Detection of Moving Objects for Aerial Surveillance of Arbitrary Terrain

Der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Dipl.-Ing. Marco Munderloh

geboren am 21.09.1977 in Wilhelmshaven.

2015

Referent: Prof. Dr.-Ing. J. Ostermann
Korreferent: Prof. Dr.-Ing. C. Heipke
Tag der Promotion: 24.03.2015

Acknowledgments

The time working at the Institut für Informationsverarbeitung (TNT) of the Gottfried Wilhelm Leibniz Universität Hannover to earn the degree Doctor of Engineering has been intensive and highly instructive. It had a big influence on my life and career. Especially the wide-ranging research topics at the institute allowed me an insight into many areas of coding and computer vision, and by their connection this work was originated.

My special thanks go to Prof. Dr.-Ing. Jörn Ostermann as the supervisor of my thesis for his support, his guidance and ideas, and for always taking the time for discussions. I also like to thank Prof. Dr.-Ing. Christian Heipke for being the external examiner and for his detailed review and helpful comments and Prof. Dr.-Ing. Bodo Rosenhahn as the chair of the examination board for his suggestions and hints during writing.

Furthermore, I thank all my colleagues at the institute. Our discussions helped me focusing and finding the right path to go. In particular, I like to mention Thorsten Laude, Holger Meuel, Hendrick Hachmann, and Julia Schmidt for their support and encouragement and Stella Graßhof, who was always open-minded for questions, especially concerning math.

Finally, a huge thanks goes to my family: my parents Heidrun and Hans-Gerd, my brother Timo and his wife Nicole, and of course to my wife Claudia and my son Martin Justus, whose smile always brings sunlight into cloudy days. I am very grateful for your understanding not having me there at so many evenings and weekends during the finalization of this thesis.

Contents

1 Introduction 1

1.1 Motion Estimation and Parameter Extraction 4
1.2 Problems of Current Ground Motion Models 5
1.3 Contributions 8
1.4 Outline 9

2 Basic Principles 10

2.1 Scene Model 10
2.2 Camera Model 11
2.3 Epipolar Geometry 19
2.4 Essential and Fundamental Matrix 20
2.5 Projective Transformation and the Homography 21
2.6 Moving Object Detection by Background Subtraction 23
2.7 Motion Estimation from Image Sequences 25
2.8 Dense Optical Flow 31

3 Planar Landscape Error Model 34

3.1 Aerial Surveillance Scene Model 34
3.2 Vertical Aerial Photo 37
3.3 Motion Parallax Displacement 42
3.4 Camera with a Tilt Angle 46
3.5 Arbitrary Camera Orientation 51
3.6 Height Restrictions based Outlier Detection 54
3.7 Detectability in Dependence of Speed and Direction of Motion 57

4 Multi-Planar Landscape Model based on Triangle Meshes 58

4.1 Mesh Creation 59
4.2 Outlier Removal and Moving Object Detection 61
4.3 Mesh-based Motion Compensation 63
4.4 Accuracy Analysis of the Mesh-based Approach 65

5 Experiments 73

5.1 Motion Vector Classification 73
5.2 Motion Compensation Performance 82
5.3 Moving Object Detection Performance 85

6 Summary and Conclusions 93

Bibliography 97

Abbreviations and Symbols

Abbreviations:

BMA	Block Matching Algorithm
CCD	Charge-Coupled Device
CMOS	Complementary Metal-Oxide Semiconductor
CRF	Corner Response Function
FOV	Field Of View
GLONASS	GLObalnaja NAwigazionnaja Sputnikowaja Sistema
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KLT	Kanade-Lucas-Tomasi feature tracker
MAV	Micro Air Vehicle
MPP	Motion Parallax Predictor classifier
PTZ	Pan, Tilt and Zoom camera system
RANSAC	RANdom SAmple Consensus
ROI	Regions of Interest
SIFT	Scale-Invariant Feature Transform
SURF	Speed-Up Robust Features
UAV	Unmanned Aerial Vehicle

Symbols:

α	angle of aperture of the camera
β, γ, θ	pan, roll, and tilt angle of the camera
λ	parameter of a line equation
λ_1,λ_2	Eigenvalues of $\mathtt{M}$
$\lambda_{\mathrm{m1}}, \lambda_{\mathrm{m2}}$	parameters of the triangle plane equation
$\mathtt{A}$	affine matrix of size 2×2
$b_k(\mathbf{n})$	binarized image intensity differences of the frame k
$\mathbf{c}(c_x,c_y)$	principal point offset
$\mathbf{C}(C_x,C_y,C_z)^\top$	position of the camera in world coordinates
$\mathbf{C}_k$	position of the camera in the frame k
$\Delta\mathbf{C}$	vector between two camera centers

$\Delta\mathbf{c}, \tilde{d}_0$	motion parallax of the ground plane in image plane coordinates
D_L	diameter of the camera lens
D_P	distance of scene point $\mathbf{P}$ to the triangle surface
d_f	minimum feature distance
$d_k(\mathbf{n})$	image intensity differences of the frame k
$\mathbf{d}(d_x,d_y)^\top$	displacement vector
$\mathbf{d}_i(d_{i,x},d_{i,y})^\top$	displacement of the ith feature
$\hat{\mathbf{d}}$	estimate of $\mathbf{d}$
$\mathbf{e}$	position of an epipole
$e_k(\mathbf{n})$	binarized image intensity differences of the frame k after erosion
$\mathtt{E}$	elementary matrix of size 3×3
f	focal length
$\mathbf{f}_{i,k}$	position of the ith feature in the frame k
$\mathtt{F}$	fundamental matrix of size 3×3
$\mathbf{g}_{k-1}$	holds the temporal derivatives of I
h	height of a scene point above the ground plane
$h_{11}...h_{33}$	the elements of $\mathtt{H}$
$\mathtt{H}$	homography matrix of size 3×3
$I(\mathbf{n})$	image intensity at the position $\mathbf{n}$
$I_k(\mathbf{n})$	image intensities of the frame k
I_x,I_y	partial derivatives of I
k	frame index
k_H	Harris weighting factor
Δk	number of frames between the source and destination frame used for feature tracking and motion compensation
$\mathtt{K}$	camera calibration matrix of size 3×3
$\mathbf{l}$	epipolar line
$\mathtt{M}$	Harris corner matrix
$\mathbf{M}_{0,1,2}$	mesh node positions of a triangle in world coordinates
$\mathbf{m}_{0,1,2}$	mesh node positions of a triangle on the image plane
n	number of features
N_x,N_y	amount of sensor elements in x- and y-direction
$\mathbf{n}(n_x,n_y)^\top$	point in image coordinates
$\bar{\mathbf{n}}, \bar{\mathbf{n}}_0$	normal vector and unit normal vector of a triangle surface
$\mathbf{N}$	position of the nadir in world coordinates
$\Delta\mathbf{n}_c, \Delta\mathbf{n}_m$	relief displacement and motion displacement in pel
$\mathbf{O}$	projection of the image plane origin onto the ground plane
$\mathbf{p}(x,y)^\top$	point on the image plane

$\mathbf{p}'(x',y')^\top$	displaced position of the point $\mathbf{p}$
$\tilde{\mathbf{p}}(x_d,y_d)^\top$	point on the image plane with lens distortions
$\mathbf{p}_k$	point on the image plane of camera $\mathbf{C}_k$
$\hat{\mathbf{p}}_k$	estimate of $\mathbf{p}_k$ through affine motion compensation
$\mathbf{p}_0,k$	projection of $\mathbf{P}'_0$ onto the image plane of camera $\mathbf{C}_k$
$\mathbf{p}_h,k$	projection of $\mathbf{P}_h$ onto the image plane of camera $\mathbf{C}_k$
$\mathbf{P}(X,Y,Z)$	point in world coordinates
$\tilde{\mathbf{P}}(X_c,Y_c,Z_c)^\top$	point in camera coordinates
$\mathbf{P}_0, \mathbf{P}'_h$	point on the ground plane in world coordinates
$\tilde{\mathbf{P}}_0, \tilde{\mathbf{P}}'_h$	point on the ground plane in camera coordinates
$\mathbf{P}_h$	point on an object width height h in world coordinates
$\tilde{\mathbf{P}}_h$	point on an object width height h in camera coordinates
$\Delta\mathbf{p}$	relief displacement of $\mathbf{p}$
$\Delta\mathbf{p}_m$	motion parallax of $\mathbf{p}$
$\Delta\mathbf{P}$	relief displacement projected to the ground plane
$\Delta\mathbf{P}_m$	motion parallax projected to the ground plane
$\mathbf{q}(q_1,q_2)^\top, q$	projective components of the homography
r, r_d	radii of $\mathbf{p}$ and $\tilde{\mathbf{p}}$ to the center of distortion
r_{fps}	frame rate
$r_{11}...r_{33}$	the elements of $\mathtt{R}$
$\mathtt{R} = \mathtt{R}_\theta \mathtt{R}_\gamma \mathtt{R}_\beta$	camera orientation matrix of size 3×3
$r_k(\mathbf{n})$	pixel-wise motion detection results of the frame k
Δr_c	relief displacement in radial direction
ΔR_c	relief displacement projected to the ground plane in radial direction
s_w, s_h	width and height of the camera sensor
T_1, T_2, T_3	thresholds of the cluster filter
T_b, T_r	binarization and erosion thresholds of the noise filter
T_d	distance threshold of the motion parallax outlier detector
$t_{i,k}$	set referencing the mesh nodes of the triangle i in the frame k
$\mathtt{T}_{i,k}$	matrix containing the nodes of the triangle i in the frame k
$\mathbf{t}$	translation vector component of a homography
$u,v,\mathbf{u}, \mathbf{v}$	arbitrary feature indexes and positions
$\mathbf{v}_{\text{plane}} = (v_x,v_y)^\top$	velocity and flight direction of aircraft
$\mathbf{v}_{\text{thresh}}$	minimal object speed needed for detection
W	search window set of pixels
Δx_c	relief displacement in x direction
ΔX_c	relief displacement projected to the ground plane in x direction

Abstract

The detection and segmentation of moving objects in aerial video sequences is a common application in safety and environmental monitoring. The challenge hereby is the non-static camera which is attached to and moves together with an aerial vehicle. To be able to detect local changes due to movement of ground objects in such a scenario, the displacements of the image pixels resulting from the motion of the camera need to be compensated between the frames of the recorded sequence. For this purpose, the motion of the camera as well as the structure of the recorded scene needs to be known to compensate the global motion without errors. While the motion of the camera can be measured accurately enough by external sensors or can even be estimated from the video feed itself, the structure of the observed scene is commonly unknown. Therefore, easy to compute universal approximations of the scene structure are made instead. The most common method is to model the global motion of the pixels by a projective transformation using a homography, in which the observed scene is assumed to be planar. While this might be accurate enough for high altitudes, small focal lengths, and vertically downwards oriented cameras, the approximation fails for scenes with high buildings or low altitudes due to motion parallax effects. As a result, the global motion for large areas of the frame is estimated and compensated incorrectly, which leads to lots of falsely detected local motion for such scenarios. In this work, the problem is addressed in two ways: first, the approximation errors made by the projective transformation for scenes with high buildings, low altitudes, and tilted or sideways looking cameras are analyzed mathematically. From the resulting aberration equations, a predictor for the motion parallax of image pixels is created and used as an outlier and moving object detector. It is called motion parallax predictor classifier is this work and able to distinguish between global motion of the background, displacements resulting from the motion parallax of static objects such as buildings, and local motion of individual objects in the scene. In contrast to similar methods, e.g. the elementary or fundamental matrix conformance test, only a small range on the epipolar line is determined as a valid representation of the possible motion parallax of static scene objects. This allows the detection of local moving objects moving along the epipolar line, which is not possible with epipolar geometry alone. However, the predictor has restrictions when it comes to objects moving along the epipolar line in the direction of the motion parallax: only objects with a displacement larger than the motion parallax are detectable. Moreover, the intrinsic and extrinsic camera parameters must be known, which requires external sensors and calibrated cameras. For this reason, an additional detector based on the clustering of frame to frame displacements of feature points (cluster filter) is developed in this work. It uses similarity constraints to join the estimated displacement into clusters of equal motion. This allows the detection of local moving objects without explicit knowledge of the flight altitude, camera parameters and motion, or the scene geometry. Compared to the homography and fundamental matrix based methods used as references, the cluster filter as well as the motion parallax predictor classifier were able to classify up to 100% of the moving objects correctly. Moreover, the negative predictive value is increased from 30 to over 90% at the same time. The second way of addressing the problem is the global motion compensation of image pixels for the use in an image differences based system. As the investigated scenario does not conform with the single planar model of the homography, a multi-planar approach using a mesh of locally adaptive triangle patches is presented. In contrast to the homography, the mesh is able to adapt to objects sticking out of the ground plane by using an individual affine mapping for each triangle, e.g. the wall of a building and the roof. This allows the compensation of the global motion nearly error free, leaving only the newly occurring background as a possible source of false detections. Compared to the single planar model, the multi-planar approach was able to reduce the amount of falsely classified pixels in the experiments by a factor of 4.

Keywords: aerial surveillance, motion detection, motion segmentation, non-planar motion compensation

Kurzfassung

Das Erkennen bewegter Objekte in Luftbildsequenzen ist eine häufige Aufgabe in der Luftüberwachung. Die Herausforderung liegt hierbei in der Unterscheidung der globalen Verschiebung der Pixel zwischen den Bildern, hervorgerufen durch die Bewegung der Kamera, und lokalen Bewegungen durch die zu erkennenden Objekte. Um diese trennen zu können, muss die globale Bewegung kompensiert werden. Hierfür muss sowohl die Bewegung der Kamera als auch die Geometrie der überwachten Szene bekannt sein. Während sich die Bewegung der Kamera mittels externer Sensoren oder aus der Bildsequenz selbst heraus ermitteln lässt, ist die überflogene Szene meist unbekannt und wird daher unter Verwendung eines einfach zu bestimmenden Modells approximiert. Das meist genutzte Modell ist hierbei die Homographie, welche die überflogene Szene durch einer Ebene annähert. Diese Approximation gilt allerdings nur für große Flughöhen, kleine Brennweiten und lotrechte Aufnahmen. Entspricht die Landschaft nicht diesem Modell, z.B. wegen hoher Gebäude, niedriger Flughöhe, etc., führen die unterschiedlichen Bewegungsparallaxen zwischen der Oberfläche und Gebäuden zu Fehldetektionen in großen Bereichen des Bildes. In dieser Arbeit wird das Problem auf zweierlei Arten angegangen. In der ersten Methode wird zunächst der Approximationsfehler des Homographiemodells mathematisch bestimmt. Aus den sich ergebenen Fehlergleichungen wird ein Prädiktor erstellt, der die Bewegungsparallaxen von statischen Objekten bis zu einer vorgegebenen Maximalhöhe voraussagt. Der in dieser Arbeit Bewegungsparallaxeklassifizierer genannte Detektor erlaubt die Unterscheidung zwischen einer Bild-zu-Bild Verschiebung aufgrund statischer Objekte wie Hintergrund oder Gebäuden und einer lokalen Verschiebung durch ein sich bewegendes Objekt anhand des Abstandes zu einem prädizierten Epipolarliniensegment. Im Gegensatz zu Verfahren, die auf der Elementar- oder der Fundamentalmatrix basieren, erlaubt dieser Ansatz auch die Detektion von Bewegungen entlang der Epipolarlinie. Allerdings werden Objekte, welche sich in Richtung der Bewegungsparallaxe bewegen, nur erkannt, wenn die Eigengeschwindigkeit ausreichend hoch ist. Außerdem müssen für dieses Verfahren die intrinsischen und extrinsischen Kameraparameter bekannt sein. Aus diesen Gründen wurde ein zweiter Detektor entwickelt, welcher auf dem Clustern von Bewegungsvektoren basiert und als Clusterfilter bezeichnet wird. Das Vektorfeld wird hierbei anhand von Ähnlichkeitsbedingungen in Bereiche gleicher Bewegung eingeteilt, wobei innerhalb der Bereiche eine sanfte Änderung der Bewegungsrichtung erlaubt wird. Hierdurch wird eine Erkennung von Objekten unabhängig von der Bewegungsrichtung und ohne zwingende Kenntniss der Flughöhe oder der Kameraparameter ermöglicht. Sowohl der Bewegungsparallaxeklassifizierer als auch das Clusterfilter erreichen dabei eine Erkennungsrate bewegter Objekte auch für nicht planare Sequenzen von bis zu 100%, bei gleichzeitig niedrigerer Fehlalarmrate als das Referenzverfahren. Die zweite Methode hat als Ziel die Verbesserung der globalen Bewegungskompensation, welche z.B. für Detektoren notwendig ist, die auf Bilddifferenzen arbeiten. Im vorgestellten Verfahren wird hierbei das Einzelebenenmodell der Homografie durch ein Multiebenenmodell auf Basis von stückweise planaren Dreiecksnetzen ersetzt, in dem jedes Dreieck eine individuelle Ebene darstellt. Hierdurch ist es möglich, auch Objekte abzubilden, die aus der Grundebene herausstehen, in dem z.B. die Bewegung der Wand oder des Daches eines Gebäudes individuell bewegungskompensiert wird. Im Gegensatz zum Referenzverfahren lässt sich hierdurch die Anzahl der fälschlicherweise als bewegt erkannten Pixel dramatisch reduzieren, so dass Fehldetektionen nur noch an neu auftauchendem Hintergrund auftreten. Im Experiment ließen sich die Fehldetektionen um den Faktor 4 reduzieren.

Schlagworte: Luftbildüberwachung, Bewegungserkennung, Bewegungssegmentierung, nicht-planare Bewegungskompensation

1 Introduction

The detection and segmentation of moving objects in video sequences is a common application in safety and environmental monitoring. One of the use cases is the detection of alarm events, e.g. something is moving in a specific area, unusual behavior of people, or abandoned objects. In a static camera scenario where the examined scene is quasi-static over time and therefore the projected image background is nearly constant and perspective distortions are practically non-existing, there has been a lot of research done and many algorithms are available. Most of them are based on background subtraction. One simple solution to separate changing foreground from static background is by subtracting the pixel luminances of two images from different points in time, as done in [2]. This results in a difference image with high energy in areas of change. To increase the robustness, a background reference can be computed from multiple images, e.g. by temporal mean or median filtering [11], and used for the image subtraction instead. This eliminates noise and temporarily visible objects from the reference and enables an adaption to slowly changing conditions such as lighting. Statistical methods such as Gaussian Mixture Models (GMM) are a more sophisticated and a widely-used way of constructing and adapting a background reference model [57, 51, 35].

If the camera is moving, however, the problem is more complex as the image background is not static anymore but changes with the camera movement. To cope with this, a global motion estimation and compensation has to be performed, meaning the displacement for each of the image pixels has to be estimated and compensated before applying the change detection algorithm (Figure 1.1). As the displacement of the image background is dependent on the camera movement only, it can be expressed by a model of the camera motion and a model of the background structure, whose parameters have to be determined.

In the case of pole or tree mounted cameras moved by e.g. wind turbulence or ground vibration, or in the case of pure rotation, the degrees of freedom are limited. A simple camera motion model with two or three parameters might be accurate enough to describe the displacement of the image content, assuming the observed scene to be planar and the camera displacement between frames to be small. Concerning more complex motion paths, as they appear in freehand or vehicle mounted cameras, or in the case of aerial surveillance, this simple approach is not sufficient to cover the extensive position and orientation changes the camera is able to perform, and a more sophisticated model has to be incorporated which allows more degrees of freedom.

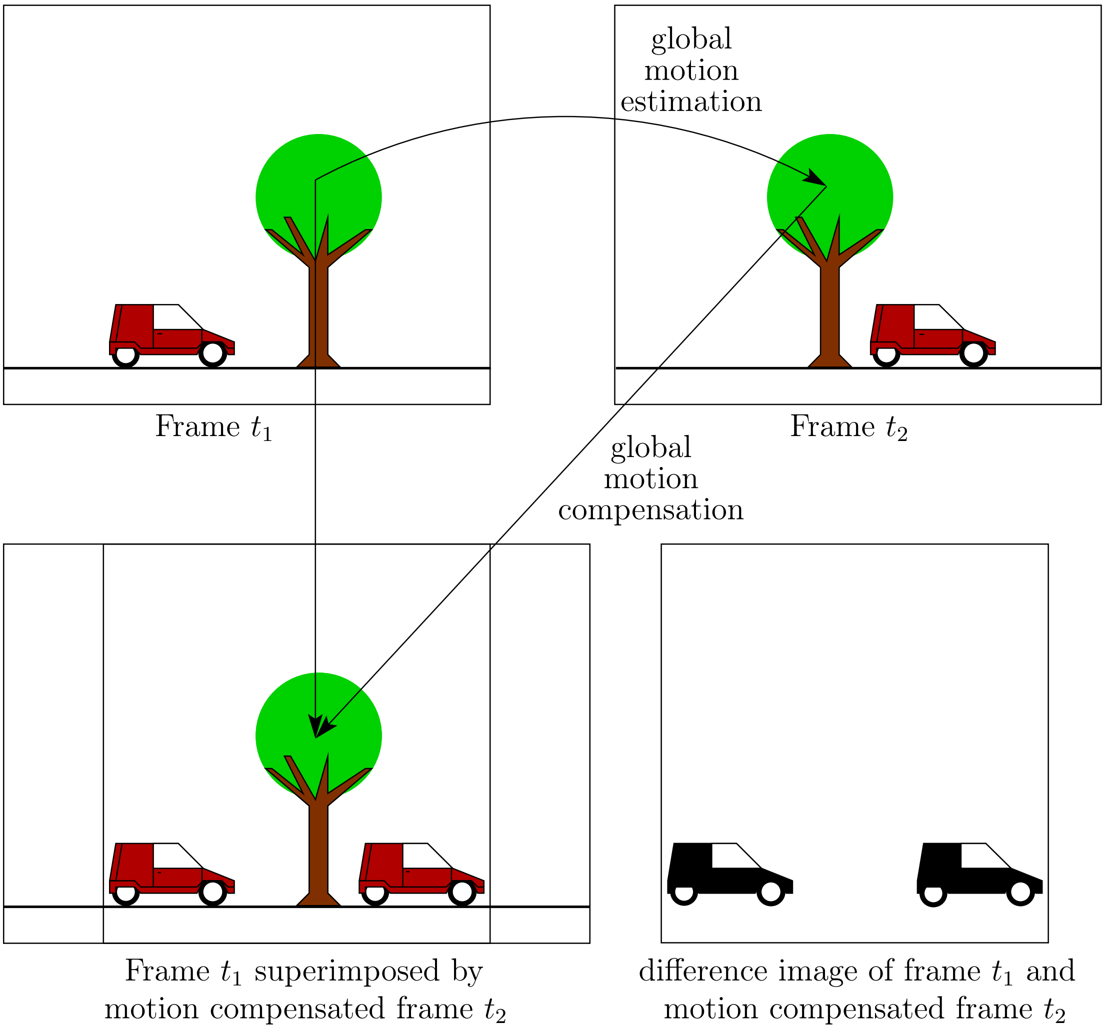

global
motion
estimation
Frame t_1
Frame t_2
global
motion
compensation
Frame t_1 superimposed by
motion compensated frame t_2
difference image of frame t_1 and
motion compensated frame t_2

Figure 1.1: Difference image calculation after global motion compensation

In the case of aerial surveillance, which is the topic of this work, a major task is the detection of objects operating at the ground level. To monitor ongoing events on the surface of the earth, usually electro-optical sensors, consisting of video or infrared cameras, are in use, which might either observe large areas for overview purposes or dynamically aim at and zoom in on interesting spots (Regions of Interest, ROI) for further inspection. Concerning the motion of the camera sensors, as they are directly mounted to the aerial vehicle, the changes in position and orientation are equivalent to the ones of the flying platform itself. To gather this motion information, it is common practice to attach a satellite navigation systems such as GPS, GLONASS or Galileo to the airborne platform as done for e.g. Synthetic Aperture Radar (SAR) imaging [47, 52]. It is assisted by inertial systems (INS/IMU) to achieve adequate accuracy and to determine the angular orientations roll, tilt, and pan.

However, to reliably estimate and compensate the global motion in the optically recorded sequences for surveillance and the detection of moving ground objects, this accuracy is not high enough. Therefore it is appropriate to use the GPS/INS data as a rough prediction and refine it by exploiting the data from the image sequence directly. This can be done by registering the sensor data to external maps or beacons [10], or by relying just on the optical data alone, e.g. by using optical flow techniques [13].

Additional to the motion information of the camera, a model of the structure of the image background is needed to differentiate local from global motion. The most common model in aerial surveillance to approximate the surface of the earth is a plane, which is valid for high altitudes, for small focal lengths, and for the sensors pointing perpendicularly downwards. Assuming a pinhole camera, this allows the movement of the image background pixels to be modeled as an affine or projective transformation of the pixels between the frames by a homography mapping. A homography describes the mapping of pixels between two frames taken from the same scene at two different positions in projective space by assuming that all scene points, which are the source of the projected pixels on the image plane, are located on a single planar surface [68]. This model is therefore valid for planar scenes only, as they appear in high altitude aerial surveillance or in the recording of remote scenarios which can be assumed as being more or less at an infinite distance. As the model is simple, robust and easy to compute, it is widely-used for the creation of panorama images by registering the single shots to each other and stitching them into one large frame. However, for low altitudes, tilted cameras, and urban scenarios, the planar model assumption is invalidated and the model fails.

To improve those scenarios is the topic of this thesis. This is done in two ways: the first one is to improve the detection of moving objects, independent of the image content, using frame-to-frame displacement vectors only. The homography is used hereby as a classifier to distinguish local object and global background motion vectors. To improve the detection, the approximation error of the homography is mathematically analyzed in detail. By incorporating known camera properties and assumptions of the maximum height of background structure into the error equations, a motion parallax predictor of static scene objects is created and used as a classifier instead. Moreover, a second classifier based on motion vector clustering is introduced. As a benefit, it does not rely on any a prior scene knowledge or known camera parameters at all. The second way of improving the detection of moving objects in aerial video sequences is to improve the global motion compensation of image pixel between the frames directly to generate a quasi-static camera scenario for background subtraction algorithms. Herby, the single-plane homography model is replaced by a multi-planar local adaptive approach, based on 2D triangle meshes.

1.1 Motion Estimation and Parameter Extraction

To determine the motion of the background from the recorded image content alone, the motion information has to be extracted from the image sequence. The estimation of motion in image sequences is a well investigated problem in the computer vision and video coding community. The main motion estimation algorithm used in video coding today is a block matching algorithm (BMA): it minimizes the sum of absolute or squared differences between blocks of pixel in the current frame and candidate blocks from preceding or succeeding frames. Those candidates are selected from an area within a search window surrounding the position of the current block. The current frame is divided into a grid of blocks and one motion vector is computed for each of them by determining the best fitting candidate block. The resulting motion vectors are applied to all pixels of the corresponding block equally [34]. To increase the resolution of the resulting motion vector field beyond the needs of the video coder, a sliding window approach might be used by shifting the position of the block over the frame in smaller steps than the size of the block leading to overlapping blocks [48, 49].

However, one problem remains: the minimum is always evaluated over the whole content of the block. This leads to problems if more than one motion exists within one block. On the other hand the size of the block cannot be arbitrarily reduced as the chance of mismatches due to image noise increases. One solution is the use of a hierarchical block matcher combined with a motion vector candidate selection on the lower levels [36]. The block matching process is started with a large block size giving an initial motion vector. The large block is then divided into smaller blocks with their search window origin set to the displacement position of the larger block and their search window size reduced. This is repeated until the block size becomes relatively small. Then only the already existing displacement vectors of the next higher levels are tested and chosen from. This is repeated for all positions of the frame, resulting in a dense motion vector field, also called optical flow.

Besides block matching algorithms, there are other widely spread optical flow techniques. One major optical flow estimator is the Lucas-Kanade method [43, 42]. It uses the least-squares approach to solve the optical flow equation for a local group of pixels assuming the motion to be constant in a local neighborhood. As the differential equation is only reliable for a small displacement, the well known Kanade-Lucas-Tomasi (KLT) [64] feature tracker uses a multi-resolution image pyramid to extend the range. As the Lucas-Kanade method relies on the image gradient distribution (so called structure tensor), unstructured areas of the image are hard to track. In the KLT feature tracker, a pre-selection step is involved to select and track only those points in the image which have a high “cornerness“ [56]. A very successful corner detector was introduced by Harris and Stephens [25] and has proven to most reliably detect consistent feature points [54]. Another way of determining the optical flow from features is the matching of so called keypoints, as done in SIFT [41], RIFT [39], or SURF [4]. Keypoints do not only represent a position in the image but additionally contain a description vector of its surroundings which is independent of scale or lighting. In contrast to feature tracking, keypoints are detected in each frame individually and are concatenated by matching their description vectors. This allows the assignment of keypoints to objects and a matching to known objects from a database. This work is based on corner features, as a high location precision and good feature distribution is more important than having a huge baseline matching ability, as it was shown in [14], or the possibility to recognize objects.

With the extracted 2D motion vectors available, the motion model parameters according to the background motion model can be determined by applying it to the model equations, e.g. by using Random Sample Consensus (RANSAC) to determine the parameters of a homography. Afterwards, all motion vectors can be tested against the calculated motion model and mismatches, so called outliers, can be removed, allowing a refinement of the model parameters by just using the remaining inliers as input. The determined motion model can then either be used to compensate the global motion by applying it as a transformation to every pixel of the frame or as a classifier to compare the displacement of a feature calculated from the motion model to the actual measurement [3, 60]. For the latter, it has to be ensured that all moving objects in the scene are covered by at least one motion vector. This is true for dense vector fields but not guaranteed for feature based systems.

In addition to the homography model with the planar constraint, it is possible to use epipolar geometry as a classifier by calculating the elementary or the fundametal matrix [26], similar to the homography matrix by using all feature displacements as input. By evaluating the distance of the measured feature displacements to the associated epipolar lines, outliers and moving objects can be detected without a constraint on the background structure [66].

1.2 Problems of Current Ground Motion Models

If the camera undergoes translational movements and the scene is not planar and not at infinity, the homography model is not sufficient to describe the displacement of the image pixels accurately enough, resulting in imperfect image registration and perspective aberrations. In the case of pixel-based motion detection, this leads to image differences between the current and motion compensated frame not only caused by the local motion of objects but by the 3D structure of the background and the simplistic surface model. This is illustrated in Figure 1.2: the non-planar parts of the building appear in the difference image additionally to the moving car. Hence moving objects and motion compensation errors cannot be reliably distinguished, which lead to false alerts.

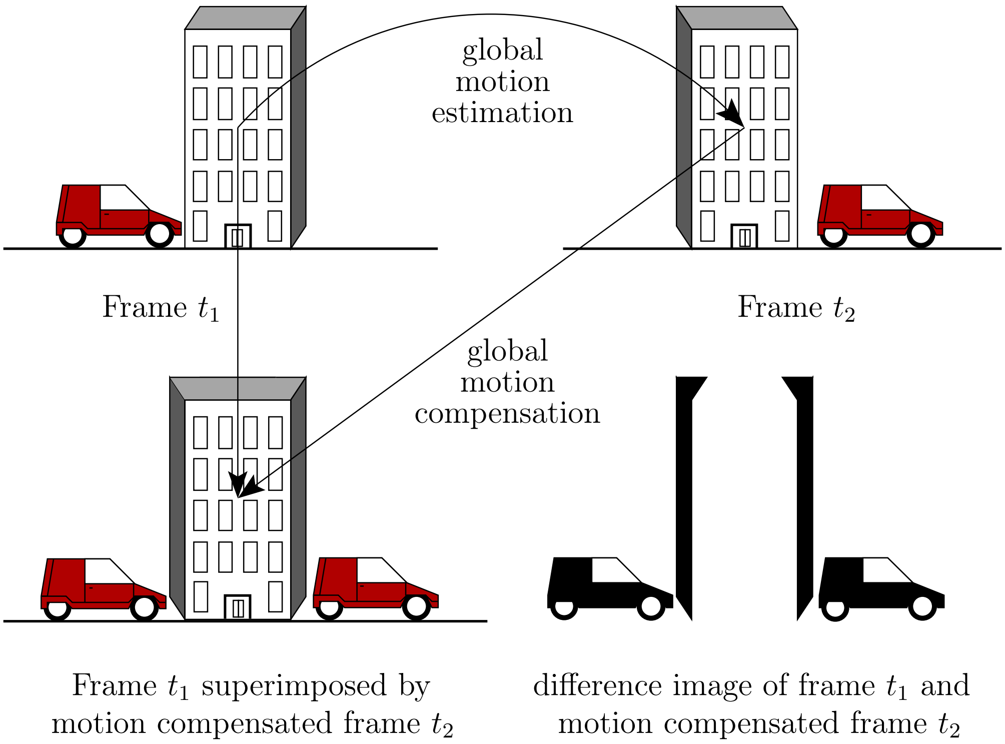

global
motion
estimation
Frame t_1
Frame t_2
global
motion
compensation
Frame t_1 superimposed by motion compensated frame t_2
difference image of frame t_1 and motion compensated frame t_2

Figure 1.2: Difference image calculation after global motion compensation with perspective distortions due to 3D scene structure

Concerning aerial surveillance, the approximation of the ground level as a plane is only valid for high altitudes and small focal lengths. However, due to the widespread use of small and cheap operating flying systems such as Micro Air Vehicles (MAV) or Micro-UAVs (Unmanned Aerial Vehicles), which usually operate at low altitudes, the insufficient approximation of the ground level becomes more and more of a problem [9] and leads to perspective distortions if objects stick out of the ground plane as illustrated in Figure 1.3: the distance between $\mathbf{p}_1$ and $\mathbf{p}_3$ stays constant between the image planes but the position of $\mathbf{p}_2$ changes towards $\mathbf{p}_1$ because of the scene point $\mathbf{P}_2$ not being on level with the ground plane. The model assumptions are violated and perspective distortions occur.

Moreover, systems which use pictures of the surface of the earth taken by airborne vehicles with the objective of creating large image mosaics have to deal with perspective distortions when stitching the picture tiles together. These pictures are imprecisely called satellite images in e.g. Google maps [24] or Bing maps [45], although the image database is based on both satellite and aerial imagery. To be able to see the sides of buildings, lower altitude images are not taken with the camera pointing directly downwards but with a tilt angle θ, and are known as oblique images. While the homography model is still valid for objects on the plane, the

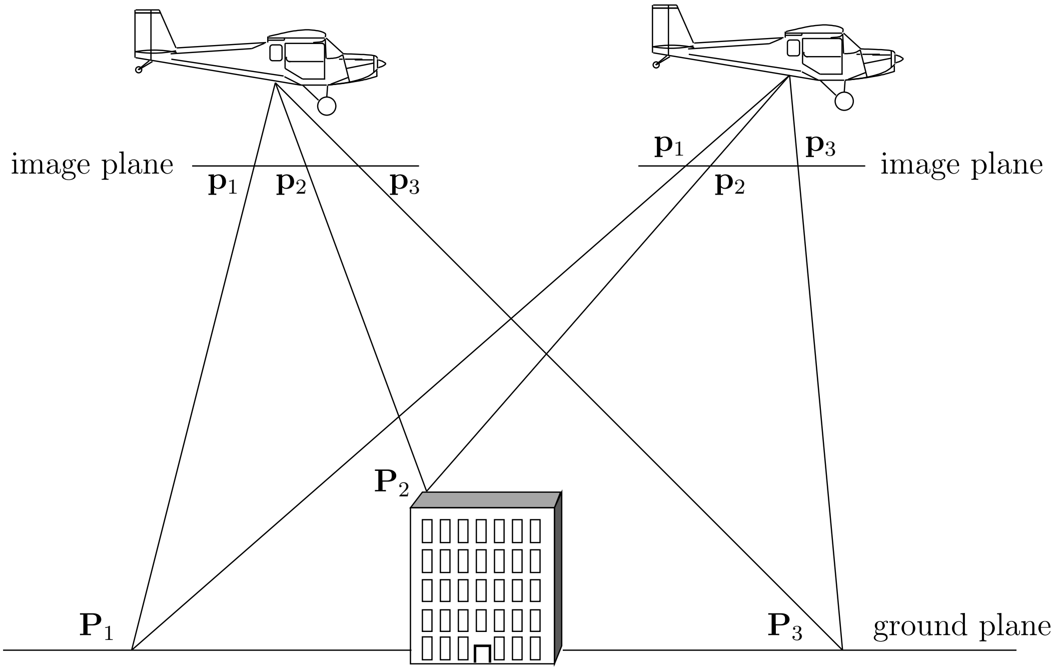


Figure 1.3: Aerial surveillance model, camera pointing downwards. Planes and building from [12].

displacement of non-planar objects in the image plane is increased compared to the case of the camera pointing directly downwards. The problem is illustrated in Figure 1.4. The projection of $\mathbf{P}_2$ onto the image plane should be the same as the one of $\mathbf{P}'_2$ to be independent from the plane's position.

To cope with this problem, state of the art techniques utilize a priori knowledge in two ways: One is to rely on externally created 3D models describing the texture and structure of the observed area as an input to compensate the object height related pixel displacements of the image background [44]. The alternative is to use geo-information data like road and building maps to mask out areas with problematic or occluded structure, and just detect moving objects on planar regions such as roads [50]. The former method needs a precise and up-to-date 3D model of the observed area mandatorily, but the on-board creation of such a model in real time is not an option due to the limited electric and computational power. Hence it is not possible to use such a system for areas where no model was created yet, the available data is outdated, or whose structure has largely changed, e.g. by a disaster like floods or earthquakes. This is also a problem for the latter approach, although the area selection is less effected by structural changes and maps are more commonly available than 3D models. It has to be added that, because of the masked out areas and the necessary safety distances, the area where motion detection is possible at all is very limited in these systems.

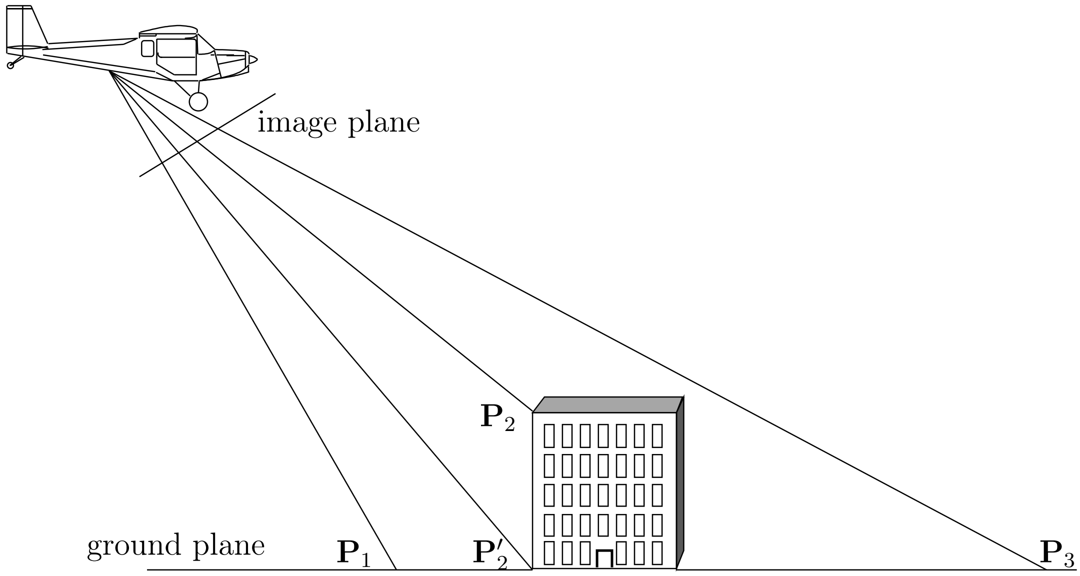


Figure 1.4: Aerial surveillance model under tilt angle. Plane and building from [12].

Regarding feature-only approaches, the separation of global and local motion has similar problems. If the model is violated, e.g. by a non-planar structure and a homography model, the model parameters become imprecise and the detection is less accurate or large regions are falsely classified as containing motion. Moreover, as outliers and moving objects cannot be distinguished, additional techniques have to be applied to eliminate the outliers. Concerning the epipolar model, if objects move in direction of or against the epipolar point, their feature positions move along the epipolar line and are therefore not detectable.

1.3 Contributions

- An accuracy analysis and aberration model is provided in this work to describe the aberrations of a homography model for the case of camera motion and non-planar scenes. Calculations are done for an aerial surveillance scenario; however, the resulting equations are valid for other purposes as well. The vertical photo case is investigated first and than extended to cameras having a tilt angle or performing arbitrary movements.
- Using the aberration model, an outlier and moving object detector is created. It predicts the motion parallax of static background objects by assuming a maximum height. This results in an epipolar line segment, which is used in this work to classify feature displacements into background and moving object by evaluating the distance between the measured displacements of the feature

and the predicted epipolar line segment. In contrast to epipolar geometry, this method also allows for the detection of objects moving along the epipolar line.

- Following this, a second outlier and moving object classifier is presented for arbitrary scene structure and camera movement. In contrast to the former method, it does not require calibrated cameras and accurate camera motion information. The only assumption made is a smooth displacement vector field of the background. It is compared to the state-of-the-art methods as well as the motion parallax predictor classifier.
- To improve the pixel-wise global motion compensation of non-planar scenes, a piece-wise planar locally adaptive mesh-based motion compensation is presented and evaluated, coping with the issues of the single planar model. It combines the vector-based and pixel-based motion detection methods by first using the displacement vector field to create a deforming mesh and then applying piece-wise image warping [21, 16] to compute the motion compensated reference image for use in a pixel-based motion detection algorithm. Compared to a full 3D ground model, this largely reduces the amount of complexity as all calculations are done in 2D, but still gives the benefits of a 3D ground model [46].
- The benefits of the presented moving object detectors as well as the performance of the proposed motion compensation algorithm are demonstrated and rated using ground truth data.

1.4 Outline

The remaining work is organized as follows: in Chapter 2, basic principles of the feature extraction and the feature tracking process are introduced, as well as the perspective camera and the scene model used is this work. This is continued by presenting the epipolar geometry, the projective and affine mapping by a homography, and the calculation of a Delaunay triangulation used in the mash-based motion compensation. Chapter 3 follows with an aberration analysis of the homography model for non-planar scenes, using the example of aerial surveillance. It starts with the aberrations occurring in vertical photos and is extended to cameras with a tilt angle and arbitrarily moving cameras. The gained knowledge is used afterwards to create a moving object and outlier detector based on the prediction of the motion parallax of static scene objects with height restrictions. To cope with the identified accuracy problems of the planar motion compensation, a mesh-based multi-planar approach is introduced in Chapter 4, including a comparison of the remaining aberrations to the single planar model. The chapter is followed by experimental results in Chapter 5 and finally concluded in Chapter 6.

2 Basic Principles

In this chapter, the fundamental principles and geometric models used in this work are presented. It starts with the introduction of a scene model and a camera model. Two coordinate systems are introduced which describe the global position of the camera and all objects in the scene in world coordinates on the one hand as well as the mapping characteristics of the camera-lens system in local camera coordinates on the other hand. Following this, the pinhole camera model is presented as a basic model in computer vision. The single camera geometry is then extended to multiple views by introducing models to describe the relations between different camera views of the same scene. This is followed by discussing the process of motion extraction and model estimation from image sequences as well as the elimination of outliers. The chapter ends with the process of interpolating a dense motion vector field from the sparse representation.

2.1 Scene Model

The scene model, illustrated in Figure 2.1, is a parametric description of the whole scene. Its global world coordinate system (X,Y,Z) contains all objects and all cameras. A diffuse ambient lighting is assumed to make the model invariant to changes of surface orientations and lighting related effects such as shadows. The camera itself has its own local coordinate system, the camera coordinate system (X_c,Y_c,Z_c).

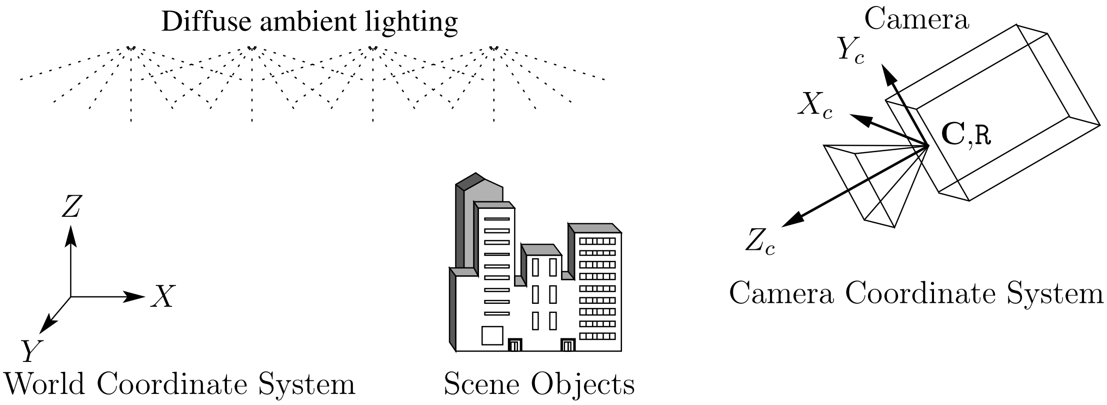


Figure 2.1: Scene Model (building from [12])

Its origin is set to the center of projection, also known as the camera center. It corresponds to the center of the light sensitive sensor element. The Z-component is pointing towards the scene through the center of the camera lens, whereas the X- and Y components are aligned to the coordinate axis of the camera sensor. Camera coordinates can be mapped to world coordinates by applying a rotation $\mathtt{R}$, expressing the local orientation of the camera coordinate system compared to the world coordinate system, and the position of the camera center $\mathbf{C}$ in world coordinates.

2.2 Camera Model

The projection of 3D scene objects onto the 2D image plane of a camera is described by the camera model. It is the concatenation of a perspective projection model, a lens model, and a sensor model [63]. Figure 2.2 illustrates the projection of a scene point $\tilde{\mathbf{P}}$ in camera coordinates through the camera model into $\mathbf{n}$ on the image plane in image coordinates. In this work, a non-distorting camera lens is assumed, so that $\tilde{\mathbf{p}} = \mathbf{p}$. In practice, distortions of a known camera lens can be corrected by a post processing step in the digital domain.

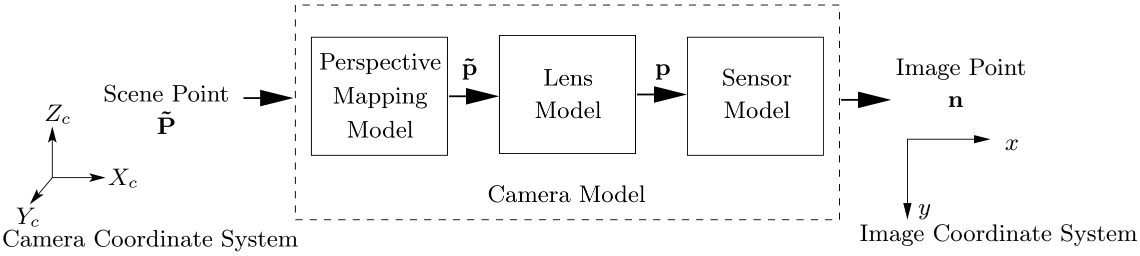


Figure 2.2: Camera mapping model (based on [63])

2.2.1 Sensor Model

Concerning the sensor model, digital video and still image cameras manufactured prior to around 2005 almost exclusively contained CCD sensors (Charge-Coupled Device). They are analog devices and consist of an array of photo diodes and energy storages. Light radiation hitting the diodes cause electric charges which are accumulated over the exposure time in integrated capacitors. The energy is read out by shifting each charge over the elements to an output node at the edge of the sensor where they are consecutively amplified, and, in the case of digital cameras, quantized and coded into intensity values. The quantized energy values are stored into the picture buffer as picture elements [31].

With the emergence of HDTV cameras, CCD sensors have more and more been replaced by CMOS devices (Complementary Metal-Oxide Semiconductor). In contrast to CCD sensors, CMOS sensor elements can be read out directly without the need of an energy storage layer or charge shifting. This makes them faster, mainly for high resolution, and more flexible, as it allows to read out only parts of the sensor. Additionally they are cheaper to produce, use less energy, and they allow the integration of additional on-chip processing steps such as amplifying or A/D conversion. The main drawbacks are less light sensitivity and, in general, a lower dynamic range than CCD elements - although research is catching up swiftly. Today, almost all camera sensors are manufactured in CMOS technology.

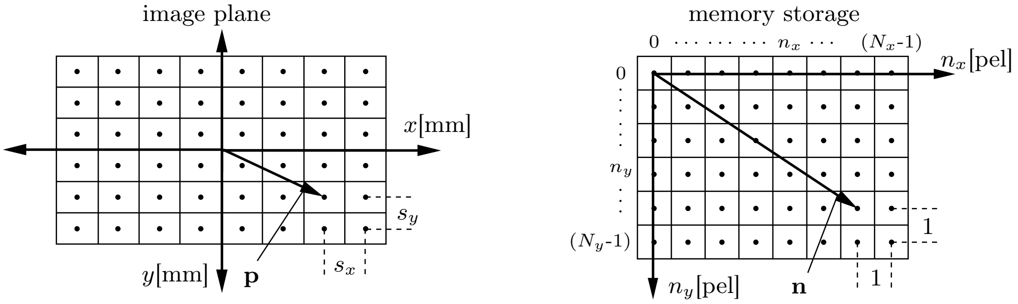


Figure 2.3: Geometric relations between the image plane on a camera sensor and the image stored in memory (based on [63])

Images are stored in computer memory as a concatenated chain of bytes. The stored elements of an image are called picture elements, or in short pixel, and have the unit [pel]$\widehat{=}$1. The size of this storage is dependent on the amount of light sensitive sensor elements, often referred to as the resolution of the sensor. We define N_x being the amount of sensor elements in one row of the sensor and N_y being the number of rows of elements. The resulting image in the memory of the computer has then a size of $N_x \times N_y$ bytes, assuming one byte per pixel (luminance only and an 8 bit quantizer). The start of the image storage is represented by a so called pointer, which points to the address of the first pixel of the image in memory. To access the n_xth pixel in the n_yth line of the image, one has to add $n_y \cdot N_x + n_x$ to that pointer. N_x is called the image width and N_y is called the image height. This representation defines a new image coordinate system, with the origin at the upper left pixel of the image (where the pointer points to). To convert the discrete image coordinates $\mathbf{n} = (n_x, n_y)^\top$ without a unit into measured units $\mathbf{p} = (x, y)^\top$ on the image plane, the origin of the coordinate system has to be displaced to the center of the image (see Figure 2.3) and the size of a pixel has to be scaled to the size of a sensor element. As the size of the sensor may not be equal in horizontal and vertical direction, we define two scaling factors for the two dimensions:

$$s_x = \frac{s_w}{N_x}, \qquad s_y = \frac{s_h}{N_y}, \tag{2.1}$$

with s_w and s_h being the width and the height of the sensor. Using the scaling factors, pixel can be converted into coordinates on the image plane:

$$\mathbf{p} = \begin{bmatrix} s_x & 0 & -0.5(N_x-1)s_x \\ 0 & s_y & -0.5(N_y-1)s_y \end{bmatrix} \mathbf{n}, \tag{2.2}$$

and image plane coordinates into pixel, respectively:

$$\mathbf{n} = \begin{bmatrix} 1/s_x & 0 & 0.5(N_x-1) \\ 0 & 1/s_y & 0.5(N_y-1) \end{bmatrix} \mathbf{p}. \tag{2.3}$$

2.2.2 Lens Model

Real camera lenses do not have a linear mapping throughout the lens. Wide angle cameras in particular tend to add non-linear behavior, resulting in geometrical distortions. In the camera model, this is treated in the lens model block. It performs a non-linear mapping between the perspective mapping coordinates $\tilde{\mathbf{p}} = (x_d, y_d)^\top$ and the real image coordinates on the image plane $\mathbf{p}$. The main geometric distortion is the radial distortion [17, 67]. It can be approximated by a Taylor expansion:

$$r = r_d(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4 + \cdots), \tag{2.4}$$

wherein r_d is the distorted and r the undistorted distance of a point on the image plane from the center of distortion. The center of distortion is hereby often assumed to be equal to the origin of the camera coordinate system. As the first coefficient κ_1 has the main influence on the final radial distortion, only the second order radial distortion is used as a approximation in the following lens model mapping equation:

$$\mathbf{p} = (1 + \kappa_1 r_d^2)\,\tilde{\mathbf{p}}, \tag{2.5}$$

with $r_d^2 = x_d^2 + y_d^2$ as the squared distance of $\tilde{\mathbf{p}}$ from the origin.

2.2.3 Additional Lens Characteristics

Figure 2.4 illustrates a simplified camera and lens model. The complex optical system of a real camera lens is approximated as one single lens. The distance between the image plane and the lens is referred to as the focal length f, as all rays emitting from the scene point $\tilde{\mathbf{P}}$ and traveling through the lens are bent such that they focus in one single point $\tilde{\mathbf{p}}$ on the image plane. In Figure 2.4 the beam of light rays is represented by the dashed lines. In the case of an infinitesimal lens (or an infinitely small aperture), only one ray of light remains (solid line). In this case, as no rays of light have to converge, all possible scene points are always in focus, independent of their distance to the lens. A camera having an infinitesimal small lens is called a pinhole camera.

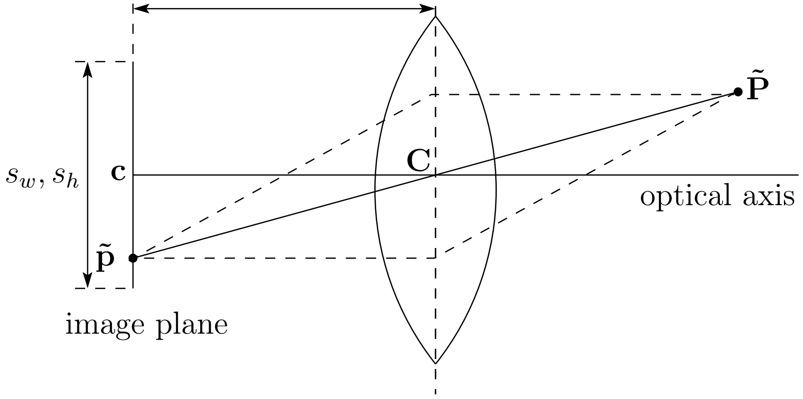


Figure 2.4: Simplified camera and lens model

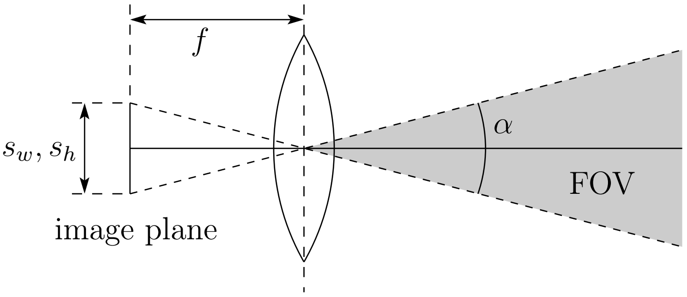


Figure 2.5: Angle of view α and field of view (FOV) for a sensor of size $s_w \times s_h$ and focal length f

Another property of a camera lens is the opening angle or angle of view α of the camera. It directly corresponds to the viewable area of the scene, called field of view (FOV) (illustrated in Figure 2.5 as the gray area). All rays of light emitted by objects in this area are mapped through the lens onto the surface of the sensor. The maximum angle of view is thereby dependent on the size of the camera sensor (s_w, s_h), the diameter of the lens D_L the light can pass, and the focal length f:

$$\alpha = 2 \arctan \left(\frac{\min(D_L, \sqrt{s_w^2 + s_h^2}}{2f} \right). \tag{2.6}$$

The lens is assumed to be large enough so the limiting factor are the sensor dimensions only. As the sensor may not be quadratic, the angle of view can be different for the x and y-axis:

$$\begin{aligned}\alpha_x &= 2\arctan\left(\frac{s_w}{2f}\right),\\ \alpha_y &= 2\arctan\left(\frac{s_h}{2f}\right).\end{aligned} \tag{2.7}$$

The common width of a camera sensor is 36mm (full frame format). All focal length specifications on optical lenses refer to this size. For sensors with different sizes, the effective focal length has to be converted: e.g. for the APS-C format with a width of 22.2mm, all focal lengths have to be multiplied by

$$\frac{36\text{mm}}{22.2\text{mm}} = 1.621. \tag{2.8}$$

The vertical size of the sensor can be derived from the aspect ratio: for HDTV cameras the aspect ratio is 1.8 (16 : 9), resulting in a vertical sensor size of

$$\frac{36\text{mm}}{1.8} = 20\text{mm}. \tag{2.9}$$

This leads to an angle of view for common focal lengths from 20mm to 200mm of:

$$84.0^\circ \geq \alpha_x \geq 10.3^\circ \tag{2.10}$$

in horizontal direction and

$$53.1^\circ \geq \alpha_y \geq 5.7^\circ \tag{2.11}$$

in vertical direction, respectively. But with increased angle of view, the resolution per angle decreases: a sensor in 1920 pel×1080 pel full-HD resolution yields to an angular resolution between

$$\frac{1920\text{ pel}}{84.0^\circ} = 22.9\frac{\text{pel}}{^\circ} \qquad \text{and} \qquad \frac{1920\text{ pel}}{10.3^\circ} = 186.7\frac{\text{pel}}{^\circ}. \tag{2.12}$$

Often, the viewable area as well as the achievable spatial resolution is of interest. If the distance of the camera to the recorded scene is known, these can be calculated from the angular resolution. For the example of aerial photography, the necessary calculations for vertical and tilted cameras are provided in Chapter 3.1.

2.2.4 Perspective Mapping Model

The perspective mapping model does only handle the mapping of an ideal pinhole camera, meaning no aberrations of the camera lens occur and all objects are in perfect focus. With this model, an object point $\tilde{\mathbf{P}}$ is mapped onto an image point $\mathbf{p}$ where a straight line connecting the object point and the pinhole meets the image plane (see Figure 2.4). In practice, camera lenses are used instead of a pinhole to achieve reasonable imaging properties and light sensitivity. Although the path of rays is different (dashed lines in Figure 2.4) than in an ideal lens, the rays combine at the same spot on the image plane and produce the same mapping a pinhole would have. Therefore (in this ideal case) the camera lens is defined by its optical enter $\mathbf{C} = (C_x, C_y, C_z)^\top$ and the distance and size of the image plane only.

The distance between the optical center and the image plane is called focal length f. The point on the image plane intersected by the optical axis is called the principal point $\mathbf{c}$. It is often assumed to be the origin of the coordinate system of the image plane.

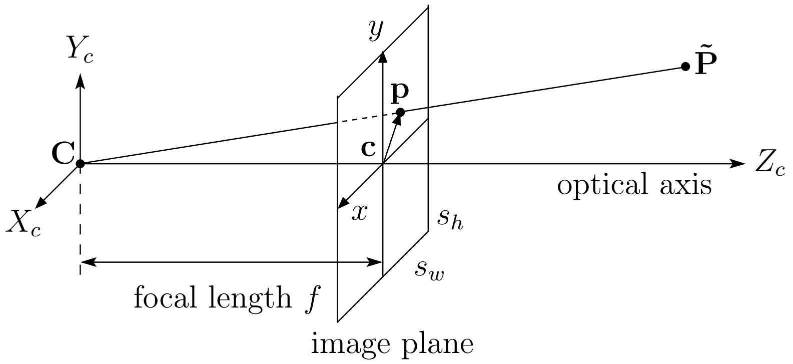


Figure 2.6: Central projection

In this representation, all image coordinates are mirrored as they would be in a real pinhole camera. However, as the camera image is inverted anyway by the camera itself and for easier calculations, it is valid to move the image plane in the model in equidistance to the other side of the pinhole, as shown in Figure 2.6. This results in uninverted image coordinates of the projected scene points. The optical center is now also referred to as the center of projection or the camera center. As all rays intersect at this central point, this representation is called central projection.

As depicted in Figure 2.6, the point $\mathbf{C}$, the point $\mathbf{p} = (x, y, f)^\top$, and the point $\tilde{\mathbf{P}} = (P_x, P_y, P_z)^\top$ are collinear (lying on the same straight line). Therefore, the mapping between them can be described by the theorem of intersecting lines:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{P_z} \cdot \begin{pmatrix} P_x \\ P_y \end{pmatrix} \tag{2.13}$$

In this theorem, the projection is carried out by assuming a virtual plane at distance P_z from the optical center $\mathbf{C}$, parallel to the image plane at distance f. The mapping is performed by scaling the remaining two space coordinates of the point at distance P_z by the ratio of their distances $\frac{f}{P_z}$.

Homogeneous Coordinates

Instead of a mapping between two parallel planes, the projection can be described directly in 3D space by using projective space geometry. In this representation, a two space point $(a,b)^\top$ is represented in 3D space in the homogeneous form $(a,b,1)^\top$. This 3D space vector represents a line which is defined by the center of projection $\mathbf{C}$ and, in this case, the point $(a,b,1)^\top$. But also all other points on this line are valid homogeneous representations of the two space point $(a,b)^\top$, as they only differ by scaling. Therefore, it its valid to write $(a,b,1)^\top = k(a,b,1)^\top = (\frac{a}{k},\frac{b}{k},k)^\top$.

In Figure 2.6, the points $\mathbf{p} = (x,y,f)$ and $\tilde{\mathbf{P}} = (P_x,P_y,P_z)$ define the same line together with $\mathbf{C}$ and are therefore per definition the same homogeneous coordinate:

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ f \end{pmatrix} = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \tilde{\mathbf{P}} \tag{2.14}$$

To get the Euclidean representation of Equation 2.14, $\mathbf{p}$ and $\tilde{\mathbf{P}}$ have to be normalized such that their third dimension becomes 1. This is done by dividing through the third dimensions:

$$\frac{1}{f}\begin{pmatrix} x \\ y \\ f \end{pmatrix} = \frac{1}{P_z}\begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} \tag{2.15}$$

The 3rd dimension can now be removed which returns the vectors back to the Euclidean representation. Both divisors can be moved to the right side, leading back to Equation 2.13 and describing the theorem of intersecting lines.

In Equation 2.13 it is assumed that the optical axis intersects the origin of the image plane coordinate system. This is only true if the origin of the image plane is perfectly aligned to the optical center of the lens. To get a more general representation for real world cameras, an principal point offset $\mathbf{c} = (c_x,c_y)^\top$ might be added to the equation:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{P_z} \cdot \begin{pmatrix} P_x \\ P_y \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{2.16}$$

The great advantage of homogeneous coordinates and the projective space geometry is that all geometrical operations become linear and can be written as matrix multiplications. If the third dimension of $\mathbf{p}$ is normalized to 1, Equation 2.16 can simply be written as:

$$\mathbf{p} = \mathtt{K}\tilde{\mathbf{P}}, \tag{2.17}$$

wherein the matrix $\mathtt{K}$ contains all inner camera parameters, in this case the focal length f and the image plane offset $\mathbf{c}$. It is therefore referred to as the camera calibration matrix:

$$\mathtt{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

World Coordinates to Camera Coordinates

Equation 2.17 only describes the mapping of a point in camera coordinates. To map an arbitrary scene point $\mathbf{P} = (X,Y,Z)^\top$ in world coordinates, the point must first be expressed as a point $\tilde{\mathbf{P}}$ in camera coordinates. This is done by a rotation and translation of the coordinate axis as follows: First, the world coordinate system is translated such that the camera center becomes the new origin. This is achieved by subtracting the position of the camera center $\mathbf{C}$ in the world coordinate system from the observed scene point $\mathbf{P}$. Following this, the axis of the translated coordinate system have to be aligned to the orientation of the camera coordinate system. The alignment is performed by a rotation around each of the axis. The angles are known as pan, roll, and tilt and they are noted β, γ, and θ as rotation around the Z-, X- and Y-axis, respectively. All three rotations can be combined into one 3×3 rotation matrix $\mathtt{R}$, with the order of rotation Y, X, Z:

$$\mathtt{R} = \mathtt{R}_\theta \mathtt{R}_\gamma \mathtt{R}_\beta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.19}$$

Using the camera center offset and the orientation, $\mathbf{P}$ can be expressed in camera coordinates as:

$$\tilde{\mathbf{P}} = \mathtt{R}(\mathbf{P} - \mathbf{C}). \tag{2.20}$$

By inserting Equation 2.20 into Equation 2.17, the projection of a point $\mathbf{P}$ in world coordinates into a point $\mathbf{p}$ on the image plane can be written as:

$$\mathbf{p} = \mathtt{KR}(\mathbf{P} - \mathbf{C}) \tag{2.21}$$

The offset of the camera center $\mathbf{C}$ and the rotation matrix $\mathtt{R}$ describe the position and orientation of the camera in the world coordinate system. They are therefore called the extrinsic camera properties.

If Equation 2.21 is transferred to the Euclidean space by normalizing the third dimension to 1, dividing thorough it, and solving the equation for x and y, it leads to the following form known as the collinearity equations in the field of photogrammetry:

$$\begin{aligned} x &= f\,\frac{r_{11}(X-C_x)+r_{12}(Y-C_y)+r_{13}(Z-C_z)}{r_{31}(X-C_x)+r_{32}(Y-C_y)+r_{33}(Z-C_z)} \\ y &= f\,\frac{r_{21}(X-C_x)+r_{22}(Y-C_y)+r_{23}(Z-C_z)}{r_{31}(X-C_x)+r_{32}(Y-C_y)+r_{33}(Z-C_z)}, \end{aligned} \tag{2.22}$$

wherein r_{11} to r_{33} are the elements of $\mathtt{R}$. Note that the sign of the Z-components must be changed between the world and the camera coordinate system as the camera coordinate system is left handed and the world coordinate system is right handed.

2.3 Epipolar Geometry

The epipolar geometry describes the geometry of two pinhole camera views of the same scene at distinct positions. By applying the pinhole camera model (Chapter 2.2), constraints can be formulated for the relationship of image points between different camera views originated by the projection of objects in 3D space on the image planes without the actual knowledge of the 3D structure of the scene.

This is illustrated in Figure 2.7. If a 3D scene point $\mathbf{P}$ is observed by a camera 1, the projection of $\mathbf{P}$ into $\mathbf{p_1}$ on the image plane 1 is, according to the pinhole camera model, the intersection of the line of sight between $\mathbf{P}$ and the camera center $\mathbf{C}_1$ and the image plane 1. If only the projection $\mathbf{p_1}$ is known, there is no information available of the distance between the camera center $\mathbf{C_1}$ and the scene point $\mathbf{P}$ (depicted as the $\mathbf{P}$?). But what is known is that the scene point $\mathbf{P}$ must lie somewhere on the line of sight.

If we now want to formulate relationships and constraints between the observed projected point $\mathbf{p_1}$ and the projection of the unknown scene point $\mathbf{P}$ onto the image plane 2 of a second camera 2, we can project the known line of sight of camera 1 onto the image plane of camera 2. This can be seen as the intersection of a plane, spanned by the line of sight of camera 1 and the connection of the two camera centers, referred to as the baseline, with the image plane 2 of the camera 2. The plane is called the epipolar plane of $\mathbf{p_1}$ and the projection of the line of sight is called the epipolar line $\mathbf{l}_2$ of $\mathbf{p_1}$. If the epipolar lines of several points are drawn, we will note that all lines intersect in one point, the so called epipole. It is also the intersection of the baseline with the image planes, as the baseline is the common part of all epipolar planes. In Figure 2.7, these points are denoted as $\mathbf{e}_1$ and $\mathbf{e}_2$. It is also the projection of the camera center of the other camera.

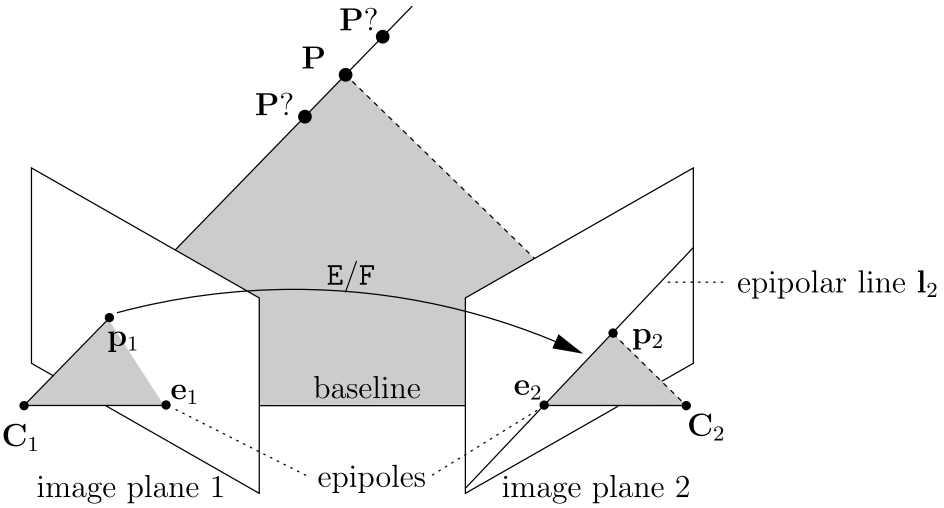


Figure 2.7: Without the knowledge of scene point $\mathbf{P}$, the projected image point $\mathbf{p}_1$ of camera $\mathbf{C_1}$ can be mapped by the essential matrix $\mathtt{E}$ (for known camera properties) or the fundamental matrix $\mathtt{F}$ (unknown camera properties) to an epipolar line $\mathbf{l}_2$ on the image plane of camera $\mathbf{C}_2$ (based on [26]).

2.4 Essential and Fundamental Matrix

The essential matrix ($\mathtt{E}$ matrix) was presented to computer vision in 1981 by Longuet-Higgins [40]. It describes the metric relationship of the epipolar geometry between two images taken of the same scene with calibrated cameras from different positions. With the essential matrix known, one point in the image plane of camera $\mathbf{C_1}$ can be mapped into the associated epipolar line in the image plane of camera $\mathbf{C_2}$ and vice versa. Without noise and errors, the projection of the original scene point $\mathbf{p}_1$ has to lie on the associated epipolar line $\mathbf{l}_2$:

$$\mathbf{l}_2 = \mathtt{E}\,\mathbf{p}_1. \tag{2.23}$$

This can also be written as a contraint:

$$\mathbf{p}_2^\top \mathtt{E}\,\mathbf{p}_1 = 0. \tag{2.24}$$

For the calculation of the $\mathtt{E}$ matrix, the cameras must be calibrated with all coordinates beeing normalized.

In 1992, Faugeras [20] and Hartley [27] pointed out that the properties of the E matrix can also be applied to uncalibrated cameras in projective geometry. In this generalized case, the matrix is called the fundamental matrix $\mathtt{F}$. Analog to the $\mathtt{E}$ matrix, the $\mathtt{F}$ matrix is also a 3×3 matrix defined as:

$$\mathbf{l}_2 = \mathtt{F}\,\mathbf{p}_1. \tag{2.25}$$

If $\mathbf{P}$ is a scene point and $\mathbf{p}_1$ and $\mathbf{p}_2$ are projections of that point onto the image planes of two cameras, the $\mathtt{F}$ matrix describes the mapping between both points such that:

$$\mathbf{p}_2^\top \mathtt{F}\, \mathbf{p}_1 = 0. \tag{2.26}$$

The essential and the fundamental matrix are related such that they can be converted by incorporating the calibration matrices of both cameras $\mathtt{K}_1$ and $\mathtt{K}_2$:

$$\mathtt{E} = \mathtt{K}_2^\top \,\mathtt{F}\, \mathtt{K}_1. \tag{2.27}$$

As the essential matrix has less degrees of freedom (5) compared to the fundamental matrix (7) due to the known calibration properties, some additional constraints can be applied for its determination: two of its singular values are equal and nonzero and the other one is zero. These properties are used in the eight-point algorithm presented by Higgins in [40] to estimate the $\mathtt{E}$-matrix from a given set of image correspondences.

However, in practice the gathered image sequences are often recorded with the camera properties unknown. Therefore, the $\mathtt{F}$-matrix is more common. To estimate the $\mathtt{F}$-matrix, the same eight-point algorithm is usable with the constraints that one singular value is zero and the others are non-zero. In practice, however, it is pretty frequent that one of the non-zero values becomes pretty close to zero compared to the other ones [19]. If more than eight points are used for the estimation which are all effected by noise, it can happen that more than one singular value becomes a candidate of being the zero one, which makes the linear system of equations hard to solve.

To address this problem, Hartley proposed in [28] a normalized eight-point algorithm. Hereby the image coordinates are transformed independently into a new coordinate system prior to calculating the $\mathtt{F}$ matrix. However, in the case of aerial image sequences, the $\mathtt{F}$-matrix is often still ill-conditioned. An additional method of computing $\mathtt{F}$ is by using the homography $\mathtt{H}$ and at least two off-plane correspondences (plane induced parallax [26]). But again it has to be made sure that the off-plane correspondences result from non-planar static objects and that the camera translation between the frames is large enough for the parallax to have any effect, which is hard to achieve in overflight sequences.

2.5 Projective Transformation and the Homography

Mathematically, a projective transformation is an invertible linear mapping from projective space to itself that maps straight lines to straight lines. Applied to computer vision it is able to describe the mapping of points between two transformed

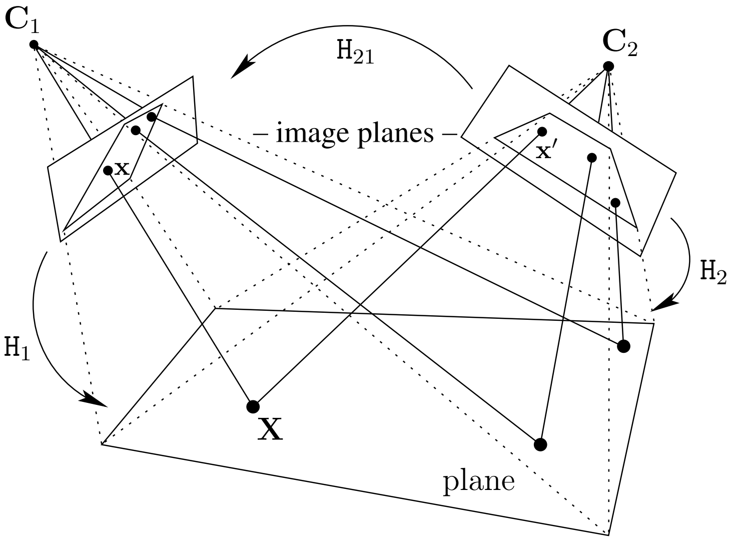


Figure 2.8: Homography mapping between a projected point on image planes of a points on a plane in 3D space

planes in space. It can also be used to describe the mapping of pixels between the image plane of a camera and a planar surface in space assuming central projection. Similar to the $\mathtt{E}$ and $\mathtt{F}$ matrices, a projective transformation, also known as a homography, is expressed as a 3×3 matrix $\mathtt{H}$ [68]:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \tag{2.28}$$

or in matrix notation:

$$\mathbf{x}' = \mathtt{H}\mathbf{x}. \tag{2.29}$$

The Euclidean form of Equation 2.28 can be derived by dividing through the additional dimension added in Equation 2.14, assuming $x_1 = x$, $x_2 = y$ and $x_3 = 1$:

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}. \tag{2.30}$$

A homography can be split up into its single properties[26]:

$$\mathtt{H} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ q_1 & q_2 & q \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{q}^\top & q \end{bmatrix}, \tag{2.31}$$

where $\mathbf{t}$ denotes the translational part, $\mathtt{A}$ describes an affine two dimensional scaling and rotation and $\mathbf{q} = (q_1, q_2)^\top$ adds the non-linear properties of a projectivity. q

is just a scaling factor and has no further influence. $\mathtt{H}$ is typically normalized such that $q = 1$. Therefore projective transformation has eight degrees of freedom.

A more restricted form of the projective transformation is the affine transformation. It requires the two coordinate systems of the transformation to be Euclidean (rectilinear). In this case, the mapping can be expressed with only six degrees of freedom, wherein the vector $\mathbf{q}$ is $(0{,}0)^\top$. This leads to the affine inhomogeneous mapping equation:

$$x' = h_{11}x + h_{12}y + h_{13}, \qquad y' = h_{21}x + h_{22}y + h_{23}. \tag{2.32}$$

A mapping of pixels between the image planes of two cameras observing one planar surface in space can be seen as a concatenation of two projective transformations ($\mathtt{H}_1$ and $\mathtt{H}_2$ in Figure 2.8). As long as the plane is static, the same calculations can be done for only one camera observing the plane from two different positions in space. The main application in this case is the retrieval of the extrinsic camera parameters.

As the coordinate systems of all the planes in Figure 2.8 are assumed to be Euclidean, each projection of the image points from the 3D plane into the image planes of the cameras is an affine transformation described by Equation 2.32. The direct mapping of one image plane into the other image plane is therefore a concatenation of two affine transformations - from the image plane of the first camera to the 3D plane $\mathbf{x} = \mathtt{H}_1\mathbf{X}$ and back to the image plane of the second camera $\mathbf{x}' = \mathtt{H}_2\mathbf{X}$ - which can also be expressed directly as one single projective transformation [26] (illustrated in Figure 2.8):

$$\mathbf{x}' = \mathtt{H}_2\mathtt{H}_1^{-1}\mathbf{x} = \mathtt{H}_{21}\mathbf{x}. \tag{2.33}$$

A homography can be estimated from four independent point correspondences (see Chapter 2.7) between the two planes, if the points in both planes are not collinear. Given Equation 2.30, this yields to eight equations which solve the eight unknowns of $\mathtt{H}$. An affine transformation has only six degrees of freedom. Therefore three independent point correspondences are enough to create the necessary six equations by using Equation 2.32.

2.6 Moving Object Detection by Background Subtraction

The detection of motion in a scene is often based on background subtraction. Hereby a simple [2, 11] or more sophisticated [57, 51, 35] model of the background is created beforehand or constructed and adapted during runtime. To detect moving objects, the constructed background image is subtracted from the current input image, leaving only the changing foreground (moving objects). These methods assume a static background and camera. In case of a moving camera, the movement of the background pixels due to the camera movement must be compensated prior to computing the model. This allows (within limits) static camera motion detection algorithms to detect local motion in a moving camera scenario. The detection results highly depend on the accuracy of the background motion compensation, as an inaccurate compensation of the background motion leads to image differences between the background model and the image background of the current frame which are not caused by moving objects and therefore lead to false positive detections.

In this work, a simple background subtraction based outlier detector is used to evaluate the performance of the state-of-the-art global motion compensation and the proposed mesh-based approach. The algorithm is kept simple, as it is only used to compare the motion compensation algorithm in the sense of background motion compensation accuracy. However, the results should be transferable to any more sophisticated motion detection algorithm.

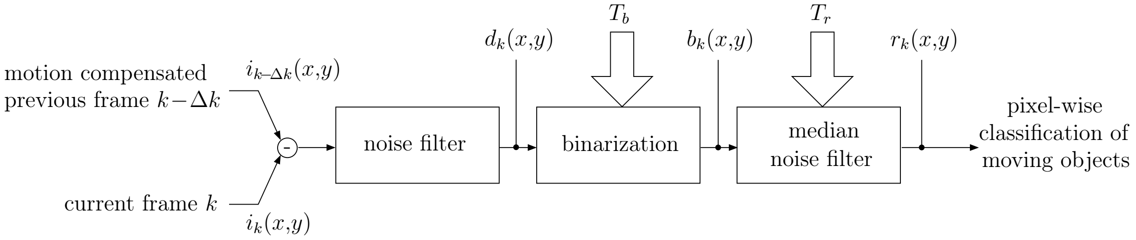


Figure 2.9: Simple image background subtractions based motion detection system.

As shown in Figure 2.9, the simple motion detection algorithm first computes a difference image by calculating the absolute pixel differences d_k between the motion compensated frame $k-\Delta k$ and the current frame k. Afterwards, a noise filtering is performed by summing up the pixel differences of a 3×3 neighborhood. These pixel differences are binarized by comparing the summed up energy value against a threshold T_b. The output of the filter is the pixel-wise binary decision b_k:

$$\begin{aligned} b_k(x,y) &= \begin{cases} 1 & \text{for } d_k(x,y) \geq T_b \\ 0 & \text{for } d_k(x,y) < T_b \end{cases}, \text{ with} \\ d_k(x,y) &= \sum_{y-1}^{y+1}\sum_{x-1}^{x+1} i_k(x,y) - i_{k-\Delta k}(x,y). \end{aligned} \tag{2.34}$$

$i_k(x,y)$ denotes the image intensity at the pixel coordinate (x,y) in the frame k and $i_{k-\Delta k}(x,y)$ the image intensity in the frame $k-\Delta k$. The binarized output is noise filtered once more using a median filter. It works similar to the first one: It sums up the amount of pixel-wise detections in a rectangular shaped sliding window of configurable size W, we used $W = 8$. If enough pixel-wise detections are found in the block ($e_k > T_r$), the center pixel r_k is marked as being part of a moving object:

$$r_k(x,y) = \begin{cases} 1 & \text{for } e_k(x,y) \geq T_r \\ 0 & \text{for } e_k(x,y) < T_r \end{cases}, \text{ with}$$
$$e_k(x,y) = \sum_{y-\frac{W}{2}}^{y+\frac{W}{2}} \sum_{x-\frac{W}{2}}^{x+\frac{W}{2}} b_k(x,y). \tag{2.35}$$

2.7 Motion Estimation from Image Sequences

The feature based motion estimation from image sequences is based on the detection and tracking of suitable image regions, so called features. This is done by three steps: First, features are located and selected in the current frame k. Secondly, the selected features are relocated in the next frame $k+1$. The resulting set of displacement vectors, also called optical flow, still contains falsely relocated features which have to be removed in an outlier removal step. The resulting sparse motion vector field can then be used to estimate a dense vector field as input for further processing steps, e.g. a motion compensation.

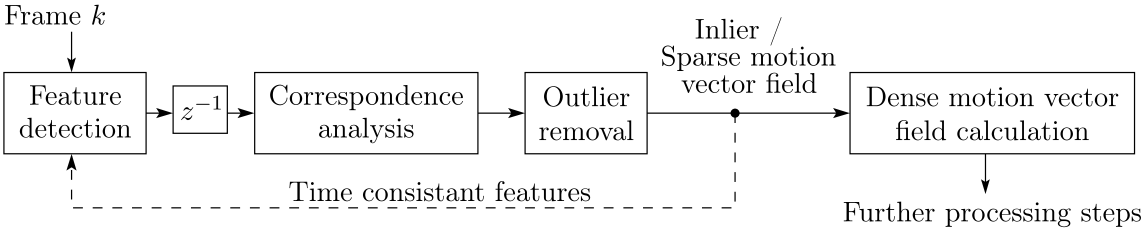


Figure 2.10: Feature tracking workflow used in structure from motion

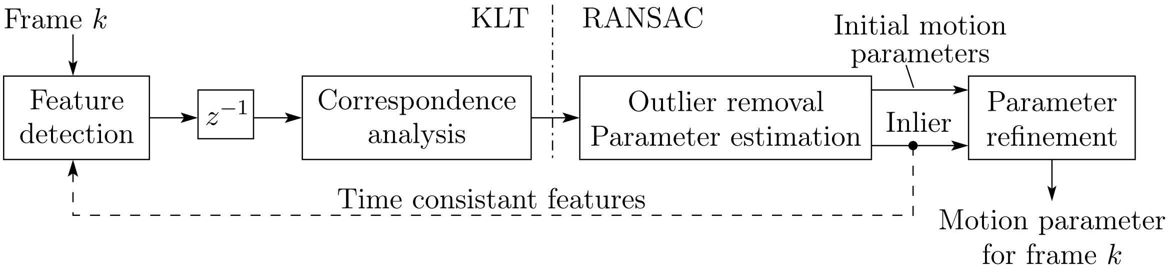


Figure 2.11: Feature tracking workflow used in this work

Two methods of feature detection and tracking are common. The first one, displayed in Figure 2.10, is more common and used in e.g. structure from motion. Hereby, features for the full frame are only located in the first frame. Afterwards, all feature positions passing the outlier removal are re-used for the next frame. Only the amount of removed features is refilled by the feature detection. This allows the tracking of feature positions over a long period of time, which is necessary to e.g. increase the baseline of the camera for structure from motion. The second method, depicted in Figure 2.11, detects the features independently for each frame. Although this does not allow the creation of consistent multi-frame feature tracks, it reduces the drift of the feature locations which occur due to accumulation of small errors resulting from changes in texture of the feature patch caused by perspective, change of shape, occlusion, etc.

2.7.1 Feature Detection

A features is a specific descriptor and position in an image. Whereas the position specifies the exact location, the duty of the associated descriptor is to contain as much information as is necessary to uniquely relocate the feature in a different image even if the description is affected by noise. Common feature types in computer vision are corner features [25], SIFT features [41] and HOG features [15]. The idea behind corner features is that the exact location of a corner is defined by high gradients in two orthogonal directions only. This property allows for a robust locatability even if the image suffers from deformation, e.g. by rotation or perspective distortions, or change in lighting. The descriptor of a corner feature mostly contains the image intensities or gradients in a surrounding window. As for SIFT, the location of a feature is defined by a local extrema in a scale space. As SIFT uses a Difference of Gaussians (DoG) pyramid to locate a feature not only in x- and y-direction but also in scale direction, it is in contrast to corner features invariant to changes in feature size. SIFT uses a 128 dimensional vector containing surrounding image gradient as descriptor. HOG features are similar to a SIFT descriptor, but are computed on an uniformly spaced grid. This work is based on corner features, as a high location precision and good feature distribution is more important than having a large baseline matching ability, as it was shown in [14].

Harris & Stephens / Shi & Tomasi Corner Detector

Harris & Stephens presented in [25] a corner detector which detects corners as gradients in two orthogonal directions. To calculate the image gradient at a position $\mathbf{n} = (x,y)^\top$ of the image, they used the following approximation, with $I(\mathbf{n})$ being the image intensity at position $\mathbf{n}$ and I_x and I_y being the partial derivatives of I:

$$\nabla I(\mathbf{n}) = \begin{pmatrix} I_x(\mathbf{n}) \\ I_y(\mathbf{n}) \end{pmatrix} = \begin{pmatrix} I_x(x,y) \\ I_y(x,y) \end{pmatrix} \approx \begin{pmatrix} \frac{I(x+1,y)-I(x-1,y)}{2} \\ \frac{I(x,y+1)-I(x,y-1)}{2} \end{pmatrix} \tag{2.36}$$

To measure the "cornerness" of an image point **n**, the calculated matrix `M`, containing the multiples of the image gradients surrounding the image point **n**:

$$\mathtt{M}(\mathbf{n}) = \begin{bmatrix} \sum\limits_{\mathbf{n}\in W} I_x^2(\mathbf{n}) & \sum\limits_{\mathbf{n}\in W} I_x(\mathbf{n})I_y(\mathbf{n}) \\ \sum\limits_{\mathbf{n}\in W} I_x(\mathbf{n})I_y(\mathbf{n}) & \sum\limits_{\mathbf{n}\in W} I_y^2(\mathbf{n}) \end{bmatrix}. \tag{2.37}$$

W is the surrounding of **n**, with **n** as its center, in this case a 7×7 block:

$$\sum_{\mathbf{n}\in W} I_x^2(\mathbf{n}) = \sum_{i=-3}^{3}\sum_{j=-3}^{3} I_x^2(x+i,y+j) \tag{2.38}$$

$$\sum_{\mathbf{n}\in W} I_y^2(\mathbf{n}) = \sum_{i=-3}^{3}\sum_{j=-3}^{3} I_y^2(x+i,y+j) \tag{2.39}$$

$$\sum_{\mathbf{n}\in W} I_x(\mathbf{n})I_y(\mathbf{n}) = \sum_{i=-3}^{3}\sum_{j=-3}^{3} I_x(x+i,y+j)I_y(x+i,y+j). \tag{2.40}$$

Evaluating the eigenvalues λ_1 and λ_2 of $\mathtt{M}(\mathbf{n})$ now give information on the structure of **n** (and its surrounding W):

$$\begin{array}{ll} \lambda_1 \approx \lambda_2 \approx 0 & : \text{no structure} \\ \lambda_1 \approx 0, \lambda_2 \gg 0 & : \text{edge} \\ \lambda_1 \gg 0, \lambda 2 \approx 0 & : \text{edge} \\ \lambda_1 \wedge \lambda_2 \gg 0 & : \text{corner} \end{array} \tag{2.41}$$

High values for λ_1 and λ_2 indicate a corner. This can be checked by the determinant of $\mathtt{M}(\mathbf{n})$:

$$\det \mathtt{M}(\mathbf{n}) = \lambda_1\lambda_2 \tag{2.42}$$

Harris & Stephens added an additional term to the determinant to reduce the influence of edgy structures and called it the corner response function (CRF):

$$\begin{aligned} CRF(\mathbf{n}) &= \det \mathtt{M}(\mathbf{n}) - k_H(\operatorname{trace} \mathtt{M}(\mathbf{n}))^2 \\ &= \lambda_1\lambda_2 - k_H(\lambda_1 + \lambda_2)^2, \end{aligned} \tag{2.43}$$

where k_H was proposed as 0.04 by the authors. In [56], Shi and Thomasi replaced the determinant and the trace by a more simple to calculate $\min(\lambda_1,\lambda_2)$ operation.

To have stable and equally distributed feature points, the CRF values of all image points are sorted in descending order. The point with the highest CRF is selected as a feature point and removed from the list. After that, all feature points in a local surrounding of size d_f of the removed feature are also removed to prevent clustering of features at single spots but achieve a uniform distribution instead. This is repeated until the desired amount of feature points is reached or the CRF values fall below a threshold.

2.7.2 Correspondence Analysis

To track the selected feature points over several images k and thereby get the motion of the specific image regions, the feature points are represented by a descriptor additionally to their position in the image. In the Kanade-Lucas-Tomasi feature tracker [43, 42], a window W of size $N \times N$ of image intensities around the feature position $\mathbf{n}$ is used to identify the feature. The main idea behind the Lukas-Kanade method is that the displacement of pixels $\mathbf{d} = (d_x,d_y)^\top$ inside a small image region can be assumed as constant and only translational, as long as the size of the region and the displacement between the frames is small enough. The same assumption is also made in video coding. In the case of optical flow [32], this assumption translates to the optical flow equation (here given for the 2D case) being valid for all pixel inside the feature window (block) W:

$$I_k(\mathbf{n}+\mathbf{d}) = I_{k-1}(\mathbf{n}). \tag{2.44}$$

To determine the displacement $\mathbf{d}$ of a feature point $\mathbf{n}$ between the frames $k-1$ and k, the sum of squared differences (SSD) of image intensities inside the block W has to be minimized over all possible $\mathbf{d}$:

$$\epsilon_d(\mathbf{d}) = \sum_{\mathbf{n}\in W} (I_k(\mathbf{n}+\mathbf{d}) - I_{k\text{-}1}(\mathbf{n}))^2. \tag{2.45}$$

This leads to an estimated value $\widehat{\mathbf{d}}$ of $\mathbf{d}$:

$$\widehat{\mathbf{d}} = \arg\min_d(\epsilon_d(\mathbf{d})). \tag{2.46}$$

Instead testing all possible $\mathbf{d}$, $\widehat{\mathbf{d}}$ can also be estimated using the local image intensity derivations. To get sub-pel accuracy, the image signal is assumed as being continuous:

$$I_k(\mathbf{n}+\mathbf{d}) \approx I_{k-1}(\mathbf{n}) + \nabla^\top I_{k-1}(\mathbf{n})\mathbf{d} + \frac{\delta I_{k-1}(\mathbf{n})}{\delta t} t_k. \tag{2.47}$$

The time derivative of the image signal can be approximated by ([7]):

$$\frac{\delta I_{k-1}(\mathbf{n})}{\delta t} \approx \frac{I_k(\mathbf{n}) - I_{k-1}(\mathbf{n})}{t_k}. \tag{2.48}$$

This leads to the continuous simplified cost function:

$$\epsilon_d(\mathbf{d}) = \sum_{\mathbf{n}\in W} (\nabla^\top I_{k-1}(\mathbf{n})\mathbf{d} + I_k(\mathbf{n}) - I_{k-1}(\mathbf{n}))^2. \tag{2.49}$$

To solve the minimization problem, the extrema of Equation 2.49 have to be found by partially differentiating with respect to dx and dy and setting it to zero. This leads to a linear equation system

$$\mathtt{M}_{k-1}\widehat{\mathbf{d}} = \mathbf{g}_{k-1}, \tag{2.50}$$

where, analog to Equation 2.36 and Equation 2.37,

$$\begin{aligned}\mathtt{M}_{k-1} &= \sum_{\mathbf{n}\in W} \nabla I_{k-1}(\mathbf{n})\nabla^{\top} I_{k-1}(\mathbf{n}) \\ &= \begin{bmatrix} \sum_{\mathbf{n}\in W} I_{x\,k-1}^{2}(\mathbf{n}) & \sum_{\mathbf{n}\in W} I_{x\,k-1}(\mathbf{n}) I_{y_{k-1}}(\mathbf{n}) \\ \sum_{\mathbf{n}\in W} I_{x\,k-1}(\mathbf{n}) I_{y_{k-1}}(\mathbf{n}) & \sum_{\mathbf{n}\in W} I_{y\,k-1}^{2}(\mathbf{n}) \end{bmatrix} \end{aligned} \tag{2.51}$$

$$\begin{aligned}\mathbf{g}_{k-1} &= -\sum_{\mathbf{n}\in W} \left(I_k(\mathbf{n}) - I_{k-1}(\mathbf{n})\right) \nabla I_{k-1}(\mathbf{n}) \\ &= -\sum_{\mathbf{n}\in W} \left(I_k(\mathbf{n}) - I_{k-1}(\mathbf{n})\right) \begin{pmatrix} I_{x\,k-1}(\mathbf{n}) \\ I_{y_{k-1}}(\mathbf{n}) \end{pmatrix}. \end{aligned} \tag{2.52}$$

M_{k-1} contains the spatial derivatives of the image intensity function and $\mathbf{g}_{k-1}$ holds the temporal derivatives. To get the estimated displacement value, Equation 2.50 can be solved for $\widehat{\mathbf{d}}$:

$$\widehat{\mathbf{d}} = \mathtt{M}_{\mathtt{k-1}}^{-1} \mathbf{g}_{k-1}, \tag{2.53}$$

Due to the approximations being made, this equation is only valid for very small displacements $\mathbf{d}$. Therefore, in [42] the equations are solved iteratively using the Newton-Raphson method, whereas the next estimate of displacement $\widehat{\mathbf{d}}_{\mathbf{i+1}}$ is calculated from the previous iteration as follows:

$$\widehat{\mathbf{d}}_{\mathbf{i+1}} = \widehat{\mathbf{d}}_{\mathbf{i}} + \mathtt{M}_{\mathtt{k-1}}^{-1} \sum_{\mathbf{n}\in W} \Big((I_{k-1}(\mathbf{n}) - I_k(\mathbf{n} + \widehat{\mathbf{d}}_{\mathbf{i}})) \nabla I_{k-1}(\mathbf{n}) \Big). \tag{2.54}$$

To get the unknown image intensity I_k at the sub-pel coordinate $\mathbf{n} + \widehat{\mathbf{d}}_{\mathbf{i}}$, it has to be estimated from known sampled positions. Typically, a simple bilinear filter is employed for this using the 4 adjacent sampling positions. The iteration is stopped, if the change in $\mathbf{d}$ for the next iteration is smaller than a threshold, commonly a value of 0.01 pel, or the maximum amount of iterations is reached.

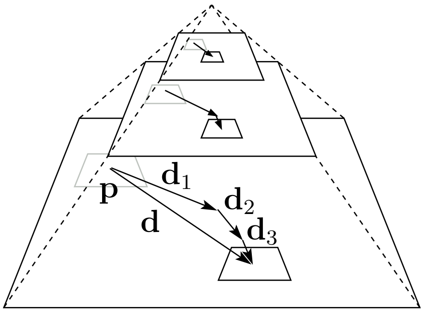


Figure 2.12: Image pyramid to handle large displacements

If the displacement between frames gets too large, the Newton-Raphson method tends to get stuck in local minima. To handle this and to reliably estimate large displacements between frames, a hierarchical search is incorporated. To do so, an image resolution pyramid is created by low-pass filtering and down sampling the intensity image several times, depending on the size of the original image and the maximum displacement that is likely to occur. The search is started in the lowest resolution stage and is refined in the higher resolution stages by going through the iteration process (Equation 2.54) in each stage. The feature position $\mathbf{n}$ and displacement $\mathbf{d}$ is appropriately scaled to each pyramid stage (see Figure 2.12). To be able to better handle deformation of objects on large displacement, the translational motion model can be replaced by a full affine model as proposed by [56], although it is less stable due to the higher number of degrees of freedom.

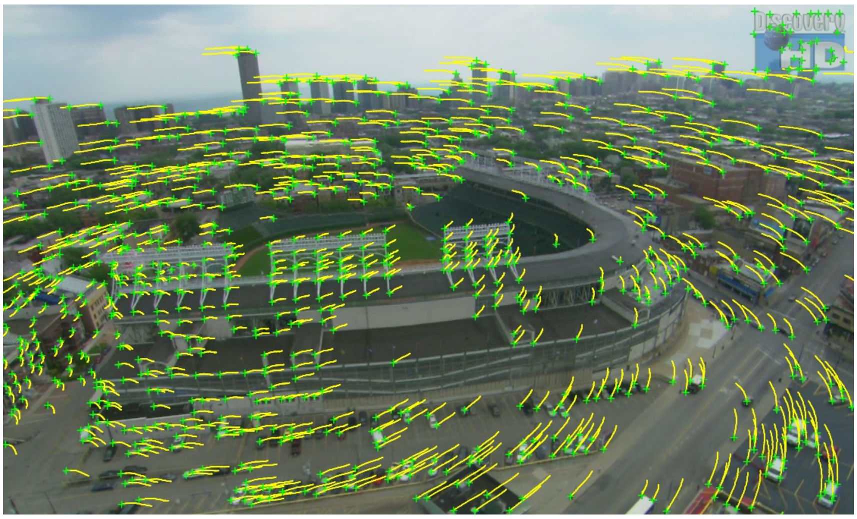


Figure 2.13: Harris features (green crosses) and KLT tracking (yellow stripes). Frame of the Chicago test sequence [62].

2.7.3 Outlier Removal: Random Sample Consensus

To calculate robust `H`-, `E`-, or `F` matrices, it has to be taken into account that the measured feature points and correspondences are affected by noise. Moreover, false correspondences or non-static image regions are common and have a great effect on the final result [22]. To get a robust mapping estimation in the presence of noise, two steps have to be made. First, falsely tracked correspondences (outliers) and correspondences describing the movement of local objects have to be removed from the input data. For this, **Ran**dom **Sa**mple **C**onsensus [22] (in short RANSAC) is suitable. It uses a minimal set of random correspondences to calculate the desired mapping model (e.g. a homography). All remaining correspondences are then tested for conformance by evaluating the distance of the calculated model mapping positions and the measured positions. If a correspondence is too far away from the calculated mapping position ($\epsilon > \epsilon_{\max}$), it is considered an outlier; otherwise it is kept as an inlier (Figure 2.14). These steps are repeated until the mean squared mapping error (MSE) of all inliers is minimal and the amount of inliers is maximal. Commonly, the number of RANSAC runs is limited to a reasonable amount, and a MSE threshold abort criterion is set to keep the run times low. After having calculated this first coarse mapping, it has to be refined and the noise has to be eliminated. This is done in a second step by minimizing the sum of squared Euclidean distances using the least squares approach and only the inliers as input [65].

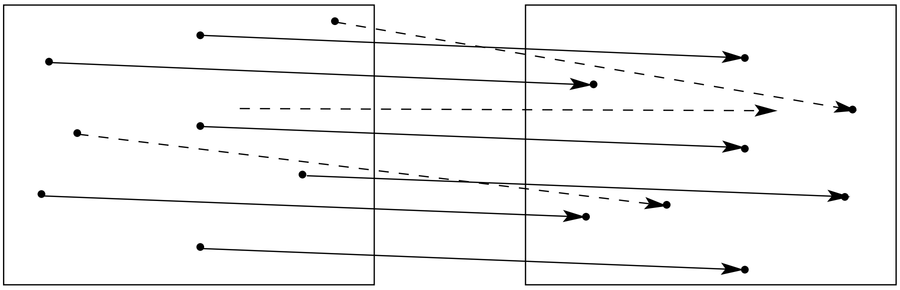


Figure 2.14: RANSAC outlier removal. Dashed correspondences are removed as they do not confirm to majority determined model (solid arrows).

2.8 Dense Optical Flow

Using the methods above, only a sparse sampling of the optical flow is known so far. This is due to the correspondences being only determined for the robust feature coordinates selected by the Shi & Tomasi corner detector. To be able to compensate the motion of the full image, a pixel-wise dense optical flow must either be estimated globally or it might be interpolated from the already known correspondences. Algorithms for a global solution have to deal with unstructured image regions, occlusions, or blurring by themselves. Most of them are based on energy minimization, as first presented by Horn and Schunck in [33]. These methods can nicely estimate a smooth and dense optical flow field, but the smoothing term also smooths out the border regions at motion discontinuities. Gradient-based local methods like Lucas-Kanade [43] and the structure tensor approach of Bigün and Granlund [5] are better at motion discontinuities but fail to determine the correct flow for large displacements or in unstructured regions. Combined methods like [55, 70, 8] try to merge the global and local approaches to have a discontinuity-preserving dense optical flow. However, all of these global methods have in common that they are relatively complex to calculate and therefore hard to implement e.g. into a small energy constrained aerial vehicle.

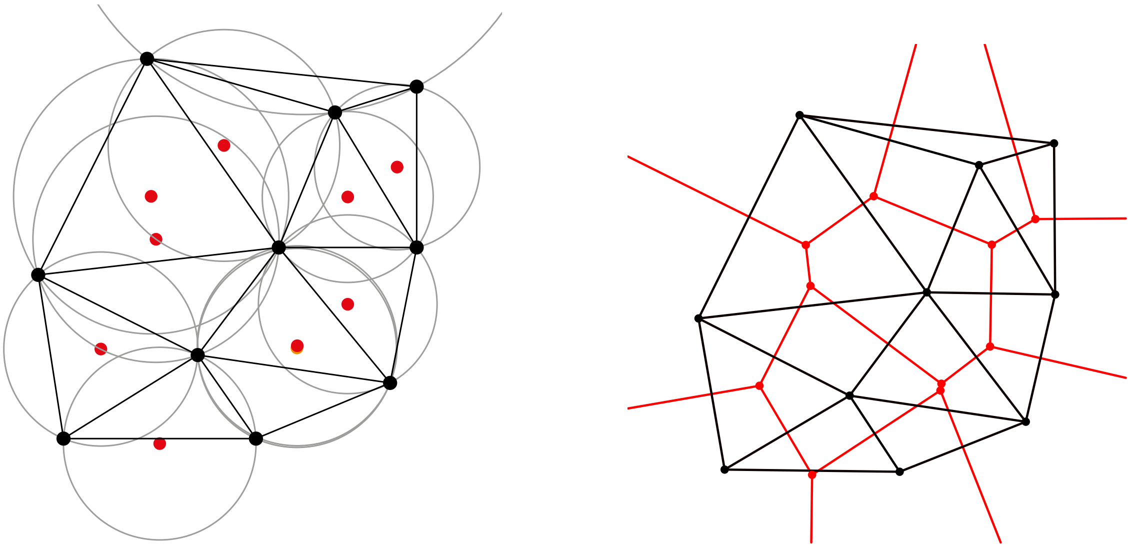
Concerning the interpolation of a dense field from sparse correspondences, methods like Markov random fields [29] or the method of Gibson and Spann [23] are able to fill in the gaps while removing outliers at the same time. However, if evaluating the underlying image intensities or gradients is involved in the interpolation process, these methods have to deal with the same run-time problem as the global solutions. A fast way of interpolating the missing information is to fit a simplified model of the scene into the sparse set of known correspondences an then extract the missing information from the model. In the case of aerial surveillance where the observed ground plane of the earth is mostly planar, a homography or even more restricted transformations are state of the art as they are easy and fast to compute and very robust against outliers. However, those models are often too simple to be used in a complex structured urban environment and, as e.g. with the homography model, fail in the presence of non-planar objects like trees.

In this work, a multi-planar mesh-based approach is presented instead. It uses a large set of small triangular patches which are connected in a mesh. The area inside each patch is described as affine. Due to the individual transformation of each patch, the multi-planar approach is able to follow non-planar scene structure while, by using simple linear equations, keeping the computational complexity low at the same time. The mesh is created by triangulating the known correspondences using a Delaunay triangulation. The affine transformations for the triangular patches are determined from the feature positions of the three correspondences defining each patch.

2.8.1 Delaunay Triangulation

The Delaunay triangulation is a method of converting a set of points on a plane into a continues, non-overlapping mesh of triangles. It is defined by the property that no other mesh node can exist in a circumcircle drawn through the three nodes of each triangle. Doing so, the Delaunay triangulation has the ability to maximize the minimum angle of all triangles whereby thin and pointy triangles are avoided. The Delaunay triangulation is related to the Voronoi diagram such that the dual graph of the Delaunay triangulation is a Voronoi diagram (connecting the centers of all circumcircles, see Figure 2.15).

A common method of creating a Delaunay triangulation is the flip algorithm. It is initialized by an arbitrary triangle mesh wherein the circumcircle constraint does not need to be fulfilled. The flip algorithm starts by testing the circumcircle constraint on a random pair of connected triangles. If the circumcircle constraint is



(a) Delaunay triangulation (black) and the circumcircles with their centers (red dots)

(b) Voronoi diagram (red lines) as the conection of the center points of the circumcircles

Figure 2.15: Delaunay triangulation [37]

fulfilled by both triangles, the next pair is chosen. If it is violated by one of the triangles, the common edge between the triangles is flipped (see Figure 2.16). As the flipping operation may destroy the circumcircle constraint for neighboring triangles, they have to be retested afterwards. Faster algorithms are the Bowyer–Watson incremental calculation method [6][69] which adds one point after another to the mesh and is also suitable for more than two dimensions, or the Dwyer divide-and-conquer approach [18]. Hereby the point cloud is divided into two single Delaunay triangulations, which are, after solving them separately, merged together (conquer step).

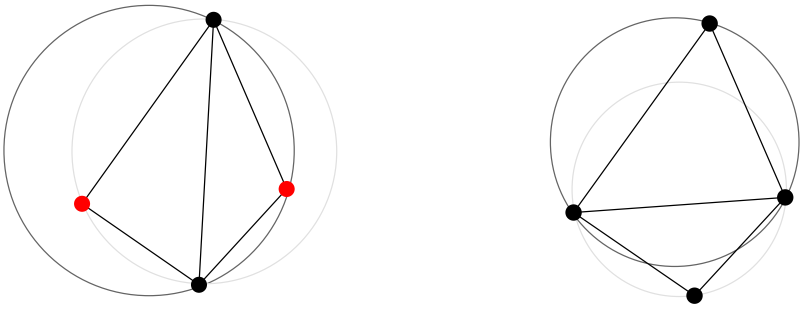


Figure 2.16: The Delaunay flip operation. The red points are inside the circumcircles of the triangles. By flipping the common edge, all other points are outside the circumcircles. [37]

3 Planar Landscape Error Model

In this chapter, the planar landscape model commonly used in aerial surveillance is described and the aberrations to reality are analyzed for different observation scenarios. The surface of the earth is in this model represented by a single plane, so the mapping of projected ground objects between the airborne camera views can be described by a projective mapping model using a homography, as introduced in Chapter 2.5. This is a state-of-the-art global motion model in aerial surveillance systems, as it is easy to compute, robust against outliers, and most of the time accurate enough for the desired tasks.

However, in reality the surface of the earth is not planar. To get an impression of the model aberration in the presence of non-planar objects like buildings or trees, or even the curvature of the earth, we derive in this chapter the absolute mapping error of projected pixels due to violation of the non-planar assumption. Only the projective mapping is considered; lens distortions are neglected, as they are rectifiable in the recording step (Chapter 2.2.2).

The chapter starts by adapting and simplifying the general scene model introduced in Chapter 2.1 for the aerial surveillance scenario. Following this, a first error model is introduced by assuming the camera to point directly downwards. In this case the image plane is parallel to the surface of the earth. After having derived the error model, it is extended for cameras having a tilt angle θ as well as arbitrary viewing directions, e.g. looking sideways. For all cases, the mapping error in pel on the image plane is determined as well as the associated displacement vector on the ground plane in meter. Finally, we will calculate the motion displacement of pixels from projected non-planar objects after motion compensation of the background motion as a result of the camera movement and parallax effects, and create a classifier for the differentiation of the motion displacement from parallax effects of static objects and the displacement of individual moving ground objects.

3.1 Aerial Surveillance Scene Model

First we take the general scene model from Chapter 2.1 and apply it to an aerial surveillance scenario (Figure 3.1). The position and orientation of the local camera coordinate system are given by the center of projection $\mathbf{C}$ of the camera in world coordinates and the rotation matrix $\mathtt{R}$, containing the roll, pan, and tilt angles γ,

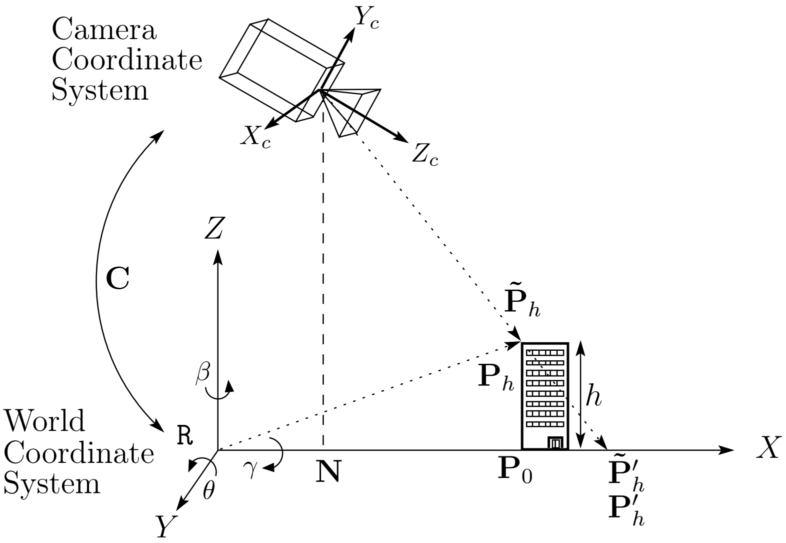


Figure 3.1: Aerial surveillance scene model

β, and θ. The nadir point of the camera center on the ground plane is labeled $\mathbf{N} = (C_x,C_y,0)^\top$. A general point in the scene is referred to as $\mathbf{P} = (X,Y,Z)^\top$. Moreover, three special points are considered: a point $\mathbf{P}_h = (X,Y,h)^\top$ at height h above the ground plane, the associated nadir point $\mathbf{P}_0 = (X,Y,0)^\top$ and the point $\mathbf{P}'_h = (X',Y',0)^\top$, which is the intercept point of a line connecting the camera center and the point $\mathbf{P}_h$ with the ground plane. Transformed to the camera coordinate system, $\mathbf{P}_h$ becomes $\tilde{\mathbf{P}}_h = (X_c,Y_c,Z_c)^\top$, $\mathbf{P}_0$ becomes $\tilde{\mathbf{P}}_0$ and $\mathbf{P}'_h$ becomes $\tilde{\mathbf{P}}'_h$. The projections onto the image plane are denoted as $\mathbf{p}$ and $\mathbf{p}'$, respectively.

To simplify the observations, Figure 3.2 gives a 2D cut of the 3D scene. The gray area represents the field of view (FOV) of the camera and is dependent on the angle of aperture α of the camera lens. It can be calculated from the focal length and the sensor dimensions (see Chapter 2.2.3). The effective viewing range on the ground plane is in this case additionally effected by the height of the camera center C_z and the tilt angle θ, such that $\theta_{\min} = \theta - \frac{\alpha}{2}$ and $\theta_{\max} = \theta + \frac{\alpha}{2}$. This leads to the visible area between $X_{\min}$ and $X_{\max}$ on the ground plane:

$$X_{\min} = C_x + C_z \cdot \tan(\theta_{\min}) = C_x + C_z \cdot \tan\left(\theta - \frac{\alpha}{2}\right) \tag{3.1}$$

$$X_{\max} = C_x + C_z \cdot \tan(\theta_{\max}) = C_x + C_z \cdot \tan\left(\theta + \frac{\alpha}{2}\right) \tag{3.2}$$

From the amount of pixels per visible area follows the size of a pixel on the ground plane, which is equal to the ground resolution of the given camera setup. It calculates from the size of a sensor element projected to the ground plane (here given for the

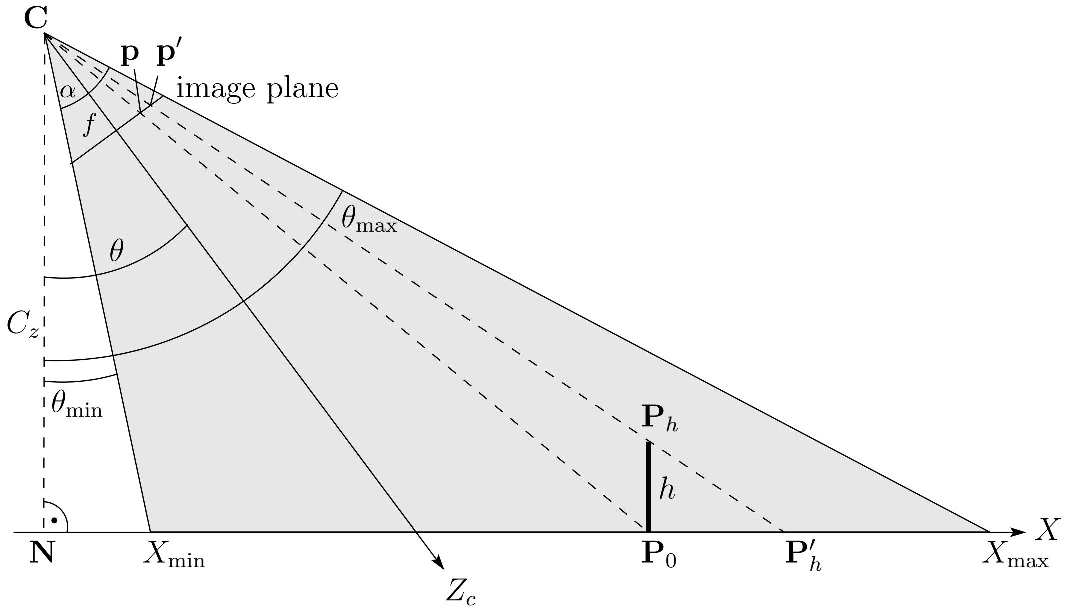


Figure 3.2: Mapping error caused by non-planar landscape

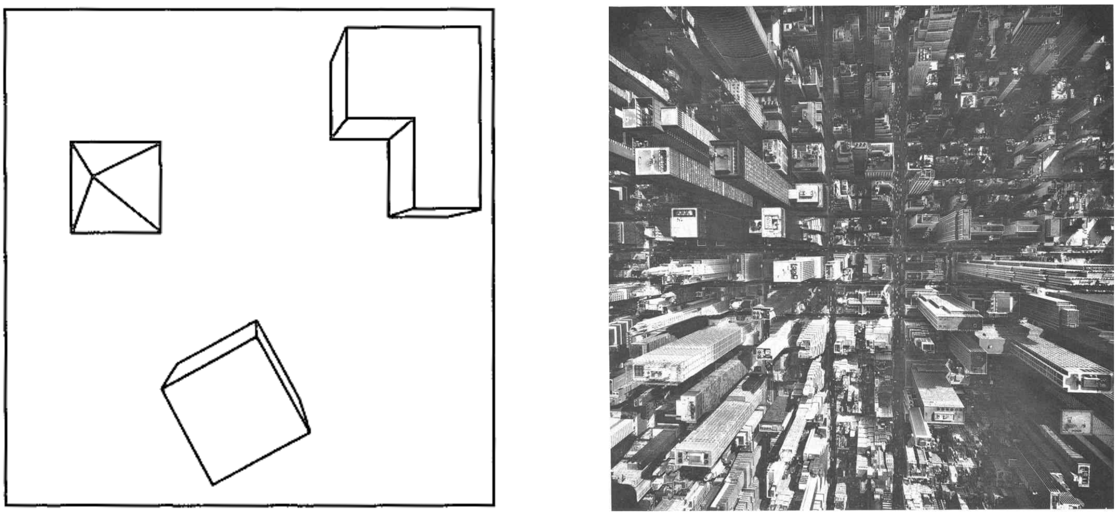
size in x-direction):

$$S_x = \frac{C_z \cdot s_x}{f \cdot \cos\left(\theta \pm \frac{\alpha}{2}\right)}. \tag{3.3}$$

To give an example, assuming a vertical aerial photo of a HD-camera (1920 × 1080 pel) with a full-frame sensor (36 × 20 mm) and a focal length of $f = 80$ mm. With a flight altitude of $C_z = 500$ m, this leads to a visible area on the ground plane of 225 m and a pixel size of $S_x = 11.7$ cm. Changes of the focal length as well as the flight altitude hereby have a linear effect on both the visible area and the pixel size.

To demonstrate the mapping error, we assume two sample points of a building standing on the ground plane: one point $\mathbf{P}_0 = (X,Y,0)^\top$ at the foot of the building and a point $\mathbf{P}_h = (X,Y,h)^\top$ at the roof at the height h. One can see that, although a point $\mathbf{p}$ on the image plane only maps to one point $\mathbf{P}_0$ in the scene, the mapping of a point $\mathbf{p}'$ on the image plane is ambiguous, as it may be projected from two scene points: the point $\mathbf{P}_h$ on the roof of the building or the point $\mathbf{P}'_h = (X',Y',0)^\top$ placed on the ground plane. Considered the other way, there is no unique mapping possible in the case of model violations. This leads to perspective aberrations, which is illustrated in Figure 3.3.

Additional aberrations occur due to the curvature of the earth as well as atmospheric refractions [53]. In normal remote sensing scenarios, these errors are relatively small compared to the error caused by non-planar ground objects. Moreover, as these errors are only dependent on the recording scenario but not on the observed scene content, they can easily be compensated [30] and are therefore not further examined.



(a) Radial displacement of non-planar objects (from [1])

(b) Vertical aerial photo of Manhattan, New York (from [30])

Figure 3.3: Radial distortion due to perspective projection

3.2 Vertical Aerial Photo

In a vertical aerial photo setup as shown in Figure 3.4, objects sticking out of the ground plane are mapped to the image plane with a displacement such that the point $\tilde{\mathbf{P}}_h$ seems to be identical to the point $\tilde{\mathbf{P}}'_h$. The displacement is dependent on the height of the object from the ground plane h as well as the distance of the object from the nadir point X_c. Objects with a positive height are mapped with an increasing distance, negative heights lead to a mapping towards the nadir point. The displacement is called relief displacement and, by assuming a point $\tilde{\mathbf{P}}_{h,0} = (0,0,C_z - h)^\top$ as the projection of $\tilde{\mathbf{P}}_h$ to the nadir, can be determined by the theorem about ratios in similar triangles, or in short similarity theorem. It states that if the triangle $\tilde{\mathbf{P}}_h\tilde{\mathbf{P}}'_h\tilde{\mathbf{P}}_\mathbf{0}$ is similar to the triangle $\mathbf{C}\tilde{\mathbf{P}}'_h\mathbf{N}$ as well as the triangle $\mathbf{C}\tilde{\mathbf{P}}_h\tilde{\mathbf{P}}_{h,0}$, the following applies:

$$\frac{\Delta X_c}{h} = \frac{X'_c}{C_z} = \frac{X_c}{C_z - h}. \tag{3.4}$$

This can be rearranged to get the displacement on the ground plane ΔX_c:

$$\Delta X_c = h\,\frac{X'_c}{C_z} = h\,\frac{X_c}{C_z - h}. \tag{3.5}$$

Note that the displacement is expressed as a function of the virtual ground position X'_c. By applying the intercept theorem, this can be expressed as the displacement

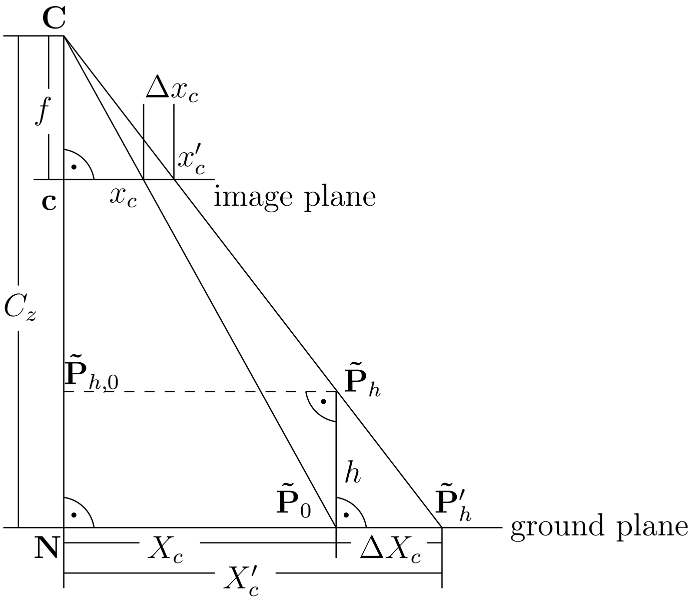


Figure 3.4: Mapping error (ideal)

on the image plane instead:

$$\Delta x_c = h\,\frac{x'_c}{C_z} = h\,\frac{x_c}{C_z - h}, \tag{3.6}$$

whereby x_c and x'_c are the projections from X_c and X'_c onto the image plane. To get the model aberrations, we have to express the displacement as a function of h and the real ground position X_c. This can be achieved by looking at the intercept theorem again:

$$\frac{x_c}{X_c} = \frac{x'_c}{X'_c} = \frac{\Delta x_c}{\Delta X_c} = \frac{f}{C_z}, \tag{3.7}$$

and by introducing a virtual plane parallel to the ground plane at object height h, this results in:

$$\frac{x'_c}{X_c} = \frac{f}{C_z - h} \quad \text{and} \tag{3.8}$$

$$\frac{X'_c}{X_c} = \frac{C_z}{C_z - h}. \tag{3.9}$$

Equations Equation 3.7 and Equation 3.8 can be rearranged to describe the projected positions on the image plane of an object on the ground plane at position X_c and having the height h:

$$x_c = f \cdot \frac{X_c}{C_z}, \quad x'_c = f \cdot \frac{X_c}{C_z - h}, \tag{3.10}$$

with C_z being the flight altitude (more precisely the distance of the camera center from the ground plane). Finally, the difference of both projected points gives us the mapping error Δx_c on the image plane:

$$\Delta x_c = x_c' - x_c = \frac{f\, X_c\, h}{(C_z - h)\, C_z}. \tag{3.11}$$

By considering Equation 3.9, we get the mapping error in meter on the ground plane:

$$\Delta X_c = X_c' - X_c = X_c \frac{h}{C_z - h} \tag{3.12}$$

As the distortion for the vertical photo case are equal in x- and y-direction, the x-dimension can simply be substituted by a radius, which is the also most common notation in the literature:

$$\Delta r_c = r_c' - r_c = \frac{f\, R_c\, h}{C_z\,(C_z - h)}, \tag{3.13}$$

$$\Delta R_c = R_c' - R_c = R_c \frac{h}{C_z - h}. \tag{3.14}$$

In vector notation this correlates to the following form:

$$\Delta \mathbf{p} = \mathbf{p}' - \mathbf{p} = \tilde{\mathbf{P}}_0 \frac{f \cdot h}{C_z\,(C_z - h)}, \tag{3.15}$$

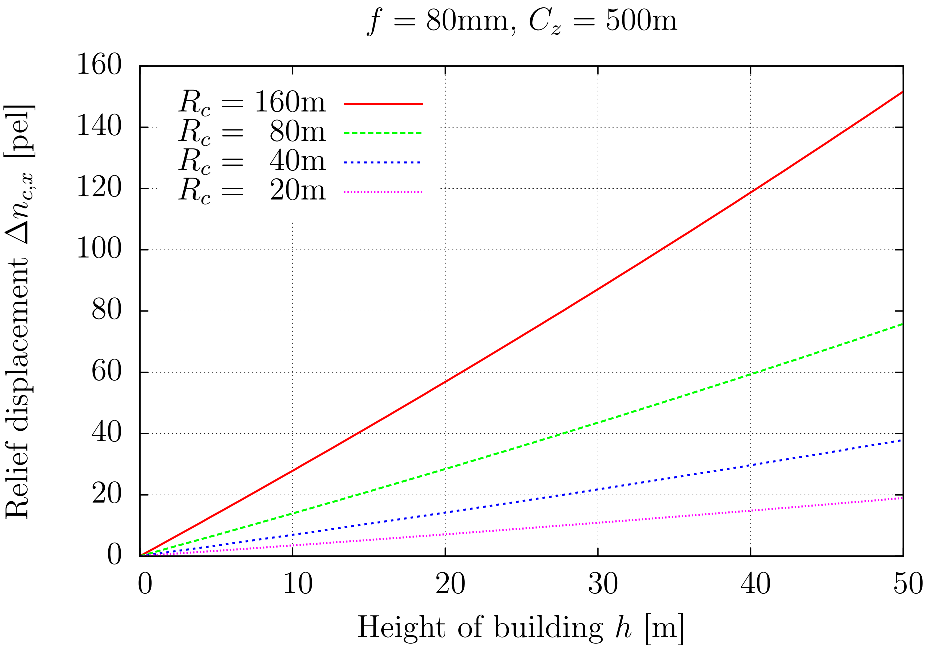
$$\Delta \mathbf{P} = \tilde{\mathbf{P}}_h' - \tilde{\mathbf{P}}_0 = \tilde{\mathbf{P}}_0 \frac{h}{C_z - h}. \tag{3.16}$$

Δr_c and $\Delta \mathbf{p}$ are specified in meter on the image plane. To get image coordinates in pel, we have to take the scaling factors from the sensor model in Equation 2.1 into account:

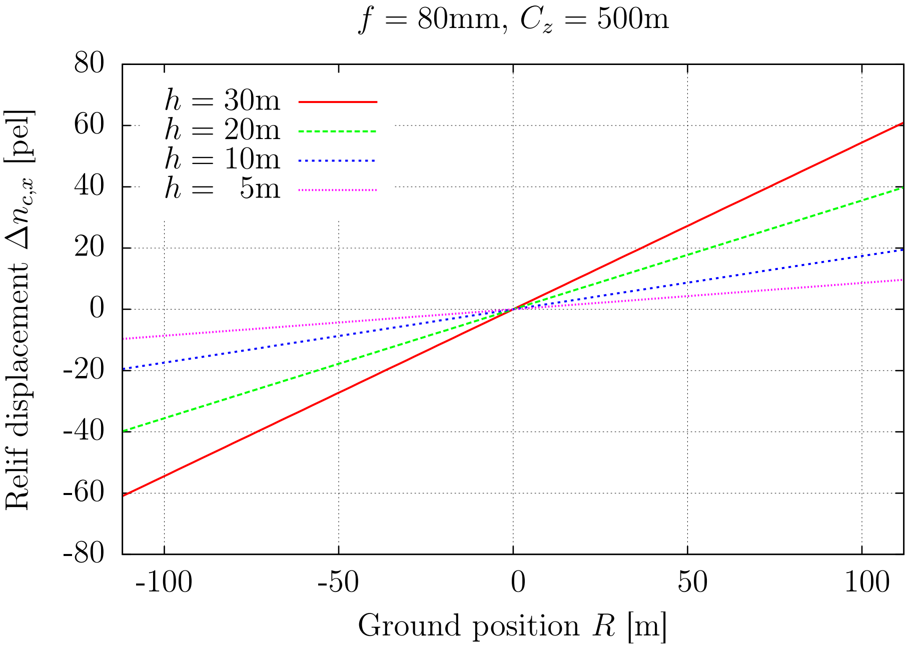
$$\Delta n_c = \frac{\Delta r_c}{s_x} = \frac{N_x}{s_w} \frac{f\, R_c\, h}{(C_z - h)\, C_z} \tag{3.17}$$

Figure 3.5 shows exemplary relief displacements Δn_c in pel for different heights of objects h, positions of the objects on ground plane R_c, aircraft altitudes C_z, and focal lengths f. A full-HD camera sensor with 1920×1080 elements on a full frame sensor is assumed. It can be seen that every variable has a linear effect on the mapping error, except for the aircraft altitude C_z, which influence decreases nearly quadratic with increasing altitude. In this realistic scenario, the relief displacement reaches quite high displacements in the range of 100 pel for small tower buildings of 30m height and flight height of 500 m. For low flying drones, the displacement is growing largely to 600 pel and more due to the non-linear behavior.

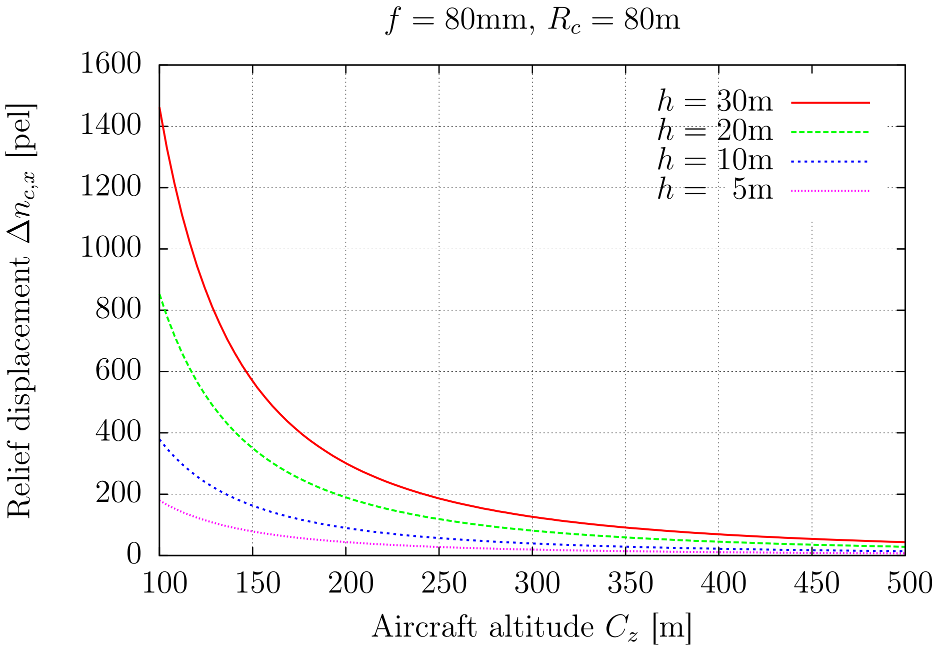
Figure 3.5: Resulting mapping error for a camera pointing downwards



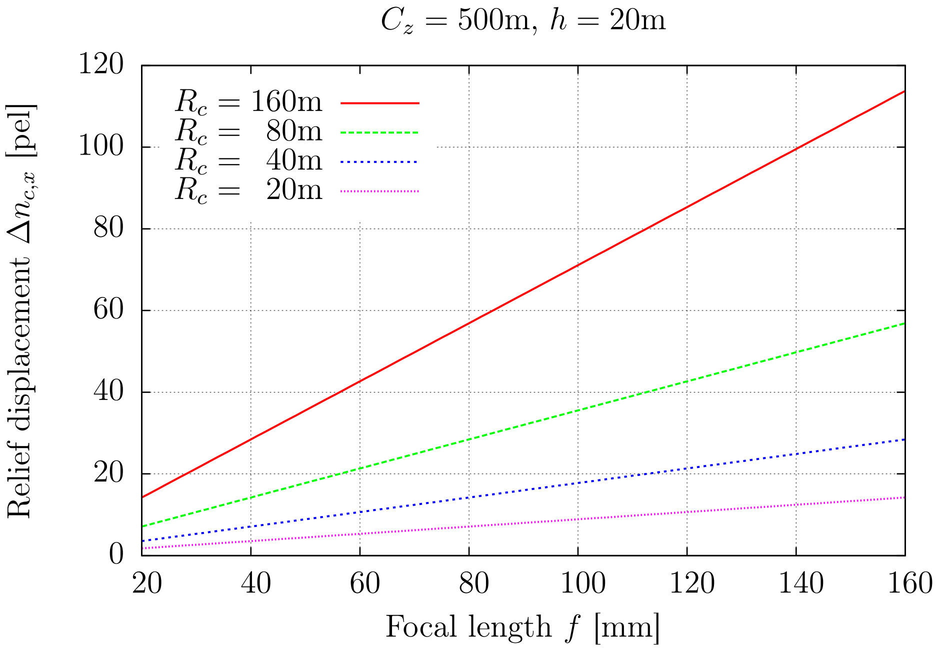
(a) Mapping error vs. building height



(b) Mapping error vs. ground position



(c) Mapping error vs. aircraft altitude



(d) Mapping error vs. focal length

3.3 Motion Parallax Displacement

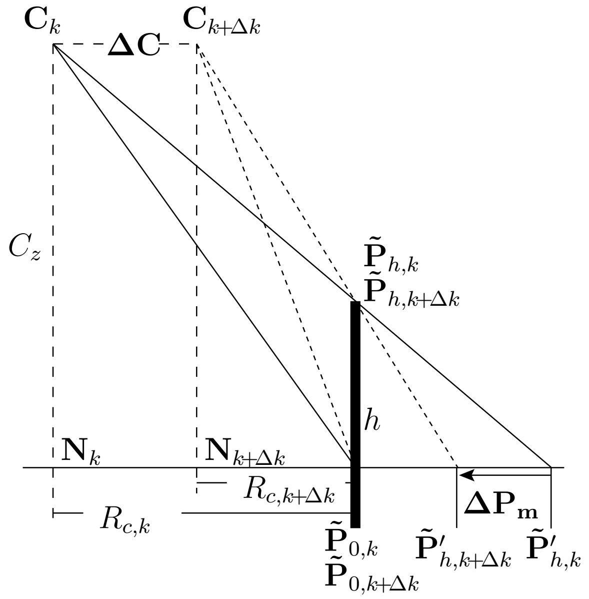
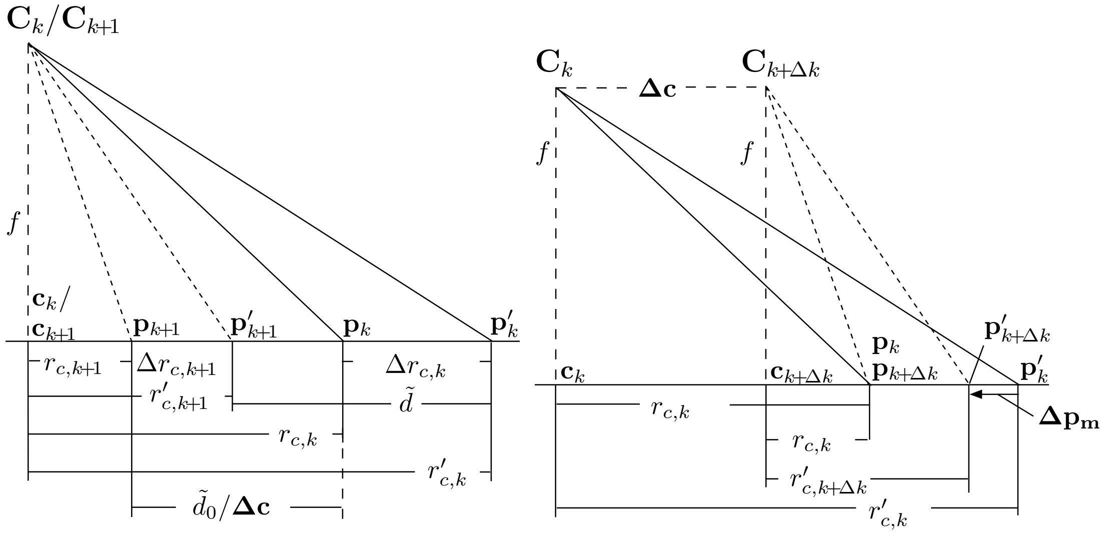

$\mathbf{C}_k$
$\mathbf{C}_{k+\Delta k}$
$\Delta\mathbf{C}$
C_z
$\tilde{\mathbf{P}}_{h,k}$
$\tilde{\mathbf{P}}_{h,k+\Delta k}$
h
$\mathbf{N}_k$
$\mathbf{N}_{k+\Delta k}$
$R_{c,k+\Delta k}$
$R_{c,k}$
$\Delta\mathbf{P}_\mathbf{m}$
$\tilde{\mathbf{P}}_{0,k}$
$\tilde{\mathbf{P}}_{0,k+\Delta k}$
$\tilde{\mathbf{P}}'_{h,k+\Delta k}$
$\tilde{\mathbf{P}}'_{h,k}$

Figure 3.6: Change in relief displacement due to different camera positions

The previous examination was only done for the static case (a single vertical photo taken at one camera position). However, if multiple pictures are made from nearly the same scene while the aircraft has moved on, the relief displacement of a specific object point with non-zero height h changes between the views as the mapping error is a function of the ground position $\tilde{\mathbf{P}}_0$, which changed with the nadir. Therefore $\tilde{\mathbf{P}}_0$ and all associated variables are further specified with an index k, referring to the referenced nadir $\mathbf{N}_k$ as the origin. It corresponds to the frame index.

The effect of the relief displacement on the image plane being dependent on the distance of the projected scene point to the moving camera center is called image parallax. Besides leading to image distortions, the effect can be used to e.g. determine the distance of a scene point $\tilde{\mathbf{P}}$ to the camera by measuring the parallax $\tilde{d}$ of the projected points $\mathbf{p}_k$ and $\mathbf{p}_{k+\Delta k}$ in the two image planes (see Figure 3.7(a)). With a known reference (e.g. the parallax of the ground plane $\tilde{d}_0$), it can also be used to determine the height of objects.

However, in aerial ground moving object detection systems, moving objects are detected by a different displacement of projected object points compared to the background motion. Hereby, the parallax effect is harmful as the displacement of objects between the frames has not arisen from actual motion of ground objects but from the parallax effect and a non-zero object height compared to the background. In this section we derive the displacements of object points due to the parallax effect and compare them to real motion of objects on the ground plane.



(a) Alignment of camera centers (b) Alignment of ground plane

Figure 3.7: Superposition of image planes

We start with the distance $\mathbf{\Delta C}$ the aircraft has traveled between two consecutive frames:

$$\mathbf{\Delta C} = \frac{\mathbf{v}_{\text{plane}}}{r_{\text{fps}}}, \tag{3.18}$$

with $\mathbf{v}_{\text{plane}}$ being the velocity and flight direction of the aircraft and r_{fps} being the number of recorded frames per second (frame rate). As the observing camera is hard-fixed on the airplane, this displacement also expresses the new coordinates of the camera center from the old position:

$$\mathbf{C}_{k+\Delta k} = \mathbf{C}_k + \mathbf{\Delta C}, \tag{3.19}$$

wherein k is the index of the photo, or the video frame respectively. In the vertical photo case, the displacement of the camera might also be expressed as a translation of all points on the ground plane (see Figure 3.6):

$$\tilde{\mathbf{P}}_{0,k+\Delta k} = \tilde{\mathbf{P}}_{0,k} + \mathbf{\Delta C}. \tag{3.20}$$

According to Equation 3.7, this translates to the image plane as:

$$\mathbf{p}_{k+\Delta k} = \mathbf{p}_k + \mathbf{\Delta c}, \tag{3.21}$$

with $\mathbf{\Delta c} = \tilde{\mathbf{d}}_0 = \mathbf{\Delta C}\frac{f}{C_z}$ as the distance of the projected nadirs $\mathbf{N}_k$ and $\mathbf{N}_{k+\Delta k}$ on the image plane. If now both image planes are superimposed such that the camera centers overlap and the image plane axis are aligned, the parallax displacement can directly be calculated as the distance between the projected scene points, as is shown in Figure 3.7(b):

$$\tilde{d} = r'_{c,k} - r'_{c,k+\Delta k}, \quad \tilde{\mathbf{d}} = \mathbf{p}'_k - \mathbf{p}'_{k+\Delta k}. \tag{3.22}$$

To get the relations between the image parallax and the height and distance of an object, Equation 3.8 is feasible by replacing X_c with $\boldsymbol{\Delta}C$:

$$\tilde{\mathbf{d}} = \boldsymbol{\Delta}\mathbf{C}\frac{f}{C_z - h} \tag{3.23}$$

To get the motion displacement $\boldsymbol{\Delta}\mathbf{p_m}$ of a projected point $\mathbf{p}'$ with non-zero height h compared to the same point but on the ground plane $\mathbf{p}$ (with $h = 0$), we have to compensate the parallax of the ground level $\mathbf{d}_0$ (the background motion), which can be calculated by simply choosing $h = 0$:

$$\tilde{\mathbf{d}}_0 = \boldsymbol{\Delta}\mathbf{C}\frac{f}{C_z}, \tag{3.24}$$

and corresponds to the distance of the projected nadirs $\boldsymbol{\Delta}\mathbf{c}$. The subtraction of both parallaxes yields to the motion parallax displacement $\boldsymbol{\Delta}\mathbf{p_m}$:

$$\boldsymbol{\Delta}\mathbf{p_m} = \tilde{\mathbf{d}} - \tilde{\mathbf{d}}_0 = -\boldsymbol{\Delta}\mathbf{C}\frac{f \cdot h}{(C_z - h)\, C_z} = -\boldsymbol{\Delta}\mathbf{c}\frac{h}{(C_z - h)}, \tag{3.25}$$

which can be converted to image pel by applying the scaling from Equation 3.17:

$$\boldsymbol{\Delta}\mathbf{n_m} = \frac{N_x}{s_x}\boldsymbol{\Delta}\mathbf{p_m} = -\boldsymbol{\Delta}\mathbf{c}\frac{N_x}{s_x}\frac{h}{(C_z - h)}. \tag{3.26}$$

The same consideration can also be made for the motion displacement on the ground plane $\boldsymbol{\Delta}\mathbf{P_m}$, which gives us the relation between the virtual ground motion caused by the parallax effect and a real moving object on the ground plane:

$$\boldsymbol{\Delta}\mathbf{P_m} = -\boldsymbol{\Delta}\mathbf{C}\frac{h}{(C_z - h)} \tag{3.27}$$

It should be noted that,in contrast to the relief displacement, the motion error with a non-tilted camera is independent of the ground position $\mathbf{P}_0$ but dependent on the aircraft motion $\boldsymbol{\Delta}\mathbf{C}$ only.

Some examples of the motion error resulting from image parallaxes for different aircraft speeds $\mathbf{v}_{\text{plane}}$ and object height h are given in Figure 3.8. A full frame (36x20mm) 16:9 camera sensor with full-HD (1920x1080) resolution is assumed in all the following measurement. The motion error itself is a lot smaller than the associated relief displacement as the motion error is, in contrast to the relief displacement, not an absolute measure of aberration but describes the relative changes of the relief displacement between the camera positions instead. This leads to virtual motion of objects in the scene after the global compensation of the ground motion, depending on their height. In a detection system of moving ground objects, this might lead to mis-detections. The equivalent speed of a moving ground object is provided. For a better classification of the displacement magnitudes, Figure 3.9 provides an overview of the achievable image resolutions and the viewable ground area (fov) for the given camera setup.

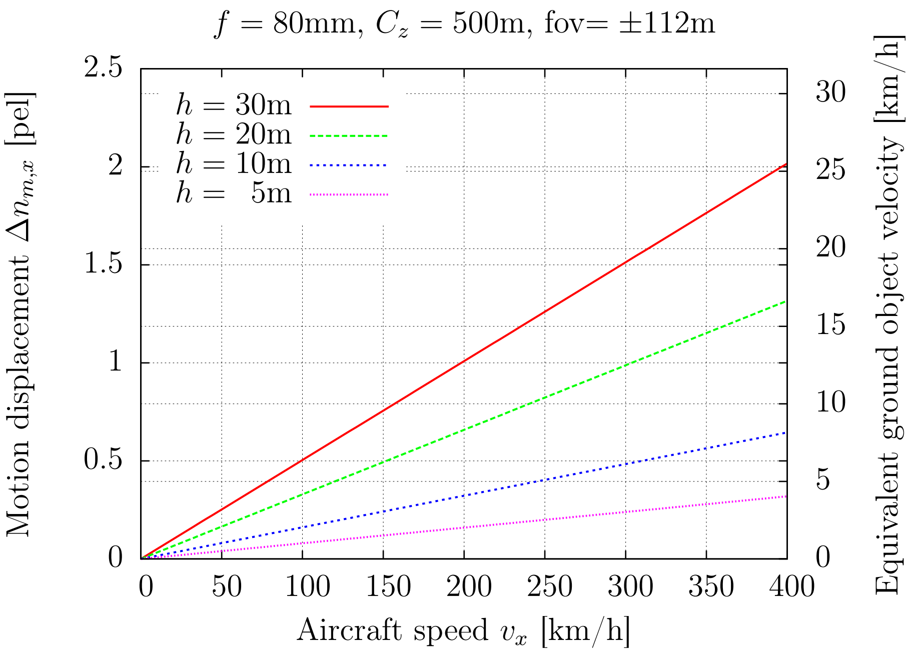


Figure 3.8: Motion displacement due to camera movement (x-component) and the equivalent ground object velocity for difference aircraft speeds.

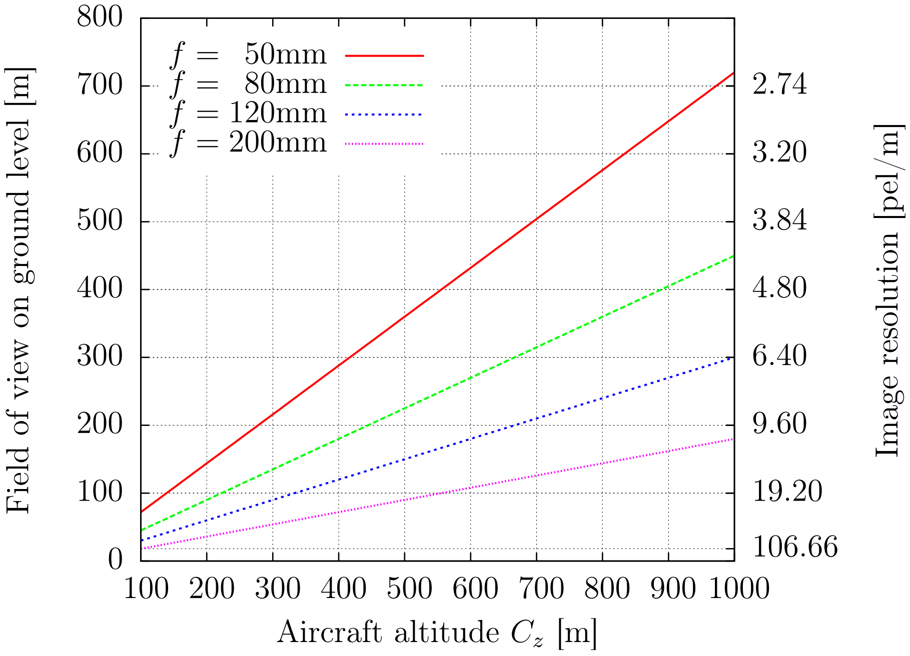


Figure 3.9: Field of view on the ground level and achievable image resolutions, depending on the flight altitude for various focal lengths.

3.4 Camera with a Tilt Angle

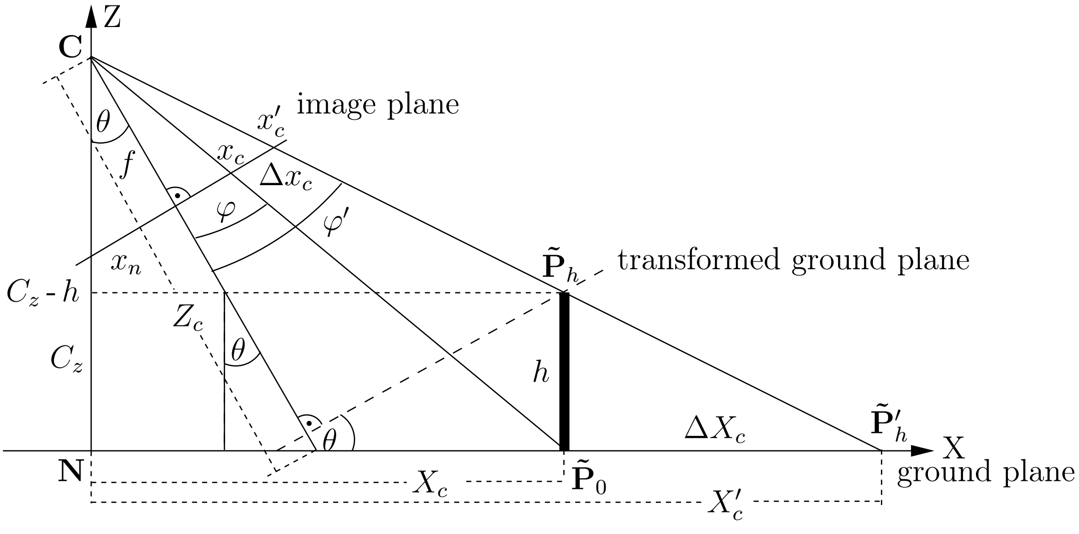


Figure 3.10: Tilted camera

In contrast to the vertical photo, the calculations become non-linear if the camera is tilted. In this first step, only a rotation around the Y-axis by an angle θ is assumed (tilting in flight direction). This is illustrated in Figure 3.10. It can be seen that the image plane and the ground plane are not parallel anymore. This results in a varying distance of the image plane to the ground plane. Moreover, the nadir is not intersecting with the principal point of the image plane, which was an implicit assumption in the previous cases.

In order to apply the previous equations to the tilted case, the image plane coordinates from Equation 3.6 have to be referenced to the nadir instead of the principal point by adding the nadir offset $x_n = f \cdot \tan\theta$:

$$\Delta x_c = h\,\frac{x'_c + f \cdot \tan\theta}{C_z} = h\,\frac{x_c + f \cdot \tan\theta}{C_z - h}. \tag{3.28}$$

To have object positions on the ground plane instead of the image plane, we have to consider the nadir offset again:

$$x_c = f \cdot \frac{X_c \cdot \cos\theta}{C_z + X_c \cdot \sin\theta} - f \cdot \tan\theta, \tag{3.29}$$

$$x'_c = f \cdot \frac{X_c \cdot \cos\theta + h \cdot \sin\theta}{C_z + X_c \cdot \sin\theta - h \cdot \cos\theta} - f \cdot \tan\theta. \tag{3.30}$$

To get the reflief displacement as a function of h, θ, and X_c, we have to calculate the difference between Equation 3.30 and Equation 3.29, as in Equation 3.13:

$$\Delta x_c = x'_c - x_c = f \cdot \left(\frac{X_c \cdot \cos\theta + h \cdot \sin\theta}{C_z + X_c \cdot \sin\theta - h \cdot \cos\theta} - \frac{X_c \cdot \cos\theta}{C_z + X_c \cdot \sin\theta} \right). \tag{3.31}$$

To have an easier expression, the tilted camera case can also be written in angular notation instead. This is done by substituting the ground positions X_c of the scene points $\tilde{\mathbf{P}}_0$ and $\tilde{\mathbf{P}}'_h$ with the angles φ and φ' between the view rays to the scene points and the optical axis, as is given in Figure 3.10:

$$\varphi = \tan^{-1}\left(\frac{X_c}{C_z}\right) - \theta, \tag{3.32}$$

$$\varphi' = \tan^{-1}\left(\frac{X_c}{C_z - h}\right) - \theta. \tag{3.33}$$

The mapping of the object points onto the image plane can now be expressed by:

$$x_c = f \cdot \tan\varphi, \tag{3.34}$$

$$x'_c = f \cdot \tan\varphi'. \tag{3.35}$$

By inserting Equation 3.32 and Equation 3.33 into Equation 3.34 and Equation 3.35, we get the mapping from an object point $\tilde{\mathbf{P}}_0$ on the ground plane onto $\mathbf{p} = (x_c, y_c)^\top$ on the image plane, or from an non-zero height object point $\tilde{\mathbf{P}}_h$ onto $\mathbf{p}' = (x'_c, y'_c)^\top$:

$$x_c = f \cdot \tan\left(\tan^{-1}\left(\frac{X_c}{C_z}\right) - \theta\right), \tag{3.36}$$

$$x'_c = f \cdot \tan\left(\tan^{-1}\left(\frac{X_c}{C_z - h}\right) - \theta\right). \tag{3.37}$$

The difference between those two is, again as in Equations 3.13 and 3.31, the relief displacement Δr_c on the image plane:

$$\Delta x_c = x'_c - x_c = f\left(\tan\left(\tan^{-1}\left(\frac{X_c}{C_z - h}\right) - \theta\right) - \tan\left(\tan^{-1}\left(\frac{X_c}{C_z}\right) - \theta\right)\right). \tag{3.38}$$

To get an universal radial expression as in Equation 3.13, one can substitute the X_c by the radius R_c and θ by its in tilted direction effective component $\theta_x = \theta \cdot \cos\beta$:

$$\Delta r_c = r'_c - r_c = f\left(\tan\left(\tan^{-1}\left(\frac{R_c}{C_z - h}\right) - \theta_x\right) - \tan\left(\tan^{-1}\left(\frac{R_c}{C_z}\right) - \theta_x\right)\right). \tag{3.39}$$

Figure 3.12 gives some exemplary plots. In Figure 3.12(a), the relief displacement is plotted dependent on the ground position and the tilt angle for building height of 20m. One can see that the relief displacement increases with the angle for closer positioned ground objects but falls slightly prior to climbing up for objects positioned further away. Common is that values of $\theta < 30°$ have only a small influence on the relief displacement for the given recording scenario, which can also bee seen in Figure 3.12(b) and (d). Figure 3.11 shows the influence of the tilt angle to the motion error, where the motion displacement Δx_m decreases with an increase of the tilt angle for a fixes object position of $X_c = 150\text{m}$.

The motion displacement can be calculated similarly to Equation 3.25 - however, one different method should be presented here by calculating the motion error from the differences in relief displacement instead. Hereby it is assumed that the camera is static but the ground position of the object has changed by the displacement of the camera:

$$\Delta x_m = \Delta x_c(X_c - \Delta C_x) - \Delta x_c(X_c) \quad \text{and} \tag{3.40}$$

$$\Delta r_m = \Delta r_c(R_c - \Delta C_x) - \Delta r_c(R_c), \tag{3.41}$$

wherein Δx_c and Δr_c are now functions of the ground positions X_c and R_c. The relief displacement and the motion displacement can be converted from m to pel similar to Equation 3.17 by applying the scaling factor from Equation 2.1.

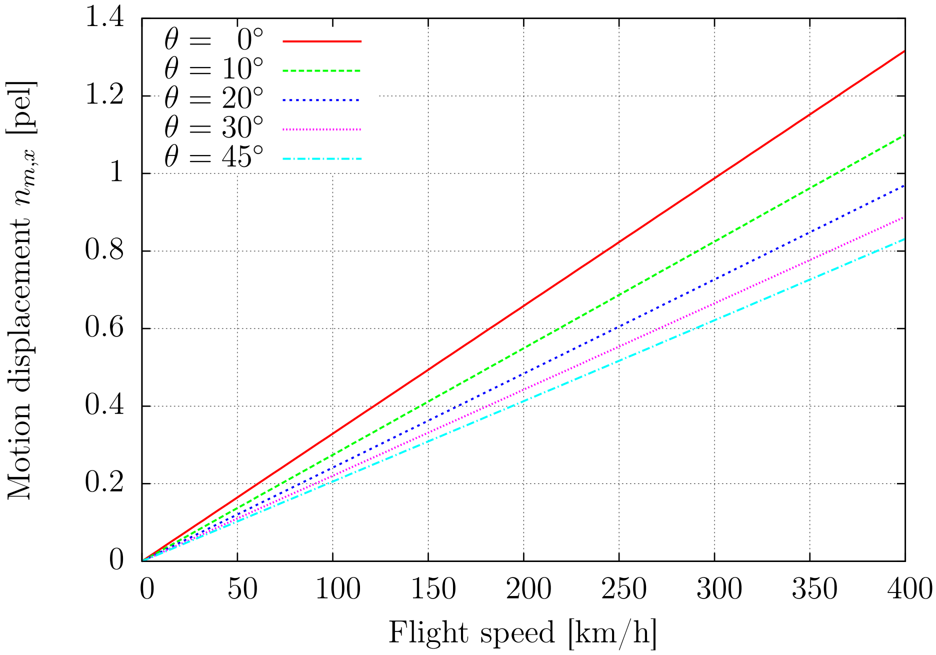
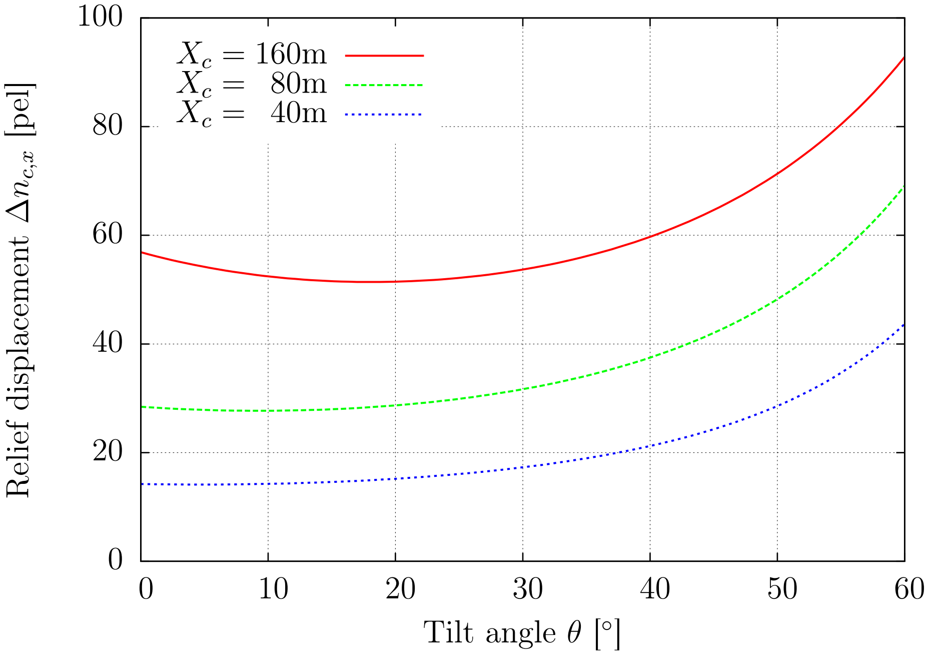
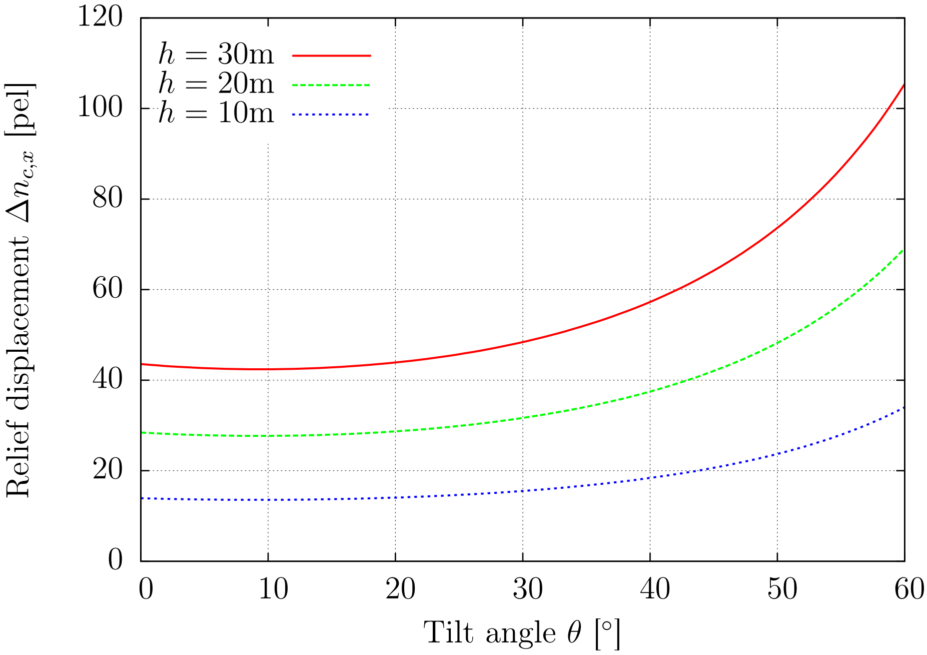


Figure 3.11: Motion displacement for different tilt angles and flight speeds. Altitude $C_z = 500\text{m}$, focal length $f = 80\text{mm}$, height of building $h = 20\text{m}$, ground position $X_c = 150\text{m}$, frame rate of recording $r_{\text{fps}} = 30\text{Hz}$, full-HD sensor.

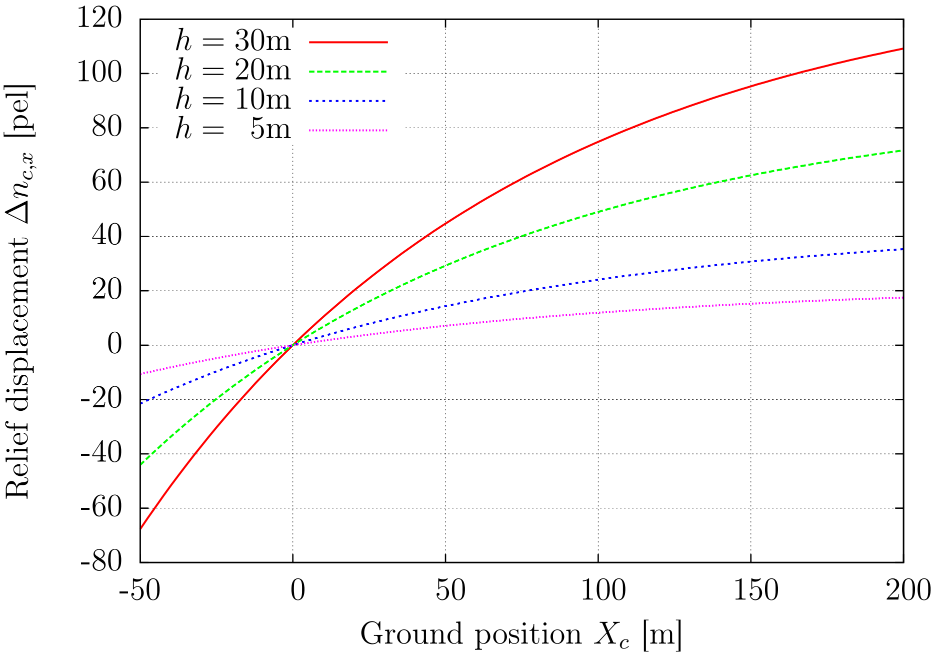
Figure 3.12: Relief displacement (x-component) for different tilt angles, ground positions and, building heights. Altitude $C_z = 500$m, focal length $f = 80$mm.



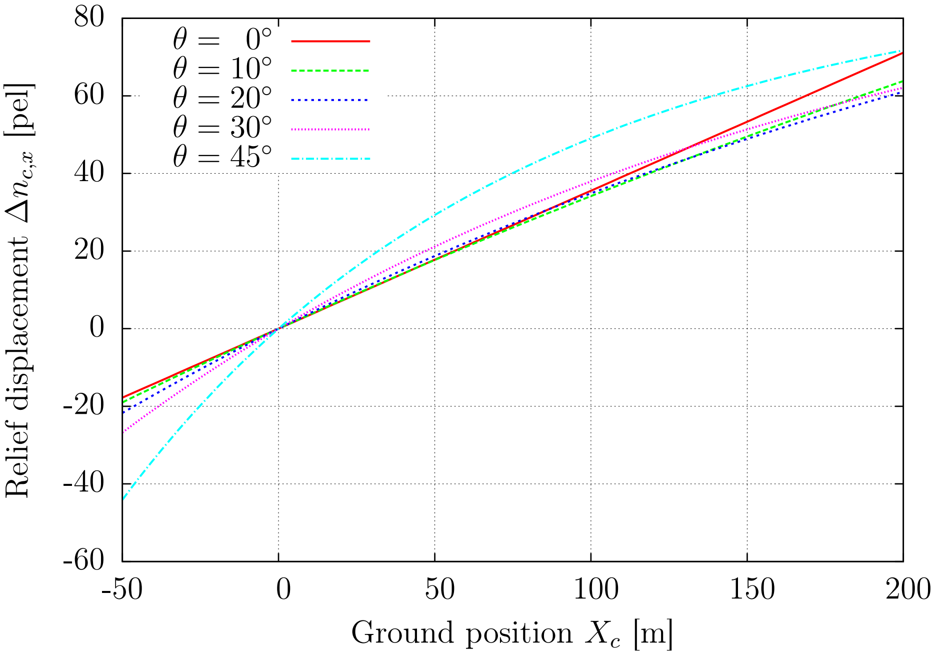
(a) Tilt angle vs. ground position. Building height $h = 20$m.



(b) Tilt angle vs. height of buildings. Ground position $X_c = 80$m.



(c) Heights of buildings vs. ground position. Tilt angle $\theta = 45°$.



(d) Ground position vs. tilt angle. Height of building $h = 20$m.

3.5 Arbitrary Camera Orientation

To express the mapping of scene points to image coordinates for arbitrary oriented cameras (Figure 3.13), we take the perspective mapping model from Chapter 2.2.4, especially Equation 2.21:

$$\mathbf{p} = \mathtt{KR}(\mathbf{P} - \mathbf{C}). \tag{3.42}$$

With the Rotation matrix $\mathtt{R}$ being

$$\mathtt{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{3.43}$$

Equation 3.42 expands to:

$$\mathbf{p} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11}(X - C_x) + r_{12}(Y - C_y) + r_{13}(Z - C_z) \\ r_{21}(X - C_x) + r_{22}(Y - C_y) + r_{23}(Z - C_z) \\ r_{31}(X - C_x) + r_{32}(Y - C_y) + r_{33}(Z - C_z) \end{bmatrix}. \tag{3.44}$$

The Euclidean form of Equation 3.44 is derived by dividing the first two vector components by the third, wherein, for the reason of simplicity and space, the principal point offset $\mathbf{c} = (c_x, c_y)^\top$ inside the matrix $\mathtt{K}$ is assumed to be $(0,0)^\top$ here:

$$\begin{aligned} x &= \frac{x'}{z'} = f\,\frac{r_{11}(X - C_x) + r_{12}(Y - C_y) + r_{13}(Z - C_z)}{r_{31}(X - C_x) + r_{32}(Y - C_y) + r_{33}(Z - C_z)}, \\ y &= \frac{y'}{z'} = f\,\frac{r_{21}(X - C_x) + r_{22}(Y - C_y) + r_{23}(Z - C_z)}{r_{31}(X - C_x) + r_{32}(Y - C_y) + r_{33}(Z - C_z)}. \end{aligned} \tag{3.45}$$

The coefficients of the rotation matrix are given by the individual rotations around the three axis by the angles β, γ, and θ, taken from Equation 2.19:

$$\begin{aligned} r_{11} &= \cos(\beta)\cos(\theta) + \sin(\beta)\sin(\gamma)\sin(\theta) \\ r_{12} &= \sin(\beta)\cos(\theta) - \cos(\beta)\sin(\gamma)\sin(\theta) \\ r_{13} &= -\cos(\gamma)\sin(\theta) \\ r_{21} &= -\sin(\beta)\cos(\gamma) \\ r_{22} &= \cos(\beta)\cos(\gamma) \\ r_{23} &= -\sin(\gamma) \\ r_{31} &= \cos(\beta)\sin(\theta) - \sin(\beta)\sin(\gamma)\cos(\theta) \\ r_{32} &= \sin(\beta)\sin(\theta) + \cos(\beta)\sin(\gamma)\cos(\theta) \\ r_{33} &= \cos(\gamma)\cos(\theta). \end{aligned} \tag{3.46}$$

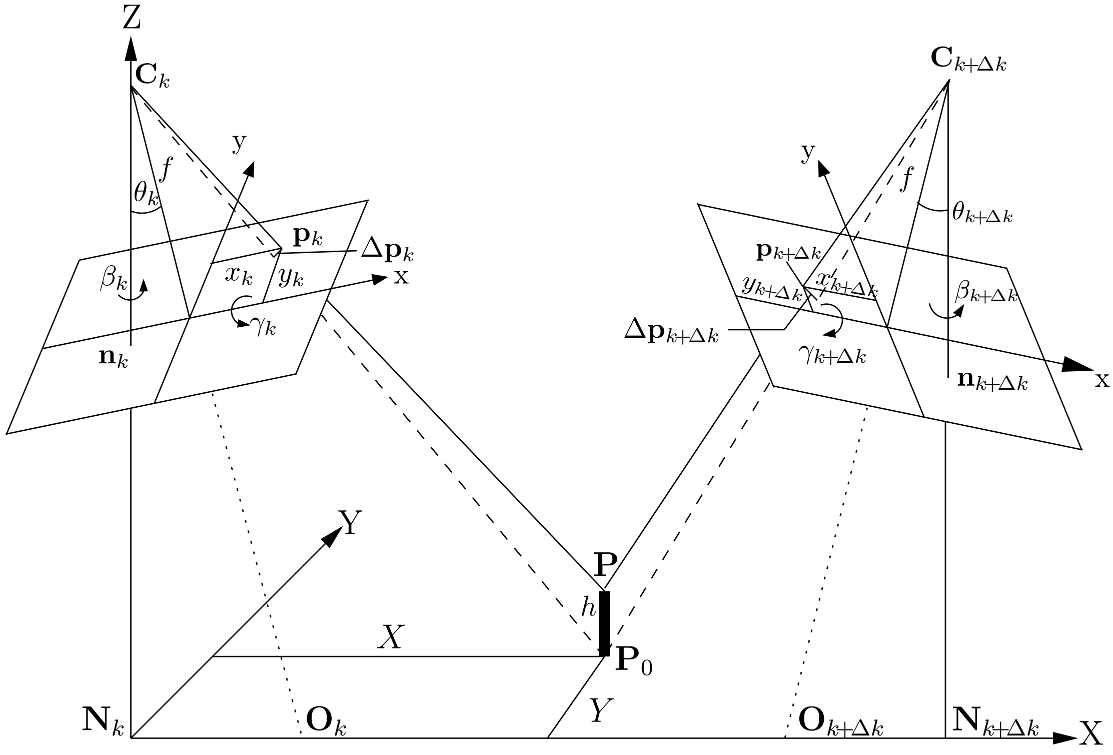


Figure 3.13: Collinearity characteristics and scene to image plane projections for arbitrary camera orientations

Note that if $\mathtt{R}$ is the identity matrix (meaning no rotations are performed), the Equations 3.45 simplify to the vertical photo case presented earlier as Equation 3.10 in Chapter 3.2, whereas having only a tilt angle θ results in the tilted only case from Equation 3.29 in Chapter 3.4.

To get the aberrations of the planar global motion compensation model for arbitrary camera movements and the imperfect compensation of non-zero height objects, we first investigate the relief displacement $\Delta\mathbf{p}_k = (\Delta x'_k,\Delta y'_k,\Delta z'_k)^\top$ for a single camera $\mathbf{C}_k$ as the difference of the projected points of a non-zero height scene point $\mathbf{P} = (X,Y,h)^\top$ in the frame k and a point $\mathbf{P}_0 = (X,Y,0)^\top$ at the same X and Y coordinates, but directly placed on the ground plane (see Figure 3.13, left part):

$$\Delta\mathbf{p}_k = \mathtt{K}_k\mathtt{R}_k(\mathbf{P} - \mathbf{C}_k) - \mathtt{K}_k\mathtt{R}_k(\mathbf{P}_0 - \mathbf{C}_k) \tag{3.47}$$

The Euclidean form expands to the following form, similar to Equation 3.45:

$$\Delta x_k = f_k \left(\frac{r_{11,k}X + r_{12,k}Y + r_{13,k}(h - C_{k,z})}{r_{31,k}X + r_{32,k}Y + r_{33,k}(h - C_{k,z})} - \frac{r_{11,k}X + r_{12,k}Y - r_{13,k}C_{k,z}}{r_{31,k}X + r_{32,k}Y - r_{33,k}C_{k,z}} \right), \tag{3.48}$$

$$\Delta y_k = f_k \left(\frac{r_{21,k}X + r_{22,k}Y + r_{23,k}(h - C_{k,z})}{r_{31,k}X + r_{32,k}Y + r_{33,k}(h - C_{k,z})} - \frac{r_{21,k}X + r_{22,k}Y - r_{23,k}C_{k,z}}{r_{31,k}X + r_{32,k}Y - r_{33,k}C_{k,z}} \right), \tag{3.49}$$

wherein, for easier representation, the camera is assumed to be positioned above the origin of the world coordinate system in the frame k, such that $\mathbf{N}_k = (0,0,0)^\top$ and $\mathbf{C}_k = (0,0,C_{k,z})$.

The same calculations as above can be applied to the frame $k+\Delta k$, resulting in the relief displacement $\Delta\mathbf{p}_{k+\Delta k}$ of $\mathbf{P}$:

$$\Delta\mathbf{p}_{k+\Delta k} = \mathtt{K}_{k+\Delta k}\mathtt{R}_{k+\Delta k}(\mathbf{P} - \mathbf{C}_{k+\Delta k}) - \mathtt{K}_{k+\Delta k}\mathtt{R}_{k+\Delta k}(\mathbf{P}_0 - \mathbf{C}_{k+\Delta k}). \tag{3.50}$$

In Euclidean notation, this is expressed as:

$$\begin{aligned}
\Delta x_{k+\Delta k} = f_{k+\Delta k} \Bigg(& \frac{r_{11}(X - C_{k+\Delta k,x}) + r_{12}(Y - C_{k+\Delta k,y}) + r_{13}(h - C_{k+\Delta k,z})}{r_{31}(X - C_{k+\Delta k,x}) + r_{32}(Y - C_{k+\Delta k,y}) + r_{33}(h - C_{k+\Delta k,z})} \\
& - \frac{r_{11}(X - C_{k+\Delta k,x}) + r_{12}(Y - C_{k+\Delta k,y}) - r_{13}C_{k+\Delta k,z}}{r_{31}(X - C_{k+\Delta k,x}) + r_{32}(Y - C_{k+\Delta k,y}) - r_{33}C_{k+\Delta k,z}} \Bigg), \\
\Delta y_{k+\Delta k} = f_{k+\Delta k} \Bigg(& \frac{r_{21}(X - C_{k+\Delta k,x}) + r_{22}(Y - C_{k+\Delta k,y}) + r_{23}(h - C_{k+\Delta k,z})}{r_{31}(X - C_{k+\Delta k,x}) + r_{32}(Y - C_{k+\Delta k,y}) + r_{33}(h - C_{k+\Delta k,z})} \\
& - \frac{r_{21}(X - C_{k+\Delta k,x}) + r_{22}(Y - C_{k+\Delta k,y}) - r_{23}C_{k+\Delta k,z}}{r_{31}(X - C_{k+\Delta k,x}) + r_{32}(Y - C_{k+\Delta k,y}) - r_{33}C_{k+\Delta k,z}} \Bigg).
\end{aligned} \tag{3.51}$$

Equations 3.47 and 3.50 cannot be further simplified, as the main minuend and the subtrahend are both individual perspective projections in homogeneous coordinates. A combination would result in the calculation of the difference of the projected 2D points on the image plane instead of the 3D points in the world coordinate system.

Due to the imperfect compensation of the background motion of non-planar objects between the frames k and $k+\Delta k$ caused by the insufficient global motion model, the relief displacement changes with the camera orientation and position (see Figure 3.14). This leads to a displacement of points on the image plane projected from static objects with a non-zero height related to the ground plane. This virtual motion of projected static objects was defined in Chapter 3.3 as the motion displacement $\Delta\mathbf{p}_{\mathbf{m}}$. As with the simplified case, it can be calculated directly from the difference of relief displacements:

$$\Delta\mathbf{p}_{\mathbf{m}} = \Delta\mathbf{p}_{k+\Delta k} - \Delta\mathbf{p}_k \tag{3.52}$$

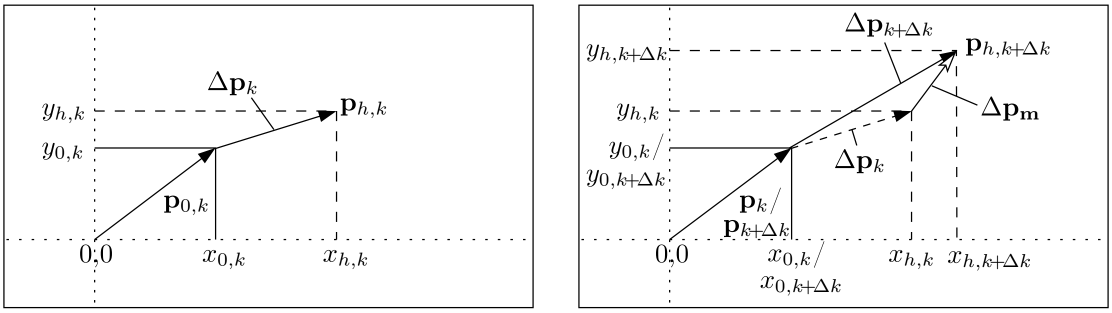


image plane frame k image plane frame $k+\Delta k$

Figure 3.14: Relief displacements $\Delta\mathbf{p}_k$ and $\Delta\mathbf{p}_{k+\Delta k}$ on the image planes of the frames k and $k+\Delta k$ as well as the resulting motion displacement $\Delta\mathbf{p}_{\mathbf{m}}$. $\mathbf{p}_k$ and $\mathbf{p}_{k+\Delta k}$ are identical here due to the global motion compensation.

3.6 Height Restrictions based Outlier Detection

Until now, the difference in displacement of the projected points $\mathbf{p}_{0,k}$ and $\mathbf{p}_{h,k} = \mathbf{p}_{0,k} + \Delta\mathbf{p}_k$ between the frames k and $k+\Delta k$ can be calculated from the known camera parameters $\mathtt{K}_k$, $\mathtt{R}_k$ and $\mathbf{C}_k$ as well as $\mathtt{K}_{k+\Delta k}$, $\mathtt{R}_{k+\Delta k}$ and $\mathtt{C}_{k+\Delta k}$ by Equation 3.52 for known positions and heights of the original scene points $\mathbf{P}_h = (X,Y,h)^\top$ and $\mathbf{P}_0 = (X,Y,0)^\top$. However, for a feature point $\mathbf{p}_k$ for which the actual position and height of the original scene point is unknown (as it is true for a real operating system), estimates of the minimum and maximum height h_{min} and h_{max} of the original scene point $\mathbf{P}_k$, e.g. as the maximum height of buildings or trees in the scene, can be made. This information can then be used to define a range of valid displacements $\Delta\mathbf{p}_{0,k}$ to $\Delta\mathbf{p}_{h_{\text{max}},k}$, which might apply to $\mathbf{p}_{k+\Delta k}$ due to motion parallax if $\mathbf{P}_k$, as the source of $\mathbf{p}_k$ and $\mathbf{p}_{k+\Delta k}$, is a static scene point with a z-coordinate between h_{min} and h_{max}. Comparing the actual measure of $\mathbf{p}_{k+\Delta k}$ to the prediction $\mathbf{p}_k + \Delta\mathbf{p}_k$ (with $\Delta\mathbf{p}_{0,k} \leq \Delta\mathbf{p}_k \leq \Delta\mathbf{p}_{h_{\text{max}},k}$) allows the detection of outliers or non-static objects on the ground plane such as moving cars or pedestrians. This is illustrated in Equation 3.53.

To do so, first we need to calculate the viewing ray from the camera center $\mathbf{C}_k$ through the measured feature point $\mathbf{p}_k$. As this is an inverse projection from the image plane into the scene, this can be achieved by inverting Equation 3.42, which was the projection from a scene point onto the image plane:

$$\mathbf{P}_h = \begin{pmatrix} X \\ Y \\ h \end{pmatrix} = (\mathtt{K}_k \mathtt{R}_k)^{-1}\, \mathbf{p}_k + \mathbf{C}_k \tag{3.53}$$

The Euclidean form of Equation 3.53 retrieves the X and Y world coordinate of a possible source of $\mathbf{p}_k$ for a scene point with a given Z-coordinate of h. This is done

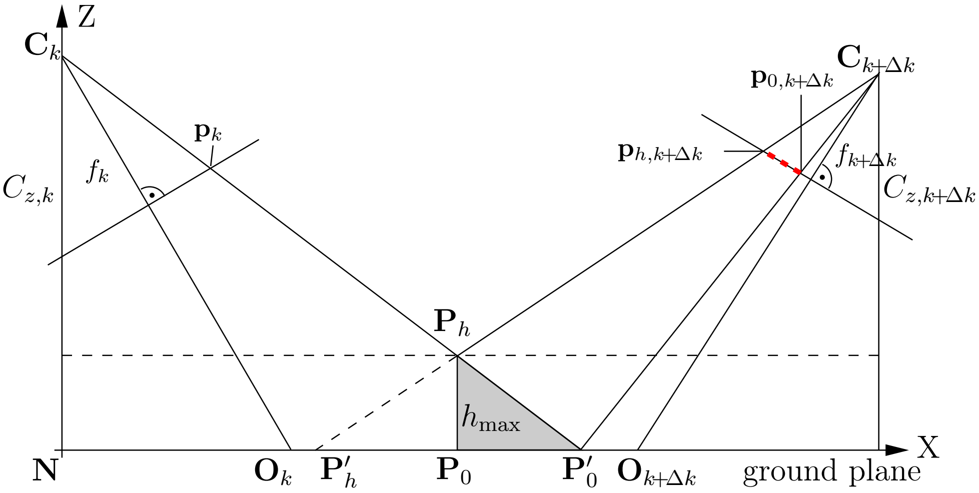


Figure 3.15: Retrieval of the parallax prediction line (red, dashed). The scene points $\mathbf{P}_h$ and $\mathbf{P}_0$ are determined by calculating the ground coordinates of the ray through $\mathbf{p}_k$ at the heights $h_{\max}$ and 0, respectively. Both scene points are projected onto the image plane of camera $\mathbf{C}_{k+\Delta k}$ as $\mathbf{p}_{h,k+\Delta k}$ and $\mathbf{p}_{0,k+\Delta k}$. They define a constrained epipolar line piece on which the point $\mathbf{p}_{k+\Delta k}$ has to lie, if it was projected from a static ground object.

by first rotating the image plane coordinate $\mathbf{p}_k$ into the world coordinate system and then projecting it to a virtual plane in parallel to the ground plane with a Z-coordinate of h. By inserting $h_{\min}$ and $h_{\max}$ for h, this yields to two possible sources of $\mathbf{p}_k$ in the Euclidean form: one with a Z-coordinate of $h_{\min}$ and one with a Z-coordinate of $h_{\max}$, respectively.

In Figure 3.15, $h_{\min} = 0$ was assumed as the minimum height of objects. This retrieves the scene points $\mathbf{P}_h$ and $\mathbf{P}'_0$ as the intersections of the viewing ray from $\mathbf{C}_k$ through $\mathbf{p}_k$ with the plane at $h_{\max}$ (illustrated by the dashed line) and the ground plane at 0. Both points - as any point on the viewing ray - are possible sources for the projected point $\mathbf{p}_k$, but as $\mathbf{P}_h$ and $\mathbf{P}'_0$ were created by assuming a maximum and minimum height of objects in the scene, they restrict the viewing ray to a possible area (the part of the viewing ray touching the gray triangle in Figure 3.15) as valid sources for $\mathbf{p}_k$. If this section of the view is projected onto the image plane of camera $\mathbf{C}_{k+\Delta k}$, it creates a line segment, limited by the points $\mathbf{p}_{h,k+\Delta k}$ and $\mathbf{p}_{0,k+\Delta k}$ as projections of $\mathbf{P}_h$ and $\mathbf{P}'_0$, on which a point $\mathbf{p}_{k+\Delta k}$ has to lie if it was projected from a static ground objects under the above height restrictions. Note that in Chapter 2.4, the projection of the viewing ray for a point on the image plane of one camera onto the image plane of a second camera was introduced as the epipolar line. For this reason, the above restriction might also be named a constrained epipolar projection.

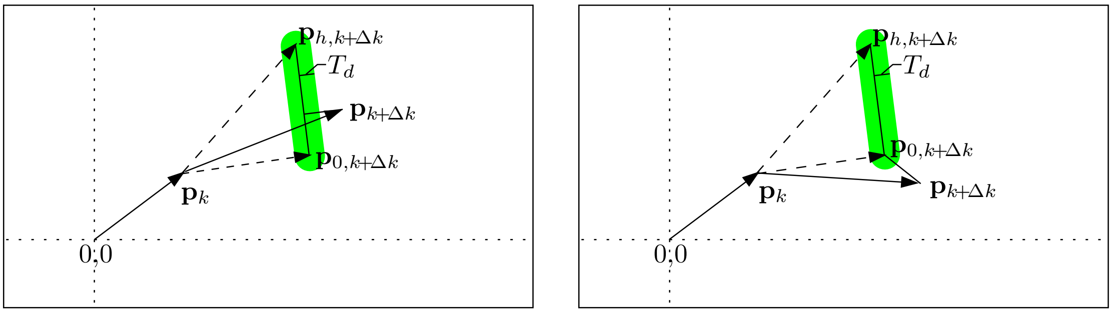

$\mathbf{p}_{h,k+\Delta k}$
T_d
$\mathbf{p}_{k+\Delta k}$
$\mathbf{p}_{0,k+\Delta k}$
$\mathbf{p}_k$
0;0
$\mathbf{p}_{h,k+\Delta k}$
T_d
$\mathbf{p}_{0,k+\Delta k}$
$\mathbf{p}_k$
$\mathbf{p}_{k+\Delta k}$
0;0

Figure 3.16: $\mathbf{p}_{k+\Delta k}$ is classified as non-background motion, if the distance of $\mathbf{p}_{k+\Delta k}$ to the epipolar line segment $\overline{\mathbf{p}_{0,k+\Delta k}, \mathbf{p}_{h,k+\Delta k}}$ is greater than a threshold T_d (green areas). The distance is always calculated to the closest element of the line segment (left: $0 \leq \lambda \leq 1$, right: $\lambda < 0$).

The projection of the line segment can be performed by projecting the start and end point of the line $\mathbf{p}_{0,k+\Delta k}$ and $\mathbf{p}_{h,k+\Delta k}$ using Equation 3.42 and the scene points $\mathbf{P}_h$ and $\mathbf{P}'_0$ as sources. The line equation is then consequently written as:

$$\mathbf{p}'_{k+\Delta k}(\lambda) = \mathbf{p}_{0,k+\Delta k} + \lambda(\mathbf{p}_{h,k+\Delta k} - \mathbf{p}_{0,k+\Delta k}), \text{with } 0 \leq \lambda \leq 1, \tag{3.54}$$

where $0 \leq \lambda \leq 1$ limits the valid range of the line to the segment given by $\mathbf{p}_{0,k+\Delta k}$ and $\mathbf{p}_{h,k+\Delta k}$. By computing the distance of the actual measurement of the feature position $\mathbf{p}_{k+\Delta k}$ to the constrained epipolar line segment $\mathbf{p}'_{k+\Delta k}(\lambda)$ and comparing it to a threshold T_d, feature displacements can be classified into being caused by either motion parallax of static objects ($\leq T_d$) or individual object motion ($> T_d$):

$$\min(|\mathbf{p}_{k+\Delta k} - \mathbf{p}'_{k+\Delta k}(\lambda)|) \leq T_d, \text{with } 0 \leq \lambda \leq 1. \tag{3.55}$$

This can be implemented by calculating the distance of the point $\mathbf{p}_{k+\Delta k}$ to the line given by $\mathbf{p}'_{k+\Delta k}(\lambda)$. However, if the closest point on the epipolar line is beyond the line segment ($\lambda < 0 \vee \lambda > 1$), the distance must be calculated to the closest point of the line segment instead ($\mathbf{p}_{0,k+\Delta k}$ or $\mathbf{p}_{h,k+\Delta k}$). Both cases are displayed in Figure 3.16. For the use in practice, the parameters $\mathtt{K}_k, \mathtt{K}_{k+\Delta k}, \mathtt{R}_k, \mathtt{R}_{k+\Delta k}, \mathtt{C}_k$, and $\mathtt{C}_{k+\Delta k}$ are available from the GPS/INS sensor of the aerial vehicle or are known from the camera settings. The minimal and maximal height of a building $h_{\max}$ can be chosen to a value representing the individual observed scene.

3.7 Detectability in Dependence of Speed and Direction of Motion

As the test above compares the measured feature displacement against a constrained epipolar line, it should be mentioned that moving objects moving along the epipolar line are only detectable, if their displacement is bigger or smaller than the maximal and minimal motion parallax from static objects, given by the position P_0 and maximum and minimum height of scene objects $h_{\max}$ and $h_{\min}$. By projecting the epipolar line segment onto the ground plane, this retrieves a range of positions in which the moving object cannot be located at the time the frame $k+\Delta k$ is taken in order to be detected. In Figure 3.16, this corresponds to the area on the ground plane between the points $\mathbf{P}'_0$ and $\mathbf{P}'_h$. This also means that an object moving opposite of the motion parallax displacement is always detectable, independent of the speed (corresponding to a location at the time of frame $k{+}\Delta k$ right of $\mathbf{P}'_0$). By incorporating the time difference between the frame k and $k{+}\Delta k$, which is expressed here by the frame rate r_{fps} of the camera, this directly defines a speed threshold $\mathbf{v}_{\text{thresh}}$ the moving object must exceed to be detected if it moves in direction of $\mathbf{P}'_h$:

$$\mathbf{v}_{\text{thresh}} = \frac{\mathbf{P}'_h - \mathbf{P}'_0}{r_{\text{fps}}}. \tag{3.56}$$

$\mathbf{P}'_h$ is calculated analogously to Equation 3.53, but with respect to the camera $\mathbf{C}_{k+\Delta k}$:

$$\mathbf{P}'_h = \begin{pmatrix} X'_h \\ Y'_h \\ 0 \end{pmatrix} = \left(\mathtt{K}_{k+\Delta k}\mathtt{R}_{k+\Delta k}\right)^{-1} \mathbf{p}_{h,k+\Delta k} + \mathbf{C}_{k+\Delta k}. \tag{3.57}$$

To give a generalized statement of detectability, the distance threshold T_d in pel from Equation 3.55 must additionally be taken into account. As the distance threshold is equally distributed around the epipolar line segment, the necessary minimal speed is dependent on the direction of motion of the moving object. Considering only the image plane, this corresponds to a displacement threshold Δp_{thresh} on the image plane the moving object must exceed to be detectable, depending on the angle δ between the direction of motion and the epipolar line segment:

$$\Delta p_{\text{thresh}} = \begin{cases} T_d & \text{for } 90^\circ \leq \delta \leq 270^\circ \\ T_d + \left|\mathbf{p}_{h,k+\Delta k} - \mathbf{p}_{0,k+\Delta k}\right| \cdot \cos\delta & \text{else} \end{cases}. \tag{3.58}$$

The equivalent ground velocity $\mathbf{v}_{\text{thresh}}$ is calculated by inserting the image plane coordinates of the minimal displacement into Equation 3.56:

$$\mathbf{v}_{\text{thresh}} = \frac{\left(\mathtt{K}_{k+\Delta k}\mathtt{R}_{k+\Delta k}\right)^{-1} \left(\mathbf{p}_{0,k+\Delta k} + \Delta\mathbf{p}_{\text{thresh}}\right) + \mathbf{C}_{k+\Delta k} - \mathbf{P}'_0}{r_{\text{fps}}}, \tag{3.59}$$

$$\text{with } \Delta\mathbf{p}_{\text{thresh}} = \Delta p_{\text{thresh}} \begin{pmatrix} \sin\delta \\ \cos\delta \end{pmatrix}.$$

4 Multi-Planar Landscape Model based on Triangle Meshes

In this chapter, a novel multi-planar ground model is presented and analyzed for approximation errors. The goal of the proposed method is to suppress the relief displacements of non-planar ground structure during motion compensation for the use in moving object detection systems. In contrast to the planar landscape model, multiple affine projections are applied to compensate the global motion of the scene locally instead of using one single homography for the whole frame. For this, the scene is overlapped by a triangle mesh wherein the single mesh nodes should be positioned as close as possible to geometric discontinuities to be able to take care of the different parallaxes. An example for an automatically created mesh overlay is given in Figure 4.1. If the motion of the mesh nodes is known between the frames and the displacements are small enough, the unknown motion of the pixels inside each patch can be approximated as affine, wherein the surrounding mesh nodes provide enough information to define the affinity.

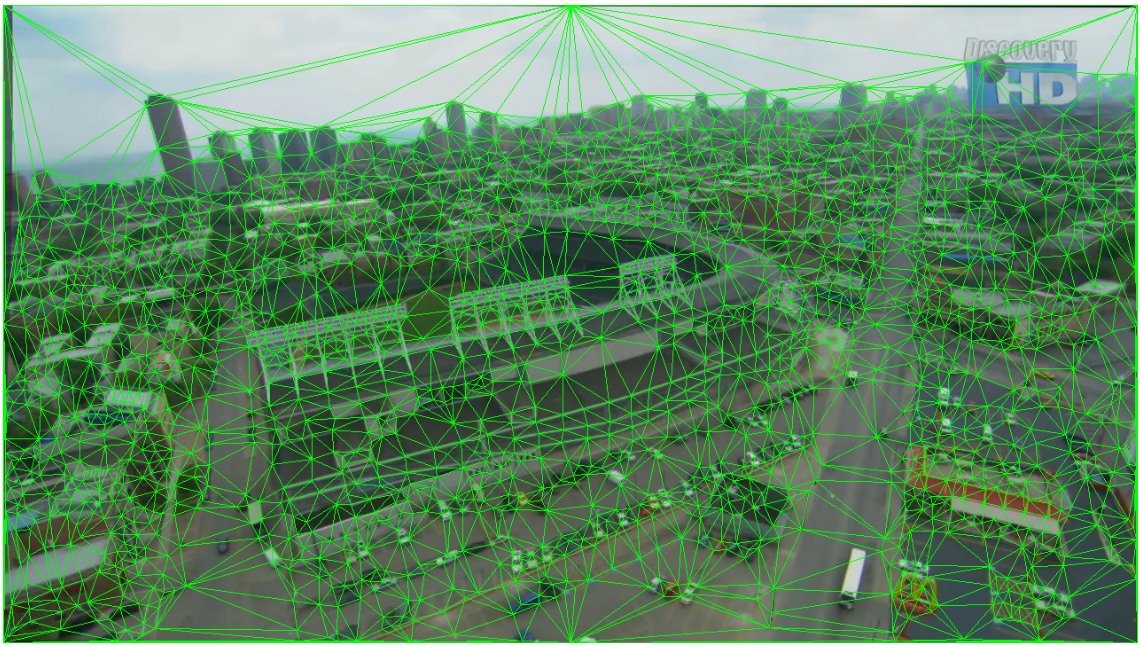


Figure 4.1: Mesh overlay for the Chicago test sequence [62].

4.1 Mesh Creation

4.1.1 Mesh Node Selection

The shape of the mesh patches should reflect the shape of planes in the scene with different distances to the camera center (different heights from the ground plane). The best possible solution to construct the mesh is by using an already existing 3D model of the scene. However, as the system should run without prior knowledge, a fully automatic technique is preferred. In this work, it is assumed that the Shi&Tomasi corner feature detector (Chapter 2.7.1) is exact enough to detect the relevant corners of buildings and other structures. The feature candidates must hereby fulfill a minimum cornerness criterion of $CRF > \lambda_{\text{min}} = 0.01$ and a minimal feature-to-feature distance of $d_f = 9$ pel – although these parameters might be adjusted for the individual use case. The corner detection is performed on the current frame k of the input video. The detected and elected n corner positions are stored as the feature position $\mathbf{f}_{n,k}$ and become the mesh nodes of frame k. An example of the corner-based feature selection was given in Figure 2.13.

4.1.2 Mesh Node Tracking

To determine the displacement of the mesh nodes between the frames of interest, the method of Kanade-Lucas-Tomasi (Chapter 2.7.2) is applied to the mesh nodes. It estimates the position of the given corner features $\mathbf{f}_{n,k}$ of frame k in a second frame $k-\Delta k$. The new positions of the features are stored as $\mathbf{f}_{n,k-\Delta k}$. This results in two possible meshes: one created by the selected corner feature positions and one created from the feature positions with the displacement applied (see Figure 4.2).

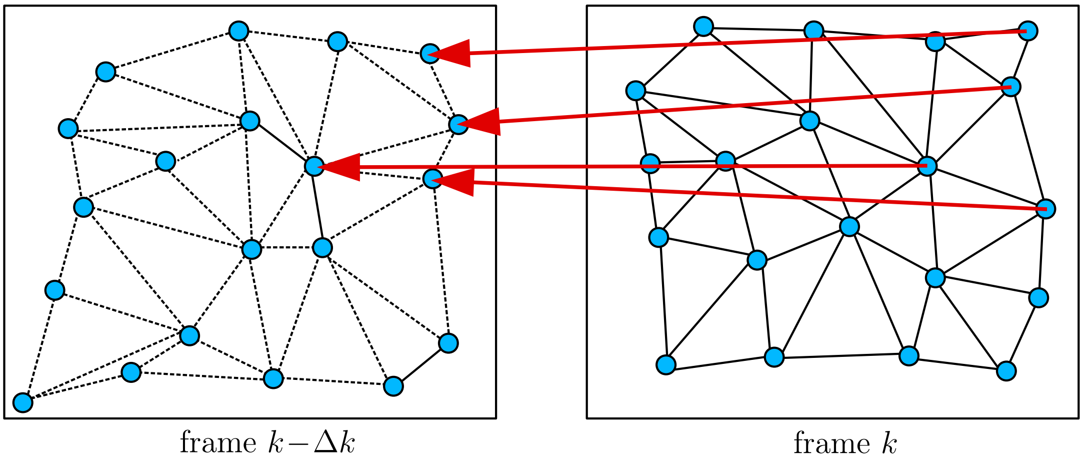
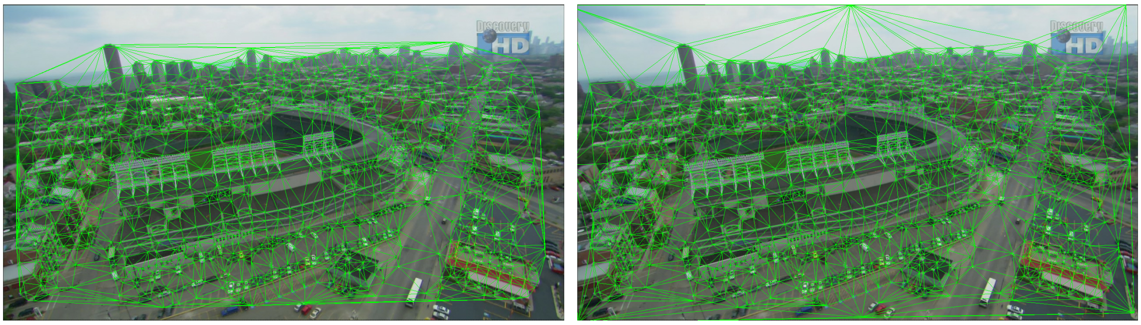


Figure 4.2: The position of the mesh nodes are tracked from frame to frame. This yields to two individual meshes, related by the mesh node displacements.

4.1.3 Mesh Coverage Padding

The resulting mesh does not cover the full image plane yet. This is due to the mesh emerging from the concatenation of the mesh nodes. The nodes themselves have been placed by the corner detection algorithm, which uses a rectangular window to calculate the cornerness and therefore is not able to reach the image boundaries. In addition, displacements cannot be estimated if features only occur in one of the frames. This happens a lot at image boundaries due to the features wandering out of the frame. The mesh coverage problem is illustrated in Figure 4.3a). Two possible solutions to this problem are to limit the moving object detection to the mesh-covered region or to generate additional feature positions and displacements at the image boundaries for the mesh to cover the full frame. In this work, the later approach was chosen by extrapolating eight additional displacement vectors at the frame boundary: four at each corner and four on the centers of the boundary line between each of the corners. More samples might be added to obtain a finer approximation. The displacements are extrapolated from known feature displacements nearby. This can be done by a (non-)linear extrapolation from multiple feature displacements e.g. by computing an affinity, fitting a polynomial, or by using belief propagation; however experiments by the author have shown that copying the displacement of the closets feature is, besides being fast, accurate enough for the scenario in this work. The filled up mesh using the eight additional boundary features and displacements is shown in Figure 4.3b).



(a) Without boundary features (b) With boundary features added

Figure 4.3: Extra features are inserted at the image boundaries to handle the uncovered border region. Their displacements are extrapolated from known feature displacements close by. Test sequence Chicago [62].

4.2 Outlier Removal and Moving Object Detection

As the mesh-based global motion compensation relies on correctly determined movement of the mesh nodes, it is mandatory to eliminate outliers from the estimated optical flow as well as any motion not emerging from the movement of the static background scene. In contrast to the planar model from Chapter 3, in this locally adaptive approach it is not possible to validate each determined feature displacement against a globally optimized model, e.g. by using the RANSAC algorithm from Chapter 2.7.3.

Instead, in this work it is proposed to identify the correctness of a feature vector by comparing its displacement to the local surrounding. As the background scene is static, all measured feature displacements of background objects must have emerged from the movement of the camera itself, overlapped by the virtual motion resulting from relief displacements due to different distances of scene points to the camera. If the spatial distance of the camera positions between the frames is small (as is true for normal video frame rates and flight speeds), and the distance to scene objects is big enough, the vector field is dominated mainly by the motion of the camera. Therefore all features spatially close should move more or less similarly between the frames, if they belong to the static background.

To exploit this assumption, a region growing approach is proposed in which a nearby feature displacement vector is clustered into an existing region if the distance of the feature position to the border of the cluster (distance constraint) and the difference in displacements (similarity constraint) between the feature vector and the border of the cluster are both small enough. As the algorithm is intended to be computed on-board small aerial devices with a hard power and battery limit, the proposed algorithm eyes on low computational complexity. Therefore, in this work, the border of the cluster is defined only by the feature vector position $\mathbf{u} = \mathbf{f}_{u,k}$ with the shortest distance to the yet to be clustered feature vector $\mathbf{v} = \mathbf{f}_{v,k}$. The distance constraint can then be written as:

$$|\mathbf{u} - \mathbf{v}| = \sqrt{(u_x - v_x)^2 + (u_y - v_y)^2} < T_1. \tag{4.1}$$

In this work, it is approximated by this easier to calculate implementation:

$$|u_x - v_x| + |u_y - v_y| < T_1. \tag{4.2}$$

The similarity constraint evaluates the difference in displacement between the same two vectors located at $\mathbf{u}$ and $\mathbf{v}$. The displacement is the difference between the coordinates of the feature position in the frames k and $k-\Delta k$, and calculates for the two vectors as follows:

$$\mathbf{d}_u = \begin{pmatrix} d_{u,x} \\ d_{u,y} \end{pmatrix} = \mathbf{f}_{u,k} - \mathbf{f}_{u,k-\Delta k}, \qquad \mathbf{d}_v = \begin{pmatrix} d_{v,x} \\ d_{v,y} \end{pmatrix} = \mathbf{f}_{v,k} - \mathbf{f}_{v,k-\Delta k}. \tag{4.3}$$

The similarity constraint is defined as the Euclidean distance of the displaced coordinates:

$$|\mathbf{d}_u - \mathbf{d}_v| = \sqrt{(d_{u,x} - d_{v,x})^2 + (d_{u,y} - d_{v,y})^2} < T_2. \tag{4.4}$$

In this work, the easier to compute linear approximation was used:

$$|d_{u,x} - d_{v,x}| + |d_{u,y} - d_{v,y}| < T_2. \tag{4.5}$$

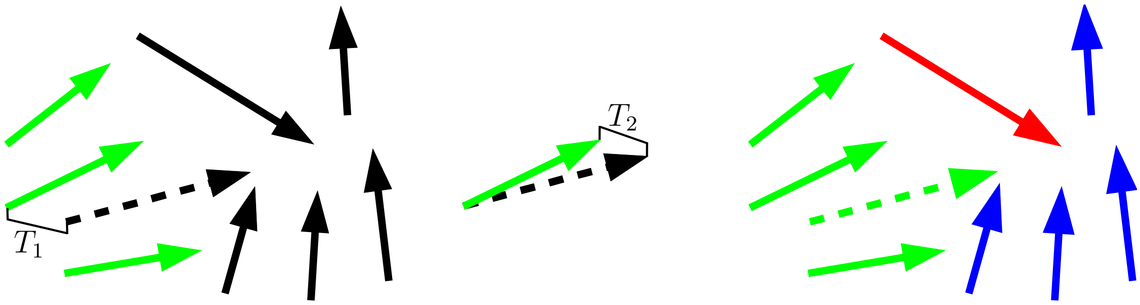
In addition, two more similarity constraints have been investigated. The first one scales the similarity threshold with the distance. It forces closer features to a higher similarity compared to those further away:

$$|d_{u,x} - d_{v,x}| + |d_{u,y} - d_{v,y}| < T_2 \cdot \frac{|u_x - v_x| + |u_y - v_y|}{T_1}. \tag{4.6}$$

The second one is based on the assumption that, for small displacements, the difference in displacement is also small if the feature displacement vectors are similar. Therefore, the similarity threshold is scaled down by the absolute value of the displacement vector, clipped at 5 pel:

$$|d_{u,x} - d_{v,x}| + |d_{u,y} - d_{v,y}| < \begin{cases} T_2 \cdot \frac{|d_{u,x} - d_{v,x}|}{5} & \text{for } |d_{u,x} - d_{v,x}| < 5.0 \\ T_2 & \text{for } |d_{u,x} - d_{v,x}| \geq 5.0 \end{cases}. \tag{4.7}$$

The clustering process is illustrated in Figure 4.4. The green arrows assemble the displacement vectors of the currently growing cluster whereas the dashed arrow is a nearby displacement vector under test. As the distance to the cluster is smaller than a threshold T_1 (Figure 4.4(a)) and the difference in displacement is smaller than the



(a) The green vectors assemble an already clustered object while the dashed vector is currenty tested againts the closest boundary vector.

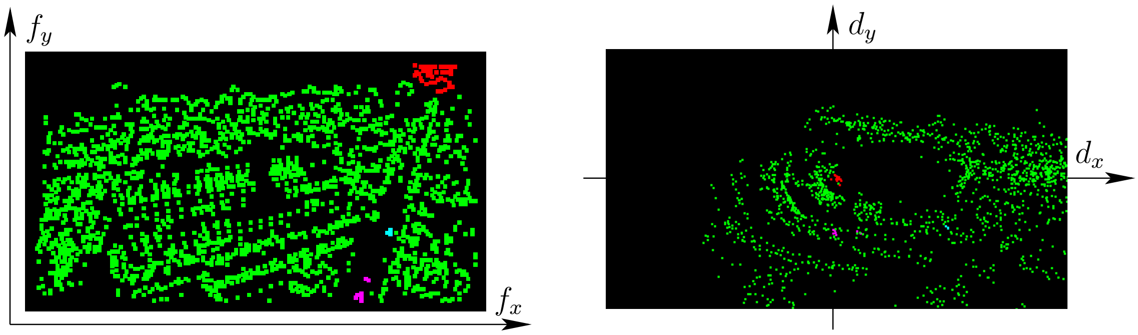
(b) The similarity is verified by the difference in displacement.

(c) The dashed vector was clustered into the green object while the blue vectors assemble a new cluster. The single red vector is marked as an outlier.

Figure 4.4: Displacement vector clustering by similarity

threshold T_2 (Figure 4.4(b)), it is merged into the cluster. This process is repeated until no further feature vector fulfills the distant and similarity constraints. In this case a new cluster is created (blue arrows in Figure 4.4(c) with one of the remaining feature vectors. The whole process is repeated until all feature vectors are clustered into individual objects. In a final outlier elimination step, all clusters containing less feature vectors than a threshold T_3 are marked as outliers and removed from the list of mesh nodes. This is illustrated by the red arrow in Figure 4.4(c).

An exemplary clustering result is given in Figure 4.5 for the chicago sequence. Figure 4.5(a) shows the feature positions and Figure 4.5(b) the displacement distribution of the feature vectors. The different colors denote different object clusters.



(a) Spatial positions of the features

(b) Displacement distribution

Figure 4.5: Clustered vector field. Green dots: background object, other colors: detected moving objects.

4.3 Mesh-based Motion Compensation

4.3.1 Triangulation of the Mesh Node Point Cloud

The Shi&Tomasi corner detector only provides an amount of point coordinates in the image plane. To create a connected mesh, the points have to be assigned to triangles. This process is known as the triangulation of a point cloud. In this work, the triangulation process is handled by the Delaunay method, which was presented in Chapter 2.8.1. As the goal is to compensate the background motion only for the purpose of moving object detection, only the feature points referenced by the background motion object are given to the triangulation process. The triangulation is performed for each of the frames k individually by using only the feature point coordinates detected in the frame k.

The result is a set of i triangles, wherein each triangles $t_{i,k}$ references the three determining mesh nodes/feature coordinates $t_{i,k} = \left\{\mathbf{f}_{t_{i,1},k}, \mathbf{f}_{t_{i,2},k}, \mathbf{f}_{t_{i,3},k}\right\}$.

4.3.2 Multi-planar Affine Transformation

The previous process actually creates two coherent meshes: the one created by the Delaunay method using the feature coordinates $\mathbf{f}_{n,k}$ of the frame k and the one created by the same mesh topology but using the displaced feature coordinates $\mathbf{f}_{n,k-\Delta k}$ of the frame $k-\Delta k$ (see Figure 4.2). Therefore, the triangulation process actually creates i pairs of associated triangles:

$$t_{i,k} = \left\{\mathbf{f}_{t_{i,1},k}, \mathbf{f}_{t_{i,2},k}, \mathbf{f}_{t_{i,3},k}\right\}, \qquad t_{i,k-\Delta k} = \left\{\mathbf{f}_{t_{i,1},k-\Delta k}, \mathbf{f}_{t_{i,2},k-\Delta k}, \mathbf{f}_{t_{i,3},k-\Delta k}\right\}. \tag{4.8}$$

Each pair of triangles provides three point correspondences, which equals six restrictions (three feature positions in two frames with two coordinates each). Assuming the motion inside each triangle as linear, this is enough information to determine an individual affine transformation for each triangle. To calculate the affine matrices $\mathtt{H}_i$, each pair of triangles has to be written as a matrix, containing the referenced feature point coordinates in their homogeneous form. For this, let $\mathbf{f}_n = (x_n, y_n, 1) = \mathbf{f}_{t_{i,n},k}$ and $\mathbf{f}'_n = (x'_n, y'_n, 1) = \mathbf{f}_{t_{i,n},k-\Delta k}$, with $n = [1,2,3]$ for the three triangle vertices. The two triangle matrices $\mathtt{T}_{i,k}$ and $\mathtt{T}_{i,k-\Delta k}$ can then be written as:

$$\mathtt{T}_{i,k} = \begin{bmatrix} \mathbf{f}_1^\top & \mathbf{f}_2^\top & \mathbf{f}_3^\top \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}, \tag{4.9}$$

$$\mathtt{T}_{i,k-\Delta k} = \begin{bmatrix} \mathbf{f}_1'^\top & \mathbf{f}_2'^\top & \mathbf{f}_3'^\top \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.10}$$

With these matrices available, the individual affine transformations $\mathtt{H}_i$ for each triangle t_i can easily be calculated by inverting one of the triangle matrices using Equation 2.29:

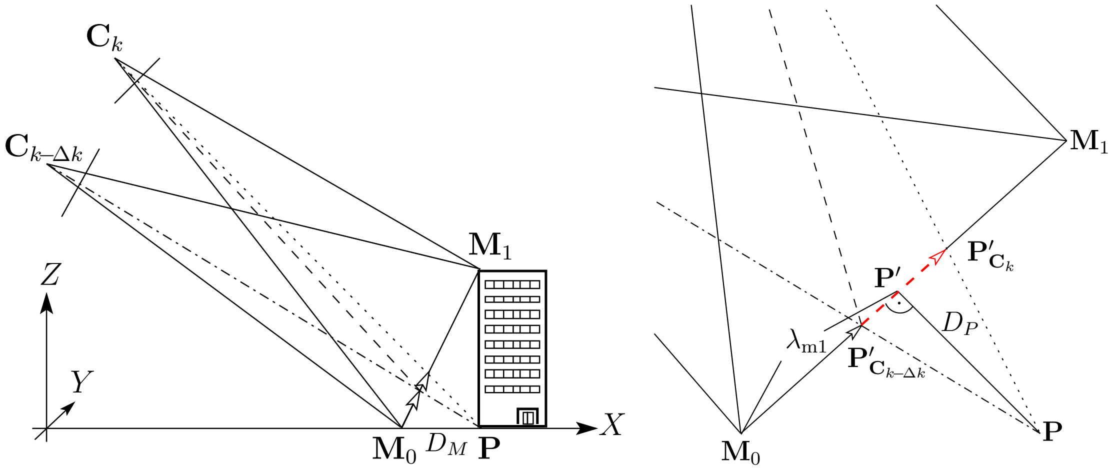
$$\mathtt{H}_i \cdot \mathtt{T}_{i,k} = \mathtt{T}_{i,k-\Delta k} \tag{4.11}$$

$$\mathtt{H}_i = \mathtt{T}_{i,k-\Delta k} \cdot \mathtt{T}_{i,k}^{-1}. \tag{4.12}$$

The individual displacement vector for each of the pixels inside the triangle t_i is retrieved by Equation 2.32 using the parameters of $\mathtt{H}_i$. The resulting vector defines the source coordinate of the pixel in the frame $k-\Delta k$, which has to be motion compensated to the position of the target pixel in the frame k.

As the resulting image coordinates are fractional, an appropriate interpolation filter must be used to determine the correct color and luminance of the intermediate pixel position from surrounding known samples. For performance reasons, this is achieved in this work by a two step filter: the first one doubles the resolution once for the entire source frame using a fixed 8-tap Lanczos filter. The second filter retrieves the final pixel properties by bi-linearly interpolating the upscaled samples at the source position using the individual fractional components as weighting.

4.4 Accuracy Analysis of the Mesh-based Approach



(a) Mesh nodes $\mathbf{M}_0$ and $\mathbf{M}_1$ represent a triangle plane projected from the image planes into the scene. $\mathbf{P}$ is an off-plane scene point observed by the two cameras $\mathbf{C}_{k-\Delta k}$ and $\mathbf{C}_k$.

(b) Cut-out of the mesh plane and the scene points. The red dashed arrow denotes the motion compensation error of $\mathbf{P}$.

Figure 4.6: Mesh deviation from real scene structure

In contrast to the global planar landscape model, the proposed mesh approach is locally adaptive. Although the relief displacement referenced to the ground plane does not change, the local adaptation of the mesh patches to the non-planar structure allows an individual motion compensation of image regions in the image plane projected from scene objects with heights different from the ground plane. This leads to a great reduction in aberrations between motion compensated frames compared to the single global plane model. However, as the mesh nodes are created from image intensity corners on the 2D image planes without knowledge of the scene structure or depths, they do not perfectly reflect discontinuities in the scene. To calculate the remaining motion compensation aberrations, the approximations done in the 2D-mesh approach are compared to the full 3D scene model in this chapter.

If perfectly tracked, the 2D mesh represents the projection of an undersampled 3D model of the scene. The undersampling is due to the mesh nodes not being placed perfectly at discontinuities in the 3D shape by the corner detector or due to undetectable feature displacements in the tracking part. An simplified example of the undersampling is given in Figure 4.6. The aberrations of the mesh is shown in this section on an exemplary triangle t_i, consisting of the mesh nodes $\mathbf{m}_0, \mathbf{m}_1$, and $\mathbf{m}_2$. Their scene points are created by the intersection of the view ray through the nodes with the first scene object and named $\mathbf{M}_0, \mathbf{M}_1$, and $\mathbf{M}_2$, respectively. In the example in Figure 4.6, the mesh node $\mathbf{M}_1$ is assumed to be perfectly placed at the top edge of the building, whereas the mesh nodes $\mathbf{M}_0$ is slightly off the base point $\mathbf{P}$ of the building. $\mathbf{M}_2$ is placed at the position of and $\mathbf{M}_0$, but shifted in y-direction. As mesh nodes are only created from successfully tracked 2D feature points, the position of each of the mesh nodes $\mathbf{M}$ in each of the cameras $\mathbf{C}$ is known and are therefore perfectly compensateable. Applied to the example in Figure 4.6, this translates to a simple displacement of the pixel values at the mesh node positions $\mathbf{m}_{0,k-\Delta k}$ and $\mathbf{m}_{1,k-\Delta k}$ from the image plane of camera $\mathbf{C}_{k-\Delta k}$ onto the associated positions $\mathbf{m}_{0,k}$ and $\mathbf{m}_{1,k}$ on the image plane of camera $\mathbf{C}_k$. The relationship between the 2D mesh node positions on the image planes and the 3D scene points are depicted as the solid black lines connecting the camera centers and the 3D mesh nodes in the figure.

The displacements of all remaining pixel positions in the image plane, however, are unknown and need to be interpolated from the known samples. The mesh-based motion compensation assumes a linear distribution of pixel displacements between the known mesh nodes. For this purpose, the mapping between the triangle shapes in both image planes is modeled as affine wherein the parameters of the transformation are determined by the displacements of the triangle vertices as described in the previous chapter. In three space, the affine mapping translates to a single plane through the 3D positions of the triangle vertices, viewed from both cameras $\mathbf{C}_k$ and $\mathbf{C}_{k-\Delta k}$. As each mesh patch has its own individual plane equation, this approximation of the real ground structure is called piece-wise planar. In Figure 4.6, the projected triangle patch is illustrated by the straight line connecting $\mathbf{M}_0$ and $\mathbf{M}_1$.

Each scene point with a 3D position directly on the surface of the triangle follows the affine model and is perfectly motion compensated into the other image plane, except for the inferior imaging properties of an affine to an projective mapping, which are neglected for now. All points off that surface, however, are motion compensated as being projected onto the surface of the triangle by the camera $\mathbf{C}_{k-\Delta k}$. In Figure 4.6b, $\mathbf{P}'_{\mathbf{C}_{k-\Delta k}}$ represents the projection of the off-plane point $\mathbf{P}$ onto the surface of the triangle by the camera center $\mathbf{C}_{k-\Delta k}$. If $\mathbf{p}_{k-\Delta k}$, as the 2D projection of $\mathbf{P}$ (and $\mathbf{P}'_{\mathbf{C}_{k-\Delta k}}$, as they are on the same ray) onto the image plane of camera $\mathbf{C}_{k-\Delta k}$ is motion compensated onto the image plane of camera $\mathbf{C}_k$ by an affine transformation, it becomes an estimation of the real position $\mathbf{p}_k$, here called $\widehat{\mathbf{p}}_k$. But as $\widehat{\mathbf{p}}_k$ was projected from $\mathbf{P}'_{\mathbf{C}_{k-\Delta k}}$ due to the linear assumption, it differs from $\mathbf{p}_k$, the real projection of $\mathbf{P}$ onto the image plane of camera $\mathbf{C}_k$. This difference is the model aberration of the piece-wise planar model compared to a real 3D structure and is called, similar to the comparable motion displacement, $\Delta\mathbf{p}_m$:

$$\Delta\mathbf{p}_m = \mathbf{p}_k - \widehat{\mathbf{p}}_k. \tag{4.13}$$

$\widehat{\mathbf{p}}_k$ was created by an affine projection of $\mathbf{p}_{k-\Delta k}$, so we can write:

$$\Delta\mathbf{p}_m = \mathbf{p}_k - \mathtt{H}_i\,\mathbf{p}_{k-\Delta k}. \tag{4.14}$$

$\mathtt{H}_i$ is the individual affine transformation matrix for the triangle t_i and is calculated from the known mesh nodes of the triangle using Equation 4.12:

$$\Delta \mathbf{p}_m = \mathbf{p}_k - \mathtt{T}_{i,k\text{–}\Delta k}\mathtt{T}_{i,k}^{-1}\ \mathbf{p}_{k\text{–}\Delta k}. \tag{4.15}$$

$\mathtt{T}_{i,k}$ and $\mathtt{T}_{i,k\text{–}\Delta k}$ consist of the 2D projected points creating the triangle t_i:

$$\mathtt{T}_{i,k} = \left[\mathbf{m}_{0,k}^{\top}\mathbf{m}_{1,k}^{\top}\mathbf{m}_{2,k}^{\top}\right], \qquad \mathtt{T}_{i,k\text{–}\Delta k} = \left[\mathbf{m}_{0,k\text{–}\Delta k}^{\top}, \mathbf{m}_{1,k\text{–}\Delta k}^{\top}, \mathbf{m}_{2,k\text{–}\Delta k}^{\top}\right]. \tag{4.16}$$

If we replace the image coordinates by the projection Equation 2.21, we get the approximation error expressed by the geometry of the scene points and mesh nodes:

$$\Delta \mathbf{p}_m = \mathtt{K}_k\mathtt{R}_k(\mathbf{P}-\mathbf{C}_k) - \mathtt{T}_{i,k\text{–}\Delta k}\mathtt{T}_{i,k}^{-1}\,\mathtt{K}_{k\text{–}\Delta k}\,\mathtt{R}_{k\text{–}\Delta k}(\mathbf{P}-\mathbf{C}_{k\text{–}\Delta k}), \tag{4.17}$$

wherein, according to Equation 3.42, each mesh node has the following projection equations:

$$\mathbf{m}_{n,k} = \mathtt{K}_k\mathtt{R}_k(\mathbf{M}_n-\mathbf{C}_k); \qquad \mathbf{m}_{n,k\text{–}\Delta k} = \mathtt{K}_{k\text{–}\Delta k}\mathtt{R}_{k\text{–}\Delta k}(\mathbf{M}_n-\mathbf{C}_{k\text{–}\Delta k}). \tag{4.18}$$

$\Delta \mathbf{p}_m$ can be scaled to $\Delta \mathbf{n}_m$ similar to Equation 3.17.

Equation 4.17 describes the accuracy of the pice-wise planar approximation, dependent on fixed coordinates of the scene points and camera positions. To keep the expression comparable to the global planar model, the position of $\mathbf{P}$ needs to be described by parameters, similar to the height of the building h of the scene point $\mathbf{P}_h$ and the position on the ground plane $\mathbf{P}_0$ in the Equations 3.27, 3.39, or 3.48. Applied to the mesh-based approach, this results in the following conditions: a point without approximation aberrations is a point on the surface of the triangle. A point off the surface has aberrations dependent on its distance. Therefore, $\mathbf{P}$ should be expressed in relation to the mesh surface as the distance to the surface D and the position of the dropped perpendicular foot $\mathbf{P}'$, similar to $\mathbf{P}_h$ being expressed by $\mathbf{P}_0$ and h.

We recollect that the mesh surface is created by projecting the tracked mesh nodes from the image plane into the 3D scene, and that each triangle in the mesh is determined by three surrounding mesh nodes, here denoted as $\mathbf{M}_0, \mathbf{M}_1$, and $\mathbf{M}_2$. $\mathbf{P}$ is assumed to be located in between them, if viewed from the direction of the cameras as illustrated in Figure 4.6.

The surface of the triangle in three space can be expressed as a plane in point-normal form, defined by the mesh node coordinates used as vertices of the triangle:

$$(\mathbf{x} - \mathbf{M}_0) \cdot \bar{\mathbf{n}} = 0. \tag{4.19}$$

The mesh node $\mathbf{M}_0$ is assumed to be the position vector whereas $\bar{\mathbf{n}}$ is a normal vector perpendicular to the triangle surface. It is calculated by the cross product of the two directional vectors given by the remaining mesh nodes:

$$\bar{\mathbf{n}} = (\mathbf{M}_1 - \mathbf{M}_0) \times (\mathbf{M}_2 - \mathbf{M}_0). \tag{4.20}$$

To keep adjacent formulas simpler, the unit normal is used here by normalizing the length of the vector to 1:

$$\bar{\mathbf{n}}_0 = \frac{\bar{\mathbf{n}}}{|\bar{\mathbf{n}}|}. \tag{4.21}$$

The distance D_P of $\mathbf{P}$ to the plane defined by the triangle can now easily be calculated using the Hessian normal form:

$$D_P = (\mathbf{P} - \mathbf{M}_0) \cdot \bar{\mathbf{n}}_0. \tag{4.22}$$

To finally eliminate the point $\mathbf{P}$, we have to determine the position $\mathbf{P}'$ of the foot of the dropped perpendicular of $\mathbf{P}$. It is the point on the triangle where the perpendicular line through $\mathbf{P}$ intercepts the plane. To get the foot point, the line equation is considered backwards with the position vector at $\mathbf{P}$ and its direction $\bar{\mathbf{n}}$ inverted. Inserting D_P as the scalar unit directly yields to the foot point:

$$\begin{aligned} \mathbf{P} &= \mathbf{P}' + D_P \cdot \bar{\mathbf{n}}_0 && (4.23) \\ &= \mathbf{P}' + D_P \cdot \frac{(\mathbf{M}_1 - \mathbf{M}_0) \times (\mathbf{M}_2 - \mathbf{M}_0)}{|(\mathbf{M}_1 - \mathbf{M}_0) \times (\mathbf{M}_2 - \mathbf{M}_0)|} && (4.24) \end{aligned}$$

The dropped perpendicular foot $\mathbf{P}'$ can furthermore be expressed by coordinates on the triangle surface instead of world coordinates. To do so, the triangle surface is expressed as a plane in parameter form. The position vector of the plane is given by one of the mesh nodes, here $\mathbf{M}_0$, whereas the differences to the remaining nodes specify the two directional vectors of the plane. These vectors are normalized to unit length and scaled by the two parameters λ_{m1} and λ_{m2} such that the position of the dropped perpendicular foot $\mathbf{P}'$ is obtained:

$$\mathbf{P}' = \mathbf{M}_0 + \lambda_{\mathrm{m1}} \cdot \frac{\mathbf{M}_1 - \mathbf{M}_0}{|\mathbf{M}_1 - \mathbf{M}_0|} + \lambda_{\mathrm{m2}} \cdot \frac{\mathbf{M}_2 - \mathbf{M}_0}{|\mathbf{M}_2 - \mathbf{M}_0|}. \tag{4.25}$$

By combining Equation 4.25 and Equation 4.24, the scene point $\mathbf{P}$ can completely be eliminated from the equation and is expressed by the position of the perpendicular foot (described by λ_{m1} and λ_{m2}) and the distance to the surface of the mesh D_P instead:

$$\mathbf{P} = \mathbf{M}_0 + \lambda_{\mathrm{m1}} \frac{\mathbf{M}_1 - \mathbf{M}_0}{|\mathbf{M}_1 - \mathbf{M}_0|} + \lambda_{\mathrm{m2}} \frac{\mathbf{M}_2 - \mathbf{M}_0}{|\mathbf{M}_2 - \mathbf{M}_0|} + D_P \frac{(\mathbf{M}_1 - \mathbf{M}_0) \times (\mathbf{M}_2 - \mathbf{M}_0)}{|(\mathbf{M}_1 - \mathbf{M}_0) \times (\mathbf{M}_2 - \mathbf{M}_0)|}. \tag{4.26}$$

By inserting Equation 4.26 into Equation 4.17, we get the final motion displacement dependent on the distance of a 3D space point to the triangle surface in three space.

The Figures 4.7 to 4.12 give examples for the accuracy. The following assumptions were made for the plots: $f = 80\text{mm}$, $C_z = 500\text{m}$, $v_x = 200\text{km/h}$, full frame full HD sensor, frame rate 30 fps, $\mathbf{P} = (160{,}0{,}0), \mathbf{M}_0 = (140{,}0{,}0), \mathbf{M}_1 = (160{,}0{,}20)$, and $\mathbf{M}_2 = (140{,}20{,}10)$. Herby, Figures 4.7 to 4.10 directly compare the mesh to the planar model by assuming a building on the ground plane with the height h. In contrast to the planar model, in which the approximation aberration on the foot of the building is zero and on the top of the building is maximal, the opposite is true for the mesh: for the example above, as the mesh node is placed directly on the top edge of the building, the approximation aberration is zero here, whereas the mesh node at the foot of the building is displaced by D_M, which makes the aberration for the foot of the building maximal as $\mathbf{P}$ has the largest distance to the mesh surface. Therefore, the individual maximum of the motion displacement is given in the figures.

Figure 4.7 gives the motion displacement in x-direction for varying D_M. If D_M is zero, the motion displacement for the mesh is also zero, as now the mesh surface lies directly on the wall of the building. Therefore the motion displacement is zero for every pixel projected from the wall. With increasing D_M, the motion displacement also increases; however, it is always lower than that of the planar model. More precisely, the planar model is the extreme case of the mesh for $D_M = \infty$. Moreover, if we assume a second mesh node being placed at the other edge of the building, the motion displacement stays zero for the full roof, whereas for the planar model the motion displacement stays constantly at the maximum value. In Figure 4.8, the motion displacement is plotted dependent on the speed of the aircraft for different tilt angles. It turned out the gains from the mesh compared to planar model are largest for small tilt angles θ, whereas, for larger tilt angles, the motion displacement for the mesh is still lower as for the planar model; however, the difference between them is smaller.

In Figure 4.9, the difference in motion displacement for the mesh and for the planar model are compared for different heights h of the building and ground positions X for the vertical photo case ($\theta = 0°$). If the camera is placed directly over the point under observation, the motion displacement is similar for both methods. However, if the distance is increased, the motion displacement decreases for the mesh whereas it stays constant for the planar model. In contrast, for a tilt angle θ of 15° the motion displacement in not independent of the ground position anymore but decreases with the distance of the object to the camera, as seen in Figure 4.11. Still, the motion displacement decreases faster for the mesh as for the planar model. Figure 4.11 shows the effect of the placement of P compared to the camera C_k and the mesh node $\mathbf{M}_0$, by shifting $\mathbf{P}$ with the distance D_P parallel to the line between $\mathbf{M}_0$ and $\mathbf{M}_1$. The plot shows that the effect of λ_{m1} is rather small compared to the distance D_P of the point to the mesh surface. The influence of the tilt angle θ on the motion displacement is provided in Figure 4.12. For small angles, the mesh performs noticeably better than the planar model. For the given example, however, the motion displacement increases for the mesh for larger values of θ while it decreases for the planar model. This is due to the mesh nodes being forced to their positions, leading to a disadvantageous triangle shape. In practice, however, the mesh node positions are selected automatically and the Delaunay triangulation avoids the creation of spike triangles.

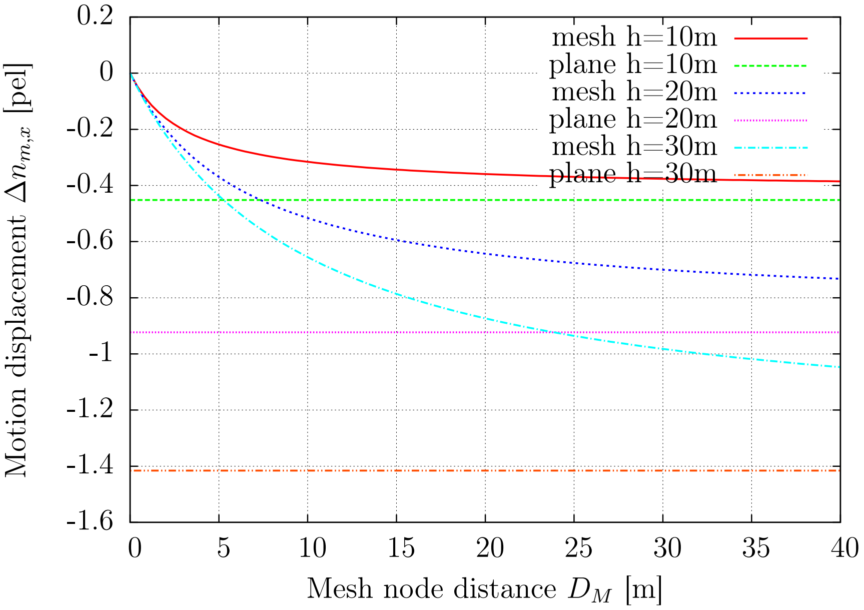


Figure 4.7: Comparison of the motion displacement $\Delta n_{m,x}$ in pel for the mesh and the planar model for different building heights h. For the mesh, the motion displacement is a function of the distance D_M between the closest mesh node $\mathbf{M}_0$ and the point at the foot of the building $\mathbf{P}$, such that $\mathbf{M}_0 = (160 - D_M, 0, 0)$.

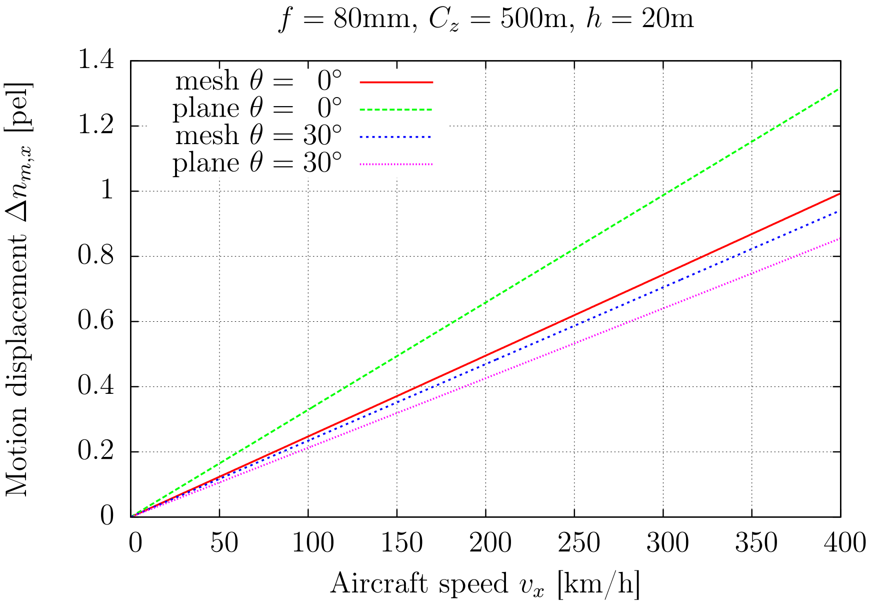


Figure 4.8: The motion displacement as a function of the aircraft velocity for several tilt angles.

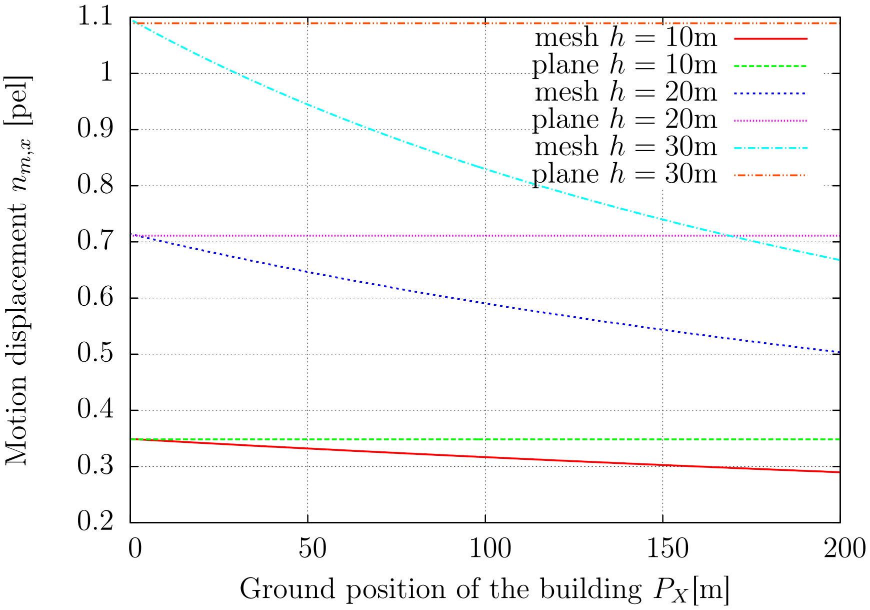


Figure 4.9: Motion displacement dependent on the ground position of the building (distance of the building to the nadir) for different building heights h and no tilt angle.

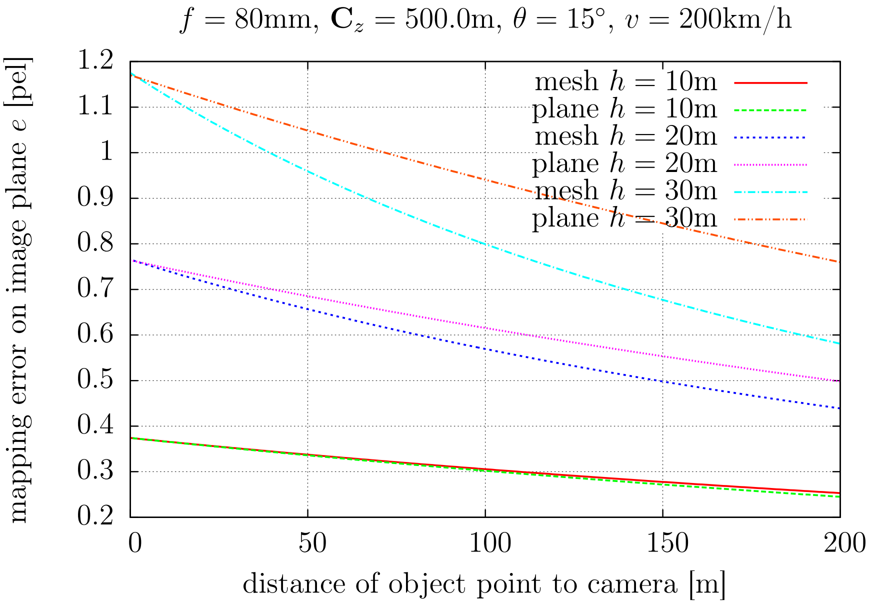


Figure 4.10: Motion displacement dependent on the ground position of the building (distance of the building to the nadir) for different building heights h and a tilt angle of $\theta = 15°$.

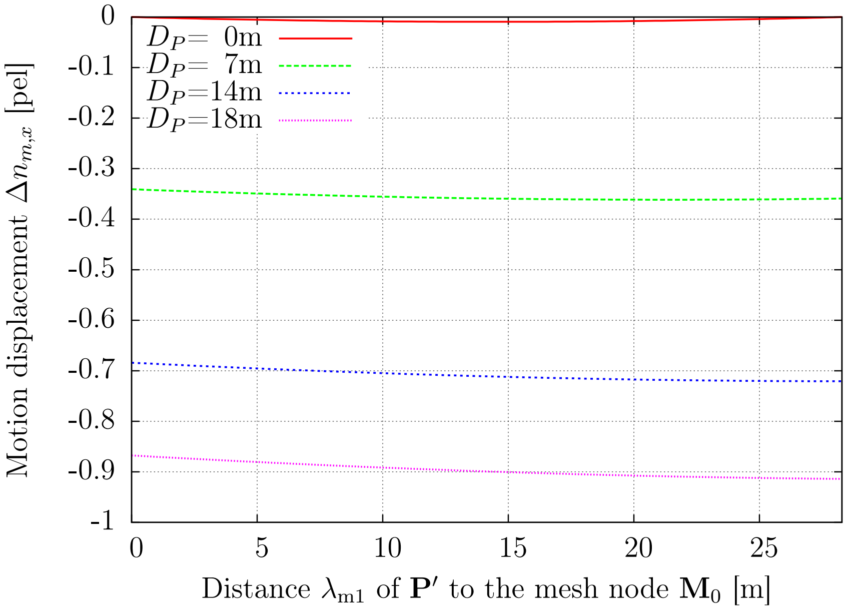


Figure 4.11: Motion displacement $\Delta n_{m,x}$ in pel as a function of the distances λ_{m1} between the closest mesh node $\mathbf{M}_0$ and the dropped perpendicular of $\mathbf{P}$ for different distances of the point $\mathbf{P}$ to the mesh surface D_P.

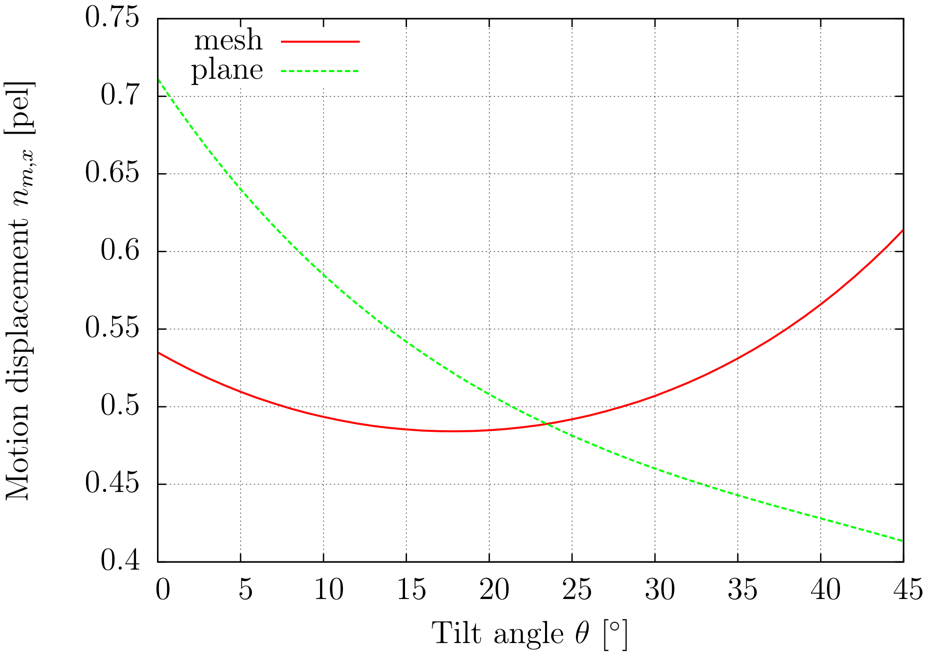


Figure 4.12: Motion displacement for a building of height $h = 20\text{m}$ for various tilt angles.

5 Experiments

In this chapter, different techniques and methods of moving object detection in moving camera scenarios are compared. In the first part, the motion detection is performed entirely on the (sparse) optical flow taken from tracked feature points. Hereby, the state-of-the-art RANSAC approach, using either a planar (homography-based) single plane background motion model of Chapter 2.5 or a `F`-matrix conformance test (Chapter 2.4), competes against the proposed cluster filter approach of Chapter 4.2 and the motion parallax predictor classifier from Equation 3.55 in Chapter 3.6. Moving objects are detected as outliers of the background motion.

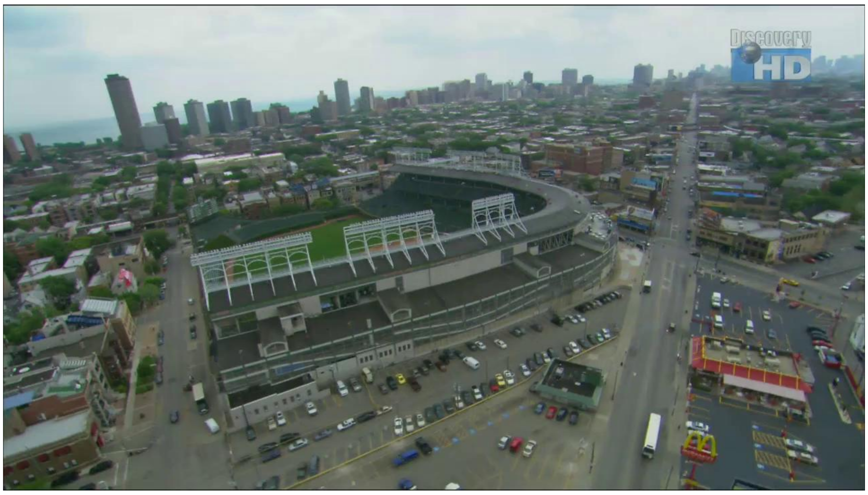
The second part of the chapter deals with the global motion compensation of pixels between the image planes of different cameras. Here, the proposed mesh-based locally adaptive global motion compensation of Chapter 4 is compared to the state-of-the-art planar (homography-based) single-plane global motion compensation, introduced in Chapter 3. As the mesh-based approach relies heavily on outlier-free mesh nodes, the motion compensation is first performed by using manually selected motion vectors of the background only, which are free of outliers and moving objects. To show the influence of the outlier detection performance on the global motion compensation, the motion compensation is additionally performed and evaluated using the background motion vectors automatically classified by the outlier detection algorithms of Chapter 5.1.

For the tests, aerial image sequences from public datasets as well as television recordings are investigated. Figure 5.1 gives the different test sequences and properties. All sequences share HD resolution and an aerial recording scenario of a non-planar scene containing moving objects.

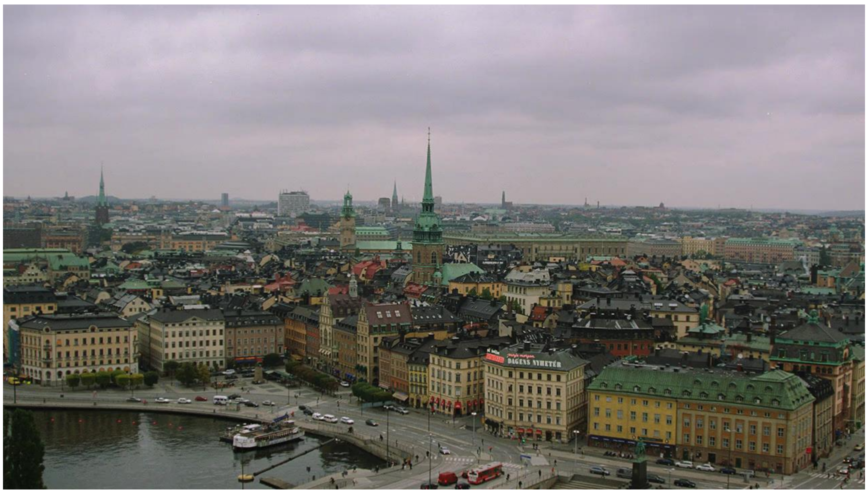
5.1 Motion Vector Classification

In this section, well known outlier detection algorithms are compared to the proposed motion parallax predictor classifier and to the cluster filter for their ability to distinguish between motion coming from static objects such as the ground plane or buildings and non-static motion such as moving objects or false tracks on a feature basis. The feature positions are determined by the Shi-Thomasi corner detector from Chapter 2.7.1 and the correspondences to a previous frame (with a variable frame skip distance of Δk frames) are determined by the KLT feature tracker (Chap-

(a) Stockholm, 1080p25, 157 frames [61]



(b) Chicago, 720p25, 472 frames [62]



(c) OldTownCross, 720p50, 500 frames [58]

Figure 5.1: Aerial video test sequences

ter 2.7.2). The maximum amount of features was set to 4000 for all sequences, of which around 3800 are successfully tracked on average. The same set of feature positions and correspondences is used for all the algorithms to get comparable results.

The `H`-matrix homography mapping (in the tables referred to as "H matrix") as well as the `F`-matrix conformance test (in the tables referred to as "F matrix") are used in RANSAC as the state-of-the-art algorithms for outlier detection. They are rated by their classification result against the proposed cluster filter, whereby the three different cost functions from Equation 4.5 to 4.7 for determining the similarity of motion vectors are evaluated. In the tables, Equation 4.5 is referred to as "CF max", Equation 4.6 as "CF max scale", and Equation 4.7 as "CF magnitude", respectively. Additionally, the motion parallax predictor classifier from Chapter 3.6 (in the tables referred to as "MPP") is used for outlier detection, with the maximum distance threshold set to $T_d = 2.2$ pel (Equation 3.55). As the motion parallax predictor classifier needs camera positions and orientations as input, these parameters were estimated from the video sequences by the Voodoo Camera Tracker [38]. The unknown scaling factor was determined by scaling the 3D scene coordinates to known reference points (e.g. the size of the church in the Stockholm sequence). In a practical setup, these values are provided by the GPS/INS system onboard the aerial vehicles.

The high resolution sequence "Stockholm" has been tested with a frame skip distance Δk (meaning the number of frames between the current frame k and the previous frame $k - \Delta k$, which are used to determine the feature correspondences) of 6 and 9. For the other sequences, a fixed value of $\Delta k = 6$ was used, as for the same camera motion the displacement for the lower resolution video in pel is smaller. Concerning the performance evaluation, the following criteria have been evaluated to compare the output of each method against a manually labeled reference:

$$
\begin{aligned}
&\text{True positive rate or recall:} && \text{tp-rate} = \frac{t_p}{t_p + f_n} \\
&\text{True negative rate or specificity:} && \text{tn-rate} = \frac{t_n}{t_n + f_p} \\
&\text{False positive rate or fallout:} && \text{fp-rate} = \frac{f_p}{f_p + t_n} \\
&\text{False negative rate or miss rate:} && \text{fn-rate} = \frac{f_n}{f_n + t_p} \\
&\text{Positive predictive value (ppv) or precision:} && \text{ppv} = \frac{t_p}{t_p + f_p} \\
&\text{Negative predictive value (npv):} && \text{npv} = \frac{f_n}{f_n + t_n} \\
&\text{Combined measure:} && \text{accuracy} = \frac{t_p + t_n}{t_p + t_n + f_p + f_n}
\end{aligned}
\tag{5.1}
$$

Table 5.1: Classification results in % for the sequence Stockholm, $\Delta k = 9$

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
CF max scale	**99.95 ± 0.1**	**85.96 ± 3.5**	**14.04**	**0.05**	**99.37**	98.71	**99.35**
CF max	**99.96 ± 0.1**	80.26 ± 4.3	19.74	**0.04**	99.12	**98.90**	99.12
CF magnitude	99.78 ± 0.2	83.36 ± 3.6	16.64	0.22	99.26	94.53	99.08
MPP	99.64 ± 0.2	82.16 ± 4.8	17.84	0.36	99.20	90.98	98.89
F matrix	92.22 ± 5.4	84.75 ± 6.1	15.25	7.78	99.26	32.79	91.90
H matrix	93.30 ± 5.4	85.21 ± 4.2	14.79	6.70	99.30	36.29	92.96

Table 5.2: Classification results in % for the sequence Stockholm, $\Delta k = 6$

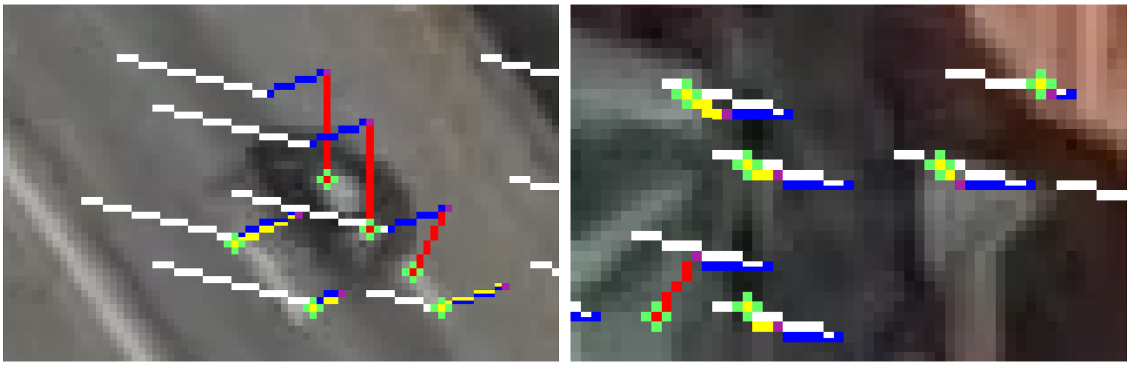
	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
CF max scale	**100.00 ± 0.0**	79.26 ± 5.8	20.74	**0.00**	**99.26**	**100.00**	**99.28**
CF max	**100.00 ± 0.0**	72.81 ± 7.5	27.19	**0.00**	99.04	**100.00**	99.06
CF magnitude	99.79 ± 0.2	79.39 ± 5.4	20.61	0.21	**99.27**	93.00	99.08
MPP	99.93 ± 0.1	79.43 ± 4.9	20.57	0.07	**99.27**	97.48	99.22
F matrix	94.18 ± 3.8	79.87 ± 10.8	20.13	5.82	99.24	32.91	93.68
H matrix	97.32 ± 4.0	**80.03 ± 4.4**	**19.97**	2.68	**99.27**	51.64	96.72

Table 5.3: Classification results in % for the sequence Chicago, $\Delta k = 6$

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
CF max scale	99.69 ± 0.4	**78.23 ± 10.0**	**21.77**	0.31	**99.67**	79.26	**99.37**
CF max	**99.80 ± 0.2**	62.47 ± 10.8	37.53	**0.20**	99.44	**82.66**	99.25
CF magnitude	98.80 ± 3.2	71.76 ± 11.5	28.24	1.20	99.57	47.46	98.40
MPP	98.48 ± 1.1	53.07 ± 16.8	46.93	1.52	99.40	30.65	97.91
F matrix	93.93 ± 4.5	60.28 ± 13.1	39.72	6.07	99.37	13.02	93.43
H matrix	85.95 ± 4.9	63.26 ± 11.4	36.74	14.05	99.36	6.35	85.61

Table 5.4: Classification results in % for the sequence OldTownCross, $\Delta k = 6$

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
CF max scaled	99.33 ± 0.3	85.07 ± 13.8	14.93	0.67	99.40	83.55	**98.78**
CF max	**99.54 ± 0.2**	79.23 ± 14.3	20.77	**0.46**	99.17	**87.24**	**98.76**
CF magnitude	99.32 ± 0.3	80.09 ± 14.3	19.91	0.68	99.20	82.40	98.58
MPP	94.59 ± 0.9	83.08 ± 15.2	16.92	5.41	99.29	38.05	94.15
F matrix	97.01 ± 2.9	67.77 ± 23.5	32.23	2.99	98.69	47.53	95.88
H matrix	86.95 ± 4.3	**89.93 ± 9.0**	**10.07**	13.05	**99.54**	21.60	87.06



(a) The car is correctly classified as moving (red displacement vector) since the current feature position (green cross) is far away from the predicted epipolar line segment (white). The two displacement vectors placed on the road (yellow) are correctly classified as background as the current feature position is close to the white line.

(b) The position of the feature (green cross) on the epipolar line segment (white line) represents the height of the 3D point in the scene. The further the green cross is away from the end of the epipolar line segment touched by the blue line (displacement vector of the ground level), the higher the scene point is (here: cut out of the church tower in Stockholm).

Figure 5.2: The classification of the motion parallax predictor classifier is based on the distance of the measured feature position in the current frame (green cross) to the predicted epipolar line segment (white line). The blue line represents the displacement vector of an object on the ground plane, the yellow and red lines are the feature displacement vectors, classified as background or moving object, respectively.

The standard deviation is only given for the tp- and the tn-rate, because the standard deviations of the fp- and the fn-rate are equal to that of the former as they can be computed from each other by fp-rate = 1−tn-rate and fn-rate = 1−tp-rate, respectively.

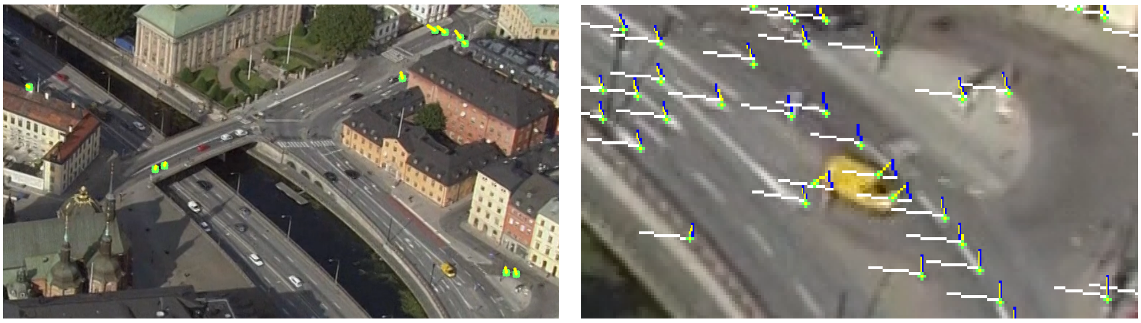
Tables 5.1 and 5.2 give the classification results for the "Stockholm" sequence using a frame skip distance Δk of 9 and 6 frames, respectively. The cluster filter approach performs best for this sequence by classifying nearly 100% of the background motion vectors correctly. Using the scaled maximum displacement, "CF max scale" also has the lowest false positive detections for $\Delta k = 9$. For $\Delta k = 6$ the `H`-matrix test reaches the lowest false positive rate for the Stockholm and the OldTownCross (Table 5.4) sequence; however, this is achieved at the expense of the true positive detection rate and the accuracy, which fall behind all the cluster filter methods and, for the OldTownCross sequence, also performs worst. Concerning the Chicago sequence in Table 5.3, the `H`-matrix test performs worst in the sense of recall and accuracy with a npv of only 6.35%. The cluster filter on the other hand classifies up to 99.8% of the true positives correctly with, at the same time, the best precision and accuracy of 99.37% and 99.37%, respectively.

The motion parallax predictor classificator comes close to the performance of the cluster filter. The slightly poorer classification results are due to moving objects moving along the epipolar line not being detectable. This is shown in Figure 5.3(b): as soon as the yellow van starts to turn left, it is not detected anymore as the feature coordinates of features placed on the van in the frame k come to lie on the predicted constrained epipolar line (white) in the frame $k-\Delta k$. Figure 5.2 shows the characteristics and the classification decisions of the motion parallax predictor more detailed.

Concerning the Chicago and the OldTownCross sequence, the estimation of the camera parameters through the Voodoo Camera Tracker fails in some way, making the camera positions and orientations imprecise. As a result, the motion parallax predictor is not able to predict the motion parallax of static objects correctly. This reflects in the tables as the high false positive rate for motion parallax predictor classifier for the Chicago sequence as well as the high false negative rate for the OldTownCross sequence. For the latter sequence, a problem in feature tracking was observed additionally, effecting all methods. Due to repetitive structure such as the windows in Figure 5.3(c), the correspondence analyses often fails to retrieve the correct displacement of feature points. This results in single false tracks possibly detected as moving objects as well as large areas of outliers, like the bottom right part of Figure 5.3(c). The latter mostly effects the cluster filter approaches, as no correct motion information is available for some areas of the frame. The mesh assumes a linear behavior in this case, which might or might not be correct.

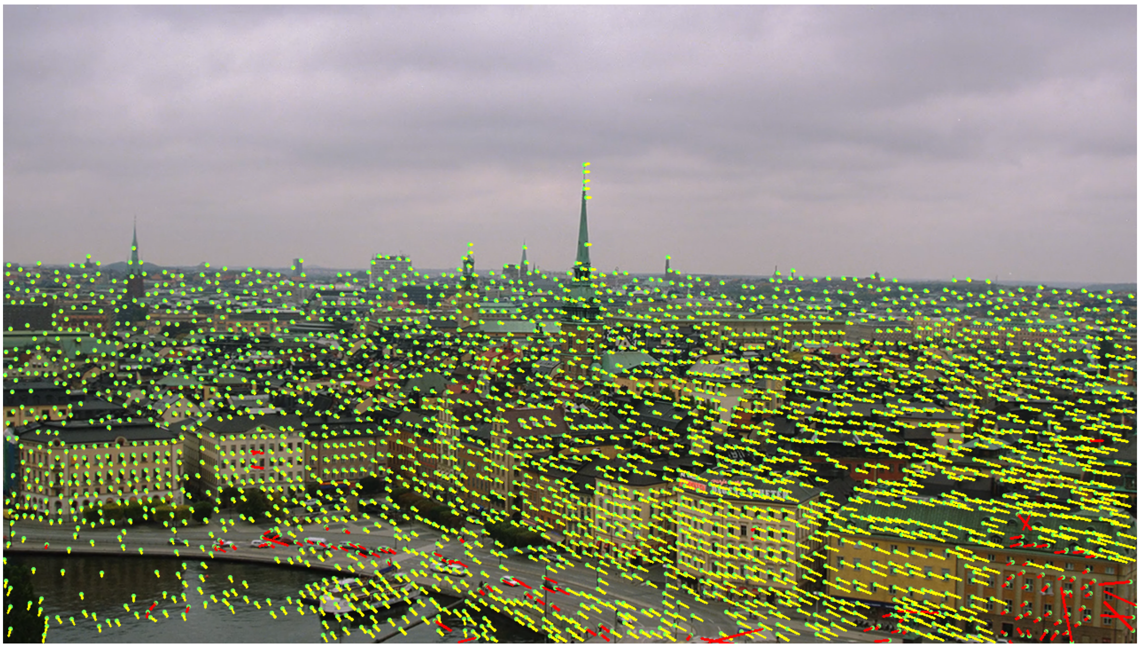
The state-of-the-art classificators lag behind a lot in the sense of true positive detections. Moreover, they have a far bigger variance in detection performance over the frames. This is presented in Figure 5.5. Whereas the proposed cluster filter and motion parallax predictor classifier stay constantly at high levels throughout the sequences, the state-of-the-art methods have a high frame-to-frame variance in the classification performances. For the homography, this is due to the model not being able to describe the motion of the non-planar scene. Therefore, the plane described by the homography is fitted somehow into the scene on a least-squares basis, which might not represent the real ground plane at all, as e.g. shown in Figure 5.4(a) where large areas of the ground plane are falsely classified as moving.

Concerning the fundamental matrix conformance test, the challenge lies in the robust estimation of the `F`-matrix under non-optimal conditions which often failed during the test. This results in areas being misclassified as well as wide area single classification errors as in Figure 5.4(b). The unreliable estimation of the `F`-matrix can be avoided by using known camera parameters for the calculation. However, as for the motion parallax predictor, the camera parameters must be correctly and robustly estimated onboard, which might not always be possible. But even if the data is available, the motion parallax predictor provides constraints on the epipolar



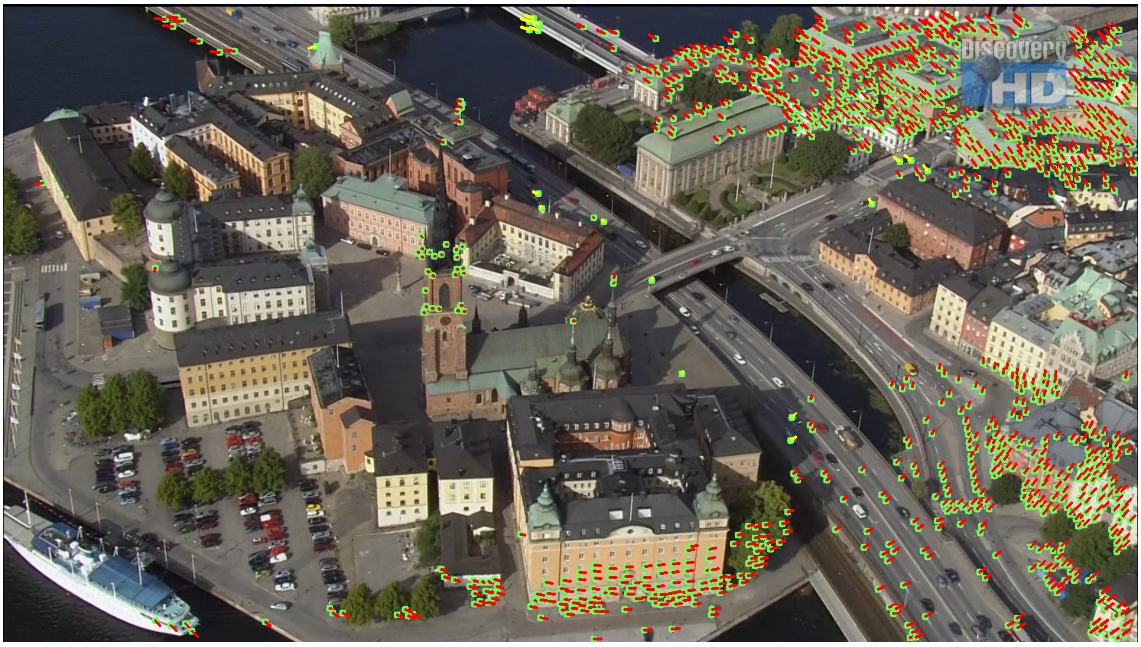
(a) Stockholm, cluster filter. Sometimes misdetection of slowly moving vehicles (green dots).

(b) Stockholm, motion parallax predictor classifier. Slow vehicles are not detected if they move along the predicted epipolar line segment (green cross close to the white line at the yellow van).

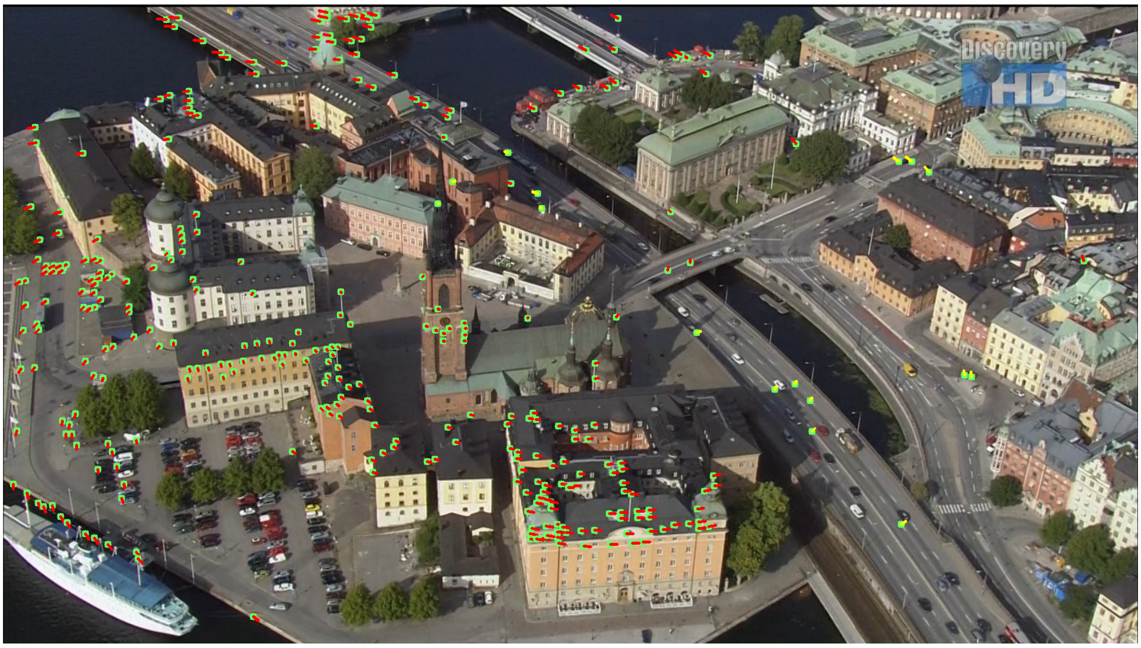


(c) OldTownCross, manual labeling. The tracking fails in some parts (bottom right, red lines) due to repetative structure (windows). Moreover, the individual movment of the waves on the sea (bottom left) makes the manual and automatic classification difficult. Displacement vectors of the background are colored yellow, outliers and moving objects are colored red.

Figure 5.3: Problems of the proposed methods and the feature tracker

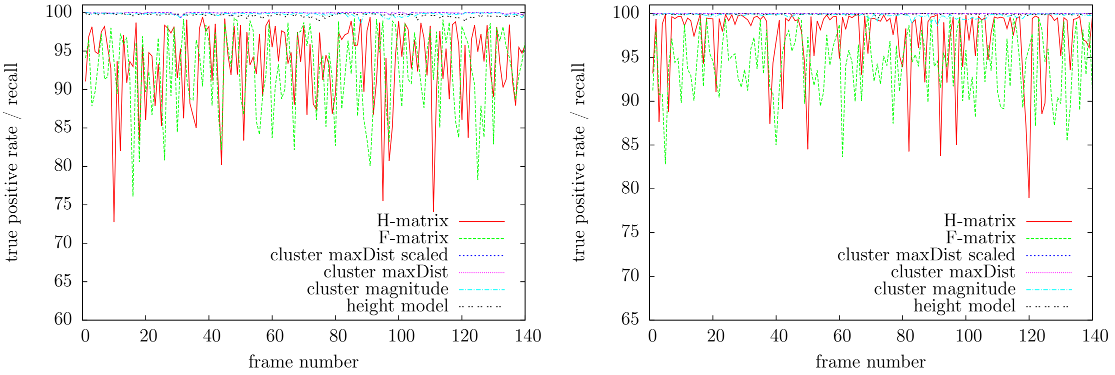


(a) Stockholm, homography classifier. Upper right and bottom regions are marked as moving due to the homography plane not being fitted onto the ground plane (mostly false positives).

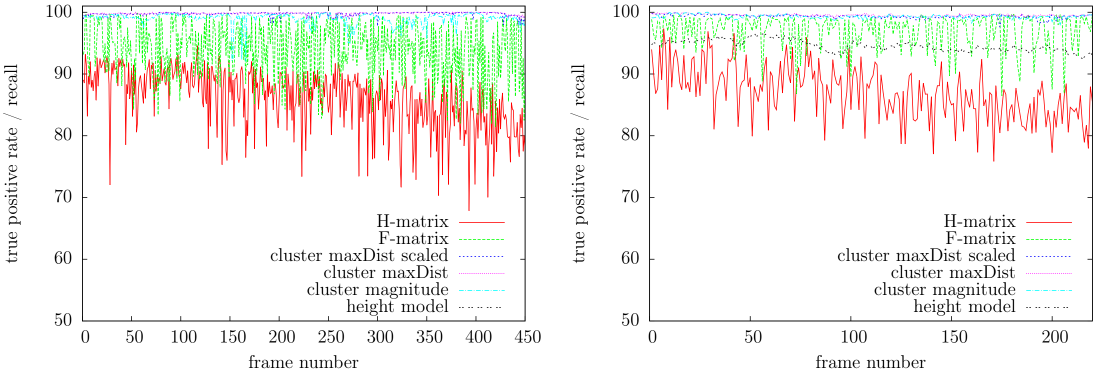


(b) Stockholm, fundamental matrix classifier. Unreliable determination of the `F`-matrix results in large areas of false classifications (false positives as well as false nagatives).

Figure 5.4: State-of-the-art classification results, difference to the manual labeling. Green is the feature location, a yellow displacement vector marks a false positive (background if actually moving), a red one marks a false negative (moving instead of background).



(a) Stockholm, $\Delta k = 9$ (b) Stockholm, $\Delta k = 6$



(c) Chicago, $\Delta k = 6$ (d) OldTownCross, $\Delta k = 6$

Figure 5.5: Reliability for true positive classification (correctly classified as background motion) over the different sequences.

lines additionally to the `F`-matrix test by defining a valid range of possible feature positions in the second camera view (which is equivalent to defining a valid range of Z-coordinates for the scene points in camera coordinates, see Figures 2.7 and 5.3(b)). This largely increases the detection performance and should therefore be used instead.

The results also reflect in the absolute values given in Table 5.5. Although the average false positives per frame are more or less similar throughout the methods and the sequences, the amount of false negatives is clearly higher for the state-of-the-art outlier classifier. Moreover, the distribution of the false classifications differs between the algorithms. The majority of the false positives is placed at moving cars for both the cluster filter and the motion parallax predictor classifier. But while for the motion parallax predictor classifier they are placed on cars moving along the epipolar line, they are located on slowly moving cars for the cluster filter instead.

Table 5.5: Average absolute difference to the manual labeling per frame

	Stockh. $\Delta k = 9$		Stockh. $\Delta k = 6$		Chicago		OldTownCross	
	fp	fn	fp	fn	fp	fn	fp	fn
CF max scaled	**22.9**	1.8	27.3	**0.0**	**8.4**	10.2	11.7	21.8
CF max	32.2	**1.5**	35.8	**0.0**	13.4	**6.8**	12.2	**16.7**
CF magnitude	27.1	7.9	27.1	7.9	10.1	31.0	11.7	21.8
MPP	29.1	13.3	27.1	2.7	0	∞	13.8	326.2
F matrix	24.9	283.3	26.5	214.5	13.0	147.5	24.4	99.9
H matrix	24.1	244.0	**26.3**	98.7	12.2	340.8	**6.5**	510.5

The homography on the other hand is able to detect objects moving at very low speeds, but totally fails in other regions at the same time where the underlying planar assumption is violated or the model is misfitted. In such cases, large coherent areas of false classifications are produced. The fundamental matrix on the other hand mostly creates wide spread areas with only single misclassifications.

The cluster filter, using the scaled maximum displacement threshold, reaches best accuracy throughout the sequences and almost always top precision by only making rough assumptions about the scene or camera motion. However, true positive detections might be lost in single frames as in Figure 5.3(a) if their local motion difference to the surrounding background motion is small. This can easily be worked around by taking the temporal context into account.

5.2 Motion Compensation Performance

After sorting out the outliers, the motion compensation of the background pixels from the previous frame to the current frame can be performed. This generates two images shot from the same camera position but at two different moments in time. The actual compensation of the camera position is performed by displacing the pixels of the previous image as if the scene was shot from the position of the camera at the current frame. As only the global motion (that of the background pixels) is compensated, changes of locally moving objects are still visible. This enables the usage of static camera based motion detection algorithms within a moving camera setup. The performance of the static motion detection algorithms hereby depends strongly on the residual error of the background motion compensation. To evaluate the performance of the proposed locally adaptive mesh-based global motion compensation from Chapter 4 with the single-plane state-of-the art homography-based method from Chapter 2.5, the residual error by means of image differences between the current frame and the compensated previous frame are evaluated. If performing perfectly, only moving objects such as cars or pedestrians should be visible in the image differences and all static objects should be hidden.

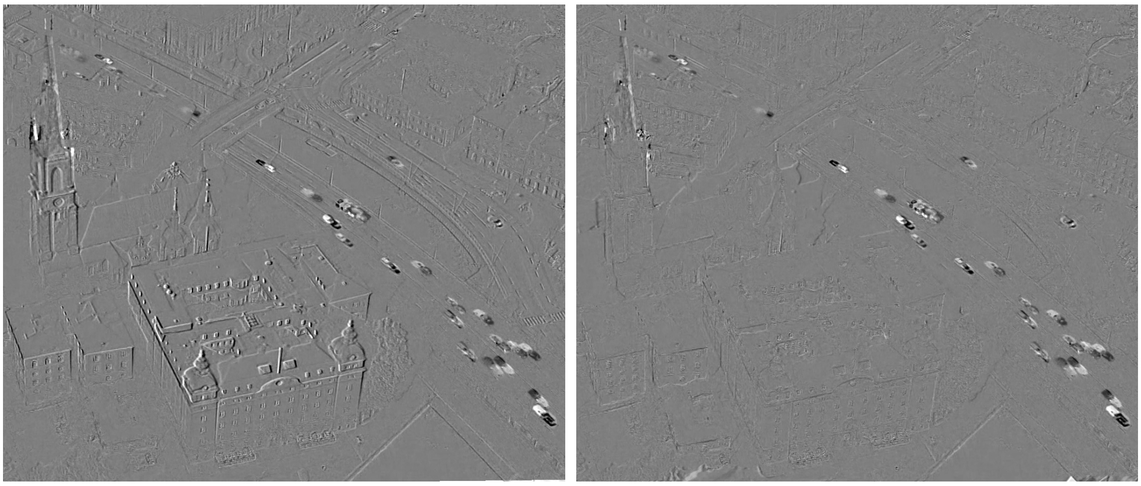
For the compensation, the integer pixel positions of the current frame are usually mapped to sub-pel positions of the previous frame. To retrieve the image and color intensities of the sub-pel positions, they must be interpolated from the known integer positions. In this work, a two-stage filter is used to achieve reasonable complexity: the first filter creates the half-pel positions in x- and y-direction (which quadruples the amount of pixels) using a fixed 8-tap Lanczos filter. The second bi-linear filter calculates the individual fractional sub-pel position of the upscaled reference frame. The actual mapping is done using Equations 2.33 and 2.30 for the homography and Equations 4.12 and 2.32 for the mesh.

5.2.1 Ideal Mesh-based Motion Compensation

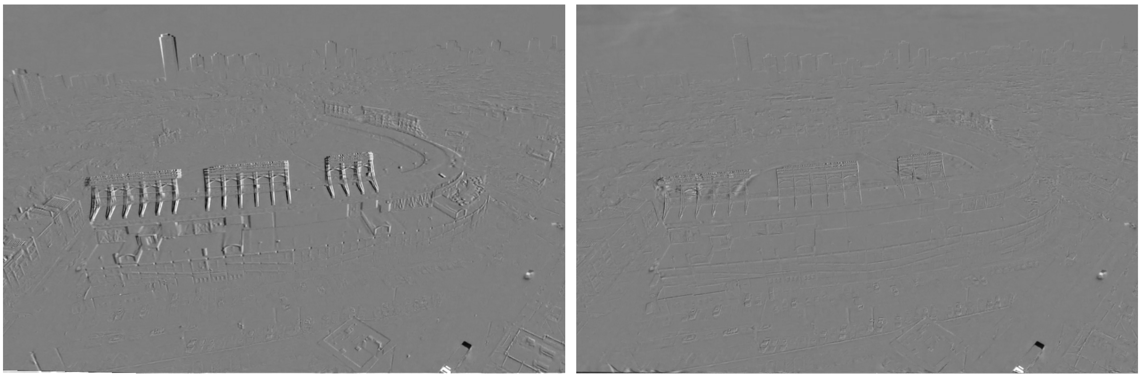
Figure 5.6 shows a comparison of the image differences between the current frame and the motion compensated frame using the homography-based global motion compensation and the mesh-based approach. Both methods get manually classified displacement vectors of only the background motion as input to achieve the best possible results and to evaluate the performance of the compensation method only and not that of the outlier detection. It can be seen that the proposed mesh-based motion compensation produces far less differences in the area of non-moving high structures but preserves the moving objects (cars) at the same time throughout the test sequences. Also, the individual affine mapping per mesh patch is accurate enough to compensate the small changes between the frames, and the mesh resolution is fine enough to handle the motion discontinuities at object borders. However, some remaining artifacts of the incorrectly compensated background motion is still visible. Concerning the Stockholm sequence in Figure 5.6(b), this is mostly due to missing feature points on the unstructured church roof and, as it applies also to the OldTownCross sequence in Figure 5.6(f), newly occurring background behind buildings. The constructions on top of the stadium in Figure 5.6(d) are problematic, too, as they are transparent and feature points were only detected on the construction but not on the background behind.

5.2.2 Fully Automatic Mesh-based Motion Compensation

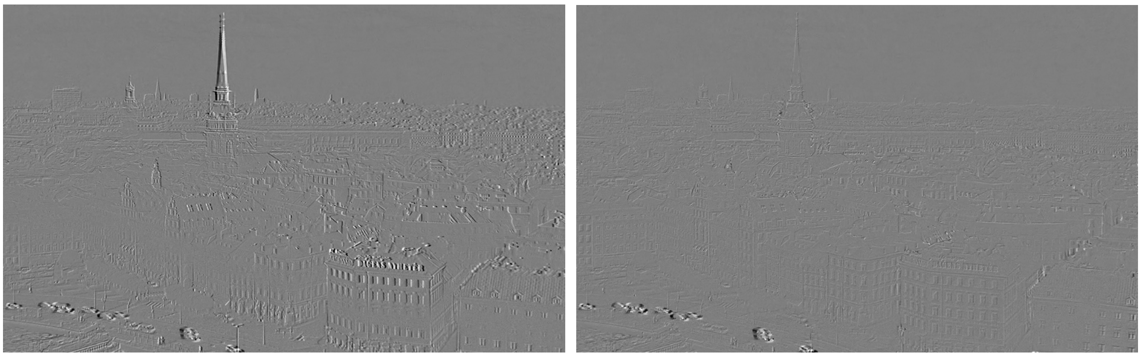
In the previous section, the mesh-based motion compensation was performed using manually labeled feature displacements of the background only (ideal mesh). In a real system, the feature detection as well as the outlier elimination and motion compensation has to be done automatically. Hence, in this section, the mesh-based motion compensation is computed using the outlier detectors from Chapter 5.1. As



(a) Stockholm, homography (b) Stockholm, mesh



(c) Chicago, homography (d) Chicago, mesh



(e) OldTownCross, homography (f) OldTownCross, mesh

Figure 5.6: Closeup of image differences between the current frame k and a motion compensated frame $k-\Delta k$. Nearly all of the unwanted image differences on the non-planar background objects vanish with the mesh method. Some errors are still visible due to missing feature points or newly occurring background.

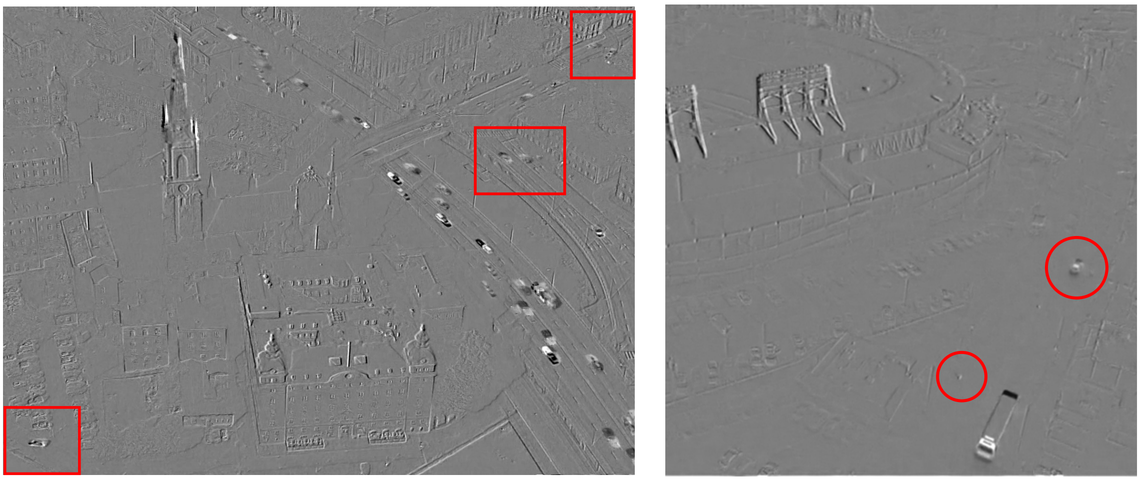
only the cluster filter, the motion parallax predictor classifier, and the fundamental matrix allow a classification of non-planar scenes, only these are evaluated with the mesh motion compensation. For the cluster filter, only the best performing similarity cost function "CF max scaled" is listed here.

The cluster filter performs close to the ideal mesh, despite the mis-classifications denoted in Table 5.5. This effects the motion compensation in a way that displacement vectors classified as background but actually located on moving objects lead to the moving objects being motion compensated. As a result, the moving objects might get lost in the image differences. This is shown in the Figures 5.7(c) and 5.7(d) by the red boxes/circles, where single moving objects are not visible compared to the homography or the ideal mesh. But as long as this happens only in single frames and with varying mis-classifications, it can easily be handled in a post processing step of the outlier detection or in the image differences-based motion detection. The overall compensation performance of the background is nearly identical to that of the ideal mesh in Figures 5.7(e) and 5.7(f). The homography instead has large discrepancies in the whole area of the stadium in Figure 5.7(b) as well for the full frame in Figure 5.7(a).

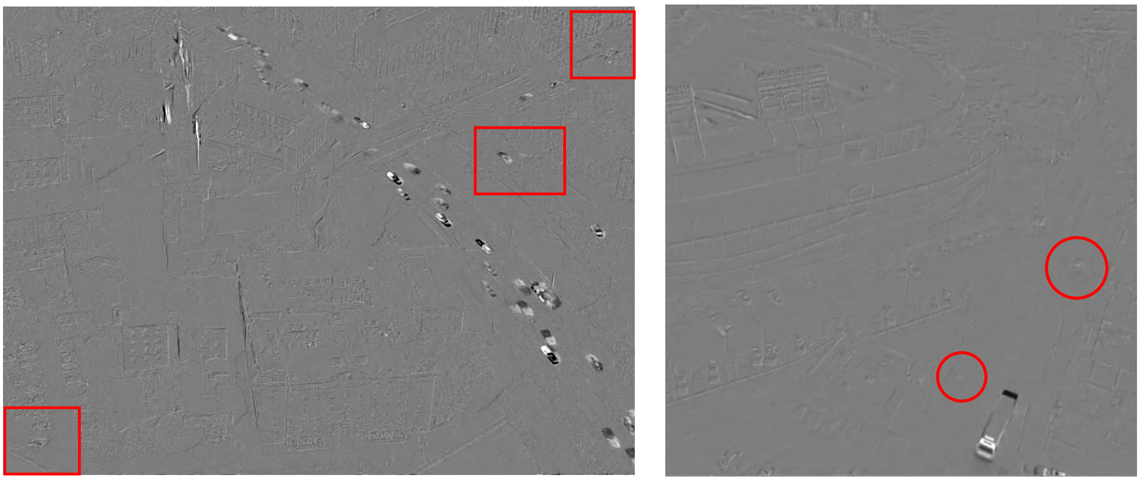
The `F`-matrix classificator has large problems with single mis-classifications spread over the frame. This is a big problem for the mesh-based motion compensation, as exemplified in Figure 5.8(b). Because of the false positives with high differences in angle and value to the local surrounding, the mesh is noticeably deformed, yielding an error in the motion compensation in the size of all mesh patches connected to that erroneous node. Concerning the cluster filter, this is prevented by design: a smooth motion vector field is the basic idea of this approach and large changes between neighboring displacement vectors lead to the vector being marked as an outlier. Although this decision might sometimes be incorrect, the area surrounding this vector is still being compensated - with a displacement interpolated through the overlying mesh patch from the neighboring background displacement vectors. That is why single false negatives of the cluster filter are not that harmful to the mesh-based motion compensation result.

5.3 Moving Object Detection Performance

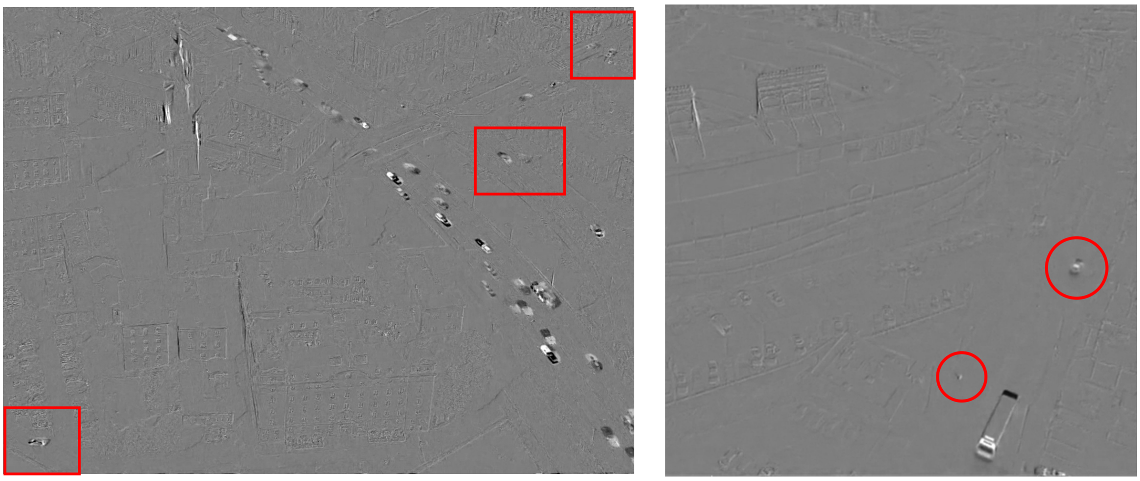
To detect moving objects from the motion compensation residual, the simple motion detection algorithm from Chapter 2.6 was used to classify each pixel into belonging to a moving object or to the background. The output after noise filter, binarization and erosion is given in the Figure 5.9 for the Stockholm Sequence and Figure 5.10 for the Chicago sequence. If available, the output was evaluated against a pixel-wise manual labeling of moving objects in the scene.



(a) Stockholm frame 1, homography (b) Chicago, homography

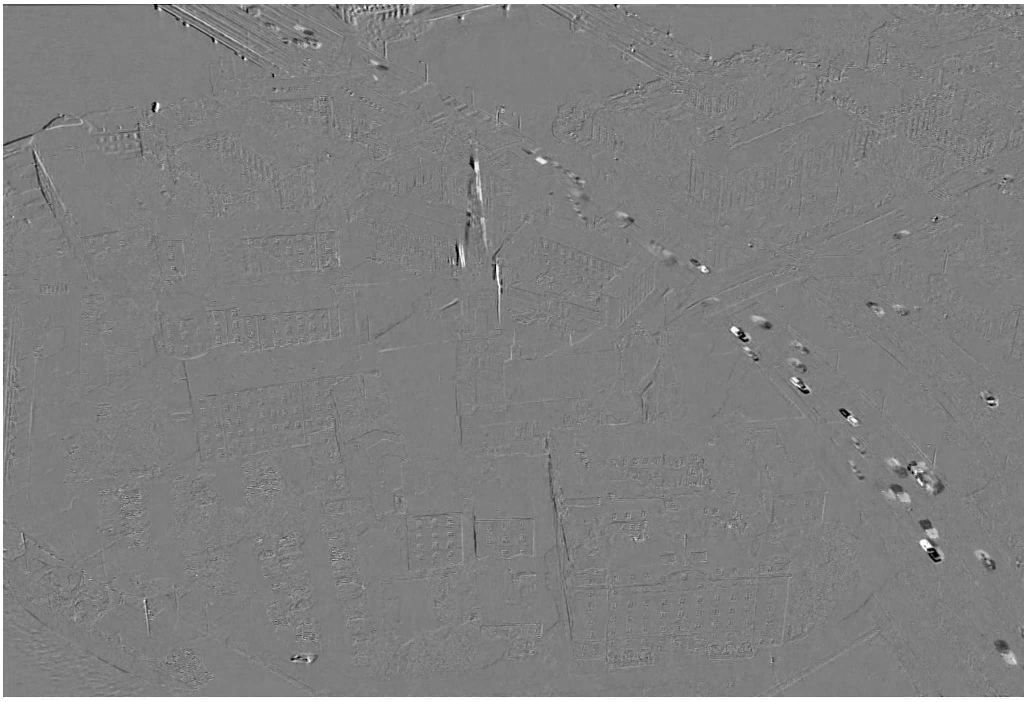


(c) Stockholm frame 1, mesh cluster filter (d) Chicago, mesh cluster filter

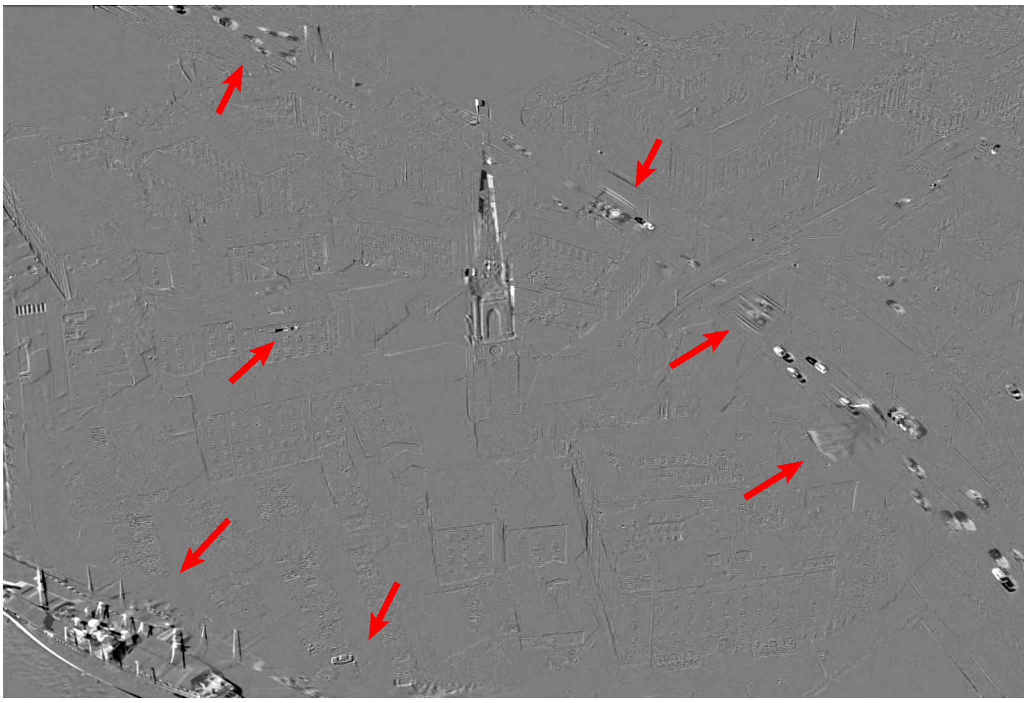


(e) Stockholm frame 1, ideal mesh (f) Chicago, ideal mesh

Figure 5.7: Closeups of the residual for different motion compensation methods and outlier detectors. The small car and the animal (red circles) vanish in single frames with the cluster filter approach. The homography produces high compensation aberrations at the stadium.



(a) Stockholm frame 1



(b) Stockholm frame 50

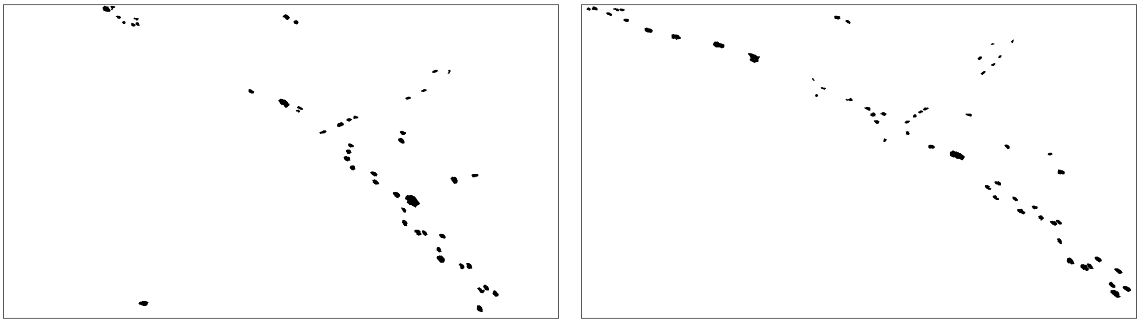
Figure 5.8: Mesh-based motion compensation using the fundamental matrix outlier classifier. Although it might work well in some frames and/or regions (a) (comparable to the motion parallax predictor classifier results), it might produce large errors if the F-matrix was incorrectly estimated (b).

As the pixel-wise motion detection is based on image differences, only pixels of moving objects with a significant difference in pixel values of the two subtrahends can possibly be detected. This leads to holes in the blobs of the moving objects for the automatic motion detection system as, e.g., the roof or the hood of cars is commonly single-colored and equally illuminated. Moreover, moving objects actually create two blobs in image differences: one at the old position and one where it went to. Concerning the manual labeling, only the pixels of the moving objects in the current frame are marked as moving. This is why the true positive rate can never reach 100% by analyzing the image differences only. However, this characteristics are common to all the motion compensation methods and therefore the results are comparable to each other. Table 5.6 shows the detection results for the Stockholm sequence using a manually labeled reference. Although the true positive rates of the homography mapping and the mesh-based approach are alike, the homography has a 4 times higher false positive rate and therefore a 3 times worse precision. The low absolute value of the true positive rate is due to the large number of true negative pixels containing no local motion and the, at the same time, relatively small amount of false positives pixels. Among themselves, the mesh-based approaches using the different outlier classifier produce similar results; however, the motion parallax predictor classifier falls back a bit in true positive detections, as it cannot detect objects moving along the epipolar line if the speed of the object is not fast enough or does not move in the opposite direction (see Chapters 3.6 and 5.1 for details). However, it reaches lowest fp-rate of all methods.

Figure 5.9 gives the pixel-wise classification results for the frames 29 and 147 of the Stockholm sequence, using a Δk of 9. The manually labeled mask is given in the Figures 5.9(a) and (b). It is noticeable that the blobs in the manually labeled mask are much bigger than that of the automatic results because of the previously mentioned reasons, and explains the true positive rate of only 33.84 in Table 5.6. Between the ideal mesh motion compensation in the Figures 5.9(e) and (f) and the mesh using the cluster filter in the Figures 5.9(g) and (h), only very small differences are visible, which are mostly due to small missing blobs in the latter. The homography mapping displayed in the Figures 5.9(c) and (d) is able to classify true positives nearly as good as the mesh (sometimes even better then the mesh with cluster filter), but totally fails in the other regions containing no local motion.

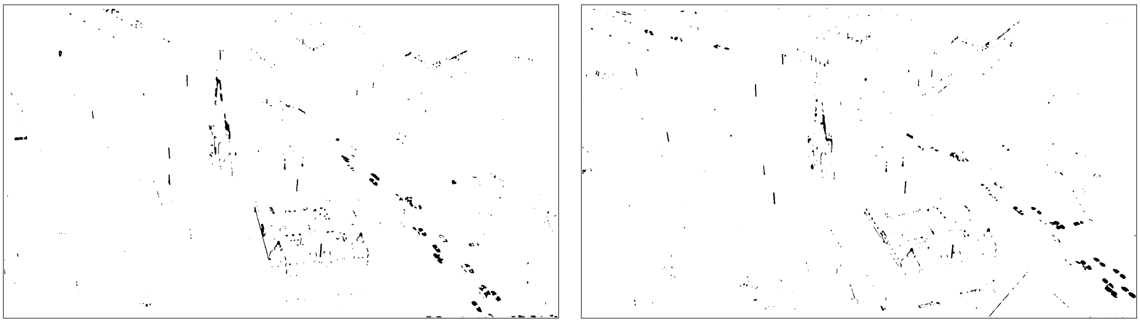
Table 5.6: Pixel-wise classification results in % for the sequence Stockholm

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
Mesh GT	33.84 ± 2.7	99.75 ± 0.0	0.25	66.16	44.88	99.60	99.36
Mesh CF	33.23 ± 2.8	99.75 ± 0.0	0.25	66.77	44.39	99.60	99.35
Mesh MPP	30.04 ± 2.2	99.78 ± 0.0	0.22	69.96	45.23	99.58	99.37
Mesh F-mat	28.38 ± 2.9	99.69 ± 0.1	0.31	71.62	35.97	99.58	99.27
Homography	33.53 ± 2.5	99.00 ± 0.1	1.00	66.47	16.75	99.60	98.61



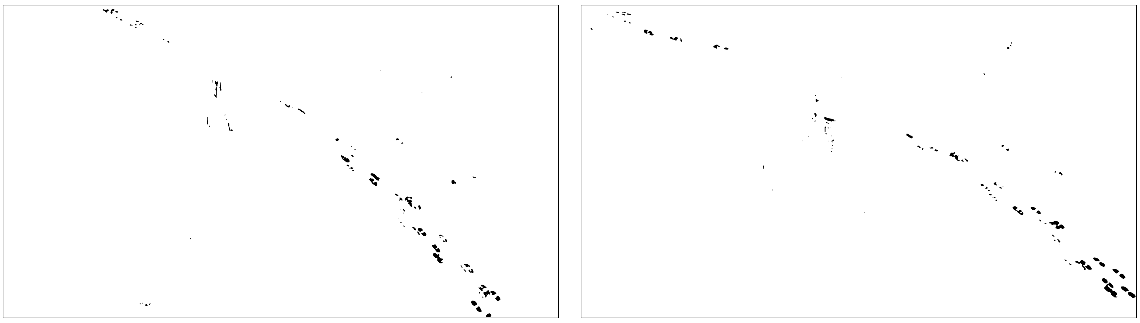
(a) Frame 29,manually labeled

(b) Frame 147, manually labeled



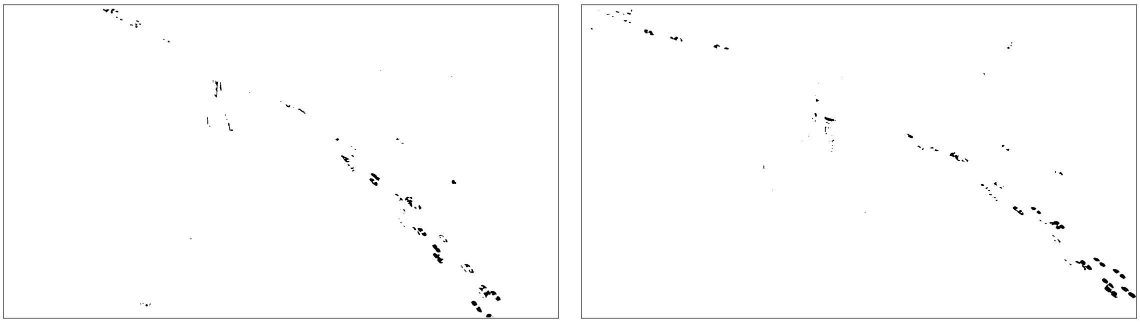
(c) Frame 29, homography

(d) Frame 147, homography



(e) Frame 29, ideal mesh

(f) Frame 147, ideal mesh truth



(g) Frame 29, mesh using cluster filter

(h) Frame 147, mesh using cluster filter

Figure 5.9: Pixel-wise moving object classification for the Stockholm test sequence using $\Delta k = 9$. The broadcasting company logo was masked out.

Table 5.7: Pixel-wise classification results in % for the sequence Chicago, referenced to the ideal mesh result

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
Mesh CF	96.79 ± 7.3	100.00 ± 0.0	0.00	3.21	94.07	100.00	100.00
Mesh F-mat	35.21 ± 17.2	99.97 ± 0.1	0.03	64.79	34.11	99.97	99.93
Homography	87.35 ± 12.7	99.02 ± 0.3	0.98	12.65	4.25	99.99	99.02

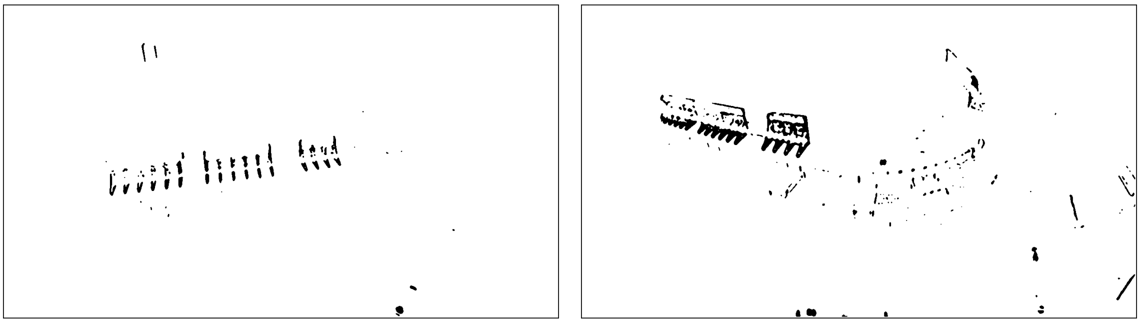
Table 5.8: Pixel-wise classification results in % for the sequence OldTownCross, referenced to the ideal mesh result

	tp-rate	tn-rate	fp-rate	fn-rate	precis.	npv	accur.
Mesh CF	83.28 ± 14.3	99.99 ± 0.0	0.01	16.72	69.03	99.99	99.98
Mesh MPP	51.18 ± 16.9	99.92 ± 0.0	0.08	48.82	20.49	99.98	99.90
Mesh F-mat	58.54 ± 30.6	99.91 ± 0.1	0.09	41.46	20.19	99.98	99.89
Homography	75.54 ± 12.0	99.81 ± 0.1	0.19	24.46	13.08	99.99	99.80

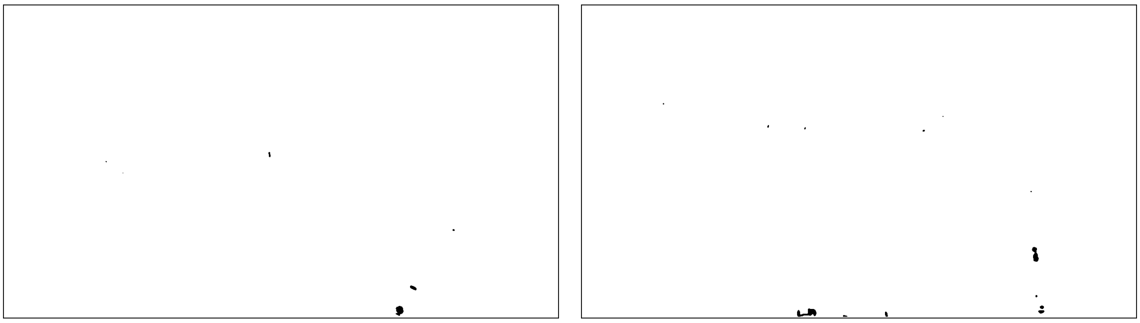
In large areas, contours of buildings and the church are visible as well as additional small errors distributed all over the frame, which make a reliable detection of moving objects nearly impossible.

These errors are even worse for the Chicago sequence in Figure 5.10. Here, the stadium takes a wide area of the frame and big contours of the superstructures of the stadium are visible, together with further smaller buildings. For the mesh on the other hand, the area of the stadium is nearly free of false positives, except for small blobs at single parts of the superstructure. However, some false positives are created by the mesh using the cluster filter at image borders in Figure 5.10(f). In this case, this is due to the broad road in that area, which does not contain enough texture for feature points to be placed there (see Figure 5.11(d)) and the emerging gap is too big for the similarity constraints of the cluster filter to be fulfilled. For the pictured frames, the true positive detections of the mesh are of the same quality as the homography results. As manually labeled masks were not available for the Chicago and the OldTown Sequences, the Tables 5.7 and 5.8 provide the relative pixel-wise classification performances of the investigated methods to the results of the mesh using feature points of the background only as reference.

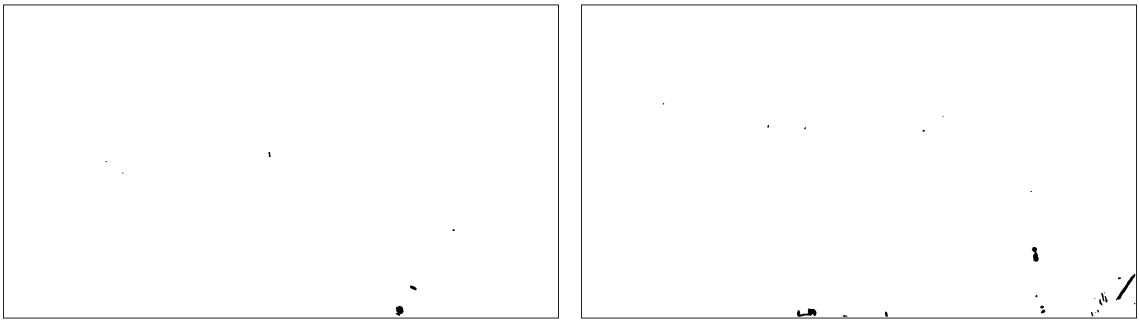
Figure 5.11 provides a comparison of the final result as an overlay of the motion detection map over the original frame for either the homography motion compensations as well as the proposed mesh-based method for all test sequences.



(a) Frame 100, homography (b) Frame 465, homography

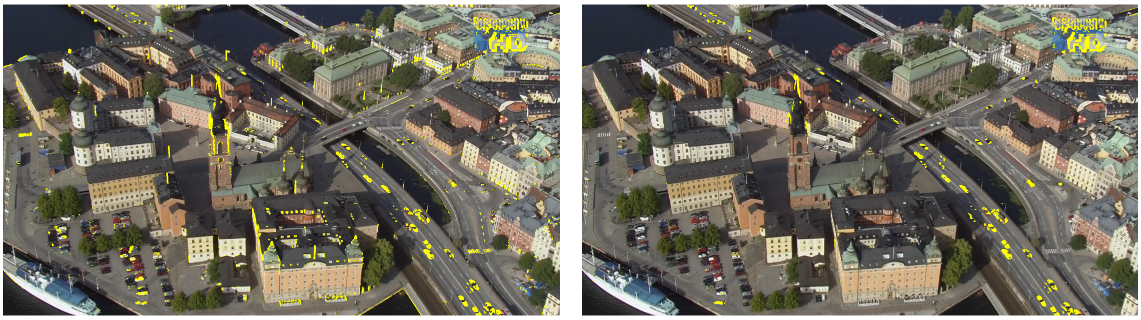


(c) Frame 100, ideal mesh (d) Frame 465, ideal mesh

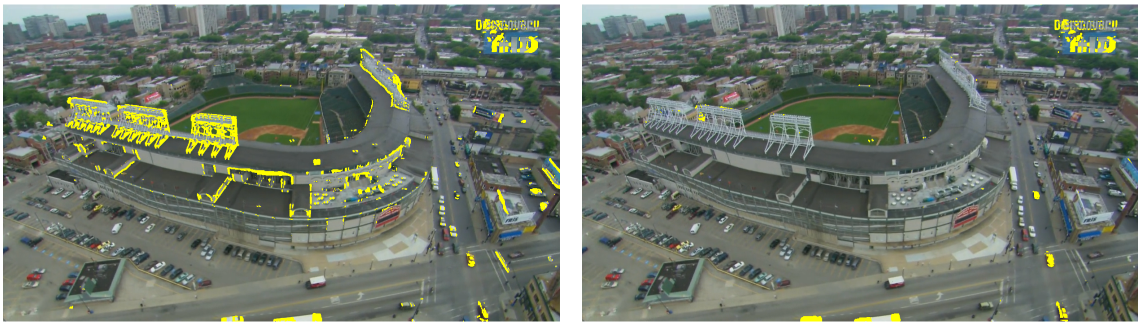


(e) Frame 100, mesh using cluster filter (f) Frame 465, mesh using cluster filter

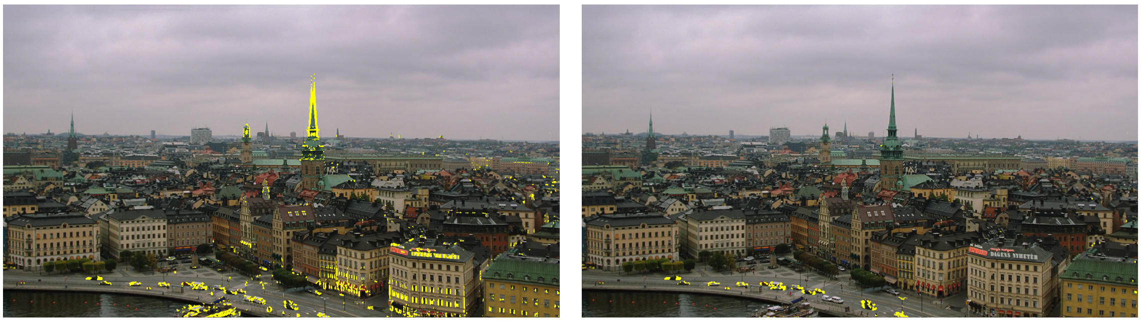
Figure 5.10: Pixel-wise moving object classification for the Chicago test sequence. The broadcasting company logo was masked out. The homography motion compensation leaves a lot of motion compensation residual for non-moving objects and therefore has a high false positive detection rate (a,b). The mesh performs a lot better, with almost only the real moving objects remaining (c,d,e). However, if the mesh node selection in not perfect due to missing feature points in unstructured areas, some artifacts can remain (bottom right in (f)).



(a) Stockholm, homography (b) Stockholm, mesh



(c) Chicago, homography (d) Chicago, mesh



(e) OldTownCross, homography (f) OldTownCross, mesh

Figure 5.11: Final selection of moving objects after erosion noise filter as overlay over the input frame. The pixel-wise moving object detections are highlighted in yellow.

6 Summary and Conclusions

The problems of the planar ground model in a motion detection system for non-planar sequences using the example of aerial surveillance were improved in this work for two different detection methods. The first method is the evaluation of the displacement vector field between the frames of the video sequence, where moving objects are detected as displacements not consistent with a global motion model. The second method is the pixel-based detection by background subtraction, in which the displacement of the pixels between the frames due to the motion of the camera needs to be compensated prior to computing the image differences.

The first improvement was done for the detection of moving objects and outliers directly on the displacement vector field between two frames of the video sequence. For this purpose, first the aberrations of the planar ground model have been analyzed in Chapter 3. To get a better understanding of the model aberrations, the examination was done first for a simplified vertical aerial photo case, with the camera looking perpendicularly downwards. It was revealed that the model aberrations lead to displacements of pixels on the image plane, depending on the height of the object compared to the ground plane. These relief displacements are radial to the nadir point of the camera and are linearly depended on the position and height of the object with respect to the camera and the ground plane.

The influence of the relief displacement on the detection process of moving objects was investigated afterwards by evaluating the change in relief displacement between different positions of the camera. These motion parallax effects lead to displacements of pixels on the image plane which are dependent on the distance of the scene point to the camera. The planar model, realized by a projective transformation using a homography, is only able to describe and compensate the displacement of the ground plane, leaving over the additional displacements from non-planar scene points. These remaining displacements are called motion displacements in this work, and as they appear as virtually moving, they might consequently be falsely detected as moving objects: considering the amount of motion only, a pedestrian walking at approximately 5 km/h cannot be distinguished from the motion displacement of an object with a height of 20 m, if the camera ($f = 80\,\text{mm}$) is moving with more than 100 km/h at a height of 500 m. These conditions already occur for small aerial vehicles, such as medium-sized drones. This is independent of the frame rate of the camera, as the sampling frequency influences the amount of motion parallax the same way as the displacement of ground objects. In contrast to the relief displacement, the motion displacement is, at least for the vertical photo case, not dependent on the ground position of the object but on the height of the object and the motion and speed of the aircraft.

In the following, the investigation of the relief and the motion displacements was extended to cameras with a tilt angle in flight direction, as well as arbitrary oriented and moving cameras. It turned out that the tilt angle has a non-linear effect on the relief and motion displacements. Moreover, the relief displacement is now dependent on the ground position of the non-planar object. Compared to the non-tilted camera, the relief displacement increases mostly with the angle, except for larger distances of the object to the nadir point of the camera, where, for tilt angles less than 30°, it is slightly decreased. The motion displacement decreases with the tilt angle.

A predictor for the motion parallax was created afterwards, based on the equations from the former investigations. It allows the prediction of the minimum and maximum motion parallax displacement of pixels in the image plane, depending on camera speed and an assumed maximum height of objects in the scene. The resulting epipolar line segment can be used to detect moving objects as well as removing outliers by comparing the actually measured displacements to the prediction.

Additionally, a second new clustering based moving object and outlier detector is introduced in this work, named cluster filter. It uses a similarity/smoothness assumption of the optical flow as the background motion model. Non-background motion is detected as discontinuities in the flow and clustered into objects of similar motion. Compared to the motion parallax predictor, the cluster filter does not require calibrated cameras or the positions of the camera during recording.

Concerning the improvement of the pixel-based detection by background subtraction algorithms, Chapter 4 introduces a new locally adaptive global motion compensation algorithm based on triangle meshes to replace the single planar model. Hereby, the motion compensation is performed by sub-dividing the scene into piece-wise planar patches, each of them having its own individual affine transformation. This allows the compensation to locally adapt to non-planar objects, and therefore the mesh has the ability to compensate the background motion more precisely. It also allows the usage of conventional motion detection algorithms based on a static camera scenario which was not possible with the state-of-the-art homography model due to its high aberrations at non-planar structures. Using a Delaunay triangulation, a triangle mesh is created from the point cloud of the background feature coordinates and extended to cover the image edges. The individual mapping for the image pixels contained in each triangle patch is determined by calculating an individual affine projection matrix from the vertexes of each of the triangle patches and their displacements given by the feature correspondences.

To evaluate the improvements over the single-planar model, the description is followed by an accuracy analysis of the mesh model similar to that of the single planar homography model in Chapter 3. The previous object height as distance of a scene point to the ground plane was replaced by the shortest distance of a scene point to the mesh surface. Moreover, the influence of the distance of a mesh node to the scene point was considered for various distances. The mathematical analysis shows that the mesh performs better for nearly all scenarios. This is due to the mesh being able to compensate planar surface patches above the ground surface, e.g. at roofs or walls, nearly error free, as long as at least three mesh nodes are placed on the patch. Remaining errors occur at mesh patches with mesh nodes on more than one plane, e.g. on a roof and on a wall. In this case, similar distortions as with the homography model appear. However, they remain relatively small due to the small patch sizes.

The performances of the proposed outlier detectors and the motion compensation technique are evaluated in Chapter 5 by comparing the detection results to manually created references. For the displacement vector field evaluation, the state-of-the-art homography-based single planar model was used as a reference. Additionally, an epipolar geometry-based detector was compared which uses a fundamental matrix conformance test for the classification. The cluster filter was able to correctly classify on average 99.7% of the background motion vectors correctly, compared to 90.8% for the homography and 94.3% for the fundamental matrix. The motion parallax predictor is on a high level, too, averaging at 98.1%. Moreover, the variance in classification results between the frames, which is a measure of robustness, is dramatically reduced from 21.8% for the homography and 18.0% for the fundamental matrix down to 0.1% for the cluster filter and 0.5% for the motion parallax predictor. The false positive rate is always lower than the state-of-the-art results. It rates from 14% to 22% compared to 10%–40% throughout the tested sequences. The combined accuracy measure averages 99.2% for the cluster filter, 97.5% for the motion parallax predictor, 90.5% for the homography, and 93.7% for the fundamental matrix. It was found that the bad results of the fundamental matrix arose from an unstable estimation from image feature displacements and could be improved using known camera parameters and the elementary matrix instead. However, moving objects moving along the epipolar line are not detectable with the fundamental matrix. This drawback is addressed by the presented motion parallax predictor by defining a small segment on the epipolar line with actual valid displacements of image pixels for non-moving objects. This reduces the amount of false negative detections a lot, as objects moving against the motion parallax displacement direction or which are fast enough are still detectable. The homography results, despite from not being able to describe feature displacements above the ground plane, are additionally affected by the least-squares fitting of the plane in a non-planar scenario. This leads to the plane being placed totally of the real ground plane at random frames and explains the high variance in the homography results.

Concerning the motion compensation of the background motion, the performance of the proposed mesh-based motion compensation was compared to the homography mapping as the state-of-the-art method. A simple image differences-based background subtraction algorithm was used for this, based on averaging, binarization, and erosion, to classify each pixel of the frames into fore- or background. The output was evaluated against manually labeled binary classification maps where available, and the results of an ideal mesh compensation using manually classified displacement vectors of the background only for the other sequences. The ideal mesh-based approach performs best by classifying 33.8% of the pixels containing moving objects correctly. The mesh using the cluster filter determined mesh nodes showed a similar or slightly lower true positive detection rate of 33.2% compared to the homography mapping which reached 33.5%. However, the false positive rate was with 1.0% four times higher for the homography than for both mesh approaches, which reached 0.25%. This was also visible in the resulting binary classification maps, which contained around the same amount of false positive pixels than true positive ones for the homography whereas the mesh approaches only had false positive detections in the area of newly occurring background. For the Chicago sequence, the results of the homography motion compensation was even worse. Due to the small number of moving objects in the scene but the high number of false positive detections due to the huge central stadium, the homography could only reach a precision of 4.3% based on the ideal mesh results, whereas the cluster filter reached 94.1% with a true positive rate of 96.8% correctly classified pixels on moving objects and 100% of correctly classified background pixels.

For the fully automatic detection of moving objects in aerial video, the cluster filter performs best for displacement vector based detection. It does not require a priori knowledge such as camera motion or calibrated cameras and achieves a true positive rate, precision, and accuracy of nearly 100%. Concerning global motion compensation used for pixel-wise detections, the mesh-based method outperforms the homography-based system by a factor of 4, and is able to eliminate nearly all distortions due to motion parallax.

Outlook

The similarity constraints of the cluster filter are chosen fairly generalized at the moment. Although this allows the usage in a wide field of applications, it limits the detection performance for well-defined tasks. For example, in the aerial surveillance scenario investigated in this work, the false positive rate is, although lower than that of the state-of-the-art methods, still quite high. A better motion model of the displacement vectors, e.g. by using a multidimensional predictor of the displacement by extrapolating from more than one object feature might improve the result. The same is true if the classification is not done for every frame individually but by taking into account the knowledge of already processed frames. This would improve the missed detections of moving objects in single frames.

Bibliography

[1] J. Albertz. *Einführung in die Fernerkundung.* Wissenschaftliche Buchgesellschaft, Darmstadt, 4th edition, 2009.

[2] C. H. Anderson, P. J. Burt, and G. S. Van der Wal. *Change Detection and Tracking Using Pyramid Transform Techniques.* SPIE Intelligent Robotics and Computer Vision IV, 579:72–78, 1985.

[3] P. Banerjee and R. Nevatia. *Dynamics Based Trajectory Segmentation for UAV Videos.* In Proceedings of the 7th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pages 345–352, Washington, DC, USA, 2010. IEEE Computer Society.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. *Speeded-Up Robust Features (SURF).* Journal of Computer Vision and Image Understanding, 110(3):346–359, June 2008.

[5] J. Bigün and G. H. Granlund. *Optical Flow Based on the Inertia Matrix of the Frequency Domain.* In Proceedings of the SSAAB Symposium on Picture Processing, pages 132–135, Lund University, Sweden, March 1988.

[6] A. Bowyer. *Computing Dirichlet Tessellations.* The Computer Journal, 24(2): 162–166, 1981.

[7] H. Broszio. *Schätzung von Brennweite und Rotation einer Kamera mit Radialverzerrung und Rotationsachsenversatz aus Bildsequenzen.* PhD thesis, Laboratorium für Informationstechnologie (LFI), Universität Hannover, October 2007.

[8] A. Bruhn, J. Weickert, and C. Schnörr. *Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods.* International Journal of Computer Vision, 61:211–231, 2005.

[9] A. Bürkle, F. Segor, and M. Kollmann. *Towards Autonomous Micro UAV Swarms.* Journal of Intelligent and Robotic Systems, 61:339–353, 2011.

[10] C. Cappelle, M. El Badaoui El Najjar, D. Pomorski, and F. Charpillet. *Localisation in urban environment using GPS and INS aided by monocular vision system and 3D geographical model.* In IEEE Intelligent Vehicles Symposium, pages 811–816, Istanbul, Turquie, 2007.

[11] S.-C. S. Cheung and C. Kamath. *Robust techniques for background subtraction in urban traffic video.* volume 5308, pages 881–892. SPIE, 2004.

[12] W. Chimiak and B. V. Smith. *The Xfig buildings library.* URL: `http://xfig.org`.

[13] Y.-C. Chung and Z. He. *Low-Complexity and Reliable Moving Objects Detection and Tracking for Aerial Video Surveillance with Small UAVS.* In ISCAS, pages 2670–2673. IEEE, 2007.

[14] K. Cordes. *Occlusion Handling in Scene Reconstruction from Video*, volume 10. VDI Verlag, 2014.

[15] N. Dalal and B. Triggs. *Histograms of Oriented Gradients for Human Detection.* In C. Schmid, S. Soatto, and C. Tomasi, editors, International Conference on Computer Vision & Pattern Recognition, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[16] P. E. Debevec, C. J. Taylor, and J. Malik. *Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach.* In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, SIGGRAPH '96, pages 11–20, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4.

[17] F. Devernay and O. Faugeras. *Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Enviroments.* Mach. Vision Appl., 13(1):14–24, Aug. 2001.

[18] R. A. Dwyer. *A faster divide-and-conquer algorithm for constructing delaunay triangulations.* Algorithmica, 2(1-4):137–151, 1987. doi: 10.1007/BF01840356.

[19] O. Faugeras, Q.-T. Luong, and T. Papadopoulou. *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications.* MIT Press, Cambridge, MA, USA, 2001. ISBN 0262062208.

[20] O. D. Faugeras. *What can be seen in three dimensions with an uncalibrated stereo rig.* In Proceedings of the Second European Conference on Computer Vision, ECCV '92, pages 563–578, London, UK, UK, 1992. Springer-Verlag. ISBN 3-540-55426-2.

[21] O. D. Faugeras and F. Lustman. *Motion and Structure From Motion in a Piecewise Planar Environment.* Intern. J. of Pattern Recogn. and Artific. Intelige., (3):485–508, 1988.

[22] M. A. Fischler and R. C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Commun. ACM, 24(6):381–395, June 1981.

[23] D. Gibson and M. Spann. *Robust optical flow estimation based on a sparse motion trajectory set.* IEEE Transactions on Image Processing, 12(4):431–445, 2003. doi: 10.1109/TIP.2003.811628.

[24] Google Inc. *Google Maps.* URL: `http://maps.google.com`.

[25] C. Harris and M. Stephens. *A Combined Corner and Edge Detection.* In Proceedings of The Fourth Alvey Vision Conference, pages 147–151, 1988.

[26] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.

[27] R. I. Hartley. *Estimation of Relative Camera Positions for Uncalibrated Cameras.* In Proceedings of the Second European Conference on Computer Vision, ECCV '92, pages 579–587, London, UK, UK, 1992. Springer-Verlag. ISBN 3-540-55426-2.

[28] R. I. Hartley. *In Defense of the Eight-Point Algorithm.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(6):580–593, June 1997.

[29] F. Heitz and P. Bouthemy. *Multimodal estimation of discontinuous optical flow using Markov random fields.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(12):1217–1232, December 1993.

[30] G. Hildebrandt. *Fernerkundung und Luftbildmessung.: Für Forstwirtschaft, Vegetationskartierung und Landschaftsökologie.* Wichmann Herbert, 1996. ISBN 9783879072385.

[31] G. Holst and T. Lomheim. *CMOS/CCD Sensors and Camera Systems.* SPIE PM. JCD Publishing, 2007. ISBN 9780819467300.

[32] B. K. Horn and B. G. Schunck. *Determining Optical Flow.* Technical report, Cambridge, MA, USA, 1980.

[33] B. K. P. Horn and B. G. Schunck. *Determining Optical Flow.* ARTIFICAL INTELLIGENCE, 17:185–203, 1981.

[34] J. Jain and A. Jain. *Displacement Measurement and Its Application in Interframe Image Coding.* 29(12):1799–1808, 1981.

[35] P. Kaewtrakulpong and R. Bowden. *An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection.* Sept. 2001.

[36] S. Klomp, M. Munderloh, and J. Ostermann. *Decoder-Side Hierarchical Motion Estimation for Dense Vector Fields.* In Proceedings of the Picture Coding Symposium, pages 362–366, Nagoya, Japan, Dec. 2010.

[37] M. Kopsch. *Delaunay_ circumcircles_ centers.svg.* Nü es derivative work [CC-BY-SA-3.0 (`http://creativecommons.org/licenses/by-sa/3.0/`) via Wikimedia Commons].

[38] Laboratorium für Informationstechnologie, Leibniz Universität Hannover. *Voodoo Camera Tracker.* URL: `http://www.viscoda.com/index.php/en/products/non-commercial/voodoo-camera-tracker`.

[39] S. Lazebnik, C. Schmid, and J. Ponce. *Semi-Local Affine Parts for Object Recognition.* In BMVC, pages 959–968, 2004.

[40] H. C. Longuet-Higgins. *A Computer Algorithm for Reconstructing a Scene from Two Projections.* Nature, 293:133–135, Sept. 1981.

[41] D. G. Lowe. *Distinctive image features from scale-invariant keypoints.* International Journal of Computer Vision (IJCV), 60(2):91–110, 2004.

[42] B. D. Lucas. *Generalized Image Matching by the Method of Differences.* PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

[43] B. D. Lucas and T. Kanade. *An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI).* In Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), pages 674–679, April 1981.

[44] M. Maurer, M. Rumpler, A. Wendel, C. Hoppe, A. Irschara, and H. Bischof. *Geo-Referenced 3D Reconstruction: Fusing Public Geographic Data and Aerial Imagery.* In In Proceedings of the International Conference on Robotics and Automation (ICRA), St. Paul, Minnesota (USA), May 2012.

[45] Microsoft Corporation. *bing Maps.* URL: `http://www.bing.com/maps`.

[46] M. Munderloh, H. Meuel, and J. Ostermann. *Mesh-based Global Motion Compensation for Robust Mosaicking and Detection of Moving Objects in Aerial Surveillance.* In IEEE CVPR 2011, 1st Workshop of Aerial Video Processing (WAVP), jun 2011.

[47] M. P. Nguyen. *Bewegungskompensation und Autofokussierung von SAR-Rohdaten mit großem Schielwinkel.* PhD thesis, Institut für Informationsverarbeitung (TNT), Leibniz Universität Hannover, 2014.

[48] S. Nogaki and M. Ohta. *An overlapped block motion compensation for high quality motion picture coding.* In Circuits and Systems, 1992. ISCAS '92., Proceedings of 1992 IEEE International Symposium on, pages 184–187, May 1992.

[49] M. T. Orchard and G. J. Sullivan. *Overlapped Block Motion Compensation: An Estimation-Theoretic Approach.* IEEE Transactions on Image Processing, 3:693–699, 1994.

[50] S. J. Piercea and J. R. Vasquez. *Context Aided Tracking in Aerial Video Surveillance.* SPIE Signal and Data Processing of Small Targets, 6969:696914–1–696914–12, 2008.

[51] P. W. Power and J. A. Schoonees. *Understanding background mixture models for foreground segmentation.* In Image and Vision Computing, page 267, 2002.

[52] P. Rosen, S. Hensley, I. Joughin, F. Li, S. Madsen, E. Rodriguez, and R. Goldstein. *Synthetic Aperture Radar Interferometry.* In Proceedings of the IEEE, volume 88, pages 333–382, March 2000.

[53] T. Schenk. *Introduction to Photogrammetry.* Department of Civil and Environmental Engineering and Geodetic Science, Ohio State University, 2005.

[54] C. Schmid, R. Mohr, and C. Bauckhage. *Comparing and Evaluating Interest Points.* pages 230–235, 1998.

[55] C. Schnörr. *On Functionals with Greyvalue-Controlled Smoothness Terms for Determining Optical Flow.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(10):1074–1079, 1993.

[56] J. Shi and C. Tomasi. *Good features to track.* In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pages 593–600, Seattle, WA, June 1994. IEEE Computer Society.

[57] C. Stauffer and W. E. L. Grimson. *Adaptive Background Mixture Models for Real-Time Tracking.* In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, volume 2, pages 246–252, Los Alamitos, CA, USA, Aug. 1999. IEEE Computer Society.

[58] Sveriges Television AB (SVT), Sweden. *The SVT High Definition Multi Format Test Set.*

[59] O. Tange. *GNU Parallel - The Command-Line Power Tool.* The USENIX Magazine, 36(1):42–47, Feb 2011. URL: `http://www.gnu.org/s/parallel`.

[60] M. Teutsch. *Moving Object Detection and Segmentation for Remote Aerial Video Surveillance.* PhD thesis, Lehrstuhl für Interaktive Echtzeitsysteme (IES), Karlsruhe Institue of Technology, 2014.

[61] The Discovery Channel. *Aerial Video of Riddarholmskyrkan, Stockholm, Sweden.* Discovery Communications LLC, 2007.

[62] The Discovery Channel. *Aerial Video of Wrigley Field stadium, Chicago, Illinois, USA.* Discovery Communications LLC, 2007.

[63] T. Thormälen. *Zuverlässige Schätzung der Kamerabewegung aus einer Bildfolge.* PhD thesis, Laboratorium für Informationstechnologie (LFI), Universität Hannover, February 2006.

[64] C. Tomasi and T. Kanade. *Detection and Tracking of Point Features.* Technical report, International Journal of Computer Vision, 1991.

[65] P. H. S. Torr and D. W. Murray. *Outlier Detection and Motion Segmentation.* 1995.

[66] P. H. S. Torr and D. W. Murray. *Context Aided Ttracking in Aerial Video Surveillance.* In Proc. SPIE Vol. 2059 Sensor Fusion VI, pages 432–443, August 2008.

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

[68] M. Unger, M. Asbach, and P. Hosten. *Enhanced background subtraction using global motion compensation and mosaicing.* In ICIP, pages 2708–2711, 2008.

[69] D. F. Watson. *Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes.* The Computer Journal, 24(2):167–172, Jan. 1981.

[70] J. Weickert and C. Schnörr. *Variational Optic Flow Computation with a Spatio-Temporal Smoothness Constraint.* Journal of Mathematical Imaging and Vision, 14(3):245–255, 2001.