[DRAFT] 3D reconstruction and semantic annotation of urban scenes

Supervisors:
Isaac Esteban and Frans Groen

T. Kostelijk
mailtjerk@gmail.com

June 4, 2012

Contents

Todo list

1 Extracting the 3D building

1.1 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 to extract a countour of the building which is use to generate a 3D model of the building.

We present a stepwise solution. First *Openstreetmap* is used to generate a basic 3D model. 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 part of the basic 3D model (the different parts of the 3D model represent the walls of the building). After this the line segments are projected to the 3D model. The result is used to estimate the height values of the planes of the 3D model. We will now elaborate on each step.

1.2 Method

1.2.1 Generating the 3D model

The 3D model is generated using a basis (groundplane) which is manually extended to an average building height. 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 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 walls.
Because the map is based on aerial images, it is in 2D and contains no information regarding the height of each wall.
The 3D model is generated by starting with the the top view 2D building contour. Next an average building height is used to extend the 2D basis in the opposite gravity direction, resulting in a 3D model.

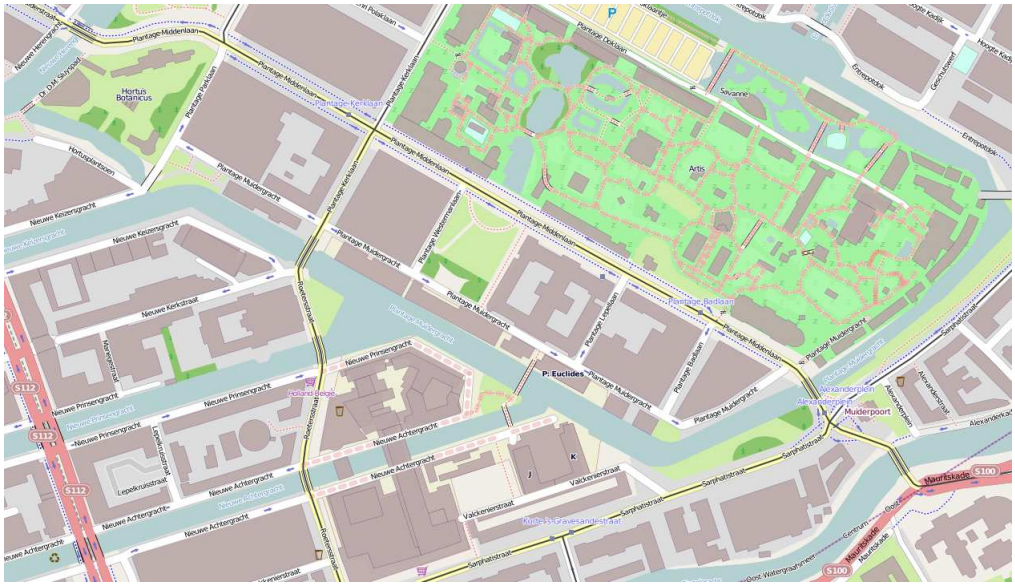

Figure 1: Openstreetmap with annotated buildings

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****.*

An example of the 3D model can be seen in Figure 2.

1.2.2 Extracting line segments

As we want to estimate the height of the buildingwalls of the 3D model we need to know how heigh 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.
Unfortunately some skyline points are outliers. To discard these outliers we decided to detect the inliers and consider the remainder as outliers. In this section we explain how we detect the inliers by extracting straight line segments.

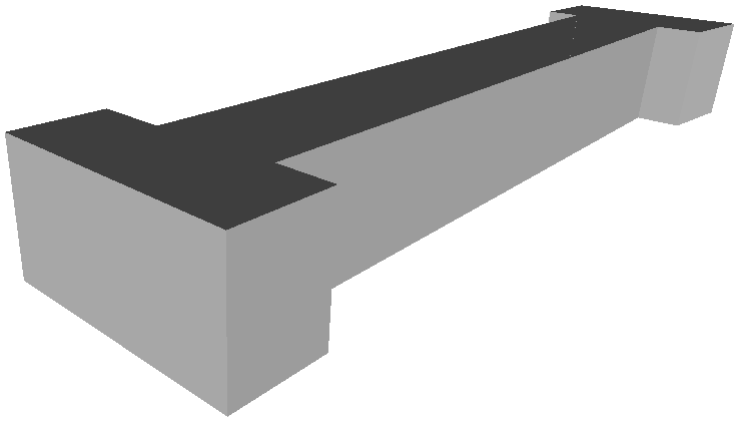

Figure 2: An example of a basic 3D model, generated by extending a basis (2D) model from Openstreetmap to an average building height

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 upperside of a building wall. If we assume a flat roof we can findthe height of the building walls without having to concern for (complex) rooftypes.

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 buildingwalls may have different heights but the roof should be flat.

Hough transform

A widely used method for extracting line segments is the Hough transform [1]. 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 methods like *RANSAC*, who often use an iterative approach). For a detailed explanation of the Hough transform please read section **??**.

The input of the Hough transform that is build in in *MATLAB* is a binary image. This is 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 (r, θ) pair that crosses this particular pixel. Lines (r, θ) 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 returns only the straight parts of the skyline. As these straight skyline parts resemble the building contour we found exactly what we where looking for.

Results of the Hough transform on the output of the skyline detector are displayed and evaluated in the Result section (1.3).

1.2.3 Project the skyline to the building

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 buildingwall 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. This is done in three steps, firtst we need to calibrate the camera, next we project the line segments to the the building and finally we determine the specific building part that is associate whith a line segment.

Camera calibration

To project the line segments to the 3D model we need to know where the camera was positioned when it took the photo. Furthermore we need to know in what way this camera manipulated the image (zoom factors, lens distortion etc.). In other words, we need find the extrinsic and intrinsic parameters, this process is refered to as Camera Calibration.

Extrinsic parameters

The extrinsic parameters present the center of the locations of the camera and the viewing directions. These are unique for every image. In some systems these are recorded by measuring the cameras position and orientation 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* [2], details of this process can be found in the preliminaries

(**??**).

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 (origined from the Fit3d toolbox [2]) comes with a calibration matrix K . For the *Anne* dataset we calculated K by the Bouguet toolbox. This method involes making images of a chessboard in different positions and orientations. For details please refer to the preliminaries (**??**).

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

What can we do with this data after it is calibrated? 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. The value of the 3rd dimension represents the distance from the 3D point to the camera that took the picture. Unfortunately this value is unknown, however because we calibrated the camera, we can reduce the possible points in 3D space to a line.

We know:

- The 2D location of the pixel
- The cameras internal parameters (K)
- The location of the camera center
- The viewing direction of the camera

We can now span the line of possible points in 3D space by two coordinates:

- C, the location of the center of the camera, and
- $A = K'p$, the point that lies on the retina of the camera, where K is the intrinsic calibration matrix and p is the homogeneous coordinate of the pixel.

Because wo now have the two required coordinates, we can set up an equation of the line of possible points in 3D.

$$C + (E - C)t, t \in \mathbb{R}$$

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.
Now we know which possible points in 3D a 2D endpoint of a skyline presents, we can use this to project the skyline to the walls.

1.2.4 Associating line segments with building walls

Building wall appearence 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 project the line segments to to all planes of the 3D building. After this we choose 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 and a direction This direction is the y-direction as assumed by the *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 divided op into 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 (spanned by the camera center and the point on the retina). This line is intersected with the planes of the building walls. This transforms every 2D endpoint in a 3D endpoint resulting in 2 x l x w points in 3D, grouped by the line segments, where 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 every plane. How do we determine the most likely wall? We only calculated the projection to the planes spanned by the walls hence we don't know which line lies on which wall. If we project l to the plane spanned by a wall W we get a line l_{proj_W} in $\mathbb{R}3$. If we assume l to come from the contour of wall W, then l_{proj_W} should have a large intersection with this wall W. 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 is associated with the wall with the largest projection overlap.

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 a different steps. First, different types of overlap are explained. Secondly the algorithm determines the *overlap type*, then the overlap amount is determined and finally the amount of overlap is 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.

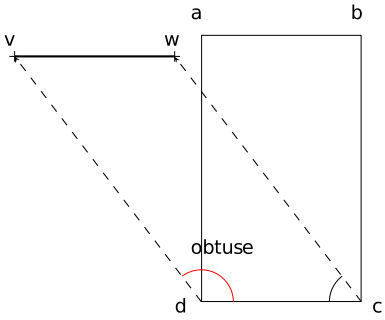

Figure 10a

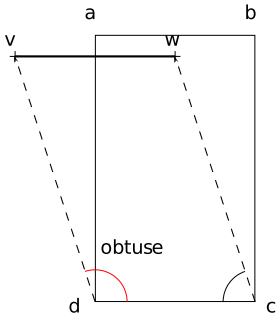

Figure 10b

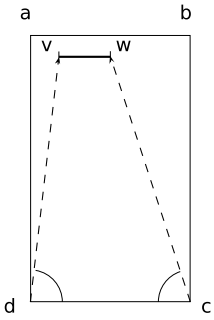

Figure 10c

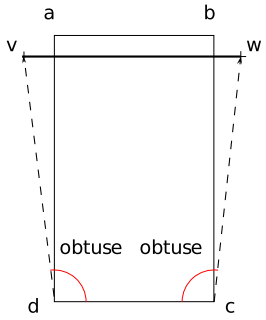

Figure 10d

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).
Table 1 represents the types of overlap with the corresponding number of points that pass the *in polygon* test and their possible line-wall overlap value.

Table 1: Types of overlap with corresponding number op points in polygon

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 (*da* or *cb* from Figure 10b).
We assume the walls to be of infinite height, therefore the partial overlapping line segment always intersects one of the vertical wall sides.
To determine which of the two vertical wall sides is crossed, we determine on which side the point that doesn't lie in the polygon (v) is on. This is done by an angle comparison.
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:

- Ff $\angle d$ 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 α_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 (uncut). 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, the missing overlap should have a negative effect. 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.

Now we have determined the normalized line-wall overlap, we use this to search for the correct line-wall association. This is achieved by associating a line segment with the wall that has the largest line-wall overlap.

1.2.5 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 of the 3D model.
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*.

1.3 Results

Figure 3: Three best ranked lines of the Hough transform on the skyline detector match the building walls

Figure 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 line segment covers the wall until the tree occludes.
Figure 4 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 two 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

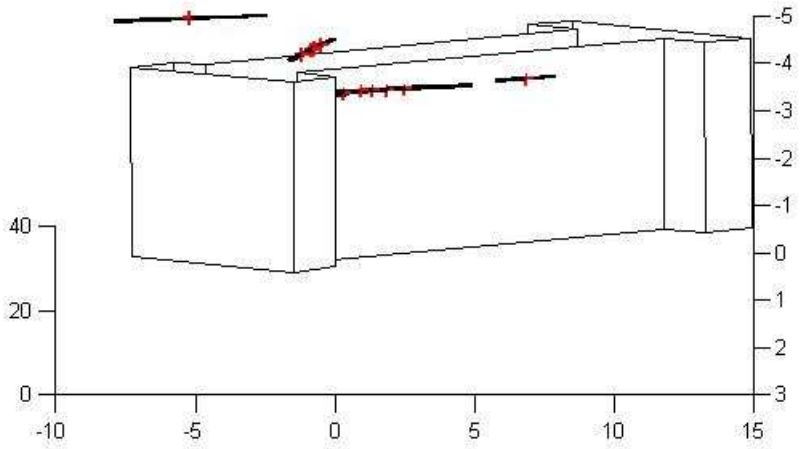

Figure 4: Houghlines projected on the most likely wall

augmented wall. The left and middle wall are extended and the right wall is shortened.

1.4 Discussion

As can be seen in Figure 3, the top three Houghlines correspond the the three most prominent building walls which is a good result. Wat also can be seen is that 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 to produce a good wall height estimate. Furthermore there are 5 other lines (originated from different views) that support the estimate for this wall.

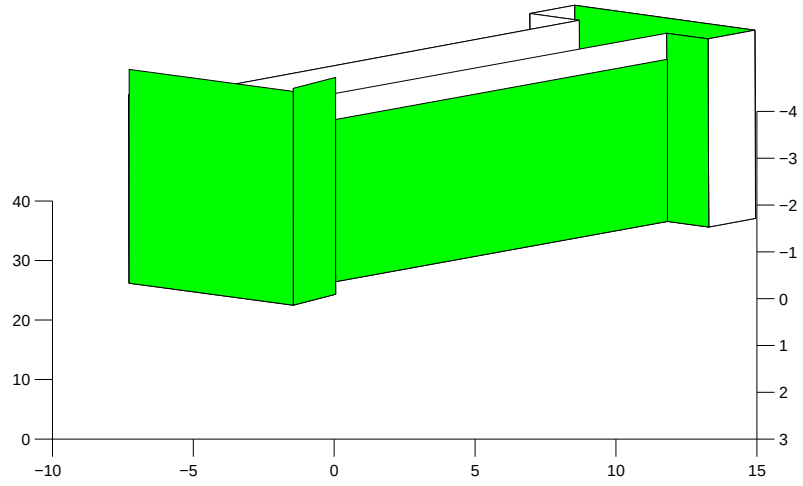

Figure 5: Improved 3D model

1.5 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 significant improvement of a 3D model based on only a few calibrated 2D images.

1.6 Future work

Sometimes 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 *Fill-*

Gap parameter is increased in each iteration. E.g. for the right wall of the building in Figure *this figure will present at the next update* two iterations would be enough, the *FillGap* parameter needs to be 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.

1.6.1 Alternative roofs

We assumed a flat roof, this doesn't mean that our method is unusable if we discard this assumption. E.g. without addaptations the method could be used to determine the (maximum) building height which is a useful application.

If we discard the flat roof assumption it should also be possible to extract a full 3D model. We will now consider other roof types and discuss what adaptations the system should require to handle these. In Figure 6, 6 different roof shapes are displayed.

Consider the *Gable Roof*, 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.

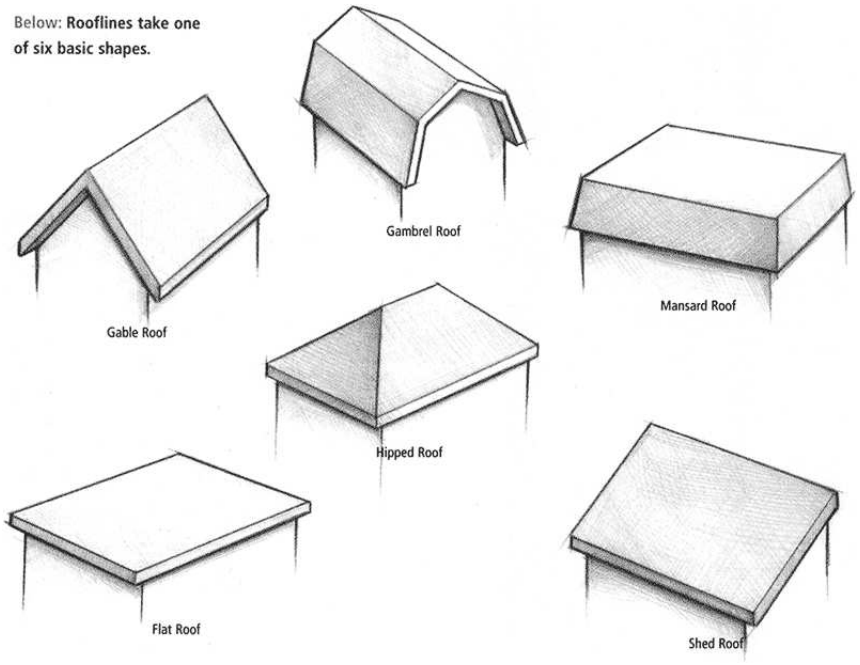

Figure 6: Different types of roofs

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 easily 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.

References

[1] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15:11–15, January 1972.

[2] I. Esteban, J. Dijk, and F.C.A. Groen. Fit3d toolbox: multiple view geometry and 3d reconstruction for matlab. In *International Symposium on Security and Defence Europe (SPIE)*, 2010.