pp. 147–151).

Hartley, R. I., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (Second ed.).

Lourakis, M. I. A., & Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software*, *36*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110.

Microsoft Cooperation. (2009). *Microsoft Photosynth.*

Noah, S., M., S. S., & Richard, S. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, *80*(2), 189–210.

Pollefeys, M., & Van Gool, L. (2002). Visual modeling with a hand held camera. *Journal of Visualization and Computer Animation*, *13*, 199–209.

Ranganathan, A. (2004). *The levenberg-marquardt algorithm.*

ScottGu. (2007). *Silverlight* [web page]. World Wide Web electronic publication.

Seo, Kim, Doo, & Choi. (2008). Optimal keyframe selection algorithm for three-dimensional reconstruction in uncalibrated multiple images. *Society of Photo-Optical Instrumentation Engineers*, *Vol. 47.*

Seo, Kim, Jho, & Hong. (2003). 3D estimation and keyframe selection for mach move. *ITC-CSCC.*

Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH conference proceedings* (pp. 835–846). New York, NY, USA: ACM Press.

Torr. (1998). Geometric Motion Segmentation and Model Selection. *Philosophical Transactions of the Royal Society London*, *356*, 1321–1340.

Torr, Fitzgibbon, & Zisserman. (1998). Maintaining multiple motion model hypotheses over many views to recover matching and structure. In *Proceedings of the 6th international conference on computer vision* (pp. 485–491).

Weingarten, J., Gruener, G., & Siegwart, R. (2004). *Probabilistic plane fitting in 3D and an application to robotic mapping.* in Proceedings of ICRA 2004.