Semantic annotation of urban scenes: Skyline and window detection

MSc Thesis

written by

Tjerk Kostelijk
mailtjerk@gmail.com
(born June 14th, 1983 in Alkmaar, the Netherlands)

under supervision of **Isaac Esteban** and **Prof. dr ir Frans C. A. Groen**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

Master of Science in Artificial Intelligence

at the *Universiteit van Amsterdam*.

Date of the public defense:	**Members of the Thesis Committee:**
Juli 6th, 2012	Prof. dr ir Frans C. A. Groen
	dr. P.H. Rodenburg
	dr. Arnoud Visser

Universiteit van Amsterdam

Contents

1 Preliminaries on Computer Vision

In this chapter we discuss the basic computer vision techniques that are used for the skyline detection, and window detection. Furthermore we discuss 3rd party software, the *FIT3D toolbox* [**?**] which is used in 3D building extraction and facade rectification.

1.1 Coordination systems

In this thesis three different coordinate reference systems are used. The first one is located at the image level and describes the location of the pixel in 2D. It is called the Image Coordinate Frame (ICF), in Figure 1 (x_0, y_0) span this frame. The second is the Camera Coordinate Frame (CCF) and it involves the projection of the image point (x,y) to the retina of the camera in 3D. The origin of the CCF is equal to the center of the camera. If the camera rotates, it rotates around this point. The coordinates are expressed in (XYZ).
The last system is the World Coordinate Frame (WCF) and it is used to describe the positions of the 3D model of the scene and the position of the cameras in the world in (ijk). An overview of the systems can be found in Table 1.

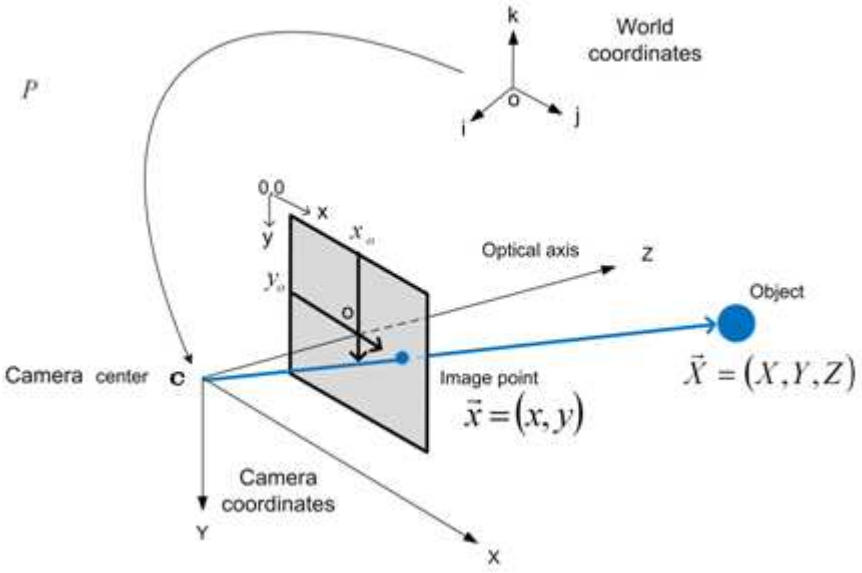

Figure 1: Coordinate systems

1.2 Camera calibration

Camera calibration is done to determine the relationship between what appears on the image (or camera retina) and where it is located in the 3D world. This

Table 1: Coordinate reference systems with their properties

Reference system name	abreviation	dimensions	axis
Image Coordinate Frame	ICF	2D	x,y
Camera Coordinate Frame	CCF	3D	X,Y,Z
World Coordinate Frame	WCF	3D	i,j,k

involves calculating the camera's intrinsic and extrinsic parameters.

Intrinsic parameters

The intrinsic parameters contain information about the internal parameters of the camera. These are focal length, pixel aspect ratio, and principal point.
These parameters come together in a calibration matrix K. The Floriande dataset (originated from the Fit3D toolbox [**?**]) comes with a calibration matrix K. For the other datasets we calculated K with the Bouguet toolbox [**?**]. This method involves taking images of a chessboard in different positions and orientations. An algorithm detects the grid on the chessboard and monitors its transformations under the different images. If enough images are given (at least 10) and the images contain enough variety in chessboard pose, the Bouguet toolbox can calculate the intrinsic parameters quitte accurate. More details about this method can be found in [**?**].

Extrinsic parameters

The extrinsic parameters present the center of the locations of the camera and the camera's headings. These are unique for every view. In some systems these are recorded by measuring the cameras position and headings at the scene. In other systems this is computed afterwards (from the images). We calculated the values also afterwards and used existing software for this: *FIT3D toolbox* [**?**], details of this process is explained next.

1.3 FIT3D toolbox ??

The *FIT3D toolbox* **??** is used for several aims in this thesis. It is used in the window detection module to rectify the facades. and the skyline detection used *FIT3D* to extract a 3D model.
In order to extract this 3D model a series of frames (originating from different views) is used to estimate the relation of the camera coordinates to the world coordinates, Next, the result is used to extract a point cloud of matching features. Finally this point cloud is converted to planes which correspond with the walls of the building.
Because the toolbox plays an assisting role we explain the steps briefly. Detailed knowledge about the methods can be found in **??**.

1.3.1 Multiple views

FIT3D uses multiple views to gather information about the 3D structure of the building. The toolbox comes with a prepared dataset of 7 consecutive images (steady (zoom, lightning, etc.) parameters) of a scene. *FIT3D* doesn't require the input images to be chronological however they need to have sufficient overlap.

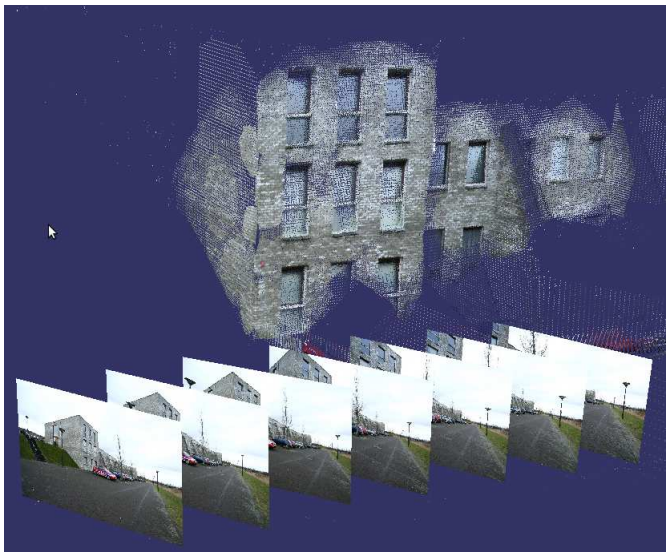

Figure 2: Example series of 7 consecutive frames, (dataset: *FIT3D toolbox*[**?**])

1.3.2 Relating the camera coordinates to the world coordinates

The different views are used to estimate the relation of the camera frame coordinates to the world coordinates, (the extrinsic parameters). In other words the different positions of the camera centers and camera's heading are estimated. This is done by calculating the relative motion between the different views.

The frame to frame motion is calculated by extracting about 25k SIFT features of each frame. Next, SIFT descriptors are used to describe and match the features within the consecutive frames. Not all features will overlap or match in the frames therefor RANSAC is used to robustly remove the outliers. After this an *8-point algorithm* together with a voting mechanism is used to extract the relative camera motion.

The frames are matched one by one which returns an estimation of the camera motion. Because this estimation is not accurate enough, a 3-frame match is done This result is more accurate but comes with a certain amount of re-

projection error which is minimized using a numerical iterative method called *bundle adjustment*.

From every frame the camera motion is stored relative to the first frame. The motion is stored as a rotation and translation in homogeneous form (3x4) $[R|t]$. This is gathered for every frame in a (7x3x4) projection matrix P. P can be used to translate camera coordinates of a specific view to 3D world coordinates and vice versa.

1.3.3 3D point cloud extraction

The next step is to use this projection matrix P to obtain a set of 3D points. These correspond to the matching SIFT features of the different views. This is achieved using a *linear triangulation* method.
The result is an accurate 3D point cloud of the building, see Figure 2. Next a RANSAC based plane fitter is used to accurately fit planes through the 3D points, see Figure 3.

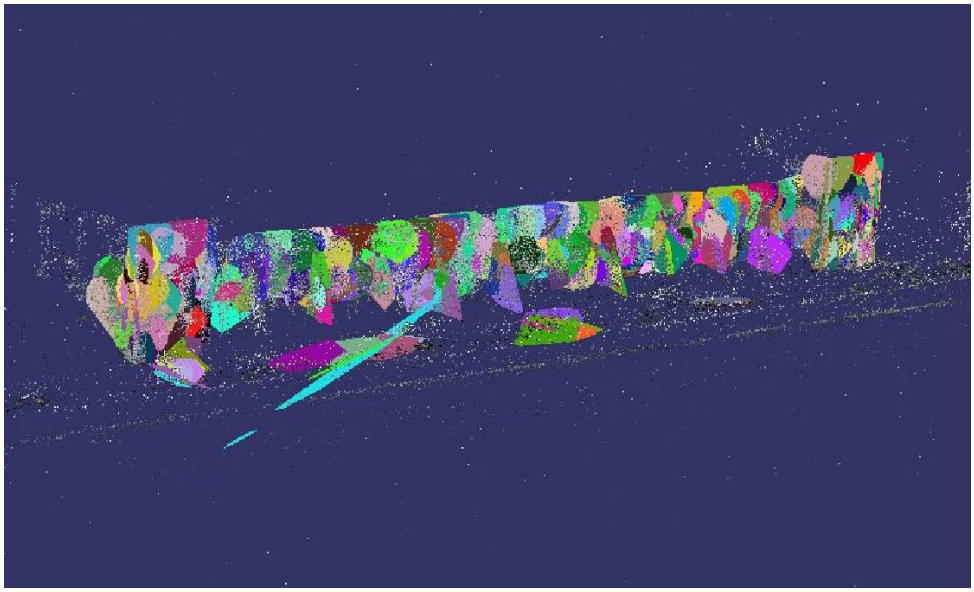

Figure 3: Fitted planes

In this thesis the plane fitting step is skipped and a new method is used to obtain the 3D model. This is explained in 2.

2 Extracting the 3D building

2.1 Introduction

In the previous chapter we explained our skyline detection algorithm which extracted the skyline of a scene. The output is a set of 2D points which was collected for a sequence of images. The aim of this chapter is to use this set of points together with an aerial 2D model of the building and the 3D pointcloud obtained by the FIT3D toolbox[**?**] (1.3) to generate a 3D model of the building.

Research question

Is it possible to use a set of (noisy) skyline points together with an aerial 2D model and a 3D pointcloud obtained by *FIT3D toolbox[**?**]* to generate a 3D model of the building?
We present a stepwise solution.
First *Openstreetmap* is used to obtain a 2D topview of the building (the different parts of the 2D model represent the walls of the building). Next the 3D pointcloud obtained by the *FIT3D toolbox[**?**]* is used to align this 2D model in the scene. After this the set of points returned by the skyline detector is transfered to a set of lines. Then each line segment is assigned to a part (wall) of the aligned 2D model. After this the line segments are projected to vertical planes spaned by the 2D model. The result is used to estimate the height values of the walls of the 2D model. The 2D model is transformed according this height values to a 3D model. We will now elaborate on each step.

2.2 Method

2.2.1 Extracting the 2D model

The basis of the generated 3D model is its ground-plane: a 2D model originated from *Openstreetmap*. *Openstreetmap* (Figure 4) is a freely accessible 2D map generated by users all over the world. It contains information about streets, building contours, building functions, museums, etc. We are interested in the building contours therefor we take a snapshot of a particular area and extract this building contour. This is a set of ordered points where each point corresponds to a corner of the building. Next we link these points to line segments which represent (the topview of) the walls of the building.

2.2.2 Aligning the 2D model

We want to align the 2D model extracted from *Openstreetmap* in the scene at the right location with the right aspect ratios. From *FIT3D[**?**]* we have the 3D point cloud of the building in world coordinates (Figure 5). The first challenge is to obtain a topview of the 3D point cloud. How can we determine the direction

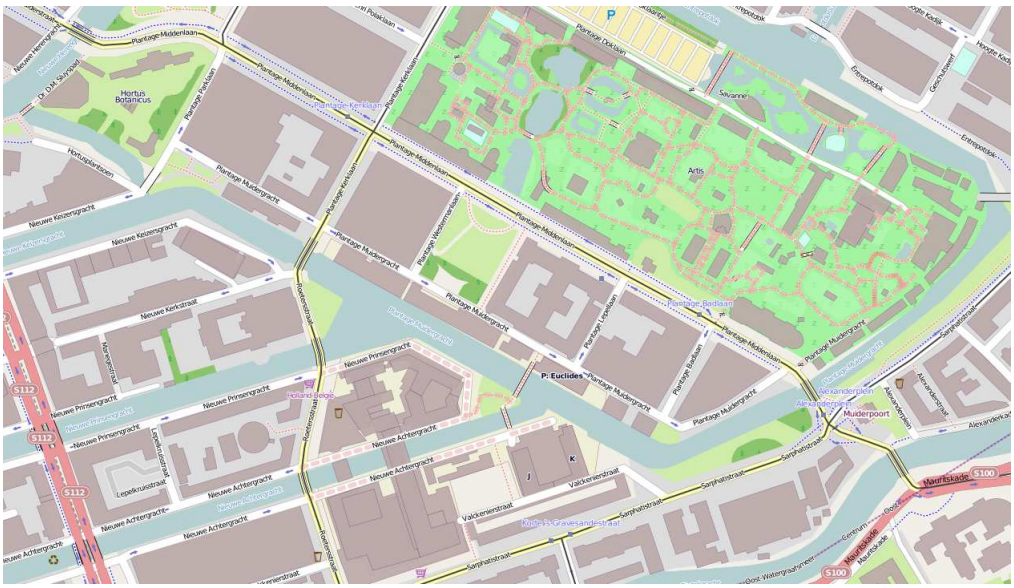

Figure 4: Openstreetmap with annotated buildings

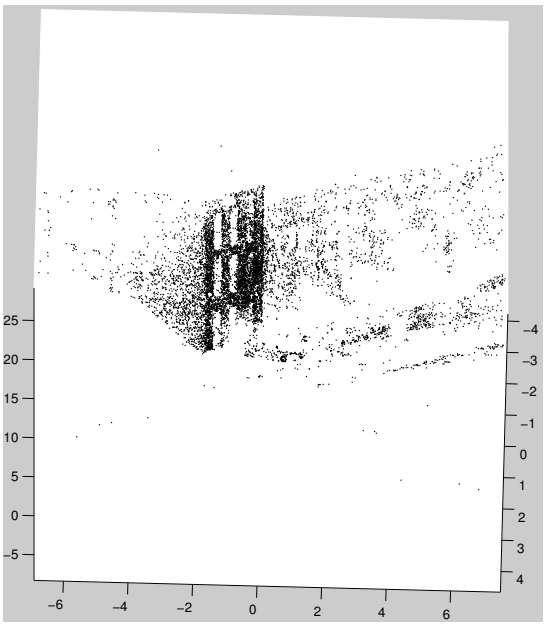

Figure 5: 3D point cloud of the walls of the building

of the top view? We want to project in the direction that is most parallel to

the walls and orthogonal to the ground-plane of the 2D model.

Gravity aligned walls assumption

The walls of the building are aligned in the opposite direction of the gravity which is orthogonal to the 2D basis from ***Openstreetmap***. This means we assume the images taken upright: the camera's *roll* = 0 (Figure **??**).

Now we have defined the wall direction, we obtain the topview of the 3D point cloud by discarding the y-dimension of the points. Note that this is equivalent to a projection to the x,z plane. The result is a set of 2D points that represent the topview of the 3D point cloud (Figure 7).
We determine the walls by fitting line segments in the point cloud. We annotated these line segments manually. (If the system needs to operate automatically, RANSAC can be used to fit lines in this point cloud.)

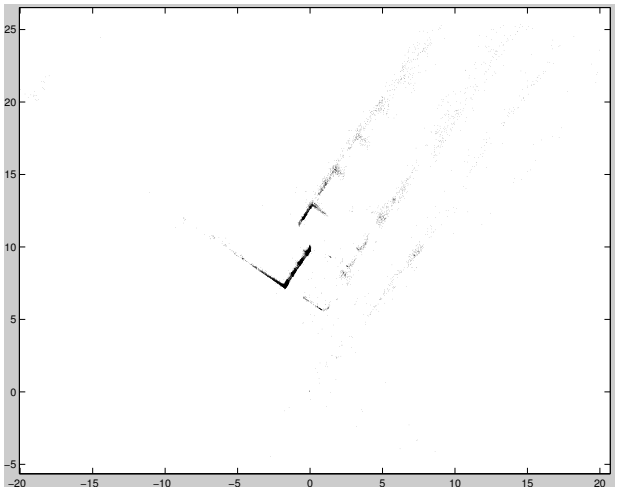

Figure 6: The projected 3D point cloud of the walls of the building

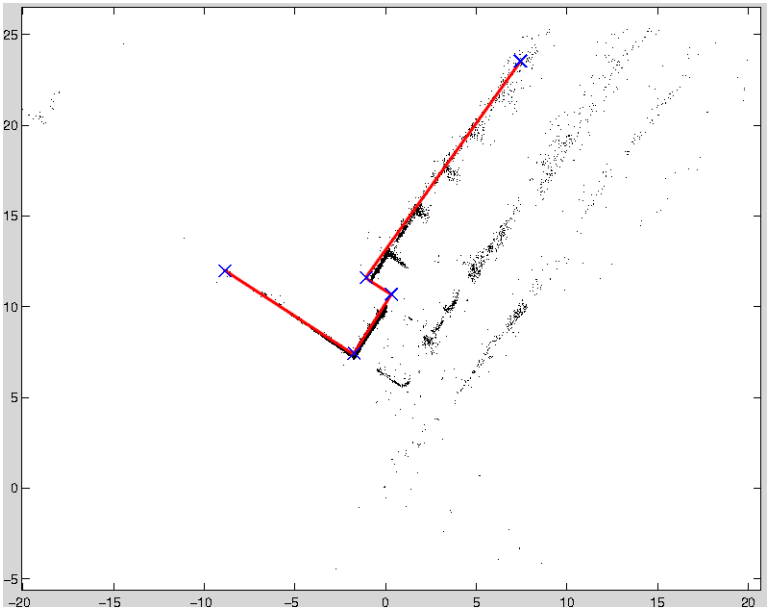

Figure 7: The top view building walls represented the fitted line segments, M1

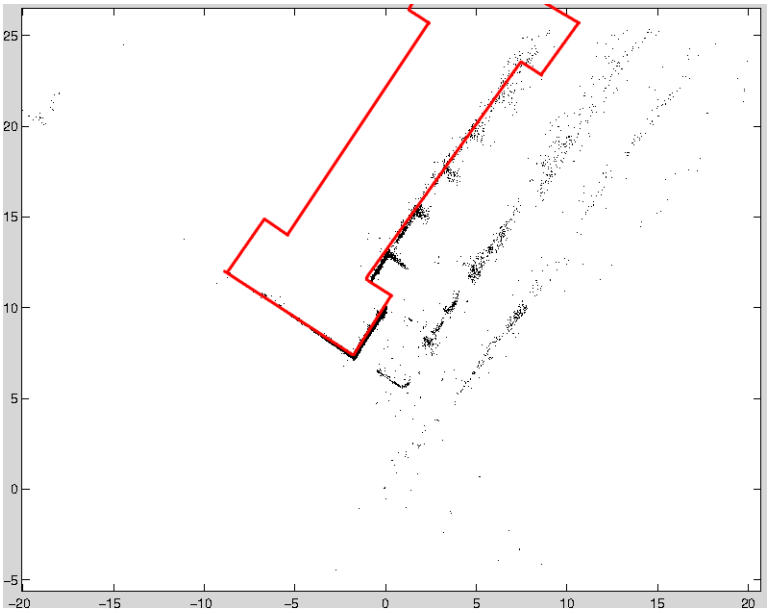

Figure 8: M2, the 2D model extracted from *Openstreetmap* aligned width the point cloud

The next step is to align the 2D *Openstreetmap* model $M1$ with these line segments of the projected pointcloud $M2$. First the endpoints of the line segments, which correspond to the wall corners, are extracted in both models. Not that the walls of $M2$ that are located at the back of the building are often missing because they are occluded by the front walls. We only consider the corners that are present in both models. Next the correspondences between these corners is annotated manually. This is used to generate a set of linear equations which are solved in closed form [**?**]. The result is a matrix A which represents the rotation, translation and scaling that is needed for a coordinate of $M1$ to be transformed to $M2$. Finally A is applied on all cornerpoints of $M1$ which results in an aligned 2D model (Figure 8).

2.2.3 Transfering the aligned 2D model to 3D

Because the 2D model is based on aerial images it contains no information regarding the height of each wall. In the next section we explain how we obtain the precise height values.

For the sake of presentation we use an average building height to generate a rough estimate of the 3D model. The 3D model is generated by taking 2D model and extend it in the opposite gravity direction. An example of the 3D model can be seen in Figure 9.

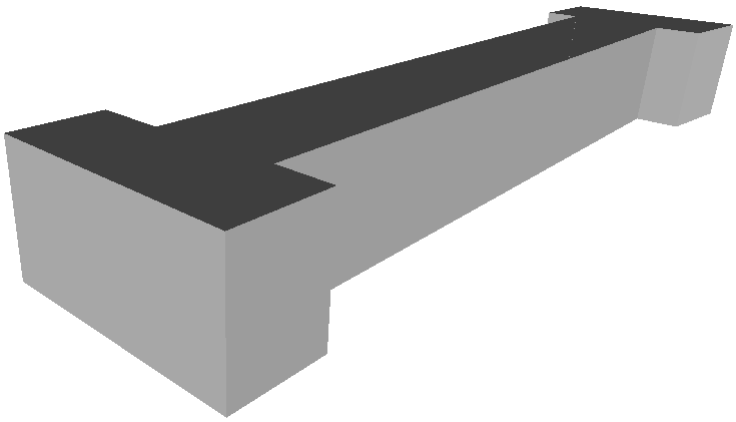

Figure 9: An example of a basic 3D model, generated by extending a basis (2D) model from *Openstreetmap* to an average building height

2.2.4 Extracting line segments

Because we want to estimate the height of the building walls of the 3D model we need to know how high the walls in the 2D images are. To estimate this we first

determine which part of the skyline is part of the building contour. Straight lines in the skyline area are likely to come from the building contour. If we have a method that extracts straight line segments from the skyline, we can use these lines segments to estimate the height of the walls of the 3D model. In this section we explain how we extracting this straight line segments.

Assumptions

Many urban areas contain buildings with a flat roof. This makes the contour of the building is always formed by a set of straight line segments. Furthermore the contour of the building is always aligned with the upper side of a building wall. If we assume a flat roof we can find the height of the building walls without having to concern for (complex) roof types.

Flat roof assumption

We assume each building has a flat roof, implicating that the building contour is aligned with the topside of a building wall. The building walls may have different heights but the roof should be flat.

Hough transform

cutted

2.2.5 Project the skyline to the 3D model

??
The Hough transform of the previous section returned a set of 2D line segments which likely present parts of the building contour. As we want to estimate the building wall heights we need to determine the wall of the building that is most likely responsible for that line segment. We can solve this problem by projecting the line segments to a specific part of the 3D model. To This is done in three steps, first the camera is calibrated, next we project the line segments to the the building and finally we determine the specific building part that is associate with a line segment.
To project the line segments to the 3D model we need to know where the camera was positioned and heading when it took the photo (extrinsic parameters). Furthermore we need to know in what way this camera transformed the image (zoom factors, lens distortion etc.) (intrinsic parameters). This process is referred to as Camera Calibration and is explained in (1.2)

From image point (2D) to possible points in scene (3D)

What can we do if we computed the camera calibration parameters? The line segment that was returned by the Hough transform consists of two endpoints v

and w. These endpoints are in 2D but are recorded in a 3D scene and therefore present a 3D point in space. We don't know which point this is as for example we don't know the distance from the 3D point to the camera that took the picture. However, because we calibrated the camera (1.2) we can reduce these possible points in 3D space to a line. Next we explain how we calculate this for one 2D image point (for example a line segment endpoint).

We know:

- $\vec{x} = (x, y)$, the image point (in the camera coordinates (xyz))
- $\vec{x}_h = (x, y, 1)'$, the homogeneous coordinate of the image point.
- $\vec{c}$ and $\vec{h}$, the camera's extrinsic parameters (the camera's center and heading in world coordinates (ijk))
- K, the cameras intrinsic parameters
- P, the projection from camera coordinates (xyz) to world coordinates (ijk). It contains implicitly the intrinsic and extrinsic parameters (the camera's center $\vec{c}$ and heading $\vec{h}$) and is applied to transfer the camera image points $\vec{x}$ (xyz) to world coordinates (ijk).

The two coordinates that span the line of possible points in 3D space are calculated as follows:

- $\vec{c}$, the location of the center of the camera in world coordinates (ijk)
- $\vec{x}_{ijk} = PK'\vec{x}_h$, the image point that lies on the retina of the camera expressed in world coordinates (ijk).

This is illustrated in Figure 10.

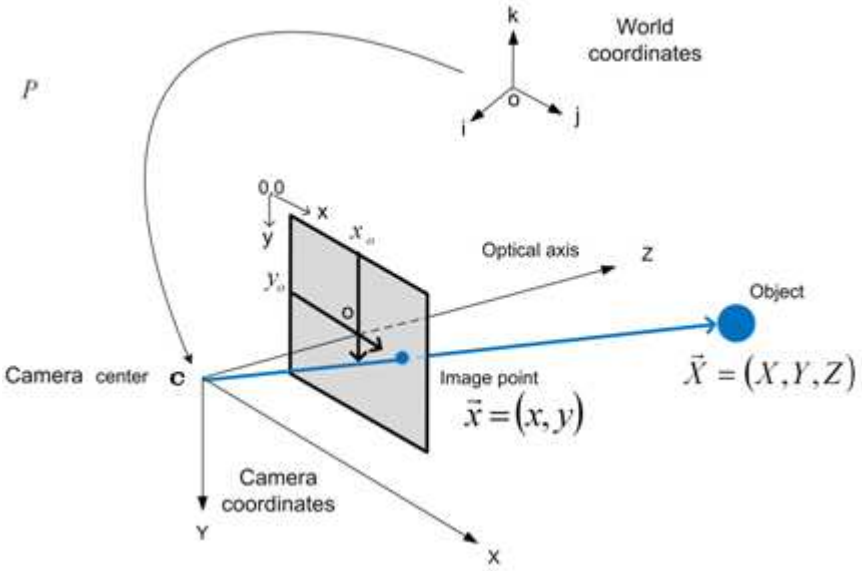

Figure 10: The blue line spanned by the camera center $\vec{c}$ and the image point $\vec{x}$ tranfered to world coordinates represent the possible 3D points in space. $\vec{X}$ is a random possible point on the line.

Using the two required coordinates we set up an equation of the line of possible points in 3D.

$$l = \mathbf{c} + (\mathbf{x}_{ijk} - \mathbf{c})t, t \in \mathbb{R}$$

for example if $t = 100$ then the point in the real world lies at 100 times the distance from the camera center to its retina. In Figure **??** t is about 4 (for point M).
Above calculations are done for each line segment endpoint line is spanned Now we know which possible points in 3D a 2D point of the skyline presents, we can use this to find the corresponding wall.

2.2.6 Associating line segments with building walls

Building wall appearance assumption:

We assume that every straight line segment of the skyline represent (a part of) the upper side of a specific wall of the building. Unfortunately we don't know which line is associated with which building wall. In this section we determine this association.
First we prepare the 3D model. Next we project the line segments to all planes of the 3D building by taking their endpoints and using the technique of the previous section (**??**) Finally we determine the most likely plane based on the largest line-wall overlap.

Preparing the 3D model

The 3D building model consists of different walls. A wall is described by two ground coordinates, a height, and a direction. The height is based on an average building height, the direction is always the y-direction (see *Gravity aligned walls assumption*).
First we transform the walls into (infinite) planes. This is done for two reasons: first this transformation is required to calculate the intersection properly. Second, because the 3D model is an estimate, the walls maybe just to small which could result in a missing intersection.

Intersect with all walls

Now we have the building walls transformed to planes we take the endpoints of the lines and project them to all the planes of the building.
Each 2D endpoint has a line of possible 3D points which we calculated in the previous section. This was the line spanned by the camera center and the image point in world coordinates. This line is intersected with all planes of the building walls. Every 2D endpoint is now associated with multiple intersections resulting in 2 x l x w points in 3D (grouped by the line segments), 2 means #endpoints of the line, l is the number of lines and w is the number of walls.

We now calculated every possible projection to (/intersection with) every plane.
cut