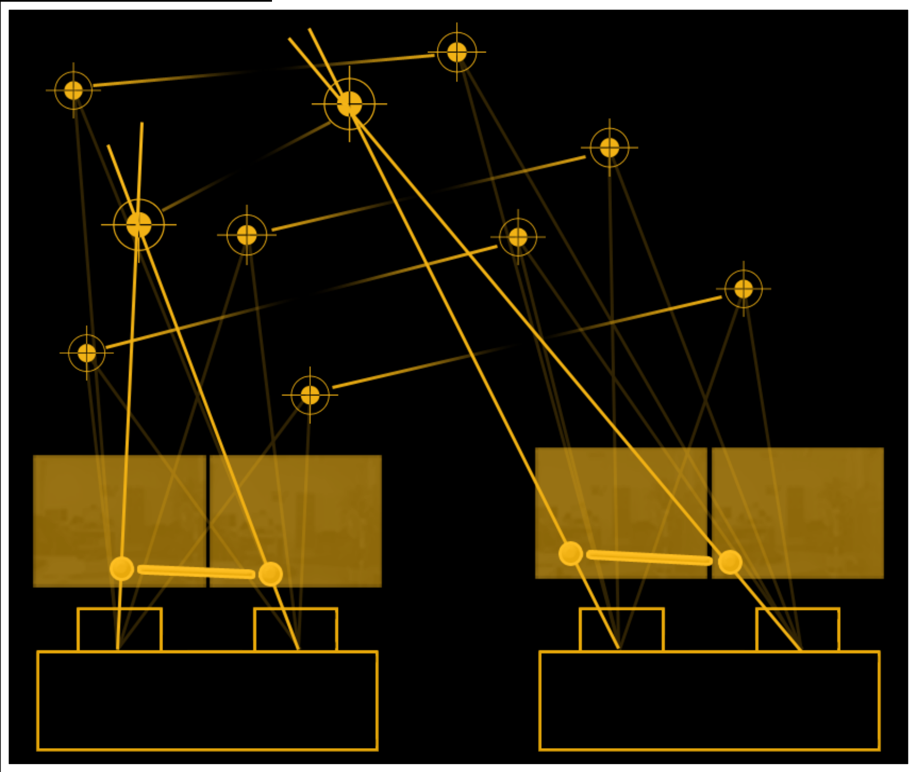
OTESC: Online Transformation Estimation between Stereo Cameras

Delft, December 2010

Version 1.0

Master's Thesis

Tiemen Schreuder



Student number:
1179462

E-mail:
tiemenschreuder@gmail.com

Thesis committee:
Prof.dr.ir. M.J.T. Reinders
Dr. E.A. Hendriks
Dr. A. Hanjalic
Dr. A. Redert

Thesis supervisors:
Dr. E.A. Hendriks
Dr. A. Redert

Information and Communication Theory Group

Delft University of Technology

Abstract

In this thesis, we describe a system that performs online transformation estimation between pre-calibrated stereo cameras. This allows the stereo cameras to be moved around and automatically re-calibrated without the use of a calibration object. This also allows the set-up to deal with ad-hoc camera relocation and to recover from accidental nudges that invalidate the extrinsic (external to the stereo camera) calibration. The obtained transformations can for example be used in virtual view rendering for 3D Video.

The relative positions and orientations of the stereo cameras are obtained using sparse point correspondences found in different views of the scene. For each stereo camera, 3D coordinates of salient scene points are triangulated and their image feature descriptors are used to locate the same points in the views of other stereo cameras. The salient point descriptors SIFT, ASIFT, SURF and FAST are compared for this purpose, together with two different error measures to see which selects the best transformation.

Two strategies are proposed and evaluated to deal with three or more stereo cameras. Given enough salient image points, the proposed solution accurately finds the transformation between stereo camera pairs with a reprojection error less than 1 pixel. Additionally we use the obtained transformations to perform virtual view rendering and evaluate the perceptual quality.

Preface

This thesis is based upon research conducted between June 2009 and December 2010 at the Department of Mediamatics, Faculty of Electrical Engineering, Mathematics, and Computer Science in the Multimedia Signal Processing group.

The purpose of this work is to give a detailed summary of my research and contributions for scientific readers. However to make the work more accessible to casual readers, such as friends and family, some basic concepts and background information are provided throughout the text. I have done my best to provide references where research of others is used.

I would like to express my sincere gratitude to my leading supervisor Emile Hendriks for lots of inspiration, comments and of course knowledge. Also his optimistic view helped me counter my skeptical attitude towards my own contributions. Finally my thanks to him for presenting part of this work in the 3D Video Processing workshop at ACM 2010, Firenze, Italy.

Thanks also to my co-supervisor André Redert, who especially helped me a lot with his technical know-how and demonstrations regarding the stereo system and various software packages. Also his ideas and comments helped a lot to improve the OTESC system and his positive attitude towards some of the results I obtained were very inspirational.

During the year and a half working on this subject I've gained more respect and interest in the scientific community and computer vision in specific. I would like to thank the members of the CVLearn group for the fun and interesting discussions on topics and papers in our field and also the various Master students that helped me with comments and ideas during the student meetings we had.

Contents

1. INTRODUCTION 5

1.1. BACKGROUND 5

1.2. PROBLEM DESCRIPTION 6

1.3. GENERAL CONCEPTS 6

1.4. RELATED WORK 11

1.5. CONTRIBUTION 12

1.6. DISPOSITION 12

2. PRELIMINARIES 13

2.1. CAMERA SET-UP 13

2.2. PRE-CALIBRATION 13

3. APPROACH 15

3.1. SALIENT IMAGE POINTS 16

3.2. STEREO-INTRINSIC MATCHING 18

3.3. STEREO-EXTRINSIC MATCHING 19

3.4. TRANSFORMATION ESTIMATION 19

3.5. MULTI CAMERA SETUP 23

4. PARAMETER CHOICES AND EVALUATION 26

4.1. EVALUATION MEASURE 26

4.2. PARAMETERS 27

4.3. COMPARISON 31

4.4. NUMBER OF RANSAC ITERATIONS 35

4.5. MULTI CAMERA RESULTS 36

4.6. SYSTEM PERFORMANCE 38

5. VIRTUAL VIEW RENDERING 43

5.1. ALGORITHM 43

5.2. IMPLEMENTATION 46

6. CONCLUSION 47

7. FUTURE WORK 48

8. REFERENCES 49

APPENDIX A IMAGE SETS 52

APPENDIX B SALIENT POINT ALGORITHMS - DETAILS 55

APPENDIX C PAPER 66

1. Introduction

In this introductory chapter the subject of this thesis is put into context. The first section explains the background and purpose of the conducted research. In section 1.2 a problem description is formalized and in 1.3 some elementary and more advanced concepts, helpful for greater understanding of this work, are introduced.

Related work is discussed in section 1.4 and subsequently the contribution of this research to the scientific community is explained in 1.5. Finally section 1.6 provides a disposition of the remainder of the work.

1.1. Background

With recent blockbusters such as *Avatar* and *Alice in Wonderland*, 3D video is gaining more and more momentum. At the same time 3D-ready television sets slowly become available on the consumer market. Most of the current 3D video material is actually stereoscopic; a different image for each eye. More advanced 3D video and 3D video equipment can provide an even stronger illusion by presenting a view based on the position of the viewer. If the viewer moves, the perspective of the 3D video changes as well. In for example the Philips 3D TV [1], this is achieved by dividing the area in front of the TV in different viewing cones, where each cone receives a different image. Other approaches actually track the physical position of the eyes [2] to adjust the viewing perspective.

One way to obtain this type of multi-view 3D video, is to place several cameras at fixed positions around the scene and use their images to interpolate the views from other positions. We refer to this as virtual view rendering. Knowing the relative position and orientation of cameras with respect to each other enables to extract the scene / geometry by relating all camera images. The scene model can subsequently be used to render new images from virtual cameras at arbitrary positions. The accuracy of the found transformations directly affects the quality of the virtual views.

Using stereo cameras instead of ordinary cameras has several advantages. Since the baseline between the two individual views of each stereo camera is known (in meters), depth triangulation is relatively easy and all scene coordinates and transformations can be determined in meters, which is not trivial when using a single camera setup.

1.2. Problem Description

The TU Delft 3D Studio consists of four stereo cameras which can be used for recording, each capturing a scene from a different angle. Prior to recording, the stereo cameras must be calibrated. Calibration consist of determining several parameters describing the set-up, such as the lens distortion of each camera and also the relative positions and orientations between each stereo camera. Current methods to perform this calibration require a user to step into the recording scene and hold up a calibration plate. Several images must be taken with the plate in different orientations each time. A software program then determines all the calibration parameters from the images. This process can easily take up in excess of 30 minutes and is required each time a stereo camera is moved even a centimeter.

The goal of this project is to investigate and develop a system which can quickly re-calibrate whenever there is any, planned or unplanned, camera movement. The user interaction should be minimal, no calibration object should be required and the scene should not be disturbed.

1.3. General Concepts

1.3.1. Image Processing

More advanced readers can safely skip this section.

A **digital image** typically consist of a rectangular grid of pixels. The size of the grid determines the **resolution** at which the (continuous) 3D scene is projected onto the 2D image plane. Each **pixel** ('picture element') stores information about the scene, usually either binary (0 or 1), grayscale or color. **Grayscale** is represented as a single intensity value. **Color** is most commonly represented as three separate intensities, one for each RGB component (red, green, blue).

A digital image can be obtained using a digital camera, which consists of a **CMOS** or **CCD** chip and a **lens**. The lens bundles the light on the chip, which consists of a grid of light-measuring sensors (photodiode). Typically one photodiode supplies the intensity value for one pixel. Color images are obtained by filtering the incident light based on wavelength, such that each photodiode measures only one color component.

In **image processing** an image is processed and the output is an (improved) image, or a set of features describing (part of) the image. A **feature** is any type of distinct information. Low-level features can be the pixel intensities themselves, while more high-level features describe structures in the image, for example edges, corners or entire objects. If a single entity (a point, an object) is described using multiple features, the combined features are called a **feature vector** or **descriptor**.

1.3.2. Coordinate Systems

A single point can be represented relative to several different coordinate systems. Calibration is required to determine the conversion from one coordinate system to another. The following coordinate systems are relevant [3]:

- *IMage Coordinate System (IMCS)*

  This coordinate system is relative to the image and the coordinates are the actual (2D) pixel coordinates. The coordinates of a point in the image are given by:

  $$\mathbf{x}^{im} = [x^{im} \quad y^{im}]^T$$

- *Camera Coordinate System (CCS)*

  This coordinate system is relative to each (individual) camera. The *x*-*y* plane is parallel to the image plane, with origin at the projection center and the *z*-axis along the optical axis. The coordinates of a 3D point in CCS is given by:

  $$\mathbf{x}^{c} = [x^{c} \quad y^{c} \quad z^{c}]^T$$

- *World Coordinate System (WCS)*

  This coordinate system is a common coordinate system for all cameras. All other coordinate systems can be defined relative to it. The CCS of one of the cameras can be chosen as the World Coordinate System, or the WCS can be chosen relative to a real-world coordinate system, for example using GPS. A point in WCS is given by:

  $$\mathbf{x}^{w} = [x^{w} \quad y^{w} \quad z^{w}]^T$$

1.3.3. Camera Calibration

Pinhole Camera Model

The Pinhole Camera Model is a model for how a 3D point is projected onto the image plane assuming the camera has no lenses and the camera aperture is a single point. The model allows a simple mapping from 3D scene to 2D image, but is only an approximation for real cameras, as effects like lens blurring

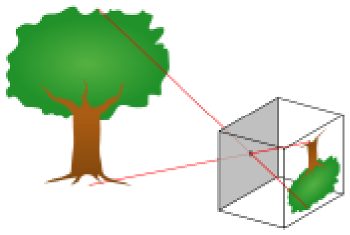


Figure 1 – Pinhole Camera Model

and distortion are not modeled. Depending on the quality of the camera and the precision required by the application, the model can be used directly as a description of the 3D to 2D (camera/world to image coordinates) mapping, otherwise the model can be combined with information about the lens distortion etc to give a meaningful conversion.

Camera Parameters

There are a large number of parameters that model the imaging process of a camera. These parameters are required to make transformations between the image coordinate system and the world coordinate system. When working with stereo cameras, some additional parameters need to be estimated. We divide the parameters in the following categories:

- **Intrinsic** (*Image to Camera Coordinates*)
    - Focal Length (f)
    - Pixel Aspect Ratio (s_x, s_y)
    - Principal Point (O_x, O_y)
- Lens Distortion Coefficients
    - Radial (k_3, k_5)
    - De-centering ($P1$,$P2$)
    - Thin Prim (s_1,s_2)
- **Extrinsic** (*Camera to World Coordinates*)
    - Rotation (α, β, γ)
    - Translation (x, y, z)

Intrinsic Parameters

Parameters internal to the camera are called intrinsic parameters, and are sufficient to model a perfect pinhole camera. Focal length (*f*) is the distance between the focal point and the image plane. The aspect ratio of the physical dimensions of a pixel (s_x/s_y) is also required, as most cameras do not have square pixels. The principal point (O_x, O_y) is the center of the projection in the image plane, ideally the centermost pixel in the image.

Lens Distortion Coefficients

Additionally several non-linear components related to lens distortion and skew can be modeled if highly accurate calibration is required. The radial distortion (k_3, k_5, sometimes as: k_1, k_2) is the most significant distortion component, and k_3 usually accounts for about 90% of the total distortion already [3]. This distortion is caused by the fact that objects at different angular distance from the lens axis undergo different magnifications. The de-centering distortion is due to the fact that the optical centers of multiple lenses are not correctly aligned with the center of the camera and the thin prim distortion arises from imperfections in lens design and manufacturing as well as the camera assembly. In this work we deal only with k_3 and k_5.

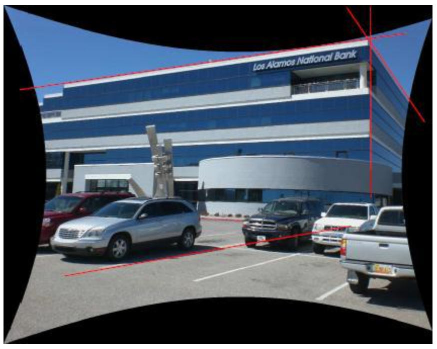


Figure 2 – Image with lens distortion effects removed.

Stereo-Intrinsic / Stereo-Extrinsic Parameters

Extrinsic parameters describe the transformation between camera and world coordinates, in other words translation and rotation. For the purpose of this thesis, we consider the extrinsic parameters *within* a stereo camera, *intrinsic* to that stereo camera. So *stereo-intrinsic* parameters consist of the intrinsic parameters (focal length, principal point, etc) of both individual cameras and the extrinsic stereo baseline (given by translation and rotation) between them. The parameters extrinsic to the entire stereo camera rig, so its position and orientation relative to the world and other stereo cameras, are referred to as *stereo-extrinsic*. Whenever a stereo camera is moved, the *stereo-extrinsic* parameters change while the *stereo-intrinsic* parameters typically do not.

1.3.4. Epipolar Geometry

When two cameras, that can be modeled using the pinhole camera model, take an image of a 3D scene from a different viewpoint, there is a geometric relationship between the location and orientation of the camera, the captured 3D points and their corresponding projections on the 2D images.

This geometric relationship allows calculating the coordinates of the 3D points given the 2D coordinates in the image plus the transformation between the cameras. This can be used to perform depth triangulation for point correspondences between two cameras, if their extrinsic calibration is known.

Alternatively, the geometric relationship also allows calculating the transformation between the cameras given the 3D coordinates and 2D coordinates. This property can be used to perform calibration between stereo camera pairs using a known-geometry object.

Epipolar line

Another useful property in epipolar geometry is called the epipolar line. For a given point in the image, its corresponding point in another view must lie along an epipolar line (see Figure 3). This property increases the chance of finding the correct correspondence and reduces the computational cost, as less points have to be considered. Finding a corresponding point is important in both triangulation (see next section) and depth selection in virtual view rendering.

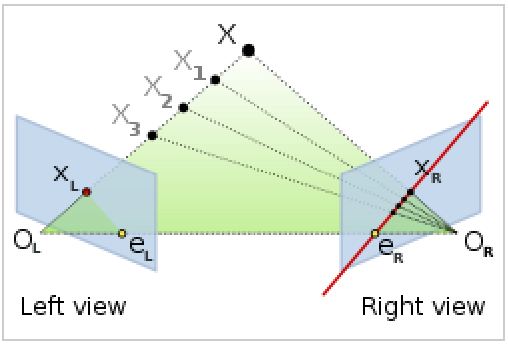


Figure 3 – Epipolar geometry and lines. For a point X_L in the left view, any correspondence in the right view must lie on the red line, which is the projection of the line from O_L through X_L.

Triangulation

Triangulation can be used to calculate the depth of a point found in two (or more) views, assuming the epipolar geometry between the two cameras is known. If the same point $\mathbf{x}^w$ in WCS is seen from two cameras, and this correspondence is found in terms of pixel coordinates, the depth of the point can be calculated. In the case of a stereo camera, the transformation between the two cameras is ideally a simple translation in one direction (the baseline distance). A ray from the focal point of one camera through the projection of $\mathbf{x}^w$ in the image plane (reverse to the rays of light) will intersect with a similar ray from the second camera at the depth at which the point $\mathbf{x}^w$ is located, as can be seen in the figure below. Together with the baseline, these two rays form a triangle, hence the name triangulation. In set-up used in this work, the extrinsic parameters are known in meters, and thus the depth can be triangulated in meters as well.

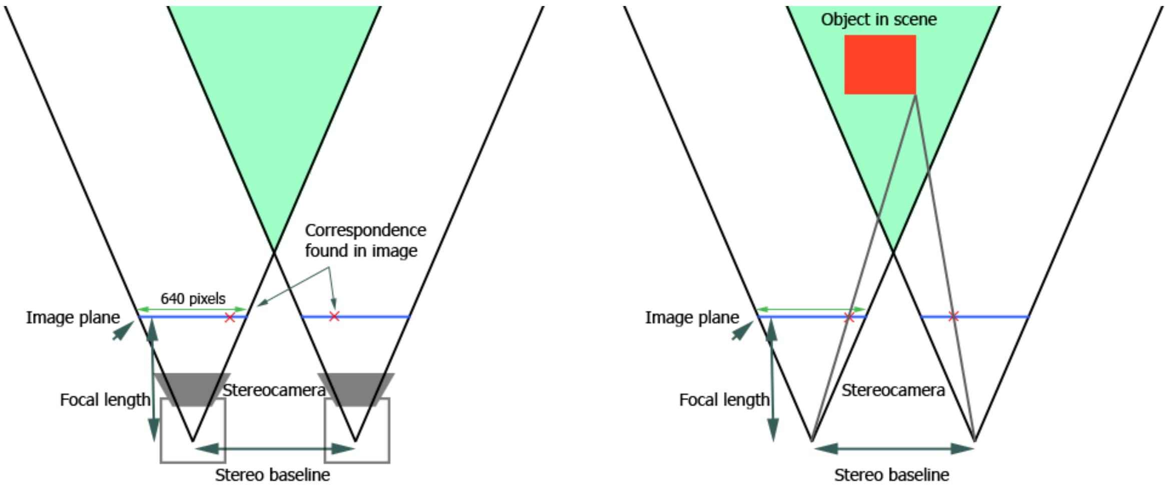


Figure 4 (Left) An image correspondence between (red X) the left and right view of a stereo camera is found. (Right) For each individual camera a ray is traced from the camera origin through the image point. For a valid correspondence, the two rays will (nearly) intersect and the 3D coordinate can be derived.

1.4. Related Work

There are two main approaches towards camera calibration. In the first approach a calibration object with known geometry, for example a plate with a checkerboard pattern, is captured by all cameras and camera parameters are computed such that they are consistent with both the known geometry and the image projections of the object. This typically results in a highly accurate calibration [4-6]. A disadvantage of this approach is the required manual interaction (placing a calibration object in the scene, usually in different orientations), which also means the normal recording, if any, must be interrupted.

The second approach aims to overcome these problems by performing calibration using only the scene information available in the images taken by the cameras. This approach is commonly referred to as self-calibration, or online calibration. For many applications this self-calibration is implemented as a Structure from Motion (SfM) approach [7], which simultaneously finds the 3D scene structure and all camera parameters by analyzing the motion of objects across views and/or over time. Many SfM approaches use (sparse) bundle adjustment [8], which takes image point correspondences as input and refines the total reprojection error, using the Levenberg-Marquardt algorithm. Any wrong point correspondences should be removed beforehand during preprocessing.

Often it is more efficient to perform only a partial calibration in systems that require periodic re-calibration. For example, in [9] the intrinsic parameters are supposed to be known a priori and in [10,11], specifically aimed at stereo cameras in distributed 3D visual sensor network, the stereo-intrinsic parameters are assumed known and fixed, leaving only the stereo-extrinsic parameters to be estimated. In [11] the focus lies mostly on the network protocols and although their approach is based on self-calibration, they require the use of a calibration target (object producing many unique image points) to deal with arbitrary scenes.

1.5. Contribution

The contribution of this work lies in the comparison and combination of existing techniques with the purpose of creating a self-calibration system for the TU Delft 3D studio. Specifically four different salient image point algorithms have been compared and two different error criteria have been looked at. Note that although the approach in this work is not based on bundle adjustment, many of the preprocessing steps and comparisons are also relevant for bundle adjustment (SfM) approaches, as bundle adjustment is just a final refinement step which can be used with different salient image point algorithms and outlier strategies.

Additionally an existing virtual view rendering algorithm has been implemented on graphical processing hardware, resulting in a significant speed increase.

1.6. Disposition

In chapter 2 some background information on the setup and required preliminary steps are discussed. In chapter 3 the details of the system are explained and the various algorithms are introduced. Chapter 4 gives the choice of various parameters, evaluation of algorithms and the overall performance of the system. Chapter 5 discusses a different subject, namely a Virtual View Rendering algorithm and its implementation in OpenCL, which was used as an evaluation method. Finally in chapter 6 a brief conclusion is drawn while in chapter 7 recommendations for future work are given.

2. Preliminaries

This chapter explains the specific camera set-up used in this work and also the stereo-intrinsic pre-calibration step required.

2.1. Camera Set-Up

Our 3D Studio consists of stereo cameras (see Figure 5) each consisting of two FireWire *AlliedVisionTec Marlin F-046C* [12] cameras bolted on a rigid aluminum frame, which in turn is mounted on an ordinary camera tripod. The cameras are connected with a PC through a FireWire hub mounted on top of the cameras and provide 640x480 images. The stereo baseline is adjustable, but was fixed at 10.4cm for all experiments.

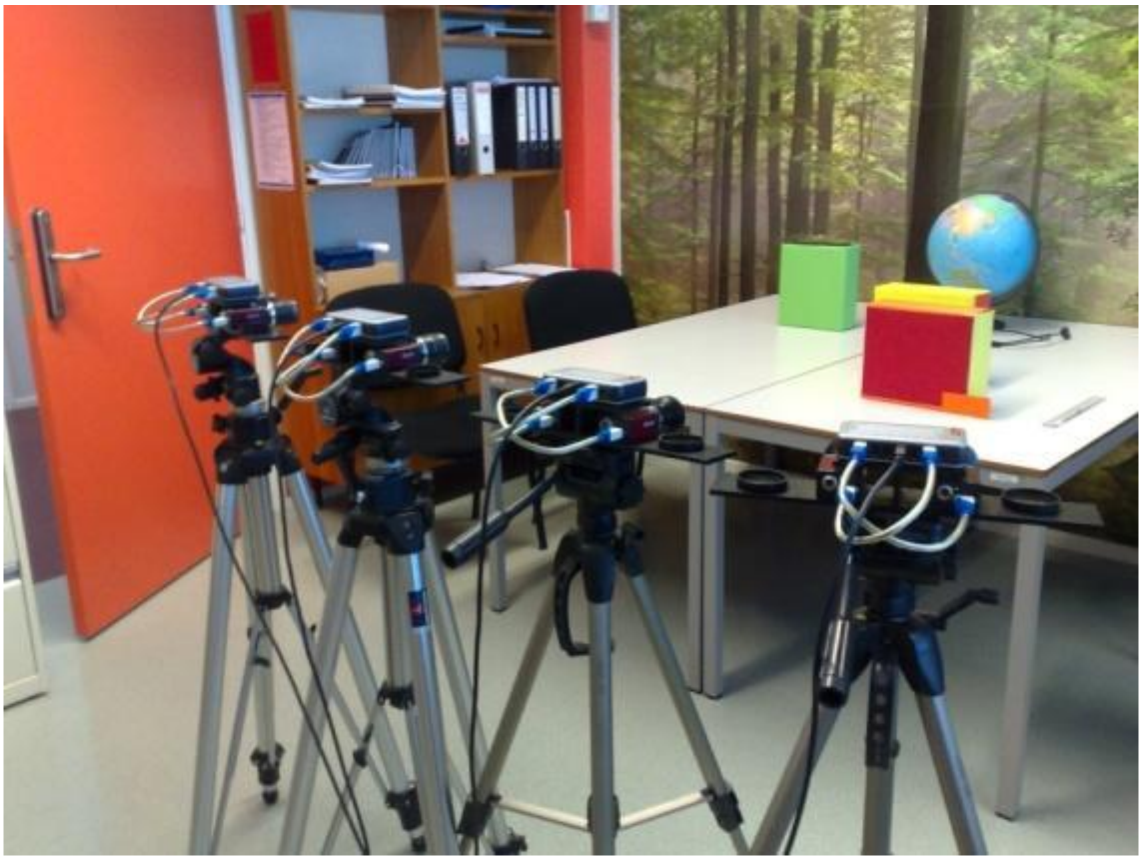


Figure 5 – Ordinary stereo camera setup with four stereo cameras positioned around a scene.

2.2. Pre-Calibration

The proposed system works with pre-calibrated stereo cameras. We make the assumption that moving stereo cameras does not change the stereo-intrinsic parameters, given that the cameras are mounted tightly on a rigid frame. Therefore a stereo camera only has to be calibrated once. The obtained calibration data should remain valid as long as the stereo camera rig is not modified (such as changing the focal length or stereo baseline). In practice the stereo-intrinsic calibration may also become invalid due to external factors such as temperature changes, so we recalibrate periodically.

All pre-calibration data for experiments in this paper was obtained with a manual calibration technique [6] using a known-geometry calibration plate (see for example Figure 6).

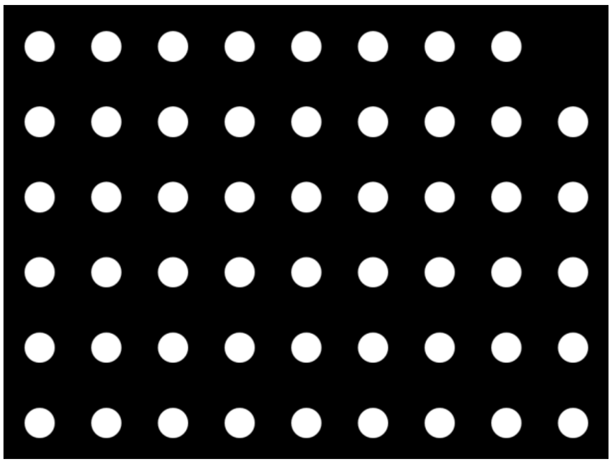


Figure 6 – Calibration plate dubbed 'Grid-'

The calibration is performed by placing the calibration plate in the scene such that two or more cameras have a good view of the plate. After taking an image (with all cameras), an algorithm then locates the dots on the plate in each view. For views where all dots are found, the correspondences can be used to calibrate those cameras relative to each other. For views where the dots were not correctly located, additional images are required. To make the calibration more accurate, the process is repeated several times with the calibration plate each time in different orientations.

3. Approach

In this chapter the details of the proposed approach are explained. The goal is to determine the transformations that describe the relative positions and orientations of the stereo cameras capturing a single scene. To perform this task, the literature was studied for applicable techniques and existing algorithms (see also previous work [13]). Using this information a new approach was formulated, which can be summarized as follows (see Figure 7):

- Detect salient image points in both views of all stereo cameras
- For each stereo camera, establish image point correspondences between the stereo views and triangulate the 3D coordinates of each correspondence
- Match 3D points between a pair of stereo cameras using the original image point descriptors
- For a selected subset of stereo camera pairs, estimate transformation from RANSAC-selected (3D) correspondences
- Resolve transformation between any stereo camera pair desired

Each of these steps will be discussed in detail in subsequent sections and for some steps several alternative algorithms are explored and compared as part of this work.

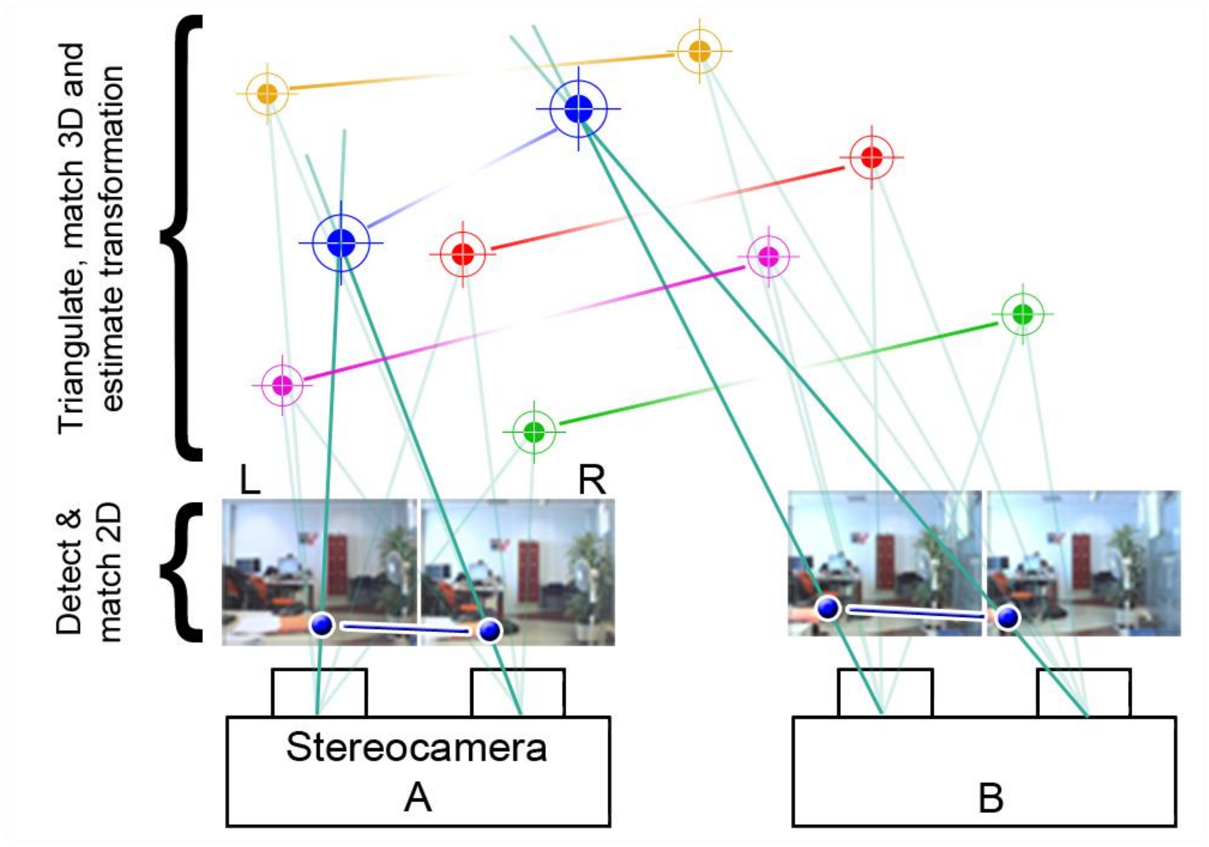


Figure 7 – Overview of the OTESC algorithm

3.1. Salient Image Points

Each stereo camera simultaneously takes an image of the scene from each view, resulting in two images per stereo camera. The first step in OTESC is to find salient image points (sometimes referred to as critical points, or keypoints) that can be uniquely matched in these two images. Several algorithms exist for this purpose and most of them aim to fulfill three requirements: detection, description and matching procedure. In the detection step salient image points are identified (eg, corners, edge points, high contrast regions) and in the description step the detected salient points are described as feature vectors. Most algorithms then define a procedure describing how the salient points are to be matched using the feature vectors. OTESC acts as a framework and allows different salient image point algorithms to be used. Several promising algorithms were tested, what follows is a brief explanation of each.

3.1.1. Salient Point Algorithms

Four salient point algorithms are briefly introduced here. These algorithms have been pre-selected for their desirable properties in terms of speed or accuracy. For a more detailed description see Appendix B or the original papers.

SIFT

SIFT [14,15] stands for Scale-Invariant Feature Transform and is an algorithm aimed at performing object recognition, by matching many local image features instead of using a few complex (for example geometric) features.

SIFT selects and describes a salient point in the image if it is a local extrema in a Difference of Gaussians function on a small neighborhood around this pixel. The resulting description is invariant to translation, rotation and scale.

SURF

A relatively new interest point detector and descriptor is SURF, Speeded Up Robust Features [16], which is partly inspired by SIFT. This algorithm is relatively quick compared to SIFT, about 3 times faster, and the authors claim it has similar or slightly better performance. In another comparison [17] it is also concluded that SURF outperforms SIFT when it comes to viewpoint changes, which is of course an important criterion. SURF does find fewer matches, and is slightly less invariant to illumination changes.

The normal version of SURF is invariant to translation, scaling and rotation. A version called U-SURF (upright SURF) is only invariant to rotations up to 15°, but is faster than regular SURF. Since in most practical situations the view rotation is typically below 15°, it may be sufficient to use U-SURF for camera calibration.

FAST

FAST (Features from Accelerated Segment Test) is a corner detector [18,19] designed for use in real-time tracking systems. The detector performs a very simple test for each pixel *p*, by examining a circle of 16 pixels around the pixel. A corner is detected if at least *n* contiguous pixels are all above or all below the intensity of *p* by some threshold *t*. The specific variant (FAST-9, FAST-12) determines the value of *n* (eg 9, 12, etc). The descriptor used is typically the 16 pixel values, which can be matched after calculating the SSD (Sum of Squared Distances).

ASIFT

Methods like SIFT and SURF normalize the translation and rotation component and simulate the scale (zoom) through image pyramids to obtain an description invariant to these parameters and partially invariant to affine transformations. A recently proposed method ASIFT (Affine SIFT) [20,21] attempts to obtain a description fully invariant to affine transformations. The method simulates all image views obtainable by varying the latitude and longitude camera angles. Next the resulting views are compared using the normal SIFT algorithm.

3.1.2. Selection

Four main techniques have been introduced as salient image point detectors. Based on claims by the authors, FAST-9 shows most promise in terms of speed. In terms of robustness ASIFT is likely to perform best. A big disadvantage of FAST-9 is that is does not provide subpixel accuracy and is not scale or rotation invariant. SIFT is commonly used and SURF is partly inspired by SIFT, aimed at providing similar quality features in less time. Both descriptors are insensitive to scale and rotation (optional for SURF). Both are invariant to small affine transformations. ASIFT should provide better results than SIFT under large viewpoint changes, however it comes at an highly increased computational cost.

The implementations used are [22] for SIFT, the OpenSURF library [23] for SURF. For FAST-9 the implementation by the original authors is used. For ASIFT the original Linux source by the authors was ported to Win32. All source code has been modified slightly to integrate with OTESC. The parameters were set to the author's recommended settings, except as otherwise noted. For example for SURF the Upright-SURF variant was used. Upright-SURF is quicker, but only orientation invariant up to about 15 degree [16], which should be sufficient for our purpose. In section 4.3.2 a comparison between the algorithms, in terms of accuracy of the proposed solution, is given.

3.2. Stereo-Intrinsic Matching

For each stereo camera, salient point detection and description is applied on both views (L and R) and sparse point correspondences are established between the two views.

The four salient point algorithms can be used with a similar matching scheme; matching happens not based on the distance between two feature vectors, but on the ratio between the distance with the best and second-best match. If an image contains repeating patterns for example, there are many valid candidates and the matching would be ambiguous. The ratio ensures that the correspondence is only accepted if it is unique among all candidates. Formally the ratio is defined as follows: if the distance between two point descriptors (i, i') is given by $d(i, i')$, a match between i and i' is only accepted if i' is the best match (smallest distance), and the ratio with the second best match $i"$ is below a threshold z:

$$\frac{d(i, i')}{d(i, i")} < z \tag{1}$$

If an image contains repeating patterns for example, there are many valid candidates and the matching would be ambiguous. The ratio ensures that the correspondence i' is only accepted if it is sufficiently unique among all candidates.

To find matches, all descriptors in one image must be compared to all descriptors in the other image. As the complexity of this matching procedure is O(n^2), the matching speed is typically improved by using a modified k-d tree algorithm to find nearest neighbors more efficiently [14]. Note that in the current implementation this optimization is not used for all salient point algorithms.

Ideally, the corresponding image points (i, i') found $(i \in L, i' \in R)$ have originated from a single 3D scene point x. Since the stereo-intrinsic calibration parameters are known, it is possible to triangulate the 3D coordinates of x relative to the stereo camera. For both individual cameras, a (theoretical) ray is traced from the optic center through the image plane (at the image point coordinates), see also section 1.3.4. The rays should converge near scene point x. If the distance between the two rays at the point of closest convergence is above a threshold g, the correspondence (i, i') is discarded, as this would indicate that the coordinates of the two salient points were inaccurate, or did not originate from the same point x.

We associate point x with the salient point descriptors of both image points for further matching with other stereo cameras. After this stereo-intrinsic matching each stereo camera has a list of scene points with each two associated image descriptors (L, R).

3.2.1. Epipolar Constraints

As noted in section 1.3.4 on epipolar geometry, when looking for a corresponding point in another view, the search area can be limited to a line. This constraint can be used in the matching phase to quickly eliminate wrong matches: for a point i in L, calculate its epipolar line l in view R. If the current point i' under consideration is not on line l, the feature vectors do not have to be compared as this cannot be a valid match. Note that in OTESC this explicit check is not used, but it is implicitly performed during the triangulation step.

3.3. Stereo-Extrinsic Matching

For a *pair* of stereo cameras (A, B) the transformation between them is obtained by first establishing correspondences between the scene points.

The matching, using the same matching algorithm as described in the previous section, is performed first using the left descriptor of each point, as follows: match(L_A, L_B). If less than 20 matches are found in this matching step, all four combinations of the two associated descriptors for each point: (match(L_A, L_B), match(L_A, R_B), etc) are done. This is repeated inversed because the matching is not symmetrical (eg, match(x, y) $\neq$ match(y, x)). So points from B are matched with A, resulting in a total of eight matching runs. These eight runs typically result in 2 to 3 times more matches (after removing duplicates) than a single run.

Scene points that have been matched are ideally the same world point seen from two different viewpoints. As the matching process is based on photometric consistencies, it will likely produce physically incorrect matches. Therefore the transformation estimation must be robust to outliers; see the next section.

3.4. Transformation Estimation

The (affine) transformation T between the two matching point clouds is calculated using the Absolute Orientation algorithm [24], see 3.4.1 for details. This algorithm requires a minimum of three 3D correspondences (however we assume a minimum of four in our implementation) and estimates the rotation and translation required to align the clouds, represented as:

$$T = \begin{bmatrix} & R & & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where R is a 3x3 rotation matrix and t is a translation vector. The transformation T is estimated such that for each correspondence (x, x'), ideally:

$$x' - Tx = 0 \tag{3}$$

with x and x' in homogeneous coordinates.

To cope with noise and incorrect matches, a RANSAC [25] scheme is employed, see 3.4.3 for a background on RANSAC. During each RANSAC iteration a random subset of four correspondences is selected and the transformation T is estimated. Remaining correspondences (x, x') are considered in-consensus, and added to the consensus set C, if their correspondence is adequately modeled by the found transformation. This is formalized in section 3.4.2. If C consists of at least 40% of all correspondences, or at least 20, the error on all inliers of the found transformation is evaluated (see section 3.4.4). The entire process is repeated a fixed number of iterations (see 4.4), and after each iteration the best transformation is kept. It is possible that no transformation is found at all, because not enough correspondences are found or in-consensus.

3.4.1. Absolute Orientation

Absolute Orientation is an algorithm which finds the relationship between two coordinate systems by using corresponding coordinates from both systems [24]. We use a closed-form solution to the least-squares problem obtained using unit quaternions, which requires at least 3 correspondences. Alternative approaches to the Absolute Orientation problem use orthonormal matrices [26], SVD or dual quaternions. As described in [27] all approaches perform similar in terms of accuracy. The approach using unit quaternions is used because it is considered most elegant.

Given two sets (l, r), each containing n coordinates ($\mathbf{r}_{\mathbf{0},l}, \mathbf{r}_{\mathbf{1},l}, \dots, \mathbf{r}_{\boldsymbol{n-1},l}$ and $\mathbf{r}_{\mathbf{0},r}, \mathbf{r}_{\mathbf{1},r}, \dots, \mathbf{r}_{\boldsymbol{n-1},r}$), the approach works by first calculating the centroid $\bar{\mathbf{r}}$ (average coordinate) of both the left and right coordinate set. The centroid is subsequently used as the new origin of each set; all coordinates are redefined (same for r):

$$\bar{\mathbf{r}}_l = \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{r}_{i,l} \qquad \mathbf{r}'_{i,l} = \mathbf{r}_{i,l} - \bar{\mathbf{r}}_l \tag{4}$$

Next the difference in scale can be obtained from the resulting coordinate sets, however we assume equal scale (scale=1). This assumption is valid from a physical point of view; all 3D coordinates were obtained in meters, and are thus in the same scale (unit). This also means we now know the translation, as this is simply the difference between the left and right centroid.

The main computational step is determining the rotation between the two point clouds. A matrix N is constructed:

$$N = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}-S_{yx} & S_{zx}-S_{xz} \\ S_{zx}-S_{xz} & S_{xy}-S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}-S_{zy} \\ S_{xy}-S_{yx} & S_{zx}-S_{xz} & S_{yz}-S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{bmatrix} \tag{5}$$

with all variants S_{xx}, S_{xy}, … ,S_{zz} calculated (note: $\mathbf{r} = [x\ y\ z]^T$) as follows:

$$S_{xx} = \sum_{i=0}^{n-1} x'_{i,l}\, x'^T_{i,r} \qquad S_{xy} = \sum_{i=0}^{n-1} x'_{i,l}\, y'^T_{i,r}\ , \qquad etc \tag{6}$$

The eigenvector of N corresponding to its most positive eigenvalue is a unit quaternion representing the required rotation. This quaternion is converted to a 3x3 rotation matrix and combined with the previously obtained translation vector to form the transformation matrix.

3.4.2. Consensus Criterion

To determine if a correspondence is correctly modeled by the transformation T under consideration, an error measure is calculated. Two different measures were evaluated; the *3D Error* and the *Reprojection Error*:

- **3D Error**

  The *3D Error* is given by the (3D) Euclidean distance between x' and its estimate Tx, thus:

$$3DE = |x' - Tx| \tag{7}$$

- **Reprojection Error**

  The *Reprojection Error* is given by the (2D) Euclidean pixel distance between the projections of x' and estimate Tx on the image plane:

$$RE = |proj(x') - proj(Tx)| \tag{8}$$

where $proj()$ projects the 3D point onto the right-most image plane of the stereo camera associated with x'. Since the stereo-intrinsic calibration parameters are known, this projection can be directly calculated.

To use these error measures in the RANSAC scheme, a threshold is applied of respectively q pixels for the *Reprojection Error* and r meters for the *3D Error*. Correspondences with an error below this threshold are regarded as in-consensus with the current transformation. See section 4.3.1 for a performance comparison between the two error measures, and the chosen thresholds.

3.4.3. RANSAC

RANSAC (RANdom SAmple Consensus) is a generic method to find a model in observed data which is corrupted by a large amount of outliers, first described in [25]. The (general) strategy employed by RANSAC works as follows. For each iteration a random subset (of size n) is chosen from the dataset. This subset is regarded as hypothetical inliers. The model is fitted on this subset, using a problem-specific fitting method, for example least squares. The data not in the subset is then evaluated and added to the hypothetical inliers if they are consistent with the model (threshold t), forming the consensus set. If the total number of hypothetical inliers is large enough (threshold d), the model is considered good. The model is improved by refitting the model on the consensus set. The model with most points in the consensus set is stored as the best model. After a fixed number of iterations (k) the best model is returned.

Variants

There are many variations on RANSAC attempting to improve some of its performance characteristics [28,29]. For example MSAC [30] uses a different criterion to determine which consensus set is best. The original RANSAC paper used a criterion aimed at minimizing the cost, namely the number of outliers; so inherently the more inliers the better the model. The actual fitness of the inliers was not taken into account. In MSAC the outliers are still regarded as a constant cost (the threshold value t), but the cost of the inliers is their error compared to the model (between 0 and t). This very simple change makes the evaluation of models much more robust [30].

Variant Used

In this specific case, the model to be found is the transformation that aligns the two point clouds. Outliers here are incorrect correspondences; 3D coordinates which do not actually describe the same scene point. The method is iterative and non-deterministic, meaning it is not guaranteed to produce correct results due to its random nature. RANSAC does not find the model itself, it merely provides a strategy to select the best subset of data, only inliers and not the outliers, to estimate the model from. Estimating the model from the subset is done by the Absolute Orientation approach described in the previous section. Although the Absolute Orientation algorithm can deal with the small alignment errors of correct correspondences, the (much larger) errors introduced by outliers would corrupt the estimation of the model.

Our RANSAC strategy uses a cost function similar to MSAC, except that the cost of outliers is not counted at all. The number of outliers is irrelevant for our purpose as long as there are enough inliers, so there is just a threshold on the number of inliers required. Also contrary to the original RANSAC algorithm, the model is not re-estimated from the entire consensus set C when using the *Reprojection Error*. Instead the transformation found from the original four points is kept, as this gives better results. An explanation could be that the *Absolute Orientation* algorithm, due to its nature, minimizes the *3D Error*, not the *Reprojection Error* used as evaluation. When using the *3D Error*, re-estimation does lead to a smaller average error, which is expected given the design of the algorithm.

3.4.4. Transformation Error

The total transformation error is given by averaging the consensus error (either 3D or Reprojection Error) over all correspondences in the consensus set C. This error gives an indication of the average error the transformation makes on correct correspondences. We use this error as the fitness measure for RANSAC.

3.5. Multi Camera Setup

The procedures described above determine, between a pair of stereo cameras (s, s'), the transformation $T(s, s')$. We will now extend this procedure for set-ups with more than two stereo cameras. Let S be the set of all stereo cameras, so $s, s' \in S$. Let $C(s, s')$ be the number of correspondences found between s and s'. The goal is to find the position and orientation of each s relative to a common origin O (one of the stereo cameras, $O \in S$). We will differentiate between direct transformation estimation (DTE), which is the technique described in the previous sections, and indirect transformation estimation (ITE), which is described in this section.

A reasonably safe assumption is that if two views have more overlap, more correspondences can be found and our approach is better at finding an accurate transformation estimation. As perhaps not all pairs (s, s') have overlapping views, and some pairs have more overlap than others, there are different strategies possible in finding all transformations. Since no knowledge is yet available about the relative positions and orientations of the stereo cameras, the number of matches $C(s, s')$ is used as an indicator of view overlap instead.

We will discuss two simple strategies to deal with multi camera set-ups (see also Figure 8). In section 4.5 we will give a numerical comparison between the two.

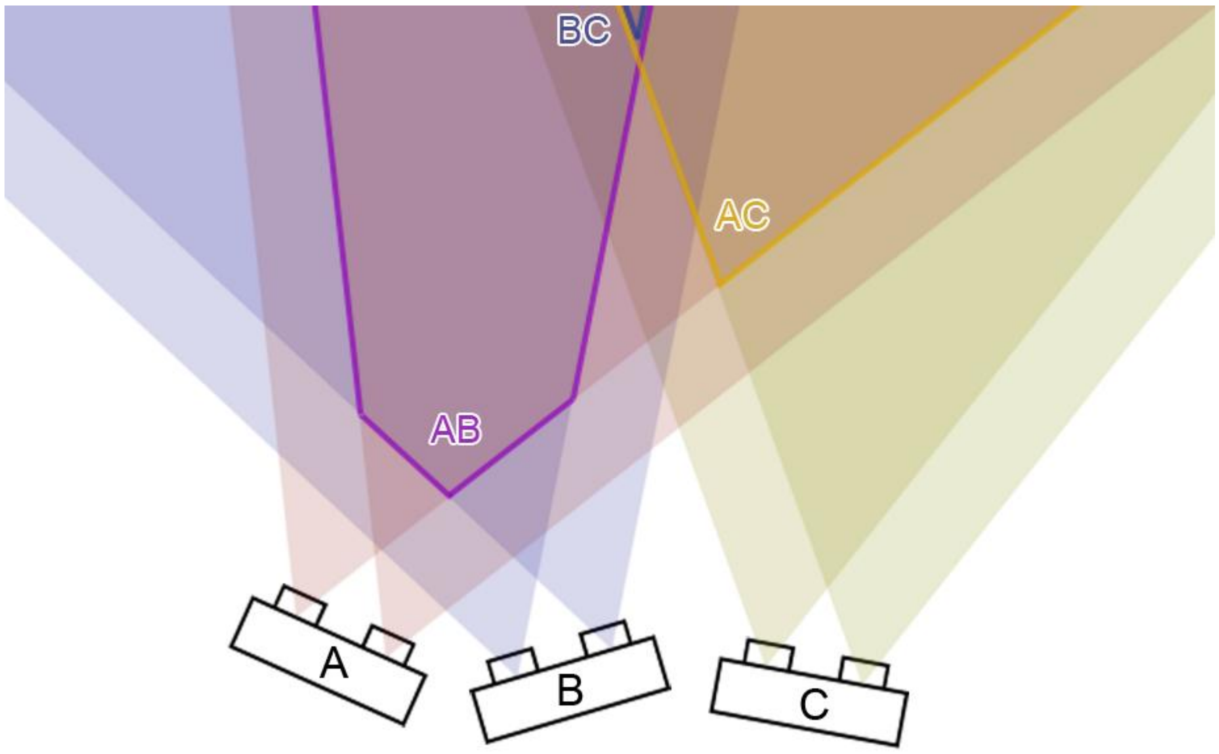


Figure 8 – An example multi (stereo) camera setup with 3 stereo cameras (A, B, C) with their respective area of overlap (eg. AB is the area which both A and B see). We assume in this image that the number of correspondences found is directly correlated to the view overlap. The origin is chosen to be B. In the Minimal Set approach, the transformation [B,C] is not estimated directly, but calculated from the transformations [A,B] and [A,C]. In the Origin approach [B,A] and [B,C] are calculated directly.

3.5.1. Minimal Set

Given three stereo cameras {1,2,3}, if the transformation between {1;2} and {2;3} is already known, the transformation between {1;3} can be deduced indirectly. We have implemented this approach as follows. All possible pairs (s, s') are evaluated in descending order of number of correspondences. A graph G is constructed with each stereo camera s as a vertex $V(s)$. Whenever the transformation between a stereo camera pair (s, s') has been estimated, the edge $(V(s), V(s'))$

is added to G. Before performing DTE between each (s, s'), a breath-first search is performed on G to see if there is a path that connects $V(s)$ and $V(s')$. If such a path exists the DTE is skipped; the transformation can already be indirectly determined. The final step is to calculate the missing (indirect) transformations between the origin O and each remaining stereo camera s ($s \neq O$) by multiplying the previously found transformation matrices along the (shortest) path between $V(O)$ and $V(s)$ (ITE).

3.5.2. Origin

Instead of finding the minimal set of most 'overlapping' pairs, this approach performs a DTE between the chosen O and any stereo camera s ($s \neq O$), regardless of the number of correspondences between them. If DTE fails for an s, an ITE is performed through a stereo camera s' for which $T(O, s')$ is already known. The stereo camera s' is selected by:

$$\arg\max_{s'}(C(s, s')) \tag{9}$$

In this approach the common origin O, if not given, is chosen by finding the stereo camera s from S, for which holds:

$$\arg\max_{s}(\min_{s',\ s' \neq s}(C(s, s'))) \tag{10}$$

In words, the minimum number of correspondences a stereo camera shares with any other stereo camera should be the highest of all stereo cameras. This should select the stereo camera which is most 'central' and has enough correspondences with any other stereo camera for a successful DTE. If multiple stereo cameras meet this criterion, the stereo camera with most correspondences overall is chosen.

4. Parameter Choices and Evaluation

4.1. Evaluation Measure

The stereo cameras were pre-calibrated using the technique described in section 2.2. The pre-calibration also provides a ground-truth transformation (T_{GT}) between the stereo cameras, which is used to evaluate our approach. The main evaluation criterion is calculated as follows; for each (x, x') in the inlier set C selected by the RANSAC scheme, calculate the average difference:

$$\epsilon_C = \frac{1}{|C|} \sum_{(x,x') \in C}^{|C|} |proj(T_{GT} * x) - proj(T_{OTESC} * x)| \tag{11}$$

where T_{OTESC} is the transformation found by our approach and $proj()$ is the pixel projection described in section 3.4.1. Note that the point x' is not required for this error measure. We will refer to this error measure as the ground truth (reprojection) error, given in pixels.

As will be shown in the final results, the relative position and orientation sometimes deviate significantly from the ground-truth transformation. It should be noted that both the OTESC transformation and the ground-truth transformation are not necessarily physically correct; they are geared towards minimizing a reprojection error. Therefore the deviation in translation and rotation is purely given for illustration purposes. If a transformation T is represented as a translation component t and the rotation R in Euler angles (α, β, γ), then the translation error is given by the Euclidean Distance:

$$|t_{GT} - t_{OTESC}| \tag{12}$$

and the rotation error is given by:

$$|\alpha_{GT} - \alpha_{OTESC}| + |\beta_{GT} - \beta_{OTESC}| + |\gamma_{GT} - \gamma_{OTESC}| \tag{13}$$

4.1.1. Normalized Ground Truth Reprojection Error

When combining the ground truth error from multiple image sets, for example when choosing optimal parameter values, we want to normalize the error measures to prevent an image set with a larger error to dominate the combined measure. Also due to the non-deterministic nature of RANSAC we must take into account the variance of the error. Therefore we introduce the following normalized measure:

$$\check{\epsilon}_{x_{ij}} = \frac{\mu_{x_{ij}} + 2\sigma_{x_{ij}}}{\max\limits_{i=0..n}(\mu_{x_{ij}} + 2\sigma_{x_{ij}})} \tag{14}$$

where x_{ij} is a distribution of ground truth errors (multiple ϵ_C) for a given image set j and set of parameters and n is the number of different parameter values checked. Note that the value $\mu_{x_i} + 2\sigma_{x_i}$ is the expected upper bound for the error in 97.8% of the cases (assuming normal distribution). The average and standard deviation were obtained over 20 to 50 samples (depending on computational costs). The normalization scales all error measures $\check{\epsilon}$ in the range [0..1], which allows them to be averaged over different image sets to analyze which parameter value gives the lowest average error. This average error per parameter set may give a unrealistic view if only a small number of images contribute to it and for the other images no solution (thus no error) was found at all. To take this into account, a weighted average is calculated as follows:

$$\overline{\overline{\check{\epsilon}_{x_\imath}}} = \frac{\overline{\check{\epsilon}_{x_\imath}}}{k} \tag{15}$$

where k is the number of images that contributed to the normalized error.

4.2. Parameters

Four salient point algorithms (SIFT, SURF, ASIFT, FAST) are compared based on their performance in the OTESC system. To make the comparison as fair as possible, several OTESC parameters have to be adjusted specifically for each algorithm. Note that the parameters internal to the algorithm, unless mentioned below, are kept at the authors recommended values.

For each of the salient point algorithms the various parameters have to be chosen. The following parameters are tuned:

- Stereo Triangulation Threshold
- Stereo Matching Ratio
- Multi Matching Ratio
- RANSAC error measure threshold
    - Reprojection Error Threshold
    - 3D Error Threshold

Since any dependencies between the parameters are mostly unknown, ideally the resulting error on all combinations of parameter values are calculated. Unfortunately this is computationally infeasible, so instead some assumptions on independency are made. Also other parameters, such as the number of RANSAC iterations, are kept fixed and are tweaked separately since they are (assumed) independent on the actual algorithm used (although depend on the number of matches found). For all parameters there is a positive correlation between its value and the resulting final error, so lower parameter values result in lower errors. However if the parameter value is set too low, the chance of finding no solution at all greatly increases.

Stereo Triangulation Threshold

This parameter is used as a threshold when triangulating a stereo correspondence into a 3D coordinate. If the distance between the rays exceed this threshold at the point of their closest convergence, the correspondence is discarded.

Stereo Matching Ratio

The stereo matching ratio is a parameter found in the matching scheme of all four salient point algorithms. In OTESC it is used within the stereo camera. The ratio describes the acceptance threshold when the first and second best match are close to each other. A higher ratio means more matches are accepted. Although the respective authors of the salient point algorithms provide a recommended setting for this matching ratio, it was tuned specifically for this system.

Multi Matching Ratio

In the second matching step, between stereo cameras, a different ratio is used. Again, a higher ratio means more matches are accepted. See Figure 9 for a comparison different matching ratios (both stereo and multi) for FAST.

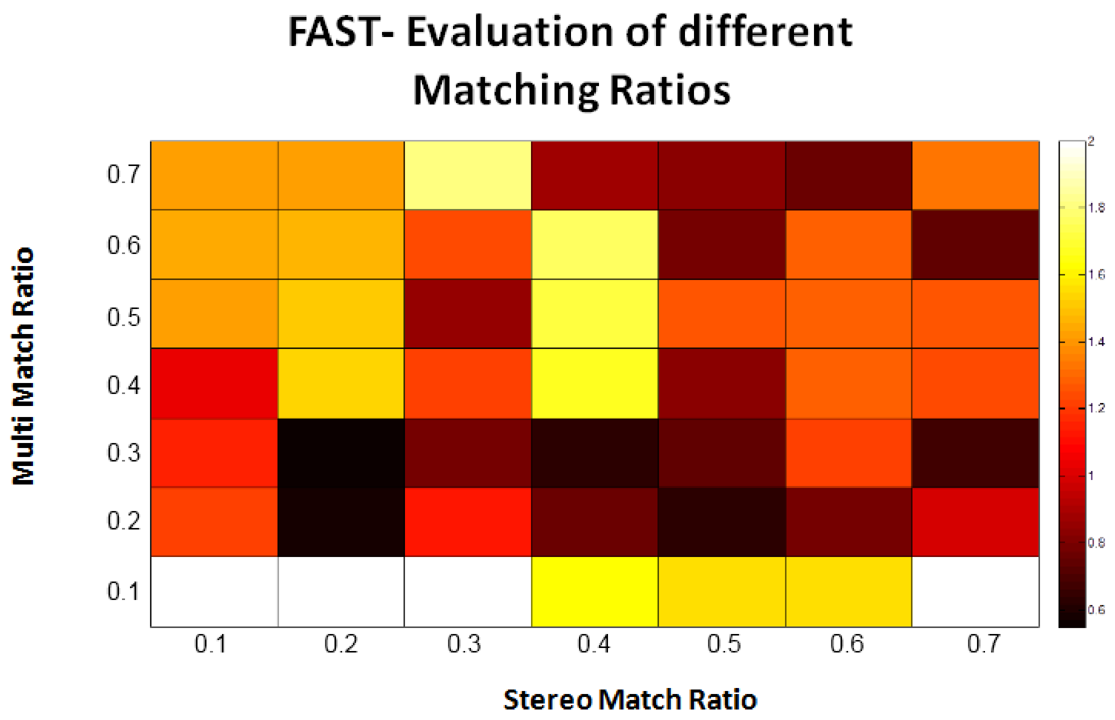


Figure 9 – Multivariate analysis of the Stereo and Multi Match Ratios. The color (see legend) indicates the average normalized reprojection error. Images for which no solution was found were given a fixed (normalized) reprojection error of 2.0.

RANSAC Reprojection Error Threshold

If as RANSAC consensus criterion the Reprojection Error is used, a specific threshold in pixels is required. See Figure 10 for the evaluation for SIFT.

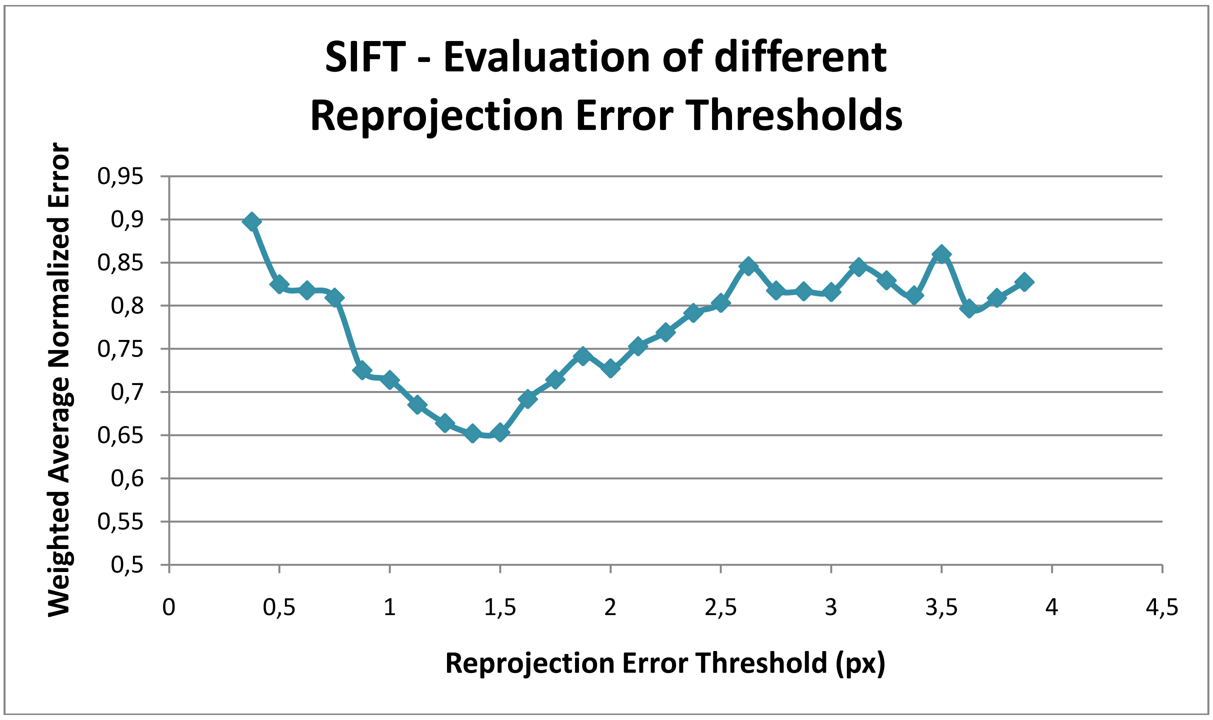


Figure 10

RANSAC 3D Error Threshold

If the 3D Error is used as the RANSAC consensus criterion a threshold in meters must be used. See Figure 11 for the evaluation of different threshold values for SURF.

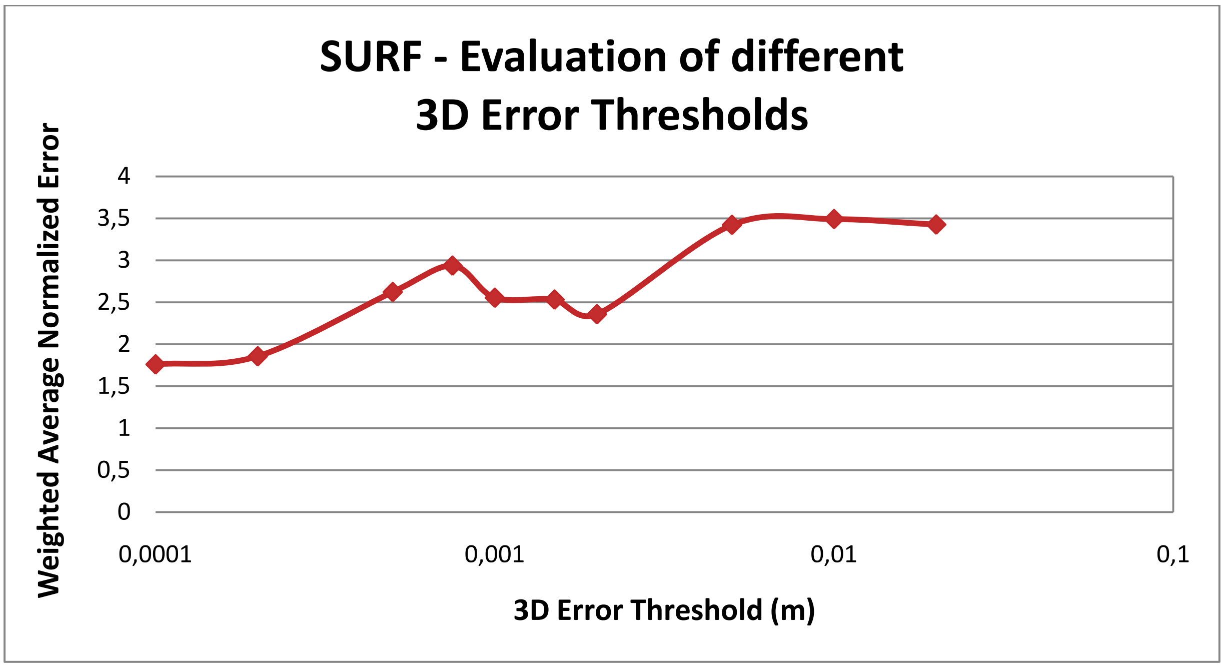


Figure 11

Final Parameters

	SIFT	SURF	FAST	ASIFT
Triangulation Threshold	2 mm	2 mm	1 mm	2 mm
Stereo Matching Ratio	0.5	0.65	0.2	0.6
Multi Matching Ratio	0.4	0.65	0.3	0.7
Reprojection Error Threshold	1.4 px	2.125 px	4 px	1.625 px
3D Error Threshold	0.5 mm	0.2 mm	0.5 mm	0.5 mm

Table 1 – Tuned parameters for the different salient point algorithms

4.2.1. Multi Descriptor Matching

As described in section 3.3, in case of few correspondences between 3D points, additional matching steps are performed to increase the number of matches. This may increase the chance of finding a solution, see section 4.3.

4.2.2. *Threshold Relaxation*

A lower RANSAC consensus threshold (Reprojection or 3D) typically leads to a lower final error. However with a lower threshold, the probability of finding no solution at all increases. To deal with this situation, a relaxation scheme is used. The thresholds listed in Table 1 are the initial thresholds. Should no transformation be found using this threshold q, the reprojection threshold is automatically relaxed by 1 px, at most 3 times. In other words, if q = 2 px provides no solution, the estimation is performed again with subsequently q = 3 px, 4 px, up to 5 px, until a solution is found. For the 3D Error the threshold is relaxed by doubling it each time (eg: subsequently 0.001, 0.002, 0.004, 0.008).

4.3. *Comparison*

4.3.1. *RANSAC Consensus Measure*

The two different consensus error measures are now compared on 14 different image sets (see Appendix A), both with and without threshold relaxation and multi-descriptor matching (denoted by the + symbol). The comparison is given per salient point algorithm.

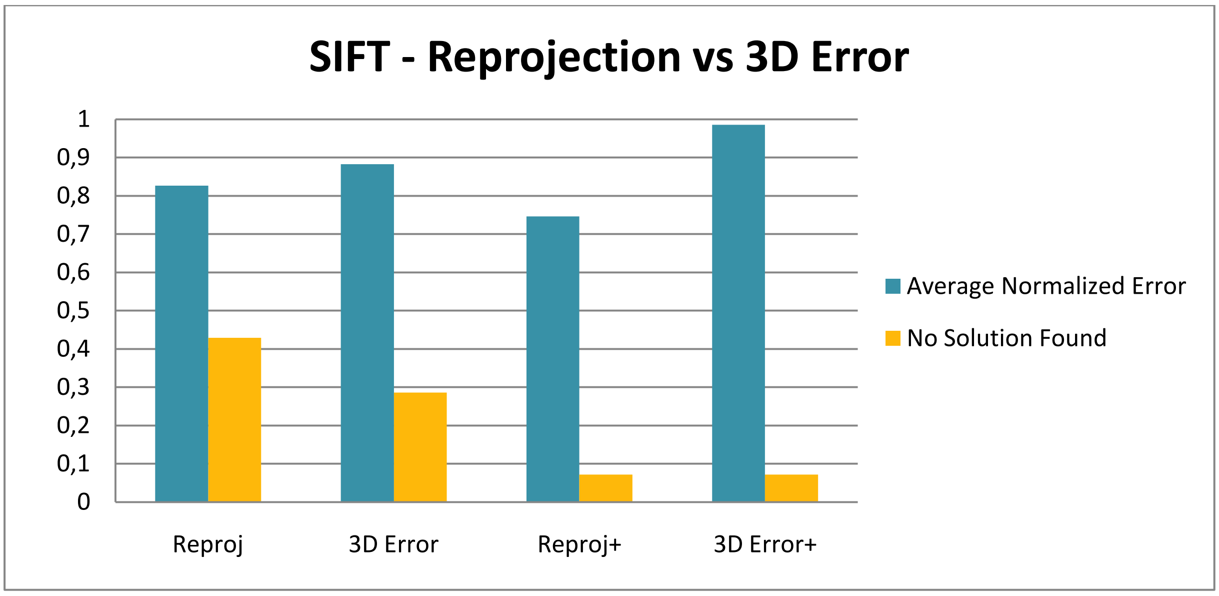


Figure 12

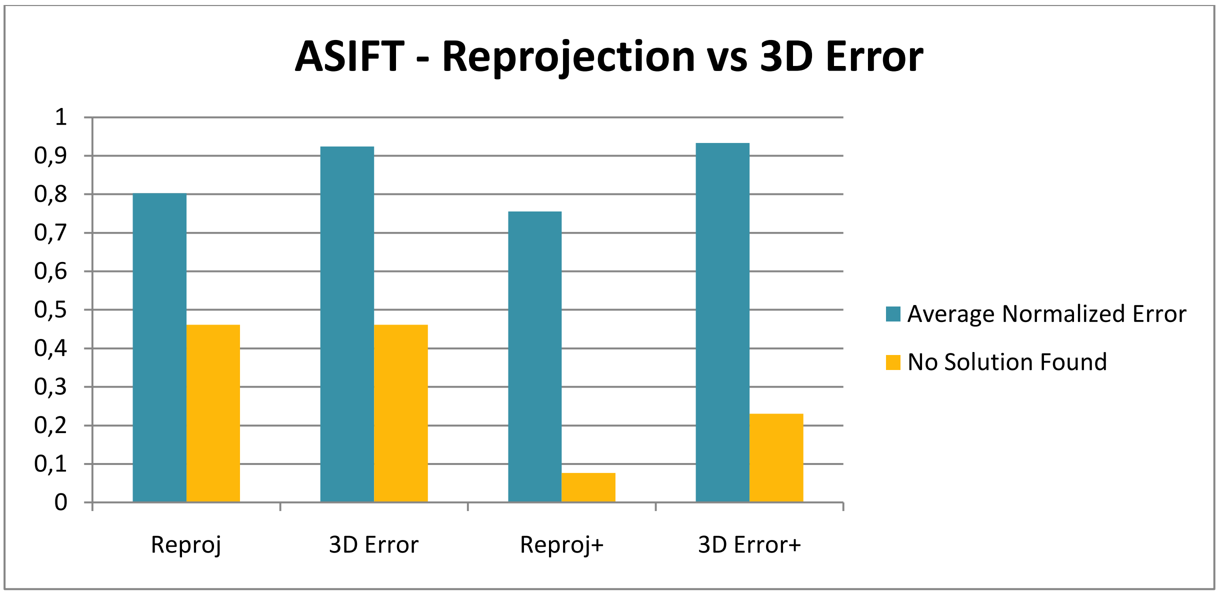


Figure 13

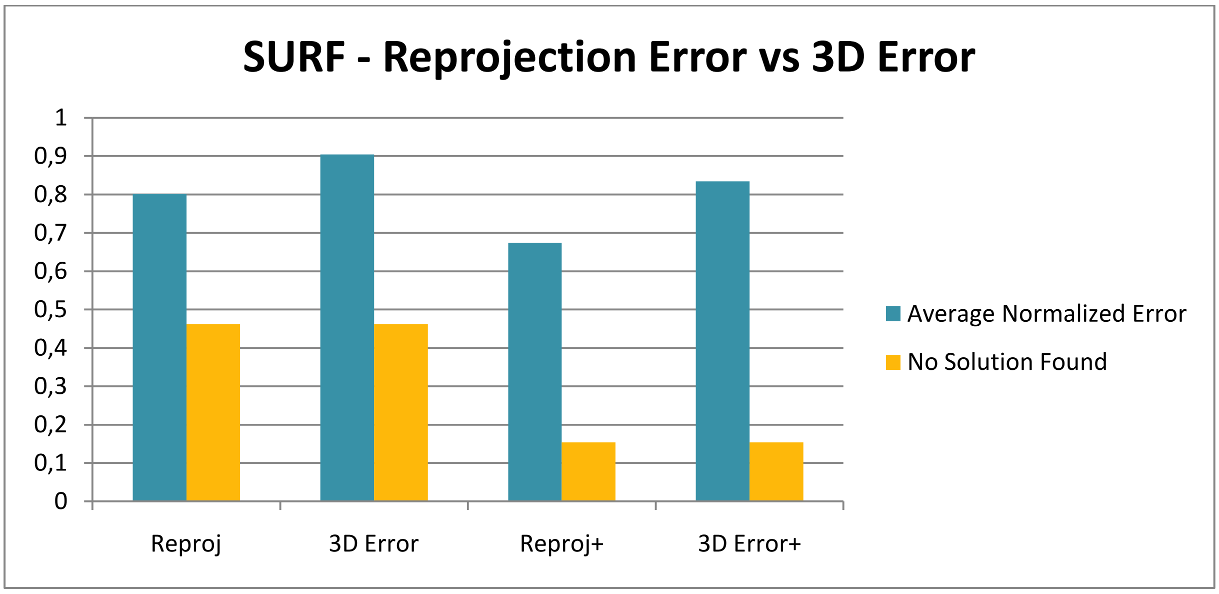


Figure 14

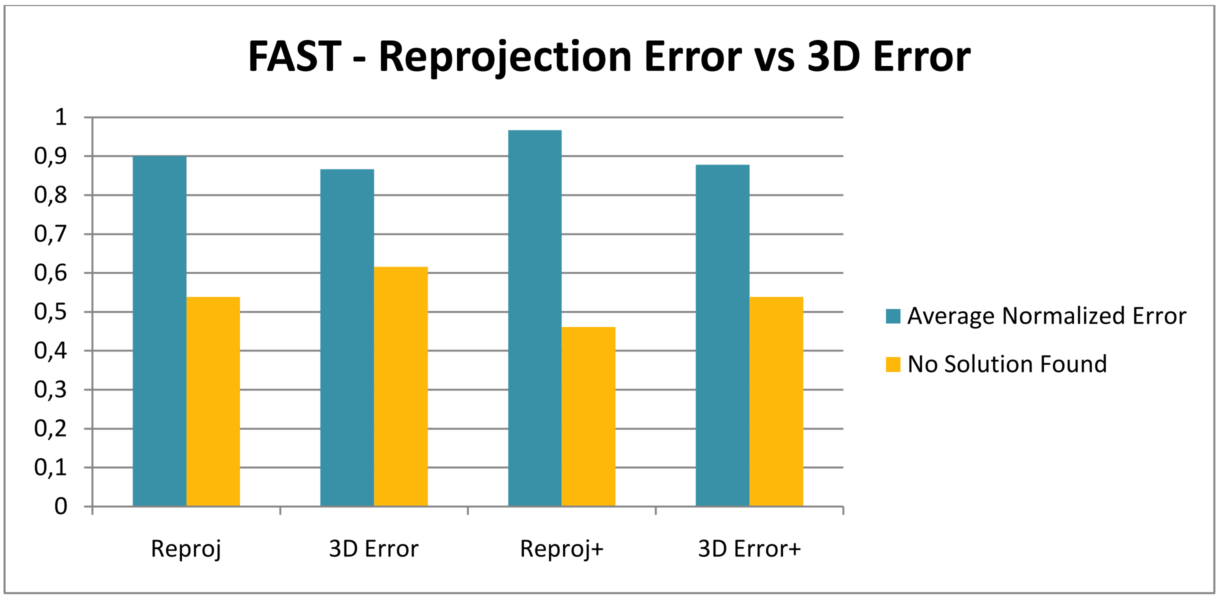


Figure 15

Although the Reprojection Error and the 3D Error measure both give a good indication of the fitness of a transformation for a given correspondence, it is clear that for SIFT, ASIFT and SURF the Reprojection Error measure gives better results. Also the addition of multi-descriptor matching and threshold relaxation improves both the error and the probability of finding a solution for these algorithms. Therefore for these algorithms 'Reproj+' is chosen as best configuration.

For FAST the results look a bit different. The Average Normalized Error is lower for the 3D Error and the extra steps (multi-descriptor and threshold relaxation) provide no improvement in terms of the average normalized error. They still increase the percentage of image sets for which a solution is found, which is an important criterion. For this reason the configuration with the highest percentage of solutions found is chosen for FAST, which is also 'Reproj+'.

The fact the Reprojection Error seems to perform best is not unexpected. Effectively the Reprojection Error measure is a weighted version of the 3D Error, diminishing the effect of errors far away from the camera. This is a desirable property, since we expect the 3D coordinates of the correspondences at those locations to be less exact; there is simply less resolution. In fact, if we lower the triangulation threshold the results of using the 3D error seem to be better than using the Reprojection Error, however for many image sets no solution can be found.

4.3.2. Salient Point Algorithms

In Figure 16 the four salient point algorithms are compared each using the Reprojection Error as RANSAC criterion and with multi-descriptor matching and threshold relaxation for optimal results.

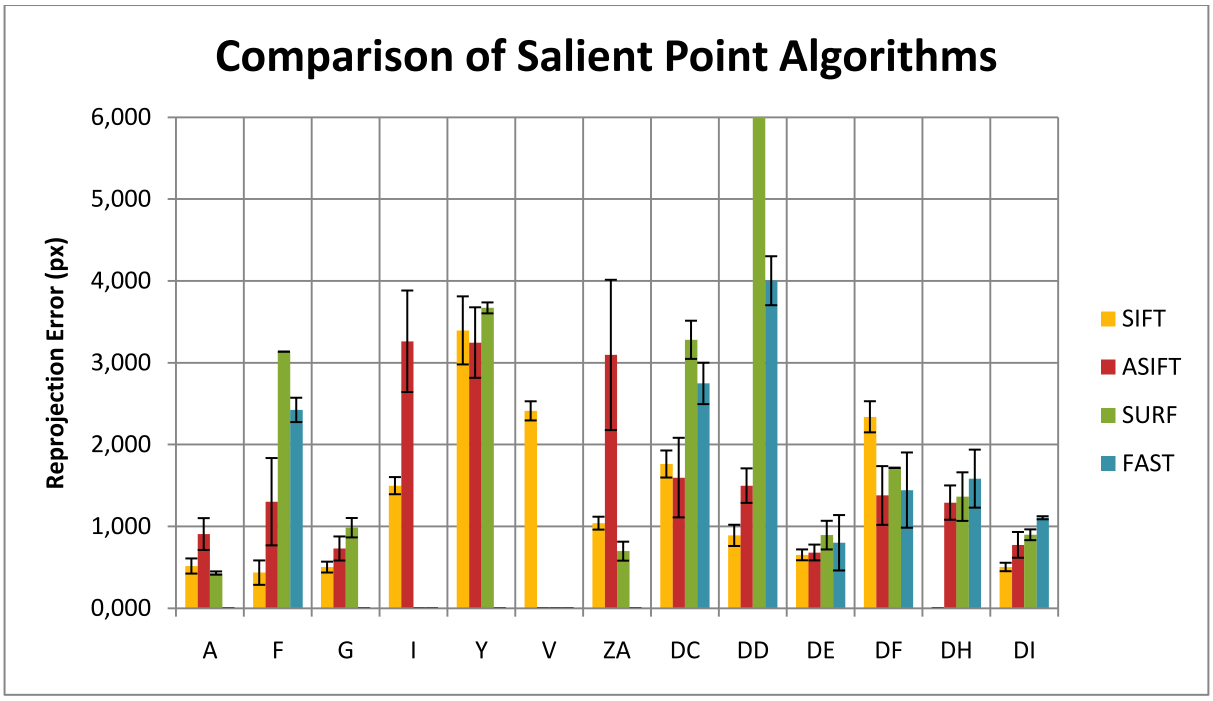


Figure 16 – OTESC performance on thirteen image sets using four different salient point algorithms.

In this graph it is not very clear which algorithms perform best. It is clear that no single algorithm performs best for all image sets. In Figure 17 the results have been normalized and averaged over all image sets for easier comparison. Based on these results it is clear that SIFT and ASIFT are most robust (find solution in most cases) and SIFT provides the lowest error, with respectively SURF and ASIFT taking second and third place. Based on this the overall best algorithm is SIFT, as it provides both a low error and most solutions.

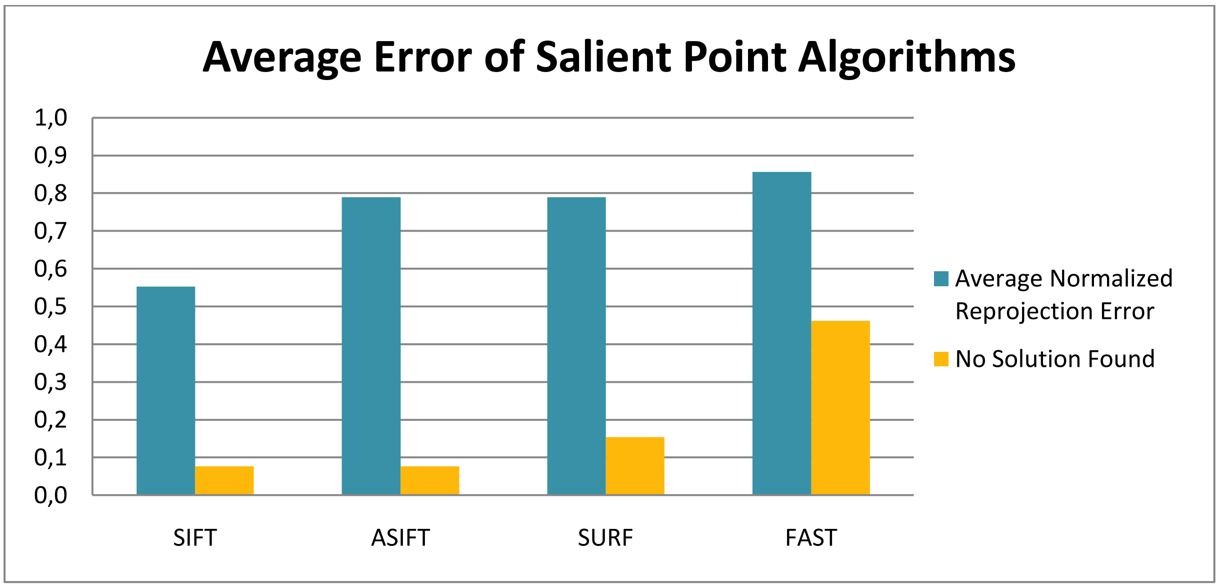


Figure 17

Another experiment conducted to compare the salient point algorithm is determining their robustness to changes in viewpoint. A computer generated image set (MA) was created with views from 14 stereo cameras increasingly further away from the first stereo camera. Next the OTESC algorithm was applied between the first stereo camera and each of the other stereo cameras (1 and 2, 1 and 3, 1 and 4, etc). The results, for each of the salient point algorithms, can be seen in Figure 18.

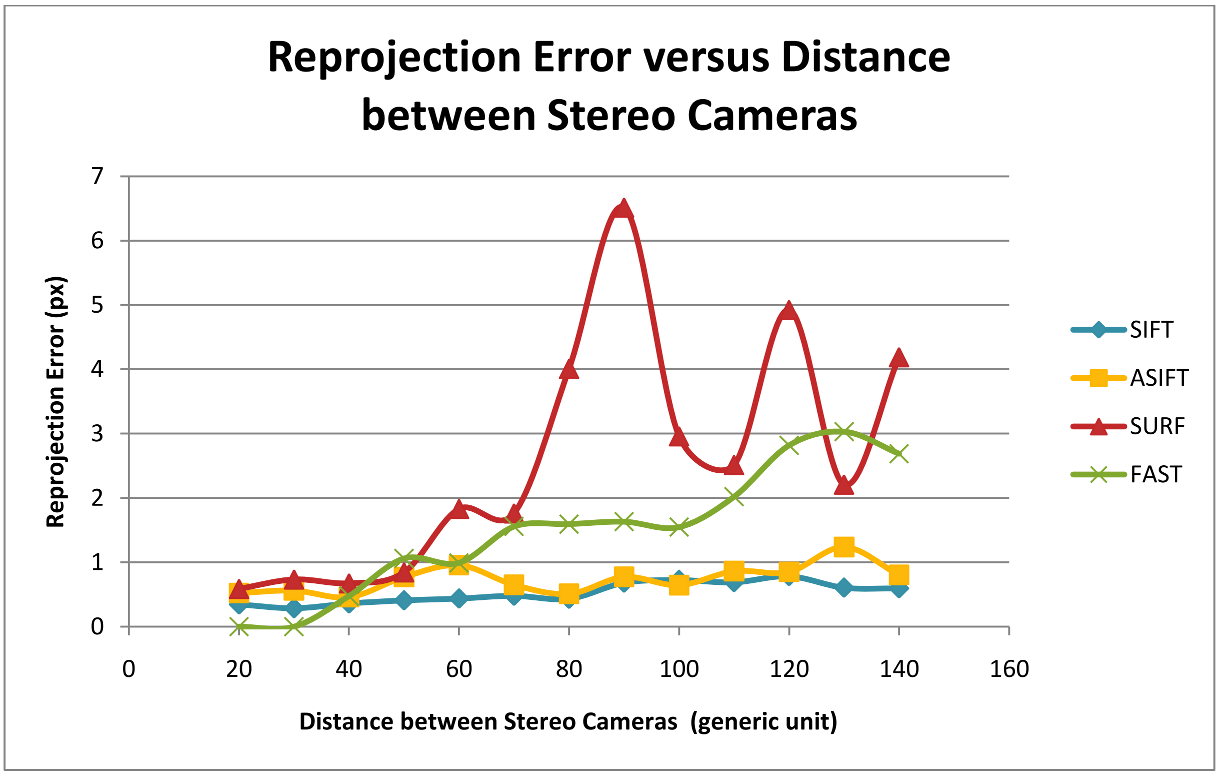


Figure 18

It is clear that SIFT comes out strong again, with very consistent performance even under increasing viewpoint differences. ASIFT has similar performance but FAST and SURF perform much weaker.

4.4. *Number of RANSAC iterations*

To determine an optimal number of iterations for the RANSAC algorithm used in OTESC, several measurements have been performed. For several image sets and with a maximum of 200000 RANSAC iterations, the results for up to 50000 iterations have been plotted in Figure 19. From these results it can be concluded that after around 10000 iterations the probability of finding a better model in the next iteration has decreased significantly (although the total probability of finding a better model (with infinite iterations) is still around 30%). Looking at the RANSAC consensus error, it is clear that around 30000 iterations the 'optimum' error has been approached by 99%, while for

10000 iterations this ratio is 84%. Based on these observations the ideal number of RANSAC iterations lies somewhere between 10000 and 30000 iterations, depending on the trade-off between accuracy and time.

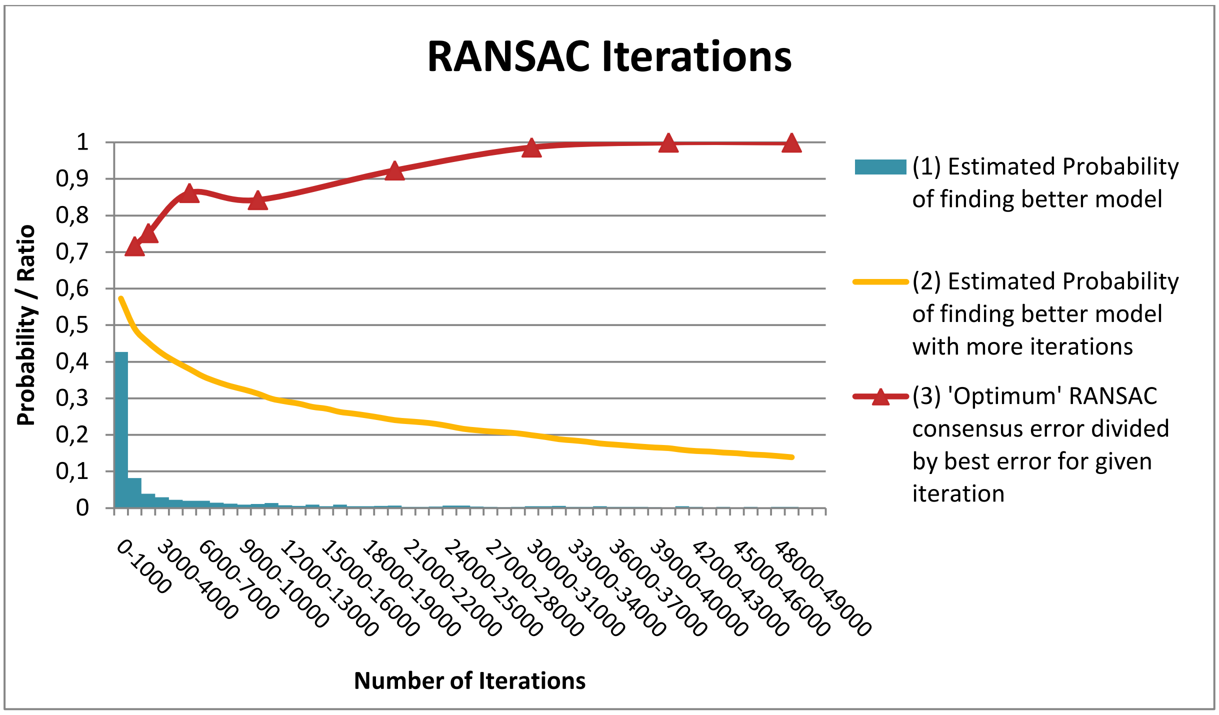


Figure 19 – Several measurements and observation to give an idea of the optimum number of RANSAC iterations for OTESC. All data was obtained using 200000 maximum iterations over several image sets. (1) is the normalized histogram of iterations (binsize = 1000) at which a better model than the previous was found. (2) is 1 minus the cumulative of (2). (3) is the best error obtained in 200000 iterations divided by the errors obtained at the given intervals.

4.5. Multi Camera Results

Two different strategies for set-ups with more than two stereo cameras were suggested. In the ideal case, both perform $n-1$ estimations for n stereo cameras. The goal is to find the position and orientation of each stereo camera relative to a common origin. The main difference between the two strategies is that with the **Minimal Set** strategy many transformations relative to the origin are calculated using ITE from few strong DTEs, however this may cause errors to propagate. For the **Origin** strategy they all have been calculated with DTE if possible, however potentially with very few correspondences.

We take a very simple case to evaluate the two strategies. Given three stereo cameras {1,2,3}, assume {1} is designated as the origin and there are very little correspondences for pair {1,3} and many between the other two pairs. Transformation T(1,3) is required. It must be verified which is more accurate: DTE from the few correspondences between {1,3} or ITE using {1,2} and {2,3} with many correspondences.

In Table 2 the results from four multi-camera image sets (with each three stereo cameras used) are shown. These results give no clear indication which strategy is superior.

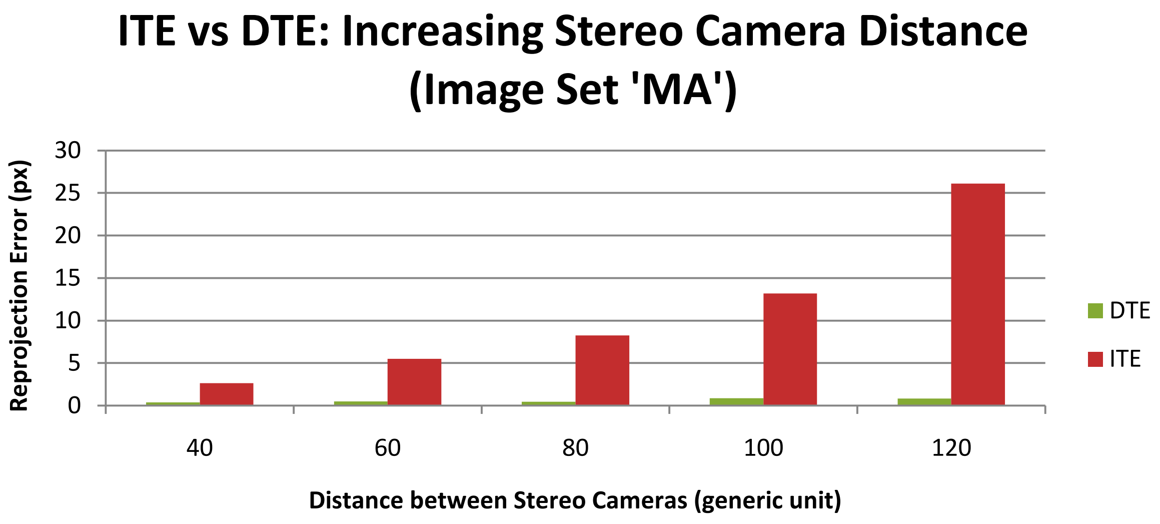
Minimal Set				
Image Set	**MA**	**MC**	**ME**	**MF**
Stereo Camera Pair {1-2}	DTE	DTE	ITE	DTE
Stereo Camera Pair {1-3}	ITE	DTE	DTE	ITE
Total Reprojection Error (px)	1,303	1,286	31,364	0,851

Origin				
Image Set	**MA**	**MC**	**ME**	**MF**
Stereo Camera Pair {1-2}	DTE	DTE	ITE	DTE
Stereo Camera Pair {1-3}	DTE	DTE	DTE	DTE
Total Reprojection Error (px)	0,68	1,26	31,41	1,05

Table 2 – DTE and ITE choices for the two multi camera strategies for different image sets. Also the resulting total reprojection error (sum of two errors) is shown. For image set MA the Origin set achieves a better performance by using a DTE instead of a ITE. However for image set MF using DTE instead of ITE leads to a larger total error. For image sets MC and ME both strategies make the same choice (on ME a DTE fails for the Origin strategy, so an ITE must be used).

A second experiment with two image sets with each 7 stereo cameras gives a clearer, although slightly ambiguous result. The origin is the left-most camera in the set, all cameras are facing the same direction (parallel) are at equal increments away from the origin. Effectively for the Minimal Set strategy all transformations between the origin and each stereo camera, except its direct neighbor, are calculated using ITE. With the Origin strategy all transformations are calculated using DTE.

The results are in Figure 20 and clearly show that ITE is highly sensitive to error propagation in image set MA. The results for image set MF are similar for the largest distance, 120 units, although less pronounced. For the smaller distances between stereo cameras in set MF the result of ITE is comparable to DTE. In none of the cases ITE gives a clear advantage over DTE, so in general the Origin strategy is the safest choice, which will only use ITE if DTE has failed completely. More information is required to determine in which cases ITE is able to outperform DTE.



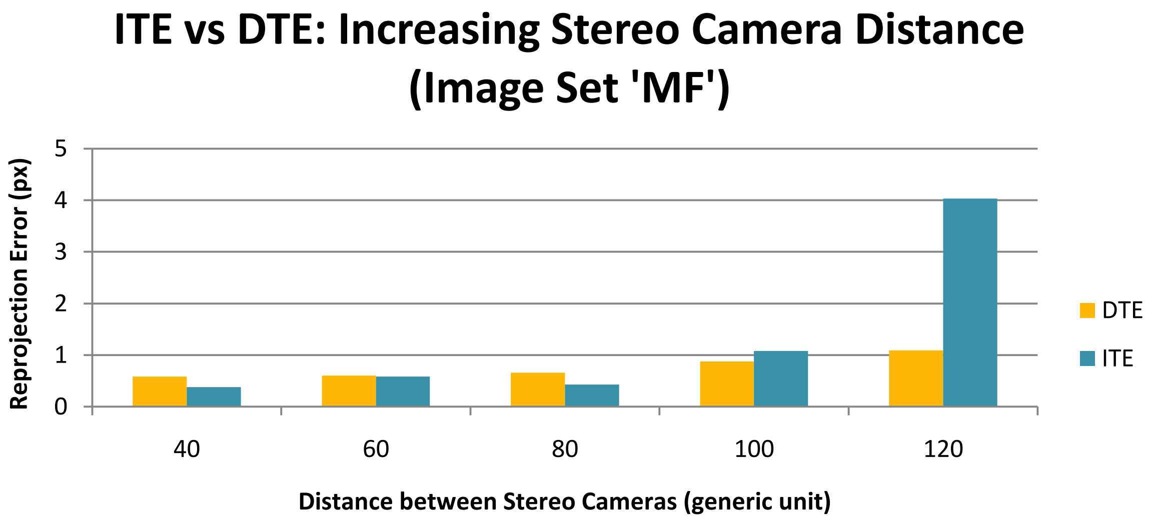


Figure 20 - Reprojection Error when using ITE or DTE to relate the origin to a stereo camera increasingly far away, for two image sets. As the camera is further away, more transformations from stereo cameras in between must be used to calculate the result, which can result in any errors to propagate.

4.6. System Performance

In the previous subsections the combination of SIFT and the Reprojection Error measure turned out to be the strongest combination and the best choice for our approach. Table 3 lists the results of our approach on several image sets and Table 4 gives some statistics for each set, such as the number of matches and processing time. A total of 13 image sets, each consisting of images from two stereo cameras, were selected to evaluate the proposed approach. Each image set has a different background and different foreground objects. Six image sets were computer generated, see Appendix A.

The distance and orientation between the stereo cameras varied, see Table 3 for an indication of the view overlap, in terms of the rough distance between the stereo cameras (specifically between the left view of both stereo cameras). For six of the thirteen images the reprojection error is smaller than 1 pixel, which is desirable for virtual view rendering. For the other sets the reprojection error is typically below 2.5 pixels, with set Y as the only exception. Note that the translation error is much larger than desired for some sets, while reasonably accurate (< 1cm) for other sets. As noted before, the found transformation is not wrong just because the translation error is very high.

Image Set	Distance Stereo Cameras (cm)	Translation Error (cm)	Rotation Error (degree)	**Reprojection Error (pixels)**
A	25	0,88	0,36	**0,516**
F	25	7,55	4,15	**0,435**
G	35	17,92	2,31	**0,502**
I	30	10,22	2,12	**1,498**
Y	55	3,87	1,64	**3,395**
V	55	25,33	3,90	**2,412**
ZA	45	1,04	1,38	**1,039**
DC	110*	4,27	3,77	**1,762**
DD	110*	0,80	1,35	**0,890**
DE	60*	1,08	0,84	**0,652**
DF	60*	1,48	1,63	**2,339**
DH	60*	N/A	N/A	**N/A**
DI	60*	13,28	0,36	**0,504**

Table 3 – Results of OTESC system (SIFT + Reprojection) on 13 image sets. The * denotes these image sets were computer generated; the mentioned distance is an indication.

Image Set	Matches stereo-extrinsic (1)	Consensus matches (2)	CPU Time (sec) (3)
A	41	18	3,28
F	26	10	3,52
G	30	20	4,81
I	26	12	3,53
Y	32	12	4,77
V	9	5	6,25

ZA	28	11	4,77
DC	67	26	5,53
DD	46	18	5,41
DE	131	60	8,27
DF	57	20	8,66
DH	23	N/A	8,91
DI	22	12	13,86

Table 4 - Statistics related to the image sets. (1) gives the number of (3D) matches found between the two stereo cameras and (2) shows how many of those matches were considered in-consensus with the final result (on average). (3) shows the processing time in seconds it took to compute an estimate (3.4Ghz Intel Core2Duo, multi-threaded).

Note that for the approach to work well, enough salient points should be detected which could constrain the type of scenes. Specifically it helps if the scene contains sufficient *interesting* objects and with little repetition of scene elements.

4.6.1. Virtual View Rendering

Virtual view rendering is interpolating the view from a virtual camera based on multiple known views. A requirement for virtual view rendering is knowing from which position (and orientation) these known views were taken relative to each other, in other words knowing the calibration. This calibration information can be obtained using the system described in this work. As the quality of this calibration has an effect on the quality of the virtual view image, it can be used as an evaluation method geared towards human perception.

Specifically we use virtual view rendering to compare a virtual view generated based on the calibration found by OTESC and based on the calibration provided by the manual calibration method. Depending on the significance of the visual differences we can conclude whether or not OTESC provides a calibration good enough for Virtual View Rendering.

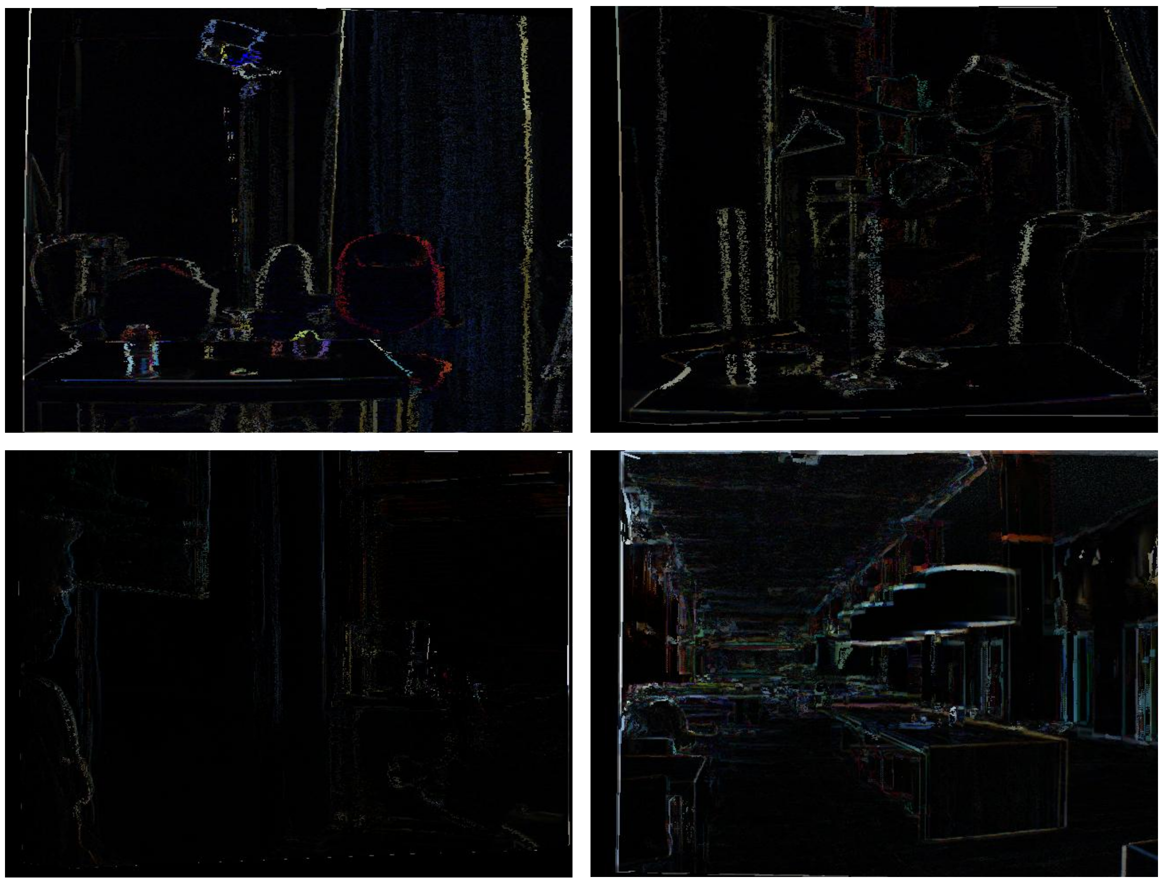


Figure 21 – Four difference images between a virtual view generated based on the ground truth calibration and the calibration found by OTESC. Image sets: $\begin{matrix} \boldsymbol{A} & \boldsymbol{F} \\ \boldsymbol{I} & \boldsymbol{V} \end{matrix}$, of which F, I and V are image sets with a quite large translation error compared to the ground truth.

As can be seen in Figure 21 there are visible differences between the virtual views generated based on OTESC and the virtual view generated based on ground truth calibration data. However these differences are mostly visible on the edges, indicating a small shift between the two images. Image set A, which has a low Reprojection, translation and rotation error, still shows significant difference at the edges. On image set V the differences seem larger than in the other images, which is consistent with the large translation / rotation error in Table 3. The virtual view based on OTESC appears slightly zoomed in.

Overall these errors do not significantly change the viewing experience for a user; the slight offset is most likely not noticeable. However based on these results it is very difficult to draw a meaningful conclusion; the quality of the virtual view rendering technique is too low. Also the virtual view rendering technique seems very sensitive to small calibration errors; there is a noticeable difference in quality even for the quite well calibrated image set A, see Figure 22. More details regarding the virtual view algorithm can be found in chapter 5.

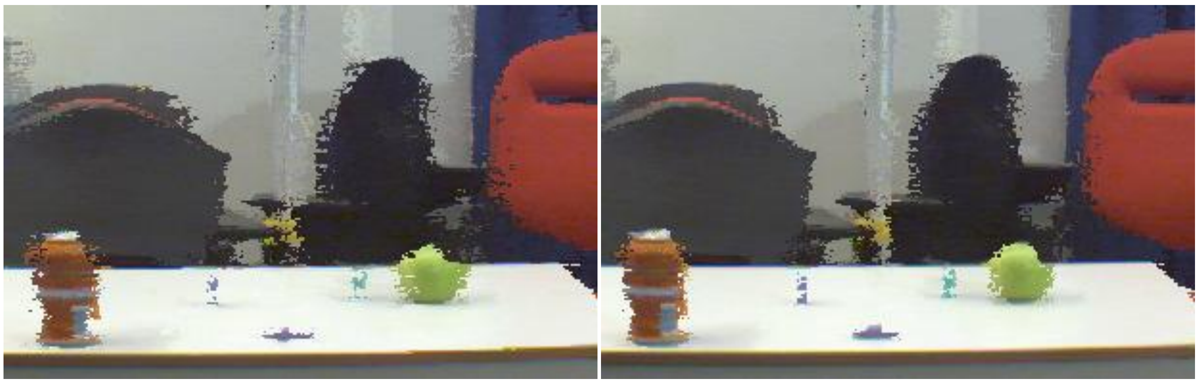


Figure 22 – Quality difference for image set A. (Left) OTESC based virtual view (Right) Ground truth based virtual view

5. *Virtual View Rendering*

Virtual view rendering, also known as multi view rendering, are techniques which can be used to generate (render) an image from a virtual camera position using imagery from real cameras. There are two kind of approaches toward virtual view rendering. Model-based rendering techniques explicitly try to reconstruct the 3D geometry (model) of the scene. Image-based rendering techniques do not use 3D geometry but work directly with the camera views. Hybrid approaches also exist [31].

The specific virtual view rendering technique used and implemented in this work is an image-based rendering technique and is discussed in more detail in [31]. Note that the algorithm is not specifically for stereo cameras; any set of calibrated cameras can be used. What follows is a brief explanation of the algorithm and its implementation.

5.1. *Algorithm*

Figure 23 gives an overview of the rendering algorithm for a given pixel. Suppose you have eight real views and want to generate a view (the empty image and grayed out camera in Figure 23) from a virtual position. Each pixel value of the virtual view is calculated independently using a ray tracing scheme, which makes this algorithm very suitable for parallel processing. The calibration information of the real views must be known and the calibration of the virtual view is also known, as it is selected by the user. The color value the virtual pixel should have, corresponds to (the projection of) a part of an 3D object (3D coordinate) in the scene. The line on which this object lies can be calculated by tracing a ray from the CCD through the lens into the scene (using the known calibration). However the depth along this line at which this object lies is unknown.

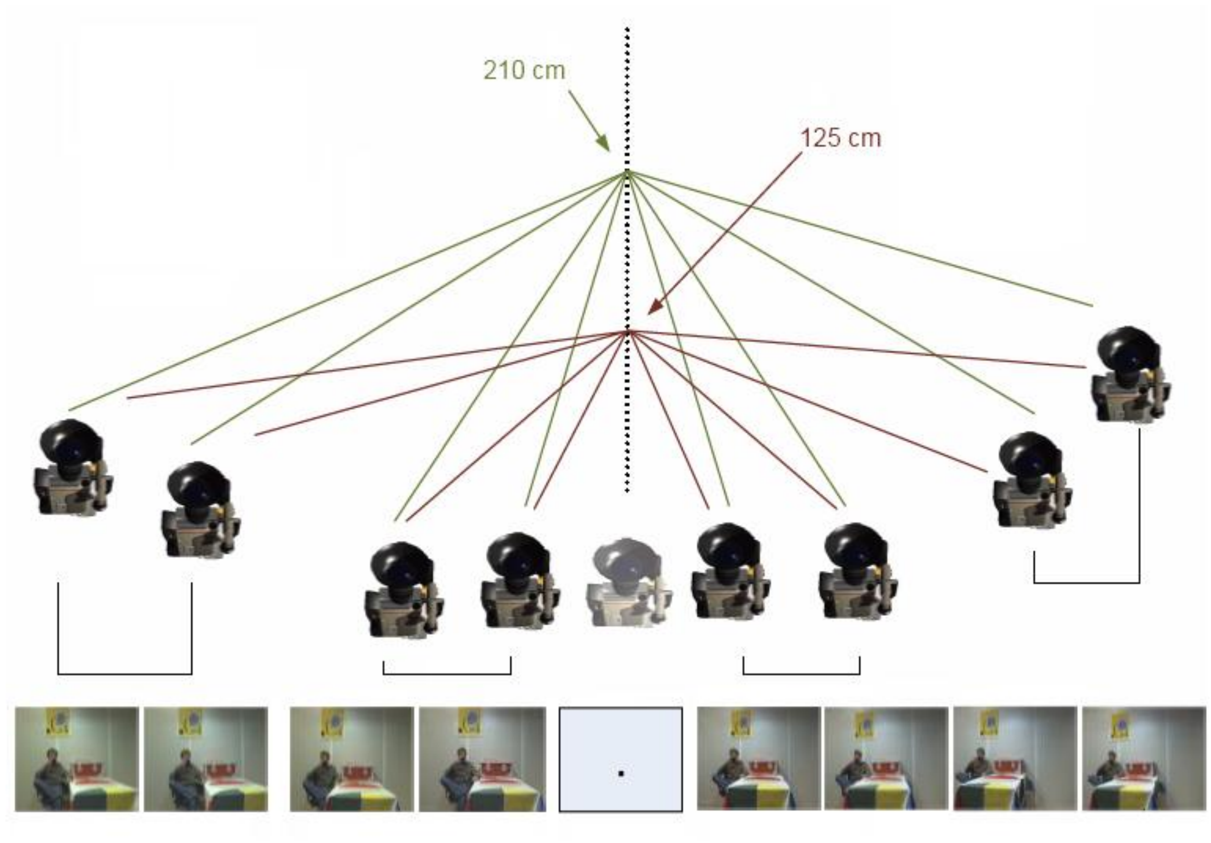


Figure 23 – Overview of Virtual View Rendering algorithm [31]**.**

The algorithm performs a brute-force search for a large number of depths along the line, specially for points between 1.00 and 3.00m along the line, with steps of 0.01m. So for each pixel a total of 200 depth hypotheses are checked. Each depth hypothesis corresponds to a 3D coordinate. Using the known calibration, the projection of this 3D coordinate (WCS) onto pixel coordinates (IMCS) for each real view are determined. If these pixel coordinates lie outside the image, the image is ignored for this specific depth hypothesis. If the pixel coordinates lie inside the image, its RGB value tells something about the object it sees. The basic assumption behind the search is that if the cameras all see a different color, there is no object at the given point and the pixels represent points behind the coordinate (Figure 24 - Left). And if all (or many) cameras see the same color for a given 3D point, there is an object (with that color) there (Figure 24 - Right). This is formalized using the variance of the RGB colors of each real view, where the depth with the lowest variance is chosen as the best depth (Figure 25). The corresponding mean color is chosen as the pixel value.

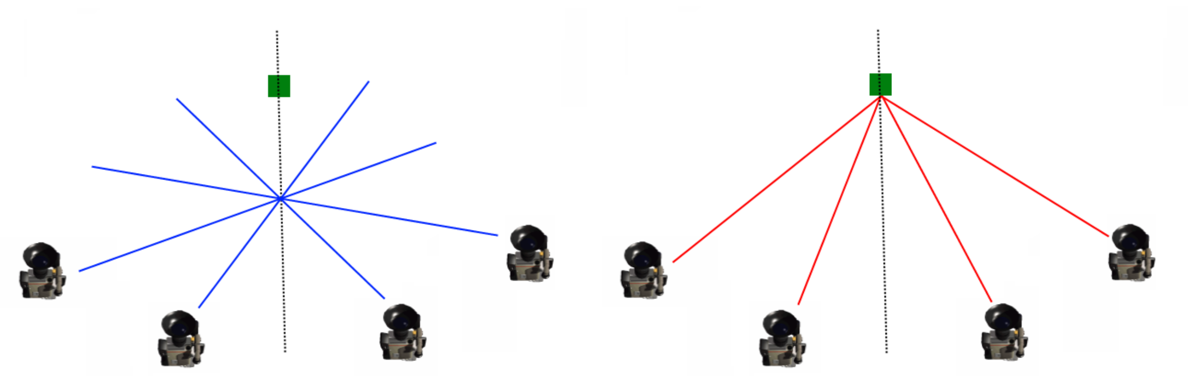


Figure 24 (Left) Disagreement between cameras; high color variance (Right) Agreement between cameras; low color variance (all green)

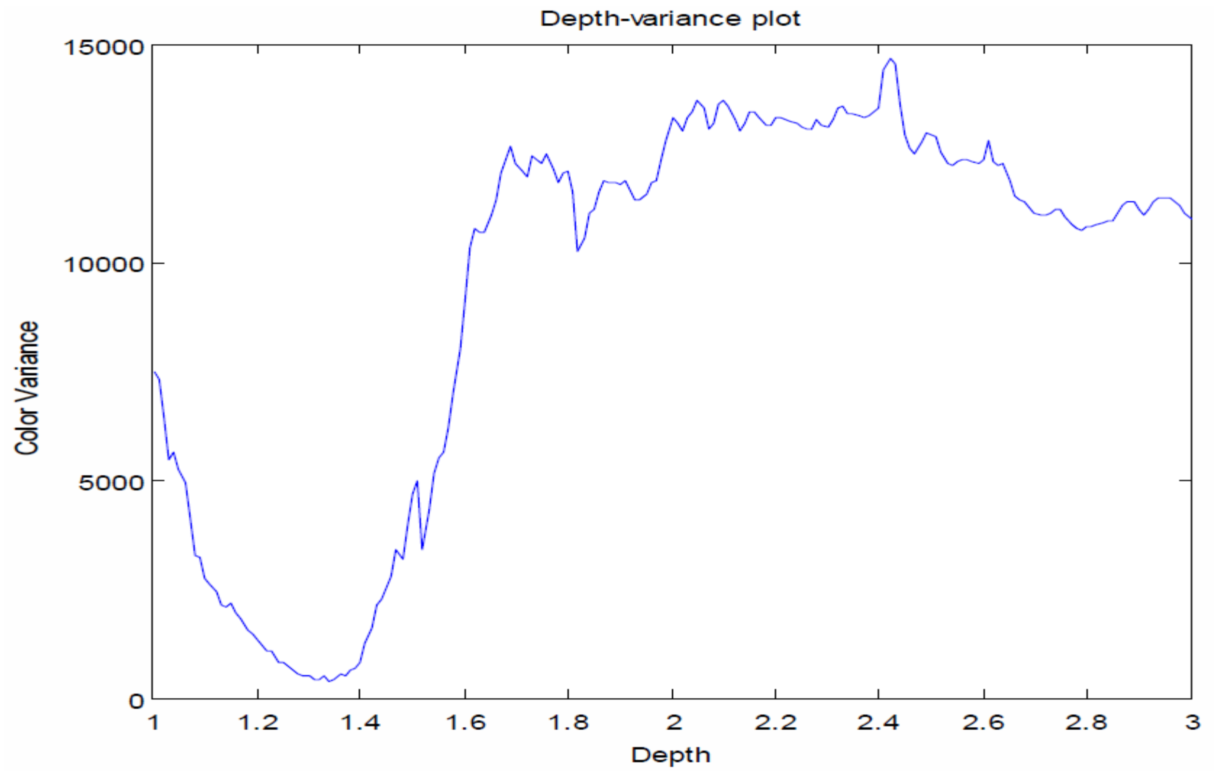


Figure 25 – Example plot of color variance for all depth levels ([31]**)**

In order to deal with occlusion and to improve the resulting image, the proximity of real cameras to the virtual camera is taken into account. Specifically each real camera c is associated with a weight w_c expressing the distance to the virtual camera vc:

$$w_c = \frac{1}{((x_c - x_{vc})^2 + (y_c - y_{vc})^2 + (z_c - z_{vc})^2 + 10^{-4})} \tag{16}$$

This weight w_c is then used in both the mean and variance calculation to give more influence to cameras near the virtual camera.

Using only one pixels’ color to match the depth among different views brings both a large advantage and a large disadvantage. The disadvantage is of course that one pixel alone is a very weak indicator of whether each camera sees the same object. The advantage is that for areas where the depth cannot be reliably estimated (by any photometric consistency matching strategy), such as uniform backgrounds, the algorithm will still pick the right color (although not the right depth) with great probability, which from a human perception point of view likely gives a smoother viewing experience.

5.2. Implementation

The algorithm was first implemented in C++ based on the details in [31] (the original source code was not available). This implementation took over 15 minutes to render a 1024x640 virtual view from eight real images, on an Intel Core2Duo 3.4Ghz PC (using multi-threading).

Next the algorithm was implemented in OpenCL. OpenCL (Open Computing Language) is a framework and programming language initially developed by Apple to perform parallel processing across heterogeneous platforms such as CPU (processor), GPU (video card) and in the near future mobile phones [32]. Currently the OpenCL specification is maintained by the Khronos Group and several large manufacturers like AMD/ATI, IBM, Intel, Nvidia are increasingly incorporating support for OpenCL in their products.

OpenCL has build-in vector math and other mathematical functions taking advantage of SIMD (Single Instruction, Multi Data) operations for speed-ups. Additionally running OpenCL code on a GPU tremendously speeds up the computation of parallel algorithms by taking advantage of the parallel architecture of GPU’s.

The resulting OpenCL implementation can render a single frame in about 0.5 seconds on a GPU (ATI Radeon HD 4870), which is roughly a 1800x speedup compared to the C++ implementation. Running the OpenCL implementation on CPU (Intel Core2Duo, 3.4Ghz) takes about 5.5 seconds, which is still a 180x speedup. Although the OpenCL code has been optimized, the algorithm structure is identical to the C++ code.

Unfortunately there is still an small unexplained difference in both quality and viewpoint position between our implementation and the one in [31]. Although the viewpoint of our implementation seems correct compared to ground truth images (whereas the one in [31] is slightly off), their output is of higher quality.

6. Conclusion

The goal of the system described in this paper is to calibrate multiple stereo cameras automatically; i.e. find the transformations that describe their relative positions and orientations. The calibration technique requires no calibration object or user interaction and the resulting calibration can be used for virtual view rendering in 3D video.

We have looked at combinations of several state-of-the-art salient point algorithms and error measures and showed that SIFT combined with the Reprojection Error criterion is most suitable for our system. Also we have applied several additional techniques, such as threshold relaxation and multi-descriptor matching to increase the robustness of our system.

Two distinct strategies to deal with three or more stereo cameras have been proposed and we have showed that combining multiple transformations quickly leads to issues due to propagation of errors.

We have showed that the chosen approach works well on a large number of image sets and for several of those sets we obtain a transformation that differs from the ground-truth with a reprojection error of less than 1 pixel.

Finally we have implemented an existing virtual view rendering algorithm in OpenCL to evaluate our system. The implementation gives a 1800 times speed increase when ran on programmable graphic hardware, compared to a reference C++ CPU implementation. Unfortunately the quality of the rendering algorithm is too low for evaluation purposes.

7. Future Work

Several issues remain unaddressed and several angles remain unexplored. For practical application of this approach the processing time should be decreased, for example using GPU hardware. Another goal should be to improve the robustness of the system in terms of practical use, for example by using parameter relaxation in more places. Also dynamic parameters can be used, for example the match ratios could perhaps be calculated based on the number of points available and the number of correspondences desired. The same applies to the number of RANSAC iterations required. When using this system in practice, the best transformation found should still be rejected if the RANSAC error measure exceeds a certain threshold. Also if using OTESC to continuously recalibrate the system, some mechanism must be in place to prevent 'overwriting' a good calibration with a worse calibration.

Another practical improvement could be to combine this approach with a method which also determines the stereo-intrinsic parameters from the scene images, using similar algorithms and techniques. This is possible, except that this stereo-intrinsic calibration cannot be determined in meters based purely on the scene, so the baseline could still be set manually. This would decrease the accuracy but would increase the flexibility of the overall system, as it would be an all-in-one calibration system.

More research needs to be done in the use of this system on multiple successive frames (at internals) to obtain more information about the scene than possible from a single frame, although it is unlikely this provides much more information when there is no scene movement.

One of the observations from the final results seems to be that the algorithm performs better on the computer generated images than on the real images. One hypothesis is that this is caused by the relatively poor quality of the real images, due to noise and out-of-focus effects. This hypothesis could be tested by using higher quality/resolution cameras to see if OTESC performs better there.

Unrelated to the OTESC system itself, but interesting for evaluation, is to use more advanced virtual view rendering techniques to compare the transformations found by OTESC with the ground truth transformations and judge the perceptive quality of both views. The virtual view rendering technique used as evaluation now, although interesting, proved to be of insufficient quality to draw solid conclusions.

8. References

[1] “Philips - 3D TV,” *http://www.philips.com/*.

[2] A. Redert, J.-jan V. Klaveren, and E. Hendriks, “Accurate 3D eye tracking for multi viewpoint systems,” 1997, pp. 224-228.

[3] B.J. Lei, E.A. Hendriks, and A.K. Katsaggelos, “Camera Calibration for 3D Reconstruction and View Transformation,” *3D Modeling and Animation: Synthesis and Analysis Techniques for the Human Body*, IRM Press, 2003.

[4] Z. Zhang, “Flexible camera calibration by viewing a plane from unknown orientations,” *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Ieee, 1999, pp. 666-673.

[5] R. Tsai, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses,” *IEEE Journal on Robotics and Automation*, vol. 3, 1987, pp. 323-344.

[6] A. Redert, J. Biemond, and E. Hendriks, “Multi-Viewpoint Systems for 3D Visual Communication,” 2000.

[7] Y. Furukawa and J. Ponce, “Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment,” *International Journal of Computer Vision*, 2009.

[8] M.I.A. Lourakis and A.A. Argyros, “SBA: A software package for generic sparse bundle adjustment,” *ACM Transactions on Mathematical Software*, vol. 36, Mar. 2009, pp. 1-30.

[9] D. Nistér, “An efficient solution to the five-point relative pose problem.,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, Jun. 2004, pp. 756-77.

[10] J. Jannotti and J. Mao, “Distributed Calibration of Smart Cameras,” *Proc Intl Wkshp on Distributed Smart Cameras*, 2006, pp. 56-61.

[11] A. Mavrinac and K. Tepe, *Feature-based calibration of distributed smart stereo camera networks*, IEEE, 2008.

[12] AlliedVisionTec, “Marlin F-046C,” *http://www.alliedvisiontec.com/us/products/cameras/firewire/marlin/f-046bc.html*.

[13] T. Schreuder, *Online Transformation Estimation between Pairs of Stereo Cameras*, 2009.

[14] D.G. Lowe, *Object recognition from local scale-invariant features*, IEEE, 1999.

[15] D.G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, Nov. 2004, pp. 91-110.

[16] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool, “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, 2008.

[17] J. Bauer, N. Sünderhauf, and P. Protzel, “Comparing Several Implementations Of Two Recently Published Feature Detectors.”

[18] E. Rosten and T. Drummond, “Fusing Points and Lines for High Performance Tracking,” *ICCV*, 2005.

[19] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” *European Conference on Computer Vision*, vol. 1, 2006, pp. 430-443.

[20] Y. Guoshen and J.-M. Morel, “A fully affine invariant image comparison method,” *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pp. 1597-1600.

[21] J.-michel Morel and G. Yu, “ASIFT: A New Framework for Fully Affine Invariant Image Comparison.”

[22] R. Hess, “SIFT Feature Detector, implementation C++,” *http://web.engr.oregonstate.edu/~hess/*, 2010.

[23] C. Evans, “Notes on the OpenSURF Library,” *csbrisacuk*, 2009.

[24] B.K.P. Horn, “Closed-form solution of absolute orientation using unit quaternions,” *Journal of the Optical Society of America A*, vol. 4, Apr. 1987, p. 629.

[25] M.A. Fischler and R.C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, 1981.

[26] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour, “Closed-form solution of absolute orientation using orthonormal matrices,” *Journal of the Optical Society of America A*, vol. 5, Jul. 1988, p. 1127.

[27] A. Lorusso, D.W. Eggert, A.L.D.W. Eggert, and R.B. Fisher, “A Comparison of Four Algorithms for Estimating 3-D Rigid Transformations,” 1995.

[28] A.J. Lacey, N. Pinitkarn, and N.A. Thacker, “An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibration,” 2000.

[29] M. Zuliani, C.S. Kenney, and B.S. Manjunath, “The multiransac algorithm and its application to detect planar homographies,” 2005.

[30] P.H.S. Torr and A. Zisserman, “MLESAC: A New Robust Estimator with Application to Estimating Image Geometry,” vol. 78, 2000.

[31] A. Manta, A. Redert, and E. Hendriks, “3DTV rendering from multiple cameras,” *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video, 2009*, pp. 1-4.

[32] “OpenCL - The open standard for parallel programming of heterogeneous systems,” *http://www.khronos.org/opencl/*.

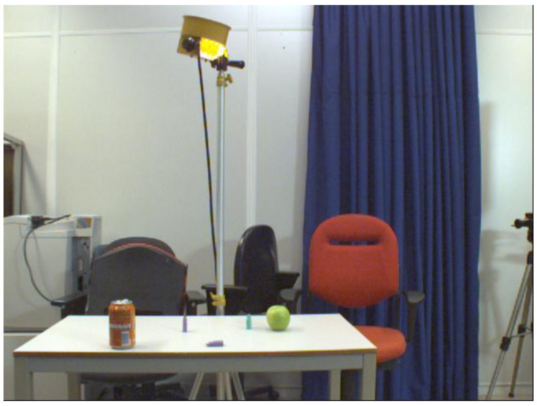
[33] M. Brown and D. Lowe, “Invariant Features from Interest Point Groups,” 2002, pp. 656-665.

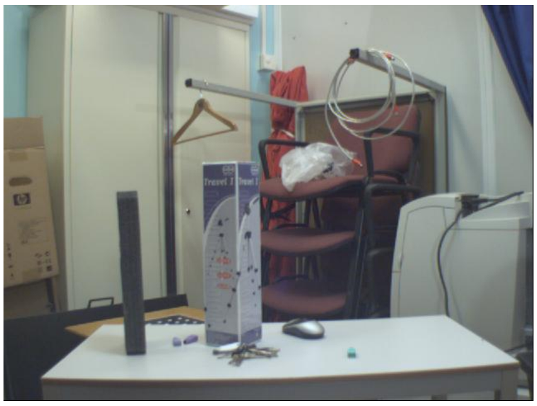
[34] P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, IEEE Comput. Soc, .

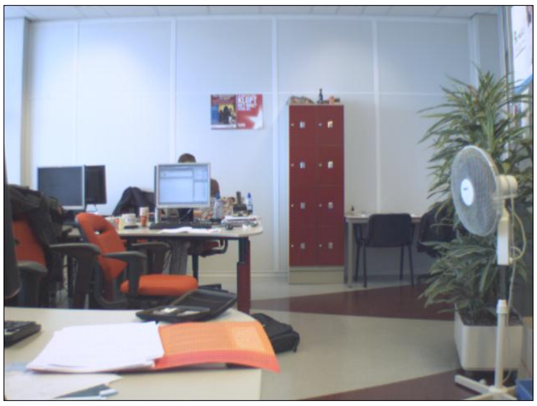
[35] P.Y. Simard, L. Bottou, P. Haffner, and Y. LeCun, “Boxlets: A fast convolution algorithm for signal processing and neural networks,” 1999.

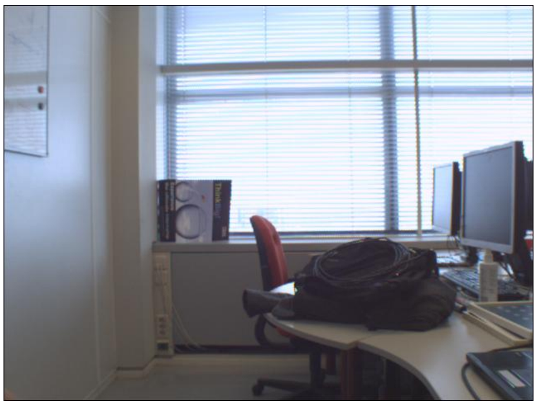
[36] E. Rosten, R. Porter, and T. Drummond, “Faster and better: a machine learning approach to corner detection.,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, Jan. 2010, pp. 105-19.

Appendix A Image Sets


A


F / MC


G


I

Y

V

ZA

DC

DD

DE

DF

DH

DI

VVB / MA

MA

MF

MD

Appendix B   Salient Point Algorithms - Details

B.1. SIFT

SIFT [14] stands for Scale-Invariant Feature Transform and is an algorithm aimed at performing object recognition, by matching many local image features instead of using a few complex (for example geometric) features.

B.1.1. Algorithm

Image points are selected as salient points (keypoints) if they are a local extreme of a Difference-of-Gaussian (DoG) function, and are searched for across all possible scales of the scale-space of the image. The scale-space $L(x, y, \sigma)$ of an image $I(x, y)$ is defined as the convolution of the image with a Gaussian G at scale σ:

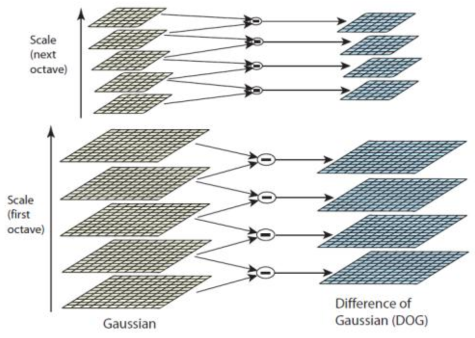


Figure 26 – Scale Space (left) and Difference of Gaussian (right)

$$L(x, y, \sigma) = I(x, y) * G(x, y, \sigma) \tag{17}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{18}$$

The scale space can be divided in several octaves, for each next octave the image is sub sampled by a factor 2, which correspond to a doubling of scale σ. Within an octave the scale σ is increased by a factor k (less than 2). Between each successive scale within an octave, the DoG is calculated with:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{19}$$

where k is $2^{1/s}$, with s the number of scales per octave. A value s of 3 scales per octave was found to give the highest repeatability [15].

Whether a point is a local extreme, and thus selected as candidate interest point, is checked by first comparing a point with its 8 neighbors at the same scale. If it is the minimum or maximum at this scale, the point is compared with it 9 neighbors in the scale below and if it's still a local extreme also with the 9 neighbors in the scale above. For a typical image (500x500 pixels) this leads to about 1000-2000 candidate interest points. Note that a total of s + 3 Gaussian images must be produced per octave, to obtain s + 2 DoG images. The two extra DoG images (highest and lowest scale within the octave) are required when determining if a point is a local extreme.

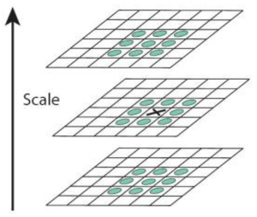


Figure 27 – Nearest neighborhood used for determining the extreme

In the original version of SIFT [14] the location and scale of the sample point closest to the extreme was used as position for the key point. In a revised version of SIFT [15] the location of the extreme is interpolated by fitting a quadratic Taylor expansion of the Difference-of-Gaussian scale-space function $D(x, y, \sigma)$ on the extreme sample point and its neighbors, with:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{20}$$

where D and its derivatives are evaluated at the sample point and:

$$\mathbf{x} = (x, y, \sigma)^T \tag{21}$$

is the offset from the sample point. This can be rewritten to determine $\hat{\mathbf{x}}$, the interpolated location of the extreme:

$$\hat{\mathbf{x}} = \frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{22}$$

This approach provides substantial improvements to matching and stability [33]. The result of the quadratic function for the interpolated extreme ($D(\hat{\mathbf{x}})$) is also used to discard points that have a low contrast (low Difference-of-Gaussian), for which the original authors apply a threshold of 0.03 (assuming pixel values in the range [0 – 1]). If the offset of the extreme $\hat{\mathbf{x}}$ is larger than 0.5 in any direction, the processing is repeated with the sample point that lies in that direction, as the extreme must lie closer to that sample point.

Next step is to discard interest points that lie on an edge. The Difference-of-Gaussian function has a strong response along edges, but unfortunately there are usually many similar points on an edge. Finding a correspondence for an edge point results in many potential matches, because the location along the edge is poorly determined [15] and therefore all edge points must be discarded. The ratio between the two principal curvatures is calculated and any keypoints with a ratio above 10 are eliminated.

B.1.2. SIFT Descriptor

All keypoints that remain are kept as interest points and are described using the SIFT descriptor. The descriptor allows easy comparing with other interest points. The descriptor consists of a feature vector of 128 elements.

Orientation

By describing the interest point relative to its main orientation, the description can be kept rotation-invariant. The main orientation is estimated by determining the dominant local image gradient direction(s). Using pixel differences to calculate the gradient orientation $\Theta(x, y)$ and gradient magnitude $m(x, y)$, an orientation histogram with a bin size of 10 degree is created.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (23)$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L((x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \quad (24)$$

These gradient calculations are performed for all samples within a region around the keypoint, in the image $L(x, y, \sigma)$ at the scale σ of the keypoint. The calculated magnitude and a Gaussian-weighted circular window (σ_{window} = 1.5 $\sigma_{keypoint}$) are used as a weighting factor for each sample in the histogram. If there are any secondary peaks within 80% of the highest peak within the histogram, additional interest points are generated for each peak. For each found peak in the histogram, a more accurate orientation is interpolated by fitting a parabola over the 3 histogram values closest to the peak.

Feature Vector

For each keypoint, again using $L(x, y, \sigma_{image})$, with σ_{image} = $\sigma_{keypoint}$, the gradient orientation $\Theta(x, y)$ and gradient magnitude $m(x, y)$ for 16x16 samples in a region around the interest point are calculated. The orientation of the region is the main orientation found in the previous step. The gradient orientation of the samples are used to construct 4x4 histograms (Figure 28), which are again weighted with their gradient magnitude and a Gaussian weighting function ($\sigma_{Gaussian}$ = 0.5 $\sigma_{keypoint}$). This gives 4x4 histograms with 8 bins, which is (4x4x8 =) 128 values in the feature vector. In the original paper [14] a descriptor of 160 values is described, using additional 2x2 histograms sampled from the image at one octave higher.

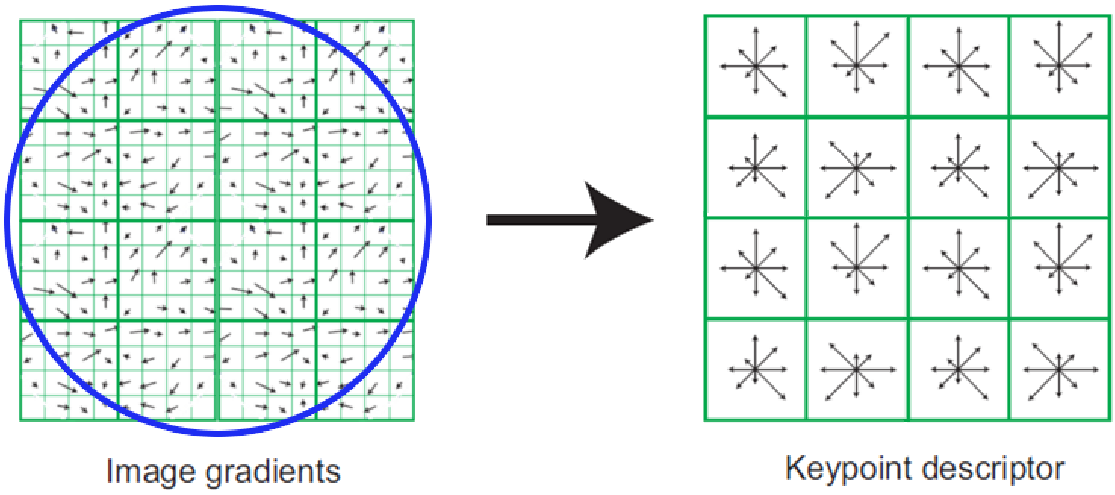


Figure 28 (Left) Sample window for the feature vector with Gaussian weighting function as circle. (Right) The histograms (each bin represented as vector) as keypoint descriptor.

To reduce the effect of illumination changes the descriptor is reduced to unit length. Any dominant orientation that has a magnitude of 0.2 or higher after normalization is clipped to 0.2. After that the vector is re-normalized. The value of 0.2 was established experimentally [15] and puts more emphasis on the distribution of orientations instead of matching the magnitudes.

B.1.3. Matching

The matching of interest points is done by taking the Euclidean distance between their descriptor vectors. To accept a match as a correspondence, the ratio between the best and second-best match must be below 0.8. If it is above that ratio, no reliable choice can be made between the two. This ratio was determined experimentally and eliminated 90% false positives and less than 5% true positives in an experiment with randomly transformed images against a database of 40000 keypoints [15].

B.2. SURF

A relatively new interest point detector and descriptor is SURF, Speeded Up Robust Features [16], which is partly inspired by SIFT. This algorithm is relatively quick compared to SIFT, about 3 times faster, and the authors claim it has similar or slightly better performance. In another comparison [17] it is also concluded that SURF outperforms SIFT when it comes to viewpoint changes, which is of course an important criterion. SURF does find fewer matches, and is slightly less invariant to illumination changes.

The normal version of SURF is invariant to translation, scaling and rotation. A version called U-SURF (upright SURF) is only invariant to rotations up to 15°, but is faster than regular SURF. Since in most practical situations the view rotation is typically below 15°, it may be sufficient to use U-SURF for camera calibration.

B.2.1. Algorithm

The algorithm works by calculating a Hessian matrix H per image point, at different scales σ. This detects blob-like structures where the determinant is highest. The Hessian matrix is given by:

$$H(\mathbf{x},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix} \tag{25}$$

Where $L_{xx}(\mathbf{x},\sigma)$ is the convolution of the image point $I(\mathbf{x})$ at coordinate $\mathbf{x}$ with a Gaussian $g(\sigma)$ second order derivative in the horizontal direction:

$$L_{xx}(\mathbf{x},\sigma) = \frac{\partial^2}{\partial x^2} g(\sigma) * I(\mathbf{x}) \tag{26}$$

In SURF the values of $L_{xx}(\mathbf{x},\sigma)$ are approximated by $D_{xx}(\mathbf{x},\sigma)$. Instead of having a discrete 2nd order Gaussian derivative, the derivate is approximated by using a box filter, having just three or four areas with a fixed multiplier, as can be seen in the figure below.

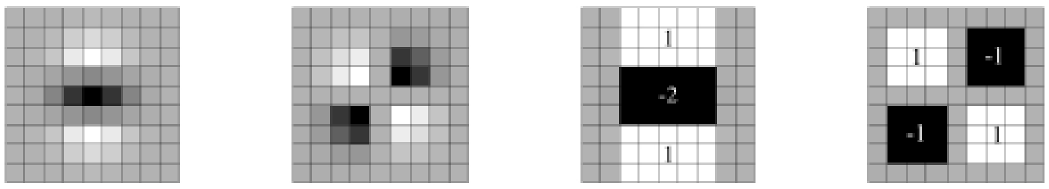


Figure 29 (Left) Discrete 2nd order Gaussian as used by L_{yy} and L_{xy}. (Right) Box approximation of 2nd order Gaussian used by D_{yy} and D_{xy}

The approximated determinant of the Hessian matrix is now given by:

$$det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{27}$$

The value w is required for energy conservation between the real and approximated Gaussian kernel. Although the exact value of w depends also on the scale, a fixed value of 0.9 can be used without significant impact [16]. The box filter approach allows an important speedup using a technique called Integral Images.

Integral Image

Integral Image [34,35] is a quick method to calculate the integral over an area. The Integral Image is pre-computed, so that every point in the image contains the integral sum of the area in the original image that is above and to the left of that point. When the integral of an area for the original image must be computed, for example to calculate the approximated second order Gaussian, the Integral Image can be used to perform this calculation without visiting all pixels, instead visiting only the corners of the area in the Integral Image. The integral of the area is given by A-B-C+D, where A, B, C, D are the corners of the area. This optimization contributes a lot to the speed of SURF, as calculating the integral is independent of the area size.

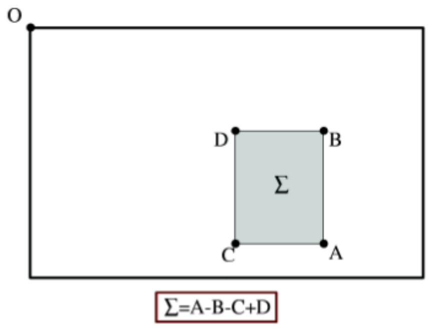


Figure 30 – Calculation of area from pre-calculated Integral Image

Scale-Space Representation

To obtain scale-invariance the determinant of the Hessian matrix is calculated at different scales, using a scale space, which is similar to SIFT. Scale-spaces are typically implemented using an image pyramid, where the image is down sampled for higher pyramid levels. In SURF the image itself is not down sampled, but is instead smoothed with a box filter of increasing size, which allows exploiting the integral image again. Just like SIFT, each octave is subdivided into a constant number of scale levels.

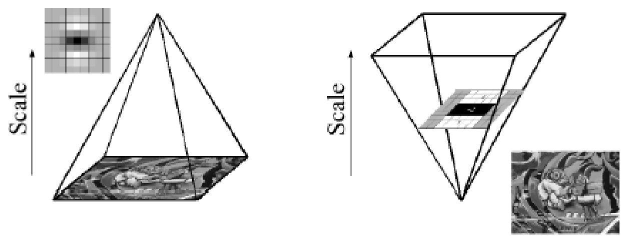


Figure 31 – Instead of scaling the image (Left), the filter kernel is scaled (Right), which can be done much more efficiently due to the use of Integral Images

The initial filter is size 9x9 for the first octave (approximately equivalent to a Gaussian derivative of σ = 1.2), and the filter size is increased by 6 (15x15) for each subsequent scale. This is continued until the filter exceeds twice the initial

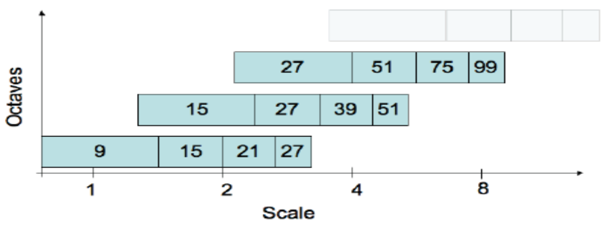


Figure 32 – Used kernel filter size at different octaves at Gaussian scales

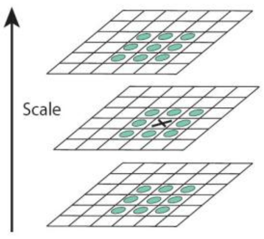


Figure 33 – Nearest neighborhood used for non-maximum suppression

size (so it exceeds the octave). This completes one octave. The next octave starts at the filter size + filter size increase. In each new octave the 'filter size increase' is doubled (6 to 12 to 24 etc). The overlap in octaves is required because the first and last scales in the octave are used only to compare against extremes, not to find them. For each potential interest point its 3x3x3 neighbor region is checked to see if this point is the local extreme determinant (non-maximum suppression), very similar to SIFT. The exact location of the interest point is interpolated in scale and image space, giving subpixel precision. The scale *s* at which the determinant was extreme is stored with the interest point for later use.

B.2.2. SURF Descriptor

The next step is to describe these points using the SURF descriptor. This description is needed to match the point with its corresponding point in another image. Depending on the purpose, the descriptor size can be chosen to be between 36 and 288 elements, the most common being 64 of 128 elements (called SURF-64 and SURF-128 resp.). A bigger descriptor allows for more exact matching, but also takes more computation time. The descriptor stores distributions of intensity using Haar wavelets as a simple gradient description.

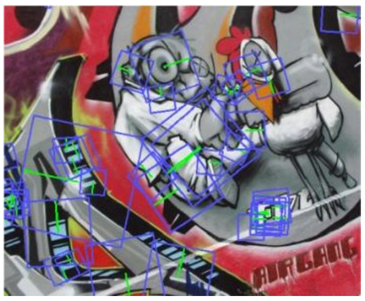


Figure 34 – Visualization of interest point scale and orientation as found by SURF

Orientation

For the case of regular SURF (in contrast to U-SURF), the main orientation of the interest point must also be determined, to allow the descriptor to describe the point in a rotation-invariant manner. Haar wavelets (side length 4*s*) are used on sample points (sampled at a step size of *s*) within a radius of 6*s* around the interest point. The Haar responses are then weighted using a Gaussian with σ=2*s* [16] ([23] claims 2.5s), and represented as points in space. From the horizontal and vertical responses, within a sliding orientation window of angle π/3, a vector is calculated. The longest vector found is chosen as the main orientation.

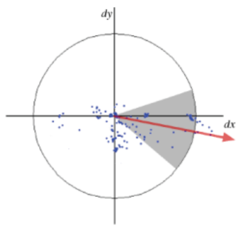


Figure 35 – Sliding orientation window, the red vector is the sum of all responses in the window

Feature Vector

The found scale *s* per interest point determines the size of the sample window, 20*s*. The sample window is split into 4x4 subregions for SURF-64, and within each region the Haar wavelet of 25 (regular) points is calculated at scale 2*s*.

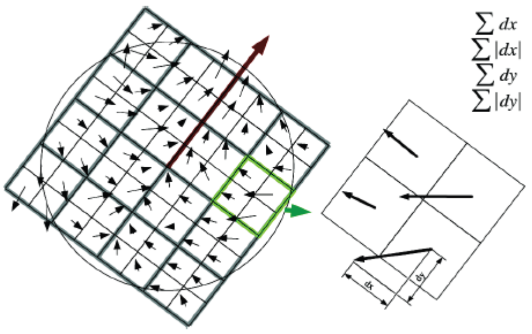


Figure 36 – Oriented sample window with Haar response per sample

The resulting Haar responses (dx and dy) are weighted by a Gaussian with σ=3.3*s* to increase robustness towards geometric deformations and localization errors. The responses are summed into four values for each subregion:

$$v_{subregion} = \left[\sum dx \quad \sum dy \quad \sum |dx| \quad \sum |dy|\right] \tag{28}$$

These four values represent the subregion, which gives (16 regions * 4 values =) 64 elements. In the case of for example SURF-144 the window is divided into 6x6 regions (36 regions * 4 values = 144). For SURF-128 the responses are summed into 8 instead of 4 values per subregion (16 regions * 8 values = 128).

B.2.3. Matching

When matching the point descriptors found in one image with the point descriptors found in another image, the distance between descriptors is calculated with the sum of squared errors between elements. There is a speedup here, to make sure only descriptors of the same contrast are compared. A match is found if the ratio between the distance of the first and second-best match is smaller than some threshold. In [16] a ratio of 0.65 was found to work well. The idea is that if the second-best match is too close, no good distinction can be made.

B.3. FAST

FAST (Features from Accelerated Segment Test) is a corner detector [18,19], designed for use in real-time tracking systems. The detector performs a very simple test for each pixel *p*, by examining a circle of 16 pixels around the pixel. A corner is detected if at least n=12 contiguous pixels (dubbed FAST-12) are all above or all below the intensity of p by some threshold t. Candidate pixels can be rejected early by testing first pixels 1, 9, and after that 5 and 13. Three of those four pixels must all be above or below the intensity of p by the threshold. For the found corners, the intensities of the 16 pixels are used as the feature vector.

Several additional optimizations are performed in the matching step. As a speedup, 'positive' corners (circle pixels higher intensity than center pixel) are not compared with 'negative' corners. To prevent the $O(n^2)$ cost of comparing all corners with all corners, the feature vectors are sorted by their mean. To find a match for a corner, a linear search is started from the feature vector with the closest mean, which is found using binary search. The matching itself is done by calculating the sum of squared distances (SSD). As a speedup, the feature vector is transformed to compact most of the energy into the first few elements; this allows early rejection on the partial SSD. The transformation is done using the Haar Wavelet Transform. The linear search can be aborted because the following bound exists:

$$SSD(f_1, f_2) \geq l(\overline{f_1}, \overline{f_2}) \tag{29}$$

where l is the size of the feature vector, f_i is a feature vector and $\overline{f_i}$ is its mean. The best match found, is picked as correspondence. The authors note [19] that while FAST performs comparable to more complex interest point detectors like SIFT, FAST is relatively sensitive to noise.

A variation called FAST-9 [36] changes the corner criteria to only n=9 contiguous pixels and uses machine learning to adapt the order of pixel testing to the distribution of corner appearances. To this end the detector can be trained on images from the target application domain. Both the speed and repeatability of FAST-9 outperforms the original FAST-12. Also non-maximum suppression is used to eliminate adjacent features. A subsequent extension called FAST-ER [36] generalizes FAST-9 even more.

B.4. A-SIFT

Methods like SIFT and SURF normalize the translation and rotation component and simulate the scale (zoom) through image pyramids to obtain an description invariant to these parameters and partially invariant to affine transformations. A recently proposed method ASIFT (Affine SIFT) [20,21] attempts to obtain a description fully invariant to affine transformations. The method simulates all image views obtainable by varying the latitude and longitude camera angles. Next the resulting views are compared using the normal SIFT algorithm.

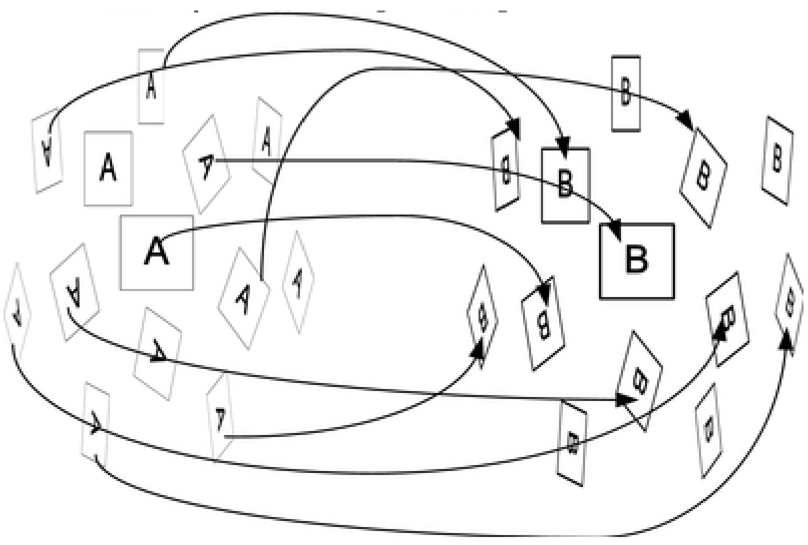


Figure 37 – Overview of ASIFT algorithm. Simulated views of both source images (A and B) are described and matched using SIFT.

B.4.1. Algorithm

Each input image is transformed by simulating a finite and small number of affine tilts (t = $t = 1/|\cos\theta|$, with θ being the latitude) and rotations (longitude ϕ)[1]. The sampling steps are distributed unevenly over the parameter space (see Figure 38) and chosen such that more samples are taken at higher tilts, as any displacement (θ or ϕ) introduces more image distortion at higher values of θ.

Given the two parameters (θ, ϕ) the view is simulated using a rotation followed by directional t-sub sampling (combined with a Gaussian convolution to prevent aliasing artifacts).

In all simulated images SIFT keypoints are then detected and described (using the standard SIFT descriptor, see B.1.2). Keypoints too close to the border of the simulated view are removed.

B.4.2. Matching

To match two images using ASIFT, the steps described above must be performed for each image. For all tilts and rotations of image 1, the matching loops over all tilts and rotations of image 2 and the normal SIFT matching strategy is used per image pair. Any identical matches (same coordinates) are removed and any many-to-one matches are removed as well. As the matching must happen for

[1] Two type of rotations have now been introduced. The former is a rotation around the optical axis, eg normal image rotation. The latter introduced here is somewhat similar to a yawing movement around the object and depending on the tilt can change the perspective considerably, see Figure 39.

all simulated views, the matching complexity of ASIFT is 180 times that of SIFT. To decrease the computational burden, a coarse-to-fine acceleration strategy is proposed.

Finally false matches are eliminated using epipolar geometry filtering. This is done using the ORSA algorithm, which works similar to the RANSAC outlier removal strategy used in this work. Note that this step is disabled as it takes considerable time and a similar outlier removal technique is already present in this work.

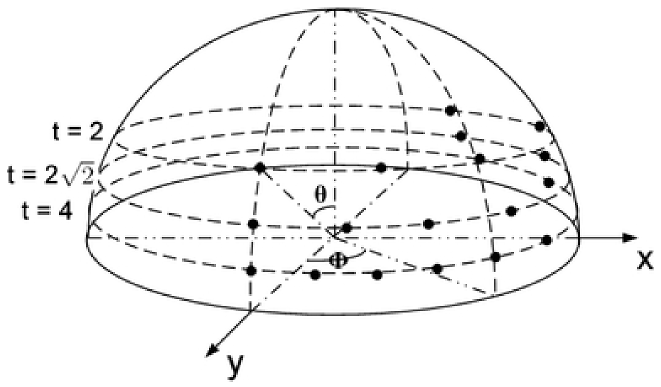


Figure 38 – Parameters space of latitude θ and longitude ϕ. The black dots show the values for which a view is simulated.

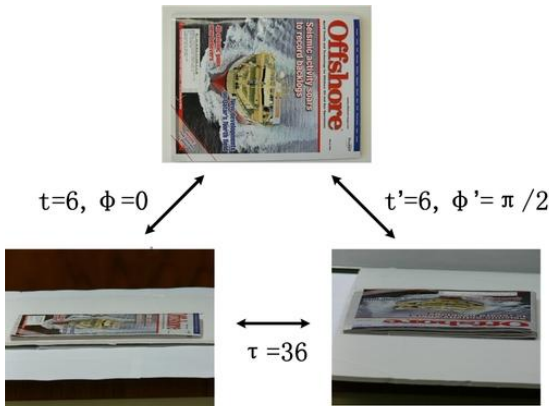


Figure 39 – The bottom left image is a tilted variant of the top image. The bottom right image is both tilted and rotated.

OTESC: Online Transformation Estimation between Stereo Cameras

Tiemen Schreuder
Delft University of Technology
Department of Mediamatics
Delft, The Netherlands
tiemenschreuder@gmail.nl

Emile A. Hendriks
Delft University of Technology
Department of Mediamatics
Delft, The Netherlands
E.A.Hendriks@tudelft.nl

André Redert
Philips Research Europe
Eindhoven, The Netherlands
Andre.Redert@philips.com

ABSTRACT

In this paper, we describe a system that performs online transformation estimation between pre-calibrated stereo cameras. This allows the stereo cameras to be moved around and automatically re-calibrated without the use of a calibration object. This also allows the set-up to recover from accidental nudges that invalidate the extrinsic (external to the stereo camera) calibration. The obtained transformations can be used in virtual view rendering for 3D Video.

The relative positions and orientations of the stereo cameras are obtained using sparse point correspondences found in different views of the scene. For each stereo camera, 3D coordinates of salient scene points are triangulated and their image feature descriptors are used to locate the same points in the views of other stereo cameras. The salient point descriptors SIFT and SURF are evaluated for this purpose.

Given enough salient image points, the proposed solution accurately finds the transformation between stereo camera pairs with a reprojection error less than 1 pixel.

Categories and Subject Descriptors: I.4.1 [**Image Processing and Computer Vision**]: Digitization and Image Capture—*Camera calibration*; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—*Stereo*

General Terms: Experimentation, Performance, Verification

1. INTRODUCTION

With recent blockbusters such as *Avatar* and *Alice in Wonderland*, 3D video is gaining more and more momentum. At the same time 3D-ready television sets slowly become available on the consumer market. Most of the current 3D material is actually stereoscopic; a different image for each eye. More advanced 3D video and 3D video equipment can provide an even stronger illusion by presenting a view based on the position of the viewer. If the viewer moves, the perspective of the 3D video changes as well.

One way to obtain this type of 3D video, is to place several cameras at fixed positions around the scene and use their images to interpolate the views from other positions. We refer to this as virtual view rendering. Knowing the relative position and orientation of cameras with respect to each other enables to extract the scene / geometry by relating all camera images. The scene model can subsequently be used to render new images from virtual cameras at arbitrary positions. The accuracy of the found transformations directly affects the quality of the virtual views.

Using stereo cameras instead of ordinary cameras has several advantages. Since the baseline between the two individual views of each stereo camera is known (in meters), depth triangulation is relatively easy and all scene coordinates and transformations can be determined in meters, which is not trivial when using a single camera setup.

For the purpose of this paper, we divide the (stereo) camera parameters in two types: *stereo-intrinsic* parameters include both parameters intrinsic to the individual cameras (focal length, lens distortion, central pixel, etc) and extrinsic parameters between the two individual cameras; the stereo baseline etc. The parameters extrinsic to a stereo camera rig, so its relative position and orientation, are referred to as *stereo-extrinsic.*

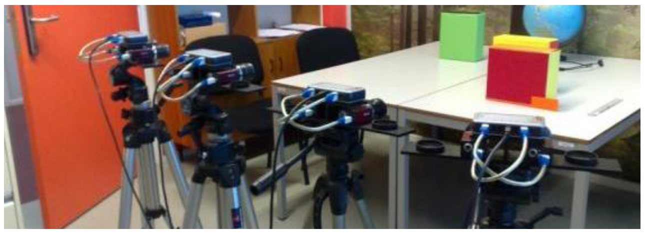


Figure 1: Four stereo cameras capturing a scene.

There are two main approaches towards camera calibration. In the first approach a calibration object with known geometry, for example a plate with a checkerboard pattern, is captured by all cameras and camera parameters are computed such that they are consistent with both the known geometry and the image projections of the object. This typically results in a highly accurate calibration [14, 13, 12]. A disadvantage of this approach is the required manual interaction (placing a calibration object in the scene, usually in different orientations), which also means the normal recording, if any, must be interrupted.

The second approach aims to overcome these problems by performing calibration using only the scene information available in the images taken by the cameras. This approach is commonly referred to as self-calibration, or online calibration. For many applications this self-calibration is implemented as a structure from motion (SfM) approach [5], which simultaneously finds the 3D scene structure and all camera parameters by analyzing the motion of objects across views and/or over time. As SfM approaches often take a significant amount of time, it may be more efficient to perform only a partial calibration in systems that require periodic re-calibration. For example, in [11] the intrinsic parameters are supposed to be known a priori and in [8, 10], specifically aimed at stereo cameras in distributed 3D visual sensor network, the *stereo-intrinsic* parameters are assumed known and fixed, leaving only the *stereo-extrinsic* parameters to be estimated.

In this paper we propose an online transformation estimation approach (dubbed *OTESC*) which is able to quickly recover the *stereo-extrinsic* parameters without manual interaction with the scene. This allows for a flexible recording environment and guards against accidental invalidation of the calibration, for example if someone nudges a stereo camera rig.

The paper is organized as follows: In section 2 we discuss some preliminaries, specifically the used camera set-up and the required pre-calibration. Section 3 gives a detailed description of the proposed approach. Results are provided in section 4.

2. PRELIMINARIES

2.1 Camera Set-Up

Our 3D Studio consists of four stereo cameras (see figure 1), each consisting of two FireWire AlliedVisionTec Marlin F-046C[1] cameras bolted on a rigid frame, which in turn is mounted on an ordinary camera tripod. The cameras are connected with a PC through a FireWire hub mounted on top of the cameras (see figure 1) and provide 640x480 images. The stereo baseline is adjustable, but was fixed at 10.4cm for all experiments.

2.2 Pre-Calibration

The proposed system works with pre-calibrated stereo cameras. We make the assumption that moving stereo cameras does not change the stereo-intrinsic parameters, given that the cameras are mounted tightly on a rigid frame. Therefore a stereo camera only has to be calibrated once. The obtained calibration data should remain valid as long as the stereo camera rig is not modified (such as changing the focal length or stereo baseline). In practice the stereo-intrinsic calibration may also become invalid due to external factors such as temperature changes, so we recalibrate periodically.

All pre-calibration data for experiments in this paper was obtained with a manual calibration technique using a calibration plate with several dots, as described in more detail in [12].

3. APPROACH

The proposed approach can be summarized as follows (see figure 2):

- Detect salient image points in each view
- For each stereo camera, establish image point correspondences between the stereo views and triangulate their 3D coordinates

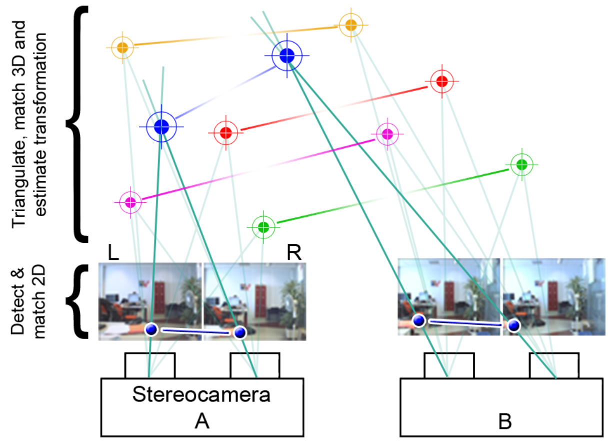


Figure 2: From bottom to top the steps of the proposed approach are visualized. First the salient image points are detected and matched between stereo views. The 3D positions of their scene points are triangulated and again matched, now with points from a second stereo camera. Finally the transformation that aligns the two point clouds is estimated.

- Match 3D points between a pair of stereo cameras using original image point descriptor
- For each pair of stereo cameras, estimate transformation from RANSAC-selected (3D) correspondences
- Resolve transformations between all stereo cameras

3.1 Salient Image Points

Each stereo camera simultaneously takes an image of the scene from each view, resulting in two images per stereo camera. The first step in OTESC is to find salient image points that can be uniquely matched in these two images. Two techniques were considered for this step. Scale-Invariant Feature Transform (SIFT) [9] and Speeded-Up Robust Features (SURF) [2] are both algorithms that find and describe salient image points, at subpixel accuracy, such that they can be robustly matched with other salient points.

SIFT is commonly used and SURF is partly inspired by SIFT, aimed at providing similar quality features in less time. Both descriptors are insensitive to scale and SIFT is invariant to small affine transformations.

The implementations used are [6] for SIFT and the OpenSURF library [3] for SURF. The parameters were set to the author's recommended settings, except that for SURF the Upright-SURF variant was used. Upright-SURF is quicker, but only orientation invariant up to about 15°[2], which is sufficient for our purpose. In section 4.1 a comparison between the two algorithms, in terms of accuracy of the proposed solution, is given.

3.2 Stereo-Intrinsic Matching

For each stereo camera, salient point detection and description is applied on both views (L and R) and sparse point correspondences are established between the two views.

SIFT and SURF use a similar matching scheme; matching happens not based on the distance between two feature vectors, but on the ratio between the distance with the best and second-best match. If an image contains repeating patterns for example, there are many valid candidates and the matching would be ambiguous. The ratio ensures that the correspondence is only accepted if it is unique among all candidates.

Ideally, the corresponding image points (i, i') found $(i \in L, i' \in R)$ have originated from a single 3D scene point x. Since the stereo-intrinsic calibration parameters are known, it is possible to triangulate the 3D coordinates of x relative to the stereo camera. For both individual cameras, a ray is traced from the optic center through the image plane (at the image point coordinates), see figure 2. The rays should converge near scene point x. If the distance between the two rays at the point of closest convergence is above a threshold g, the correspondence (i, i') is discarded, as this would indicate that the coordinates of the two salient points were inaccurate, or did not originate from the same point x.

We associate point x with the salient point descriptors of both image points for further matching with other stereo cameras. After this stereo-intrinsic matching each stereo camera has a list of scene points with each two associated image descriptors (L, R).

3.3 Stereo-Extrinsic Matching

For a *pair* of stereo cameras (A, B) the transformation between them is obtained by first establishing correspondences between the scene points. The matching is performed for all four combinations of the two associated descriptors for each point: $(match(L_A, L_B), match(L_A, R_B)$, etc). This is repeated inversed: points from B are matched with A, resulting in a total of eight matching runs. These eight runs typically result in 2 to 3 times more matches (after removing duplicates) than a single run.

Scene points that have been matched are ideally the same world point seen from two different viewpoints. As the matching process is based on photometric consistencies, it will likely produce physically incorrect matches. Therefore the transformation estimation must be robust to outliers; see the next section.

3.4 Transformation Estimation

The (affine) transformation T between the two matching point clouds is calculated using the Absolute Orientation algorithm [7]. This algorithm requires a minimum of four 3D correspondences and estimates the rotation and translation required to align the clouds, represented as:

$$T = \begin{pmatrix} R & t \\ 0\,0\,0 & 1 \end{pmatrix} \tag{1}$$

where R is a 3x3 rotation matrix and t is a translation vector.

The transformation T is estimated such that for each correspondence (x, x'), ideally:

$$x' - Tx = 0 \tag{2}$$

with x and x' in homogeneous coordinates.

To cope with noise and incorrect matches, a RANSAC [4] scheme is employed. During each RANSAC iteration a random subset of four correspondences is selected and the transformation T is estimated. Remaining correspondences (x, x') are considered in-consensus, and added to the consensus set C, if their correspondence is adequately modeled by the found transformation. This is formalized in section *Consensus Criterion.* If C consists of at least 50% of all correspondences, or at least 20, the error on all inliers of the found transformation is evaluated (see section *Transformation Error*). The entire process is repeated a fixed number of iterations (k=10000), and after each iteration the best transformation is kept. It is possible that no transformation is found at all, because not enough correspondences are found or in-consensus.

Consensus Criterion

To determine if a correspondence is correctly modeled by the transformation T under consideration, an error measure is calculated. Two different measures were evaluated; the *3D Error* and the *Reprojection Error.*

- **3D Error** The *3D Error* is given by the (3D) Euclidean distance between x' and its estimate Tx, thus:

$$3DE = |x' - Tx| \tag{3}$$

- **Reprojection Error** The *Reprojection Error* is given by the (2D) Euclidean pixel distance between the projections of x' and estimate Tx on the image plane:

$$RE = |proj(x') - proj(Tx)| \tag{4}$$

  where $proj()$ projects the 3D point onto the right-most image plane of the stereo camera associated with x'. Since the stereo-intrinsic calibration parameters are known, this projection can be directly calculated.

To use these error measures in the RANSAC scheme, a threshold is applied of respectively q pixels for the *Reprojection Error* and r meters for the *3D Error.* Correspondences with an error below this threshold are regarded as in-consensus with the current transformation. See section 4.2 for a performance comparison between the two error measures, and the chosen thresholds.

Note that contrary to the original RANSAC algorithm, the model is not re-estimated from the entire consensus set C when using the *Reprojection Error.* Instead the transformation found from the original four points is kept, as this gives better results. An explanation could be that the Absolute Orientation algorithm, due to its nature, minimizes the *3D Error* not the *Reprojection Error* used as evaluation. When using the *3D Error*, re-estimation does lead to a smaller average error, which is expected.

Transformation Error

The total transformation error is given by averaging the consensus error (either *3D* or *Reprojection Error*) over all correspondences in the consensus set C. This error gives an indication of the average error the transformation makes on correct correspondences. We use this error as the fitness measure for RANSAC.

3.5 Multi Camera Setup

The procedures described above determine, between a pair of stereo cameras (s, s'), the transformation $T(s, s')$. We will now extend this procedure for set-ups with more than two stereo cameras. Let S be the set of all stereo cameras, so $s, s' \in S$. Let $C(s, s')$ be the number of correspondences found between s and s'. The goal is to find the position and orientation of each s relative to a common origin O (one of the stereo cameras, $O \in S$). We will differentiate between direct transformation estimation (DTE), which is the technique described in the previous sections, and indirect transformation estimation (ITE), which is described in this section.

A reasonably safe assumption is that if two views have more overlap, more correspondences can be found and our approach is better at finding an accurate transformation estimation. As perhaps not all pairs (s, s') have overlapping views, and some pairs have more overlap than others, there are different strategies possible in finding all transformations. Since no knowledge is yet available about the relative positions and orientations of the stereo cameras, the number of matches $C(s, s')$ is used as an indicator of view overlap instead.

We will discuss two simple strategies to deal with multi camera set-ups (see also figure 3). In section 4.3 we will give a numerical comparison between the two.

Minimal Set. Given three stereo cameras {1,2,3}, if the transformation between {1;2} and {2;3} is already known, the transformation between {1;3} can be deduced indirectly. We have implemented this approach as follows. All possible pairs (s, s') are evaluated in descending order of number of correspondences. A graph G is constructed with each stereo camera s as a vertex $V(s)$. Whenever the transformation between a stereo camera pair (s, s') has been estimated, the edge $(V(s), V(s'))$ is added to G. Before performing DTE between each (s, s'), a breath-first search is performed on G to see if there is a path that connects $V(s)$ and $V(s')$. If such a path exists the DTE is skipped; the transformation can already be indirectly determined. The final step is to calculate the missing (indirect) transformations between the origin O and each remaining stereo camera s $(s \neq O)$ by multiplying the previously found transformation matrices along the (shortest) path between $V(O)$ and $V(s)$ (ITE).

Origin. Instead of finding the minimal set of most 'overlapping' pairs, this approach performs a DTE between the chosen O and any stereo camera s $(s \neq O)$, regardless of the number of correspondences between them. If DTE fails for an s, an ITE is performed through a stereo camera s' for which $T(O, s')$ is already known. s' is selected by:

$$\arg\max_{s'}(C(s, s')) \tag{5}$$

In this approach the common origin O, if not given, is chosen by finding the stereo camera s from S, for which holds:

$$\arg\max_{s}(\min_{s', s' \neq s}(C(s, s'))) \tag{6}$$

In words, the minimum number of correspondences a stereo camera shares with any other stereo camera should be the highest of all stereo cameras. This should select the stereo camera which is most 'central' and has enough correspondences with any other stereo camera for a successful DTE. If multiple stereo cameras meet this criterion, the stereo camera with most correspondences overall is chosen.

4. EXPERIMENTAL RESULTS

A total of six image sets, each consisting of images from two stereo cameras, were selected to evaluate the proposed approach, see figure 6. Each image sets has a different background and different foreground objects. All images were shot indoors.

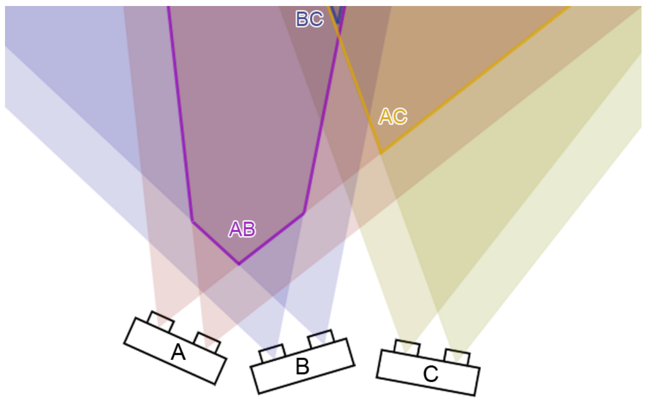


Figure 3: An example multi (stereo) camera setup with 3 stereo cameras (A, B, C) with their respective area of overlap (eg. AB is the area which both A and B see). We assume in this image that the number of correspondences found is directly correlated to the view overlap. The origin is chosen to be B. In the Minimal Set approach, the transformation [B,C] is not estimated directly, but calculated from the transformations [A,B] and [A,C]. In the Origin approach [B,A] and [B,C] are calculated directly.

The distance and orientation between the stereo cameras varied. See table 3 for an indication of the view overlap, in terms of the rough distance between the stereo cameras (specifically between the left view of both stereo cameras).

4.1 SIFT versus SURF

As can be seen in figure 4, SIFT outperforms SURF for all image sets, decreasing the error by 12% to 45%. Image set studio 2 shows an even larger decrease: 80%. It is not clear why the error on this particular image set is much larger with SURF.

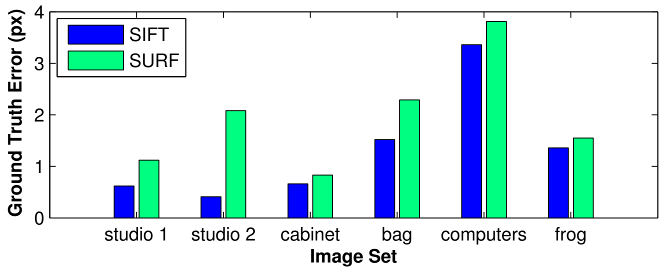


Figure 4: SIFT versus SURF, the y axis shows the average error relative to the ground-truth transformation (see equation 7).

4.2 Reprojection Error versus 3D Error

Although both the *Reprojection Error* and the *3D Error* measure give a good indication of the fitness of a transformation for a given correspondence, the *Reprojection Error* measure gives better results, see figure 5. Only for image set computers the *3D Error* measure gives a slightly smaller error.

Effectively the *Reprojection Error* measure is a weighted version of the *3D Error*, diminishing the effect of errors far away from the camera. This is a desirable property, since we expect the 3D coordinates of the correspondences at those locations to be less exact; there is simply less resolution.

A threshold of $q = 2$ pixels was found to work best on average and was thus used in all experiments. Should no transformation be found using this threshold q, the threshold is automatically relaxed by 1 px, at most 3 times. In other words, if $q = 2$ px provides no solution, the estimation is performed again with subsequently $q = 3$ px, 4 px, up to 5 px, until a solution is found.

For the *3D Error*, a threshold of $r = 0.4$ mm (relaxed to 1.4 mm, 2.4 mm, 3.4 mm if needed) was used for the 3D Error criterion.

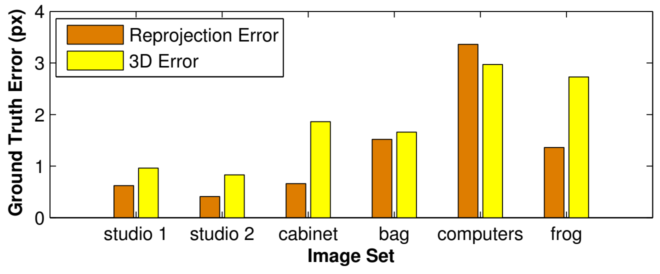


Figure 5: Reprojection Error versus 3D Error as RANSAC criterion, the y axis shows the average error relative to the ground-truth transformation (see equation 7).

4.3 Multi Camera Results

Two different strategies for set-ups with more than two stereo cameras were suggested. In the ideal case, both perform $n-1$ estimations for n stereo cameras. The goal is to find the position and orientation of each stereo camera relative to a common origin. The main difference between the two strategies is that with the **Minimal Set** strategy many transformations relative to the origin are calculated using ITE from strong DTEs, however this may cause errors to propagate. For the **Origin** strategy they have been calculated with DTE, however potentially with very few correspondences.

Image Set	DTE matches	ITE matches	DTE vs ITE
a	44	89 + 97	159.2%
b	10	45 + 38	34.0%

Table 1: Multi-view results for two image sets, each with three stereo cameras 1,2,3, DTE is compared with ITE to find T(1,3). The second column is the number of correspondences available for DTE of T(1,3). The third column is the number of correspondences available for ITE: T(1,2)*T(2,3)=T(1,3). The last column shows the average reprojection error (equation 4) ratio $\frac{RE_{DTE}}{RE_{ITE}}$.

We take a very simple case to evaluate the two strategies. Given three stereo cameras 1,2,3, assume 1 is designated as the origin and there are very little correspondences for pair 1,3 and many between the other two pairs. T(1,3) is required. It must be verified which is more accurate: DTE from the few correspondences between 1,3 or ITE using 1,2 and 2,3 with many correspondences.

In table 1 the results from two multi-camera image sets (with each three stereo cameras) are shown. It is clear that more experiments are required, but these preliminary results support the hypothesis that DTE will outperform ITE if enough correspondences are available, but otherwise ITE will give better results. With more experiments a strategy can be developed which makes an informed choice between DTE and ITE for specific cases.

	Translation Error (cm)	Rotation Error (degr)	Reprojection Error (px)
studio 1	0.74	0.33	**0.62**
studio 2	3.43	2.16	**0.41**
cabinet	8.27	1.10	**0.66**
bag	4.56	0.98	**1.52**
computers	4.40	1.97	**3.36**
frog	0.84	0.84	**1.36**

Table 2: Results of OTESC's performance on several different image sets, when compared to the ground-truth obtained in the pre-calibration step. The translation error (equation 8) and rotation error (equation 9) are given for illustration purposes, see text. The consensus reprojection error is the main evaluation criterion (equation 7).

4.4 System Performance

In the previous subsections the combination of SIFT and the Reprojection Error measure turned out to be the strongest combination and the best choice for our approach. Table 2 lists the results of our approach on several image sets and table 3 gives some statistics for each set, such as the number of matches and processing time. The stereo cameras were pre-calibrated using the technique described in section 2.2. The pre-calibration also provides a ground-truth transformation (T_{GT}) between the stereo cameras, which is used to evaluate our approach. The main evaluation criterion is calculated as follows: For each (x, x') in the inlier set C selected by the RANSAC scheme, calculate the difference and average:

$$\frac{1}{|C|} \sum_{(x,x') \in C}^{|C|} |proj(T_{GT} * x) - proj(T_{OTESC} * x)| \qquad (7)$$

where T_{OTESC} is the transformation found by our approach and $proj()$ is the pixel projection described in section 3.4. Note that the point x' is not required for this error measure.

As can be seen in table 2, the relative position and orientation sometimes deviate significantly from the ground-truth transformation. It should be noted that both the OTESC transformation and the ground-truth transformation are not necessarily physically correct; they are geared towards minimizing a reprojection error. Therefore the deviation in translation and rotation is purely given for illustration purposes. If a transformation T is represented as a translation component t and the rotation R in Euler angles (α, β, γ), then the translation error is given by the Euclidean Distance:

$$|t_{GT} - t_{OTESC}| \qquad (8)$$

	Avg points per view (1)	Avg matches stereo-intrinsic (2)	Matches stereo-extrinsic (3)	Consensus matches (4)	Distance stereo cameras (cm) (5)	CPU Time (sec) (6)
studio 1	511	153	44	25	25	3.94
studio 2	618	189	29	14	25	4.00
cabinet	1237	321	35	25	35	5.93
bag	625	113	20	13	25	3.92
computers	785	253	22	11	55	4.46
frog	783	219	47	21	50	6.54

Table 3: Statistics related to the image sets. (1) gives the average number of salient image points found per view. (2) gives the average number of correspondences found within the stereo cameras, that remain after matching and triangulation. (3) is the number of (3D) matches found between the two stereo cameras and (4) shows how many of those matches were considered in-consensus with the final result (on average). (5) gives the (rough) distance between the (left view of the) stereo cameras. (6) shows the processing time in seconds it took to compute an estimate (3.4Ghz Intel Core2Duo, multi-threaded).

and the rotation error is given by:

$$|\alpha_{GT} - \alpha_{OTESC}| + |\beta_{GT} - \beta_{OTESC}| + |\gamma_{GT} - \gamma_{OTESC}| \quad (9)$$

In general the results are good for the six image sets and the average error smaller than 1 pixel is certainly desirable for virtual view rendering. Note that for the approach to work well, enough salient points should be detected which could constrain the type of scenes. Specifically it helps if the scene contains sufficient *interesting* objects and with little repetition of scene elements.

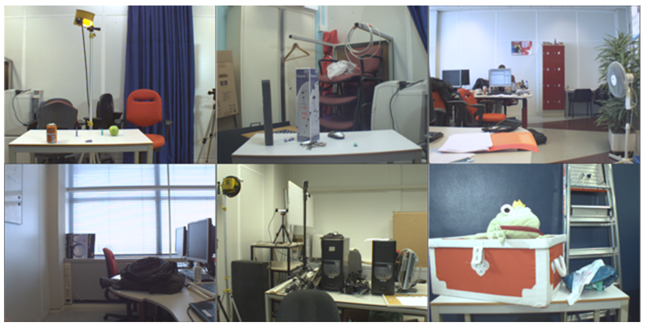


Figure 6: $\begin{bmatrix} a, b, c \\ d, e, f \end{bmatrix}$ **From each of the six image sets used for evaluation, a single image is shown. In alphabetical order (a - f) the image sets are *studio 1*, *studio 2*, *cabinet*, *bag*, *computers* and *frog*.**

5. CONCLUSIONS

The goal of the proposed system described in this paper is to calibrate multiple stereo cameras, for example to be used for virtual view rendering in 3D video. We have looked at several combinations of algorithms and error measures and showed that SIFT combined with the Reprojection Error criterion performs best. We demonstrated our system on a number of image sets and for several image sets we obtain a transformation that differs from the ground-truth with a reprojection error of less than 1 pixel, which would make the found transformation suitable for application in virtual view rendering. From the results it becomes clear that a larger distance between the stereo cameras generally leads to less consensus matches and to a larger error. Especially in the matching between distant stereo cameras there is still room for improvement.

6. FUTURE WORK

Finding corresponding points between stereo cameras is a difficult task. One approach worth investigating is a multi-pass approach, in which the transformation is first estimated using the technique described in this paper, and then re-estimated while discarding matches between stereo cameras that are not reasonably consistent with the initial transformation found.

Another important step could be to investigate salient point matching algorithms other than SIFT and SURF, specifically algorithms more focused on matching between distant views.

7. REFERENCES

[1] Allied Vision Technologies. Marlin F-046C Specs.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 2008.

[3] C. Evans. Notes on the OpenSURF Library, 2009.

[4] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.

[5] Y. Furukawa and J. Ponce. Accurate Camera Calibration from Multi-View Stereo and Bundle Adjustment. *International Journal of Computer Vision*, 2009.

[6] R. Hess. SIFT Feature Detector, impl. C++, 2010.

[7] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629, Apr. 1987.

[8] J. Jannotti and J. Mao. Distributed Calibration of Smart Cameras. In *Proc Intl Wkshp on Distributed Smart Cameras*, pages 56–61, 2006.

[9] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[10] A. Mavrinac and K. Tepe. *Feature-based calibration of distributed smart stereo camera networks.* IEEE, 2008.

[11] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–77, 2004.

[12] A. Redert, J. Biemond, and E. Hendriks. Multi-Viewpoint Systems for 3D Visual Communication. Delft University of Technology, 2000.

[13] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[14] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 00, pages 666–673. Microsoft Research, Ieee, 1999.