Thermal-visible-depth image registration

Long Master's thesis

Chris Bahnsen

Vision, Graphics and Interactive Systems
Spring 2013

Aalborg University
School of Information and Communication Technology
Selma Lagerløfsvej 300
DK-9220 Aalborg Ø

School of information and Communication Technology
Selma Lagerløfsvej 300
DK-9220 Aalborg Ø
http://sict.aau.dk

AALBORG UNIVERSITY
Master's Thesis

Title:
Thermal-visible-depth image registration

Theme:
Master's thesis

Project Period:
9th & 10th semester, 2012 - 2013

Project Group:
13gr1045

Participant(s):
Chris Bahnsen

Supervisor(s):
Thomas B. Moeslund

Copies: 4

Page Numbers: 121

Date of Completion:
June 6, 2013

Abstract:

This thesis deals with the problem of obtaining an accurate registration for objects in multi-modal imagery within the range of 1 – 4 m. The work focuses on providing a registration for all objects inside the range. An acquisition platform is built in both hardware and software that enables the synchronized capture of visual, thermal, and depth images. The depth image is by default registered to the visual image. Three scenes within the range are defined. For each scene, a custom-built thermal-visible calibration rig is used to provide point correspondences between the views. Two rectification algorithms have been investigated and tested by the use of the generated point correspondences; stereo rectification and rectification by multiple homographies. The stereo rectification is however shown to provide poor results. The rectification by multiple homographies is based on training data to generate k homographies which are used to map points between the thermal and visible modalities. This method is shown to provide a better accuracy of mapping visible points to the thermal modality than the baseline of a single homography. However, the method falls short on providing the opposite transfer.

School of information and Communication Technology
Selma Lagerløfsvej 300
DK-9220 Aalborg Ø
http://sict.aau.dk

AALBORG UNIVERSITET
Kandidatspeciale

Titel:
Termisk-visuel-dybde billedregistrering

Tema:
Kandidatspeciale

Projektperiode:
9. – 10. semester, 2012 - 2013

Projektgruppe:
13gr1045

Deltager(e):
Chris Bahnsen

Vejleder(e):
Thomas B. Moeslund

Oplagstal: 4

Sidetal: 121

Afleveringsdato:
6. juni 2013

Synopsis:

Dette speciale tager udgangspunkt i problemstillingen med at registrere multi-modale billedsekvenser indenfor en rækkevidde på 1 – 4 m. Rapporten tager udgangspunkt i at registrere alle objekter inden for dette område. Hertil er der konstrueret en platform i både hardware og software der muliggør en synkroniseret optagelse af visuelle-, termiske- og dybdebilleder. Dybdebillederne er som udgangspunkt registeret til de visuelle billeder. Tre scener er defineret inden for den ønskede rækkevidde. Punktkorrespondancer mellem hver modalitet er genereret ved hjælp af specialkonstrueret kalibreringsudstyr. To registreringsalgoritmer er blevet undersøgt og testet ved hjælp af de genererede punktkorrespondancer: Stereorektificering og rektificering ved flere homografier. Tests viser dog, at stereorektificering giver dårlige resultater. Rektificeringen ved flere homografier bygger på de generede træningsdata, hvorpå k homografier er estimeret. Disse homografier bruges til at overføre punkter mellem modaliteterne. Det viser sig, at denne metode giver en større nøjagtighed ved overførslen af punkter fra visuelle til termiske billeder end referencemetoden, der bruger en enkelt homografi. Imidlertid er metoden ikke i stand til at skabe den samme nøjagtighed ved den modsatte overførsel.

Preface

This Master's thesis concludes the work of the 9th and 10th semester of the Vision, Graphics and Interactive Systems master's programme at Aalborg University. The work is conducted from autumn 2012 to spring 2013.

The work on this thesis originates from a project proposal on 'Multi-modal tracking of People'. The part on registering the multi-modal imagery was originally estimated to take a couple of weeks. However, the process of accurately registering objects in short-range multi-modal imagery was found to be more difficult than the initial estimate and lies as the foundation for this thesis.

The thesis is formed in collaboration with the Visual Analysis of People (VAP) group at Aalborg University, which is collaborating with the University of Barcelona on providing schemes for tri-modal tracking, pose estimation, and activity recognition of people in a number of pre-defined scenes. Prior to doing so, the imagery must be accurately registered. The work of this thesis is defined in the context of providing an acceptable registration performance of the thermal, visible, and depth modalities of these scenes.

I want to express my gratitude to the members of the VAP-group at Aalborg University who have provided me with invaluable assistance through the course of this project. I have especially appreciated the professional discussions of the work with Andreas Møgelmose who has provided his support in numerous occasions throughout the time frame of the work.

Aalborg University, June 6, 2013

Chris Bahnsen
<cbahns08@student.aau.dk>

Reading guide

This report is divided into the following parts: Introduction and analysis, design, verification and conclusion, and appendices.

In the part on introduction and analysis we will focus on properly identifying the problem and the challenges thereof and investigate some of the techniques applied in previous work. At the end of the part we will define the boundaries and requirements of the proposed system.

The second part is dedicated to the design of the prototype, including the design and set-up of hardware, and the design of the algorithms included in the software. The third part focuses on the verification of the system and such, several tests are performed to assess the performance of the system based on the criteria stated in part one. At the end of part three, we will open the discussion on further improving the performance of the proposed system. Part four covers the appendices which are provided to support the contents of part one and two. However, the report might be understood in its entirety without visiting the appendices.

The formulas and equations stated throughout the report rely on the following notation: Matrices are always denoted by using upper-case fonts, such as F. Vectors are written as lower-case, bold letters, whereas scalars are written using regular fonts. The use of an apostrophe behind a vector or a matrix denotes that the matrix or vector belongs to *the other view*. Such is the point $\mathbf{x}$ referring to the position of a point in the first view whereas the point $\mathbf{x}'$ is referring to the corresponding position of the point in the other view.

To support the content of the thesis, a CD is enclosed. On this CD, the reader may find code related to the main problem, as well as several test scripts and additional test data. All references to the code on the CD are done using a CD icon,  , which refers to the position of the content with respect to the root of the CD.

Contents

Part I

Introduction and analysis

Chapter 1

Introduction

The field of computer vision has been rapidly expanding through the past decades as advances in manufacturing processes and sharp decreases in cost has opened the market for everyday use of imaging devices. The fields of active research within computer vision are numerous and includes people detection, security installations, traffic monitoring, and military applications.

There are several methods for segmenting and processing visible light images, including Gaussian models [Stauffer and Grimson, 2000], code-book based models [Kim et al., 2005] and the state-of-the-art ViBe background subtraction scheme [Barnich and Van Droogenbroeck, 2011]. However, these methods rely fundamentally on the quality of the spectral and spatial information of the image such as colour and gradient. These features are dependent on external lighting such as the sun or artificial lighting and as the intensity or the direction of the light fluctuates, the colours and gradients might change significantly. However, if the objects are not immersed in light they are not even visible. The reliance on stable lighting conditions makes it difficult for visible light systems to operate in harsh weather conditions and impossible to operate in the darkness.

In order to overcome the downsides of visual imaging, a new field in computer vision centred on thermal cameras has emerged. The field of thermal cameras has long been limited to military applications due to excessive manufacturing costs, but rapid advances in spatial resolution and sensor cost has opened the field of scientific research within the last decade. Thermal cameras in the mid and long-wavelength infrared range captures the radiation of objects with a temperature between 190 K and 1000 K [Gade and Moeslund, 2013] and are thus perfectly capable of detecting humans if the body temperature is significantly different from the surroundings. The thermal characteristics of a human body is independent of colour, texture, and external lighting and the thermal sensor might even see through material that blocks visible light but lets radiation through. Thermal cameras are presently used in agriculture [Vadivambal and Jayas, 2011], building inspection [Al-Kassir et al., 2005], industrial gas detection, industrial manufacturing, rescue operations, and unmanned aerial vehicles, to name some.

Whereas the thermal camera is superior for detecting humans in a broad range of situations, the nature of the thermal information makes it difficult to use thermal

imaging for person identification and recognition. Compared to visible imagery, thermal images provide fewer features and the thermal appearance might also vary according to the physical condition of the subject. However, when combined, the two modalities might complement each other and provide stable features for tracking and recognition where the separate modalities otherwise will fail. A possible use case is the detection of humans where the thermal imagery is used for robust people detection after which the visible imagery is used for identification of the humans found.

Due to the immanent differences of thermal and visible images and the imaging devices, fusion of the images implies an accurate image registration. In the following chapter, we will review the literature for methods on thermal-visible registration and define the further scope of this project.

Chapter 2

Preliminary analysis

In this chapter, we will give an introduction to the current state-of-the-art on thermal-visible registration as well as the limitations of those. At the end of the chapter, we will provide a problem statement for the further work of this report.

Thermal-visual image fusion is broadly classified in three categories: *pixel-level fusion*, *feature-level fusion*, and *decision-level fusion*. Each level of fusion requires different levels of accuracy of the thermal-visible registration and while all fusion schemes benefits from an accurate registration, the pixel-level fusion requires, naturally, a very accurate pixel-to-pixel level registration. The image registration methods mentioned below use their registration schemes for either blob-based pixel-level or feature-level fusion.

2.1 Methods on thermal-visible registration

The field on thermal-visible registration has been actively researched within the last decade and though a lot of interesting methods have been presented, a single 'gold standard' for accurately registering video frames from the thermal and visible modalities is yet to be presented. As the visible and thermal modalities have immanently different properties, performing an accurate registration at every point in the 3D plane has shown to be a difficult task, according to the current literature on the matter.

Articles from Krotosky and Trivedi [2007] and Zhao and Sen-ching [2012] have provided a thorough survey of current techniques on the registration of the modalities. Krotosky and Trivedi [2007] divides the research into categories based on whether or not the registration includes 3D information of the scene, the registration method, calibration measures, and the assumptions necessary for the registration to work. Zhao and Sen-ching [2012] divides the literature by the method of performing the 'geometric fusion' between the thermal and visible modalities. In this overview, we will use their notation. Zhao and Sen-ching [2012] distinguish between *optical fusion*, *alignment assumed*, *image warping*, *3D reconstruction*, and *blob homography*. The methods are summarized in Table 2.1. While the former notations are widely used in the literature, the novel notion of 'blob homography' covers a variety of methods

for cross-image rectification that are based partially on the correction of the epipolar lines in the modality images and a subsequent search for corresponding blobs in order to compensate for the remaining disparity.

Geometric fusion	Alignment level	Accuracy
Optical fusion	Pixel	Low-medium
Alignment assumed	Pixel	High
Image warping	Frame	Low–High
3D reconstruction	Pixel–blob	Medium
Blob homography	Blob	Medium–High

Table 2.1: Thermal-visible camera registration methods, overview [Zhao and Sen-ching, 2012]

In the following, we will give a brief introduction to the methods listed.

2.1.1 Optical fusion

In optical fusion, a beam-splitter is used for feeding both the thermal and visible camera with light. In this way, the displacement of the sensors is made negligible, and both sensors will see the scene from the exact same positions. Differences in field-of-view and lens properties of the visible and thermal cameras will entail inevitable differences in the images of the two modalities, though.

Optical fusion is used by Ó Conaire et al. [2005] who use a standard-window glass for splitting the light beams, as seen from Figure 2.1. The clear advantage of this approach is that the need for geometric fusion is very sparse, and in the work of Ó Conaire et al. [2005], the image registration is accomplished by using a simple planar homography. However, the performance of this registration algorithm is not evaluated quantitatively and only qualitatively via visual inspection. The downside of optical fusion, though, is the amount of light absorbed by the beam splitter. The absence of much of the light reduces the contrast of the thermal image [Ó Conaire et al., 2005].

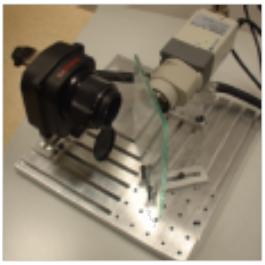


Figure 2.1: Beamsplitter camera rig. Courtesy of [Ó Conaire et al., 2005].

2.1.2 Alignment assumed

This method is fairly simple - by simply assuming that the image alignment is one-to-one, no image registration is needed. This assumption is used where scenes are pre-registered and is thus of no interest to us.

2.1.3 Image warping

The techniques of image warping use manually extracted corresponding points from both modalities to construct either a planar or infinite homography for image rectification. This requires the important assumption, however, that the scene is planar, which means that the difference of the distances of moving objects in the scene are much smaller than the distances of the scene from the camera. However, when this is the case, a accurate pixel-to-pixel correspondence is achieved. The actual accuracy of this mapping is hard to assess, though, as the authors performing image warping do not present any quantitative measures on the performance of the registration. Image warping with the above limitation of inter-object distances is presented by Krotosky and Trivedi [2007] as "Global registration", whereas the image warping used on long-distance imagery is using an infinite homography. More on this in Figure 2.2.

The majority of research on thermal-visible registration, tracking, and image fusion use the image warping technique, such as Davis and Sharma [2007]. The widespread use of the thermal-visible datasets from OTCBVS [Davis and Sharma, 2005] encourages the use of an infinite image homography, as the data set contains surveillance data from a high altitude, thus providing a good ratio of the distances between the camera and the scene and the inter-object distances of the scene.

Without using a homography, the work of St-Laurent et al. [2010] uses a cascade of geometrical operations to register imagery at a fixed distance to the scene. Instead of using a homography, the authors use a cascade of operations to geometrically correct the registration at a fixed distance, compensate for differences in Field of View (FOV), and tilt angle.

2.1.4 Blob homography

The techniques of "blob homography" use either a calibrated approach with chessboards [Krotosky and Trivedi, 2007], an uncalibrated stereo rectification [Zhao and Sen-ching, 2012], or a depth-induced V-disparity transformation [Bertozzi et al., 2006] for pre-registering the images. The paper of Krotosky and Trivedi [2007] use a promising disparity voting algorithm based on the dual input of the two cameras. Bertozzi et al. [2006] use stereo pairs of thermal and visible cameras to detect pedestrians by performing a V-disparity computation for each modality and doing a subsequent bounding-box based search to register the pedestrians individually. Zhao and Sen-ching [2012] use the silhouettes of the detected blobs to perform the registration.

All three methods above rely on the assumption that foreground objects in the thermal image are substantially hotter than the background and the methods may only be used to detect objects fulfilling this assumption, thus excluding a pixel-wise correspondence of every object in the scene.

The performance of "blob homography" image rectification systems are provided by most authors and range from $95 - 99.4$ % accuracy [Krotosky and Trivedi, 2007], and a false positive error of 0.19 and a false negative error of 0.01 [Zhao and Sen-ching, 2012]. In all systems, the results are presented as contour overlays on colour images. For the naked eye, the results here seems to be accurate on a blob-to-blob level. Is is hard, though, to provide any measure on the exact pixel-to-pixel correspondence.

Other approaches to blob registration includes registration based on optical flow [Zhang et al., 2012] and silhouette tracking [Torabi et al., 2011].

2.1.5 3D reconstruction

3D reconstruction is a road to image registration that is substantially less travelled than image warping techniques or blob homography. Johnson and Bajcsy [2008] and Lee et al. [2009] use a set-up of tree grey-scale cameras, one RGB camera, and one thermal camera. The authors perform a calibration of both modalities to extract internal and external camera parameters. From these parameters, the authors project the RGB points to world coordinate space and from there to the coordinate space of the thermal camera. Although the authors provide a pixel-wise mapping of the thermal and visible images, the only quantitative measurement of the papers is the measurement of the numbers of pixels that were correctly classified as foreground/background, thus not containing any explicit data of the quality of the image registration.

The work of Krotosky and Trivedi [2008] use a dual stereo configuration of two thermal and two visible light cameras to construct trifocal tensors relating two cameras of one modality to one camera of another modality. With the trifocal tensor, an accurate image registration is performed. The trifocal tensor is the generalisation of the fundamental matrix when used with a trinocular set-up. The fundamental matrix is explained in greater detail in Section 7.2.

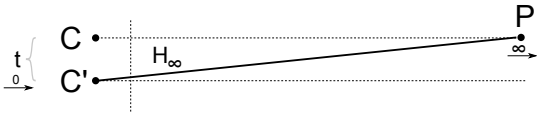
2.1.6 Geometric interpretation

The notions of Zhao and Sen-ching [2012] may also be presented by their geometric implications as presented originally by Krotosky and Trivedi [2007]. The overview is seen from Figure 2.2.

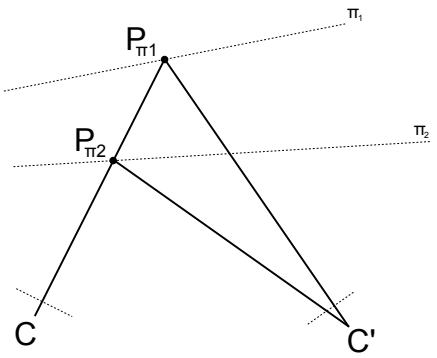
According to Krotosky and Trivedi [2007], the infinite homography and single homography methods are the most commonly used methods on image registration. Only a few authors use the stereo geometry alone, although this method often is a prerequisite for methods on “blob homography” as described above. As seen in Figure 2.2c, the registration for a single homography is only accurate within the exact plane that the registration is done within. The further the world point deviates from the registered plane, the more the parallax increases. This also applies when using multiple homographies, however the image registration is in this case valid for multiple planes.

2.2 Accurate pixel-to-pixel level registration

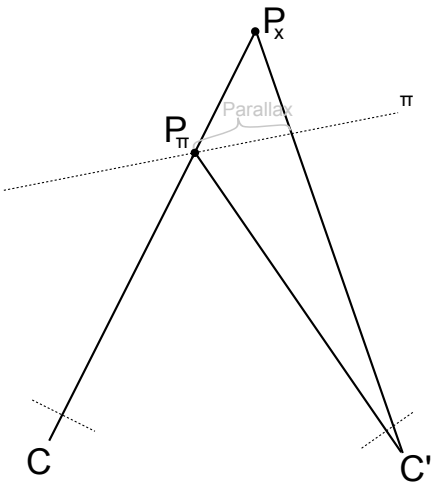
The methods described in Section 2.1 provide viable thermal-visible image registration for a long range of different scenes, differing from short range distance of 1 – 10 m to long-range scenarios with 30 – 50 m from the cameras to the scene. In general, the long-range registration is achieved using an infinite homography with high accuracy. In the short range registration, however, a number of techniques have emerged



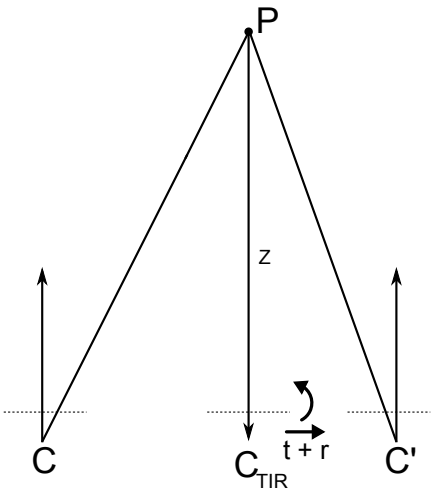
(a) Infinite homography



(b) Multiple homographies - Partial image ROI



(c) Single homography - Global registration



(d) Stereo Geometry

Figure 2.2: Geometric overview of methods for image rectification, inspired by [Krotosky and Trivedi, 2007]. The cameras C and C' are seen from above, with image planes seen as dotted lines and the registered plane of interest noted π. In *infinite homography*, the core assumption is that the distance between the two cameras is minimal and the distance to the scene is large. With a *single homography*, we have registered the scene for a single image plane π only, whereas with *multiple homographies*, we may register the scene at multiple depths or planes - however, as with the single homography, movement outside the plane(s) is subject to a noticeable parallax. *Stereo geometry* uses information of the cameras and the scene to obtain a rectified image for both modalities where the only displacement of corresponding points is found in either the vertical or horisontal direction.

that either limits movement to one or more predefined planes or is reliant on a dual stereo configuration to recreate the 3D structure of the scene in both modalities. The above mentioned methods on "blob homography" does not limit the movement in the plane nor requires a dual stereo configuration. However, these methods are solely restricted to people detection or detection of objects with consistently defined contours in both the thermal and visible modalities. Otherwise, the search for the remaining disparity in one of the image axes are prone to failing.

If one wishes to register thermal-visible imagery for any object in the scene, this is currently only achievable via 3D reconstruction by a dual stereo configuration, which requires a minimum of two thermal and two visual cameras. Previous articles on 3D reconstruction do not provide any quantitative data on the registration accuracy and thus makes it hard to prove the validity of the algorithms used. The field of thermal-visible image registration is still relatively young though, and there might thus exist other registration methods that allows an accurate registration of an entire scene without the use of an expensive dual thermal configuration. This project aims to investigate such methods to thermal-visible image registration.

2.3 Problem statement

Following the above, the goal of this project is to study how we may create a robust, accurate thermal-visible registration algorithm. Unlike many of the algorithms listed above, the registration algorithm of this project should not be limited to registering humans or be limited to registering objects that have a clear distinction in either the visible or thermal modalities and should thus be able to register all objects within a predefined range.

Many of the proposed methods on thermal-visible registration use the properties of infinite homographies due to the fact the scene is located far from the camera. In this project, we do not wish to exploit this condition and thus focus on providing an accurate registration where other methods will induce noticeable parallax. As a result, the algorithm should be limited to registering scenes within the range of 1 – 4 metres from the camera.

Unlike some of the methods presented above, we will not use a dual stereo configuration with two thermal and two RGB cameras. However, we still want to utilize depth information of the scene either via a visible light stereo camera or active infrared sensors such as the Microsoft Kinect or ASUS Xtion. Due to the high cost of thermal cameras, the set-up should only consist of a single thermal camera supplemented by one or more visible light cameras.

Summarized, the project aims to solve the following problem:

- How to develop a registration algorithm for registration of thermal-visible imagery of objects within the range of 1 – 4 m by the use of a single thermal camera and a visible camera configuration providing depth information of the scene.

Based on this problem statement, the following chapter will provide a thorough

review of the resulting requirements and implications for the system to be designed.

Chapter 3

Requirement specification

The previous chapter introduced the notion of thermal-visible image registration and the multitude of methods developed for providing an accurate registration between imagery of the two modalities. In Section 2.3, the problem statement of the project is presented as *how to develop a registration algorithm for registration of thermal-visible imagery of objects within the range of 1 – 4 m by the use of a single thermal camera and a visible camera configuration providing depth information of the scene.*

Based on this problem statement, this chapter will set the requirements in order to create a framework for acquiring and registering thermal-visible imagery.

3.1 Image acquisition platform

An accurate thermal-visible registration requires an acquisition platform capable of capturing and pre-processing the imagery. This section will describe the requirements for this platform, both in terms of the hardware needed for the actual image acquisition and the software needed for the extraction of frames and the temporal alignment of image data.

3.1.1 Synchronization

In order to provide an accurate registration, the imagery of the two modalities must be carefully synchronized in time to make the movement of objects synchronous in both modalities. The difference in time between any two synchronized frames of the imagery should not exceed half of the inter-frame interval of the camera with the slowest frame rate. This is indeed an important criterion since if this is not met, the synchronization algorithm has either skipped frames that fitted better or the synchronization is skewed in time. The property of the half inter-frame interval is seen from Figure 3.1. As a high frame rate is important, we choose a maximum synchronization error of 40 ms, which leads to a minimum frame rate of 12.5 frames per second.

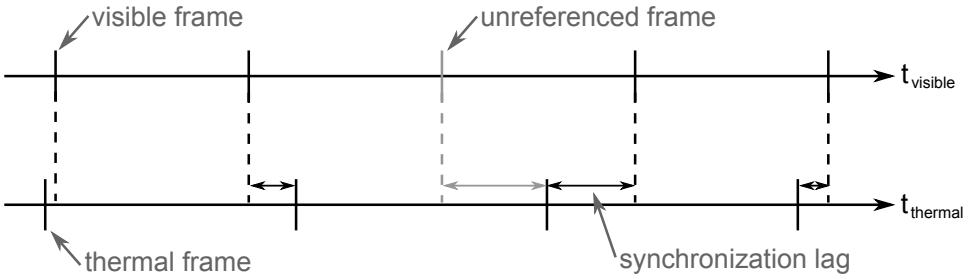


Figure 3.1: Synchronization of frames. The vertical lines show frames for both the visible and thermal camera. In this example, the frame rate of the two cameras are not equal, which means that some frames from the visible camera are left unreferenced to any frames of the thermal camera, and are thus discarded.

3.1.2 Thermal imagery

Thermal images should be provided of the scene by a thermal camera capable of capturing radiation emitted by humans and objects used by humans, which is defined as the range from -10°C to 50°C. The minimum resolution for the thermal images is chosen to be QVGA, 320 x 240.

3.1.3 Visible imagery

Visible images should be provided by a RGB camera which takes images at minimum VGA resolution, 640 x 480.

3.1.4 Depth information

The visible camera(s) should provide depth information of objects placed within 1 – 4 metres from the cameras. The depth of the object should be determined even if the object does not contain any texture information.

3.1.5 Depth registration

The visible cameras(s) should provide a registration between the depth information and the visible image. The acquisition platform should store this registration for further use.

3.1.6 Baseline

The thermal and visible cameras should be co-located and the baseline between the camera sensors should be minimized in order to ease the process of image registration and minimize the effect of image parallax. It is therefore required, that the image sensors of the thermal and visible cameras are not situated further apart than 100 mm.

3.1.7 Summary

The final requirements for the acquisition platform are summarized in Table 3.1.

Parameter	Requirement	Explanation
Synchronization lag	$\leq$ 40 ms	Lag between synchronized frames
Minimum frame rate	$\geq$ 12.5 FPS	Minimum allowed frame rate
Thermal range	-10°C – 50°C	Temperature range visible to the thermal camera
Minimum visible resolution	640x480	Minimum resolution for the RGB camera, in pixels
Minimum thermal resolution	320x240	Minimum resolution for the thermal camera, in pixels
Baseline between cameras	$\leq$ 100 mm	As small as possible, but below 100 mm
Depth information	Available	Available for at least one camera
Depth registration	Stored	The registration between the depth information and the visible image should be provided by the manufacturer and stored during capture
Depth range	1 m – 4 m	Range for which the depth information should be provided by the depth sensor

Table 3.1: Requirements for the image acquisition platform.

3.2 Thermal-visible registration algorithm

The registration algorithm relates objects in the visual image to the same objects in the thermal image and vice versa, thus providing pixel-to-pixel correspondences between the images. As the depth is registered to the visual image by the acquisition platform, we will only define the accuracy requirements for the thermal-visible registration.

3.2.1 Distribution of corresponding point sets

The sets of corresponding points, from which the performance of the registration algorithm should be measured against, should be distributed equally throughout the entire depth range. Validation points are, in this context, individually extracted corresponding points from both the thermal and visible imagery. The exact placement of the corresponding point sets is treated in the acceptance test specification in Chapter 4.

3.2.2 Number of validation points

The number of validation points should not be less than 1000.

3.2.3 Registration range

The registration should work for objects in the scene placed within a distance of 1 – 4 metres.

3.2.4 Accuracy

The accuracy of the registration is the pixel error of registering both visible to thermal and thermal to visible imagery. In the following, we will distinguish between two error measures; the *one-directional transfer error*, and the two-dimensional transfer error, better known as the *symmetric transfer error* (STE) between corresponding points in both modalities.

The one-directional transfer error (OTE) is the sum of squared errors of mapping either visible to thermal points or thermal to visible points, and thus, there exist two OTE's: rgb→t and t→rgb.

The symmetric transfer error (STE) is the sum of both OTE's and thus describes the overall performance of the registration. It is necessary to distinguish between these error measures as the performance of the registration might not necessary be symmetric. It might be that the registration for RGB to thermal points is far superior to the registration for thermal to RGB points, depending on the proposed registration algorithm.

The STE is formalized as:

$$\sum_i d(\mathbf{x}_i, H_{\mathrm{TtoRGB}}\ \mathbf{x}'_i)^2 + d(\mathbf{x}'_i, H_{\mathrm{RGBtoT}}\ \mathbf{x}_i)^2 \tag{3.1}$$

where $\mathbf{x}_i$ is the point vector in the RGB image, $\mathbf{x}'_i$ is the corresponding point vector in the thermal image, H_{RGBtoT} is the homography mapping points from the RGB image to the thermal image, and H_{TtoRGB} is the homography mapping points from the thermal image to the RGB image.

The distance function, d is not explicitly defined and may be set as the function of choice. This work will distinguish between two distance functions, *geometric distance* and *algebraic distance*, which in this context will be defined as:

$$d_{\mathrm{alg,x}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = |x_i - \hat{x}_i| \tag{3.2}$$

$$d_{\mathrm{alg,y}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = |y_i - \hat{y}_i| \tag{3.3}$$

$$d_{\mathrm{geom}}(\mathbf{x}_i, \hat{\mathbf{x}}_i) = \left((y_i - \hat{y}_i)^2 + (x_i - \hat{x}_i)^2\right)^{1/2} \tag{3.4}$$

where $\mathbf{x}$ is the measured image point vector, $\hat{\mathbf{x}}$ is the estimated image point vector, and x_i, y_i and $\hat{x}_i$, $\hat{y}_i$ are the x and y coordinates of the measured and estimated points, respectively.

The definition of the algebraic distance differs from the definition by Hartley and Zissermann [2003], but in this framework, the algebraic distance is used to measure if the mapping error is biased in either the x or y direction. This could also be measured by computing the phase of the geometric error, but is computationally easier to implement by the algebraic error.

Other measures than the STE such as the reprojection error might also be used. The reprojection error estimates 'how much it is necessary to correct the measurements in each of the two images in order to obtain a perfectly matched set of image points' [Hartley and Zissermann, 2003]. This measure is indeed similar to the one of STE, but in this case, we would explicitly like to measure the sum of mapping errors rather than finding the distance to perfectly mapped points. Therefore, the symmetric transfer error is chosen as he measure of choice.

Base-lining the accuracy The actual thresholds for the STE and OTE are hard to set. In general, we would want a zero-error registration between the two modalities such that a unique pixel-to-pixel correspondence indeed does exist. However, an error-free correspondence assumes zero noise which is indeed impossoble and as such, we have to specify exactly what is *acceptable*. The literature on image registration does not provide any quantitative data defining if an error is 'acceptable', and if one wants to set a tolerated error in terms of the OTE and STE, the defined threshold would indeed be arbitrary. Instead, we will choose a baseline, for which the performance of the proposed registration algorithm is measured against.

The baseline chosen for comparing the registration accuracy of the proposed registration algorithm is the global registration technique which registers the images of the two modalities by a single homography. The global registration technique is described in brief in Section 2.1. The global registration technique might be considered as the 'minimum configuration', as the technique is simple, well known, and readily implemented in many frameworks for computer vision. The proposed method should be able to increase the performance of the registration compared to the baseline in terms of lower OTE and STE for all scenes, measured in geometric distance.

3.2.5 Assumptions

It is assumed that the scenes for which the registration should be applied are known prior to any registration. Furthermore, the intrinsic parameters of the cameras should be known. Therefore, in order to provide a valid thermal-visible registration, the scene and the cameras must be calibrated before any image registration may take place. The calibration includes, but is not limited to, estimation of intrinsic camera parameters, undistortion of images, and determination of world points seen through a calibration device.

To provide a reliable depth view of the scene, it is also assumed that scenes are located indoors and under controlled lighting.

If scenes are limited in range, for instance by the placement of furniture, walls, and doors, the requirements for the registration distance and distribution of control points should be limited to reside inside the range of the scene.

3.2.6 Summary

A summary of the requirements for the registration algorithm is listed in Table 3.2.

Parameter	Requirement	Explanation
Geometric OTE, rgb→t	< baseline	The mean and standard deviation of the one-dimensional transfer error when rectifying points from rgb→t, measured in geometric distance, should be lower than the corresponding error of the baseline.
Geometric OTE, t→rgb	< baseline	The mean and standard deviation of the one-dimensional transfer error when rectifying points from t→rgb, measured in geometric distance, should be lower than the corresponding error of the baseline.
Geometric STE	< baseline	The mean and standard deviation of the symmetric transfer error, measured in geometric distance, should be lower than the corresponding error of the baseline.
Registration range	1 m – 4 m	Objects should be registered with the listed accuracy within this range.
Number of validation points	≥ 1000	The number of validation points used for measuring the registration accuracy should be greater than or equal to 1000.
Distribution of validation points	-	The validation points should be distributed equally throughout the entire range of the scene.

Table 3.2: Requirements for the image acquisition platform.

Chapter 4

Acceptance test specification

The acceptance test specification will set the conditions, for which the requirements of Chapter 3 will be evaluated. Consequently, the acceptance test specification will be divided in parts according to the requirement specification.

4.1 Image acquisition platform

The requirements for the image acquisition platform are relatively easy to test, since these typically are based on the detection whether a physical property of the platform is fulfilled or not. The test specification for the image acquisition platform is divided into three parts: the *physical platform*, *synchronization*, and *depth information*.

4.1.1 Physical platform

An acquisition platform is be constructed that meets the requirements of the pixel resolution, minimum frame rate, and the baseline between the camera centres listed in Table 3.1.

4.1.2 Synchronization

In order to measure the performance of the synchronization of frames from the visible, thermal, and depth modality, each frame should be traceable in time. In other words, each frame of every modality is measured against a single clock. This clock might be a system clock of the acquisition platform or a clock of one of the cameras as long as every frame relates to the time of this clock. The traceability of the frames may not be readily available directly after the capture of scenes and should, in this case, be obtained through post-processing of the image stream.

Once traceability is available for every frame of the image streams of the cameras, the requirements of Section 3.1.1 is easily tested by measuring the time distance between synchronized frames against the same clock, and finding the maximum distance in time.

4.1.3 Depth information

As listed in the requirement specification, depth information should be obtained and registered with respect to the visible camera.

The *range* of the depth information is tested by placing an object, a blank A4 paper or equal, in front of the depth camera at the boundaries of the specified depth range, in this case at 1 and 4 metres. The validity of the depth information inside this range is verified by extracting depth information for calibration rigs defined in Section 4.2.2. The accuracy of the depth measurements is a topic for another project and is thus not estimated here. However, the accuracy of most depth sensors is provided by most manufacturers or, in the case of the Microsoft Kinect, available by scientific measurements, such as the one from Khoshelham [2011].

The *registration* of the depth images to the visible image should provide a mapping for all pixels inside the depth image to corresponding pixels of the visible image. If this exists, the criterion on depth registration is fulfilled. The evaluation of the factory calibrated depth registration performance is beyond the scope of this project and will thus not be tested.

4.2 Thermal-visible registration algorithm

The range of the registration algorithm is bounded in space by the requirement specification which also defines that the registration should work for indoor scenarios. This sets some very important delimitations for defining the test scenes for which the thermal-visible registration algorithm should be evaluated.

4.2.1 Defining scenes

Three scenes are defined for validating the registration algorithm; two general scenes and a planar scene. The scenes are defined in a overall context defined by the collaboration with the VAP group at Aalborg University, and the definitions of the scenes are therefore different from one would immediately expect to be defined in the context of thermal-visible registration. For these scenes, the registration algorithm should register humans and objects within the defined range. These scenes will be explained in the following:

Scene 1 takes place in a room bounded by a wall at the maximum registration distance, and a door on the left. A table and two chairs are placed in front of the cameras.

Scene 2 is a planar subset of scene 1, which means that objects are placed within a fixed distance to the cameras. The actual distance to the cameras is irrelevant, as long as objects are placed within the range of 1 – 4 metres. However, in order to obtain a near-planar scene, the objects should be placed within a range of ± 0.25 m from the chosen distance.

Scene 3 is, unlike scene 1, not bounded by walls at the maximum registration distance. The scene contains at least one table, one chair, and one couch.

The definitions of Scene 1 – 3 are listed in Table 4.1.

Scene	Depth range	Open environment	Chair	Table	Indoor
1	1 – 4 m		x	x	x
2	$d \pm 0.25$ m		x	x	x
3	1 – 4 m	x	x	x	x

Table 4.1: Scenes defined for the validation of registration performance.

4.2.2 Obtaining corresponding point sets

In order to measure the registration performance of scenes defined in Section 4.2.1, sets of corresponding points in the thermal and visible images must be defined. These sets should come from a calibration rig from which corresponding points are easily extracted in both the visible and the thermal modality. The nature of the calibration rig is left undefined but should enable the extraction of corresponding points in the thermal and visible images with sub-pixel accuracy. The point correspondences from the calibration rig might also be used for undistorting the imagery from the cameras and estimating the camera parameters.

In order to fulfil the requirement of distribution of validation points, the thermal-visible calibration rig is swept through the entire range of the scene in what is defined as a *calibration sequence*. The duration of this sequence is limited by the performance of the calibration rig but should at least be 1 minute. Within this minute, the calibration rig is swept through as much as the scene as possible, covering multiple depths, poses, and ranges.

When the calibration sequence is conducted, a number of frames is extracted from the sequence, from which the corresponding point sets are extracted. As these point sets shall be used for a variety of applications, including the calibration and registration of cameras, at least 25 frames are extracted from each calibration sequence. These frames have to be picked carefully, and should reflect a broad range of depths, poses, and ranges of the calibration rig, reflecting the coverage of the calibration rig in the calibration sequence.

In order to generate as many point correspondences of each scene as possible, three calibration sequences are conducted for each scene. Calibration sequence 1 and 2 are used for training the registration algorithm whereas calibration sequence 3 is used for the validation of the algorithm. By choosing sequence 3 to provide the validation point correspondences, one might calculate the minimum amount of point correspondences generated by one view of the calibration rig:

$$\text{correspondences per rig} = \frac{1000}{\text{number of calibration sequences}} = 40 \tag{4.1}$$

The registration algorithm is thus trained with the point correspondences of calibration sequence 1 and 2 and validated with the point correspondences of calibration sequence 3.

4.2.3 Registration accuracy

The performance of the registration algorithm is measured for both the training and validation set. For each adjustable parameter of the registration algorithm, the registration procedure is run 20 times on both the training and validation set. The successive runs is required to level out eventual irregularities if the registration algorithm is based on non-deterministic measures.

The number of trials for the registration algorithm is thus defined as:

$$N = \text{number of scenes} \cdot 20 \cdot \text{number of parameters} \cdot 2 \tag{4.2}$$
$$= 3 \cdot 20 \cdot \text{number of parameters} \cdot 2 \tag{4.3}$$
$$= 120 \cdot \text{number of parameters} \tag{4.4}$$

The performance of the algorithm is measured by the symmetric transfer error and one-directional transfer errors specified in Section 3.2.4.

Creating the baseline

As specified in the requirement specification, the baseline for which the registration algorithm is compared against is the technique of global registration. The global registration uses a single homography to register the thermal and visible images. This homography is easily obtained using OpenCV's `findHomography` which uses the RANSAC technique described in Section 7.3.3. The RANSAC technique relies on a threshold for filtering the outliers, which might be arbitrarily defined. In order to find the best fitting homography, one should test multiple values for the threshold. Given that the coordinates are given in pixel, we will in this context vary the threshold between 1 and 30 and find the homography featuring the lowest symmetric transfer error of each test set.

Part II

Design

Chapter 5

Acquisition platform

The following chapter will go through the design and construction of a platform for acquiring the thermal and visible imagery. This includes the creation of a hardware platform and the use of software SDK's to provide a synchronised capture of images from both modalities.

5.1 Hardware platform

As stated in section 2.3, we will need a configuration consisting of one thermal camera and either a visible stereo camera configuration or a single visible camera equipped with 3D information of the scene. In the next section, we will investigate available vision equipment and choose an appropriate camera set-up for the image acquisition.

5.1.1 Visible light cameras

As noted above, we want to utilize depth information of the scene to enhance the performance of the registration algorithm. This means that we will either use a stereo camera set-up which uses texture information to induce the depth of the scene or a camera equipped with an active emitter to measure the depth of the scene.

Stereo cameras: Bumblebee XB3

The Bumblebee XB3 is a stereo camera by Point Grey. It includes three different cameras, each of which has a resolution of 1280x960 for a frame rate of 15 frames per second. Equipped with three cameras, the system comes with to baselines: wide mode with a baseline of 24 cm or narrow mode with a baseline of 12 cm [Point Grey Research, 2011]. The camera communicates with a computer through a FireWire interface. In terms of software, the system is shipped with the FlyCapture and Triclops SDK, which allow use of the Bumblebee in a C program; FlyCapture is specifically designed for image acquisition, whereas the Triclops SDK focuses on computer vision such as rectification of images and computation of the disparity. Table 5.1 gives more specifications of the Bumblebee XB3.

Property	Value
Focal length	3.8 mm or 6 mm
H_{FOV}	66 deg
	43 deg
Baseline, narrow mode	12 cm
Baseline, wide mode	24 cm
Maximum disparity	1024 px
Signal-to-Noise ratio	54 dB
Resolution	1280x960 @ 15 FPS
Dimensions	227 mm x 37 mm x 41.8 mm

Table 5.1: Bumblebee XB3 specifications [Point Grey Research, 2011]

The Bumblebee XB3 is a stereo vision system. This implies that the robustness of the output greatly depends both on the algorithms used for the computation and some elements in the scene that the system is known to be weak against. Specifically, this includes, but is not limited to, specular highlights and large non-textured areas.

By nature, stereo vision algorithms need a lot of computation to give the depth of a pixel: first, the two images have to be rectified so that epipolar lines are parallel; then several points are to be matched in both images; finally, the disparity has to be calculated, from which the depth is calculated. However, the Bumblebee XB3 can output rectified images directly, meaning that the first step may be skipped.

Accuracy and precision Based on experimental results, a good rule of thumb is to estimate a matching error of 0.22 pixels in the disparity [Point Grey Research, 2004]. Furthermore, the Bumblebee XB3 camera is pre-calibrated to guarantee a root-mean square error of maximum 0.08 pixels [Point Grey Research, 2010].

Microsoft Kinect

The Microsoft XBOX 360 Kinect is a camera containing a RGB camera, an infrared laser emitter, an infrared camera, a multi-array microphone, and a motorized tilt mechanism. The Kinect is available for about 176 EUR [Play.com, 2013].

According to the technical specifications of the Kinect, it has a horizontal field of view of 57 degrees and a vertical field of view of 43 degrees [Microsoft Developer Network, 2013a]. Further specifications may be seen from Table 5.2.

The Kinect is compatible with the the OpenKinect project and the OpenNI framework for natural interaction which allows for interfacing the Kinect with MATLAB if the 'Kinect for MATLAB' wrapper is utilized [Shirai and Yu, 2011].

Microsoft provides the Kinect for Windows SDK that provides speech recognition, skeleton tracking, and several modes for extracting the depth and colour information [Microsoft Corporation, 2013]. In this context however, we will only need to access the raw depth and colour information. The SDK is available for C++, C#, and Visual Basic and allows interaction with MATLAB using the 'Kinect Bridge'. The

Property	Value
H_{FOV}	57 deg
V_{FOV}	43 deg
Depth range, near mode	0.4 m - 3 m
Depth range, default mode	0.8 m - 4 m
Colour resolution	1280x960 32-bit colour @ 12 FPS 640x480 32-bit colour @ 30 FPS
Depth resolution	640x480, 16-bit depth @ 30 FPS
Audio resolution	24-bit at 16 KHz

Table 5.2: Technical specifications for Microsoft Kinect [Microsoft Developer Network, 2013a], [Eisler, 2013]

device requires power via a separate power supply and is connected to a computer through an USB cable.

Accuracy and precision Microsoft has not provided any public data on the accuracy and precision of the Kinect sensors. However, the community behind the Robot Operating System library has made measurements of those parameters [ROS.org OpenNI Wiki, 2013], and the accuracy of the *calibrated* Kinect sensor 'turns out to be very high', in numbers between ±1 mm. The error of the depth measurements is found to be almost proportional to the distance to the object squared. For one Kinect, the equation is presented as [ROS.org OpenNI Wiki, 2013]:

$$E = d^2 \cdot 0.0075 \tag{5.1}$$

According to the work of Khoshelham [2011], the error does indeed follow this pattern. According to the author, the Kinect has a depth resolution about 5 cm at a distance of 4 meters. At five meters, which is the 'extended' range with newer Kinect SDK's, the resolution increases to 7 cm. The decrease in accuracy is a consequence of the nature of the infrared pattern that the Kinect emits to measure the depth by triangulation.

ASUS Xtion

ASUS manufactures the Xtion Pro and Xtion Pro Live which share the majority of specifications with the Kinect. The Xtion Pro comes with depth sensors similar to the ones found in the Kinect, however, it does not include the RGB camera nor the microphone of the Kinect. The Xtion Pro Live includes those two emissions. Both sensors are interfaced through the OpenNI framework with C++ and C#. The range of the depth sensor is, according to ASUS, between 0.8 and 3.5 m - comparable to the default mode of the Kinect [Asus.com, 2013].

5.1.2 Thermal cameras

Commercial thermal cameras are available in a number of different configurations, most importantly distinguished by the range of infrared radiation visible to the cam-

era. The infrared range is defined between wavelengths of 0,7 μm to 1.000 μm. The range of interest for thermal camera manufacturers is substantially smaller, and might be sub-divided into categories shown in Table 5.3.

Division name	Abbreviation	Wavelength
Short-wave	SWIR	0,7 - 1,4 μm
Mid-wave	MWIR	3 - 5 μm
Long-wave	LWIR	8 - 14 μm

Table 5.3: Sub-division ranges for thermal cameras [Gade and Moeslund, 2013].

Humans with a body temperature around 37 °C are visible in both the mid- and long-wave infrared ranges. According to the physics of Planck's Law, the cooler the object, the longer the wavelength of the emitted radiation. This means that objects cooler than 0 °C are almost invisible for MWIR cameras, whereas LWIR cameras are able to see objects with temperatures ranging from -20 to 100°C. We are interested in cameras that are able to see humans and objects in the temperature ranges of 'everyday life' both outdoors and indoors, which makes LWIR cameras a good choice.

Due to the excessive cost of thermal cameras, we only investigate one thermal camera in the long-wawe range, the AXIS Q1922. The AXIS is readily available at the Visual Analysis of People (VAP) Laboratory at Aalborg University, whereas other cameras might only be accessible at an additional cost.

AXIS Q1922

The AXIS Q1922 thermal network camera is capable of capturing radiation in the LWIR-range. The camera uses an uncooled micro bolometer to passively capture the radiation and comes in a indoor and an outdoor configuration, the Q1922-E. Technical specifications of the camera is seen from Figure 5.4.

Property	Value
Lens	19 mm
Image resolution	640x480 @ 30 FPS*
Spectral range	8 – 14 μm
Sensitivity	$<$ 100 mK
Audio resolution	Stereo AAC LC @ 16 KHz

Table 5.4: Technical specifications for AXIS Q1922. *Frame rate only applicable in selected countries such as the EU and USA [AXIS Corporation, 2013a].

The data stream of the camera is accessible through an Ethernet cable. Unlike the Kinect camera which provides a framework for capturing single imagery, the AXIS camera comes with built-in support for video compression codes such as Motion JPEG or H.264. The camera and the data stream may be controlled from the AXIS SDK's for C++ or C#. The streamed imagery may also be viewed and controlled using a browser with AXIS Media Control. The camera is powered via either Power

over Ethernet or a separate DC-adapter. The AXIS camera is available at around $ 7000 [dss.com, 2013].

5.1.3 System configuration

For the current purpose of registering images at distances between 1 – 4 m, it is chosen to go with the Microsoft Kinect as the visual camera. This choice is based on the notion that the Kinect is superior to the Bumblebee XB3 at correctly determining the distance to objects in the short range. Stereo vision set-ups are dependent on the quality of the spectral information of the objects in the scene, which might vary due to lighting conditions or bad texture information of the object. As we want to have depth information of the scene even in bad lighting conditions or for objects containing sparse texture information, the Kinect is the device of choice. Should we, however, decide to extend the scene to depths outside the range of the Kinect, the Bumblebee will be a subtle choice.

As described in the previous section, the choice of thermal cameras is limited to the devices available at the VAP Laboratory. The available camera, the AXIS Q1922, however, is excellent in terms of image quality, frame rate and resolution, where it mirrors the resolution and frame rate of the Kinect. The SDK's of the Kinect and AXIS cameras both support C++ and C#, so it should be possible to make a decent software acquisition system for both set-ups.

Physical platform

It is necessary to construct a common hardware platform consisting of the two cameras. The platform should fix the position of both cameras and make it hard to alter the positions of the cameras when capturing a scene. If the position of the cameras is altered on-the-fly, it will be increasingly difficult to register the imagery of the two modalities.

Such a platform has been constructed and is seen from Figure 5.1.

The cameras have been placed on top of each other in order to minimize the baseline between the camera sensors. The smaller the baseline, the smaller the difference in field-of-view of the cameras, and the smaller the parallax on close objects.

The Kinect camera is connected to the computer through a USB port and powered by the 230 V adapter included with the camera. The AXIS camera is connected to the computer through an Ethernet cable and powered through a stock 12 V adapter.

5.2 Acquisition software

As described in Section 5.1, the AXIS and Kinect camera both utilize SDKs that allows interfacing in C++ and C#. In the following, we will go through the respective SDKs for the cameras and choose the SDKs for constructing a robust yet simple platform for acquiring imagery from both cameras. The development and execution

Figure 5.1: Hardware platform containing the AXIS Q1922 and the Microsoft Kinect for XBOX.

of the software interface is done using Windows, as this is the platform of the native SDK for the AXIS camera. The image streams from the cameras shall provide the basis for registering the two cameras, and thus there are criteria that should be fulfilled:

1. *Synchronization:* The image streams should either be fully synchronized in time or be traceable. That is, for each frame in the image stream, the exact system time should be known.

2. *Depth registration:* The depth and RGB streams are factory registered on the Kinect. Those registrations should be stored during capture.

3. *Calibration mode:* The interface should distinguish between a calibration mode and a capture mode. The calibration mode is for capturing imagery for camera calibration and registration, whereas the capture mode is for capturing the actual scene.

5.2.1 Interfacing the AXIS camera

The AXIS Q1922 thermal camera comes with the AXIS Media Control framework for Windows. The SDK is available to the general public and thus not subject to any restrictions on use nor distribution. According to the documentation, the AXIS Media Control is 'efficient for... capturing snapshot images, recording video sequences in ASF format...' [AXIS Corporation, 2013b]. The SDK comes with code samples for both Visual C++ and C#. Unfortunately, the C++ samples require the full version of Visual Studio Professional due to requirements for the Microsoft Foundation Class Library. This restriction is not apparent in the implementation with C# which fortunately work well with freeware such as Microsoft Visual Studio 2010 Express.

The sample code uses the AXIS specific AXISMEDIACONTROL Library which a Windows Forms application is built upon. The sample is capable of recording video

in 30 FPS from the camera out-of-the-box and as such, only a few adjustments are needed. The thermal camera is remotely controlled using a web interface from which we may control the time of the camera, and even more important, add an overlay on the images representing the frame rate and the actual time of the frame, as seen from Figure 5.2. However, these time stamps are referring to the camera system time which might deviate from the computer system time, even though these are aligned prior to capturing the frames.

Figure 5.2: Overlay shown in the thermal images. The current frame rate and camera time of the frame is shown.

In order to apply to the synchronisation criterion above, we need to provide the system time for each frame in another way. This is done by creating a list of strings which contains the system time stamps for each instance of the recorded thermal frames. Each time a new frame is generated from the camera, the function `myAMC_OnNewImage` is triggered which adds the current system time to the end of the list. At the end of a capture sequence, the function `writeLogToFile` writes the timestamps to the file `ThermalTimestamps.log`.

5.2.2 Interfacing the Kinect

When interfacing the Microsoft Kinect on Windows, we have multiple choices for choosing the framework. The OpenNi is a standard framework for 3D sensing [Openni.org, 2013] and widely used with multiple image sensors. As of December 2012, the framework is at version 2.0 and is increasingly mature. Alongside the OpenNi, Microsoft has developed its own framework for interfacing the Kinect, the Kinect for Windows SDK which is at version 1.6. Compared with its OpenNi counterpart, the Kinect for Windows SDK comes with multitudes of code examples and documentation, whereas the documentation for interfacing the Kinect with OpenNi is a little more sparse. Based on the level of documentation and the expected time to get the system up and running, it is chosen to use the Kinect for Windwos SDK for this implementation.

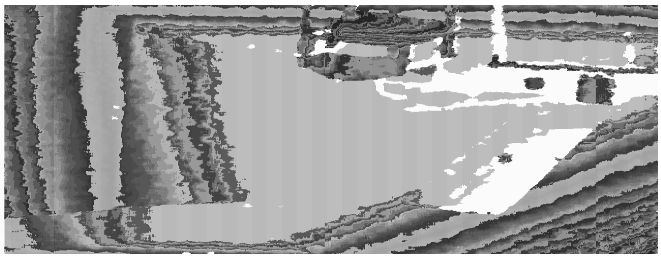
Contrary to the Axis camera, the Kinect for Windows SDK does not provide a native framework for capturing a video stream for the RGB and depth cameras. Therefore, we will either have to add a framework for capturing video or storing each frame individually. The latter approach is chosen as this eliminates the need for making a separate list of time stamps and enables us to provide the time of the frames via the file name. As we want to interact between the AXIS and Kinect capture platforms, the implementation is done by building upon the C# Color Basics Windows Presentation Foundation sample of the Kinect for Windows Developer Toolkit.

The image stream from the depth and colour sensors are enabled separately and handed to a `WritableBitmap` of the `System.Windows.Media.Imaging` Class. As seen from Table A.1 on page 96, there are several options of the resolution and frame rate of the image streams.

Those streams are coupled to an event handler that is called whenever both the depth and colour streams are available. Whenever this is the case, the function `SensorAllFramesReady` is called and processes the image streams. A detailed overview of the function is available in Appendix A.1 on page 96. During the development process, it was shown that the scope of taking images from streams and storing them on disk was outside the performance criteria for real-time performance at 30 FPS. Therefore, it is necessary to split the task of saving images to disk in separately threaded processes so that the function `SensorAllFramesReady` is able to end in timely fashion before the next event is triggered. This leads us to two new functions `processColor` and `processDepth` which are explained in detail in Appendix A.1 on page 96.

The RGB images are saved easily using the `save` method of the WriteableBitmap and thus the depth image should follow in this fashion. However, the data type of the depth image is different - whereas the RGB image is a conventional 4-channel 32 bit BGRA image, the depth image is a single channel 16-bit grey-scale image. The built-in method for saving 16-bit images in C# is substantially slower than the equivalent method for saving 8-bit images, making it impossible to achieve real-time performance at 30 or even 15 FPS. The OpenCV library comes with built-in functionality to save 16-bit images but in the implementation for C# using the Emgu Wrapper, the functionality is yet to be implemented. In order to improve performance, the depth image is converted to a single-channel 8-bit image of double the width of the original image and saved without further ado. In Figure 5.3, a double sized 8-bit image is shown besides the original 16-bit image. This workaround necessitates that we subsequently convert the depth images back to the original representation, which is done with the MATLAB script `convertDepthDataToUShort.m`. The script is explained in Appendix A.2.1.

(a) Original 16-bit image of size 640x480.



(b) Converted 8-bit image of size 1280x480.

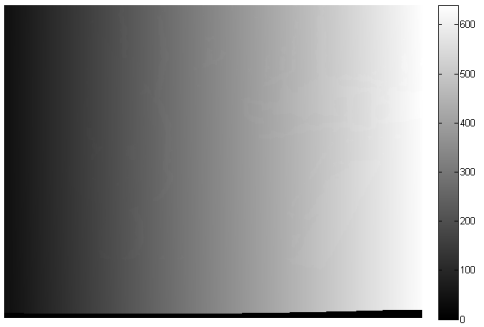
Figure 5.3: Depth images from the Kinect. In order to save the 16-bit images in real time, a conversion to the double-width 8-bit images seen in figure b is necessary. The images are flipped horizontally because of the orientation of the sensor.

Depth registration

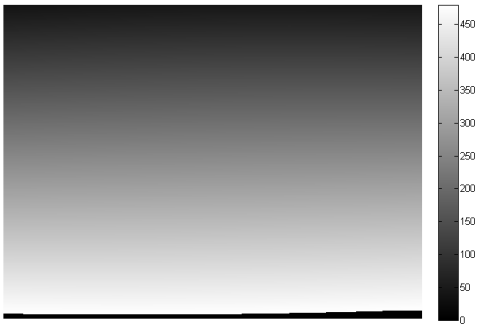
As described in the introduction to this chapter, the depth and RGB cameras of the Kinect are factory calibrated with respect to each other, meaning that for each pixel in the depth image, the camera provides an accurate registration to the corresponding pixel in the RGB image. This registration is available via the `CoordinateMapper` method of the Kinect for Windows SDK. Due to a bug in the 1.6 version of the SDK,

the mapping is available for registering the depth image to the RGB image - but not the other way around. This implies that we have to do the reverse mapping ourselves, which is done by the function `convertD2RGBtoRGB2D.m` described in Appendix A.2.2.

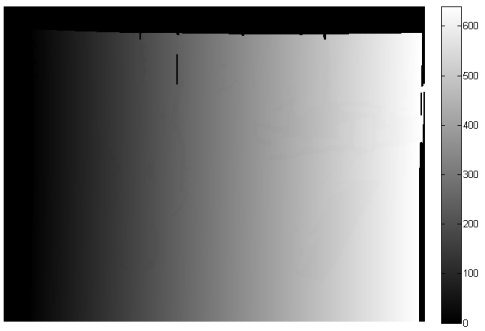
We assume that the depth-to-RGB registration is unchanged during capture of the scenes. This assumption gives us the freedom to initialise the registration once every scene. The assumption also eliminates the requirement of real-time registration performance which enables us to save the registration as 16-bit images. The conversion of the registration from the Kinects `CoordinateMapper` to an 16-bit image is performed using the function `calibrateDepthD2RGB`. The registration is saved as two 16-bit images where one image serves as a look-up-table for the x-coordinate and the other image serves as a look-up-table for the y-coordinate. The images of the look-up-tables for the depth to RGB and the mapped tables for RGB to depth is seen from Figure 5.4. One might see that the reversed mapping for the x coordinate contains certain artefacts. These artefacts are related to the nature of the inverse map and although the MATLAB function `convertD2RGBtoRGB2D.m` handles smaller artefacts such as small holes in the map, bigger artifacts such as the 'rip' in Figure 5.4c are too big to be handled. Thus, no meaningful mapping exists in such a hole. More details on this in Appendix A.2.2.



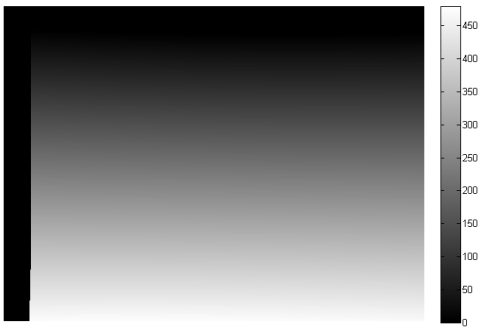
(a) Depth to RGB registration, x coordinate



(b) Depth to RGB registration, y coordinate



(c) RGB to depth registration, x coordinate



(d) RGB to depth registration, y coordinate

Figure 5.4: Depth to RGB and RGB to depth registration images. The images serves as look-up-tables for registering imagery from both modalities. The depth to RGB registration is obtained natively from the Kinect SDK whereas the RGB to depth registration is obtained by a transformation of the former registration.

5.2.3 Creating a common framework

In the above, we have described how to record images from the AXIS and Kinect sensors by using the native SDK's of the sensors. In this subsection, we will focus on the creation of a common framework for capturing the depth, visual, and thermal image streams simultaneously. As both interfaces are written in C#, we are able to integrate the AXIS platform into the code for the Kinect platform. At the top of the section, we required that the image streams should be synchronized, the depth registration should be stored, and a calibration mode be enabled. In the above, we have paved the way for image synchronization by providing each frame of the thermal camera with a separate time stamp in a text file and providing the system time of the depth and RGB streams of the Kinect at the file name of each frame. The registration of the depth image to the RGB image is implemented, and only the requirement of a calibration mode needs to be fulfilled.

This is done by introducing several modes in the combined graphical user interface (GUI), each corresponding to a flag. We may thus choose between a depth registration mode, a calibration mode, and a capture mode. These modes are implemented in the final GUI and is seen from Figure 5.5. A more detailed overview of the implementation is found in Appendix A.1. When pressing the "Calibrate depth" button, 15 frames of the depth-to-RGB registration is saved as 15 images of the mapped x and y coordinates respectively. The depth registration is saved in the calibration sub folder chosen by the radio buttons '1', '2', and '3'. The "Calibrate RGB + T" and "Capture RGB + D + T" buttons are essentially providing the same functionality - both buttons enable the capture of depth, thermal, and RGB streams. The former button does, however, store the captured image and video steams in the calibration folder whereas the latter button stores the data in the root folder provided.

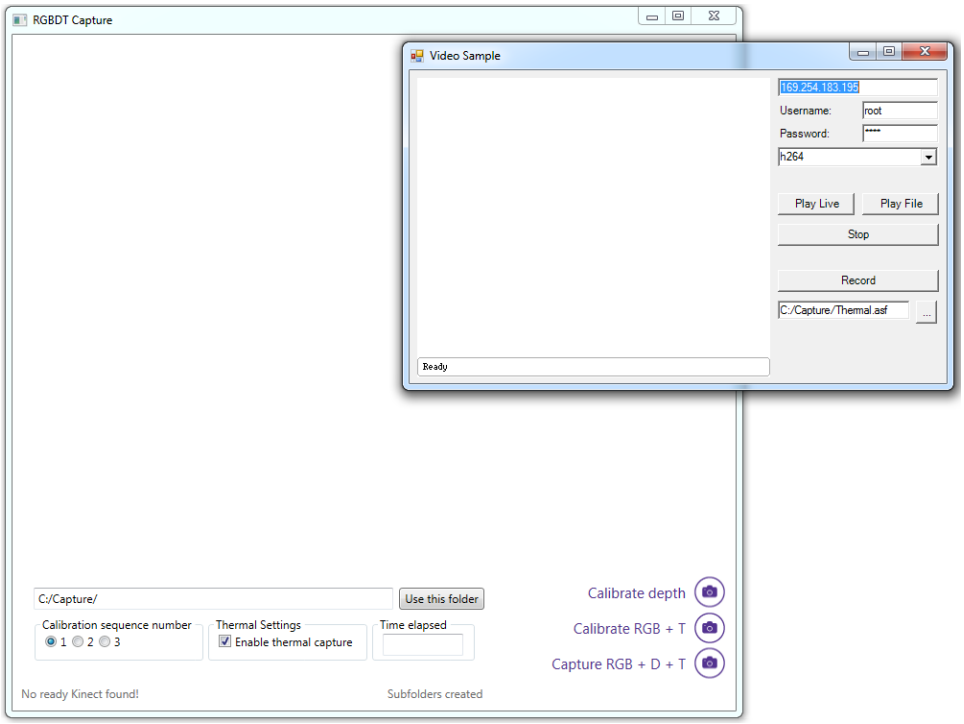


Figure 5.5: The GUI for recording the image streams from the Kinect and AXIS cameras.

The "video sample" window in Figure 5.5 is the AXIS part of the interface. The window is used for connecting to and opening the video stream of the AXIS camera and choosing the video codec. When these tasks are conducted, the control of the thermal camera is maintained from the "RGBDT Capture" window.

Selecting the frame rate

Both devices are equipped with sensors that provide images at 30 frames per second at equal resolution, 640x480 pixels. Nevertheless, it seems that the Kinect for Windows SDK is not built for real-time image recording. In Section 5.2.2 on interfacing the Kinect we experienced performance issues when saving the depth images. It turns out that the combined capture of depth, RGB and thermal images is too much for the SDK to handle, even with multiple optimizations such as threaded saving and the testing of several save methods for C#. At 30 FPS, the recording remains stable for up to one minute but suffers subsequently from a lag of approx. 0.8 s where no information is stored. This pattern repeats itself upon further capture and the nature of the lag seems non-deterministic. Therefore, the frame rate of the RGB sensor is lowered to 15 FPS which runs stable with no lag. The thermal camera still captures images at 30 FPS. The difference in frame rates does not pose any problem, however, as the synchronization step described in Section 6.1 will take care of this.

5.2.4 Post-processing the thermal images

The section above has described how to create and capture the image streams from the AXIS thermal and Kinect depth and RGB cameras. In order to further process the thermal imagery however, we need to split the video frames of the thermal camera. This is done by applying a FFMPEG-script that dumps each individual frame to a single file:

```
1 ffmpeg -i thermal.asf -r 30 -vsync 0 -f image2 T/%05d.png
```

With this amendment, we now have created a framework for capturing unsynchronized streams the thermal, depth, and RGB cameras. The synchronization of the image streams is treated in the next chapter.

Chapter 6

Camera synchronization and calibration

In the previous chapter we have described how to capture image streams from the visible, thermal, and depth cameras. In this chapter the synchronization, or the temporal alignment, of the imagery is implemented, and the thermal and RGB images are calibrated to remove distortion and to provide point-to-point correspondences for registration purposes.

6.1 Camera synchronization

The individual RGB and depth frames are provided with time stamps of the system time whereas the individual frames of the thermal camera are provided with two time stamps; the camera time of the frame printed in the top of the image frame and the system time of the frame provided in the text file `ThermalTimestampsThermal.log`. An example of a frame from the thermal camera is seen in Figure 6.1.

6.1.1 Aligning the thermal frames

Is it not guaranteed though, that the first extracted frame of the thermal camera corresponds to the first entry of the time stamp log. Therefore, a control frame is saved at the beginning of each capture with the system time specified in the file name. Such a frame is seen from Figure 6.1.

By including a control frame, the correspondence is provided between the system time and camera time. This allows the exact correspondence of the entries of the time stamp log and the extracted thermal frames. It is found that only the offset between the time stamp log and the extracted thermal frames might be wrong. When the offset is fixed, the correspondence between the time stamps and the extracted frames is found to be one-to-one. The offset is removed in the MATLAB-function `alignThermalFrames.m`.

The MATLAB-function reads the control frame and the first extracted frame of

Figure 6.1: Control frame from the thermal camera. The camera time, along with the frame rate, is printed on top of the image. The system time for the frame is 09-30-02-99 which is written in the file name.

the thermal image stream and prompts the user to input the camera time of both images. As we now know the correspondence between the system time and the camera time, the function is able to calculate the system time of the first extracted frame of the thermal image stream. The file of thermal time stamps is subsequently opened and searched to find the best matching time stamp of the first frame. When this time stamp is found, any earlier time stamps are discarded and a new log file is written. Thus, the correspondence between the thermal time stamps and the extracted frames of the thermal camera is now one-to-one.

6.1.2 Synchronizing the captured image streams

Now that we have a proper correspondence between the thermal images and the time stamps, a full synchronization between the thermal, depth, and RGB images is in place. The synchronization is implemented in the MATLAB-function `synchronizeData.m` and is described in the following.

The function loads time stamps from the thermal log file and the directory of the recorded RGB frames. The recorded imagery from the RGB and depth cameras are already synchronized and so there is no need to load time stamps for the depth modality. With time stamps of the frames from both the AXIS and Kinect cameras, we might begin to match the individual time stamps. In this implementation, the time stamps of the Kinect RGB frames are matched with the thermal frames one-by-one within a predefined search window. For each RGB time stamp, we search for the lowest time difference in the search window based on the best match of the *previous* RGB time stamp. This is illustrated in Figure 6.2. The search window is currently defined to be very wide in order to go with belt and braces but might be optimized for speed.

The time differences between the current RGB time stamp and all thermal time stamps in the search window are computed, and the thermal time stamp corresponding to the lowest time difference is chosen as the best match, as seen in Figure 6.2.

The correspondence pair is written into the 'Best match' array until corresponding thermal time stamps for all the Kinect frames are found.

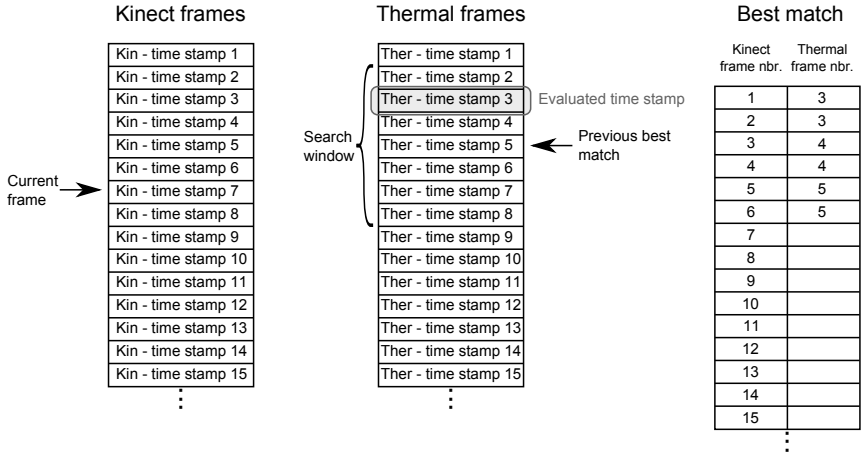


Figure 6.2: The synchronization in overview. For each RGB frame, the closest thermal frame in time is sought. In order to not search through the entire array of thermal frames for each correspondence, the search is limited to a search window centred on the best match of the precious RGB frame. For each RGB frame, the closest thermal frame is written into the array of best matches.

However, the 'best match' array will almost certainly not provide a one-to-one mapping between the RGB and thermal frames, especially when their frame rates are different, which is the case here. There might be several RGB frames which maps to one thermal frame and vice versa.

In order to handle this, a run through the 'best match' array is conducted and each correspondence is checked. If two RGB frames refer to the same thermal frame, their differences in time to the thermal time are compared and only the correspondence with the lowest time difference remains. Thermal frames which are left unreferenced in the 'best match' array are omitted through an exclusive OR operator.

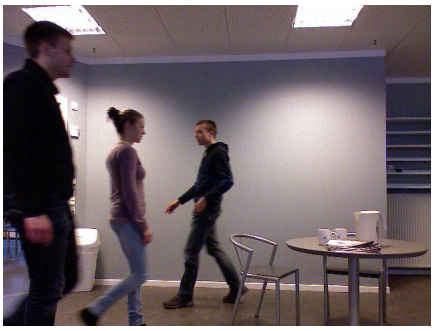
The arrays of RGB and thermal time stamps are now strictly one-to-one and we might now read the corresponding frames and write them into three synchronized folders, *SyncD*, *SyncT*, and *SyncRGB*. Samples of synchronized imagery is seen from Figure 6.3.

6.2 Camera calibration

With synchronized sets of images of RGB, thermal, and depth modalities, we might focus on calibrating the RGB and thermal cameras in order to remove image distortion and to provide point-to-point correspondences for registration purposes.

6.2.1 Camera model

In order to undistort the images, the notion of camera geometry needs to be introduced. The image formation process is geometrically presented as the projection of a 3D scene unto a 2D plane and is described mathematically using projective geometry.



(a) RGB image

(b) Thermal image

(c) Depth image

Figure 6.3: Synchronized imagery from the Kinect and AXIS cameras. Notice the differences in field of view of the three modalities.

Multiple camera models exists that describe the projection and the most common is the *pinhole camera model* which is illustrated in Figure 6.4. The image plane is placed at $Z = f$ and is the plane, where the image is formed. The point at the world coordinate $\mathbf{X} = (X, Y, Z)^T$ is projected to the image plane at the intersection of the plane with the line connecting the camera centre C and the world coordinate $\mathbf{X}$. The projected point on the image plane is denoted as $\mathbf{x} = (x, y)^T$. The image plane is centred at the principal axis $\mathbf{Z}$ which is the Z-axis of the right handed Euclidean coordinate system with origin at the camera centre. In order to comply with the default coordinate system used with images, the y axis is inverted. The point where the principal axis intersects the image plane is called the principal point, p, and is typically in the centre of the image.

The mathematical relationship between the point in world coordinates and image coordinates is described by [Hartley and Zissermann, 2003]:

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \tag{6.1}$$

where f is the focal length.

By using homogeneous coordinates the mapping is described as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6.2}$$

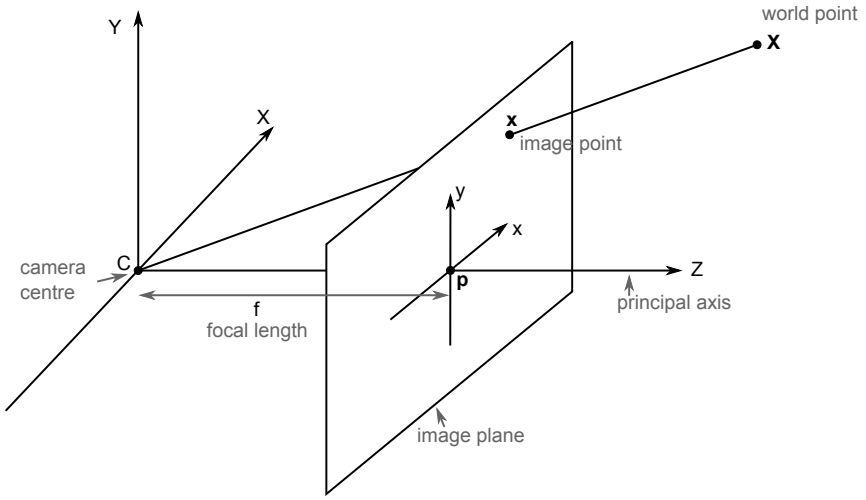


Figure 6.4: Geometric relations for the pinhole camera model. The centre C denotes the centre of the projective camera and P denotes the principal point of the camera. With inspiration from Hartley and Zissermann [2003] and Laganière [2011].

The above equation does hold if world points are expressed in the camera coordinates. However, points are often expressed in a different coordinate frame and thus needs a rotation and translation to be represented in the camera coordinate frame. The rotation is expressed as a 3x3 matrix and the translation is expressed as a 3-vector. When combined in one matrix, equation 6.2 might be generalised as:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_2 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6.3}$$

The above equation describes the idealized projection of points onto the image plane. However, the physical camera does not act as the ideal model as physical imperfections or manufacturing inaccuracies introduce the following parameters:

- **Principal point offset**
  In the above, the origin of the image plane is placed exactly at the principal point. The real centre might be subject to a small offset, so the parameters p_x and p_y are used to express the coordinates of the principal point in the image. In pixel dimensions, the principal point is expressed as $x_0 = m_x p_x$ and $y_0 = m_y p_y$ in the x and y directions respectively.

- **Non-square pixels**
  The pinhole camera model assumes that the pixels in the x and y directions are equal in scale. With common CCD cameras, this is not guaranteed to be the case [Hartley and Zissermann, 2003]. Therefore, a scale factor m_x, m_y is multiplied on the focal length, thus representing the focal length in terms of pixels in the x and y direction. The generalised focal length is expressed as: $\alpha_x = f m_x$, $\alpha_y = f m_y$.

- **Skew**
  In case of non-parallel image axes, a skew parameter, s is introduced. In practice however, the parameter will almost remain zero.

With the above additions, the pinhole camera model is noted as the *finite projective camera* [Hartley and Zissermann, 2003]:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_2 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6.4}$$

The 3x3 matrix in the above equation is known as the *calibration matrix*, K, of the camera and describes the *intrinsic parameters* of the camera. The intrinsic parameters remain unchanged when capturing multiple scenes and might thus only be determined once for a single camera. The rotation and translation are known as the *extrinsic parameters* of the camera and relates the world to the camera frame.

The above equation might be expressed in short form as:

$$P = K[R|\mathbf{t}] \tag{6.5}$$

where P is the camera matrix, K is 3x3 matrix of intrinsic parameters, R is the 3x3 rotation matrix and t is the translation 3-vector. The camera matrix P is a 3x4 matrix with 11 degrees of freedom.

In the creation of the finite projective camera model, we have this far treated the model as if no lens existed or if the lens was created with no artefacts. However, most lenses introduce optical distortions, and with lenses of bad quality or short focal length, noticeable *radial distortion* is apparent [Laganière, 2011]. If the centre of the lens is not positioned carefully at the principal point, the image is also subject to *tangential distortion*. The radial distortion is most noticeable when the distance to the principal point is large and results in the bending of otherwise straight lines. In the widely used Camera Calibration Toolbox for Matlab and the implementation in OpenCV, the distortion is modelled as [Bouguet, 2004]:

$$\mathbf{x}_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + kc_1\ r^2 + kc_2\ r^4 + kc_5\ r^6)\mathbf{x}_n + \mathbf{dx} \tag{6.6}$$

where $\mathbf{x}_d$ is the normalized image coordinates, r is the radial distance to the image centre, $r^2 = x^2 + y^2$, kc_x is the distortion coefficients to be determined, and $\mathbf{d}_x$ is the tangential distortion vector defined as:

$$\mathbf{dx} = \begin{bmatrix} 2kc_3\ x\ y + kc_4(r^2 + 2x^2) \\ kc_3(r^2 + 2y^2) + 2kc_4\ x\ y \end{bmatrix} \tag{6.7}$$

Once coordinates are undistorted, they might be used for estimating the parameters of the camera matrix of Equation 6.4.

6.2.2 Estimating the camera matrix

The camera matrix of Equation 6.5 has 11 degrees of freedom and thus requires 11 equations to solve. The matrix is solvable from point correspondences relating world points to image points. Each point correspondence gives two degrees of freedom, thus giving us a minimum of 6 correspondences to solve the camera matrix. The

minimum configuration necessitates that no noise is apparent in the image points nor the world coordinates; otherwise the estimated camera matrix will be faulty. As the data might not be exact and there inevitably will be noise in the extracted image coordinates and in the world coordinates, we will in general need much more points to achieve a robust solution. As such, the solution for estimating the camera matrix is overdetermined and as noise exist in the observations, one exact solution does not exist. A solution is obtained using optimization methods as implemented in the *Camera Calibration Toolbox for Matlab* [Bouguet, 2004] and its OpenCV counterpart [Bradski, 2000].

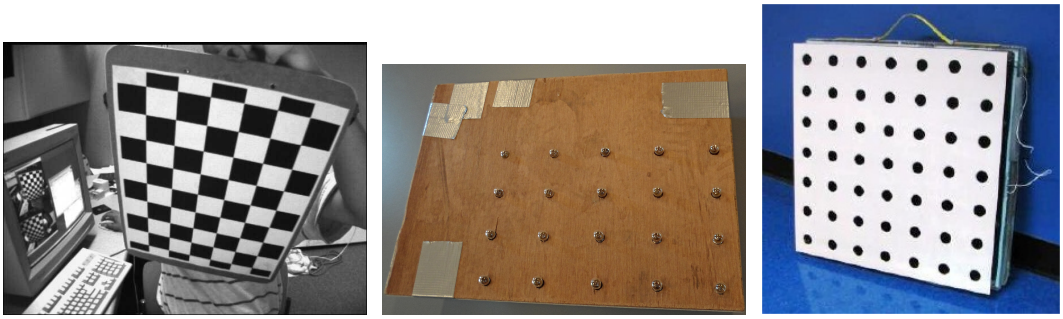
To automate the process of finding the point correspondences, a widely used method in Computer Vision is to show a chessboard pattern to the camera. The chessboard is shown in different depths and poses to the camera and when varied properly, 10 to 20 views of the chessboard are sufficient [Laganière, 2011]. When calibrating multiple modalities however, the need for multimodal calibration rigs occur. This will be treated below.

Multimodal calibration boards

The black and white chessboard pattern seen in Figure 6.5a is the standard chessboard used for calibration of visible imagery. However, the pattern is hardly visible when seen by a thermal camera unless the pattern is illuminated with high intensity halogen bulbs as used by Krotosky and Trivedi [2007]. The high energy use and the size of the halogen bulbs make this a hardly removable nor flexible configuration. There is thus a need to find a more versatile approach of providing a multi-modal chessboard that is visible in both thermal and visible light. We will not use calibration procedures for the depth camera as the depth stream is factory registered to the RGB camera and there is thus no need for further calibration or registration purposes.

A thermal calibration rig has been proposed by St-Laurent et al. [2010] which presumably uses a special construction to heat a board with circular cut-outs. The exact configuration is hard to find, though, as the authors do not provide any hint on how to remake the calibration rig shown in Figure 6.5c. Zhao and Cheung [2009] do not use a chessboard for providing point-to-point correspondences but uses a coloured, heated metal tag seen in Figure 6.5d. Another approach is to use a chessboard constructed by material with different emissivities as proposed by Ursine et al. [2012] and seen in Figure 6.5e. The authors use copper squares mounted on a backplate of other material. When light hits the chessboard, the emittance of the squares will vary, thus resulting in a pattern visible to the thermal camera.

In Figure 6.5f, the proposal of Vidas et al. [2012] is shown. Instead of providing the chessboard with different emissivity properties, the chessboard mask is a cardboard plate with square cut-outs. The mask is held on top of a backdrop with different emissivity properties. The authors suggests that backdrops includes warm objects and powered computer monitors. From Figure 6.5f, the object is seen. Anders Jørgensen of the VAP group at Aalborg University has proposed a custom mask of light bulbs seen from Figure 6.5b. The method works fine when obtaining images with the thermal camera but fails in the visible domain due to the fact that the centres of the bulbs are hard to distinguish from another on the Kinect RGB camera.



(a) Standard calibration chessboard. Bundled with OpenCV [Bradski, 2000]

(b) Chessboard of light bulbs

(c) Thermal calibration rig [St-Laurent et al., 2010]

(d) Depth image [Zhao and Cheung, 2009]

(e) Copper chessboard [Ursine et al., 2012]

(f) Geometric mask with backdrop [Vidas et al., 2012]

Figure 6.5: Thermal-visible calibration devices for capturing point correspondences in both modalities.

For the calibration of the thermal-visible cameras for this project, the proposed method of Vidas et al. [2012] is chosen. The method shown in Figure 6.5c is interesting, but without further information, the calibration rig cannot be recreated. The performance of the calibration board of Figure 6.5e is yet to be documented and thus questionable for this project. The proposal of the method shown in Figure 6.5d is interesting but provides only one correspondences per image. The performance of the chosen method is well-documented and relatively easy to recreate.

Opposed to the original proposal, we have enlarged the calibration board to comprise of an A3 sheet cut out from a ring binder, depicted in Figure 6.6. The backdrop used is an A3 sized 10 mm foam board made of polystyrene. Prior to a calibration recording, the backdrop is heated using a heat gun. Images of the calibration board in active use is seen from Figure 6.7.

In order to calibrate the cameras with the proposed mask, the mask is swept throughout the entire scene in a variety of depths and poses during a one minute pass recorded by the AXIS and Kinect cameras. This process is repeated twice or trice in order to capture a sufficient amount of calibration data. The full procedure of capturing the calibration sequences is described in Appendix E.2.

6.2.3 Processing the calibration images

Once the calibration images are captured and synchronized, the processing of the image frames for calibration is begun. By using the MATLAB-function `getCali-`

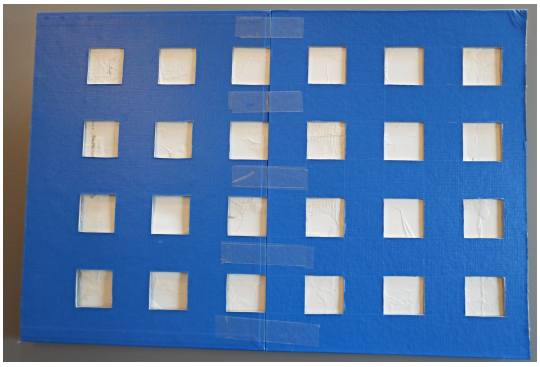


Figure 6.6: The calibration board used for the thermal-visible calibration.

(a) RGB image **(b)** Thermal image **(c)** Depth image

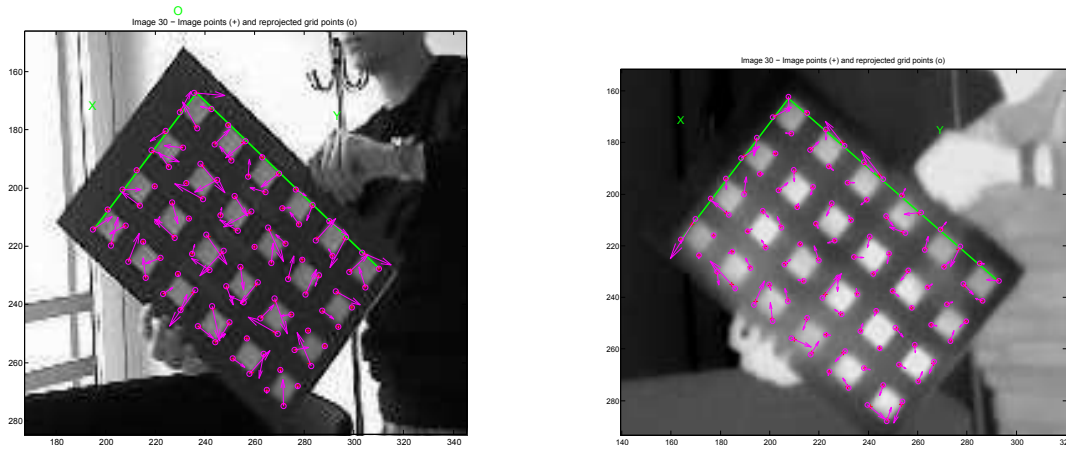
Figure 6.7: The calibration board as captured by the thermal, depth, and RGB cameras. The frames are synchronized as described in Section 6.1.

`brationImages.m`, a defined number of calibration images is saved to the folder 'CalibrationHelper' of each individual calibration folder. The number of calibration images and the start and stop image is input to the function which selects a subset of calibration images uniformly spread throughout the chosen range. The automatic extraction of image frames for the calibration is based upon the assumption that movement of the calibration board is distributed evenly in space throughout the calibration take. After the extraction of calibration images, the image set is manually examined to spot if any image regions are better represented than others and in this case, more calibration images are added. The Calibration Toolbox for MATLAB [Bouguet, 2004] is used for the extraction of the chessboard corners in both the thermal and visible modalities. An example of points extracted using the toolbox for both modalities is seen in Figure 6.8. Through multiple passes of corner extraction, re-projection and refinement, an accurate point set is obtained for both modalities.

The extracted points and the computed camera matrices are stored as .mat files. Data for the thermal camera are stored as `calib_data_right.mat` whereas data for the RGB camera is stored as `calib_data_left.mat`.

However, as we want to modify the images further in OpenCV, we need to export the point correspondences to a file readable by OpenCV. This is done by the MATLAB-function `saveCalibVarAsText.m` which outputs files containing the image and object coordinates of the extracted chessboard points. The coordinates are subsequently loaded into OpenCV where the function `calibrateCamera` calibrates the thermal and visible cameras one-by-one.

By using the computed camera matrices and distortion parameters, the function



(a) RGB image **(b)** Thermal image

Figure 6.8: Extracted and re-projected points using the Camera Calibration Toolbox for MATLAB. The extracted points are marked with x and the reprojected points with O.

`undistort` is able to undistort the images for lens distortion as shown in Figure 6.9.

(a) Undistorted RGB image **(b)** Undistorted Thermal image

Figure 6.9: Undistorted calibration images. The lens distortion is removed from both cameras and straight lines appear straight in both images.

Chapter 7

Image registration

7.1 Introduction to image registration

Whereas Chapter 5 and 6 treated the construction of the image acquisition platform and the synchronization and calibration of the captured imagery, this chapter will treat the problem of constructing a registration algorithm for accurately registering objects in the thermal and visible images.

In the preliminary analysis in Chapter 2, several methods on thermal-visible image registration are listed. Table 2.1 and Figure 2.2 provide a good overview of these methods. Based on the conditions set by the requirement specification of Chapter 3, the goal is to create a registration algorithm that works for objects in located in the range of 1 – 4 metres from the cameras.

Due to the short range, the method on infinite homography of Figure 2.2 is easily rejected as the short distance between the scene and the cameras will inject severe parallax if one tries to apply a homography that fits the entire scene. This also applies for the baseline technique of global registration, which applies a single homography in order to register a restricted region in the scene. In theory, those techniques could be applied for a strictly planar region of the scene. However, even if movement in the scene is restricted to a straight line, the differences in depth of the human body would induce noticeable parallax due to the low distance ratio between the camera and the scene and the inter-object distances perpendicular to the registered plane.

This leaves us with the techniques of *stereo geometry* and *multiple homographies* which will be treated in depth in the following. The implications of stereo geometry on stereo imagery is known in Computer Vision as *stereo rectification* and might be applied for any generic two-camera configuration.

7.2 Stereo rectification

In order to understand the technique of stereo rectification, one must understand the fundamentals of *epipolar geometry*, which in brief will be reviewed here. Epipolar geometry describes the intrinsic relationship between two cameras which is defined

by the intrinsic properties and the camera pose of the two cameras.

In this framework, a *epipolar point* is defined, which is the camera centre of the first camera pictured by the second camera - and vice versa. The epipolar point is thus the point in each image where the camera centre of the other camera is located - with respect to the current camera. The epipolar point and the epipolar geometry is illustrated in Figure 7.1. The epipolar point is not necessarily inside the image and will, in typical stereo camera configurations, be situated far beyond the visual image. The epipolar points are always situated on the baseline connecting the two camera centres. Planes which include the baseline are called epipolar planes and span a one-dimensional family of epipolar planes. The epipolar plane intersects the image plane of the two cameras in *epipolar lines*. The family of epipolar lines connects every pixel of the image with the epipolar point. By knowing the epipolar geometry of two cameras, the search for corresponding points in two images is reduced to a one-dimensional search along the corresponding epipolar line in the second image.

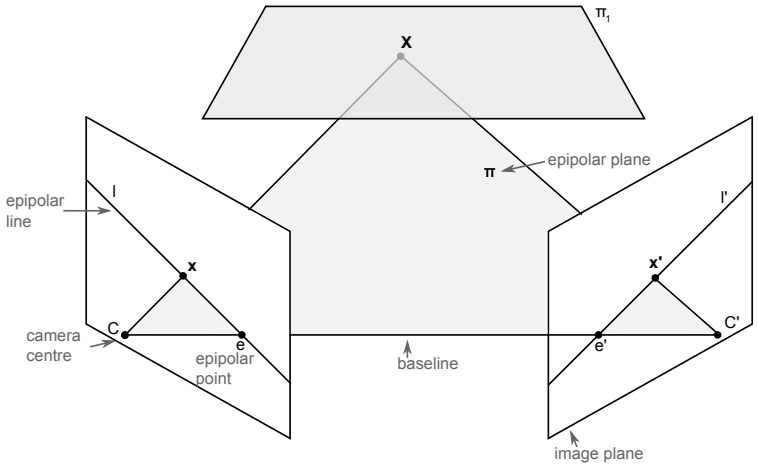


Figure 7.1: Epipolar geometry between two views. The epipolar point in one view is the image point of the camera centre of the other view.

The technique of stereo rectification applies transformations to images from both cameras to transfer epipolar points to the point of infinity, thus resulting in perfectly vertical or horizontal epipolar lines. The point at infinity is represented in homogeneous coordinates as $(1, 0, 0)^{\mathrm{T}}$. With epipoles at infinity, the search for corresponding points is thus be reduced to a one-dimensional search within the x- or y-axes of the images.

When world points are located on a plane π_i, there exists a 2D homography mapping each of the points on the plane seen by the first camera to the image plane of the second camera. The homography, $H_{\pi i}$, is transferring points given the equation:

$$\mathbf{x}' = H_{\pi i}\mathbf{x} \tag{7.1}$$

This property, along with the notion that the epipolar line might be expressed as the cross product of the epipolar point and the point in one image, introduces the *fundamental matrix*, F: [Hartley and Zissermann, 2003]

$$\mathbf{l'} = \mathbf{e}' \times H_{\pi i}\mathbf{x} = F\mathbf{x} \tag{7.2}$$

The fundamental matrix is a 3x3 matrix that encapsulates the epipolar geometry algebraically. It has 7 degrees of freedom and the important relationship on corresponding points:

$$\mathbf{x}'^{\mathrm{T}} F \mathbf{x} = 0 \tag{7.3}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are pairs of corresponding points in two images.

The fundamental matrix may also be calculated from the camera projection matrices or from at least 8 corresponding points. This leads to two different approaches to stereo rectification, the calibrated and the uncalibrated approach, respectively. In the following, the implementation of these methods will be described.

7.2.1 Calibrated stereo rectification

In calibrated stereo rectification, the fundamental matrix is computed using the following property:

$$F = \mathbf{e}' \times P' P^{+} \tag{7.4}$$

where P' is the camera projection matrix of the second camera and P^{+} is the pseudo-inverse of P, the camera projection matrix of the first camera.

Stereo calibration estimates the camera projection matrices of a dual camera configuration such that the reprojection errors of the corresponding points in both cameras are minimized, much like the calibration process for a single camera described in Section 6.2. As corresponding points for both modalities are generated by the thermal-visible calibration board of Vidas et al. [2012], the required data is in place to perform a full stereo calibration. In the following, we will distinguish between two cameras; the left and the right, but we could as well refer to a upper or lower camera, as long as the cameras are placed in a dual configuration.

The OpenCV framework for C++ features robust functionality for performing calibrated stereo rectification. The `StereoCalibrate` function of the framework tries to minimize the reprojection error of the camera corners in both the left and the right view by adjusting the intrinsic and extrinsic calibration parameters of the two cameras. The function computes the fundamental matrix of the two views and computes a rotation and translation matrix that projects the coordinate system of the right camera onto the coordinate system of the left camera. As the intrinsic parameters for each camera are already computed in Section 6.2, the function is called with the `CV_CALIB_FIX_INTRINSIC` flag which means that `StereoCalibate` just optimizes the extrinsic parameters of the camera matrices, thus reducing the parameter space to be optimized and generating a more robust solution. Fully calibrated, the function allows the transfer of 3D coordinates between two cameras views.

However, the correspondence problem between corresponding point pairs of the views is yet to be solved. The epipolar points in both images still need to be shifted to the point at infinity and transformations for doing so should be computed. This is done by the function `StereoRectify`. The function accepts as input the camera matrices and projective transformations computed by `StereoCalibrate` and computes the projections required to map the epipoles to infinity, thus resembling a canonical

camera configuration. In the canonical camera configuration, both image planes are residing on the same world plane, and thus epipolar lines in one image are parallel to epipolar lines in the other image.

In overall terms, the `StereoRectify` function does the following: [Lee, 2012]

1. Construct a rotation matrix R_{rect} which rotates the left camera to make the left epipole go to infinity.
2. Apply the rotation matrix R_{rect} on the right camera.
3. Rotate the right camera by the rotation matrix R obtained in `StereoCalibrate`, thus resembling a canonical camera configuration.
4. For each pixel of the rectified left and right cameras, calculate the corresponding point in the original, unrectified images.
5. Adjust the scale of the rectified images.

Step 5 includes the scaling of the rectified images. In the OpenCV implementation, one might choose an arbitrary number between 0 and 1. When fixed at 0, the rectified images only includes 'valid pixels', whereas a scaling parameter of 1 includes all pixels of the original images such that these are retained in the the rectified images. Due to transformation applied, a scaling of 1 also includes non-valid pixel regions due to the transformation of the rectified images.

Preliminary tests

The rectification by calibrated stereo is tested against the proposed scenes in Appendix D.1. The results show that the algorithm is unable to find a stable solution of the image rectification matrices, resulting in heavily distorted images that are not suitable for image registration. This applies no matter the size of the dataset that is used for the calculation of the calibrated stereo. Based on these results, the further work on the calibrated stereo approach for image rectification is discarded.

7.2.2 Uncalibrated stereo rectification

If the calibrated approach to stereo rectification is impossible to perform due to the absence of calibration data, one is able to construct a rectified view between two images based on the estimation of the fundamental matrix and the rectification matrices. The fundamental matrix might be estimated up to a projective ambiguity from 8 or more corresponding point pairs. Several methods for computing the fundamental matrix exists, such as the normalized 8-point algorithm, the least median of squares (LMS), and the RANSAC-approach [Hartley and Zissermann, 2003]. The estimation methods are based on the definition of the fundamental matrix:

$$\mathbf{x}'^{\mathrm{T}} F \mathbf{x} = 0 \tag{7.5}$$

If the point correspondences are optimal, the linear solution of the 8-point algorithm will provide a perfect estimate of the fundamental matrix. However, as there

will inevitably be noise in the image coordinates, the robust methods of LMS or RANSAC should be applied. The RANSAC-scheme is chosen for this implementation as it computes the optimal solution no matter the amount of outliers in the point set provided. The RANSAC-method relies on a threshold for discriminating the outliers from the inliers, which needs to be set manually. The fundamentals of the RANSAC-method is described in more detail in Section 7.3.3.

When computed, the fundamental matrix is used to find the epipoles in the two images using the following properties: [Hartley and Zissermann, 2003]

$$F\mathbf{e}_l = 0 \qquad {\mathbf{e}_r}^{\mathrm{T}} F = 0 \tag{7.6}$$

where $\mathbf{e}_l$ and $\mathbf{e}_r$ are the epipoles in the left and the right image, respectively.

The epipoles of the left and the right image is thus given by the right and the left null vector of F, respectively. With the estimation of both the fundamental matrix and the epipolar points in each image, we may now construct a transformation that maps the epipole to infinity. In the uncalibrated stereo rectification of Hartley [1999], the epipolar lines of the two images are mapped to lines parallel to the x-axis, which means that the epipoles should be mapped to the point at infinity given as (1, 0, 0). If points should lie parallel to the y-axis, the epipoles should correspondingly be mapped to (0,1,0).

It is found that the following transformation maps the epipole to infinity: [Hartley, 1999]

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{bmatrix} \tag{7.7}$$

The corresponding transformation that maps the epipole to infinity in the other image is found by an iterative process that minimizes the distance function:

$$\sum_i d(H\mathbf{x}_i, H'\mathbf{x}_i')^2 \tag{7.8}$$

The corresponding pair of projective transformations, or homographies, is mapping corresponding pairs of epipolar lines on top of each other when transferring from one image to the other. With the algorithm of Hartley [1999], two projective 2D transformations are computed that rectifies the stereo images such that the epipoles are mapped to the point at infinity.

OpenCV features an implementation for computing the fundamental matrix from point correspondences, `findFundmantalMat`. The fundamental matrix computed from this is led to the function `stereoRectifyUncalibrated` which computes the rectification matrices based on the theory of Hartley [1999] listed above.

The computation of the fundamental matrix from point correspondences is generally robust, but as the algorithm relies heavily on epipolar geometry, if might fail severely if lens distortion is apparent. Therefore, images are undistorted before being led to the process of uncalibrated rectification.

Preliminary tests

The performance of the uncalibrated stereo rectification is tested on scene 1 – 3 in Appendix D.2. The results show that the rectification algorithm performs worse than the baseline provided by a single homography in scene 1 and 3. For scene 2, the performance is generally better than the baseline, but only for the test set. For the training set, the algorithm falls short on providing a decent mapping between corresponding point sets. As the baseline is seen as the least acceptable performance of the mapping, the method on uncalibrated stereo rectification is consequently not included in the acceptance test. Methods on refining the performance of the algorithm are conducted, though, and will be presented in the next section.

7.2.3 Depth refinement

The methods on stereo rectification rectify the images so that the search for corresponding pixels is limited to only one direction. However, this assumes that one is able to search for corresponding pixels or features along the rectified epipolar lines. The search for correspondence is complicated greatly in thermal-visible imagery due to the immanently different properties of the modalities. In the preliminary tests above, we have assumed that the correspondence is good enough 'as is' and that no correction along the rectified epipolar lines should be added. As might be seen from the tests, the registration with this assumption is not satisfactory, and we might turn to other methods to gain further accuracy.

When cameras are only translated with respect to each other in a stereo camera configuration, the relationship between corresponding pixels is reduced to: [Hartley and Zissermann, 2003, pp. 249]:

$$\mathbf{x'} = \mathbf{x} + K\mathbf{t}/Z \tag{7.9}$$

where K is the camera matrix, $\mathbf{t}$ is the translation vector between the camera centres, and Z is the Euclidean depth to the object to be registered.

This set-up is indeed the canonical camera configuration and resembles the rectified imagery obtained from stereo rectification. This means that the corresponding points only move along either the x- or the y-axis, and the equation might be simplified as:

$$\mathbf{x'} = \mathbf{x} + \mathbf{t}/Z \tag{7.10}$$

Which means that disparity between points along either the x- or y-axis is equal to the inverted Euclidean distance to the corresponding world point. This dependence might be used for improving the accuracy of the stereo rectification. In order to test this, the relationship of the inverted depth and the distance in the x-direction for the uncalibrated stereo rectification of scene 1 has been extracted and plotted in Figure 7.2. In the camera configuration shown in Figure 5.1, the cameras are placed vertically with respect to each other, so one would expect the disparity to be apparent in the y-direction. However, the disparity is shown in rectified coordinates, and as the uncalibrated stereo rectification algorithm is only capable of aligning images horizontally, the images are rotated to fit this assumption, thus resulting in a

disparity in the x-direction. Regressions for Scene 2 and 3 is seen from Figure D.3b and D.3c in the appendix.

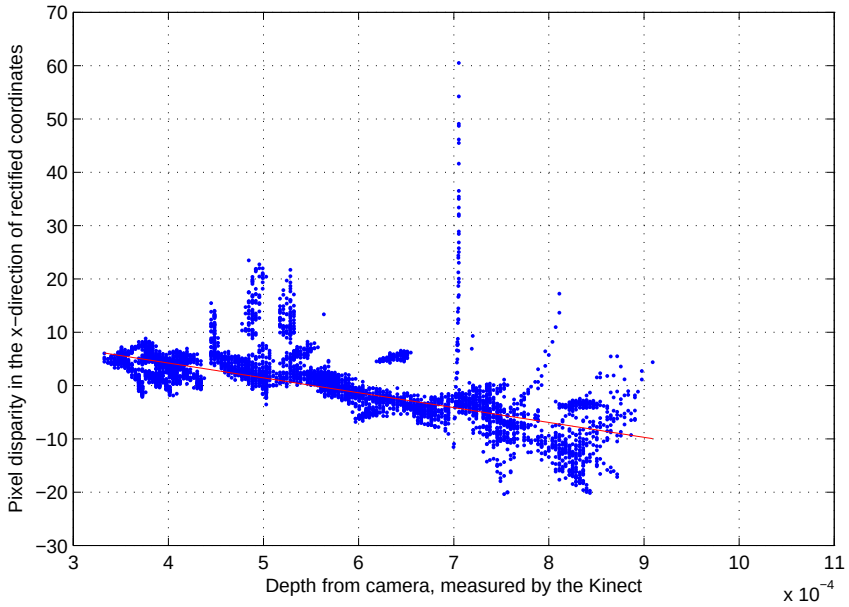


Figure 7.2: Relationship between the inverted depth in the x-direction of the rectified images in scene 1

With the robust fitting tool of MATLAB, a linear regression is estimated from the points. It is seen from Figure 7.2 that the observations do follow a descending pattern, but due to noise of the observations is is hard to say if the relationship between the disparity and the inverted depth is strictly linear. The depth refinement technique is used to register the tri-modal dataset presented in [Escalera et al., 2013] which is co-authored by the author of this thesis.

Preliminary tests

In Appendix D.3, the performance of the depth refinement technique is measured against the original implementation of the uncalibrated stereo rectification. The use of a linear regression does improve the accuracy for scene 1 and 3, but doubles the transfer error of scene 2 compared to the uncalibrated stereo rectification. The improved performance of scene 1 and 3 is bringing the mapping on par with the baseline, for which similar results are measured. We would like to improve the performance beyond the baseline, however, and thus the depth refinement technique is discarded.

7.3 Rectification by multiple homographies

In the previous section it is shown that traditional stereo rectification techniques fall short when used for registering objects with the thermal-visible imagery of the proposed set-up.

The poor performance of traditional stereo vision techniques might be due to the short range of the registered scenes and the differences in field-of-view of the thermal and visible cameras. The short range of the scene introduce noticeable parallax when trying to apply a single homography or rectification view, as seen in the previous section.

This section will treat a different approach to image registration by introducing multiple homographies that registers multiple subsets of the scene. From the above treatment on epipolar geometry is is known that a single homography is able to accurately map corresponding points between two views if the points are residing on a world plane. If points are outside the registered plane π, the resulting parallax of the mapping is given by: [Hartley and Zissermann, 2003, Chap. 13]

$$\mathbf{x}' = H\mathbf{x} + \rho e' \tag{7.11}$$

Where ρ is the parallax relative to the homography and $\mathbf{e}'$ is the epipolar point of the second view. The scalar ρ might be interpreted as the depth from the plane π.

If only one homography is used for registering the imagery, only one plane is accurately registered. When using the RANSAC or Least-Median-of-Squares methods to find the best fitting homography of corresponding point pairs, such as in OpenCV, the points chosen for the homography are the points that span the largest possible plane in world coordinates. The homography will estimate a good correspondence for points close to the plane, but for points further away from the plane, the parallax will be noticeable according to Equation 7.11.

That this holds true for real world points is seen from Figure 7.3. For the generated calibration data of Scene 2, 20 homographies have been generated, scattered across the scene. Then, for each of the calibration points used for the camera calibration, the distance to the closest homography in world coordinates is compared with the actual transfer error induced by the closest homography when used for mapping the image point to thermal coordinates. As might be seen from Figure 7.3, there exists a relationship between the distance to the homography and the parallax relative to the homography as claimed in Equation 7.11.

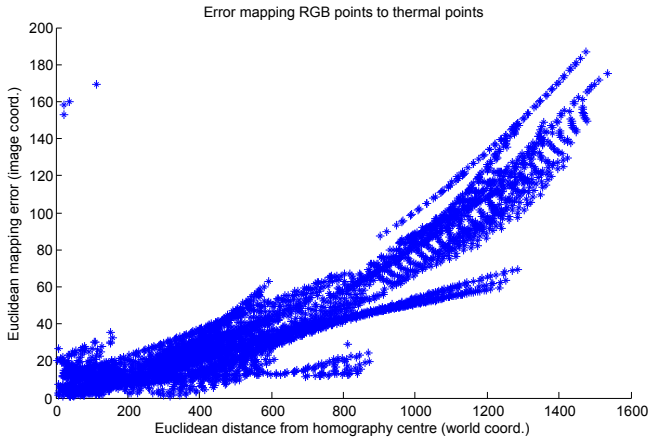


Figure 7.3: Transfer errors versus the distances to the closest homography for calibration points of Scene 2.

By using multiple homographies, the registration parallax will be reduced as the distances to the closest planar homography are reduced. The distances to the nearest

homographies vary according to the number of homographies and positioning of homographies between world points. If a significantly high number of homographies is chosen and the homographies are carefully distributed throughout the scene, the parallax for points lying outside a registered plane will be sufficiently small to provide an accurate pixel-to-pixel registration for all points inside a defined range. For points placed between multiple homographies, the positioning of corresponding points might be calculated by a weighted sum of neighbouring homographies.

However, a registration method based on multiple homographies is subject to several deliberations:

- How many homographies are needed for an accurate registration?
- How should the homographies be positioned?
- How should the homographies be generated?
- How should the interpolation between homographies be designed? Should one always choose the closest homography?
- Which measure defines the 'closest' homography?

Previous work of using multiple homographies is limited to using a very small subset, usually two or three, homographies for multiple regions of interest in the scene. In this context, the goal is to create a set of homographies that enables the entire scene to be registered, which in this context is the range between 1 and 4 metres from the cameras. This is an approach which, to the best knowledge of the author, has retrieved very sparse attention.

Therefore, there do not exist an unambiguous answer to the questions stated above. Some of the answers are provided by design choices of the algorithm, while others are included as test parameters for the acceptance test.

The rectification by multiple homographies will only work if homographies are scattered evenly through the regions of the scene for which we want to register the images. Thus for each scene, multiple homographies should be present in the range of $1 - 4$ metres from the camera. There exist an infinite amount of solutions on how the homographies should be scattered and thus many optimal solutions might exist based on the number of homographies chosen for the scene.

In order to compute 2D homographies between two camera views, one must know at least 4 point correspondences of the thermal and visible cameras. These correspondences are already generated in the camera calibration stage as the calibration rig is swept through the entire scene. If individual frames of these calibration correspondences are picked so that the entire scene is covered by the union of the correspondences, one might assemble the correspondences of the calibration boards in k clusters and from those clusters generate k homographies. These homographies will relate thermal and visible points within a range of the registered plane. When one wants to map points from the RGB camera to the thermal camera or vice versa, the closest homography or a subset of homographies is chosen to provide the best available point map. The process of creating this map is summarized in Figure 7.4. In the following, the steps of Figure 7.4 will be treated one-by-one.

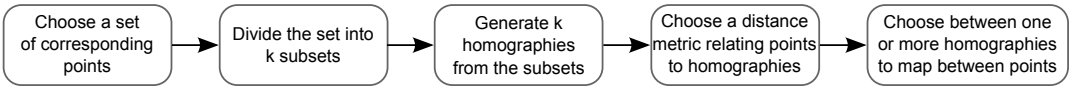


Figure 7.4: Generating and choosing between multiple homographies, in brief.

7.3.1 Generating a set of corresponding points

As concluded above, the homographies should be distributed evenly throughout the entire scene at different positions and depth relative to the camera. This requirement is met by extracting corresponding point pairs of images of the calibration board used for the thermal-visible calibration. During the acquisition of the calibration images, the board is swept through the entire scene to provide an accurate registration everywhere inside the defined range. The corresponding point pairs are generated from at least 50 different views of the calibration board from two calibration sequences, and thus it is highly probable that the point correspondences cover the range of the scene we want to register.

However, the point correspondences obtained from the camera calibration only gives the coordinates in the image plane. In order to find the 3D world coordinate of the point, the corresponding point in the depth image is found through the RGB-to-depth registration of Section 5.2.2 and the depth of the point is looked up in the depth image. The depth is relative to the camera coordinate frame and might be interpreted as the z-value of the world coordinate of the point. However, the image points are still in *image coordinates* and thus needs to be transferred to world coordinates.

In order to transfer the points, the we might use the formula for mapping world points in the camera coordinate frame to image coordinates by means of the intrinsic parameters of the camera:

$$\begin{pmatrix} xw \\ yw \\ w \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{7.12}$$

where x and y are the image coordinates and X, Y, and Z are the world coordinates in the camera frame.

As we already know the value of the depth Z, one may derive the world coordinates X and Y from the following formulas:

$$X = \frac{xZ - c_x Z}{f_x} \tag{7.13}$$

$$Y = \frac{yZ - c_y Z}{f_y} \tag{7.14}$$

This operation is applied for each image point of the calibration data.

The data extracted from calibration sequence 1 and 2 of Scene 3 is seen from Figure 7.5.

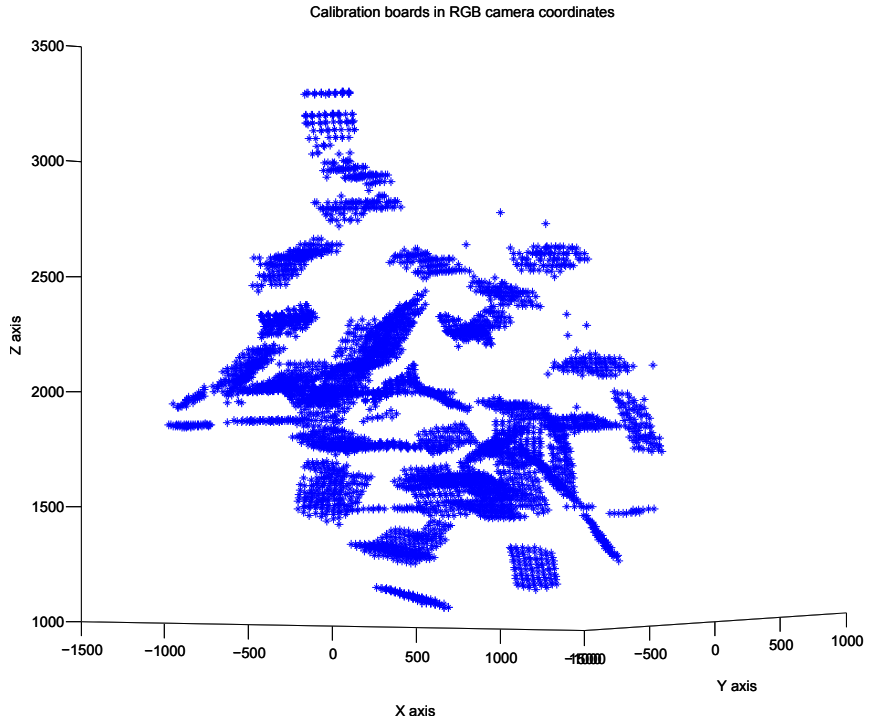


Figure 7.5: Point cloud obtained from multiple views of the calibration board, shown in camera coordinates of the RGB camera. The depth of the points is obtained through the depth sensor of the Kinect. The camera centre is located at the origin of the coordinate system.

7.3.2 Positioning of point subsets

The positioning of the k homographies may be chosen manually or through a classification algorithm. In this case, the *k-means* clustering algorithm is chosen to divide the set of points into k clusters. From these clusters, k unique homographies are computed. The k-means clustering algorithm attempts to minimize the following cost function: [Hartigan and Wong, 1979]

$$\sum_{j=1}^{K}\sum_{i=1}^{N}\left(\text{argmin}_j\|\mathbf{x}_i^j - \mathbf{c}_j\|_2^2\right) \tag{7.15}$$

where $\mathbf{x}_i^j$ is the coordinate of the ith point residing in the jth cluster and $\mathbf{c}_j$ is the centre of the jth cluster.

The k-means clustering algorithm is not guaranteed to find the optimal solution though, and thus multiple runs with different initializations of the clusters should be conducted to find a solution that is closer to or is the global minimum of the cost function. The algorithm uses the Euclidean distance to minimize the cost function which might not be optimal if the dimensions of the data are skewed. Therefore, data should be normalized by some metric before clustering.

The generated point cloud is normalized by dividing each coordinate by the standard deviation of the respective dimension, thus resulting in a dataset with a standard deviation of unity in each dimension.

The normalized point coordinates are led into the OpenCV implementation of the k-means algorithm, `kmeans`. The algorithm generates k random centres and iterates

from these to minimize the cost function. Due to the fact that the k-means algorithm is vulnerable to the positioning of cluster centres, the algorithm is repeated for 100 random sets of clusters. At the end of the trials, the set of clusters representing the smallest cost function is chosen. Each coordinate in the point cloud is assigned to a unique cluster.

The clustering is applied for world coordinates of the calibration points seen from the RGB camera. However, these coordinates are easily mapped to world coordinates seen by the thermal camera. This mapping is described in Section 7.3.4.

7.3.3 Generating the homographies

Once the k-means algorithm has found k clusters and classified the points into these clusters, it is time to generate k homographies. From the theory of the k-means algorithm, we know that points classified in one cluster are lying close together in Euclidean distance. The density of the cluster is affected by the distribution of points and the number of clusters chosen. When the number of clusters is sufficiently high, the world points will lie so close that it is a reasonable assumption to say that a large subset of the points constitutes a plane for which a homography might be generated. However, we would want a homography that maps image points between the RGB and thermal images, not between their world coordinates. Luckily, the coordinates of the image frames are easily generated as they constituted the basis for generating the world points in Equation 7.14.

As we have a one-to-one mapping between world points and the corresponding image points of the RGB and thermal cameras, the image points might be classified according to the k clusters. From these subsets of point correspondences, k homographies are generated using the `findHomography` function of OpenCV.

The 2D homography H has 9 entries but only 8 degrees of freedom, which means that at least 4 point correspondences are required to find a 2D homography between images. However, as the view of the calibration rig typically consists of 96 point correspondences we have an overdetermined solution, and as several chessboard views may be united to create one cluster, even more point correspondences occurs. Due to image noise and errors of the corner extraction process, the point correspondences do not represent an exact map between the true world coordinates of the corners and their depicted counterparts. Furthermore, as not all points in a cluster are lying on the same plane, there exist no single solution for finding a homography that fits all points.

Robust homography estimation Thus, one must apply a robust method for finding a homography that fits the largest possible subset of points. This is possible by using the RANdom SAmple Consensus (RANSAC) algorithm that is readily implemented in OpenCV. The RANSAC algorithm for computing a 2D homography uses the following approach: [Hartley and Zissermann, 2003, chap. 4], [Bradski, 2000]

1. Select 4 random point correspondences and compute the 2D homography H.
2. For each point correspondence in the point set, calculate the re-projection error

with respect to the homography H.

3. Compute the number of inliers. An inlier is defined as a point whose re-projection error does exceed a *threshold* set by the user.

The above procedure is repeated N times, a number that is determined adaptively at runtime [Hartley and Zissermann, 2003]. When the runs are complete, the homography with the largest number of inliers is chosen. The RANSAC algorithm is illustrated in Figure 7.6 by the process of fitting a line through a noisy point set.

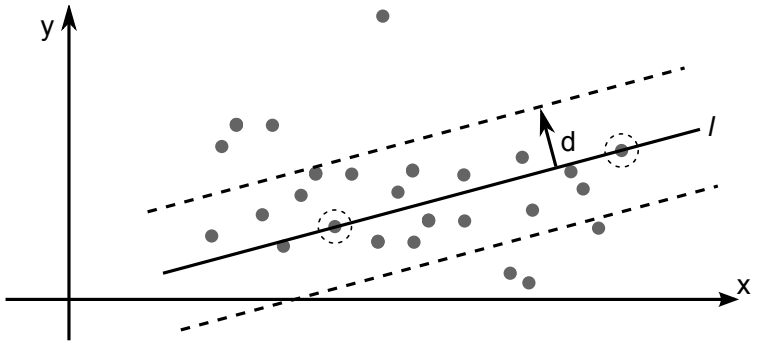


Figure 7.6: The RANSAC algorithm fitting a 1D line onto a subset of points. The two encircled dots are chosen as basis for the line l. For this line, the number of points that lie inside the thresholded distance d are counted. The algorithm is repeated N times and upon completion, the line with the greatest number of inliers is chosen.

The threshold used for filtering the outliers might be chosen arbitrarily and should be adapted to the current level of noise in image data. The OpenCV documentation suggests a threshold between 1 and 10, and such, a threshold of 3 is chosen for the current implementation of `findHomography`. The generated homographies only apply for the mapping of points from the visible to the thermal modality. In order to provide the the opposite mapping, the homographies are inverted.

7.3.4 Choosing a distance metric

The current implementation of the clustering and homography generation does not generate information on the actual planes that are the theoretical framework behind the computed homographies. However, we know the centre of the cluster and the points for which the homography is generated, which may be used to constitute a useful depth measure. The simplest measure of the distance between homographies and points is to use the Euclidean distance between the world coordinates of the cluster centres and the point. However, the Euclidean distance between the point and the cluster centre might not be an optimal solution since the valid region of the homography spans a plane, not a point in world space. With a purely Euclidean distance measure, a point might be substantially close to a plane spanned by a homography but far away when measuring the distance to the centre of the cluster.

A better distance measure when one does not know the actual plane is the *Mahalanobis distance* given by: [Mahalanobis, 1936]

$$D_M = \sqrt{(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)} \tag{7.16}$$

where $\mathbf{x}$ is a point vector, μ is the mean vector of the dataset, and Σ is the covariance matrix of the dataset.

The mean and covariance should in this respect be treated as the mean and covariance of the points classified in a cluster. The mean value of the cluster is by definition the cluster centre. The covariance matrix is computed from the points spanning the homography generated in Section 7.3.3. This subset of points used for computing the covariance matrix might not necessarily consist of all the points of the given cluster but only the points returned from `findHomography` which are classified as inliers by the RANSAC-estimation of the homography.

The mean and covariance should in this respect be treated as the mean and covariance of the points classified in a cluster. The mean value of the cluster is by definition the cluster centre. The covariance matrix is computed from the points spanning the homography generated in Section 7.3.3. This subset of points used for computing the covariance matrix might not necessarily consist of all the points of the given cluster but only the points returned from `findHomography` which are classified as inliers by the RANSAC-estimation of the homography.

The Mahalanobis distance measure mimics the process of forming the homography. If, for instance, the majority of the variation of the points lies along a line or plane in space, the RANSAC algorithm is likely to estimate a homography for a plane that is defined by points residing on a this line or plane. When computing the Mahalanobis distance of points residing on this plane, the distance will be substantially smaller than if the point was residing outside the plane, but with the same Euclidean distance to the cluster centre. This relationship is depicted in Figure 7.7 for a multivariate Gaussian.

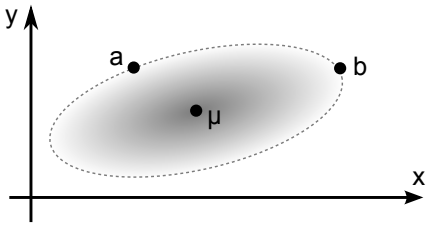


Figure 7.7: Mahalanobis distance for a multivariate Gaussian with centre μ. Even though the points a and b are located at different Euclidean distances from the centre, they share the same Mahalanobis distance.

Distances wrt. the thermal camera The above mentioned methods of measuring the distance between world points does only apply if one knows the depth for every image point. Otherwise, it is only possible to back-project image points to world coordinates up to a one-dimensional ambiguity, thus mapping points to lines. This is indeed the case for points obtained from the thermal camera. For image points on the thermal camera, depth information might only be provided if the corresponding point in the visible camera is known - but in order to provide this correspondence, the depth information is required to select the closest homography!

This relationship might be solved by applying an iterative approach for which the depth of a pixel is guessed, projected to RGB coordinates, then extracting the actual depth, and then re-projecting to thermal coordinates where the mapping error might

be used for a second guess of the depth of the object. This process is repeated until the re-projection error is below a certain threshold. However, the implementation of such an algorithm is beyond the scope of this project, and we will focus on a simpler implementation.

When building the point cloud from views of the calibration rig seen through the visible camera and depth sensor, a parallel point cloud is built that projects points of the thermal camera into the world space by a fixed depth. This essentially denotes that we are projecting the image plane onto a broader world plane, but this is necessary in order to use the same metrics across both modalities. The 'thermal' point cloud is generated trough the intrinsic properties of the thermal camera. This means that in order to measure the distance from thermal image points to the closest homography, centres and covariance matrices defined above have to be recalculated with respect to the coordinates of the thermal point cloud.

Thus, even though a visible and a thermal point is affected by the same homography when the points are mapped to the other modality, they will feature different distance metrics. This is the direct cause of the difference in depth information provided by the two modalities.

7.3.5 Interpolation between homographies

The chosen distance measure enables us to find nearby homographies and sort them according to their distance to a current point. The homographies map points residing in their defined plane perfectly, but how should they behave when points are positioned outside the planes? Should only the closest homography be selected or should we interpolate between a subset of homographies?

It is preferable not to choose only the closest homography for certain reasons. If a point is situated in the middle of two planes spanned by two homographies, both homographies should contribute equally to the mapping of the point. Furthermore, the mapping between adjacent points should be smooth, which means that a smooth and continuous transition between homographies is required. If we pick only the k nearest neighbours, the influence of one homography might go from substantial to zero within a few pixel if the homography is no longer apparent in the set of the k nearest homographies.

Instead, it is chosen to use a weighting scheme of the homographies based on the relative distance from a point in space to the nearest homography. This means that homographies close to a point will receive a high weighting compared to homographies further away and that points which are *very* close to a homography will be mapped almost entirely by this homography.

The proposed scheme of interpolating between the homographies at an arbitrary world point is the following:

1. For given world coordinate $\mathbf{X}_j$, compute the distances to all homographies given the Mahalanobis distance defined in Section 7.3.4. In order to exaggerate the differences in distance between homographies, the distances are squared.
2. Find the minimum distance, $d_{\min}$.

3. For all homographies, compute the relative distance with respect to the closest homography:

$$w_{\text{rel}i} = \frac{d_{\min}}{d_i} \tag{7.17}$$

4. Sum the weights of the relative distances

5. Divide each weight by the sum of weights. Now, the weights sum to unity:

$$w_i = \frac{w_{\text{rel}i}}{\sum_{i=1}^{N} w_{\text{rel}i}} \tag{7.18}$$

With this approach, we favour homographies close to a point by penalizing every other homography by the distance ratio to the closest homography. When moving between homography centres, the transition between homographies is smooth. However, even if the Mahalanobis distance is squared, homographies 'far away' from the point might still be weighted by 5–10 %. This is indeed not wanted, as homographies will induce significant parallax the further the point is located from the image point according to Equation 7.11. Therefore, a window function is introduced to eliminate homographies which are further away than a distance threshold d_{window}. The window function should be 'invisible' to distances within the allowed threshold and should make a smooth transition for distances outside the range. These characteristics apply for the Tukey window function described by the following equation: [Tukey, 1967]

$$W(x) = \begin{cases} \frac{1}{2}\left(1 + cos\left(\frac{2\pi}{r}(x - r/2)\right)\right) & 0 \geq x < \frac{r}{2} \\ 1 & \frac{r}{2} \geq x < 1 - \frac{r}{2} \\ \frac{1}{2}\left(1 + cos\left(\frac{2\pi}{r}(x - 1 + r/2)\right)\right) & 1 - \frac{r}{2} \geq x \geq \frac{r}{2} \end{cases} \tag{7.19}$$

The Tukey window function consists of a rectangular window for the middle $1-r$ par of samples and is a partial cosine function for the first and last $r/2$ of samples. Thus, it makes a good compromise of not affecting the distance values and making a smooth transition at the end of the desired range. The parameter r is set at 0.1, which means that 10 % of the window is in the cosine-tapered section. The time-domain characteristic of the Tukey window at $r = 0.1$ is seen from Figure 7.8.

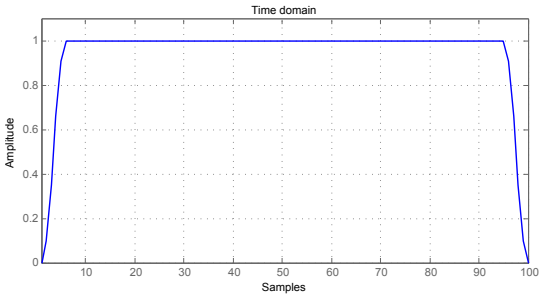


Figure 7.8: Time-domain view of the Tukey window function with $r = 0.1$.

The usage of the window function affects the distance calculation in Section 7.3.4 which is Step 1 in the process described above. As we want to penalize distances

larger than d_{window}, the measured distance is divided by the scaling of the window function:

$$w_i = \frac{w_i}{W(w_i)} \tag{7.20}$$

Distances outside the range of d_{window} are fixed at a fixed number, $1 \cdot 10^9$, instead of being clamped to infinity.

As seen from Figure 7.8, the window function effectively suppresses distances outside the allowed threshold. However, this leaves one question open: How to set the threshold d_{window}?

The optimal threshold should (1) be so low that distances that result in the weighting of homographies with significant parallax are suppressed but (2) be so large that every point within the scene boundaries should be associated with at least one homography. Thus, the threshold is closely related to the distribution of the world points used for clustering and the number and positioning of the generated clusters.

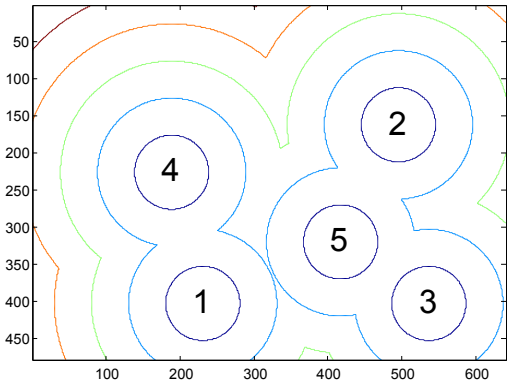
If condition (2) is to be held, the distances to the closest homography should be calculated for all points used for the clustering process of Section 7.3.1. The largest distance should then be picked as the window length, d_{window}. In this way, every point will have at least one homography within its reach. For each scene, the distances to the closest homography by using the training set is calculated, and the maximum distance is found. As the points used for the validation of the algorithm will likely feature a longer distance to the nearest homography than the training points, the maximum distance is multiplied by a factor of 1.5 which is found empirically. Indeed, two window lengths are specified: one applying to points in the visible image and one applying to points in the thermal image.

Visualization of the weighting scheme

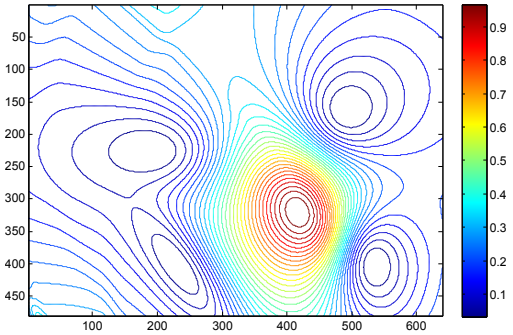
The interpolation algorithm is difficult to visualize since it is implemented for 3D points. However, when scaled in 2 dimensions, such as the case for thermal image points, the performance of the algorithm might be shown. The following describes the implementation in MATLAB, which is equivalent to the implementation of the algorithm in OpenCV.

Five clusters have been generated throughout the image plane as seen from Figure 7.9a. The entries of the covariance matrices of the clusters are scaled between 0.2 and 0.6, and the clusters therefore have different *reach* in the x- and y directions, modelling bivariate Gaussian distributions.

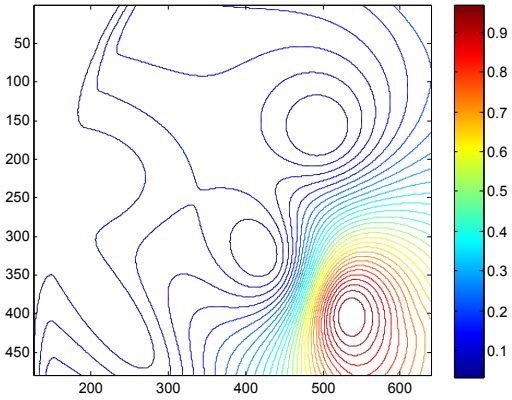
In Figure 7.9b and 7.9c, the impact of cluster 5 and 3 on neighbouring points is seen. The *impact*, or the percentage of the total weighting of homographies that the corresponding homography represents at the current pixel, is seen from contour plots of Figure 7.9b and 7.9b. The contour scale goes from blue, which represents a value of $[0; 0.30]$, to brownish red, which represents a value close to 1. It is seen from both contour plots that points in the vicinity of the cluster centres are influenced almost entirely by the current homography, which indeed should be the case. From the contour plot of cluster 3 in Figure 7.9c, the impact of the window function is



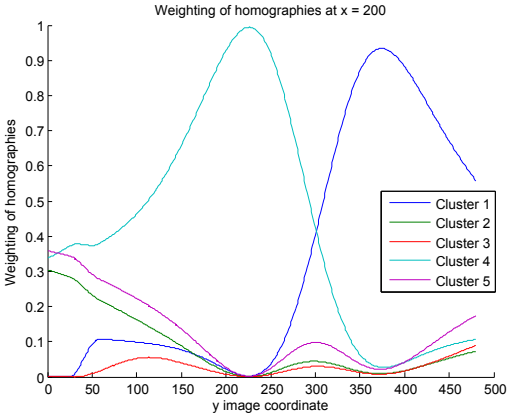
(a) Distance transform of the cluster centres, measured in Euclidean distance



(b) Contour plot of distances for homography 5



(c) Contour plot of distances for homography 3



(d) Cross-sectional view of the weighting of homographies at $x = 200$

Figure 7.9: Depth images from the Kinect. The 8 bit-image is the saved depth image from the Kinect in order to comply with performance requirements.

seen in the upper left region of the image. Before the impact of the window function, the weighting of the homography at the upper left corner is below 0.10, however, the impact of the window slowly slides the weight of the homography to zero.

Figure 7.9d shows a cross-sectional view of the weighting of the homographies at the line $x = 200$. It is easily seen from the Figure that the line goes straight through cluster 4 and lies very close to cluster 1, as the weighting of these homographies are peaking at their respective centres. It is also seen how the window function cancels the contributions from cluster 1 and 3 as the distances to these centres are outside the allowed range at the beginning of the line.

Summary on the image registration

This chapter has treated the design of two significantly different image registration techniques: stereo rectification and rectification by multiple homographies. The technique of stereo rectification is widely applied in the computer vision community in stereo configurations with visible cameras. Therefore, one would assume that an implementation on the thermal-visible imagery might provide good results. The stereo rectification is implemented in both the calibrated and uncalibrated case, and a method to automatize the search for corresponding points along the rectified epipolar lines has been proposed. However, results show that the stereo rectification methods provide a poor estimate of rectifying the thermal-visible imagery of the proposed set-up.

A technique on generating multiple homographies for each scene is proposed. The homographies are generated through the sets of corresponding points that are divided in clusters according to the k-means clustering algorithm. Image points residing either in the visible or thermal modality are mapped to the other modality by a weighting of the closest homographies available at each point. The distance from a image point to a homography is defined by projecting the image point to world coordinates and using the Mahalanobis distance function evaluated at the points that span each homography.

Both rectification methods are implemented in the OpenCV framework for C++. The code is available in the CD provided[1] and is described in brief in Appendix B.

In the next chapter, the acceptance test will evaluate the performance of the rectification by multiple homographies.

[1] Code/DualCalibration/DualCalibration.sln

Part III

Verification and conclusion

Chapter 8

Acceptance test

The acceptance test specification of this chapter is validating the requirements set in Chapter 3 upon the methods defined in the acceptance test specification in Chapter 4. The beginning of this chapter will assess the requirements of the image acquisition platform, whereas the latter part of will evaluate the performance of the registration algorithm.

8.1 Image acquisition platform

8.1.1 Physical platform

The creation of the physical image acquisition platform is described in Section 5.1 and consists of the Microsoft Kinect for XBOX and the AXIS Q1922 thermal camera. The core specifications of the cameras are listed in Table 8.1.

	Microsoft Kinect for XBOX	AXIS Q1922
Modality	Visible, depth	Thermal
Resolution	640x480	640x480
Frame rate	15 FPS	30 FPS
Spectral range	Visible*	8 – 14 μm

Table 8.1: Properties of the thermal and visible cameras. The Kinect is able to record frames at up to 30 FPS. Due to performance issues though, the actual frame rate is decreased to 15 FPS. *The spectral range is not available in any data sheet for the Kinect, but as for any common RGB-sensor, the camera captures visible light.

As seen from the specifications of Table 8.1, the cameras fulfil the requirements on the minimum resolution and frame rate of the platform. The baseline of the platform is 70 mm, thus fulfilling the requirement of a maximum distance between the thermal and visible cameras of 100 mm. The AXIS thermal camera has a spectral range in the LWIR range of 8 – 14 μm, which is perfectly capable of 'seeing' objects in the temperature range of -10°C – 50°C. The temperature range will not be tested though, and we will rely on the properties of the sensor provided by the manufacturer.

8.1.2 Synchronization

In correspondence with the accept test specification, three scenes have been recorded with the acquisition platform. The acquired frames for each calibration sequence of scene 1 – 3 has been synchronized using the measures described by Section 6.1. The synchronization lag is measured in Appendix C.1.1 and the results are replicated in Table 8.2 for convenience.

		Synchronization lag			Largest frame lag	
Scene	Calibration sequence	Max.	Mean	Std.dev.	Thermal	RGB
1	1	31.3 ms	9.31 ms	8.71 ms	374.4 ms	78.0 ms
	2	46.8 ms	9.43 ms	8.70 ms	78.1 ms	93.6 ms
	3	46.8 ms	9.26 ms	8.40 ms	358 ms	93.6 ms
2	1	31.2 ms	9.07 ms	8.48 ms	109 ms	93.6 ms
	2	31.2 ms	9.35 ms	8.99 ms	129 ms	93.6 ms
3	1	62.4 ms	9.53 ms	9.30 ms	218 ms	93.6 ms
	2	46.8 ms	9.34 ms	8.83 ms	359 ms	94.6 ms
	3	31.2 ms	9.13 ms	8.62 ms	359 ms	93.6 ms

Table 8.2: Synchronization lag between and frame lag within cameras of the 8 calibration sequences. The synchronization lag is the lag between synchronized frames of different modalities, and the largest frame lag is the inter-frame distance between synchronized frames of the same modality. The depth frames are not mentioned here as these frames feature a perfect synchronization with the RGB frames - and thus, the terms RGB and depth are interchangeable in this table.

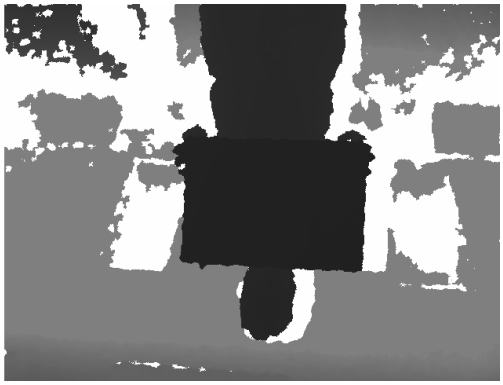
It is seen from the table that the requirement of a minimum synchronization lag of 40 ms is not met in four of the eight calibration sequences. This is not a fault of the synchronization algorithm but a result of sudden 'jumps' in the frame rate of either the thermal or RGB cameras. This jump in the frame rate does indeed lead to inter-frame lag up to 374 ms, which is the case for two thermal frames in calibration sequence 1 of scene 1. However, these lags will only lead to excessive synchronization lag if the RGB frames are also lagging - otherwise, the RGB frames that resides in between the laggy frames will be discarded. This is the case in four calibration sequences where a synchronization lag up to 62 ms is measured. These spikes in the synchronization lag does only occur once or twice through a calibration sequence but in these occurrences, objects in motion will occur out of sync, and point correspondences extracted from these frames will be degenerate.

However, the average performance of the synchronization of the multi-modal frames is significantly below the threshold of 40 ms. The mean of the synchronization lag lies consistently between 9.07 – 9.53 ms and the standard deviation of the lag never exceeds 10 ms. But notwithstanding the good performance on the average, the requirements of the synchronization lag are only partially met.

8.1.3 Depth information

The Kinect for XBOX camera is equipped with a built in depth sensor that estimates the depth of objects for 0.8 – 4 metres according to the specifications of the

manufacturer [Eisler, 2013]. The requirement specification defines a minimum range of 1 – 4 metres which is verified by placing an object at exact these distances, as seen from Figure 8.1.



(a) Depth image of object at 1 metre from the sensor. The depth value returned from the Kinect sensor is 863 mm.

(b) Depth image of object at 4 metres from the sensor. The depth value returned from the Kinect sensor is 3074 mm.

Figure 8.1: Depth images from the Kinect at the bounds of the specified interval.

8.1.4 Depth registration

The depth images are factory registered to the RGB images of the Kinect. The registration is saved by the image acquisition platform and is processed further to provide a mapping between pixels in the depth image to pixels in the RGB image and vice versa. An example of this map is seen from Figure 5.4, fulfilling the requirement on the depth registration.

8.1.5 Summary

The requirements for the image acquisition platform set in Table 3.1 has all but one been fulfilled. For 50 % of the calibration sequences, the synchronization lag is greater than 40 ms, for a few frames, thus only partially fulfilling the requirement on the synchronization lag.

The compliance with the requirements for the image acquisition platform is listed in Table 8.3.

8.2 Thermal-visible registration algorithm

This section will test the requirements for the performance of the registration algorithm between the thermal and visible images. As stated in the requirement specification, the performance of the depth registration will not be tested, and we thus rely solely on the built-in registration between the visible and depth images as provided by the Kinect for Windows SDK. Although multiple image registration techniques based on stereo rectification and multiple homographies have been presented, only

Parameter	Requirement	Test result	Passed
Synchronization lag	$\leq$ 40 ms	$\leq$ 62.4 ms	Failed
Minimum frame rate	$\geq$ 12.5 FPS	15 FPS	Passed
Thermal range	-10°C – 50°C	Factory spec.	Passed
Minimum visible resolution	640x480	640x480	Passed
Minimum thermal resolution	320x240	640x480	Passed
Baseline between cameras	$\leq$ 100 mm	70 mm	Passed
Depth information	Available	Available	Passed
Depth registration	Stored	Stored	Passed
Depth range	1 m – 4 m	1 m – 4 m	Passed

Table 8.3: Compliance with the requirements for the image acquisition platform.

the rectification by multiple homographies is evaluated here. The stereo rectification algorithms are evaluated in Appendix D and it is found that the performance of these algorithms is not within the acceptable range. Therefore, these are not included in the acceptance test.

8.2.1 Scenes

Based on the definitions of Section 4.2.1, three scenes are created. These scenes are seen from Figure 8.2. For scene 1 and 3, three calibration sequences are conducted, whereas only two calibration sequences are conducted for scene 2. This is not necessarily a problem though, which is discussed in the next section.

8.2.2 Obtaining corresponding point sets

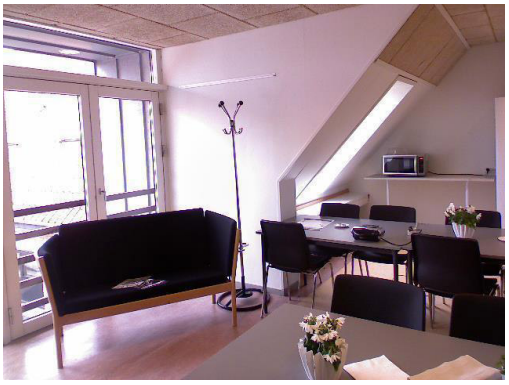
For each calibration sequence, between 25 and 30 frames have been picked, and corners of the calibration rig is manually extracted according to the process described in Appendix C.2. Due to the shortage of calibration sequences of scene 2, an additional 15 frames have been extracted from calibration sequence 2. Although these frames are taken from the same calibration sequence, they are taken with different time intervals and starting conditions in time. Consequently, these frames may be used for the validation of the registration algorithm for scene 2, and will, in this framework, be defined as calibration sequence 3.

Number of validation points

The number of active frames extracted for each calibration sequence is listed in Table 8.4. Here, we distinguish between the number of frames extracted from each frame and the number of *active frames* from which the corners of the calibration rig have been successfully extracted. The multi-modal calibration rig is described in Section 6.2.2 and consists of 96 corners, which translate to 96 point correspondences between

(a) View of Scene 1

(b) View of Scene 2



(c) View of Scene 3

Figure 8.2: The three scenes created for the test and validation of the registration algorithm.

Scene	Calibration sequence	Active frames	Number of point correspondences
1	1	25	2312
	2	24	2304
	3	21	1968
2	1	22	2112
	2	22	2112
	3	15	1440
3	1	28	2584
	2	28	2584
	3	24	2208

Table 8.4: Number of active frames and the number of point correspondences extracted for each calibration sequence. More frames might be extracted from a calibration sequence than the above listed 'active frames', but these are discarded due to either synchronization issues, poor visibility of the corners, or problems with the positioning and pose of the calibration rig.

the thermal and visible images. However, the calibration rig is not fully visible in all views, and therefore one cannot obtain the total number of point correspondences by multiplying the number of active frames by 96. It is seen from the entries of Table 8.4 that the number of point correspondences for each calibration sequence exceeds the required amount of calibration points by at least 40 %.

If one was sure that the internal positioning of the cameras was constant between the different scenes, a bigger dataset for training and testing the registration algorithm might be created from a concatenation of the calibration sequences of all three scenes. However, as this is not guaranteed, the registration algorithm is trained and tested separately for each scene.

Distribution and range of validation points

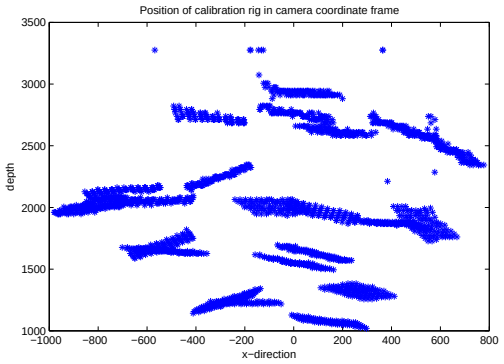
The requirements on the distribution of the corresponding point sets are harder to assess. The method chosen here is to manually validate the point clouds extracted from the visible and depth information available of the corresponding point sets. Furthermore, the set of active frames are validated by manual inspection. In order to generate the point cloud, the image coordinates of the calibration points in the RGB image are back-projected to world coordinates by the intrinsic camera parameters of the RGB camera and the depth obtained by the depth sensor of the Kinect. The point cloud is thus given in the coordinate frame of the RGB camera where the origin is the camera centre. The claim from Microsoft is that the depth obtained from the Kinect might be interpreted in mm [Microsoft Developer Network, 2013b]. It is seen from the measurements of Section 8.1.3 that this is indeed a poor estimate. The test object placed 1 m from the camera is measured as having a depth of 863 mm whereas the object placed at 4 m is measured as 3074 mm. This indicates that Kinect depth measurements at 3 m should indeed be translated to a real-world distance of 4 m. The validation of the depth measurements from the Kinect sensor is beyond the scope of this project so in this context, we will use the raw sensor values from the Kinect, however with the range of the measurements in mind.

The raw data for generating the point cloud are found in Dataset/Scene x/y/CalibrationHelper/pointCloudRgb.txt where x and y denotes the scene and calibration sequence number, respectively. The point cloud is easily visualized with the tool of choice. In this context, the `dlmread` and `plot` functionality of MATLAB is used.

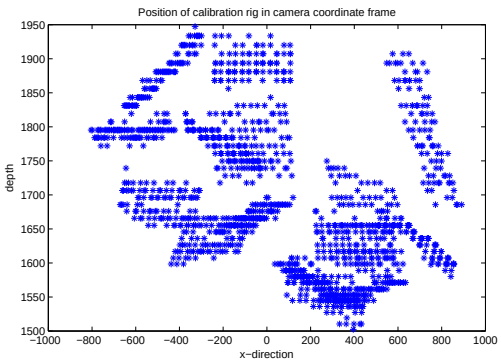
Although the three-dimensional point cloud is indeed very difficult to visualize on paper, the point clouds of the validation sequences for scene 1 – 3 are shown in Figure 8.3. The figures show the x- and z-coordinates of the point clouds and give an indication of the positioning of the calibration rig in the plane $y = 0$. From the point clouds it is seen that the depth range of the validation points of scene 2 is indeed limited compared to the ranges for scene 1 and 3.

Through an inspection of the point clouds and the corresponding images of the calibration rig, it is verified that both the training and validation points are distributed equally through their respective scenes.

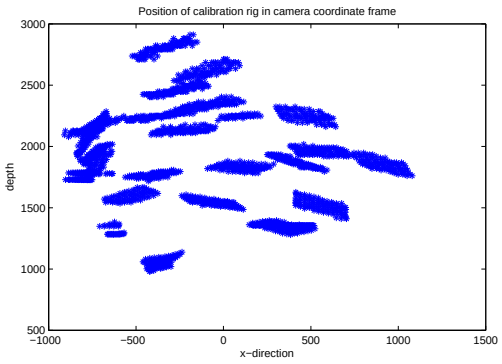
The range is formalized even further in Table 8.5. The table shows the 1 and 99



(a) Validation points for scene 1



(b) Validation points for scene 2



(c) Validation points for scene 3

Figure 8.3: Distribution of the validation points for all scenes shown in world coordinates in the reference frame of the RGB camera. As only a 2D view is provided, the points are projected to the plane y = 0. The validation points for each scene are corresponding to the 3rd calibration sequence.

% quantile of the depth of the corresponding point sets of each calibration sequence. From the entries of the table it is easily seen that the range requirement for scene 2 is fulfilled as the points are within the range 1700 mm ± 200 mm. For scene 1 and 3 it is seen that the points do indeed lie inside the range 1 – 4 m. However, it is noticed that points in some scenes are not contained in the full range of up to 4 m, which translates to a Kinect measurement of 3074 mm. This property is due to the nature of the scenes which does not allow the calibration rig to be placed further away. The conditions for the range requirement are therefore met for all scenes.

Scene	Calibration sequence	1 % quantile	99 % quantile
1	1	1142 mm	2882 mm
	2	1182 mm	2912 mm
	3	1056 mm	2942 mm
2	1	1470 mm	1894 mm
	2	1510 mm	1894 mm
	3	1536 mm	1921 mm
3	1	1067 mm	2561 mm
	2	1298 mm	3276 mm
	3	1018 mm	2852 mm

Table 8.5: The 1 and 99 % quantile of the depth of the corresponding point set for each calibration sequence. The depth is equal to the z-entry of the point cloud.

8.2.3 Registration accuracy

As the registration by multiple homographies is indeed a non-deterministic algorithm, we need to evaluate each adjustable parameter k times and evaluate the algorithm based on the average performance of the runs. The image registration by multiple homographies is characterised by the n number of homographies for a scene, a number which might be chosen arbitrarily. Thus, in order to find the number of homographies which contributes to a better image registration, the algorithm is run for each scene with a varying number of homographies in the range from 1 – 30.

In Section 7.3.3, the RANSAC-threshold for finding the homographies from each cluster of point correspondences is set to 3. In order to challenge this threshold, the performance is tested with both the threshold of 3 and a threshold of 10, which is the largest recommended threshold of OpenCV. Future work will assess the impact of the threshold through further investigation.

The acceptance test specification states that 20 runs should be conducted for each varying parameter of the registration algorithm, which for three scenes gives us $3 \cdot 30 \cdot 20 \cdot 2 = 3600$ runs. Furthermore, we would like to test the registration performance on both the training data and the validation set.

As stated in the requirement specification, the baseline chosen to compare the registration against is the global registration method, which might be interpreted as the multiple homographies technique with only one cluster. The global registration relies on the OpenCV function `findHomography`, for which the the best RANSAC-threshold in the interval 1 – 30 is found for each scene.

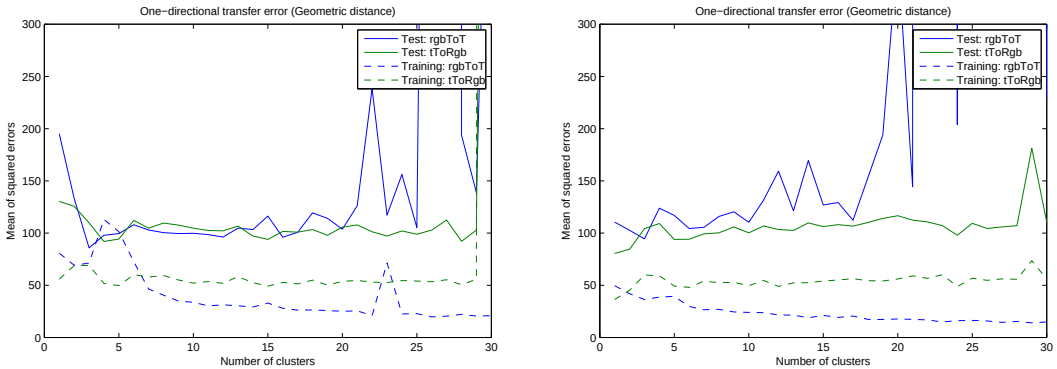
For each run, a log file is created with the error in pixels between the mapping and the ground truth points. Given the size of the dataset and the proportion of the tests, it is not feasible to show all results here. For each batch of 20 runs testing the performance of the registration algorithm with n clusters, the average performance is listed in text files on the CD Logs/Hom/ .

For each scene, the registration performance by 5, 10, and 20 homographies are shown alongside with the best performance of the global registration. In the following, the validation data provided by calibration sequence 3 of each scene will be treated as test data, whereas the data of calibration sequence 1 and 2 will be treated as training data. The training data are used for feeding both the multiple homography registration and the global registration algorithm with point correspondences.

Scene 1

The performance of the registration algorithm versus the number of clusters, for which homographies are generated, is seen from Figure 8.4. From the graphs corresponding to a RANSAC-threshold of 3 it is seen that the transfer errors of the test set decreases from 150 and 200 to a relatively stable level with only five clusters. From here, the increase in the number of clusters does not affect the performance of the test set until 20 and 21 clusters are reached. Beyond 21 clusters it seems that the resulting homographies are modelling noise rather than the actual characteristics of the thermal-visible camera configuration, as the errors for the mapping rgb→t are fluctuating greatly. This does not affect the mapping from t→rgb, which might be due to the reduced complexity in the thermal point cloud. With a RANSAC-threshold of 10, one might see that the errors of the mapping for the test set do not fall with the increase of the number of clusters.

It is seen from the graphs of Table 8.4 that the mapping of the training data is improved, the larger the number of clusters, although the transfer error is decreasing slowly above 10 clusters.



(a) RANSAC-threshold of 3 **(b)** RANSAC-threshold of 10

Figure 8.4: Relationship between number of clusters and corresponding one-directional transfer error for scene 1

Figure 8.5 shows the relationship between the squared Mahalanobis distance to the nearest cluster and the geometric OTE with a RANSAC-threshold of 3. It is seen

from the graphs of the figure that the OTE of the test data is nearly independent on the positioning of the clusters. The OTE for the mapping from t→rgb even decreases with larger distances, and the OTE for the mapping from rgb→t is increasing very slightly up to a squared distance of 5. This characteristic generally also applies for the mapping of the training data from t→rgb. For the mapping of the training from rgb→t, we see a steady increase in the OTE as the distance to the nearest homography increases, as one would expect from theory.

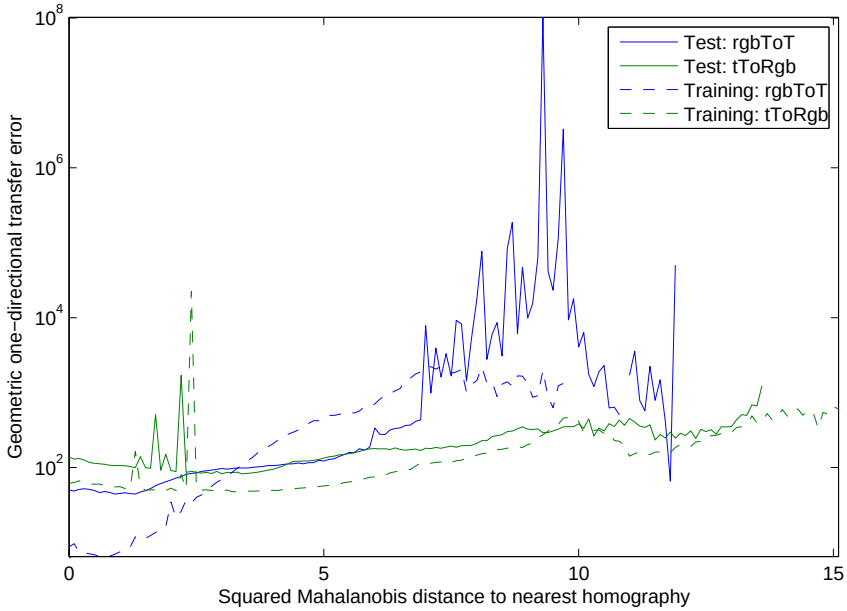


Figure 8.5: The distance to the nearest homography, or cluster, compared to the geometric OTE for scene 1. The graphs are generated with a RANSAC-threshold of 3 but are equivalent to the corresponding graph generated with a threshold of 10.

The accuracy for mapping the test and training points by 5, 10, 15, and 20 clusters with a RANSAC-threshold of 3 is seen from Table 8.6 and 8.7, respectively. The best fitting homography with a RANSAC-threshold of 10 is also shown. The mapping errors of the table reflects the graphs of Figure 8.4, and it is thus seen from the data that the best overall mapping is obtained with 5 homographies and a RANSAC-threshold of 3. At the bottom of the table is shown the accuracy of the best fitting planar homography, which is obtained with a RANSAC-threshold of 12. It is seen that this planar homography is superior to the method of multiple homographies when measuring the geometric error of the STE. However, the mapping from rgb→t is more accurate with the method on multiple homographies. With 3 homographies and a RANSAC-threshold of 10, the average geometric transfer error from rgb→t is down from 104.4 to 94.5 pixels, a 9 % decrease. The mapping from t→rgb of the multiple homographies comes nowhere near of the performance of the single homography, however. An example of the performance of the mapping is shown in Figure 8.6 for which two mappings of each modality in the test set is performed.

The transfer errors of the training set are listed in Table 8.7. Unsurprisingly, the transfer errors are considerably lower than the errors of the test set. One might see from the table that the geometric mean of the STE decreases with increasing cluster size and thus is on par with or even lower than the STE of the planar homography. The decrease in STE is enabled by sharp decreases in the mapping error from rgb→t

Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
5	3	OTE	rgb→t	46.02	53.48	81.63	69.2	99.5	108
			t→rgb	35.25	59.02	83.26	70.38	94.27	108
		STE	rgb↔t	81.28	112.5	148.2	126.6	193.8	193.4
10	3	OTE	rgb→t	40.28	59.52	135.8	78.76	99.8	174.7
			t→rgb	40.79	63.82	95.92	79.6	104.6	126
		STE	rgb↔t	81.07	123.3	201	132	204.4	253.2
15	3	OTE	rgb→t	56.53	59.71	312.2	84.63	116.2	352.7
			t→rgb	30.02	63.88	77.2	89.31	93.9	127.2
		STE	rgb↔t	86.55	123.6	339.9	142.2	210.1	397.9
20	3	OTE	rgb→t	45.31	58.47	190.3	129.3	103.8	272.7
			t→rgb	31.36	74.24	69.07	96.89	105.6	122.7
		STE	rgb↔t	76.67	132.7	215.6	175.3	209.4	317.3
3	10	OTE	rgb→t	36.84	57.66	84.46	68.84	94.5	111.5
			t→rgb	49.05	55.1	150.5	75.45	104.1	195.1
		STE	rgb↔t	85.89	112.8	198	130.7	198.6	258.7
Planar	12	OTE	rgb→t	33.29	71.08	73.59	85.06	104.4	116.8
			t→rgb	25.43	50.86	52.83	60.62	76.28	84.29
		STE	rgb↔t	58.72	121.9	126.3	145.6	180.7	201

Table 8.6: Errors for the test of the image registration algorithm on scene 1 by using *Nbr. hom.* clusters. The 'Planar 12' is the best fitting planar homography. 12 denotes the RANSAC threshold used for filtering the outliers.

(a) Test image 1, rgb→t

(b) Test image 1, t→rgb

(c) Test image 14, rgb→t

(d) Test image 14, t→rgb

Figure 8.6: The performance of the rectification algorithm on two sets of images of scene 1. Ground truth points are shown in green and rectified points are shown in red. The rectification is performed using 3 homographies generated by a RANSAC-threshold of 10. The upper images show a nearly perfect registration between the modalities, whereas the two lower images show a considerable mismatch of the rectification.

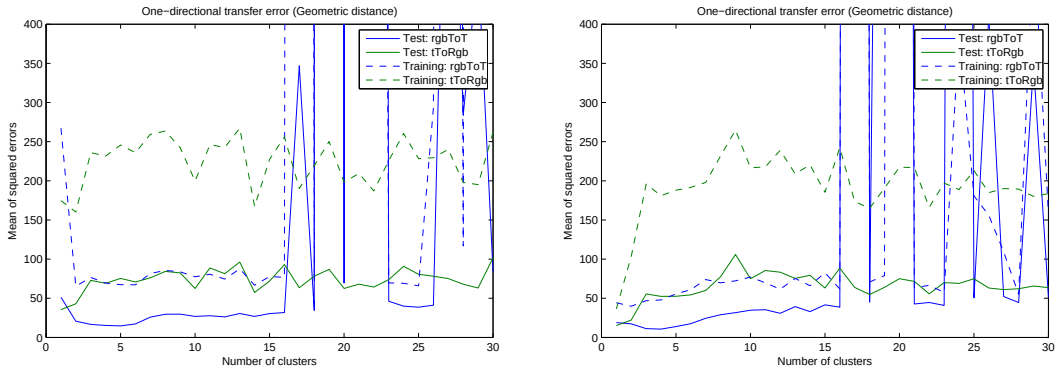
whereas the accuracy for t→rgb remains largely unchanged. The best mapping from rgb→t is enabled by a RANSAC-threshold of 10 with 30 clusters supported, which gives a geometric mean of the OTE of 14.85, a decrease of 67 % compared to the baseline.

Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
5	3	OTE	rgb→t	71.04	30.23	326.3	117.6	101.3	417.8
			t→rgb	16.23	33.44	32.76	40.18	49.66	53.79
		STE	rgb↔t	87.26	63.66	345	131.7	150.9	440.5
10	3	OTE	rgb→t	16.91	16.72	68.9	59.38	33.63	116.8
			t→rgb	20.25	31.75	48.5	47.22	52	74.19
		STE	rgb↔t	37.16	48.47	98.37	95.51	85.63	170.6
15	3	OTE	rgb→t	17.6	15.3	79.46	54.7	32.9	118.7
			t→rgb	17.11	32.08	41.8	45.67	49.2	66.32
		STE	rgb↔t	34.71	47.39	100.8	88.26	82.1	163.3
20	3	OTE	rgb→t	12.45	12.67	41.09	39.47	25.12	67.83
			t→rgb	17.07	36.7	55.03	49.3	53.77	77.55
		STE	rgb↔t	29.52	49.37	77.79	73.2	78.89	120.3
20	10	OTE	rgb→t	7.485	10.19	20.31	31.92	17.68	45.51
			t→rgb	20.52	35.5	98.64	48.96	56.01	119.2
		STE	rgb↔t	28	45.69	102.8	66.33	73.69	134.8
30	10	OTE	rgb→t	6.432	8.418	14.93	22.87	14.85	30.96
			t→rgb	17.91	37.95	86.4	57.69	55.86	120.1
		STE	rgb↔t	24.34	46.37	89.21	65.37	70.71	127.4
Planar	12	OTE	rgb→t	12.16	33.26	31.84	51.22	45.42	65.95
			t→rgb	9.373	24.08	25.47	37	33.45	49.75
		STE	rgb↔t	21.53	57.34	57.28	88.2	78.87	115.6

Table 8.7: Errors for the training of the image registration algorithm on scene 1 by using *Nbr. hom.* clusters. The 'Planar 12' is the best fitting planar homography. 12 denotes the RANSAC threshold used for filtering the outliers.

Scene 2

Scene 2 is by far the most planar of the three scenes, and one might therefore expect the scene to be successfully modelled by a small subset of homographies. This is reflected in Figure 8.7 which shows the number of clusters versus the transfer error. Indeed, the performance of the registration algorithm on the test data shows that the mapping from rgb→t might effectively be handled with less than 6 homographies. The performance of the mapping from t→rgb is significantly worse and shows little improvement by increasing the number of clusters. Beyond 15 clusters is the performance of the mapping indeed random, as multiple homographies in the 3D space is modelling the noise and inaccuracies of the training data. However, this does only apply for the mapping from rgb→t, whereas the mapping from t→rgb seems to be inresponsive to the number of clusters.



(a) RANSAC-threshold of 3 **(b)** RANSAC-threshold of 10

Figure 8.7: The number of clusters and the resulting error for the training and test set of scene 2. The graph clearly shows that the mapping from rgb→ is unstable for 17 clusters or above.

The performance of the mapping of training data from t→rgb is significantly worse than the mapping from rgb→t. This is also seen from the graph of Malahanobis distances versus the transfer error, which is shown in Figure 8.8. It is seen from the figure that the transfer errors for the mapping from t→rgb starts at 40 and 110 pixels test and training data, respectively, even if the nearest cluster is at zero distance. This does indeed show that the proposed implementation of the clustering scheme for mapping thermal points to RGB does not perform well in planar scenes.

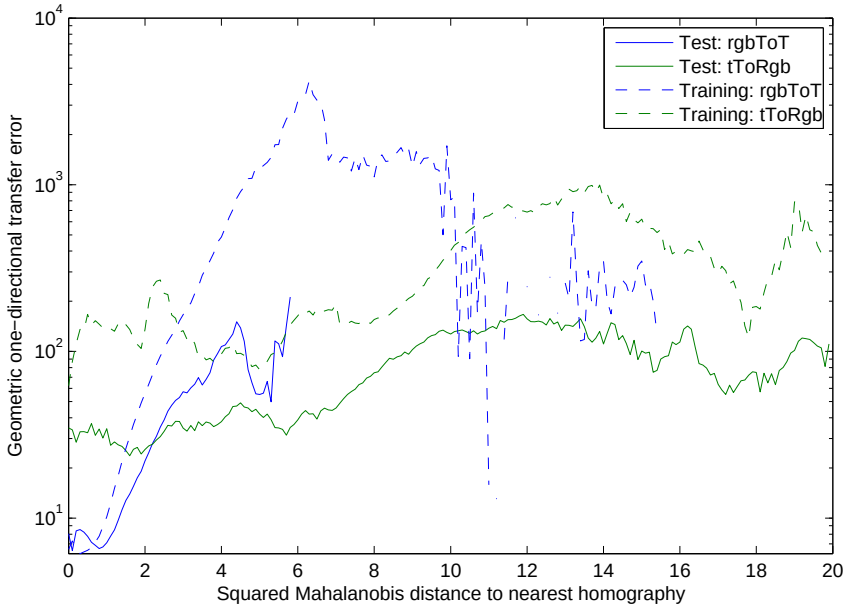


Figure 8.8: The distance to the nearest homography and the average transfer error for test and training points of scene 2. Below a distance of 1, the rectification algorithm delivers an average transfer error below 10. Beyond this threshold, the error shows an exponential relationship with the increase of the squared error.

The same might not be said for the mapping of RGB points to thermal. It is seen that the error does actually increase with the distance to the nearest homography up to a certain point. Beyond this point, the error fluctuates with no sign of any pattern. The performance of the multiple homographies on the test set is listed in Table 8.8

and the performance of the best fitting set of homographies is shown for a dual set of images in Figure 8.9. The table shows the OTE and STE for selected set of numbers of homographies along with the best performing planar homography, which in this case is generated with a RANSAC-threshold of 5. The table verifies that the mapping from t→rgb provided by the multiple homographies is lagging behind the baseline greatly. The best performance from t→rgb is provided by a single homography which is indeed similar to the baseline, just with different threshold for the RANSAC estimation. The performance of the opposite mapping from rgb→t is different, though. The transfer error by using 3 homographies with a RANSAC-threshold of 10 is smaller on than the corresponding error by using a planar homography. The mean of the geometric error is down from 19.56 to 11.23, a 43 % decrease. The standard deviation of the error is down from 22.74 to 16.53, which corresponds to a decrease of 27 %.

(a) Test image 10, rgb→t

(b) Test image 10, t→rgb

(c) Test image 3, rgb→t

(d) Test image 3, t→rgb

Figure 8.9: The performance of the rectification algorithm on two chosen test images of scene 2. Ground truth points are shown in green and rectified points are shown in red. The mappings are performed by using 3 homographies generated by a RANSAC-threshold of 10. The mapping of points from t→rgb for test image 10 shown in the upper right corner shows a nearly perfect mapping of the thermal points in the visible modality. The opposite mapping shown in the upper left corner shows that the mapping from rgb→t suffers from noticeable parallax. The lower images of test image 3 shows the severity of the parallax induced from the mapping of points from t→rgb. Although the mapping from rgb→t is not exact, it is substantially more accurate than its counterpart.

The transfer errors of the training data shown in Table 8.9 shows that the model of multiple homographies is struggling to model the point correspondences of the training set. The mappings from t→rgb are still erroneous and does not constitute a useful registration. The planar homography that fitted the test set of Table 8.8 does

Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
5	3	OTE	rgb→t	8.735	5.98	20.15	11.55	14.71	24.46
			t→rgb	27.69	47.7	69.14	104.1	75.38	144
		STE	rgb↔t	36.42	53.68	79.67	104.4	90.1	148.9
10	3	OTE	rgb→t	17.26	9.606	35.8	21.53	26.87	44.4
			t→rgb	26.32	36.21	57.38	85.21	62.53	118.6
		STE	rgb↔t	43.58	45.82	74.2	88.51	89.4	127.5
15	3	OTE	rgb→t	21.48	9.01	54.31	19.43	30.49	60.07
			t→rgb	32.7	39.28	79.73	82.61	71.98	132.4
		STE	rgb↔t	54.18	48.29	104.7	88.36	102.5	151.2
20	3	OTE	rgb→t	42.1	27.43	770	1797	69.53	2560
			t→rgb	25.46	37.05	62.02	84.22	62.51	117.4
		STE	rgb↔t	67.56	64.48	772.9	1799	132	2562
3	10	OTE	rgb→t	5.698	5.532	13.99	8.902	11.23	16.53
			t→rgb	24.09	31.44	59.88	62.76	55.53	95.93
		STE	rgb↔t	29.79	36.97	65	63.32	66.76	99.36
Planar 5		OTE	rgb→t	9.546	10.01	16.77	13.89	19.56	22.74
			t→rgb	7.397	7.757	12.8	10.32	15.15	17.1
		STE	rgb↔t	16.94	17.77	29.53	24.09	34.71	39.69

Table 8.8: Errors for the test of the image registration algorithm on scene 2 by using *Nbr. hom.* clusters. The 'Planar 5' is the best fitting planar homography. 5 denotes the RANSAC threshold used for filtering the outliers.

show to be performing a lot worse fitting the training data and thus, another planar homography is included which fits the training set better. This homography does indeed outperform the method on multiple homographies by a large margin when measured by the STE.

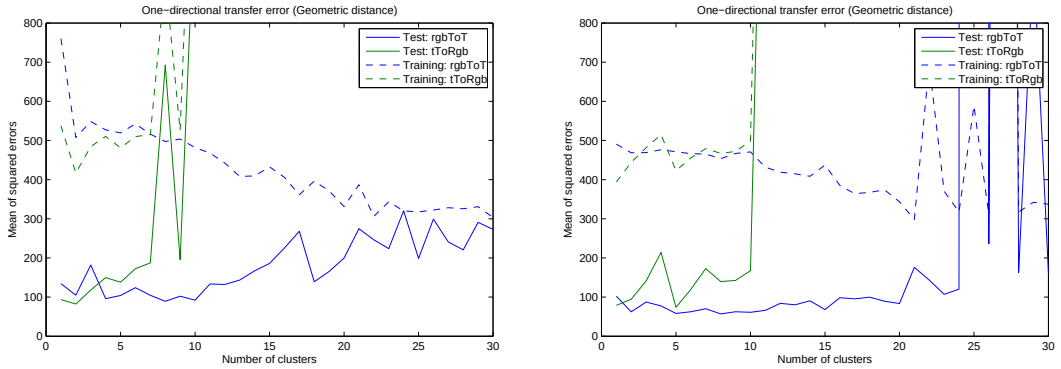
Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
5	3	OTE	rgb→t	36.45	30.94	95.91	106.7	67.39	182.2
			t→rgb	101.3	144.4	272	430.7	245.7	661.1
		STE	rgb↔t	137.7	175.3	332.9	503.2	313	783.5
10	3	OTE	rgb→t	45.39	32.06	123.2	105.5	77.45	202.6
			t→rgb	91.36	108.5	223.9	324.6	199.8	514.8
		STE	rgb↔t	136.7	140.5	281.6	378.3	277.3	610.6
15	3	OTE	rgb→t	47.98	29.85	135.8	111.6	77.83	224.7
			t→rgb	102.9	124.5	266.8	370.9	227.4	600
		STE	rgb↔t	150.9	154.4	334.6 .	438.5	305.2	722
20	3	OTE	rgb→t	1354	472.5	3.7e05	1.3e05	1827	4.9e05
			t→rgb	80.38	117.9	202.1	348.4	198.3	516.8
		STE	rgb↔t	1434	590.4	3.7e05	1.3e5	2025	4.9e05
2	10	OTE	rgb→t	20.76	19.06	55.07	69.08	39.83	108.3
			t→rgb	39.94	63.43	124.8	213.2	103.4	326.4
		STE	rgb↔t	60.7	82.5	157.9	267.6	143.2	405.8
Planar 5		OTE	rgb→t	31.98	22.05	75.53	53.75	54.03	107.7
			t→rgb	23.24	18.78	52.55	55.44	42.02	88.6
		STE	rgb↔t	55.22	40.83	127.6	108.3	96.05	194
Planar 30		OTE	rgb→t	14.93	16.84	24.1	44.76	31.77	57.1
			t→rgb	12.17	15.57	19.67	46.43	27.74	58.64
		STE	rgb↔t	27.1	32.4	43.41	90.98	59.51	115.2

Table 8.9: Errors for the training of the image registration algorithm on scene 2 by using *Nbr. hom.* clusters. The 'Planar 5' is the best fitting planar homography of the test data. However, this homography does not perform as well for the training data as the homography with a RANSAC-threshold of 30, which is therefore included. 30 denotes the RANSAC threshold used for filtering the outliers.

Scene 3

The transfer error for the test and training set for different cluster sizes is seen the graphs of Figure 8.10 and the entries of Table 8.10 and 8.11. It is observed from the graphs of Figure 8.10 that the transfer errors of the mapping from t→rgb goes beyond the graph window for clusters sizes greater than 8 – 10. The best performance for this mapping is achieved with the smallest amount of clusters, thus resembling a planar homography. The picture is different when it comes to the performance of the mapping from rgb→t. Here, the method on multiple clusters does seem to improve the performance by adding more clusters. The transfer error

for the test data is lowered until 10 clusters have been reached. When the number of clusters increases beyond this point, the error increases rapidly. For the training set, however, the mapping from rgb→ improves steadily with the higher number of clusters which shows that in the case of scene 3, the proposed method is making a better approximation of the data set by applying multiple homographies. This does only apply for a RANSAC-threshold of 3, however. For a threshold of 10, the error decreases until 20 clusters have been reached. Beyond this point, the errors of the mapping from rgb→t on the training and test are indeed very unstable.



(a) RANSAC-threshold of 3 **(b)** RANSAC-threshold of 10

Figure 8.10: The number of clusters and the resulting error for the training and test set of scene 3.

The relationship between the distance to the nearest homography and the resulting average transfer error for the training and testing of scene 3 is shown in Figure 8.11. The graphs are consistent with the equivalents of scene 1 and 2 and shows that the error of the mapping of points from rgb→t is going towards zero as the distance to the nearest homography decreases. It is seen from the graphs that the error of the mapping from t→rgb for the test and training data is indeed larger with increasing distances. However, at zero distance to the nearest homography, the mapping from t→rgb results in a relatively large transfer error of 70 and 500 pixels for the test and training datasets, respectively.

The bad performance of the mapping from t→rgb is reflected in Table 8.10 and Table 8.11 for the test and training data of scene 3. The performance of the mapping does not come on par with with the baseline of the planar homography in neither the training nor test set. The best performance of the mapping from rgb→t in the test set is found with 8 clusters and a RANSAC-threshold of 10. The average of the geometric OTE features a decrease in the error from 69.98 to 59.96. However, the standard deviation of the error remains high, which indicates that some points are mapped with suboptimal accuracy, thus resulting in large errors. The rectification is applied on a subset of the test images which are found in Figure 8.12.

The mapping error for the training set of scene 3 shows that the planar homography is struggling to deliver a good performance of the mapping from rgb→t. The average of the geometric error has increased by a factor of 5.7, resulting in a average OTE from rgb→t of 466.2 pixels. The method on multiple homographies delivers considerably better performance in representing the training set by 30 clusters, resulting in an average OTE from from rgb→t of 303.2 pixels, a decrease of 35 %

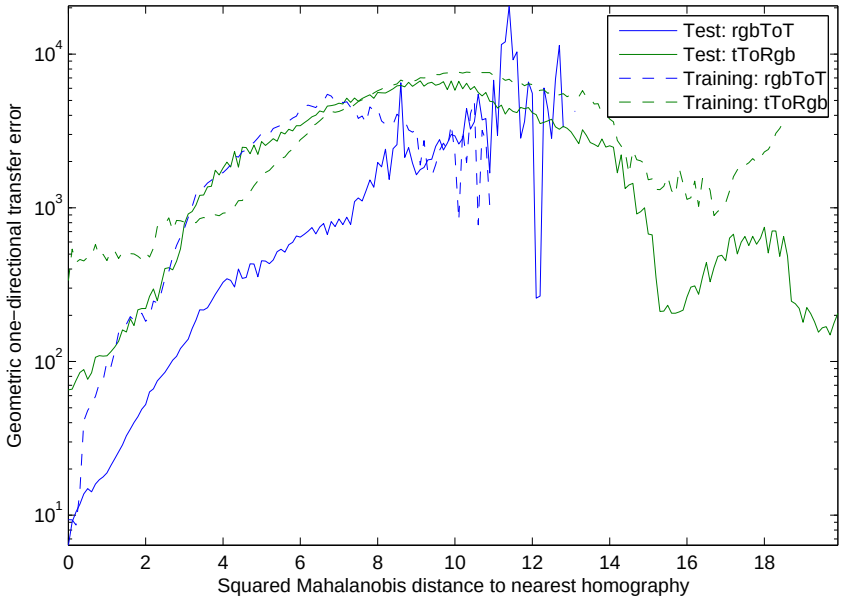
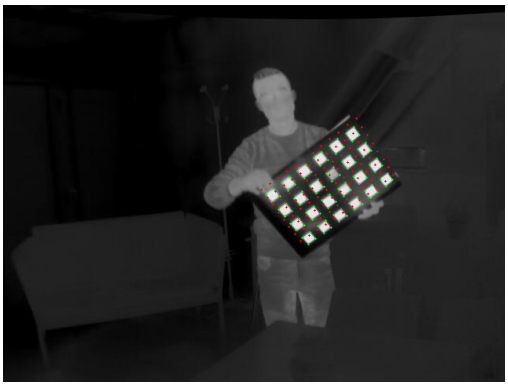


Figure 8.11: The distance to the nearest homography and the average transfer error for test and training points of scene 3.

Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Algebraic error (x,y) Std.dev		Geometric error Mean	Geometric error Std.dev
5	3	OTE	rgb→t	59.64	44.43	274.1	132.6	104.1	373.9
			t→rgb	81.16	56.97	182.1	121.2	138.1	268.5
		STE	rgb↔t	140.8	101.4	340.9	210.1	242.2	497.2
10	3	OTE	rgb→t	53	39.18	155.4	115.2	92.18	234.3
			t→rgb	739.2	375.2	2493	1545	1114	3404
		STE	rgb↔t	792.2	414.3	2522	1554	1207	3436
15	3	OTE	rgb→t	122.2	63.93	546.1	189.4	186.1	633.2
			t→rgb	3042	1150	6939	2546	4192	8495
		STE	rgb↔t	3164	1214	7106	2597	4378	8684
20	3	OTE	rgb→t	128.2	71.62	460.6	257.8	199.8	594
			t→rgb	5010	1817	7091	2637	6827	8358
		STE	rgb↔t	5138	1889	7133	2684	7027	8407
8	10	OTE	rgb→t	24.79	32.17	76.06	127	56.96	180.7
			t→rgb	77.08	62.67	160.4	109.3	139.7	234.2
		STE	rgb↔t	101.9	94.83	190	207.3	196.7	343.2
Planar 30		OTE	rgb→t	28.45	41.53	37.63	87.72	69.98	103.6
			t→rgb	23.15	33.41	30.02	67.64	56.56	79.2
		STE	rgb↔t	51.6	74.93	67.5	155.3	126.5	182.6

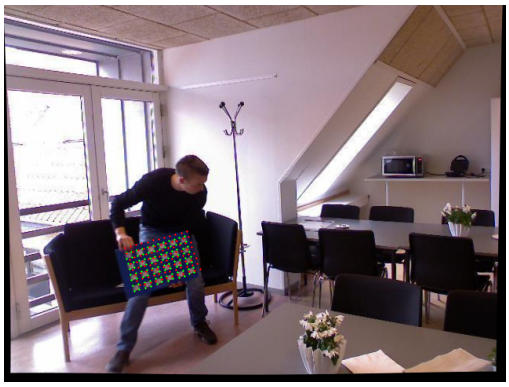
Table 8.10: Errors for the test of the image registration algorithm on scene 3 by using *Nbr. hom.* clusters. The 'Planar 30' is the best fitting planar homography. 30 denotes the RANSAC threshold used for filtering the outliers.



(a) Test image 10, rgb→t

(b) Test image 10, t→rgb

(c) Test image 20, rgb→t



(d) Test image 20, t→rgb

Figure 8.12: The performance of the rectification algorithm on two chosen test images of scene 3. Ground truth points are shown in green and rectified points are shown in red. The rectification is performed by using 8 homographies generated by a RANSAC-threshold of 10. The upper two images show that both the mapping from rgb→t and t→rgb suffers from nearly the same parallax as points are mapped either too high or too low with respect to the y-axis. The lower two images show the impact of the depth information contained by the mapping from rgb→t. Even though this mapping is skewed a few pixels for the lower points of the calibration rig, the mapping is indeed more accurate than the corresponding mapping from t→rgb which does not take advantage of the additional depth information.

compared to the single planar homography.

Nbr. Hom.	RANSAC thresh.			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
5	3	OTE	rgb→t	199.6	319.7	832.1	1719	519.3	2252
			t→rgb	208.6	272.7	615	1285	481.3	1660
		STE	rgb↔t	408.3	592.4	1412	2995	1001	3885
10	3	OTE	rgb→t	179.9	301.9	810.5	1709	481.8	2227
			t→rgb	817.2	662.5	2679	2261	1480	4015
		STE	rgb↔t	997	964.4	2948	3475	1961	5248
15	3	OTE	rgb→t	167.8	264.7	746.7	1536	432.6	2001
			t→rgb	2560	1357	6279	2911	3917	7730
		STE	rgb↔t	2728	1622	6431	3674	4350	8276
20	3	OTE	rgb→t	119.7	211.6	605.1	1266	331.2	1646
			t→rgb	4062	1955	6592	2770	6017	7536
		STE	rgb↔t	4182	2166	6669	3200	6348	7811
30	3	OTE	rgb→t	103.2	200	570.7	1206	303.2	1563
			t→rgb	2666	1560	5221	2559	4226	6289
		STE	rgb↔t	2769	1760	5315	3055	4529	6665
Planar	30	OTE	rgb→t	154.3	311.9	734.9	1657	466.2	2125
			t→rgb	127.8	262.9	613.4	1408	390.7	1799
		STE	rgb↔t	282.1	574.8	1348	3065	856.9	3924

Table 8.11: Errors for the training of the image registration algorithm on scene 3 by using *Nbr. hom.* clusters. The 'Planar 30' is the best fitting planar homography. 30 denotes the RANSAC threshold used for filtering the outliers. The performance of the mapping with a RANSAC-threshold of 10 is not shown as the training error is above what is achieved by a RANSAC-threshold of 3.

8.2.4 Summary

The compliance with the requirements of the thermal-visible registration algorithm are listed in short form in Table 8.12 and 8.13.

8.3 Conclusion on the acceptance test

This section will summarize the most important conclusions from the acceptance test:

- An image acquisition platform is built that enables the synchronized capture of visual, thermal, and depth imagery from the Microsoft Kinect sensor and an AXIS thermal camera. The baseline of the RGB and thermal image sensors is down to 70 mm.
- Depth information is contained in indoor conditions in the range of 1 – 4 m and is registered against the RGB image of the Kinect.

Scene			Geometric error Mean	Std.dev
1	OTE	rgb→t	-9,5 %	-4,5 %
		t→rgb	+23,1 %	+33,7 %
	STE	rgb↔t	+7,2 %	+3,8 %
2	OTE	rgb→t	-42,5 %	-27,4 %
		t→rgb	+266 %	+445 %
	STE	rgb↔t	+92,3 %	+158%
3	OTE	rgb→t	-18,6 %	+74,4 %
		t→rgb	+147 %	+196 %
	STE	rgb↔t	+55,5 %	+88,0%

Table 8.12: Compliance with the accuracy requirements for the image acquisition platform. It is seen that the proposed registration algorithm performs significantly worse than the baseline in the mapping of thermal points to visible. However, the proposed algorithm is providing more accurate results when mapping visible points to thermal, except for the standard deviation of the test set of scene 3.

Parameter	Requirement	Test result	Passed
Registration range	1 m – 4 m	Registered within the scene boundaries	Passed
Number of validation points	$\geq$ 1000	$\geq$ 1440	Passed
Distribution of validation points	-	-	Passed

Table 8.13: Compliance with the additional requirements for the image acquisition platform.

- The acquisition platform features a combined maximum frame rate of 15 FPS. Frames from the two cameras are synchronized with an average lag of 9.53 ms. However, there are rare spikes in the synchronization performance which means that some frames feature a lag of up to 62.4 ms, which is too much if objects are in motion.
- In order to test the performance of the proposed registration algorithm, three scenes are created. Scene 1 and 2 feature point correspondences within 1 – 4 m whereas scene 2 is largely planar and features point correspondences within a depth range of $\pm$ 0.25 m.
- For each scene, between 4224 and 5168 point correspondences are generated, which by manual inspection is found to be distributed throughout the wanted range of the scene.
- The proposed registration algorithm of multiple homographies is shown to perform better on average than the baseline method when mapping points from the visible to the thermal modality. However, the mapping of thermal points to the visible modality is shown to be substantially worse than the baseline of using a single homography.
- With the use of the proposed distance metric of squared Mahalanobis distance, it is shown that the rectification error is smaller the lower the distance to the closest homography. This shows that the Mahalanobis distance is relevant when defining 'nearby' homographies.
- The great divide of the performance of mapping points between modalities is most certainly a property of the missing depth information for pixels of the thermal camera. Without this, the mapping of thermal points to visible is only made possible through an estimation of the depth, which means that the mapping of thermal points is vulnerable to parallax.
- The results of the training sets shows that the proposed registration algorithm is substantially better at approximating the mapping of points from visible to thermal by using the depth information and finding the nearest homography than by using a fitted single homography. This does however only apply for the more 'complicated' scenarios of scene 1 and 3. For scene 2, the planar structure of the scene means that the training sets of corresponding points are better fitted by using a single homography.
- The visual images of the rectification of visible points to thermal shows that although the accuracy is increased with respect to the baseline, there is room for improvement. In some test images, the rectified points still feature a parallax that is noticeable if one wants to use the imagery for further image processing.

Chapter 9

Conclusion

The point of departure for this thesis is to investigate the registration of thermal, visible, and depth imagery for objects placed in close range to the cameras. Related work on multi-modal image registration is largely focused on developing an accurate registration between the thermal and visible modalities for objects residing relatively far away from the cameras or for objects lying on a plane. Recent work has been conducted that performs thermal-visible registration for surveillance scenarios where objects are placed closer to the camera. However, these algorithms either rely on a dual stereo camera configuration or the requirement that only blobs that are clearly distinguishable in the thermal modality may be accurately registered.

In this project, we do want to limit ourselves to a dual stereo camera configuration or the assumption that humans are always non-occluded and significantly hotter than the background. Therefore, alternative image registration techniques are proposed that registers objects in the thermal, visible, and depth images. In order to do so, multi-modal imagery must be available for the training and testing of the registration algorithms. This imagery is captured using a custom-built image acquisition platform that features a Microsoft Kinect for XBOX camera and a AXIS Q1922 camera for acquiring the visible, depth, and thermal images respectively. The range of the data is limited by the depth sensor that is specified within a range of $0.8 - 4$ m. Imagery from the multi-modal sensors is captured by a software acquisition platform which enables the simultaneous capture and synchronization of image frames from the cameras. The depth images obtained from the Kinect are registered to the visible images by the included SDK, which means that depth information is provided out-of-the-box for the visual image. Images from the three modalities are captured with a minimum frame rate of 15 FPS. Frames from each modality are synchronized through post-capture processing and feature an average synchronization lag of 9.53 ms. However, during a sequence of 1 minute, single frames may suffer from a synchronization lag of up to 62.4 ms.

In order to generate point correspondences in the thermal and visible images, we use the proposal of Vidas et al. [2012] to create an A3-sized calibration rig. Three scenes are defined, for which the registration algorithm is tested. Two scenes are containing objects in the full range of the depth camera of the Kinect whereas one scene only contains objects within a planar range. For each scene, the calibration rig is swept through the entire range of the scene, and through manual extraction

of the chessboard corners, three sets of point correspondences are generated for each scene. Each set contains at least 1440 point correspondences generated by minimum 15 views of the calibration rig. For each scene, two of the point sets are used for training the rectification algorithm whereas one is used for testing.

Two image registration methods have been investigated: stereo rectification and the rectification by multiple homographies. As the depth is factory registered to the visible image, only the visible and thermal images needs to be registered. The stereo rectification technique is well known and is readily implemented in OpenCV and MATLAB. Stereo rectification aligns each image such that the search for correspondence of two views is limited to either the horizontal or vertical axis. However, the search for corresponding pixels is indeed difficult in thermal-visible imagery as the modalities are fundamentally different. To compensate for this, a depth refinement technique is proposed that uses the depth information of objects to compensate for the parallax. However, even with this depth refinement technique, tests show that the stereo rectification falls short of providing an image registration that provides a better accuracy than the chosen baseline of a single homography.

This leaves us with the method of **multiple homographies**. By using multiple homographies to facilitate the mapping of points from one modality to another, one might reduce the effect of the depth parallax. The depth parallax is noticeable in scenes where the inter-object distances of the scene are large compared to the distances from the objects to the camera, which indeed is the case of scenes within 1 – 4 m. For each scene, homographies relating subsets of points in the thermal and visible images are generated using a k-means clustering scheme succeeded by a robust homography estimation based on the clustered points. The training points that lie inside an acceptable range of the homography are used for forming a Mahalanobis distance measure. The Mahalanobis distance is used for relating points to homographies, enabling the definition of nearby homographies for image points. A weighting scheme is applied to ensure a smooth transition between homographies as points are moved in space.

The acceptance test shows that the proposed rectification algorithm of multiple homographies is enabling a more accurate mapping of visible points to the thermal modality than the baseline, which uses a single homography. The registration by multiple homographies shows an average transfer error that is respectively 9.5 %, 42.5 %, and 18.6 % below the baseline for scene 1, 2, and 3. However, the better performance does only apply for mapping visible points to the thermal modality. When mapping thermal points to the visible modality, the proposed method is shown to contribute to an average transfer error that is up to 2.6 times larger than the baseline. The non-symmetric performance of the rectification may correspond to the absence of depth information of pixels in the thermal modality which entails that the mapping of points from the thermal modality is vulnerable to parallax for different depths.

The current state of the proposed algorithm thus enables the refinement of transferring visible points to the thermal modality at short range compared to the baseline. However, tests show that there is still room for improvement in order to provide an accurate pixel-to-pixel mapping between the modalities.

Chapter 10

Discussion

The work on the acquisition and registration of the tri-modal imagery has induced some deliberations that might contribute to a more accurate and more robust image registration algorithm than what is currently proposed. These deliberations are listed in the following:

Varying the RANSAC-threshold for homography generation Better accuracy may eventually be achieved by varying the RANSAC-threshold used for generating the multiple homographies of the registration method. In the acceptance test it was shown that RANSAC-thresholds of up to 30 pixels delivered the best performance for a single homography. Currently, only thresholds of 3 and 10 have been used for the generation of multiple homographies. The results show that a small threshold is good at estimating the training data, while a larger threshold is providing a more general estimate, thus delivering a better accuracy for registering points not lying in the training set.

Homography weighting scheme The proposed homography weighting scheme is chosen due to its continuous properties and relative weighting of nearby homographies. However, other weighting schemes might be proposed that gives a better estimate of evaluating nearby homographies at a current point in space. The current weighting scheme might be paired with a k-nearest neighbour approach such that only the k nearest homographies are chosen. Furthermore, it might show that no weighting scheme is required and choosing the nearest homography will provide the best approximation. This may work if homographies are placed very closely - otherwise, it might lead to a non-continuous mapping if nearby points are mapped by two different homographies.

Improving the performance of the thermal rectification The current state of the rectification of thermal points to the visible modality is indeed not satisfactory. Better performance might be introduced by varying the RANSAC-thresholds even more as stated above. The current method of finding clusters and homographies for the thermal training points relies on depth data extracted from the corresponding points in the visible modality. However, as the thermal points do not directly induce

any depth information, one might generate two homographies which are distant in 3D space but very close in terms of the projective ambiguity introduced by the lack of depth information. In order to correct this, a separate point cloud might be generated for the thermal points, in two dimensions only, for which the homographies are generated. However, even though such a scheme will certainly level out the depth parallax, it will not abolish it. In order to do this, one might use an iterative scheme, which is discussed in the following.

Iterative rectification As described above, the mapping of thermal points suffers from depth parallax caused by the absence of depth information for the thermal pixels. Instead of assuming a planar state as proposed above, a method by iterative rectification might make the mapping more stable and cancel the depth parallax. For a given point in the thermal image, one might make one or multiple guesses of the depth of the object. Based on the guess, the thermal point is mapped to the visible modality for which the actual depth of the object is known. Using the measured depth, the point is reprojected to the thermal image where the distance to the original thermal point is measured. If the reprojected and the original thermal points are coincident, the mapping is accurate. Otherwise, a new estimate of the depth of the thermal point is used to find a suitable mapping until the points coincide.

Using a different distance metric In the proposed method, the distance from a point to a homography is measured using the Mahalanobis distance of the training points used for the generating the homography. One might recall that points on a single plane are mapped accurately between two cameras views by a single homography. The Mahalanobis distance is a an estimate of the planar structure of the homography, thus penalizing points that does not lie close to plane of points used for the generation of the homography. However, we may go even further and estimate the plane in which the training points reside. Distances for points to homographies might then be calculated by measuring the distance to the closest plane.

Increasing the training set An even larger set of training points will enable a better estimation of the already existing homographies and provide the basis for a more accurate image rectification.

Bibliography

Al-Kassir, A. R., Fernandez, J., Tinaut, F., and Castro, F. (2005). Thermographic study of energetic installations. *Applied thermal engineering*, 25(2):183–190.

Asus.com (2013). http://www.asus.com/Multimedia/Xtion_PRO_LIVE#specifications. ASUS - Xtion PRO LIVE.

AXIS Corporation (2013a). http://www.axis.com/files/datasheet/ds_q1922_q1922-e_46221_en_1301_lo.pdf. AXIS A1922/-E Thermal Network Cameras - Datasheet.

AXIS Corporation (2013b). http://www.axis.com/techsup/cam_servers/dev/activex.htm. AXIS Media Control ActiveX components.

Barnich, O. and Van Droogenbroeck, M. (2011). Vibe: A universal background subtraction algorithm for video sequences. *Image Processing, IEEE Transactions on*, 20(6):1709–1724.

Bertozzi, M., Broggi, A., Felisa, M., Vezzoni, G., and Del Rose, M. (2006). Low-level pedestrian detection by means of visible and far infra-red tetra-vision. In *Intelligent Vehicles Symposium, 2006 IEEE*, pages 231–236. IEEE.

Bouguet, J.-Y. (2004). Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Davis, J. W. and Sharma, V. (2005). Fusion-based background-subtraction using contour saliency. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 11–11. IEEE.

Davis, J. W. and Sharma, V. (2007). Background-subtraction using contour-based fusion of thermal and visible imagery. *Computer Vision and Image Understanding*, 106(2):162–182.

dss.com (2013). http://www.dssvideo.com/shop/thermal-cameras/axis-q1922-10mm-30fps-network-camera/. AXIS Q1922.

Eisler, C. (2013). http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/20/near-mode-what-it-is-and-isn-t.aspx. Near Mode: What it is (and isn't).

Escalera, S., Mogelmose, A., Clapes, A., Bahnsen, C., and Moeslund, T. (2013). "tri-modal person re-identification with rgb, depth and thermal features. *9th IEEE CVPR workshop on Perception Beyond the Visible Spectrum (PBVS'2013)*.

Gade, R. and Moeslund, T. B. (2013). Thermal cameras and applications: A survey. Under peer review.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Hartley, R. and Zissermann, A. (2003). *Multiple View Geometry*. Cambridge, 2nd edition.

Hartley, R. I. (1999). Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2):115–127.

Johnson, M. J. and Bajcsy, P. (2008). Integration of thermal and visible imagery for robust foreground detection in tele-immersive spaces. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE.

Khoshelham, K. (2011). Accuracy analysis of kinect depth data. In *ISPRS workshop laser scanning*, volume 38, page 1.

Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3):172–185.

Krotosky, S. J. and Trivedi, M. M. (2007). Mutual information based registration of multimodal stereo videos for person tracking. *Computer Vision and Image Understanding*, 106(2):270–287.

Krotosky, S. J. and Trivedi, M. M. (2008). Person surveillance using visual and infrared imagery. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(8):1096–1105.

Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing, 1st edition.

Lee, D. J. (2012). Stereo calibration and rectification. Technical report, Brigham Young University. Slides on Robotic Vision.

Lee, S. K., McHenry, K., Kooper, R., and Bajcsy, P. (2009). Characterizing human subjects in real-time and three-dimensional spaces by integrating thermal-infrared and visible spectrum cameras. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, pages 1708–1711. IEEE.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. In *Proceedings of the national institute of sciences of India*, volume 2, pages 49–55. New Delhi.

Microsoft Corporation (2013). `http://www.microsoft.com/en-us/kinectforwindows/discover/features.aspx`. Product Features | Microsoft Kinect for Windows.

Microsoft Developer Network (2013a). http://msdn.microsoft.com/en-us/library/jj131033.aspx. Kinect for Windows Sensor Components.

Microsoft Developer Network (2013b). http://msdn.microsoft.com/en-us/library/hh973078.aspx#Depth_Ranges. Kinect for Windows Sensor Components.

Ó Conaire, C., Cooke, E., O'Connor, N. E., Murphy, N., and Smeaton, A. F. (2005). Fusion of infrared and visible spectrum video for indoor surveillance. In *WIAMIS 2005 - 6th International Workshop on Image Analysis for Multimedia Interactive Services*.

Openni.org (2013). http://www.openni.org/openni-sdk/#.UYyiuMpd27s. Open NI SDK | OpenNI.

Play.com (2013). http://www.play.com/Games/Xbox360/4-/10296372/Project-Natal/Product.html. Technical Details of the Kinect.

Point Grey Research, I. (2004). http://www.ptgrey.com/support/kb/data/kbStereoAccuracyShort.pdf. Stereo Accuracy and Error Modeling.

Point Grey Research, I. (2010). http://www.ptgrey.com/support/kb/index.asp?a=4&q=63. How is depth determined from a disparity image?

Point Grey Research, I. (2011). http://www.ptgrey.com/products/bbxb3/bumblebeeXB3_stereo_camera.asp. Bumblebee XB3 CCD FireWire Camera.

ROS.org OpenNI Wiki (2013). http://www.ros.org/wiki/openni_kinect/kinect_accuracy. Openni_kinect/kinect_accuracy.

Shirai, H. and Yu, O. (2011). http://sourceforge.net/projects/kinect-mex/. Kinect for MATLAB.

St-Laurent, L., Prévost, D., and Maldague, X. (2010). Fast and accurate calibration-based thermal/colour sensors registration. In *Proceedings of 10th Quantitative InfraRed Thermography conference, paper QIRT2010-126 Québec (Canada)*.

Stauffer, C. and Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757.

Torabi, A., Massé, G., and Bilodeau, G.-A. (2011). An iterative integrated framework for thermal-visible image registration, sensor fusion, and people tracking for video surveillance applications. *Computer Vision and Image Understanding*.

Tukey, J. W. (1967). An introduction to the calculations of numerical spectrum analysis. *Spectral Analysis of Time Series*, pages 25–46.

Ursine, W., Calado, F., Teixeira, G., Diniz, H., Silvino, S., and de Andrade, R. (2012). Thermal/visible autonomous stereo visio system calibration methodology for non-controlled environments. In *11'th International Conference on Quantitative InfraRed Thermography*.

Vadivambal, R. and Jayas, D. S. (2011). Applications of thermal imaging in agriculture and food industry—a review. *Food and Bioprocess Technology*, 4(2):186–199.

Vidas, S., Lakemond, R., Denman, S., Fookes, C., Sridharan, S., and Wark, T. (2012). A mask-based approach for the geometric calibration of thermal-infrared cameras. *Instrumentation and Measurement, IEEE Transactions on*, 61(6):1625–1635.

Zhang, Y., Zhang, X., Maybank, S. J., and Yu, R. (2012). An ir and visible image sequence automatic registration method based on optical flow. *Machine Vision and Applications*, pages 1–12.

Zhao, J. and Cheung, S. (2009). Human segmentation by fusing visible-light and thermal imaginary. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1185–1192. IEEE.

Zhao, J. and Sen-ching, S. C. (2012). Human segmentation by geometrically fusing visible-light and thermal imageries. *Multimedia Tools and Applications*, pages 1–29.

Part IV

Appendix

Appendix A

Acquisition software

A.1 Creating the software platform

The sections on interfacing the AXIS and Kinect cameras on page 28 give an overview of the creation of a platform for recording the imagery of the sensors. This section does not serve as a full documentation of the code written but will provide a deeper look into the mechanisms that we utilize in the platform.

The Kinect for Windows SDK enables us to choose between a subset of resolutions and frame rates for the depth and colour sensors. Some of the most important options are listed in Table A.1 below. The AXIS thermal camera is capable of selecting any

Colour stream	Depth stream
1280x960 RGB @ 12 FPS	640x480 @ 30 FPS
640x480 RGB @ 30 FPS	320x240 @ 30 FPS
640x480 YUV @ 15 FPS	80x60 @ 30 FPS

Table A.1: Available depth and colour streams of the Kinect for Windows SDK. Raw colour streams are not included in the table.

frame rate between 1 and 30 FPS, but according to research by Anders Jørgensen[1], the clock frequency inside the camera remains unchanged which means that the firmware drops frames according to the wanted frame rate. If one wants to pick a frame rate for the thermal camera different from 30 FPS, the frame rate should be a integer dividend of 30 FPS such as 7.5, 10, and 15 FPS. Because of the performance problems described in Section 5.2.2, the colour stream is chosen as 640x480 YUV @ 15 FPS. The YUV term is of no importance as the `WriteableBitmap` that the pixels are written into neatly converts the image to BGR.

As we would like the program to be able to register the depth, take multiple calibration sessions for each scene, and record multiple scenes, certain flags are introduced in the program:

- DEPTHCAL_CAPTURE_FLAG

[1]Research assistant at the VAP group. andjor@create.aau.dk

Flag for capturing the depth registration.

- DEPTH_CAPTURE_FLAG
  Enabling the capture of the depth stream.
- RGB_CAPTURE_FLAG
  Enabling the capture of the RGB stream.
- CALIBRATE_RGB_FLAG
  Setting the calibration mode of the RGB and thermal streams. The implication of this flag is to set the save paths for the RGB, thermal, and depth calibration streams.
- ENABLE_THERMAL_FLAG *Enabling the capture of the thermal image stream.*

These flags are set by the software buttons of the GUI shown in Figure 5.5.

A.1.1 AllFramesReady

The function `AllFramesReady` is called when both the RGB and depth streams fire an event simultaneously. This means that only the lowest frame rate of the depth and RGB streams matters - which in the current configuration is 15 FPS. The function distinguishes between different use cases specified by the flags set. The overall functionality of the `AllFramesReady` is seen from Figure A.1.

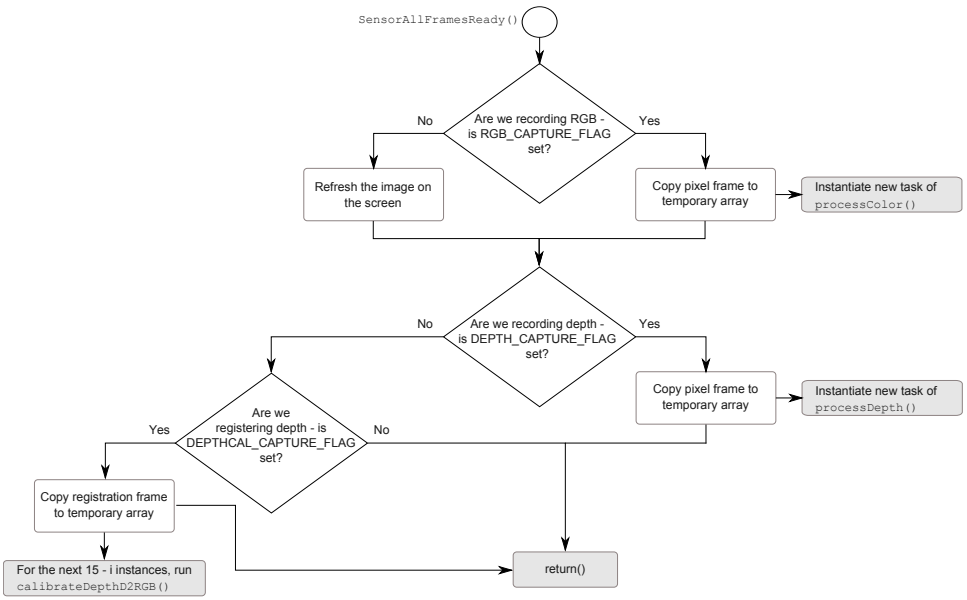


Figure A.1: Flowchart of function `AllFramesReady`.

The function mainly acts as a task handler - the core part of saving imagery is split upon three functions: `processColor`, `processDepth`, and `calibrateDepthD2RGB`. The functions `processColor` and `processDepth` are instantiated as new threads to improve real-time performance, whereas we have no need for this when calling the latter function.

A.1.2 ProcessColor

The `processColor` function is fairly simple. It is called with the colour pixels to process, the time stamp to add to the file, and the flag that decides whether the file should be saved in a calibration directory or not. By using the Save method of the Bitmap class, the RGB image is saved.

A.1.3 ProcessDepth

The `processDepth` function works in many ways like its sibling. The crucial difference is the way the bitmap is handled. In `processDepth`, the 16-bit bitmap is copied to a double-sized 8-bit indexed bitmap using the Marshal.Copy method. The indexed bitmap does, however, need a colour palette in order to be stored properly as an image, so we have to create the palette at run time. The palette is applied to the double-sized bitmap which may now be stored to disk. It is of no doubt that the need to re-create the palette every time a depth frame is stored is a performance hog - but no other workaround allowed us to store the full information of the depth image.

A.1.4 Depth registration

The function `calibrateDepthD2RGB` is responsible for converting the CoordinateMapper object of the Kinect's DepthImageFrame. This is done by running through every pixel of the CoordinateMapper, transferring the registration for this pixel to a two-dimensional point and then transferring the x and y coordinates of this point to the look-up-tables of the x and y direction respectively. Those look-up-tables are ushort arrays the size of 640x480, and so these are stored in the relevant calibration folder using the Save method of the BitmapEncoder.

A.1.5 Folder structure

When pressing the 'Use this folder' button of the GUI shown in Figure 5.5, folders are created in the directory provided in the text box as shown below:

- Cal
  - 1
    - D
    - RGB
  - 2
    - D
    - RGB
  - 3
    - D
    - RGB

- D
- RGB

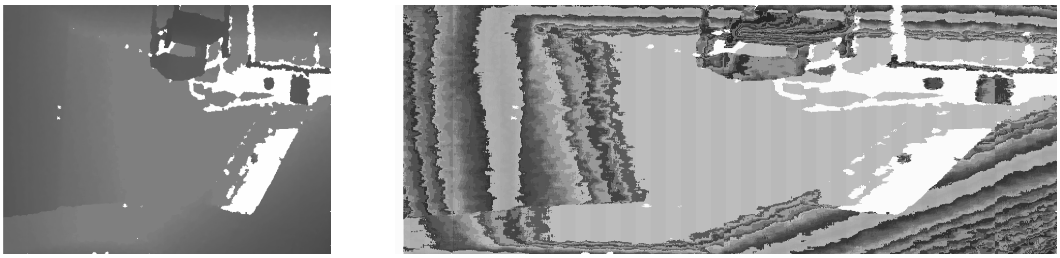
Depth and RGB frames are saved in the directories named accordingly whereas the thermal video and time stamps are saved in the root folders of the current capture. Three sub folders are created for calibration footage.

A.2 Post-processing the image stream

This section contains the MATLAB and Python scripts that are developed for post-processing the image stream and to ensure a robust output of the acquisition platform.

A.2.1 convertDepthDataToUshort

Due to performance requirements and an inefficient implementation of the 16-bit PNG save method in C#, the depth image is saved as a double-sized 8 bit image.



(a) Original 16-bit image of size 640x480.

(b) Converted 8-bit image of size 1280x408.

Figure A.2: Depth images from the Kinect. The 8 bit-image is the saved depth image from the Kinect in order to comply with performance requirements.

This transformation needs to be reversed in order to have a meaningful representation of the depth images, as seen from Figure A.2. This is done by the MATLAB-function `convertDepthDataToUShort.m`. The core MATLAB functions of the function are listed below

```
1     for i = 1:numImg
2         waitbar(i/numImg,h);
3
4         % Read and process all the .png images in the directory
5         img = imread(d(i).name);
6
7         if isa(img,        ) % We only want to process uint8 images
8             dim = size(img);
9
10            % Reshape image array and typecast
11            img = reshape(img',size(img,1)*size(img,2),1);
12            img = typecast(img,         );
13
```

```
14              % Then reshape it back
15              img = reshape(img,dim(2)/2,dim(1))';
16
17              % Rename the filename
18              imgStr = strrep(d(i).name,filetype,          );
19
20              % Save the converted image
21              imwrite(img,imgStr,     );
22          end
23      end
```

Most of the heavy lifting is done by the MATLAB-function `reshape`. In order for the function to work, we need however to reshape the image array to a vector - and back.

A.2.2 convertD2RGBtoRGB2D

As described in Section 5.2.2, the Kinect for Windows SDK is able to provide a registration from the depth image to the RGB image - but not the other way around. This leaves with us with the option to reverse the transformation ourselves which is performed in the MATLAB-function `convertD2RGBtoRGB2D.m`. When reversing a conventional image mapping, one would simply use the original transform to back-project the image. However, this is not easily doable is this case as the transform here is a look-up-table, making a relation of pixel x,y in the depth image to its corresponding x or y value in the RGB image. Instead, we choose to use a forward mapping. It may be expressed as:

$$\text{x-map:} \quad kx'_{(kx_{x,y},ky_{x,y})} = x \tag{A.1}$$

$$\text{y-map:} \quad ky'_{(kx_{x,y},ky_{x,y})} = y \tag{A.2}$$

The algorithm runs through each coordinate of the look-up-table and collects the x and y coordinate of the image point and the x and y coordinate of the corresponding mapping, $(kx_{x,y}, ky_{x,y})$. At the position of the mapped point in the new look-up-table, the original coordinates of the look-up-table is placed. In order to induce simplicity of the mapping, the x and y coordinates of the look-up-table are stored in two separate images.

Unfortunately, the method of forward mapping comes with certain downsides. We are not guaranteed that a one-to-one mapping exists between the original look-up-table and the reversed one; the mapping might be one-to-many, many-to-one, or a combination of both. Such a case is seen from Figure A.3 and one might see 'holes' in the reversed mapping where a correspondence does not exist.

As one might see from the table in Figure A.3, a strong correlation exists between vertical entries in the table, whereas the correlation between horizontal entries is existing, but weaker. The correlation exists in the x look-up-tables. In the y look-up-table, the correlation is reversed so the strongest correlation is between horizontal entries. This correlation might be exposed to correct the mapping by designing a structuring element and performing a morphological closing operation. If a proper structuring element is chosen, the morphological closing will fill the holes of the look-

	18	519	520	521	522	523	524	525	526	527	528	529	530	531	5
181	519	520	521	522	523	524	525	526	527	528	0	0	0	0	
182	519	520	521	522	523	524	525	0	0	0	0	0	526	527	
183	519	520	521	522	523	524	0	0	0	0	0	525	526	527	
184	519	520	521	522	523	524	0	0	0	0	0	525	526	527	
185	519	520	521	522	523	0	0	0	0	0	524	525	526	527	
186	519	520	521	522	523	524	0	0	0	0	0	525	526	527	
187	519	520	521	522	523	524	0	0	0	0	0	525	526	528	
188	519	520	521	522	523	524	0	0	0	0	0	525	526	528	
189	519	520	521	522	523	524	0	0	0	0	0	525	526	528	
190	519	520	521	522	523	524	525	0	0	0	0	0	526	528	
191	519	520	521	522	523	524	525	0	0	0	0	0	527	528	
192	519	520	521	522	523	524	525	526	0	0	0	0	527	528	
193	519	520	521	522	523	524	525	526	0	0	0	0	527	528	
194	519	520	521	522	523	524	525	0	0	0	0	0	527	528	
195	519	520	521	522	523	524	525	526	0	0	0	0	527	528	
196	519	520	521	522	523	524	525	0	0	0	0	526	527	528	
197	519	520	521	522	523	524	0	0	0	0	0	526	527	528	
198	519	520	521	522	523	0	0	0	0	0	524	526	527	528	
199	519	520	521	522	523	0	0	0	0	0	524	526	527	528	
200	519	520	521	0	0	0	0	0	522	523	524	526	527	528	

Figure A.3: The look-up-table relating the RGB image to the depth image, shown directly after the forward transformation of the look-up-table of depth to RGB. As one might see from the image, the transformation is not one-to-one, and there exist coordinates where a mapping is non-existing.

up-table with correlated entries but not affect the borders of the look-up-table where a correlation might not exist at all due to to different field-of-view of the cameras.

In order to exploit the high vertical correlation of the mapping, we construct a rectangular structuring element of size 30 x 3 and 3 x 30 for the x and y look-up-tables respectively. This is a rather large structuring element, but the size allows us to close nearly all holes in the mapping. The large size does have its shortcomings, however. Patterns not large enough to fit into the structuring element will be smoothed or removed and thus might some of the detail of the mapping be lost. This is solved by only applying the enhancements of the morphological operation on pixels in the look-up-table with zero-entries. The morphological closing operation is applied on the full image and subsequently only the pixels that were previously zero are transferred to the reversed look-up-table. This gives us the look-up-table of Figure A.4.

	518	519	520	521	522	523	524	525	526	527	528	529	530	531
181	519	520	521	522	523	524	525	526	527	528	524	525	526	528
182	519	520	521	522	523	524	525	526	523	524	524	525	526	527
183	519	520	521	522	523	524	525	526	523	524	524	525	526	527
184	519	520	521	522	523	524	525	526	523	524	524	525	526	527
185	519	520	521	522	523	524	525	526	523	524	524	525	526	527
186	519	520	521	522	523	524	525	526	523	524	524	525	526	527
187	519	520	521	522	523	524	525	526	523	524	524	525	526	528
188	519	520	521	522	523	524	525	526	523	524	524	525	526	528
189	519	520	521	522	523	524	525	526	523	524	524	525	526	528
190	519	520	521	522	523	524	525	526	523	524	524	525	526	528
191	519	520	521	522	523	524	525	526	523	524	524	525	527	528
192	519	520	521	522	523	524	525	526	523	524	524	525	527	528
193	519	520	521	522	523	524	525	526	523	524	524	525	527	528
194	519	520	521	522	523	524	525	526	523	524	524	525	527	528
195	519	520	521	522	523	524	525	526	523	524	524	525	527	528
196	519	520	521	522	523	524	525	526	523	524	524	526	527	528
197	519	520	521	522	523	524	525	526	523	524	524	526	527	528
198	519	520	521	522	523	524	525	526	523	524	524	526	527	528
199	519	520	521	522	523	524	525	526	523	524	524	526	527	528
200	519	520	521	522	523	524	525	526	522	523	524	526	527	528
201	519	520	521	522	523	524	525	521	522	523	525	526	527	528

Figure A.4: The look-up-table relating the RGB image to the depth image, after enhancements of the morphological closing operation. The holes of Figure A.3 have been successfully closed.

The implementation in MATLAB of the above mentioned function is seen in the Listing below. It is seen from Figure 5.4 on page 31 that we have handled all but one of the holes in the inverse map. The remaining hole would be removed by using a larger structuring element. However, this should be of the size $> 40x3$, which would introduce multiple artifacts in regions close to the border of the image. Therefore, we will note the current mapping as acceptable for the purpose of our work.

```
1 for i = 1:numImg
```

```
2       xImg = imread(xContents(i).name);
3       yImg = imread(yContents(i).name);
4
5       for ii = 1:size(xImg,1) % Displace coordinates
6           for jj = 1:size(xImg,2)
7               xcoord = xImg(ii,jj);
8               ycoord = yImg(ii,jj);
9
10              if ((xcoord > 0) && (ycoord > 0))
11                  newCoordImgX(ycoord,xcoord) = jj;
12                  newCoordImgY(ycoord,xcoord) = ii;
13              end
14          end
15      end
16
17
18      % Use structuring element to straighten out imbalances
19      seX = strel(           ,[30 3]);
20      seY = strel(           ,[3 30]);
21      newCoordImgXrecon = imclose(newCoordImgX,seX);
22      newCoordImgYrecon = imclose(newCoordImgY,seY);
23
24      % Filter the reconstructed images to only contain entries where
            the
25      % previous mapping had zero entries
26      newCoordImgXrecon = newCoordImgXrecon .* uint16(newCoordImgX ==
            0);
27      newCoordImgYrecon = newCoordImgYrecon .* uint16(newCoordImgY ==
            0);
28
29      % Create the final image
30      newCoordImgX = newCoordImgX + newCoordImgXrecon;
31      newCoordImgY = newCoordImgY + newCoordImgYrecon;
32  end
```

Appendix B

Implementation in OpenCV

This appendix will provide an overview of the code providing the functionality of the creation and validation of the image rectification algorithms presented. This will by no means serve as a thorough review of the code but will provide a description of the behaviour and output of the core functions. The code is implemented in Visual C++ in Microsoft Visual Studio 2010 Express and uses the OpenCV framework. The code is available in the CD provided[1].

The implementation supports the multiple image rectification algorithms proposed and might train and validate each of those rectification algorithms one-by-one. In order to do so, a number of flags are introduced:

USE_FUNDAMENTAL_MATRIX Is set if any method that rectifies the images based on the fundamental matrix is used, e.g. stereo rectification.

RECTFY_USING_F Is set if the images are in rectified view by the stereo rectification.

USE_DEPTH_FOR_RECT When using the depth refinement technique of Section 7.2.3, this flag is set.

IS_VERTICAL_STEREO The search for correspondence in rectified stereo imagery is reduced to either the vertical or horizontal axis. This flag determines in which direction one should search for the correspondence.

UNDISTORT_IMAGES If images or image points are undistorted, this flag is set.

USE_PLANAR_HOMOGRAPHIES When the method of rectification by multiple homographies presented in Section 7.3 is chosen, this flag is set.

computeEpipolarHomography This is the core function providing the entry points for training and validating the rectification algorithms. Using utility functions such as `readCoordinatesFromFile2D` and `readCoordinatesFromFile3D`, the

[1] Code/DualCalibration/DualCalibration.sln

function calibrates the camera views based on the sets of corresponding points extracted from MATLAB. These are saved in the file `intrinsics.yml` which is created on the first run. The function prompts the user to choose the wanted rectification algorithm and gives an estimate on the performance of the current algorithm on the current training set. By calling the function `validateCalibration`, the performance of the rectification algorithm is measured against the test set, and the transfer errors are written to log files in the Validation folder of the respective data set.

createPointCloud Is called within `computeEpipolarHomography` to create and save a point cloud for the training data. Points clouds seen through both the visible and thermal cameras are generated and saved in the respective CalibrationHelper folder.

computePlanarHomographies This function uses the points clouds generated by the above mentioned function to divide the points cloud into k clusters based on the normalized point cloud. The points that lie within these clusters are used for forming a homography by the built-in OpenCV function `findHomography`. This function returns a mask which is used to find the points that lie within the desired range of the computed homography. The world coordinates of these points in the visible and thermal camera frame are used to find the centre and the covariance matrix that corresponds to the homography.

computeCorrespondingRgbPointFromDepth The functions uses the pre-loaded registration images to map depth points to the RGB image. If no map is found, the function returns the coordinate (0,0).

computeCorrespondingDepthPointFromRgb Same functionality as the above mentioned function, however mapping RGB points to depth points. If the RGB points are undistorted, they need to be re-distorted prior to the lookup.

computeCorrespondingThermalPointFromRgb Maps vectors of points from RGB to thermal by means of the chosen rectification method. Returns the depth displacement, minimum distance to the closest homography, and the index of the closest homography, depending on the rectification algorithm.

computeCorrespondingRgbPointFromThermal Provides the reverse map of the above mentioned function.

computeHomographyMapping Base function for mapping points by the means of multiple homographies. May map points in either direction depending on the input vectors. Is called by both `computeCorrespondingThermalPointFromRgb` and `computeCorrespondingRgbPointFromThermal` when the flag USE_PLANAR_HOMOGRAPHIES is set.

Appendix C

Post-processing the calibration images

In Appendix E.2, the acquisition of the calibration images is described. The calibration images are used for camera calibration and the training and testing of image registration algorithms. However, the raw data streams obtained from thermal, visible, and depth camera are useless without any post-processing of the data. The post-processing consists of the synchronization, transformation, and calibration of image streams and the methods used and developed for this purpose are described in Chapter 6.

This chapter will provide a more formal step-by-step description of the post-processing of the calibration images and will present the results of the synchronization, camera calibration, extraction of chessboard corners, and generation of a point cloud.

The following will describe the post-processing of the image streams after the capture described by the measurement journal of Appendix E.2.

C.1 Pre-calibration processing

The pre-calibration processing is the manipulation of image streams to fit into the calibration process. This process is done by the following steps which are repeated for each calibration sequence of the scenes:

1. Thermal frames are extracted from the image stream with the FFMPEG script described in Section 5.2.4[1].
2. Depth images are converted from double-sized 8-bit images to 16-bit images of the native resolution of the depth camera by the MATLAB-function `convertDepthDataToUshort.m`[2]. Results are equivalent to the images shown in Figure 5.3.

[1] Scripts/splitFrames.txt

[2] Scripts/convertDepthDataToUshort.m

3. Depth registration images are created for the RGB-to-Depth correspondence by running the MATLAB-function `Scripts/convertD2RGBtoRGB2D.m`[3].

4. Image frames are synchronized by the MATLAB-function `synchronizeData.m`[4] which also calls `alignThermalFrames.m`[5] to align the thermal time stamps to system time.

C.1.1 Synchronization lag

The synchronization lag for all scenes and calibration sequences is seen from Table C.1. The synchronization lag is measured by comparing the individual time stamps of the system time for each synchronized frame. The frame lag as shown in the left side of the table is calculated by finding the maximum time difference between consecutive frames of each camera. Considering that RGB and depth frames are synchronous at capture time, data for the depth frames is included in the RGB measures.

		Synchronization lag			Largest frame lag	
Scene	Calibration	Max.	Mean	Std.dev.	Thermal	RGB
1	1	31.3 ms	9.31 ms	8.71 ms	374.4 ms	78.0 ms
	2	46.8 ms	9.43 ms	8.70 ms	78.1 ms	93.6 ms
	3	46.8 ms	9.26 ms	8.40 ms	358 ms	93.6 ms
2	1	31.2 ms	9.07 ms	8.48 ms	109 ms	93.6 ms
	2	31.2 ms	9.35 ms	8.99 ms	129 ms	93.6 ms
3	1	62.4 ms	9.53 ms	9.30 ms	218 ms	93.6 ms
	2	46.8 ms	9.34 ms	8.83 ms	359 ms	94.6 ms
	3	31.2 ms	9.13 ms	8.62 ms	359 ms	93.6 ms

Table C.1: Synchronization lag between and frame lag within cameras of the 8 calibration sequences.

The synchronization lag is discussed in the Acceptance test in Chapter 8.

C.2 Calibrating the images

Once the initial post-processing and synchronization of the image streams have found place, the extraction of chessboard corners for the calibration may follow. The extraction of corners lays the foundation for undistorting the images, estimating the camera parameters, and generating training and test data for the registration algorithms.

The calibration process consists of the following steps:

[3] Scripts/convertD2RGBtoRGB2D.m

[4] Scipts/synchronizeData.m

[5] Scripts/alignThermalFrames.m

1. The synchronized folder of thermal images, SyncT, is evaluated, and the last frame for which the corners of the calibration board may be correctly extracted is chosen.

2. This frame number is input to the MATLAB-function `getCalibrationImages.m`[6] alongside the number of frames to be generated.

3. The folder CalibrationHelper is now created with the wanted number of synchronized calibration frames of the visible, thermal, and depth modalities.

4. Inside the CalibrationHelper folder, the Camera Calibration Toolbox for MATLAB[7] [Bouguet, 2004] is run. The calibration process consists of the following steps:

   (a) RGB images is read, and the extraction of grid corners is done manually. The grid size of the calibration board is 35 mm.

   (b) Once all images are extracted, the camera is calibrated using the 'Calibrate' button.

   (c) Once the calibration is finished, all corners are reprojected on the images. Problematic images are detected, where the reprojected corners do no fit on the actual corners. For those images, the grid corners are re-extracted.

   (d) Another calibration run is done, and corners are reprojected on images again. The images for which the reprojected corners still does not fit are excluded from the calibration through the button 'Add/Suppress images'. The final number of active images is seen from Table C.2.

   (e) As the calibration sequence now only consists of valid calibration images, the corners are recompiled via the 'Recomp. corners' button with a window size of 3x3.

   (f) The third calibration run is done. The image corners and camera parameters are saved by using the 'Save' button. The pixel error of the reprojected image corners and the extracted corners is listed in Table C.2.

5. Step 4 is repeated for both the thermal and visible images. One must make sure that the activated images for the calibration are the same for both the left and right modality. Otherwise, the sub-sequent calibration in OpenCV will fail. The output files from the calibration are renamed with the suffix '_right' or '_left' for the thermal and visible images, respectively.

6. The calibration parameters of the .mat-files are transferred into text files readable by OpenCV by the MATLAB-function `saveCalibVarAsText.m`[8].

Once the calibration process is completed, the image coordinates of the corners of the calibration board are saved in text files in the CalibrationHelper folders of the respective calibration sequences. The following six files are generated:

[6] Scripts/getCalibrationImages.m

[7] A slightly modified version of the toolbox is used, which is found at Scripts/toolbox_calib .

[8] Scripts/saveCalibVarAsText.m . The function is dependent on saveCalibrationPoints-Depth.m .

Scene	Calibration	Active chessboards	Pixel error (x,y) RGB	Thermal
1	1	25	0.560, 0.494	0.340, 0.354
	2	24	0.357, 0.319	0.338, 0.359
	3	21	0.330, 0.290	0.492, 0.504
2	1	23	0.277, 0.258	0.277, 0.258
	2	22	0.613, 0.469	0.406, 0.452
3	1	29	0.308, 0.319	0.425, 0.512
	2	28	0.314, 0.307	0.317, 0.342
	3	24	0.334, 0.340	0.338, 0.357

Table C.2: Number of active chessboards for each calibration sequence and the pixel error of the reprojection of corners by the Calibration Toolbox for MATLAB.

- `lChessboard3D_imagec.txt`, `rChessboard3D_imagec.txt`
  These files contains the image coordinates (in pixels) of the chessboard views for the RGB camera (l) and the thermal camera (r). Only coordinates of the 'active' calibration views are stored.

- `lChessboard3d_object.txt`, `rChessboard3D_object.txt`
  These files store the object coordinates of the chessboards as seen from either the RGB or thermal camera. The object coordinates are the 3D coordinates of the calibration board, in coordinates defined by the coordinate frame of the calibration board. The extrinsic parameters of the camera define the translation and rotation matrices that transforms these coordinates into the camera coordinate frame and ultimately, into image coordinates.

- `lChessboard3D_nbrCorners.txt`, `rChessboard3D_nbrCorners.txt`
  The line of active images for the calibration of the RGB and thermal camera are listed in the above files. Also, the number of corners in both the x- and y-directions are listed for every active chessboard of the calibration.

The above listed files are used in the implementation of the registration algorithm in OpenCV[9].

C.2.1 Calibration in OpenCV

The implementation of calibration algorithms is found to be more robust in OpenCV than in the Calibration Toolbox for MATLAB, which is why the calibration and undistortion of the thermal and visible imagery is chosen to be implemented in OpenCV. Even though the intrinsic parameters of the thermal and visible cameras do not change, the parameters are estimated for each calibration scene. In order optimize the estimation of calibration parameters for each scene, the OpenCV implementation uses point correspondences for both calibration sequence 1 and 2. The calibration is

[9]Visual C++ project solution file at Code/DualCalibration/DualCalibration.sln

computationally very expensive at might take up to one minute on a modern, dual-core laptop[10]. Once the calibration is performed, the calibration parameters are stored in the yml-file `intrinsics.yml` at the CalibrationHelper folder of calibration sequence 1 for each scene. The calibration parameters obtained for scene 1 – 3 are listed in Table C.3 and C.4.

Scene	7Calibration	Intrinsic parameters Focal length (x,y)		Principal point (x,y)	
1	RGB	489.1	488.5	334.9	231.6
	Thermal	1254	1237	319.4	238.0
2	RGB	499.0	327.8	327.8	497.3
	Thermal	2905	3318	319.2	237.9
3	RGB	500.6	501.1	333.0	232.6
	Thermal	2142	2034	319.4	240.1

Table C.3: Intrinsic camera parameters for the visible and thermal cameras, determined for each scene.

Scene	Calibration	Distortion parameters K1	K2	P1	P2	K3
1	RGB	1.501e-1	-4.589e-1	1.715e-2	3.192e-3	3.625e-1
	Thermal	-1.115	-5.169	9.569e-3	5.063e-3	6.193e1
2	RGB	8.267e-2	-1.163e-1	-2.336e-2	-6.885e-3	9.362e-3
	Thermal	-7.223	2.277	-9.150e-2	4.854e-2	1.542e3
3	RGB	1.142e-1	-1.510e-1	1.492e-2	3.208e-3	2.940e-2
	Thermal	-9.706e-1	-75.00	2.981e-2	8.880e-2	1.185074e3

Table C.4: Distortion parameters of the thermal and visible cameras, according to the 5-parameter distortion model of OpenCV.

[10]HP Probook 6560b, w. Intel 2'nd generation Intel Core i5-2520M @ 2.5 GHz

Appendix D

Performance of stereo rectification algorithms

In this Appendix, the performance of the stereo rectification algorithms is measured. The stereo rectification algorithms are described in Section 7.2 and includes calibrated stereo rectification, uncalibrated stereo rectification, and a method on depth refinement of these methods. In the following, the performance of the algorithms are measured according to the procedure of the acceptance test specification of Chapter 4. However, due to the instability and poor performance of the methods, these are not included in the general acceptance test of the registration algorithms. The set-up for the tests described in this set-up is similar to the one of the acceptance test of Chapter 8. All validation images, their point correspondences, and the rectified point correspondences are found in Logs/Stereo .

The stereo rectification algorithms are compared to the baseline specified in the requirement specification of Chaper 3. The performance of the baseline is on the validation set is seen from Table D.1. The baseline is computed by the OpenCV `findHomography` function which relies on a threshold for filtering outliers in the RANSAC-estimation method used. The homography is computed using the training samples of each scene and validated against the test sample. Thresholds for the RANSAC estimation are validated in the range from $0 - 30$, and the threshold resulting in the best fit for each scene is chosen.

D.1 Calibrated stereo

The method on calibrated stereo rectification as described in Section 7.2.1 is tested against the validation dataset of Scene $1 - 3$. The transfer errors of the validation points are listed in Table D.2 for both the One-dimensional Transfer Error (OTE) and the Symmetric Transfer Error (STE). From the transfer errors of the table it is easily seen that the rectification algorithm performs very poorly. As seen from Figure D.1a and D.1b, the rectified views of the modalities are distorted in ways that renders the images useless. This unfortunate transformation of the imagery might be due to multiple causes, which are hard to identity. The most likely reason is that the iterative estimation 'runs away' due to a misfit of the corresponding point sets.

Scene			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
Scene 1	OTE	rgb→t	33.29	71.08	73.59	85.06	104.4	116.8
Planar 12		t→rgb	25.43	50.86	52.83	60.62	76.28	84.29
	STE	rgb↔t	58.72	121.9	126.3	145.6	180.7	201
Scene 2	OTE	rgb→t	9.546	10.01	16.77	13.89	19.56	22.74
Planar 5		t→rgb	7.397	7.757	12.8	10.32	15.15	17.1
	STE	rgb↔t	16.94	17.77	29.53	24.09	34.71	39.69
Scene 3	OTE	rgb→t	28.45	41.53	37.63	87.72	69.98	103.6
Planar 30		t→rgb	23.15	33.41	30.02	67.64	56.56	79.2
	STE	rgb↔t	51.6	74.93	67.5	155.3	126.5	182.6

Table D.1: Transfer errors for the 'baseline' rectification method. The Number listed after 'Planar' is the RANSAC threshold used in obtaining the single planar homography.

Scene			Algebraic error (x,y) Mean		Std.dev		Geometric error Mean	Std.dev
1	OTE	rgb→t	3.162e04	1.764e04	3.937e04	1.651e04	4.925e04	4.713e04
		t→rgb	1.241e05	4.868e04	9.062e04	4.377e04	1.728e05	9.012e04
	STE	rgb↔t	1.557e05	6.632e04	9.474e04	4.213e04	2.221e05	9.764e04
2	OTE	rgb→t	1.653e05	7.174e04	1.09e05	5.745e04	2.37e05	1.202e05
		t→rgb	1.285e05	5.079e04	8.512e04	4.275e04	1.793e05	9.31e04
	STE	rgb↔t	2.938e05	1.225e05	1.936e05	1.002e05	4.163e05	2.127e05
2	OTE	rgb→t	9.344e04	5.66e04	1.029e05	6.122e04	1.5e05	1.196e05
		t→rgb	8.683e04	6.936e04	8.332e04	5.094e04	1.562e05	8.263e04
	STE	rgb↔t	1.803e05	1.26e05	1.772e05	9.715e04	3.062e05	1.784e05

Table D.2: Transfer errors for validation points by using calibrated stereo rectification, in pixels. As seen by the numbers, the algorithm is malfunctioning for all scenes. The errors are measured with a scaling parameter of the stereo rectification of 0.8, which is mentioned in Section 7.2.1. Varying the scaling parameter, however, does not change the performance of the algorithm radically from what is listed above.

However, even when multiple sets of corresponding points of the different calibration sequences are provided for the rectification algorithm, the results are similar. Therefore, one must conclude that the stereo rectification algorithm is unstable for the dataset provided and thus must be discarded.

(a) Rectified thermal image **(b)** Rectified RGB image

Figure D.1: 'Rectified' views of the Thermal and RGB views provided by the calibrated stereo algorithm. It is seen clearly from the two views, that the iterative methods of the stereo rectification has caused a runaway of the rectification.

D.2 Uncalibrated stereo

The method on uncalibrated stereo rectification is outlined in Section 7.2.2. As uncalibrated stereo rectification depends on robust estimation to calculate the fundamental matrix and rectifying homographies, the best values for the RANSAC estimation threshold has to be determined by trial and error. Two thresholds are to be considered:

- The RANSAC threshold for estimating the fundamental matrix
- The threshold for performing uncalibrated rectification

Both thresholds have been tested individually for each scene in the interval [0;30], and the thresholds resulting in the lowest transfer errors have been chosen. These are seen from Table D.3

Scene	Fundamental matrix	Uncal. rectification
1	3	0
2	2	0
3	6	0

Table D.3: Thresholds chosen for the robust estimation of the fundamental matrix and the matrices for uncalibrated rectification.

The results for the validation of the algorithm is seen from Table D.4. It is seen from the transfer errors of the table that the uncalibrated stereo rectification algorithm is functional as opposed to the implementation of the calibrated stereo rectification. From the geometric mean and standard deviation of the STE, the transfer

errors of the uncalibrated stereo rectification for scene 1 and 2 are slightly above the corresponding errors of the baseline. The transfer errors between modalities of Scene 2 are substantially lower than what is achieved by the baseline method. However, when compared against the training set, the uncalibrated stereo rectification results in a 62 percent increase in the average geometric error.

Scene			Algebraic error (x,y) Mean		Algebraic error (x,y) Std.dev		Geometric error Mean	Geometric error Std.dev
1	OTE	rgb→t	35.79	71.43	61.67	107	107.2	125.4
		t→rgb	27.24	49.02	46.43	69.68	76.26	84.92
	STE	rgb↔t	63.03	120.5	107.9	176.4	183.5	209.8
2	OTE	rgb→t	4.506	8.206	6.667	11.75	12.71	14.08
		t→rgb	3.458	6.63	5.014	9.478	10.09	11.2
	STE	rgb↔t	7.964	14.84	11.65	21.18	22.8	25.19
2-Training	OTE	rgb→t	27.29	26.01	57.63	69.61	53.3	111.2
		t→rgb	20.82	22.63	44.68	70.08	43.45	104.3
	STE	rgb↔t	48.11	48.63	101.5	139.3	96.75	214.1
3	OTE	rgb→t	44.07	33.35	81.9	72.54	77.42	108.3
		t→rgb	36.08	27.76	67.93	60.63	63.84	90.67
	STE	rgb↔t	80.15	61.11	149.8	133.2	141.3	198.9

Table D.4: Transfer errors for the uncalibrated stereo rectification.

The performance of the algorithm on two of the validation images of Scene 3 is seen from Figure D.2. The mapping errors are varying depending on the depth and the position in the image of the object. Larger errors tend to occur at depths far from 'the middle' of the scene and at the image corners. Even though the performance of

(a) RGB validation image 8 **(b)** RGB validation image 20

Figure D.2: Validation images of Scene 3, along with the mapped points of the chessboard corners from the thermal image (red). The ground truth points of the calibration board are shown in green.

the uncalibrated stereo rectification is better on the planar test set of scene 2, the method is not included in the overall acceptance test due to the bad performance of the algorithm in the remaining scenes. The image rectification should at least be better than the simple technique of using a single homography. Furthermore, the uncalibrated stereo rectification tends to suffer from the same problems with parallax

as the single homography, which means that objects far away from the middle of the registered scene plane are subject to large rectification errors.

D.3 Depth refinement

The technique of depth refinement is described in Section 7.2.3. The technique might be used for both the calibrated and uncalibrated stereo rectification, but due to the poor performance of the calibrated stereo rectification, the depth refinement is only applied on the uncalibrated rectification. The depth refinement technique is based on the physical property that the parallax of the image registration is proportional to the inverted depth from the corresponding point in space to the camera.

In order to investigate this property, the disparity and the depth of point pairs used for training the uncalibrated rectification is processed using OpenCV. The disparity in this context is the rectification error of the corresponding point pair measured in the rectified image space. The rectified image space is obtained by applying the rectification matrices on each image, thus projecting the epipolar lines to infinity. The disparity and the inverted depth for each corresponding point pair is saved to a text file and loaded into the Curve Fitting Tool of MATLAB, from which a linear regression is estimated using robust methods. For each scene, a regression is estimated, and the coefficients of the regression is inserted in OpenCV.

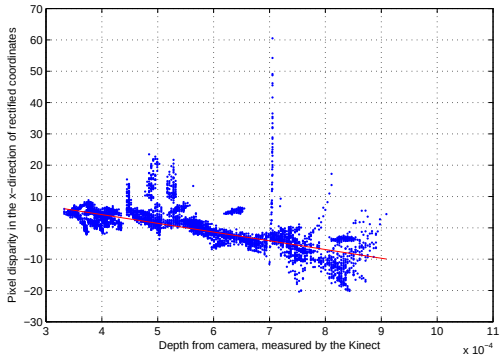
The regressions, along with the corresponding data, are shown from Figure D.3a – D.3c. Is is seen from the graphs that although a relationship exists between the inverted depth and the disparity of the mapping in the x-direction, the relationship is indeed very noisy and filled with outliers. This might be due to the fact that the fundamental matrices are estimated and thus not exact, the difference in field-of-views of the cameras, and noise in the point correspondences used for the estimation. The points in Figure D.3b do not imply this characteristic, however, and the estimated regression is, by visual inspection, very poor. This might be due to the fact that Scene 2 is largely planar, thus minimizing the effect of the depth parallax.

When linear regressions for each scene are estimated, the process of mapping points from one modality to the other by means of the depth refinement technique is then the following:

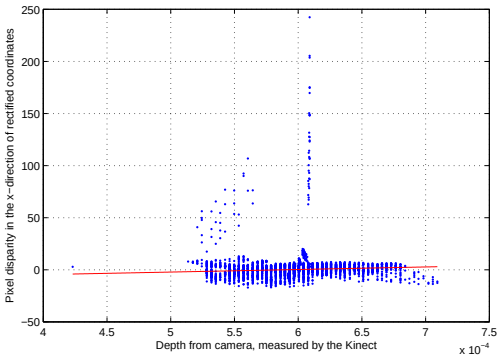
1. Rectify the point according to the uncalibrated rectification algorithm. The point is now in 'rectified' coordinates.
2. Find the depth of the point by mapping the point to the depth image, and look up the depth.
3. Calculate the depth rectification disparity by the following equation:

$$\mathbf{x}'_{\text{rect}} = \begin{cases} x'_{\text{rect}} = x'_{\text{rect}} \pm (a \cdot d + b) \\ y'_{\text{rect}} = y'_{\text{rect}} \end{cases} \tag{D.1}$$

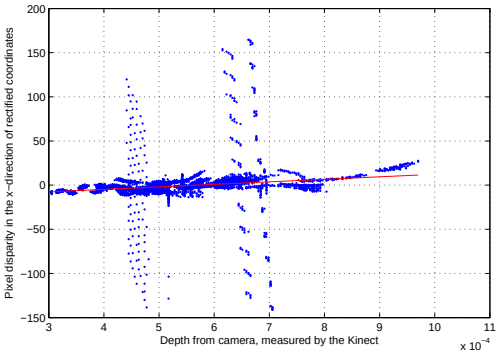
where $\mathbf{x}'$ is the mapped point in *rectified* coordinates, d is the depth from the camera to the point as measured by the Kinect, and a and b are the parameters of the linear regression.



(a) Scene 1. $a = -27858, b = 15.35, r^2 = 0.76$



(b) Scene 2. $a = 24500, b = -14.45, r^2 = 0.87$



(c) Scene 3. $a = 28520, b = -16.23, r^2 = 0.94$

Figure D.3: Relationship between the disparity in the x-direction and the inverted depth of corresponding point sets. For each Scene, a linear regression $(ax + b)$ is estimated.

The ± sign is indicating that the disparity is either subtracted or added to the rectified x-coordinate based on the direction of the mapping. If points are mapped from RGB to thermal, the sign should be negative, and should therefore be positive when mapping thermal points to RGB.

4. Map the point back to unrectified coordinates by applying the inverse homography.

This approach presents a dilemma, though: How to determine the depth of thermal points?

The determination of the depth for thermal points is indeed not possible as the registration of the thermal and depth modalities depends on the registration between the thermal and RGB images - which is what we are estimating here. However, the depth of thermal points might be estimated by assuming that the disparity between the thermal and RGB points in rectified coordinates is close to zero. This means that the rectified thermal points might be treated as RGB points, for which the corresponding depth is found. Once the depth is found, the corresponding depth rectification disparity is computed, and the RGB point is estimated This approach introduces some errors on both the rectification and the depth measurement, as one cannot guarantee that the depths are similar. However, this method gives a pleasing estimate of the depth if objects are relatively large and uniform in depth.

The transfer errors for the depth refinement technique used on the uncalibrated stereo rectification are listed in Table D.5. The refinement methods uses the same thresholds for estimating the fundamental matrices and rectification matrices as the uncalibrated approach. These thresholds are listed in Table D.3. The results of the

			Algebraic error (x,y)				Geometric error	
Scene			Mean		Std.dev		Mean	Std.dev
1	OTE	rgb→t	41.95	62.36	76.61	85.08	104.3	115.8
		t→rgb	27.24	49.02	46.45	69.91	76.26	85.06
	STE	rgb↔t	69.19	111.4	122	126.6	180.6	177.4
2	OTE	rgb→t	7.669	23.1	10.17	36.62	30.77	42.48
		t→rgb	3.506	6.821	5.129	10.39	10.33	12.37
	STE	rgb↔t	11.17	29.92	13.54	43.8	41.09	50.2
3	OTE	rgb→t	45.88	21.37	83.86	56.94	67.25	103.1
		t→rgb	36.07	27.52	67.88	60.2	63.6	90.3
	STE	rgb↔t	81.95	48.9	151.6	107.3	130.8	186.9

Table D.5: Transfer errors for the uncalibrated stereo rectification with the depth refinement technique applied.

refinement technique on scene 1 shows a slight improvement of the average geometric STE compared to the original rectification. However, the standard deviation is down by 32,4 pixels, which is a considerable improvement. The depth refinement is however not working on scene 2. Both the average error and standard deviation of the geometric STE is nearly doubled by using the refinement technique. The performance of the mapping of scene 3 is improved by a 10 pixels for both the mean and standard deviation of the STE.

However, these improvements to the mappings of scene 1 and 3 only sets them *on par* with the performance of the baseline of the planar homography. Even with the refinements, the technique is unable to surpass the performance of a single homography for each scene. Due to the above performance, we will not contribute to any further work on the depth refinement technique.

Appendix E

Measurement records

E.1 Default set up for multi-modal capture

In this measurement record, we will describe the default set up of the camera systems and the acquisition software. This set-up is the minimum configuration for capturing imagery with the thermal, visible, and depth cameras.

E.1.1 Equipment used

The equipment used for the set-up is listed in Table E.1.

Instrument	Manufacturer, type etc.
RGB camera	Microsoft Kinect for XBOX Camera
Depth camera	Microsoft Kinect for XBOX Camera
Thermal camera	AXIS Q1922
12 V DC-adapter	Noname
Laptop	HP ProBook 6560b

Table E.1: Equipment used for the multi-modal setup

E.1.2 Set-up

The Kinect and AXIS cameras are placed neatly in the mount depicted in Figure E.1. Care must be taken to position the cameras such that their horizontal axes align and the field-of-view of the cameras overlap as much as possible.

The Kinect is connected to a Windows-powered PC through a USB cable and powered through the included adaptor whereas the AXIS camera is connected to the same PC through an Ethernet cable and powered through a 12 V DC-adapter. If the scene is dimly lit, external light sources are recommended in order to improve the low-light performance of the RGB sensor of the Kinect.

Figure E.1: The test setup consisting of the AXIS and Kinect cameras.

E.1.3 Procedure

The following is the standard procedure for initializing the multi-modal acquisition platform:

1. The set-up is configured as described in the previous subsection.
2. The AXIS and Kinect cameras are powered and connected to the same PC.
3. Once the cameras are initialized, the recording program RGBDT Capture is opened. In the window of the thermal camera, the proper IP address of the AXIS camera is written and the stream from the thermal camera is started. If the IP address of the AXIS camera is unknown, it may be found by opening the 'Network' folder in Windows Explorer.
4. In the file path text box of the capture program, the file path of the current directory used for the capture is written. Once completed, the 'Use this folder' button is pressed.
5. If both the RGB and thermal streams are shown, the initialization is complete.

E.2 Acquiring calibration images

In this measurement record, we will describe the process of obtaining imagery for calibrating the cameras used.

Purpose

The purpose of this measurement is to provide a sufficient amount of calibration footage for the camera calibration and registration.

Equipment used

The basic equipment used for the measurements is similar to the listings in Table E.1. Furthermore, the equipment of table E.2 is used. The design of the calibration

Instrument	Manufacturer, type etc.
Calibration board	AAU VAP
Heat gun	Nakachi 1500 W

Table E.2: Additional equipment used for acquiring calibration images.

board is described in Section 6.2.2.

Set-up

The basic set up is equivalent to the process described in Section E.1. Furthermore, the procedure of Section E.1 must be followed and the image streams of both the thermal and RGB cameras should be visible in the windows of the acquisition program.

E.2.1 Procedure

The following procedure is repeated for all three scenes defined in the requirement specification of Chapter 3. The exact set-up of these scenes are described in the acceptance test in Chapter 8, and it is for these scenes, that the following procedure is repeated.

1. The set-up is configured as described in the previous subsection.
2. The 'calibrate depth' button is pressed.
3. The calibration mask is cooled either outside or in a fridge prior to calibration capture.
4. The backdrop is heated uniformly by using the heat gun.
5. The mask is placed firmly on top of the backdrop and the button 'Calibrate RGB + T' is pressed
6. The calibration board is swept through the entire scene, thus representing a broad variety of depth and pose angles to the cameras.
7. Once the patterns of the calibration board are barely visible from the thermal camera (approx. 1 min), the 'Stop capture" button is pressed.
8. The 'calibration sequence number' is incremented and step 2-6 is repeated until three calibration sequences are recorded.

E.2.2 Results

The images may be found on the enclosed CD Dataset/[1;3] . Samples from the measurement is seen in Figure 6.7 on page 42.

E.2.3 Uncertainty of measurements

The quality of the images is limited to the physical properties of the cameras such as focal length, lens size, and resolution. If the room is not sufficiently lit, the colour quality of the Kinect RGB camera will suffer greatly, thus making the process of extracting corners from the calibration rig difficult. The treatment of the reprojection errors and the evaluation of the spatial positioning of the calibration rig is made in Appendix C.