Aalto University
School of Science

Jussi Sainio

Multi-Camera Object Tracking and Camera Calibration in a Video Content Analysis System

Thesis submitted for examination for the degree of
Master of Science in Technology.

Espoo 11.12.2013

Thesis supervisor: Prof. Harri Ehtamo

Thesis advisor: M.Sc. (Tech.) Markus Kuusisto

AALTO-YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Jussi Sainio

Työn nimi: Kohteiden seuranta usealla kameralla ja kameran kalibrointi videosisällönanalyysijärjestelmässä

Päivämäärä: 11.12.2013 Kieli: Englanti Sivumäärä:8+46

Matematiikan ja systeemianalyysin laitos

Professuuri: Sovellettu matematiikka Koodi: Mat-2

Valvoja: Prof. Harri Ehtamo

Ohjaaja: DI Markus Kuusisto

Konenäön ja videosisältöanalyysin tavoitteena on purkaa merkityksellistä dataa videodatasta. Yksi yleisistä konenäön haasteista on kamerakalibrointi, jota vaaditaan fyysisten pituusmittojen johtamiseen kamerakuvasta. Kalibrointia tarvitaan myös monikamerajärjestelmissä, missä kameroiden näkemät alueet osittain peittävät toisiaan.

Tässä työssä kehitetään laskennallisesti kevyt menetelmä kameroiden kalibroimiseksi kolmiuloitteeseen avaruuteen. Esitelty menetelmä on kaksivaiheinen. Ensimmäisessä vaiheessa linssivääristymät ja kameran sisäiset parametrit (polttoväli) mitataan ottamalla kuvia tunnetusta shakkilautakuviosta eri asennoissa. Toisessa vaiheessa, ulkoiset parametrit (kallistuskulma ja kameran korkeus suhteessa maatasoon) määritetään muutamasta kohdetunnistuksesta. Simuloimalla näytetään, että toisen asteen estimointi pikselihorisontille tuottaa merkittävästi parempia tuloksia kallistetuilla kameroilla, verrattuna yleisiin katoamispistemenetelmiin tai ensimmäisen asteen horisonttiestimaattisovituksiin.

Lisäksi työssä kehitetään menetelmä ylhäältä alas kuvaavien monikamerajärjestelmien kalibrointiin sekä kohteiden seuraamiseen niissä. Tämä menetelmä käyttää sovitettuja pistepareja toisiaan leikkaavissa kamerakuvissa, joista lasketaan similariteettimuunnokset. Muunnosten avulla kohdeseurannan polut yksittäisistä kameroista muunnetaan yleiseen koordinaattiavaruuteen. Yksittäiset polut merkitään samaksi, jos ne risteävät toisensa tässä avaruudessa.

Työn menetelmiä ja tuloksia voidaan hyödyntää esimerkiksi asiakasreittien mittaamiseen myymälöissä tavallisilla turvakameroilla, tai kohteiden luokitteluun niiden fyysisen koon perusteella.

Avainsanat: konenäkö, piirteenirroitus, kamerakalibraatio, ulkoiset parametrit, videosisältöanalyysi, videovalvonta

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Jussi Sainio

Title: Multi-Camera Object Tracking and Camera Calibration in a Video Content Analysis System

Date: 11.12.2013 Language: English Number of pages:8+46

Department of Mathematics and Systems Analysis

Professorship: Applied Mathematics Code: Mat-2

Supervisor: Prof. Harri Ehtamo

Advisor: M.Sc. (Tech.) Markus Kuusisto

The objective of computer vision and video content analysis is to extract meaningful feature data from video data. One of the common challenges in computer vision is camera calibration, which is required for converting physical length measures from the camera image. Calibration is also required in multi-camera systems, where camera images from different cameras partly overlap.

In this thesis, a computationally lightweight method for calibrating cameras in three-dimensional space is developed. The proposed method is divided in two stages. In the first stage, lens distortions and camera intrinsic parameters (focal length) are measured with taking images of a known chessboard pattern in various positions. In the second stage, extrinsic parameters (tilt angle and camera height relative to a ground plane) are determined from few object detections. Using simulation, a second order pixel horizon estimation is shown to perform significantly better with tilted cameras than a common method of using vanishing lines or first-order fit for horizon estimation.

In addition, a method for calibrating and object tracking in top-down multi-camera systems is developed. This method uses matched point pairs for overlapping camera images, from which similarity transforms are calculated. The transforms are used to convert object tracking trails from individual cameras onto a global coordinate space. Individual trails are then marked joinable, if they cross each other close enough in that space.

The methods and results of this thesis are usable for example when measuring customer routes in stores using common video surveillance cameras, or for classifying objects based on their physical size.

Keywords: computer vision, feature extraction, camera calibration, extrinsic parameters, video content analysis, video surveillance

Acknowledgements

The excellence of a residence is in (the suitability of) the place; that of the mind is in abysmal stillness; that of associations is in their being with the virtuous; that of government is in its securing good order; that of (the conduct of) affairs is in its ability; and that of (the initiation of) any movement is in its timeliness.

— 老子道德經 (Lǎozǐ: *Dàodéjīng*, tr. James Legge)

This thesis was done at Mirasys Ltd., which is a globally operating supplier of digital video surveillance systems. I want to thank my instructor Markus Kuusisto for his helpful and supportive insight, and outstanding patience. I also want to thank professor Harri Ehtamo for supervising the thesis.

Last but not least, I say thanks to my parents for always caring and raising the bar to be successful in life.

In Espoo, 11 December 2013

Jussi Sainio

Contents

Symbols and Abbreviations

Symbols

Φ	vertical angle of view
f	focal length of the camera-lens system
y_c	camera height above the ground plane
θ_x	camera tilt angle
u,v,w	camera image (homogeneous) coordinates
x,y,z,W	world (homogeneous) coordinates
v_b	object bottom coordinate in image coordinates
v_t	object top coordinate in image coordinates
y_o	object height in world coordinates
v_h	pixel/ground plane horizon in camera image coordinates
c	abbreviation of $\cos\theta_x$
s	abbreviation of $\sin\theta_x$
$\hat{z}$	object z coordinate estimate
α	object z coordinate estimate scale factor

Abbreviations

3D	three-dimensional
BFS	breadth-first search
CCD	charge-coupled device
CCTV	closed-circuit television
DLT	direct linear transformation algorithm
DVR	digital video recorder
FOV	field-of-view
GPS	global positioning system
GUI	graphical user interface
LBP	local binary pattern
LED	light-emitting diode
MoG	mixture of gaussians
RANSAC	random sample consensus
RGB	red, green, blue
SVD	singular value decomposition
VCA	video content analysis
VGA	video graphics array, used here to refer to the video resolution of 640x480

1 Introduction

Es gibt Systeme.

— Niklas Luhmann: *Soziale Systeme*

1.1 Background

A camera is a universal sensor. Digital video cameras produce vast amounts of information in a very short period of time. To transform this massive amount of information into more useful, smaller amounts of meaningful information is the purpose of computer vision research.

Data mining and data farming are popular themes today. They mean either combing through or producing large amounts of data to uncover interesting phenomena for fun and profit. When you have a lot of data, relatively simple techniques can be applied to achieve surprising results. For instance, statistical methods have been successfully applied to machine translation (Google), automated contextual spell, style and grammar checking (After the Deadline).

Traditionally, video surveillance has relied on computer vision motion detection algorithms and then saving the video where motion is detected for later analysis to be done by a human. Today, the industry is moving into more fine-grained *video content analysis* (VCA) for producing mineable data. Computer vision and video analysis has been a vibrant research field since the 1970's, and lot of interesting algorithms are already available for this. However, often these are unsuitable for real-time multi-camera systems with constrained processing power or the data is not interesting in surveillance or retail marketing.

New technology opens new possibilities and often, these possibilities need to be limited with regulations and laws. For example, video surveillance has increased drastically in central Europe and, in response, privacy laws in many countries have been updated to be more strict than before. Raw surveillance video material has limits on its use, but it can contain valuable information, which can be extracted and anonymized using computer vision.

1.2 Objective of the thesis and the challenges involved

This work concentrates on building a workflow for camera calibration and object tracking data extraction in video surveillance systems. Many algorithms and methods have been developed for both camera calibration and feature extraction, but none of those featured in the literature review fulfill usability, speed and implementation requirements of set for this work. Some these requirements are discussed below, and later more in detail in the section 3.1. Only few solutions for the whole workflow exist. The objective is to facilitate new sources of measurement data through camera calibration for automated analysis. The analysis of the measurement data, such as object classification or customer behaviour analysis, is outside the scope of this thesis.

This thesis builds upon a video content analysis system developed at Mirasys Ltd. The video content analysis system has already motion detection and object tracking algorithms for single cameras in place and they work well. This work augments the capability of the system by designing and implementing a new camera calibration framework, algorithms and a methodology for it.

The purpose of camera calibration is to provide real world coordinate data from tracked objects, so that the 3D position and speed of moving objects can be extracted and used for further analysis. The scope is limited to simple plane approximation, where the 3D position can be correctly approximated only for objects moving on the ground.

1.2.1 Real-time operation

Since most video surveillance applications feature multiple cameras, the algorithm efficiency and lightweightness is an important factor, when real-time operation is desired. Many, if not most, of the published motion detection and object tracking algorithms and methods are too computationally intensive to be used in real-time multi-camera operation on today's commodity desktop and server computers. In embedded appliance space (e.g. "smart" cameras with on-board processing units), the computational resources are even more constrained, and thus the low computational complexity of the algorithm becomes important.

If camera calibration is to be done separately and not as a part of the system, algorithm efficiency is not that big problem. But also here real-time or near real-time operation is desired, whilst doing the actual calibration.

1.2.2 Implementability

Often, sophisticated solutions require sophisticated mathematical algorithms. These algorithms may only exist as closed source binary libraries for certain platforms. This makes them impossible to port to new platforms, and they may be very time-consuming to implement and test from scratch.

The objective of this work is to come up with relatively simple and effective solutions, so that the implementation can be reasonably easily made on any computing platform and environment. Closed-source libraries are best avoided [24].

As software becomes more complex, clean architecture and modularity become necessary for maintainance and even for development.

1.3 Content of the thesis

This thesis describes a method for calibrating camera extrinsic parameters. The proposed method consists of two phases, namely:

1. Use a checkerboard to determine the camera focal length f (or the field of view angle), and determining lens distortion coefficients.
2. Use three or more image positions of objects with known real world height to determine the camera height y_c and tilt angle θ_x.

At first, the thesis defines the necessary coordinate systems, variables and angles. Then we take a look at a pinhole camera in 3D world, and finally transform our calibration problem into 2D world problem, which is solved.

Section 2 reviews the earlier work in this field and looks at the most common methods used for object tracking and camera calibration. Section 3 defines the objectives for this thesis and discusses the challenges involved. The proposed workflow is presented, how the algorithms and methods are organized. Section 4 defines necessary coordinate systems for a pinhole camera. These definitions are later used in the camera calibration method. Section 5 develops a lightweight method for camera calibration in 3D space. Section 6 develops a method for calibrating top-down multi-camera systems. Section 7 presents test results obtained and compares them to other methods. Finally, section 8 discusses the results and concludes the thesis.

2 Literature review on video surveillance systems and camera calibration

If I have seen further, it is by standing on ye sholders of Giants.

— Sir Isaac Newton *(in a letter to Robert Hooke)*

This section reviews published algorithms and methods for object tracking and camera calibration. Advantages and disadvantages for methods are discussed.

The section starts with an terminology overview, discusses existing research for overall system architecture, motion detection and single camera object tracking algorithms. The section then reviews existing calibration methods for both intrinsic and extrinsic camera parameters. Finally, the section discusses about multi-camera object tracking methods and alternative camera technologies.

In a camera system, lens distortions affect the measurement accuracy. For example, an object at the same distance from the camera might have its features (such as width and height in pixels) distorted at the edge of the camera view compared to the center of the camera view. This can make measuring the object size fairly inaccurate, if an ideal pinhole camera model is used. Existing research provides good methods for making an average camera-lens system perform closer to an ideal pinhole camera, almost free of distortions.

2.1 Terminology

The definitions of the terminology is limited to what they mean in this thesis.

A *blob* is a set of pixels that is segmented apart from an image.

Intrinsic parameters are the internal geometric and optical characteristics of a camera [12]. These include focal length, projection center and lens distortion parameters.

Extrinsic parameters are the three-dimensional position and orientation of the camera image plane relative to a certain world coordinate system [12].

Camera calibration means the process of figuring out intrinsic and extrensic parameters of a camera or set of cameras.

Vertical angle of view is the angle that the camera projects onto image plane on vertical axis.

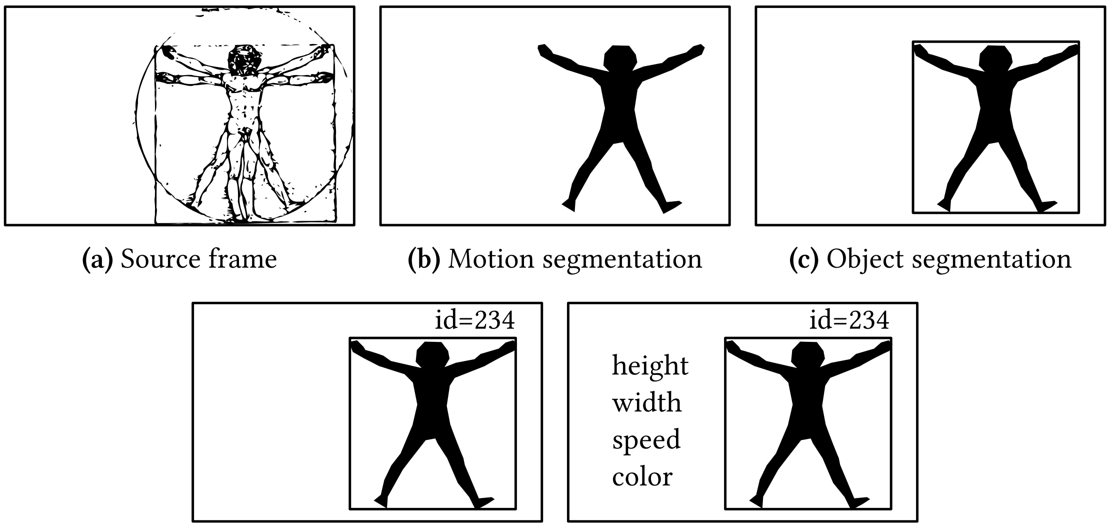
Focal length is a measure of how strongly the camera lens converges light. Focal length f is related to the vertical angle of view Φ such that [29, p. 53]

$$f = \frac{d}{2\tan\left(\frac{\Phi}{2}\right)}, \tag{1}$$

where d is half of the vertical image resolution or half of the physical height of the image sensor. This depends whether the focal length is measured in image resolution units (pixels) or physical units.

Camera handoff is the moment when object tracking crosses a multi-camera boundary, and the object tracking is continued by another camera.

2.2 Overview of video surveillance system architectures



(a) Source frame **(b)** Motion segmentation **(c)** Object segmentation

(d) Object labeling **(e)** Feature extraction

Figure 1: Stages of video surveillance content analysis.

A typical architecture of a video surveillance system with video content analysis is shown in [32, p. 6] and [15].

A video surveillance system contains a number of cameras. Cameras produce video stream. Video stream is analysed by content analysis system, which is usually a computer or a set of networked computers. Usually at first, the content analysis does motion detection and segmentation for the video stream. This produces pixel blobs which denote moving objects (foreground). Pixel blobs are classified into objects, which are tracked over time. [32, p. 6] [15]

Cameras can be stationary or equipped with zoom lenses and pan-tilt functionality (dome cameras). They can be analog or digital, connected over wire or wireless. For processing, the video is always converted to a digital format, which is often compressed. Video stream resolutions range typically from VGA (640x480) to multiple megapixels. Typical video surveillance framerates range from 1 to 30 frames per second.

Modern video surveillance systems analyse the video stream automatically. One of the most important advantages of a modern video surveillance system is that it only records when there is movement in the video. Various methods have been developed for detecting motion in video. Most common approach is to use background subtraction, where the background of a scene is modeled [23]. When a part of an image differs enough from the background, motion is detected.

Various methods have been developed to discern unwanted detection from e.g. shadows or movement of tree branches in outdoor scenes. [20] Motion is segmented into motion blobs. Good motion segmentation is helps object tracking. Due to occlutions, one pixel blob can contain multiple objects. The task of object segmentation and tracking is to separate the motion pixel blobs into tracked object blobs. [32, p. 5]

Object blobs can be subjected to more in-depth feature extraction. Typical features that are of interest are location, speed, loitering time (how much time an object is staying at the same place), color, type classification (whether the object is a human, a car, or something else).

2.3 Motion detection using background subtraction

There are many methods available for motion detection. An extensive literature review into motion detection algorithms is done by Kuusisto [20]. Also Moeslund et al. [23] has an extensive survey on motion capture and analysis algorithms, which cites over 350 papers.

According to Moeslund, Mixture-of-Gaussians (MoG) is still the most common method for motion detection and background subtraction [23]. However, Kuusisto proposes a motion detection and segmentation algorithm, which is based on Local Binary Pattern (LBP) texture reconstruction, which is superior to the reviewed algorithms in terms of processing time and accuracy.

Since this motion detection and segmentation algorithm is texture reconstruction based, it is quite insensitive to illumination changes (such as shadows) in the scene, it is better than the MoG-based method for surveillance purposes. Kuusisto doesn't provide an objective measure for algorithm performance comparison, but backs this claim instead on experience in testing the algorithm on real-life surveillance video recordings and assessing the results manually. [20]

Further review of motion detection algorithms is outside the scope of this thesis, as this thesis builds upon the work done by Kuusisto [20].

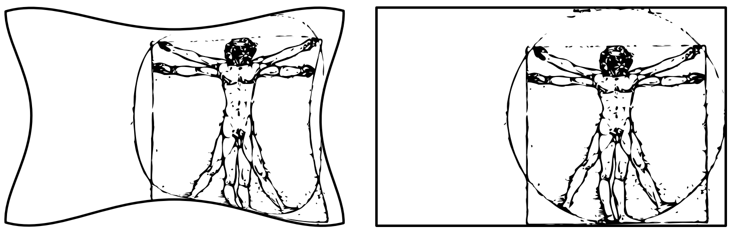
2.4 Object tracking with a single camera

The single object tracking algorithm uses segmentation, which discerns motion detected blobs. Then, using vanishing point horizon estimation [13] and Kalman filtering for estimating the object blob movement, the object blobs are tracked and labeled accordingly, even if they occlude each other.

Further review of single camera object tracking algorithms is outside the scope of this thesis.

2.5 Calibration of intrinsic parameters

2.5.1 Lens distortion removal



(a) Distorted source image **(b)** Rectified source image

Figure 2: An example of severe pincushion lens distortion.

An ideal pinhole camera has no distortions and the projection is perfect. However in practice, most camera lenses introduce significant distortions to the image. These distortions make the projection imperfect, which hampers accurate measurement of propotions from the camera image. Fortunately, the most drastic geometric distortions can be modeled, measured and corrected.

A good geometric distortion model captures as much of the distortions in few terms. Due to the manufacturing process and physics of inexpensive CCD video cameras, most of these distortions fall in to two classes: radial distortions and tangential distortions. [1, p. 377]. The correspondence of between a rectified image and an image with radial and tangential distortions can be expressed mathematically as Brown's lens distortion model [2][34][33]

$$\begin{aligned} x_{\mathrm{u}} &= x_{\mathrm{d}} + x_{\mathrm{dc}}(K_1 r^2 + K_2 r^4 + \ldots) \\ &\quad + (P_1(r^2 + 2x_{\mathrm{dc}}^2) + 2P_2 x_{\mathrm{dc}} y_{\mathrm{dc}})(1 + P_3 r^2 + \ldots) \end{aligned} \tag{2}$$

$$\begin{aligned} y_{\mathrm{u}} &= y_{\mathrm{d}} + y_{\mathrm{dc}}(K_1 r^2 + K_2 r^4 + \ldots) \\ &\quad + (P_2(r^2 + 2y_{\mathrm{dc}}^2) + 2P_1 x_{\mathrm{dc}} y_{\mathrm{dc}})(1 + P_3 r^2 + \ldots), \end{aligned} \tag{3}$$

where $(x_{\mathrm{u}},\ y_{\mathrm{u}})$ is a rectified image point, $(x_{\mathrm{d}},\ y_{\mathrm{d}})$ is a distorted image point, $(x_{\mathrm{c}},\ y_{\mathrm{c}})$ = centre of distortion (i.e. the principal point), K_n is the n^{th} radial distortion coefficient, P_n is the n^{th} tangential distortion coefficient, $x_{\mathrm{dc}} = x_{\mathrm{d}} - x_{\mathrm{c}}$, $y_{\mathrm{dc}} = y_{\mathrm{d}} - y_{\mathrm{c}}$, and $r = \sqrt{(x_{\mathrm{d}} - x_{\mathrm{c}})^2 + (y_{\mathrm{d}} - y_{\mathrm{c}})^2}$

Employing Brown's lens distortion model is fairly standard practice in computer vision, especially in stereo vision. Good rules of thumb for the number of correction coefficients exist. According to Heikkilä [12], two coefficients for both radial and tangential distortion are often enough. Bradski et al. [1, p. 375] notes that a third radial distortion term k_3 might be needed for cameras equipped with highly distorting fish-eye lenses.

For object tracking purposes, we can either do the lens distortion removal for the whole image, before any motion is detected (more computionally intensive) or just for the metadata (less computionally intensive).

When the lenses and cameras are manufactured similar enough, a common calibration database for each camera model/lens combination can be constructed. Then the individual cameras don't need to be calibrated anymore for lens distortion removal. This reduces the installation costs compared to individual calibration and still should produce comparable rectification results. Professional photography systems contain similar lens databases for automatic lens distortion and chromatic aberration removal.

2.5.2 Checkerboard calibration

Each camera and lens combination is more or less unique. Thus the coefficient parameters in Brown's lens distortion model have to be calibrated to achieve good distortion removal. As this calibration problem is very common, methods for achieving a good calibration with usual camera and lens combinations already exist. [1, p. 375][34][33]

A popular method for determining these coefficients is calibration with a checkerboard. An implementation for such calibration method can be found in the open source OpenCV library. [1] This includes detecting checkerboard corner positions from camera images and calculating the coefficients from these.

If the checkerboard pattern size is known in physical length units, this method also yields the focal length of a lens-camera combination. A suitable calibration checkerboard pattern can be easily produced using a common office printer and a sufficiently rigid plate. [1]

Figures 3 and 4 show an example of the calibration process and distortion removal result, respectively. In this case, a checkerboard pattern is printed on paper with a laser printer and attached with tape on a master's thesis book. The paper is slightly wavy, which makes slight imperfections to the positions of the corner positions. Some 10-15 pictures of the checkerboard are taken from various angles.

Checkerboard calibration is not the only possible method, but it is one of the most practical and cost-effective methods. The calibration results are good enough for all-round CCTV surveillance purposes. The calibration method produces not only the required coefficients for Brown's lens distortion model but also provides an estimate for the focal length of the camera-lens system. This focal length estimate is essential for the camera calibration method developed in this thesis.

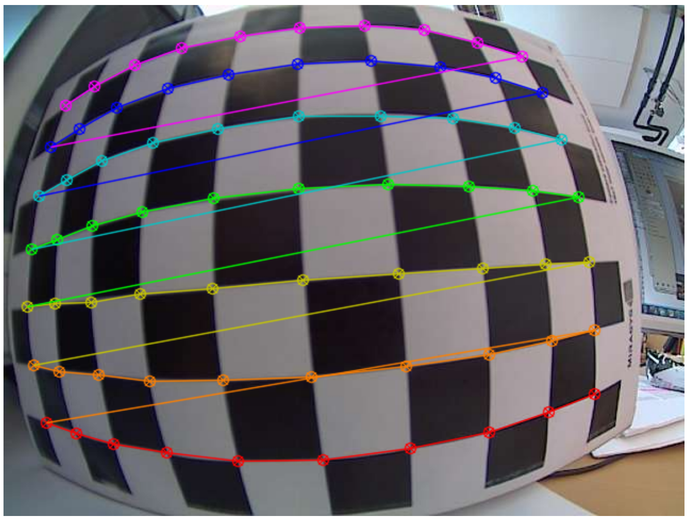


Figure 3: Heavily distorted image of a checkerboard calibration board. The image is distorted by the fish-eye lens in the camera. Detected checkerboard pattern is visualized with colored lines.

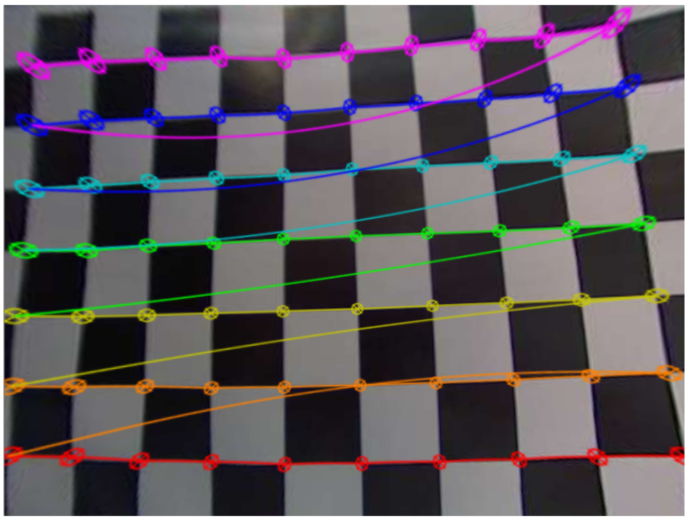


Figure 4: Rectified image of the same scene. Now the checkerboard lines are quite well straight, as they should be, and the overall performance is much closer to an ideal lens.

2.6 Calibration of extrinsic parameters

2.6.1 Manual calibration

In a commercially available VCA library, camera intrinsic and extrinsic matrix calibration is done manually. The patented method relies on a graphical 3D model representation of reference objects (such as human models), which are manually adjusted to match similar objects in the camera image. The method uses a pinhole camera model, which requires only focal length, tilt angle and camera height to reconstruct the intrinsic and extrinsic matrices. [30][8]

The method doesn't take lens distortions into account, which can make measurement results inaccurate. Manually adjusting a 3D model to visually match the underlying camera image is rather slow. In a product demostration video, the calibration process takes 25 seconds and the camera height is seemingly known beforehand [31]. This is kind of calibration can be slow and cumbersome, as the camera height is not always known in advance. Furthermore, the method cannot be easily transformed to use automatic object detections.

One simple method to aid manual calibration is to do pixel horizon estimation. Hoiem [14] uses linear approximation of object detections to estimate the horizon position. The method relies on the assumtion that the camera tilt angle is close to zero (i.e. the camera is pointing at the horizon).

2.6.2 Calibration using known markers in the scene

Micusik et al. [22] presents an interesting method on human detections and camera calibration. It uses trained human detector to figure out all basic intrinsic and extrinsic camera parameters, and also radial lens distortion parameter. The advantage is that no specialized calibration tools are needed, but this comes at a computational cost, because detecting humans from the camera image is computationally intensive. The amount of detections needed is too high for manual setup.

Micusik et al. also present a method for evaluating and comparing different algorithms. They use synthetic test, where human detections for calibration are generated from a model and zero-mean Gaussian noise is added to the models. The test is repeated 100 times at each noise variance level [22], which provides a rough mean and variance for focal length estimates.

2.6.3 Using 3D scene reconstruction and feature matching

By creating a 3D scene using e.g. depth camera sensors, such as Kinect [16] a camera position relative to 3D scene can be inferred by finding matching features [27]. An example of 3D point cloud scene constructed using a iterative closest point algorithm from multiple depth camera images is shown in figure 5. These consumer-grade cameras either employ structured light techniques or time-of-flight techniques for sensing the depth information. Both of them usually use infrared light, and their maximum operating distance is limited and also they usually work only indoors due to high amounts of infrared in sunlight. [7, p. 11][6, p. 16]

3D scene reconstruction requires a lot of human labour and inexpensive depth cameras have limited depth sensing range. Inexpensive depth cameras use infrared laser projectors, which are not intensive enough to be used in bright outdoor scene. This makes capturing large scenes and areas (such as building outdoors) infeasible. However, in a limited space or in an indoor setting, this method can be useful.

If no physically referenced depth information is used, and 3D scene is reconstructed using image matching methods [27], a physical reference measure is needed to fix the model scale. Such reference can be obtained by using e.g. differential GPS, which provides spatial accuracy of about 10cm. State of the art methods can yield fairly geographically accurate scenes from large amount of images. [5] This method is also very laborous and requires heavy computation, so using it real-time is not feasible currently on general-purpose computers.

2.7 Calibration and object tracking with multiple overlapping cameras

In object tracking across multiple overlapping cameras, the objective is to find which object detections in separate cameras are the same object. This is achieved by comparing features between the objects. Often, features that derive the physical location of the object are used but also other matching methods can be used, if location information is not important.

Figuring out is the location of the object in the image and in the world eads to a two-fold problem: first, a calibration between all the cameras (extrinsic parameters) has to achieved in a common world coordinate system and second, object matching (camera handoff) has to be done, when a moving object crosses a camera boundary. If the extrinsic camera parameters are known relative to each other, we can utilize this knowledge to project the blobs into 2D or 3D space and see if they are close to each other or not. When the scene projections are inside a certain threshold, objects are regarded as being the same. This method is fairly simple and computationally inexpensive, but fails with occluding and crossing objects.

More finer feature extractors can be used, such as color, pattern and physical size features and trained match classificators. Methods based on such features can sometimes function even if the position of cameras relative to each other is not known or if the cameras don't overlap. However, the interest in non-overlapping cameras is outside the scope of this work. Extra features can also be combined with the location information. Extra features can be used to rule out occlusions and object crossings. Advanced feature extraction or classification methods are typically computationally expensive still today, and as such limited to few-camera systems in a realtime setting if at all.

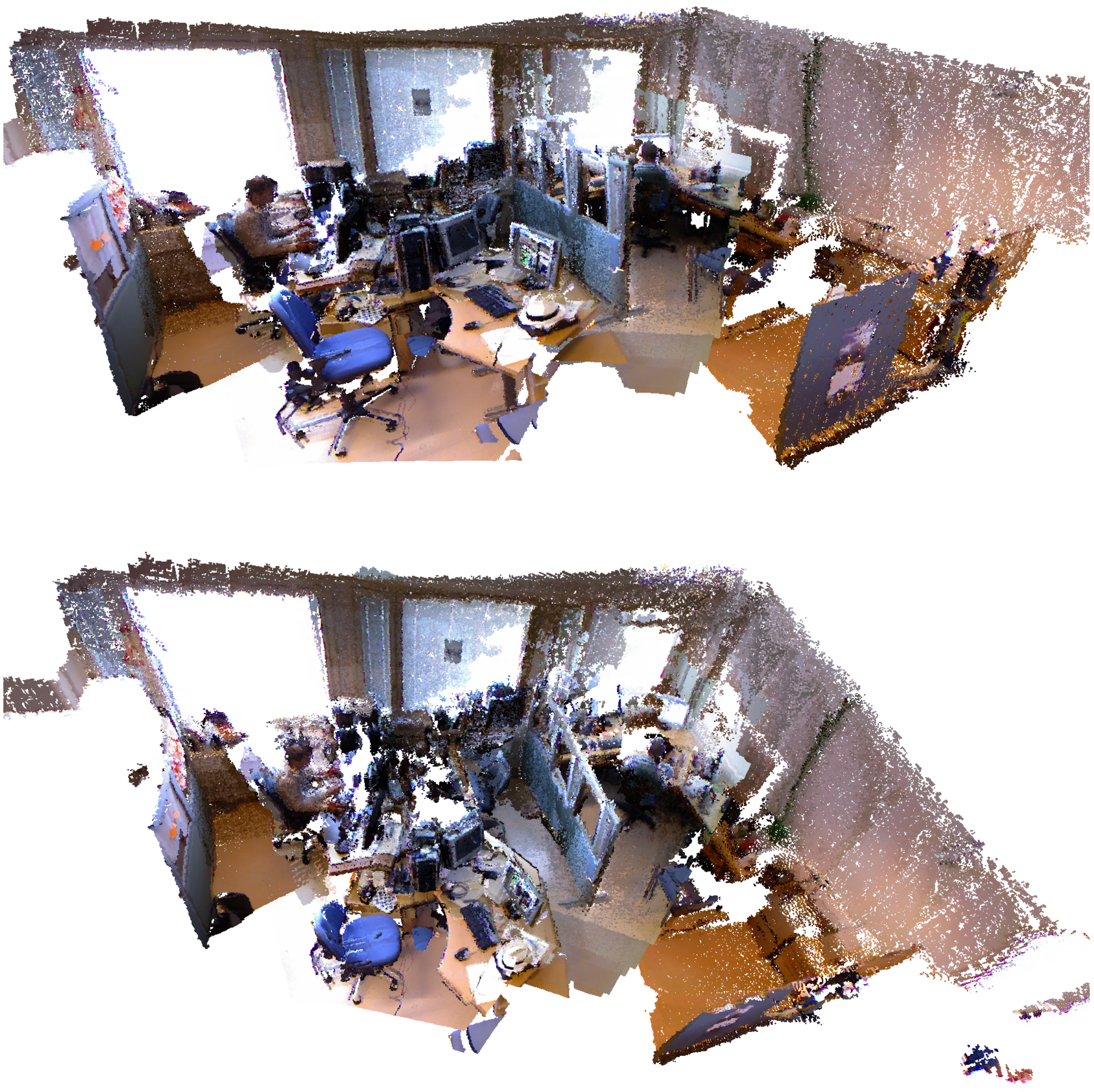


Figure 5: Example of a 3D point cloud created from an indoor scene using a depth and RGB camera. Iterative closest point algorithm is used to combine the multiple depth + RGB measurements into a compound 3D point cloud.

2.7.1 Top-down calibration and tracking

The top-down setting is a special case in overlapped multi-camera tracking. In a top-down setting, the cameras are installed in parallel pointing directly to the ground. Top-down arrangement makes calibration a bit easier. However, parallax makes this a bit challenging with wide angle lenses. When the cameras are mounted higher, there is less parallax.

Fukuda [9] proposes a system, where people are tracked using motion tracking and hair area and color detection. The system can analyse images in realtime, one camera per machine at 20-30Hz [9]. Fukuda does not provide any accuracy or error rate measures.

2.7.2 Freely positioned calibration and tracking

In the more general case of freely positioned cameras, some published algorithms rely on rather heavy-weight 3D reconstruction and structure from motion methods. This is unacceptable in a real-time system using current desktop computers with multiple cameras.

Liem and Gavrila propose a system that works with offline-calibrated cameras. In their work, they project moving background subtracted blobs onto a common ground surface. Their solution is mostly concentrated on a single scene, which all of the cameras see from various angles. They don't discuss about camera handoff. Also, they don't mention whether their tracking is real-time capable on commodity desktop computers. They also don't provide any details about the offline calibration and how much time does it take to do the calibration. [21]

Kang et al. present a system for continuous detection and tracking objects using multiple stationary cameras for a crowded scene. Their solution works without calibration. Soccer game, projected to top-down-like representation [17]. The claimed speed for the system is one frame per second, which is too slow for real-time surveillance and metadata generation purposes.

Zhao et al. proposes a realtime tracking system with multiple networked stereo cameras [36]. Their method is to do single stereo camera tracking first and then do fusion in a separate multi-camera tracker. Before tracking, lens distortions are corrected. Calibration is achieved using a parallax based approach [19], and cameras are mapped pair-wise into a common world coordinate system.

From the single stereo camera tracking a semi-3D object representation is created. The representation encodes object's ground position and uncertainty combined with 2D shape, like a cardboard placed vertically on a ground plane. Space-time cues are then used to match overlapping tracks to achieve camera handoff and tracking over multiple sensors. They provide two useful metrics, track fragmentation rate and track confusion rate, which are used to characterize the performance of the whole multi-camera tracking system [36]. The system is capable of real-time operation and does not require tedious manual calibration. As it relies on stereo cameras, the single camera tracker is does not fit our problem, but we could substitute that with our own.

Chen et al. present a multi-camera calibration method using an identifiable point target, such as a bright LED [4]. This can be used as an alternative to parallax based approach [19] for a number of single cameras.

Zhang et al. present a multi-camera tracking system for video surveillance purposes. In the system, cameras are calibrated manually against a global map plane using matching point and line features. According to Zhang et al, line features are more reliable and more representative than a single point, yielding a more precise calibration. The calibration for each camera is a homography matrix using direct linear transformation algorithm. Individual object detections are placed on the global map plane along with their physical size [35].

Khan and Shah propose a field-of-view (FOV) lines based approach [18] for calibrating cameras. They recover the field-of-view lines by observing motion and use this to obtain homography between the global map plane and different cameras.

3 Objective and challenges

Engineering still has a flavor of art.

— S. Theodoridis & K. Koutroumbas: *Pattern Recognition*

This section sets the objective of this thesis. The camera parameter calibration problem is defined.

3.1 The objective

The purpose of camera calibration is to provide real world coordinate data from tracked objects, so that the 3D position and speed of objects can be used for analysis. These tracked objects are inferred from the camera video stream using motion detection and object tracking algorithms, which provide camera image coordinates of the moving object. The task of the camera calibration is to provide additional parameters in order to convert these image coordinates into physical units in the real world.

The scope is limited to simple plane approximation, where 3D position can be correctly approximated for objects moving on the ground.

3.1.1 Usability requirements

The most important feature and requirement for a camera parameter calibration method is that the workflow must be easy to perform and efficient. Personnel who install video surveillance systems have to be able to use the system with as little extra education as possible, and they should be able to verify that the achieved calibration is good enough.

3.1.2 Speed requirements

An video surveillance object tracking system has a strict requirement for being able to work in real time. Usually one desktop class computer takes several camera inputs at the same time. The computer should be able to process the video and save the extracted metadata from all cameras.

Since most CCTV surveillance applications feature multiple cameras, the algorithm efficiency and lightweightness is an important factor, when real-time operation is desired. Many, if not most, of the published motion detection and object tracking algorithms and methods are too computationally intensive to be used in real-time operation on today's common desktop computers.

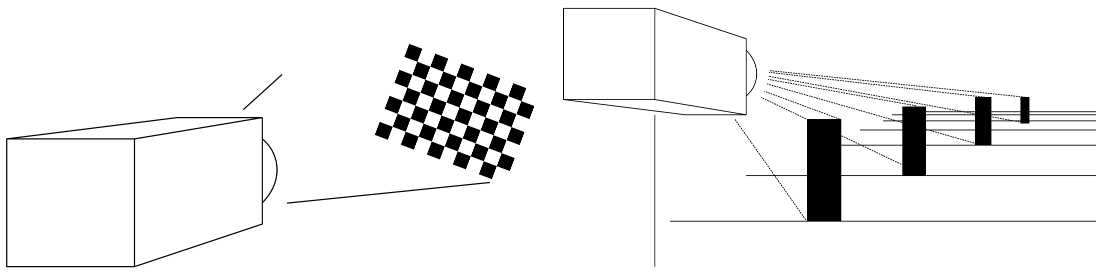
If camera calibration is to be done separately and not as a part of the system, algorithm efficiency is not that big problem. But when it is desirable to run camera calibration often, it is good to be capable of real-time or near real-time operation without using much resources. This comes especially important when implementing these methods using the low-power embedded processors in camera platform. There computation resources are more scarce than on general-purpose server or desktop computers.

3.1.3 Implementation requirements

Many of the methods proposed in the literature are highly sophisticated, but hard to implement (i.e. require complex calculation libraries not available on open-source or embedded platforms) and often too complex to maintain in cost-effective systems. To maintain easy implementability and maintainability, the objective is to come up with relatively simple and effective solutions, so that the implementation can be reasonably easily made on any computing platform and environment. Closed-source libraries are best avoided [24].

3.2 Workflow overview

The proposed camera calibration workflow consists of two stages. In the first stage, intrinsic camera parameters and lens distortion removal parameters are calibrated. In the second stage, extrinsic camera parameters are calibrated.



(a) First stage: intrinsics calibration. **(b)** Second stage: extrinsics calibration.

Figure 6: Camera calibration stages in the proposed solution.

3.2.1 First stage: Intrinsic camera parameters and lens distortion removal calibration

The goal of the camera intrinsic parameter calibration is to make the camera-lens system perform like an ideal pinhole camera with a known focal length.

The selected lens distortion model uses three radial distortion coefficients and two tangential distortion coefficients, following the recommendations by Bradski et al. [1, p. 375] and [12]. These coefficients are calibrated using the checkerboard calibration method presented by Bradski et al. [1, p. 375]. This method also yields the focal length of the camera-lens-system.

This can be done off-site or on-site using the checkerboard calibration described in the section 2.5.2. If a specific camera-lens system model has consistent high quality, it might suffice to do the intrinsic parameter and lens distortion removal calibration only once for that model.

3.2.2 Second stage: Extrinsic camera parameters calibration

The goal of camera extrinsic parameter calibration is to determine the tilt angle and camera height relative to a ground plane. Since the focal length is known, this is enough to solve the camera's projection matrix and convert object image coordinates into world coordinates for objects on the ground plane. The camera is assumed to have a zero roll angle relative to ground plane and a sufficiently small tilt angle (less than 45 degrees).

4 Pinhole camera model

Essentially, all models are wrong, but some are useful.

— George E. P. Box: *Empirical Model-Building and Response Surfaces*

The objective of the camera calibration algorithm is to define camera position in 3D world. As explained in section 2, the lens distortion correction method resolves the focal length of a camera and after distortion removal, the resulting image is almost the same as if it had been taken by an ideal pinhole camera. Thus, it is useful to utilize a well-known pinhole camera model, which this section defines in the context of this thesis. From the pinhole camera model, forward and inverse projections are derived, including their error estimates using total differentials.

4.1 Definitions

To start off, we have to select a coordinate system that we use to represent the physical world. The world coordinate system x, y, z is left-handed, as presented in the figure 8. This choice for the coordinate system follows from the ray tracing engine POV-Ray, which is a simulator later in this study. Choosing the same coordinate system makes testing and verification against the simulated situations easier.

A camera is a device that projects 3D scene on to a 2D surface called image plane. The camera image plane is shown in figure 7. The image plane coordinate system is u, v and its origin is at the center of the image. These coordinates are measured as pixels.

We use remove lens distortions with software, so most camera-lens-systems perform almost as an ideal pinhole camera. This makes further analysis more simple and elegant and also more accurate. Let us define the coordinate systems, where we do the calculations. First off, we have a simple pinhole camera model, which is shown at different tilt angles in figures 9 and 10.

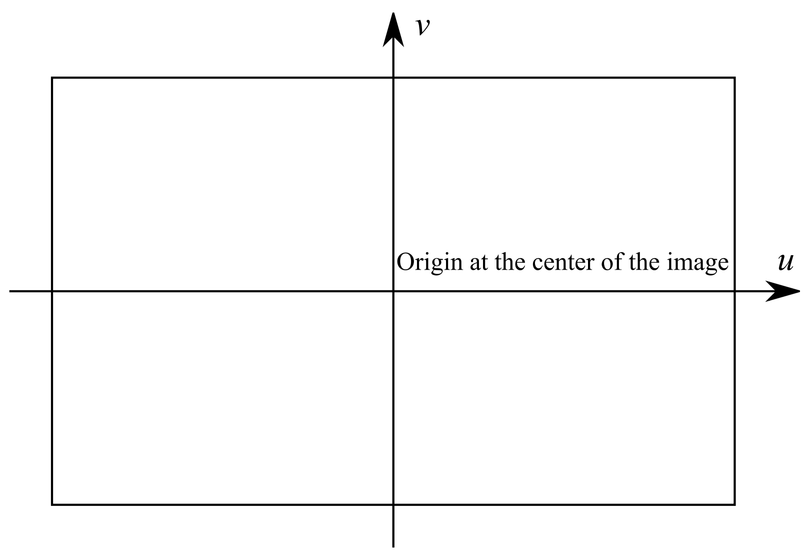


Figure 7: Image plane (u, v), where u, v are measured in pixels. Thus the visible area is $(-res_x/2, res_x/2)$ for u and $(-res_y/2, res_y/2)$ for v component.

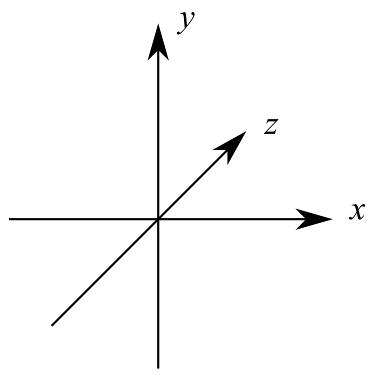


Figure 8: Left-handed world coordinate system (x, y, z). Units are in meters.

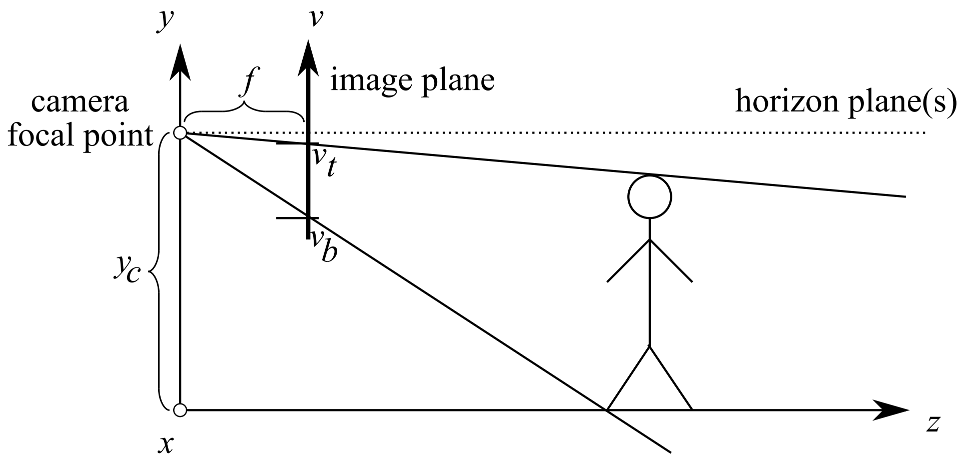


Figure 9: Pinhole camera that points to horizon. The focal length f and vertical image coordinate v are measured in pixels.

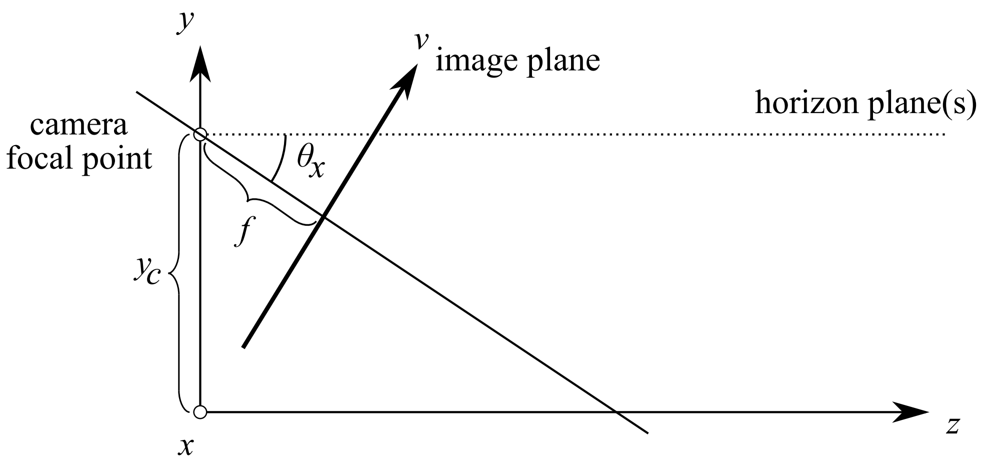


Figure 10: Pinhole camera that points below horizon in an angle of θ_x.

It might seem counter-intuitive, but the camera height y_c doesn't affect the position of pixel horizon on the image plane. This is due to the fact the ground in this model is a *plane*, not a geoid. By definition, the horizon is at the infinity, where planes parallel to the ground plane meet. Thus, we can draw one of those parallel planes to be at the height of y_c and get the pixel horizon position

The focal length f and the vertical field of view angle Φ are connected [29, p. 53]

$$\Phi = 2\arctan\left(\frac{d}{2f}\right), \tag{4}$$

or

$$f = \frac{d}{2\tan\left(\frac{\Phi}{2}\right)}, \tag{5}$$

where d is half of the vertical image resolution (focal length measured in pixels) or half of the vertical image sensor size (focal length measured in physical units).

In matrix form, the pinhole camera projection is as follows. Since (x, y, z) is a left-handed coordinate system, we add minus to v and y coordinate in the equation. Let

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \tag{7}$$

$$\mathbf{U} = \begin{bmatrix} uw \\ -vw \\ w \end{bmatrix} \tag{8}$$

$$\mathbf{X} = \begin{bmatrix} x \\ y_c - y \\ z \end{bmatrix}, \tag{9}$$

where **K** is the camera projection matrix, **R** is the rotation matrix, **U** is the projected point on image plane in homogeneous coordinates [25], and **X** are the world coordinates of a point to be projected. A similar formulation is used in [13].

Bradski et al. uses a camera projection matrix in the form [1, p. 374]

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

In this work, this matrix is simplified by two assumptions. First, this work assumes fixed pixel aspect ratio of $1 : 1$ (square pixels), i.e. $f_x = f_y = f$. It should be noted that this has nothing to do with the image resolution or the image aspect ratio. Furthermore, this work assumes that the optical center is at the image center, i.e. $c_x = c_y = 0$. Both of these assumptions can be fulfilled with good enough accuracy for most cameras using the distortion removal obtained from checkerboard lens calibration [1, pp. 370–404].

4.2 Calculation of the forward projection

The forward projection transforms real world coordinates into image coordinates. This forward projection is used to develop a solution for the calibration algorithm.

The forward projection can be written as $\mathbf{U} = \frac{1}{z}\mathbf{KRX}$, but we define a helper variable $W = \frac{1}{z}$ to obtain a linear equation system. W is 0 when z is at $\pm\infty$. Thus, the forward projection in matrix form is

$$\mathbf{U} = W\mathbf{KRX}, \tag{11}$$

from which we obtain

$$\begin{pmatrix} uw \\ -vw \\ w \end{pmatrix} = \begin{pmatrix} fWx \\ W(f(y_c - y)\cos\theta_x - fz\sin\theta_x) \\ W(z\cos\theta_x + (y_c - y)\sin\theta_x) \end{pmatrix}. \tag{12}$$

4.3 Calculation of the inverse projections with known y or z

The inverse projection transforms image coordinates into real world coordinates. This inverse projection is used to obtain object location and speed estimates, after the camera calibration is done.

The idea behind the use of these inverse projections is to calculate first the object distance z from the camera with known height $y = 0$ (bottom image coordinate). Then the distance is known, and the object height y can be calculated using the now known distance z value.

The inverse projection can be written in the matrix form

$$W\mathbf{X} = \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{U}, \tag{13}$$

which expands to

$$Wx = \frac{uw}{f} \tag{14}$$

$$W(\text{yc} - y) = w\left(\sin(\theta_x) - \frac{v\cos(\theta_x)}{f}\right) \tag{15}$$

$$Wz = w\left(\cos(\theta_x) + \frac{v\sin(\theta_x)}{f}\right) \tag{16}$$

From this, if variables **K**, **R**, u, v, w, and y are known, W can be solved from equation (15):

$$W = \frac{w\left(\sin(\theta_x) - \frac{v\cos(\theta_x)}{f}\right)}{y_c - y} \tag{17}$$

If $W \neq 0$, ie. the point is not at infinity, then from equations (14) and (16), one can solve x and z:

$$x = \frac{uw}{fW} \tag{18}$$

$$z = \frac{w\left(\frac{v\sin(\theta_x)}{f} + \cos(\theta_x)\right)}{W} \tag{19}$$

An error estimate for the x and z coordinates can be obtained using a total differential

$$\Delta x = \left|\frac{\partial x}{\partial \theta_x}\Delta\theta_x\right| + \left|\frac{\partial x}{\partial y_c}\Delta y_c\right| + \left|\frac{\partial x}{\partial f}\Delta f\right| + \left|\frac{\partial x}{\partial u}\Delta u\right| \tag{20}$$

$$\Delta z = \left|\frac{\partial z}{\partial \theta_x}\Delta\theta_x\right| + \left|\frac{\partial z}{\partial y_c}\Delta y_c\right| + \left|\frac{\partial z}{\partial f}\Delta f\right| + \left|\frac{\partial z}{\partial v}\Delta v\right|. \tag{21}$$

This assumes that all variables are independent and component errors are small enough to induce an approximately linear change error.

Similarly, if variables **K**, **R**, u, v, w, and z are known, W can be solved from equation (16):

$$W = \frac{w\left(\frac{v\sin(\theta_x)}{f} + \cos(\theta_x)\right)}{z} \tag{22}$$

If $W \neq 0$, ie. the point is not at infinity, then from equations (14) and (15), one can solve x and y:

$$x = \frac{uw}{fW} \tag{23}$$

$$y = y_c - \frac{w\left(\sin(\theta_x) - \frac{v\cos(\theta_x)}{f}\right)}{W} \tag{24}$$

Again, an error estimate based on total differential can be obtained using raw derivation:

$$\Delta x = \left|\frac{\partial x}{\partial \theta_x}\Delta\theta_x\right| + \left|\frac{\partial x}{\partial y_c}\Delta y_c\right| + \left|\frac{\partial x}{\partial f}\Delta f\right| + \left|\frac{\partial x}{\partial u}\Delta u\right| \tag{25}$$

$$\Delta y = \left|\frac{\partial y}{\partial \theta_x}\Delta\theta_x\right| + \left|\frac{\partial y}{\partial y_c}\Delta y_c\right| + \left|\frac{\partial y}{\partial f}\Delta f\right| + \left|\frac{\partial y}{\partial v}\Delta v\right| \tag{26}$$

5 Calibration of a pinhole camera in 2D world

More tasty mathematics here.

— Neal Stephenson: *Cryptonomicon*

This section presents a method to calibrate a pinhole camera relative to a ground plane. First, a pixel horizon for the camera is determined, and that is used to calculate (up to a scale factor) the distance of the object relative to camera. Then, through careful formulation, algorithms to determine camera height and tilt angle are presented.

5.1 Problem definition

Our camera calibration problem is to find the camera height y_c, focal length f and tilt angle θ_x in the equation (12). We have an upright object of a known physical height, which moves or can be moved towards or away from the camera. The object is detected from the images either manually or automatically. From the detections, we can deduct the bottom and top image coordinates (v_b, v_t) of the object. These coordinates are in pixels.

The camera calibration problem can be made easier with some reasonable assumptions. If the camera horizon is parallel to the plane horizon (i.e. the camera roll angle θ_z is zero) and the pixel aspect ratio is $1 : 1$ (i.e. $f_x = f_y = f$), the problem reduces into a two-dimensional problem. Both the roll angle and aspect ratio can be made to fulfill these assumptions for any camera. The aspect ratio can be corrected at the same time with lens distortion corrections. The roll angle can be corrected by simply rotating the 2D camera image. Leaving out the x axis means also that the camera yaw angle does not affect the problem, so it can be safely discarded from the consideration.

The calibration method should work with manual and automatic object detections. In manual detection, it is desired to work with a small number of detections, such that the amount of manual labor is minimized. This means that the algorithm should need only a few "clicks" in the user interface. Alternately, automatic methods, such as automatic people detection, as in [22], produce a lot of detections, to overcome measurement noise in the detections.

5.2 Derivation of 2D object projection equations

When looking at the pinhole camera model from the side and removing the horizontal dimension (x in world coordinates, u in image coordinates), the equation (12) simplifies to

$$\begin{aligned}\begin{bmatrix} wv \\ w \end{bmatrix} = \mathbf{KR} \begin{bmatrix} y - y_c \\ \alpha \hat{z} \end{bmatrix} = \begin{bmatrix} f & \\ & 1 \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} y - y_c \\ z \end{bmatrix} \\ = \begin{bmatrix} fc & fs \\ -s & c \end{bmatrix} \begin{bmatrix} y - y_c \\ z \end{bmatrix} = \begin{bmatrix} fc(y - y_c) + fsz \\ -s(y - y_c) + cz \end{bmatrix},\end{aligned} \tag{27}$$

where $s = \sin\theta_x$ and $c = \cos\theta_x$.

Again, the projection result is in homogeneous coordinates. The actual vertical image coordinate v, is obtained by normalizing the homogeneous coordinate, i.e. by dividing wv by w.

$$\frac{wv}{w} = f\frac{cy - cy_c + sz}{-sy + sy_c + cz}. \tag{28}$$

Multiplyng this with the left-over denominator yields

$$(-sy + sy_c + cz)v = f(cy - cy_c + sz), \tag{29}$$

which is essentially a linear equation. By plugging in object top and bottom image coordinates v_t and v_b ($y = 0$), this produces handy linear equations for an upright object:

$$-syv_t + sy_cv_t + czv_t - fcy + fcy_c - fsz = 0 \tag{30}$$

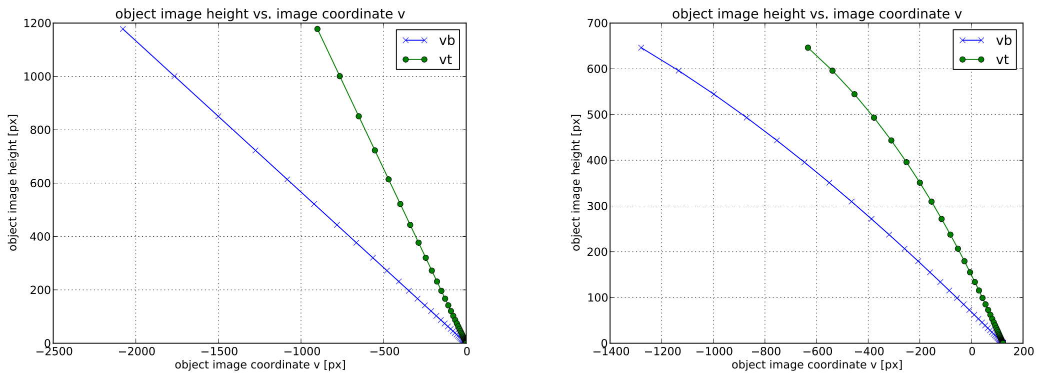
$$sy_cv_b + czv_b \qquad + fcy_c - fsz = 0. \tag{31}$$

Now these equations are in a useful form, as we'll see later.

5.3 Determining the pixel horizon

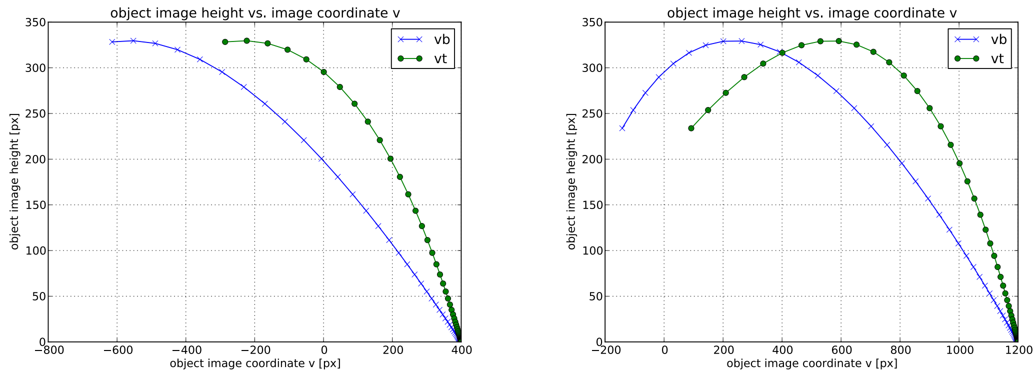
Many texts refer to vanishing point or the pixel horizon. This is the point or level at which any object moving on a plane will disappear or vanish, when it is moving away from the camera. If we have multiple detections of the same object or different objects of known height at different distances from the camera, we can use coordinates from detections to estimate the pixel horizon.

Let us consider an upright object that is moving towards (or away) from the camera, such as in the figure 9 (page 19). Let us choose a camera height y_c be 3 m and vary the tilt angle. For each tilt angle, we plot the bottom and top image coordinates of the object, using the equation (27).



(a) At a tilt angle $\theta_x = 0$, the correspondence is linear.

(b) At a tilt angle $\theta_x = 10°$, the correspondence approximately linear.



(c) At a tilt angle $\theta_x = 30°$), the correspondence starts to skew.

(d) At a tilt angle $\theta_x = 60°$, the correspondence is very non-linear.

Figure 11: Object top and bottom image coordinates vs. object image height. Both coordinates converge at the pixel horizon, where object height is zero.

We gain plots like in the figure 11. It should be noted that not all of the coordinates are actually visible; the camera sees only a limited range of coordinates. For example, an ideal pinhole camera with a vertical resolution of 480 pixels the visible range would be $v \in [-240, 240]$. The pixel horizon may exist outside the area captured by the camera.

From these plots in the figure 11., we can deduct that linear pixel horizon estimation is not adequate for high tilt angles.

The pixel horizon can be calculated from object heights in pixels $h_v = v_t - v_b$ and object bottom coordinates v_b.

The first method is to use a first order linear fit and the more theoretically accurate one is to use "skewed" parabolic fit. The first order fit is good enough for small tilt angles and it's more robust to errors.

The equation for fitting is

$$h_v = av_b + b, \tag{32}$$

and we figure out parameters a and b with a least squares fit. From these we can calculate the horizon position (the point where $h_v = 0$) in pixels

$$v_h = -\frac{b}{a}. \tag{33}$$

For higher tilt angles, a second order parabolic fit provides more accurate estimates. The equation for fitting is

$$h_v = av_b^2 + bv_b + c, \tag{34}$$

If a is close enough to zero (tilt angle is close to zero), we can determine the root with using the previous equation. However, at higher tilt angles, a is negative, so we take the negative branch of the solution

$$v_h = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{35}$$

5.4 First algorithm to solve unknown focal length, camera height and tilt angle

First, we can try to solve all three parameters in our problem.

If object image coordinates v_t, v_b, object height y and distance z are known for a sufficient number of point, a linear algebra problem of the form $\mathbf{Ax} = \mathbf{0}$ can be constructed.

The solution requires calculating the object relative distance $\hat{z}$, which is the real distance of an object along z such that

$$z = \alpha\hat{z}, \tag{36}$$

where α is a scale factor.

After we have determined the pixel horizon v_h, it is possible to approximate the relative object distance $\hat{z}$ such that

$$\frac{1}{\hat{z}} = v_b - v_h. \tag{37}$$

This is a linear simplification, so this does not provide good estimates with high tilt angles.

The first straightforward formulation obtained from the equations (30) and (31) is as follows. Object top and bottom image coordinates v_t and v_b and object height y_o is known, which we can put into equations (30) and (31) to obtain a homogeneous linear equation system $\mathbf{Ax} = \mathbf{0}$

$$\begin{bmatrix} -yv_t & v_t & \hat{z}v_t & -y_o & 1 & -\hat{z} \\ 0 & v_b & \hat{z}v_b & 0 & 1 & -\hat{z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \beta \begin{bmatrix} s \\ sy_c \\ c\alpha \\ fc \\ fcy_c \\ fs\alpha \end{bmatrix} = \mathbf{0}. \tag{38}$$

As the number of unknowns is six, a unique solution requires at least three object detections (three top and three bottom coordinates). Each object detection adds two equations, i.e. two rows in to the matrix. If the system is overdetermined (more than three detections), the equation can be solved (up to a scale factor β) using singular value decomposition (SVD). See [10, pp. 88–91].

$$\mathbf{A} = \mathbf{UDV}^{\mathrm{T}}, \tag{39}$$

such that the estimate for $\mathbf{x}$ is the last column of $\mathbf{V}$.

From the estimate, we can determine the value of β using the identity $\sin^2 x + \cos^2 x = 1$:

$$\beta^2 = \frac{(s)}{(fs\alpha)}((fs\alpha)(s) + (fc)(c\alpha)) \tag{40}$$

Rest of the parameters can be solved as follows

$$\theta_x = \arcsin\left(\frac{(s)}{\beta}\right) \tag{41}$$

$$\alpha = \frac{(c\alpha)}{\beta cos(\theta_x)} \tag{42}$$

$$f = \frac{(fc)}{cos(\theta_x)\beta} \tag{43}$$

$$y_c = \frac{(fcy_c)cos(\theta_x)}{\beta f} \tag{44}$$

5.5 Second algorithm to solving unknown camera height and tilt angle

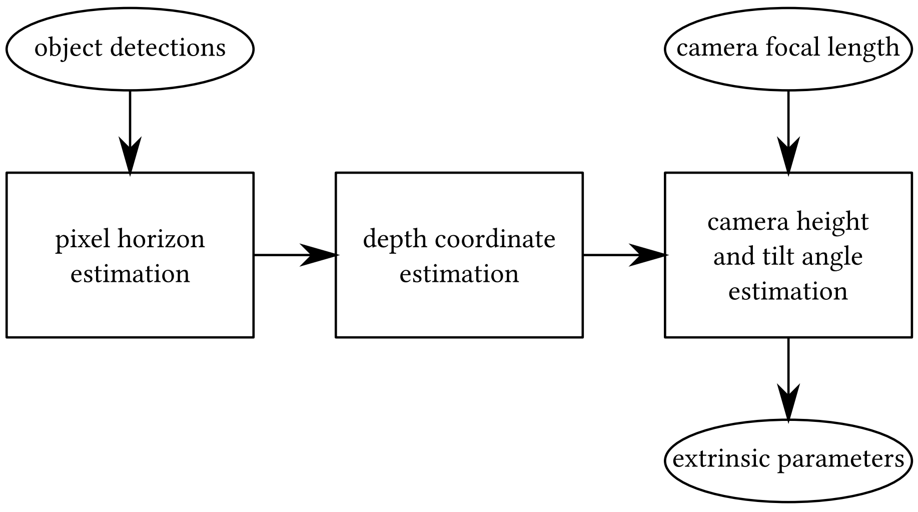


Figure 12: Block level representation of the second calibration algorithm.

The first attempt proved to be very sensitive to noise in parameters and it did not converge to a sensible result in many cases. Thus the idea had to be developed a little bit further.

If the camera focal length f is known, a linear algebra problem of the form $\mathbf{Ax} = \mathbf{b}$ can be constructed. These standard problems can be solved using, respectively, e.g. singular value decomposition (SVD) or QR decomposition. These decompositions are implemented widely in various numerical linear algebra software libraries.

The second formulation assumes that the camera focal length f is known. This requirement can be fulfilled by using the lens distortion calibration (see chapter 2.5.2, page 8), which provides this information as a useful by-product. Given the focal length f, the tilt angle θ_x can be directly calculated from pixel horizon v_h

$$\theta_x = \arctan\left(\frac{v_h}{f}\right). \tag{45}$$

Object top v_t and bottom v_b image coordinates, object height y_o and focal length f are known. Since the variables $c = \cos\theta_x$ and $s = \sin\theta_x$, the only remaining unknowns in equations (30) and (31) are y_c and α. Using this information, we get a linear equation system $\mathbf{Ax} = \mathbf{b}$

$$\begin{bmatrix} sv_t + cf & c\hat{z}v_t - sf\hat{z} \\ sv_b + cf & c\hat{z}v_b - sf\hat{z} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} y_c \\ \alpha \end{bmatrix} = \begin{bmatrix} sy_ov_t + cfy_o \\ 0 \\ \vdots \end{bmatrix}. \tag{46}$$

In this equation, the rows in $\mathbf{A}$ and $\mathbf{b}$ are repeated as many times as there are object detections. Each object detection adds two equations, i.e. two rows to the matrix $\mathbf{A}$ and to the vector $\mathbf{b}$. There are only two unknowns, therefore one object detection should suffice to solve the matrix. However, the pixel horizon and $\hat{z}$ cannot be estimated with just one detection, the actual minimum is two object detections.

The two object detections make this linear system always overdetermined. Since it is overdetermined, there is usually no exact solution due to noise in measured parameters. An approximate solution can be found using a regression algorithm, e.g. by minimizing either the square or absolute error. These common methods are called linear least squares (L_2 norm) regression or linear least absolute deviations (L_1 norm) regression.

5.5.1 Solution with Least Squares

The overdetermined linear system in the equation (46) can be solved using ordinary least squares fit

$$\min ||\mathbf{Ax} = \mathbf{b}||_2 \tag{47}$$

The least squares solution for $\mathbf{x}$ can be calculated easily using the standard QR matrix decomposition [28, p. 293]

$$\mathbf{A} = \mathbf{Q}\begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}, \tag{48}$$

such that the estimate for $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \mathbf{R}^{-1}\mathbf{Q}^{\mathrm{T}}\mathbf{b}. \tag{49}$$

As we can see from the equation (46), the estimate for the camera height y_c is the first element of $\hat{\mathbf{x}}$.

QR decomposition or any LLS fit is available in most numerical linear algebra packages, which makes the implementation of this solution easy.

5.5.2 An error estimate for LLS solution

The in-sample standard error estimate $\mathbf{E_x}$ for the fitted $\hat{\mathbf{x}}$ can be obtained by calculating the estimate for the variance of the error term [11, p. 19, 29]

$$s^2 = \frac{||\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}||_2^2}{m - n} \tag{50}$$

$$\mathbf{S} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1} s^2 \tag{51}$$

$$\mathbf{E_x} = \sqrt{\mathrm{diag}(\mathbf{S})}, \tag{52}$$

where m is the number of rows and n number of columns in the matrix $\mathbf{A}$ and $\mathrm{diag}(\mathbf{S})$ is a vector containing the diagonal elements of the covariance matrix $\mathbf{S}$.

This in-sample error estimate for $\hat{\mathbf{x}}$ is only good, when there are many object detections, that is when the sample is sufficiently representative. When manual object detections are used, this error measure might not be useful. It is potentially useful for automatic object detection methods, when trying to get rid of outliers using e.g. RANSAC.

5.6 Third algorithm to solve unknown camera height and tilt angle

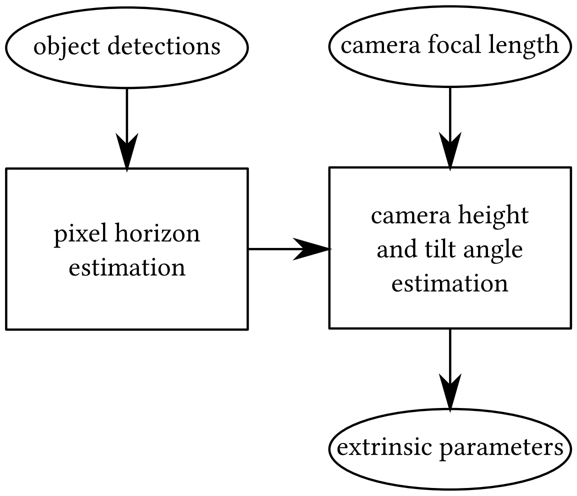


Figure 13: Block level representation of the third calibration algorithm.

After simulation, the second algorithm seemed to work robustly for cameras pointing directly at horizon (zero tilt angle). However, in the simulation, that solution produced inaccurate estimates for situations, where the camera is installed at more than 5 meters high and the camera has some tilt angle (that is, the camera is not pointing directly at the virtual horizon).

By rethinking the approach and looking at the equations again, it becomes evident that solving object distances before solving the linear algebra problem is redundant. Since the tilt angle can be deducted from the equation (45), the object distances z can be just simply substituted away.

$$\begin{aligned} -syv_t + sy_cv_t + czv_t - fcy + fcy_c - fsz &= 0 \qquad (53) \\ sy_cv_b + czv_b \qquad\quad + fcy_c - fsz &= 0 \qquad (54) \end{aligned}$$

Solving z from the second equation

$$z = \frac{-sy_cv_b - fcy_c}{cv_b - fs} \qquad (55)$$

and substituting z in the first equation (53) yields

$$y_c = \frac{syv_t + fcy}{sv_t + fc + (cv_t - fs)\left(\frac{-sv_b - fs}{cv_b - fs}\right)}. \qquad (56)$$

Thus, the camera height y_c can be solved for each object detection. If there are multiple detections, the resulting camera height estimate can be obtained taking an arithmetic mean of all individual solved camera heights.

6 Object Tracking with Overlapping Top-Down Cameras

Every step you take / I'll be watching you.
— The Police: *Every Breath You Take*

The idea is rather simple: calibrate the cameras so that we can produce a stiched overall image. Then combine the paths from different cameras that are near each other in the space, which dimensions are world coordinates and time.

6.1 Calibration

After the lens distortions have been removed, relative positioning of the cameras can be determined such that the camera images can be stitched together. In an indoor scene, the floor plane can be used as the reference for the stitching. If the cameras are not pointing top-down, projective transform transform is needed to create proper rectification [10, p. 55].

A crude stitching can be made using a general affine transform [10, p. 39] of the form [26, p. 371]

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{57}$$

This affine transform from one image's coordinates u, v to another x, y, which stitches these images on top of each other, can be found using either a least-squares algorithm [26, p. 371] or direct linear transform (DLT) algorithm [10, p. 130]. The stitching needs at least 3 matched point pairs.

However, if the cameras are installed to point straightly down, it suffices to use a limited form of the affine transform, namely a similarity transform [10, p. 39]. Experimenting with affine transform, errors in matched points would produce fairly skewed transforms, which are mostly incorrect. Similarity transform only consists of scale s, rotation θ and translation t_x, t_y parameters, and does not allow skewing.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{58}$$

To determine a similarity transform to align two images, two matched point pairs are required. The similarity transform is better, since the skewing in CCD cameras is usually nominal [12], especially after lens calibration and distortion removal. In order to solve the required transform, we can use the algorithm described in [26, p. 365].

Let $x^i_{1,2}, y^i_{1,2}$ be the points on first image and $x^w_{1,2}, y^w_{1,2}$ be the respective matched points on the second image. First we determine the rotation angle θ

$$\theta_i = \arctan\left(\frac{y^i_2 - y^i_1}{x^i_2 - x^i_1}\right) \tag{59}$$

$$\theta_w = \arctan\left(\frac{y^w_2 - y^w_1}{x^w_2 - x^w_1}\right) \tag{60}$$

$$\theta = \theta_w - \theta_i. \tag{61}$$

Then using this angle, the final transform can be determined

$$c = \cos\theta \tag{62}$$

$$s = \sin\theta \tag{63}$$

$$S = \frac{x^w_1 - x^2_2}{x^i_1 c - y^i_1 s - x^i_2 c + y^i_2 s} \tag{64}$$

$$t_x = y^w_1 - S(x^i_1 s + y^i_1 c) \tag{65}$$

$$t_y = x^w_1 + S(-x^i_1 c + y^i_1 s) \tag{66}$$

in matrix form

$$\mathbf{T}_{sim} = \begin{bmatrix} Sc\cos\theta & -Ss\sin\theta & t_x \\ Ss\sin\theta & Sc\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{67}$$

After matching, we can calculate pair-wise transforms between the images, as shown in the figure 14. After an anchor image is selected, all the transforms can calculated respective to it. To create a composite image, a bounding box for the whole result image must be calculated, and this is done by going through all images and transforms and figuring out their respective bounding boxes. The resulting bounding box is derived from these by getting the minimum and maximum of the all sub-image bounding boxes.

We can figure out the correct order for combining the transforms by constructing a graph. In the graph, all nodes are images and edges are pairings between them. At first, all the pairings are added as bi-directional edges. Then we select the anchor image and initiate a breadth-first search (BFS) and reduce the graph into a directed acyclic graph.

Now, the directed acyclic graph can be traversed recursively in BFS manner, and the transforms respective to the anchor image can be calculated by multiplying them together.

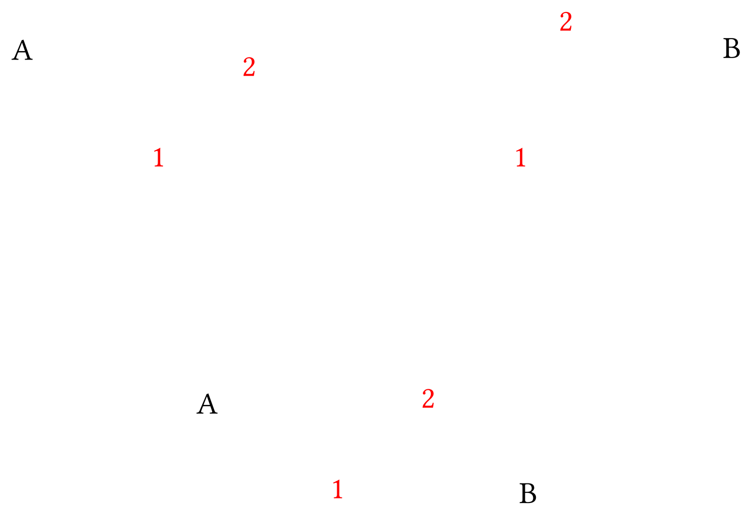


Figure 14: Matching point pairs (1, 2) and stitching images (A, B) together using a similarity transform. The bounding box of the result image is shown with a dotted line.

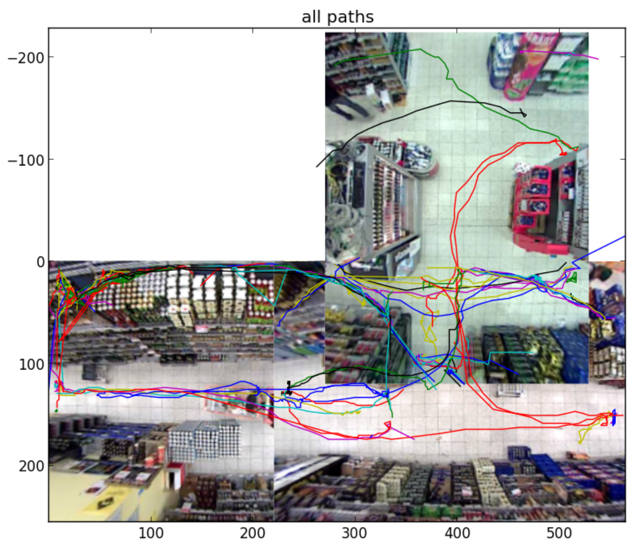


Figure 15: Image of a real multi-camera test setup. Camera images have had their distortion removed and then they are stitched together using similiarity transforms. Single-camera object tracking results (paths) are shown as colored lines on top of the images. (Few stray lines are result of the Kalman-based path predicition of the single-camera tracking algorithm.)

6.2 Simple algorithm for finding the combinable paths

Each camera has tracking algorithm, which provides us tracked object paths. The paths in figure 15 have time information in them and thus they can be represented better in a 3D space (figure 16).

A straigthforward method for joining these paths is to form an ellipsoid around a point on one path on one camera and compare it to a point on a path on the other camera. The size of the ellipsoid acts as a threshold: if the other point is inside the ellipsoid around the first one, the paths are marked as being the same path and same object. Using this method, the joinable paths can be seen in figure 17. The threshold value has to be adjusted depending on the scene.

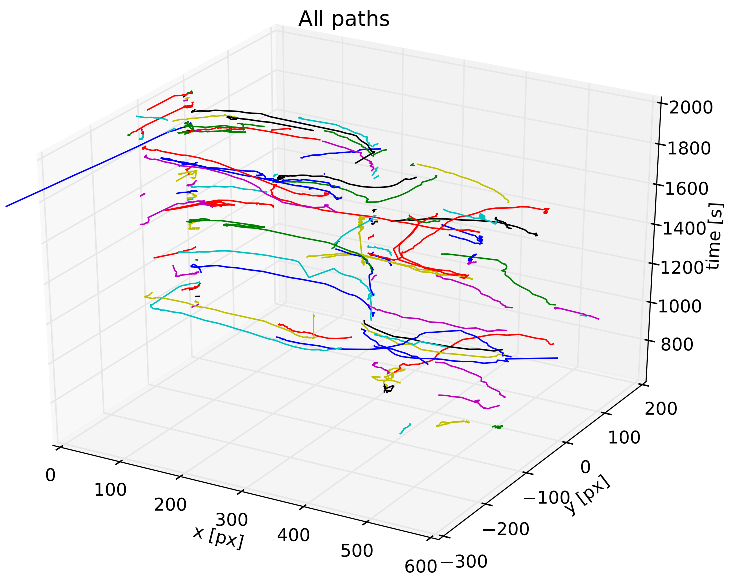


Figure 16: Visualization of the paths in time and space.

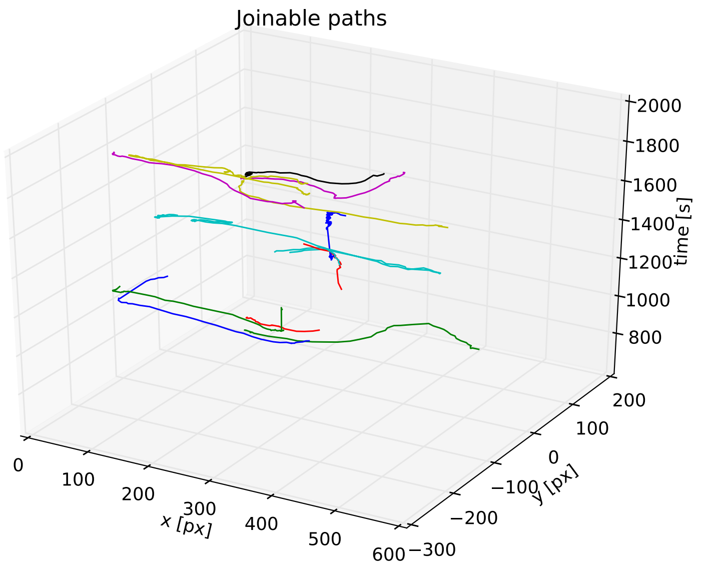


Figure 17: Paths that are deemed joinable by the algorithm.

7 Results and Comparision

The first principle is that you must not fool yourself — and you are the easiest person to fool.

— Richard P. Feynman: *Surely You're Joking, Mr. Feynman! / Cargo Cult Science*

How do we measure, whether our methods are any good? This section describes testing setups and defines performance measures for the camera calibration algorithm. The section also draws comparions to other camera calibration methods.

7.1 Camera Calibration

For the algorithm development purposes, a numerical simulation of object detections was built using the direct projection of the pinhole camera model (see section 4). This allows quick, automated testing of various camera angles and heights, and different scenes. The object detections had optional noise added to the image coordinates. This simulates imperfect object detection due to lack of image resolution for objects at a greater distance and inexactness of operator input.

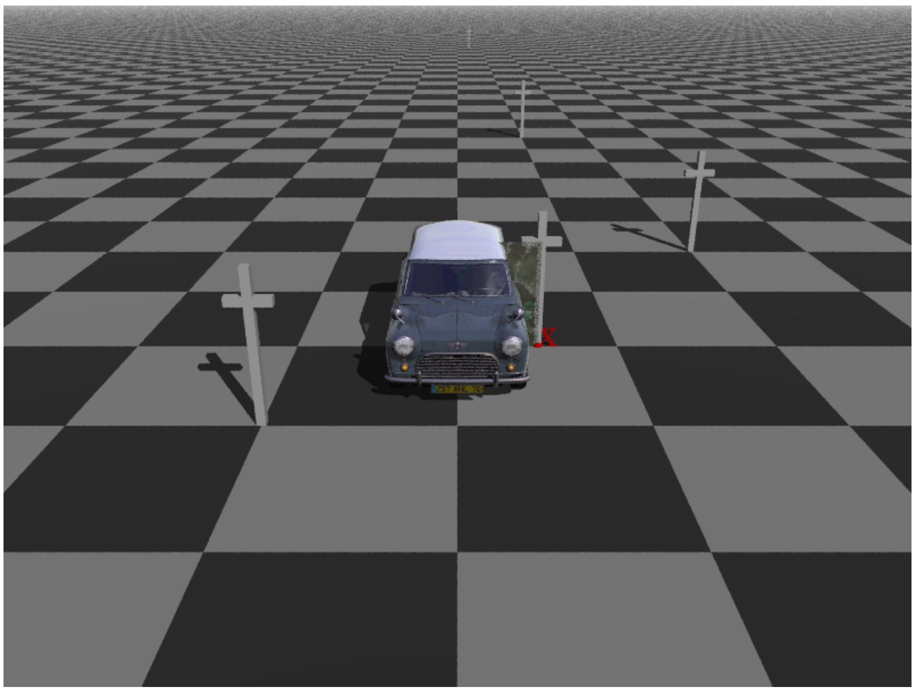


Figure 18: Example of a rendered test scene for manual object detection testing. The crosses represent objects of known height, such as humans.

For manual detection testing and user interface development purposes, a rendered simulation was built. Actual video material with measured ground truth values is hard to come by and slow to produce, so it was much easier to use 3D modeling. The rendered simulation uses a 3D ray tracing engine POV-Ray [3] for producing photorealistic test scenes, such as in figure 18. POV-Ray can model a lot of various realistic phenomena, such as shadows, focus blur and lens distortions. However, in the testing these were not used.

Finally, a proof of concept testbed was implemented in Python and C++. This extended a system made by Kuusisto [20], which had motion detection and object tracking algorithms already in place. The camera calibration in the testbed had of a lens distortion removal step and a manual object detection step. After performing the camera calibration, the test bed annotated moving objects with physical dimension and speed estimates.

The performance of the 3D camera scene calibration method is hard to objectively assess. The method depends on manual input. Thus, the performance of the method depends both on the accuracy of the manual input (i.e. how careful the operator is) and the algorithm itself.

For lenses with narrow angle of view (less than 60 degrees), the lens distortion calibration accuracy for the focal length is good enough for most video surveillance purposes. However, with wide-angle or fish-eye lenses, focal length calibration using the seems to be rather next to useless. The error might stem from warped calibration paper and because of the extreme distortions at lens corners.

7.2 Performance metrics

For the operator, one quantitative performance metric is, how quickly a satisfactory calibration can be achieved. Qualitative performance metrics are how easy the method is to use and how much training is needed for an operator to learn to use the system.

For the algorithm, typical accuracy of the input can be estimated, and this can be taken into account in simulating the operator input. Sensible quantitative performance metrics can be: how much input data is required, what is the error in the results (estimated parameters) relative to ground truth (actual parameters), how much an error in input affects the error in the results (resiliency to noise) and how much computation is needed.

7.3 Automated numerical simulation

3D camera scene calibration algorithm was tested using purely generated, simulated object detection data and also with manual object detection data for rendered and real recorded scenes. Results of simulations are reproduced here, because they measure the performance of the algorithm best. Real world situation tests were only used to verify that simulated data tests captured reality well enough.

In the simulation setup, image coordinates for five object detections are calculated at distances of 10 to 50m. The camera horizontal field of view was 60 degrees, and lens distortions were assumed to be removed. Camera tilt angle and height was varied. Image coordinates were subjected to random Gaussian noise, so that the noise standard deviation was 5 percent of the height of the object. Each 5 object detection scene was sampled 200 times to obtain variance estimates for the result.

The simulation results are shown in figure 19. Figures 19a and c show the performance of the second algorithm and figures 19b and d the performance of the third algorithm. As we can see in figure 19c, a simple linear estimation of the pixel horizon fails to estimate the camera height properly with tilt angles larger than 10 degrees. The second-order estimation estimates the camera height better.

This provides a simple way to obtain the ground truth and check up the algorithm performance against it. Simulation calculates ”detection points” for ”humans” standing at variable distances in the scene. These detection points can be introduced with noise and distortion to see their effect on the parameter estimates.

The first method with three unknowns (focal length, tilt angle, camera height) proved to be highly sensitive to bias and noise in the coordinate parameters. In the presence of noise, the algorithm usually did not converge at all. There is a lot of room for improvement though, e.g. a constrained version of this could work better.

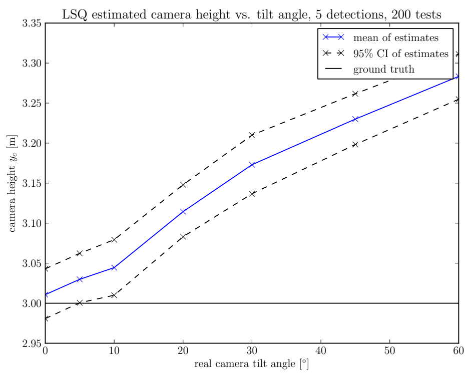
If the focal length is given, for example from using a lens calibration, this reduces the number of unknowns to two (tilt angle, camera height). Here the algorithm works robustly (i.e. a sensible solution is found) and result accuracy raises significantly. Solutions for tilt angle and camera height proved to be fairly robust in the tests. They need at least four points (two object detections, second algorithm) or six points (three object detections, third algorithm) at minimum to determine the tilt angle θ_x and camera height y_c.

Based on these results, this is the recommended method for calibration accompanied with the checkerboard calibration, which can be used to find the focal length f of a camera.

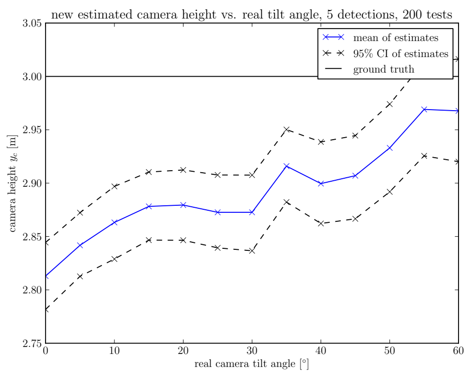
7.4 Manual testing using rendered and real images

In addition to automated numerical testing, the calibration method was also tested using rendered images, such as in figure 18. Here the operator (user) chose the object top and bottom coordinate matches in a GUI using a mouse and then the calibration algorithm was run.

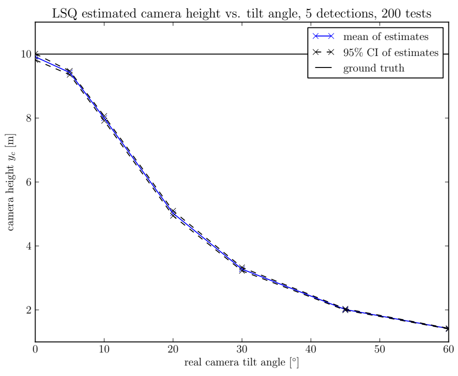
When using manual calibration, the setup time was roughly one third compared to a commercially available calibration method [30] with a known focal length, given that a camera image contains objects where matching can be easily made.



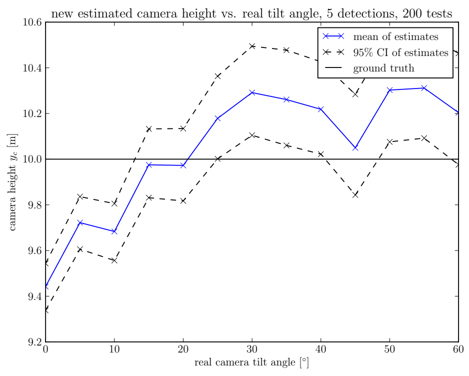
(a) Second algorithm at camera height $y_c = 3m$.



(b) Third algorithm at camera height $y_c = 3m$.



(c) Second algorithm at camera height $y_c = 10m$.



(d) Third algorithm at camera height $y_c = 10m$.

Figure 19: Camera at 3 meters, the performance of both the second and third solution is roughly equivalent in terms of error. However, at 10 meters, the second solution breaks down at high tilt angles, while the third solution still provides reasonable estimates.

7.5 Multi-camera tracking

As most time in this work went into determining a practical 3D calibration method for cameras, multi-camera tracking method was not tested much. The proposed method was mainly tested using real-world data from a system, which consisted of three ceiling-mounted cameras attached to a DVR. The method could be also tested better with synthetic data, but due to lack of time, this was not done.

If objects start to overlap (objects join together) or overlapping objects desist to overlap (objects split), it is not possible to recreate the correct paths in all cases. The frequency of such events is dependent on the moving object density (e.g. whether it is a rush hour in a store or not) and other parameters.

8 Discussion and Conclusions

Ever tried. Ever failed. No matter. Try again. Fail again. Fail better.

— Samuel Beckett: *Worstward Ho*

In this thesis, a flexible method is developed for calibrating a camera in 3D space using object detections of object of known height. The method is computationally lightweight and easy to implement using standard open-source computer vision and linear algebra libraries. Our test result suggest that it is robust to noise in calibration parameters. The method can be used either manually (setting object top and bottom image coordinates by hand) or automatically (detecting moving objects such as humans of known height).

Furthermore, we have developed a method for calibrating multiple overlapping top-down cameras and joining object tracking paths in them.

8.1 Camera calibration

The major result is that vanishing point analysis or a linear fit for the horizon does not work properly when calibrating a camera that is tilted away from the plane horizon. A second-order fit produces better results in these conditions.

The numerical simulation results show, that the camera calibration algorithm works fairly well with tilt angles less than 30 degrees and low camera install heights (2-10 meters). A low number of object detections for calibration are required (only two detections or four points are required), when operated manually, but also extra object detections can be used in automatic operation to reduce calibration error from noise. Thus it is applicable in many typical video surveillance situations.

In manual tests the calibration method requires much less time and manual tweaking of input parameters than a comparable commercial method [30][8]. Also, the proposed calibration method can be utilized with an object detection algorithm, making the system fully automated (provided that the focal length of a camera-lens system is known or calibrated). The above-mentioned commercial method does not have this property.

8.1.1 Sources of error in results

There are multiple sources for error in the estimation method. There are imperfections in the camera projection even if it has been calibrated and lens effects are accounted for. The focal length measurement using a chessboard method is not always exact, especially for very wide-angle lenses. These introduce systematic error later in the proposed calibration method.

We assume that the object is moving on the ground which is a plane. This is not often true outdoors, and causes also systematic error. The second order fit is not ideal even for a perfect plane, which introduces systematic error.

There are also inaccuracies and errors in the object detection (manual or automatic). The object might not be upright and its camera-facing side might not be a plane perpendicular to the ground plane. This affects the object detection, and the nature of the error can be both stochastic and systematic.

In the tests, these errors have not caused too much inaccuracy to undermine practical value of the results. The calibration method is good enough to be used in a video surveillance setting to help determine physical location, speed and size of moving objects.

8.1.2 Limitations and directions for further research

In the current method, camera roll rotation angle is not taken into account. Instead an angle of zero degrees is assumed, i.e. the angle between camera horizontal axis and horizon is zero. This is not problematic as the roll angle can be corrected in post-processing, but it must be accounted for, if the roll angle is large. This work does not propose any method to determine the roll angle.

Error estimates could be much improved and a proper maximum-likelihood estimate (MLE) for the calibration method could be derived. When there are a minimal number of object detections for the calibration, some a priori error estimates could be used to provide with some error estimates instead of none.

It would be also interesting to calibrate the cameras against a global world coordinate system. Once extrinsic parameters for each single camera are established independently, it would be possible to match parallel detections between overlapping cameras and determine the camera positions relative to each other. After that, multi-camera tracking is possible, e.g. using the methods discussed in the literature review.

For fully automatic operation, it would be interesting to combine the camera calibration method with human detector. Camera calibration would be run automatically for only humans moving towards or away from the camera, and a population height average would be used for as an estimate of the object's real height.

Further research is required to overcome the limitations of using a planar ground. It is possible to derive the depth map from multiple views or depth cameras (e.g. Kinect).

Automated simulation and testing of computer vision algorithms using 3D engines (such as ray tracing or modern photorealistic game engines) did prove very useful for the computer vision algorithm development. It is easy to obtain ground truth masks and position data for simulated scenes compared to real camera footage, where this has to be done either with other algorithms, precise machinery or by hand. Ray tracing and animated 3D scenes can model all kinds of real light phenomena, such as shadows, moving vegetation, time of the day effects and occluding moving objects in a precisely controlled fashion without setup high costs.

8.2 Top-down multi-camera calibration and tracking

The multi-camera calibration and tracking setup explored here is a special case. A top-down setup, where all the cameras point directly towards the ground simplifies the calibration a lot. The objective here was to determine object tracks. The proposed method is a really bare-bones approach that does not require much computation resources for operation on top of the single-camera background subtraction and object tracking. In this work, not much testing of this method was conducted.

8.2.1 Limitations and directions for further research

In multi-camera systems, there are various delays and sources of timing difference in the frames received by the computer. This poses a challenge for the multi-camera object tracking algorithms, since it increases the difficulty to make the camera handoff correctly. This work assumes that the all camera images are tagged with a global time stamp that is accurate enough and that the frame rate is high enough for operation. Path prediction and interpolation methods could be used to improve matching paths from non-synchronized cameras.

Doing a tracking separately for each camera and then doing path matching might not be the best approach. It would be interesting to experiment with various path matching approaches and use proper performance metrics discussed in the literature review. For testing, a ray tracing based animation system could be worthwhile to provide ground truth data easily. Especially testing object occlusions and overlaps, joins and splits would be much improved with a controllable test setup. This would provide better data to measure the algorithm performance.

From the experience in testing, top-down setup may not be particularly cost effective when considering e.g. customer tracking in shopping malls or shops where the ceiling height is low. Tilted, freely positioned cameras can cover larger areas with fewer units, but partial or full object occlusions are then more common.

8.3 Note

As of writing, a patent application for the camera calibration method presented in this thesis was submitted to the Finnish Patent Office. The patent application number is FI20125277 (“Method, arrangement, and computer program product for coordinating video information with other measurements”). That patent application contains material from a draft version of this thesis.

Bibliography

[1] G. Bradski and A. Kaehler. *Learning OpenCV.* O'Reilly Media, 2008.

[2] D.C. Brown. "Close-range camera calibration". In: *Photogrammetric engineering* 37.8 (1971), pp. 855–866.

[3] David K. Buck and Aaron A. Collins. "POV-Ray - The Persistence of Vision Raytracer". URL: `http://www.povray.org/`.

[4] X. Chen, J. Davis, and P. Slusallek. "Wide area camera calibration using virtual calibration objects". In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on.* Vol. 2. IEEE. 2000, pp. 520–527.

[5] D. Crandall et al. "Discrete-continuous optimization for large-scale structure from motion". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on.* IEEE. 2011, pp. 3001–3008.

[6] Mark Theodore Draelos. "The Kinect Up Close: Modifications for Short-Range Depth Imaging". MSc thesis. USA: North Carolina State University, 2012.

[7] Frederic Dufaux, Beatrice Pesquet-Popescu, and Marco Cagnazzo. *Emerging technologies for 3D video: creation, coding, transmission and rendering.* Wiley, 2013.

[8] E. Eccles et al. "Apparatus and method for camera parameter calibration". Patent US 2011/0149041 A1. 2011.

[9] T. Fukuda et al. "Seamless tracking system with multiple cameras". In: *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Confjerence of the IEEE.* Vol. 2. IEEE. 2000, pp. 1249–1254.

[10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Second. Cambridge University Press, 2004. ISBN: 0521540518.

[11] F. Hayashi. *Econometrics.* Princeton University Press, 2000.

[12] J. Heikkilä and O. Silvén. "A four-step camera calibration procedure with implicit image correction". In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on.* IEEE. 1997, pp. 1106–1112.

[13] D. Hoiem, A.A. Efros, and M. Hebert. "Putting objects in perspective". In: *International Journal of Computer Vision* 80.1 (2008), pp. 3–15.

[14] Derek Hoiem, Alexei A Efros, and Martial Hebert. "Putting objects in perspective". In: *International Journal of Computer Vision* 80.1 (2008), pp. 3–15.

[15] W. Hu et al. "A survey on visual surveillance of object motion and behaviors". In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 34.3 (2004), pp. 334–352.

[16] S. Izadi et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology.* ACM. 2011, pp. 559–568.

[17] J. Kang, I. Cohen, and G. Medioni. “Tracking people in crowded scenes across multiple cameras”. In: *Asian Conference on Computer Vision*. Vol. 7. Citeseer. 2004, p. 15.

[18] Sohaib Khan and Mubarak Shah. “Consistent Labeling of Tracked Objects in Multiple Cameras with Overlapping Fields of View”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.10 (2003), pp. 1355–1360.

[19] Rakesh Kumar, P Anandan, and Keith Hanna. “Direct recovery of shape from multiple views: A parallax based approach”. In: *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*. Vol. 1. IEEE. 1994, pp. 685–688.

[20] M. Kuusisto. “Texture and reconstruction based motion detection method for video recording systems”. MSc thesis. Finland: Helsinki University of Technology, 2008.

[21] M. Liem and D. M. Gavrila. “Multi-person tracking with overlapping cameras in complex, dynamic environments”. In: *International Journal of Computer Vision* 38.3 (2000), pp. 199–218.

[22] B. Micusik and T. Pajdla. “Simultaneous surveillance camera calibration and foot-head homology estimation from human detections”. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* (2010), pp. 1562–1569. DOI: `http://doi.ieeecomputersociety.org/10.1109/CVPR.2010.5539786`.

[23] T.B. Moeslund, A. Hilton, and V. Kruger. “A survey of advances in vision-based human motion capture and analysis”. In: *Computer vision and image understanding* 104.2-3 (2006), pp. 90–126.

[24] E.S. Raymond. *The Art of Unix Programming*. Pearson Education, 2003.

[25] A. C. Sankaranarayanan, A. Veeraraghavan, and R. Chellappa. “Object detection, tracking and recognition for multiple smart cameras”. In: *Proceedings of the IEEE* 96.10 (2008), pp. 1606–1624.

[26] L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 2001.

[27] N. Snavely, S.M. Seitz, and R. Szeliski. “Modeling the world from internet photo collections”. In: *International Journal of Computer Vision* 80.2 (2008), pp. 189–210.

[28] G. W. Stewart. *Matrix Algorithms, Volume I: Basic Decompositions*. Society for Industrial and Applied Mathematics, 1998.

[29] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.

[30] UDP Technology. *Product Application Note: Video Analytics – A New Standard*. `http://www.vcatechnology.com/public/downloads/Video-Analytics-A-New-Standard.pdf`. [Internet; fetched on 12 Nov 2013]. 2010.

[31] UDP Technology. *Product demonstration video: UDP Technology - 3D Camera Calibration*. `http://www.youtube.com/watch?v=OVIhoLpj_Io`. [Internet; fetched on 12 Nov 2013]. 2010.

[32] P. J. Withagen. “Object detection and segmentation for visual surveillance”. PhD thesis. 2006.

[33] Z. Zhang. “A flexible new technique for camera calibration”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11 (2000), pp. 1330–1334.

[34] Z. Zhang. “Flexible camera calibration by viewing a plane from unknown orientations”. In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on.* Vol. 1. IEEE. 1999, pp. 666–673.

[35] Z. Zhang et al. “Video Surveillance using a Multi-Camera Tracking and Fusion System”. In: *Multi-camera networks: principles and applications* (2008), p. 435.

[36] T. Zhao et al. “Real-time wide area multi-camera stereo tracking”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 1. IEEE. 2005, pp. 976–983.