Improve 3D models from 2D images

T. Kostelijk
mailtjerk@gmail.com

January 30, 2012

Contents

1 Window detection

1.1 Introduction

Figure 1: Original image

From the previous section we know that from a serie of images a 3D model of a building can be extracted. Furthermore we saw that with the 3D information the scene could be viewed from another viewing point.
We projected the scene to a frontal view of a building, where a building wall appears orthogonal, and showed what interesting possibilities this opens. One example is robust window detection. In this section we present three developed methods for robust window detection and discuss the effect of the scene transformation.
We begin with an approach that is invariant to viewing direction. Then we present our second method that assumes orthogonal and aligned windows.

Figure 2: Rectified image

Then we present a third feature based method. Finally we show the power of combining the methods.

1.1.1 Related work

TODO

1.2 Edge detection and Houghline extraction

Edge detection and Houghline extraction is done as is described in chapter ?? The results can be seen in Figure 4 and 5.

Figure 3: Original image

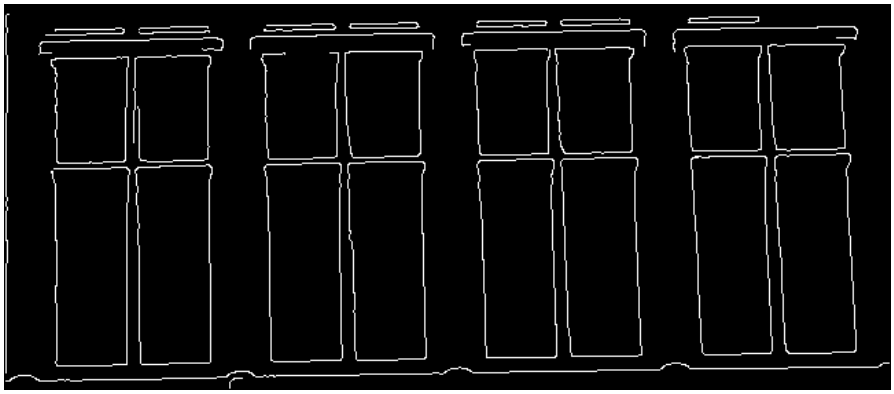

Figure 4: Result edge detection

1.2.1 Efficient Projecting

We are interested in the frontal view of the building and it would be straight forward to project the original image, however this is computational expen-

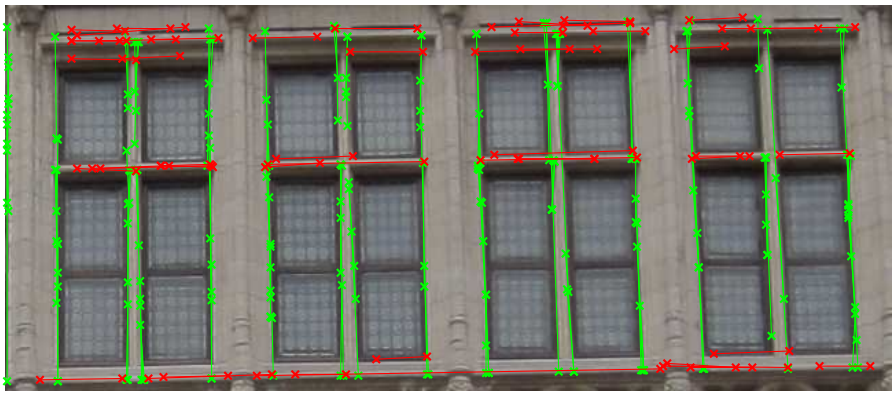

Figure 5: Houghlines with endpoints

sive. To keep the computational cost to a minimum we project only the Houghlines. The edge detection and Houghline extraction is done on the original unprojected image. We only project the Houghline segment endpoints. If h is the number of Houghlines, the number of projections is $2h$ When we project the original image this is w x h where w,h are the dimensions of the image. To give an indication for dataset this means 600 projections in stead of 1572864.

1.3 Method I: Connected corner approach

1.3.1 Situation and assumptions

We introduce the concept *connected corner*, this is a corner that is connected to a horizontal and vertical line. In this method we search for connected corners based on edge information. The connected corners give a good indication of the position of the windows, as a window consists of a complex structure involving a lot of connected horizontal and vertical lines.
In this approach the windows could be arbitrarily located and they don't need to be aligned to each other neither to the X and Y axis of the image.

1.3.2 Method

From the edge image we extract two groups of Houghlines, horizontal and vertical. We set the θbin ranges in the Hough transform that control the allowed angles of the Houghlines to extract the two groups.
Next we pair up horizontal and vertical lines to form a connected corner. Often a connected corner is not fully connected or over connected. We consider different types of connected corners, see Figure 6

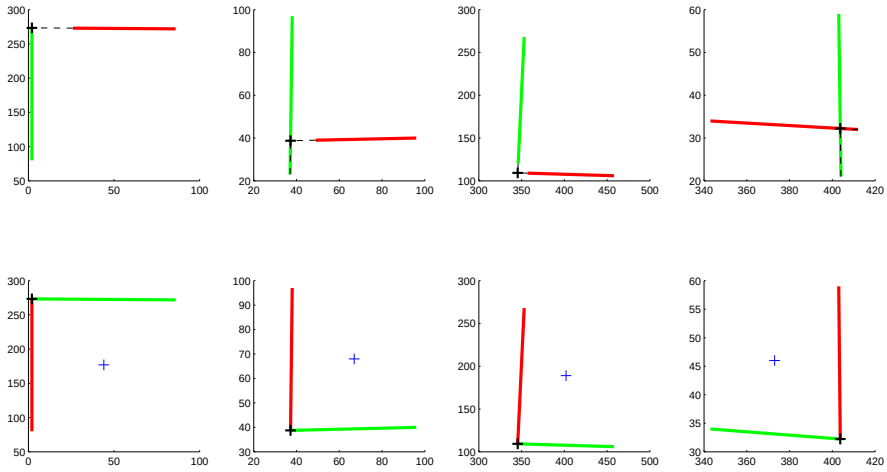

Figure 6: First row: different type of connected corner candidates. Second row: the result the clean connected corner

To clean up the lines, the algorithm discards intersections between the horizontal and vertical Houghlines that are far. Two intersection point distances are measured: d_h for the horizontal Houghline and d_v for the vertical Houghline. If the intersection falls on both associated Houghlines, the total distance $D = 0$. Otherwise the Euclidean distance is measured from the intersection to the closest endpoint. This is done for both Houghlines. If the intersection falls outside both Houghlines (Figure 6(IV)) ($d_h > 0$ and $d_v > 0$), the total distance is calculated by $D = (d_h + d_v)/2$.
Next D is compared to a *maximum intersection distance* threshold $midT$. And if $D <= midT$, the intersection is close enough to form a connected corner.

After two Houghlines are classified as a connected corner, they are stretched or trimmed, depending on the situation. The results are shown in the second row in Figure 6. In Figure 6(I) the horizontal line is stretched. Figure 6(II) shows that the vertical line is trimmed. In Figure 6(III) both lines are stretched. At last Figure 6(IV) shows how both lines are trimmed.
Because we know the orientation of the connected corner we can estimate where a window could be located. We add a vote in the middle of the window. This is represented as a blue cross in Figure 6. This coordinate is retrieved using the X coordinate of the middle point of the horizontal line and the Y coordinate of the middle point of the vertical line of the connected corner. Note that this is officially not allowed as we did not assume orthogonal windows neither did we assume the windows to be aligned with the X and Y axis of the image.

1.3.3 Results

1.3.4 Future work

Connect more parts of the window to form U shaped windows or complete rectangles.
Increase vote when window has small sub windows that are included.
More accurate middle point of window estimation.

1.4 Method II: Histogram based approach

1.4.1 Situation and assumptions

In this method we assume that the wall containing the windows is rectified. To be more precise we assume the windows have orthogonal sides. Furthermore we assume that the windows are aligned.

1.4.2 Method

The main idea is that we extract the alignment of the windows based on calculating histograms of the Houghlines' endpoints.

Alignment lines We introduce the concept alignment line. We define this as a horizontal or vertical line that aligns multiple windows. In Figure 8 we show the alignment lines as two groups, horizontal (red) and vertical (green) alignment lines. The combination of both groups give a grid of rectangles that has to be classified as window or non-window. First we explain the extraction of the alignment lines which consist of several steps.

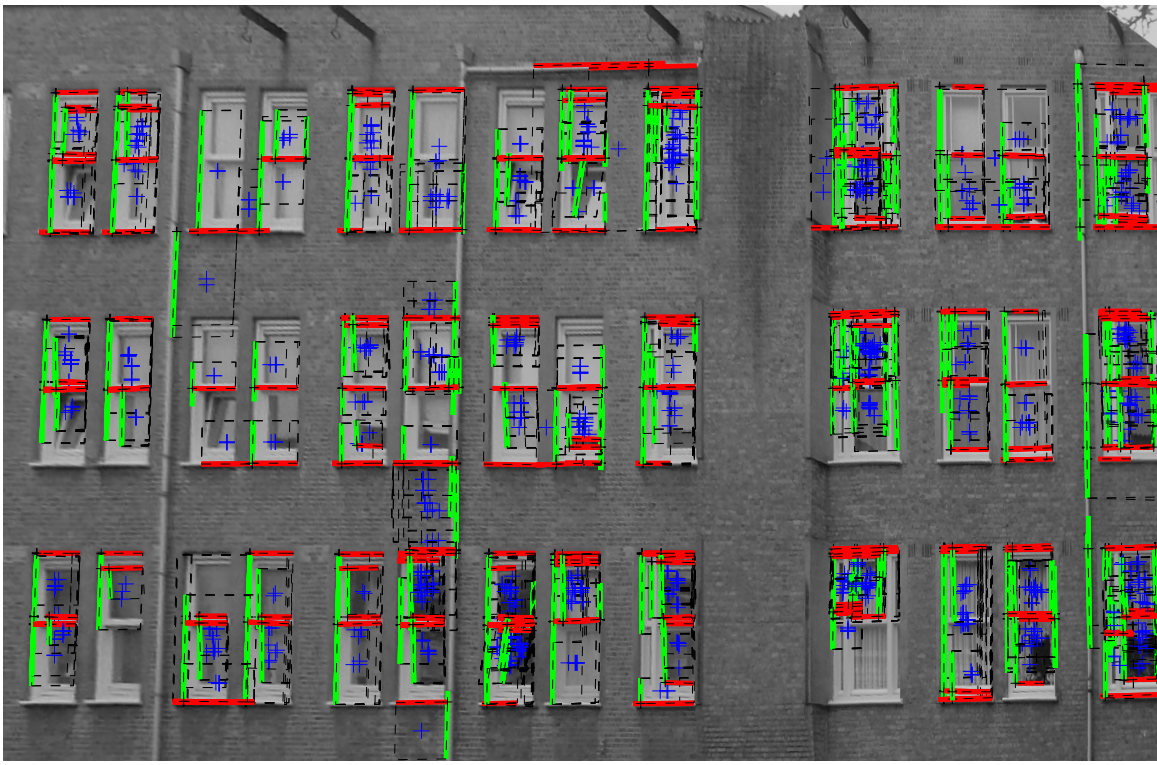

Figure 7: TODO

We begin by extracting the coordinates of the endpoints of the Hough transformed line segments. We store them in two groups, horizontal and vertical (crosses in Figure 5).
We project the coordinates to the axis that is orthogonal to the group. This means that the horizontal Houghlines are projected to the X axis and the vertical Houghlines are projected in the Y axis, transforming the data in two groups of 1 dimensional coordinates.
We calculate two histograms H(orizontal) and V(ertical), containing respectively w and h bins where wxh is the dimension of the image. The histograms are presented as small yellow bars in Figure 8.
The peaks are located at the positions where an increased number of Houghlines start or end. These are the interesting positions as they are highly

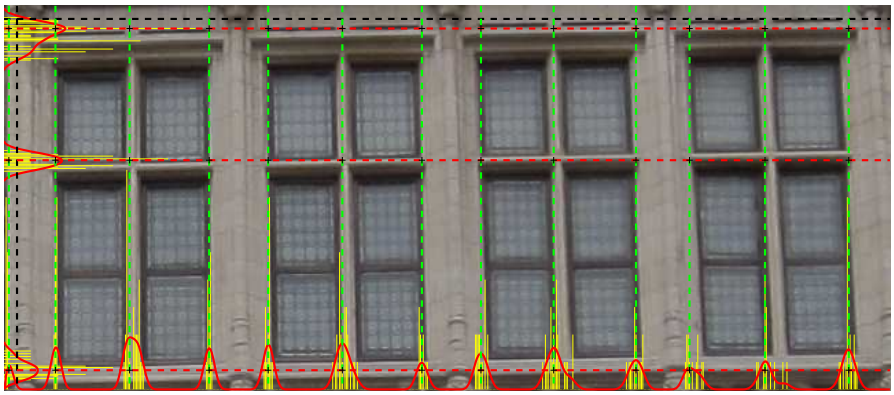

Figure 8: (smoothed) Histograms and window alignment lines

correlated to the alignment lines of the windows.
It is easy to see that the number of peaks is fare more then the desired number of alignment lines. A common solution would be to decrease the number of bins of the histograms. A disadvantage of this method is that this also decreases the accuracy. Therefor we keep the maximum resolution and, instead, smooth the function. The smooth function uses a moving average filter. The result, red lines in Figure 8 , is a smooth function which contains the right number of peaks. Also the peaks are located at the right positions.
Next step is to calculate the exact positions of these peaks. Before we find the peak positions we extract the peak *areas* by thresholding the function. The two thresholds are presented as black dotted lines in Figure 8.
Next we create a binary function P that returns 1 for positions that are contained in a peak, i.e. are above the threshold, and 0 otherwise.
We detect the peak areas by searching for the positions where the function passes the threshold line. If we loop through the values of P we detect a peak-start on position s if $P(s-1), P(s) = 0, 1$ and a peak-end on e if $P(e-1), P(e) = 1, 0$. I.e. if $\mathrm{P} = 0011000011100$, then two peaks are present. The first covers positions $(3, 4)$, the second covers $(9, 10, 11)$.
Having classified the peak areas, the next step is to extract the peak positions. Each peak area has only one peak and, since we used an average

smoothing filter,the shape of each peak area is concave. Therefor we can extract the peaks by locating the max of each peak area. On these locations we have drawn the window alignment lines, dotted red and dotted green lines in Figure 8
The image is now divided in a grid of rectangular areas. The next challenge is to classify the rectangles as a window or non-window area.

1.4.3 Results

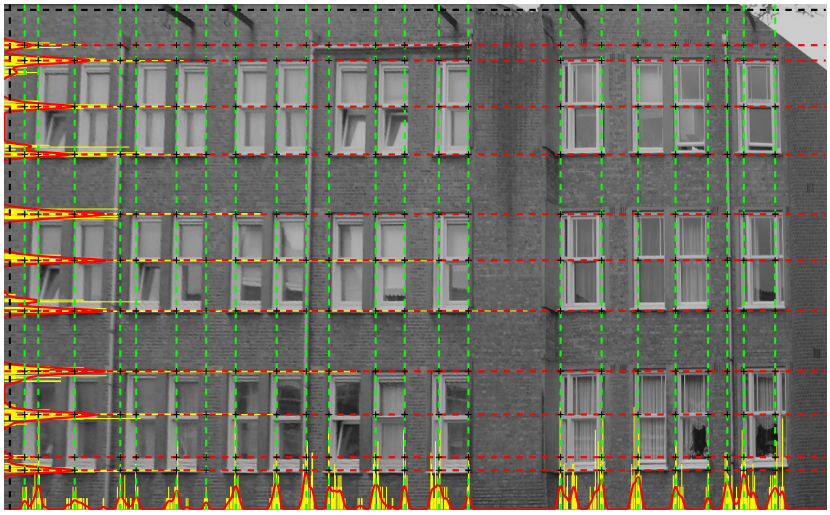

Figure 9: TODO

1.4.4 Future work

Some alternative ideas to classify the rectangles as window or non-window

- sum and normalize edge pixels for every block, large amount of edge pixels means window behind rectangle (working on it right now)

1.5 Method III: Feature detection approach

TODO figure and explain contribution and method of multi scale harris corner detector

1.6 Fusing the methods

TODO

1.7 Results

1.8 Discussion

1.9 Conclusion and Future work