# Improve 3D models from 2D images

T. Kostelijk
mailtjerk@gmail.com

September 12, 2011

## 1 Comments on this thesis

Dear Frans,
I write this in English so Isaac can read it also. This in an update of my thesis chapter. Isaac read it and give me feedback. This feedback and all feedback you gave me are incorporated except for
-the test on different datasets
(the dataset is not ready yet)
-references
(I'll do this on the end)
-large introduction
Explanation:I expanded the introduction a bit more then my previous version but I think you want more background information. I do this in a few chapters earlier which is not included in this document. Therefore some backreferences to sections also contain a questionmark.

Furthermore, I processed many small changes but a few big changes are:
-I explicitly defined all assumptions.
-I updated the way line segments are associated with building parts and illustrated it with Figure 10.
-I wrote a new subsection about alternative roof-types.

Would you give me feedback about the level of detail, where do I go in to much and where is more detail desired?
Isaac told me to make the future work section smaller, do you agree with this and yes would you tell me which part you think we can drop?
I tried to put some mathematics but didn't do it everywhere. Could you give me feedback where to put (more) formulas or symbols and where not?
I think I need to put my work in context with the work of others. But on this specific part I can't find any work. How do I deal with that?
Do I remember it right that you are leaving for 1 month beginning at half september?
Would be nice if we can discuss your feedback before but that will depend on

your schedule.
Please let me know when the feedback is done and lets make an appointment.
I think Isaacs presence will not be necesarry.

Thanks in advance for reading my thesis chapter and giving me useful feedback.
Kind regards, Tjerk Kostelijk

# 2 Improving the 3D building

## 2.1 Introduction

In the previous chapter we extracted the building contour with the skyline detector. The output was a set of 2D points and we collected this set for every view of the building. The aim of this chapter is to use this set of points to improve a basic 3D model.
The point cloud from the skyline detector included a lot of noise caused mostly by occluding objects like trees. How do we detect those outliers? And if we have an outlier free point cloud how can we use this information to improve a basic 3D model? And how can a point be associated to a specific part of the building? These questions are addressed in this chapter.
We present a stepwise solution. First *Openstreetmap* is used to generate a basic 3D model of a building. Secondly the set of points returned by the skyline detector is transfered to a set of lines. Then each line segment is assigned to a wall of the building. After this the lines are projected to these assigned walls in the 3D model.
The projections are used to estimate the new heights of the building walls. The 3D model is then improved by updating the walls according these heights.
We will now elaborate on each step.

## 2.2 Generating the 3D model

The 3D model is generated using a basis (groundplane) which is manually extended.
The basis (viewed from top) of the generated 3D model is originated from *Openstreetmap.*
Openstreetmap, see Figure 1,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. We take a snapshot of one 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 walls.
Because the map is based on aerial images, it is in 2D and contains no information about the height of each wall. Also information regarding the rooftype or the number of floors is not present.

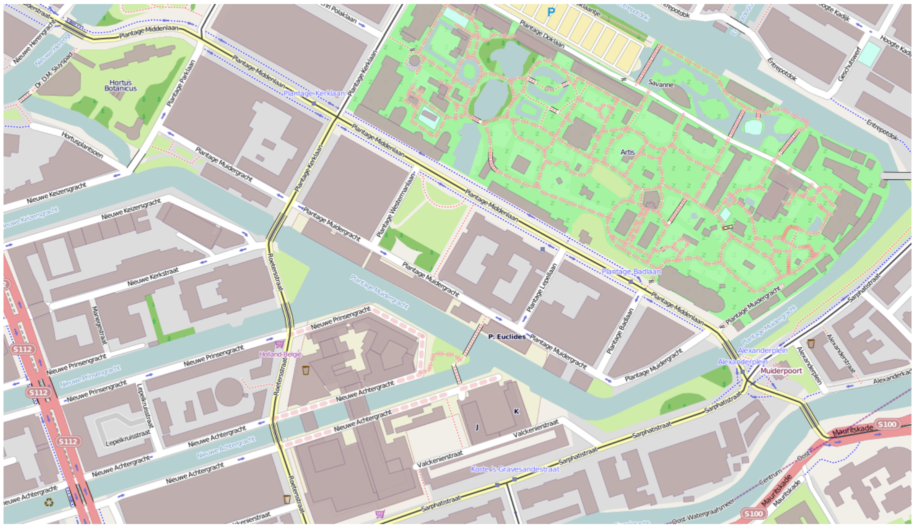


Figure 1:

The final 3D model is generated by starting with the the 2D building contour as its basis. We estimate the wall heights by hand and extend the 2D basis in the gravity direction.

*Gravity aligned walls assumption*

*The walls of the building are aligned in the direction of the gravity which is orthogonal to the 2D basis from* ***Openstreetmap***

An example of the 3D model can be seen in Figure 4.

## 2.3 Extracting line segments

### 2.3.1 Introduction

The skyline detector returned a set of points. If some of these points lie on the same line, they form a straight line. Straight lines are likely to come from the building contour.

If we have a method that extracts these straight line segments, we can use these line segments to find parts of the building contour and finally use this to improve the 3D model.

Unfortunately A problem arises that some points are outliers. To discard these outliers we detect the inliers and consider the remainder as outliers. In this section we draw a assumption, explain how straight line segments are extracted and how outliers are discarded.

### 2.3.2 Assuming a flat roof

Many urban areas contain building with a flat roof. This means that the contour of the building is mostly formed by straight lines. We use this fact to simplify our problem:
*Flat roof assumption*
*We assume each building has a flat roof, implicating that each building wall has a straight upper contour. The walls may have different heights but the roof should be flat.*

This assumption is very useful as it let us focus on finding the height of the buildingwalls from the building contour without having to concern for (complex) rooftypes. This doesn't mean that the method described in this thesis is unusable for building that contain roofs. E.g. with a small adjustment the method could be used to at least gain the building height which is a useful application.
Ideas about how to handle rooftypes explicitly can be found in the Future work section.

### 2.3.3 Hough transform

A widely used method for extracting line segments is the Hough transform (invented by P. Hough). We regard this as a suitable method because it is used a lot for this kind of problems. This is probably because it is unique in its low complexity (compared to other (iterative) methods like *RANSAC*).
In the Hough transform, the main idea is to consider the characteristics of a straight line not as its image points $(x1, y1)$, $(x2, y2)$, etc., but in terms of the parameters of the straight line formula $y = mx + b$. i.e., the slope parameter $m$ and the intercept parameter $b$.
The input of a Hough transform is a binary image, in our case the output of the skyline detector (chapter ?).
If a pixel is classified as a skyline pixel (a pixel that lies on the skyline according the skyline detector), the Hough transform increases a vote value for every valid line ($m$,$b$ pair) that crosses this particular pixel. Lines ($m$, $b$ pairs) that receive a large amount of votes contain a large amount of skyline pixels.
Because the algorithm detects straight lines containing only skyline pixels it is most likely that it returns parts of the skyline and therefore the building contour.
The Hough transform is implemented in *Matlab* and has some useful extra functions.
The algorithm can optionally return the start- and endpoint of the found lines which is very useful as it helps to associate which part of the building is described by the line.
Furthermore it has the parameter *FillGap* that specifies the distance between two line segments associated with the same $m$,$b$ pair. When this is less then the

*FillGap* parameter, it merges the line segments into a single line segment. In our application this parameter is of particular interest when we want to merge lines that are interrupted by for example an occluding tree.
Results of the Hough transform on the 2D output of the skyline detector are displayed and evaluated in the Result section.

## 2.4 Associating line segments with building walls

### 2.4.1 Introduction

The Hough transform of the previous section returned a set of 2D line segments. If we can find a way to associate certain walls of the building with the 2D line segments, we can estimate new heights of this walls and improve our basic 3D model. This section explains how we associate the line segments with the most likely walls of the building.
First, a few assumption are made then the details of the problem are analyzed and finally the developed method is explained.

### 2.4.2 Assumptions

We consider each building consisting of seperate walls and associate each line segment with a wall of the building that is most likely responsible for that line segment.

*Unique wall assumption*:
*We assume that the output of the Hough transform are line segments that each represent a single wall of the building, e.g. if the Hough transform finds 3 line segments there are 3 walls present.*

*Building wall appearence assumption*:
*We assume that every line segment in the output of the Hough transform represent (a part of) the upper side of a specific wall of the building.*
If a line segment is assumed to represent a single wall then the projection to that wall should have a large overlap with this wall. To be more precise, let's define $l$ in $\mathbb{R}2$ as a line segment that is generated by the Hough transform. If we project $l$ to the plane spanned by a wall $W$ we get a line $l_{projW}$ in $\mathbb{R}3$. If we assume $l$ to come from contour of wall $W$, then $l_{projW}$ should have a large intersection with this wall $W$ (as we overestimated the height of the building). We call this the line-wall overlap value, *lwo*. A mathematical definition of *lwo* follows. Note that the projection of $l$ with the other walls should have a small *lwo* value .
*Largest line-wall overlap assumption*:
*A line segment describes the contour of the wall where its projection has the the largest overlap with.*

### 2.4.3 Line-wall association algorithm

Having defined the assumptions, the situation and the idea behind the line-wall association, we can now explain the line-wall matching algorithm.
A line segment is projected to all walls and the amount of line-wall overlap, *lwo* is calculated. The wall with the largest overlap with the specific line segment is classified as the most likely wall for that line segment. Next the line segments are projected to their most likely wall and the algorithm outputs this set of lines in $\mathbb{R}3$.
This line-wall overlap is calculated in different steps. First different types of overlap are explained. After the algorithm determines the *overlap type*, the overlap amount is determined and normalized.
$l_{proj_W}$ can overlap $W$ in four different scenarios, this is explained in Figure 10. The wall $W$ is spanned by $abcd$, and $l_{proj_W}$ is spanned by $vw$.

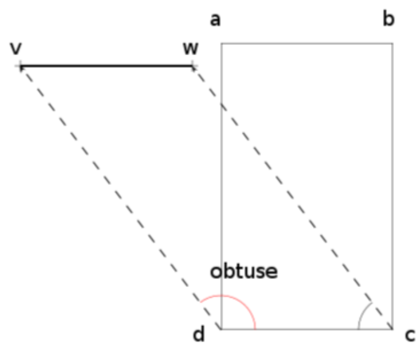


Figure 10a

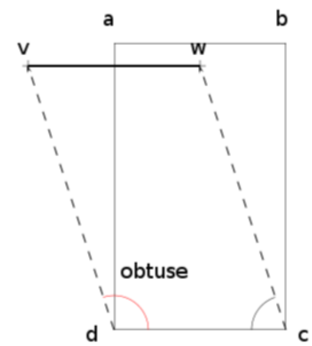


Figure 10b

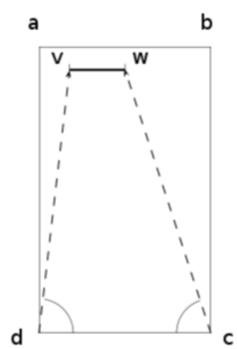


Figure 10c

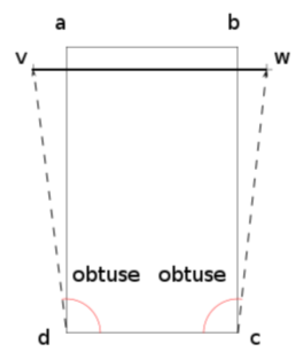


Figure 10d

The type of overlap is defined by exposing the endpoints of the line segments to an *in polygon* test, where the polygon represents a wall of the building (e.g.

*abcd* in Figure 10).
The table below represents the types of overlap with the corresponding number of points that pass the *in polygon* test and their possible line-wall overlap value.

| Type of line-wall overlap | Points in polygon | Line-wall overlap | Figure |
|---|---|---|---|
| No overlap | 0 | 0 | 10a |
| Partial overlap | 1 | [0..1] | 10b |
| Full overlap (included) | 2 | 1 | 10c |
| Full overlap (overextended) | 0 | 1 | 10d |

**No overlap** If the point in polygon test returns 0, the line-wall overlap calculation is skipped and 0 is returned. The remaining overlap types, partial and full, are treated individually:

**Partial overlap** Let's first consider the partial overlap type (Figure 10b), the *in polygon* test returned 1, that means that one of the line segments endpoint lies inside and one lies outside the wall.
To calculate the amount of line-wall overlap, the line segment is cropped to the part that overlaps the wall and the length is measured.
The cropped line has two coordinates, first of course the point that passed the *in polygon* test and secondly the intersection of the line segment with one of the vertical wall sides.
(Note that because we assume the walls to be of infinite height, the partial overlapping line segment always intersects one of the vertical wall sides.)
To determine which of the two vertical wall sides (*da* or *cb* from Figure 10b) is crossed, we determine on which side the point that didn't lie in the polygon (v) lies. This is done by an angle comparison (as in section ?).
First, two groups of two vectors are defined: *dv*, *dc* and *cw*, *cd* (see Figure 10b). We measure the angles between the vectors and call them $\angle d$, and $\angle c$. Because one of the line segment endpoints lies outside the wall $\angle d$ or $\angle c$ is obtuse, in this case $\angle d$ is obtuse. (Note that this holds because the walls are orthogonal to the basis which we assumed in the *Gravity aligned walls assumption*
To be more precise:

- if $\angle d$ is obtuse, the left vertical wall side *da*, is crossed.

- If $\angle c$ is obtuse, the right vertical wall side *cb*, is crossed.

The angles are acute or obtuse if the dot product of the vectors involved are respectively positive or negative. The advantage of this method is that it's simple and has low computational costs.

*Line-wall overlap calculation*
The amount of line-wall overlap is calculated by cutting of the point where $l$

intersects the determined vertical wall side ($da$ or $cb$) and measuring its remaining length.

**Full or no overlap** Now let's consider the overlap types where the *in polygon* test returned 0. As you can see in Figure 10a and 10d this resulted in either full or no overlap. Again we analyze the vector angles to determine the remaining overlap-type. If only one of the angles is obtuse with no points in the polygon, like in Figure 10a, the whole line segment lies outside the wall. An overlap value of zero is returned.
Otherwise, if both angles $\angle d$ and $\angle c$ are obtuse or acute (Figure 10d), both endpoints lie on a different side of the wall, and they cross the wall somewhere in between. Full overlap is concluded here.
The amount of overlap is now calculated by measuring the length of the line segment which is cut down by his intersections with $da$ and $cb$. In this case this is the same as line $dc$, but its easy to see that this is not the case when $vw$ is not parallel to $dc$.

**Line-wall overlap normalization** Finally the line-wall overlap is normalized by the line segments length:

$$\alpha_l = \frac{lwo}{|l|} \tag{1}$$

Where $\alpha_l$ is the normalized line-wall overlap, $lwo$ is the unnormalized amount of line-wall overlap, and ($|l|$) is the total length of the line segment.
The intuition behind this is that line segments that are likely to present a wall not only have a large overlap but also have a small part that has no overlap. By calculating the relative overlap, both amounts of overlap and -missing overlap are taken into account.
The maximum of the normalized line-wall overlap is used to associate a line segment with its most likely wall. To summarize, the overlap type is defined by calculating the numbers of in polygon points and evaluating two dotproducts. Next the line segment is cut off depending on the overlap type and the line is normalized. The maximum normalized line-wall overlap is used to determine the correct line-wall association.

### 2.4.4 Improving the 3D model by wall height estimation

In the previous section we associated the line segments with their most likely wall. In this section this information is used to estimate the heights of the walls which will eventually be used to update the 3D model in the next section.
Now all line segments are associated with a certain wall, we re-project the line segment from the different views on their associated wall. The re-projection

is done by intersecting both endpoints of the line segment to the plane that is spanned by the associated wall.
Next the 3D intersection points are collected and averaged, this gives us an average of the midpoints of the projected line segments. We do this for every wall separately, returning the average height of the line segments. These averages are then used as the new heights of the walls of the building. Note that this is only permitted in presence of the *flat roof assumption*.
The new individual heights are used in the 3D model by adjusting the location of the existing upper corner points of the walls. We copy the bottom left and right corner points and add the estimated height from the previous section to its y-value. The y-value is the direction of the gravity which is permitted by the *Gravity aligned walls assumption*.

## 2.5 Results

Let's return to the output of the skyline detector in Figure 1. Figure

Figure 2:

3 shows the top 3 longest Houghlines, the endpoints are denoted with a black and blue cross. All three line segments lie on the building contour. The left line segment covers only a part of the building wall. The middle line segment covers the full wall. The left and middle line segment are connected. The right

Figure 3:

line segment covers the wall until the tree occludes.

Figure 3 displays the line segments (originated from different views) projected on to their associated walls. For a clear view we've only selected the lines that were associated with three specific walls of the building. The red cross in the middle of the line represents the average of its endpoints.

Figure 5 displays the updated 3D model. The corner points of the walls are adjusted according the calculated wall heights. The green plane displays the augmented wall. The left and middle wall are extended and the right wall is shortened.

### 2.6 Discussion

As can be seen in Figure **??** the left line segment doesn't cover the whole building wall. This is caused by the use of strict parameters in the Hough transform (like a small line thickness parameter). If some ascending skyline pixels fall just outside the Houghlines, a gap is created and the line segment is cut down at that point. This is however not a big problem because the lines are long enough

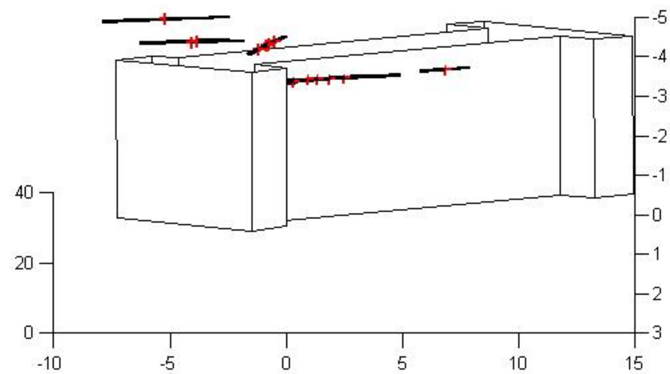


Figure 4:

to produce a good wall height estimate. Furthermore there are at least 5 other lines (originated from different views) that support the estimate for this wall. *I think it is best to add more discussion when the other datasets and results are ready*

## 2.7 Conclusion

To conclude, we showed that a Houghline transform is a useful method to detect outliers and find prominent structure in the contour of a building with a flat roof. We introduced a method to pair up line segments with their associated walls. This was used to produce new wall heights which were propagated to the 3D model. Existing and novel AI computer vision techniques were powerfully combined resulting in an accurate 3D model based on only a few calibrated 2D images.

## 2.8 Future work

As can be seen in Figure 4 two line segments appear on the same single wall. This means that they have a double influence on the average wall height, which is unjustified. A simple solution would be to add a normalization pre-process step, so each view has only one wall height vote per wall. A more decent solution would be to merge the two (or more) line segments to a single line segment. This could be achieved with an iterative Hough transform where the *FillGap* parameter is increased in each iteration. E.g. for the right wall of the building in Figure 4 two iterations would be enough, the *FillGap* parameter needs to be

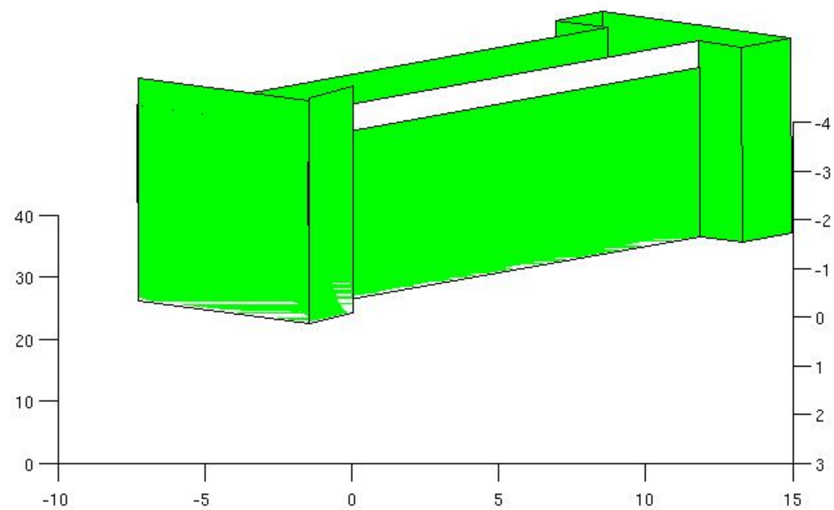


Figure 5:

at least as big as the occluding tree in the second iteration.

In this thesis little is discussed about the computational costs. This is because the computations are done efficiently (e.g. using matrix multiplications in Matlab) and off line, making the calculation process accomplishable in reasonable time. To make the application real time the next speedup would be useful.
To determine the best line-wall association the line segments are now projected to every wall and for every wall the amount of line-wall overlap is calculated. This is computational very expensive and looks a bit like an overkill.
It would be a significant speedup to reduce the set of walls to only the walls that contain the middle point of the line segments. To be more concrete the middle point needs te be calculated by averaging the line segments endpoints, this middlepoint is used in the *in polygon* test for every wall. Next the line-wall association algorithm only treats the walls that pass this test.
The downside of this method is that it will be inaccurate, resulting in more false negatives: a linesegment that overlaps the wall with only 1/3 could be of use in the height estimation but instead it is discarded. What can be concluded is that there is a trade of in the accurateness of the height estimation and the computational costs.

### 2.8.1 Alternative roofs

To make the algorithm more generic, the flat roof assumption could be stretched or even discarded. We'll now consider other roof types and discuss what adaptations the system should require to handle these. In Figure 5, 6 different roof shapes are displayed.

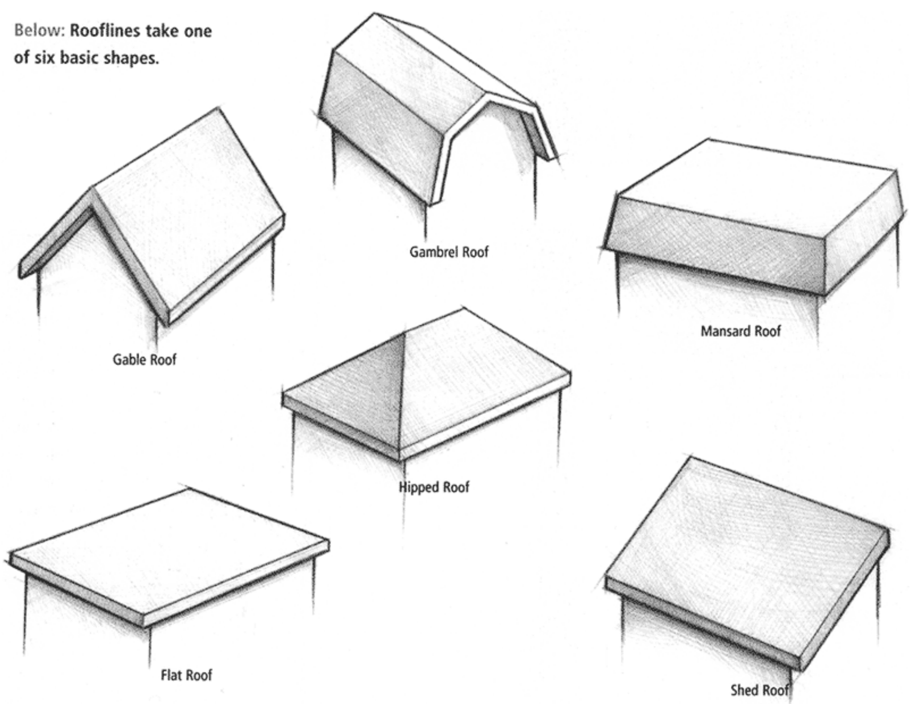


Figure 6:

Consider the Gable Roof in Figure 5, it is a roof consisting of two planes which are not parallel with the facade of the building. This makes the problem of extracting the 3D model more complex, but not infeasible.
Because we assume that the roof images are taken from the ground, the skyline detector will always detect the top of the building. In case of a flat roof this is also the top of the building walls. In case of an alternative roof, this will be just the top of the building. The building walls however could lie a lot lower, therefore something else needs to be developed to find the wall heights. It would be useful to develop a method that can detect which roof type we are dealing with, what the wall heights are, and finally generate an entire 3D model.
Some ideas about this are now proposed:

- Use an object detector to detect doors, windows and dormers so the number of floors, the location of the wall-roof separation and the exclusion of some roof types (e.g. a dormer is never located on a flat roof) could be determined.

- Use the Hough transform to search for horizontal lines to detect the wall-roof separation, and use the the ground plane and the top roof line to guid the search. Some building have a gutter, because of this the number of horizontal lines on the wall-roof separation will be larger which could be of great use.

- Use geographic information (a database of roof types) with gps location to classify the roof type.

- The skyline detector detects the building height, if we could use predefined information about the ratio between the wall height and total height of the building, the wall heights could be estimated.

Assuming we determined the roof type,the building height and wall heights, the 3D model could easy be generated. For the *Gable* roof for example this will involve connecting two surfaces from the upper side of the walls with the high roof line (returned by the skyline detector). For the other roof types, the building height and wall height together with a template structure of the roof could be used to generate the 3D model.

## 2.9 References

Under construction