Improve 3D models from 2D images

T. Kostelijk
mailtjerk@gmail.com

January 29, 2012

Contents

1 Window detection

1.1 Introduction

Figure 1: Original image

From the previous section we saw that from a serie of images a 3D model of a building could be extracted. Furthermore we saw that with the 3D information the scene could be viewed from another viewing direction.
We projected the scene to a frontal view of a building, where a building wall appears orthogonal, and show what interesting possibilities this opens. One example is robust window detection. In this section we present three developed method 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 result can be seen in Figure 4 and 5.

Figure 3: Original image

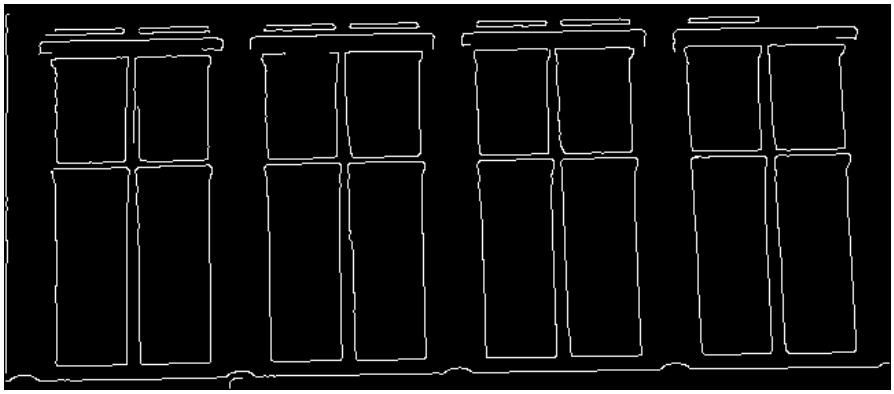

Figure 4: Result edge detection

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-

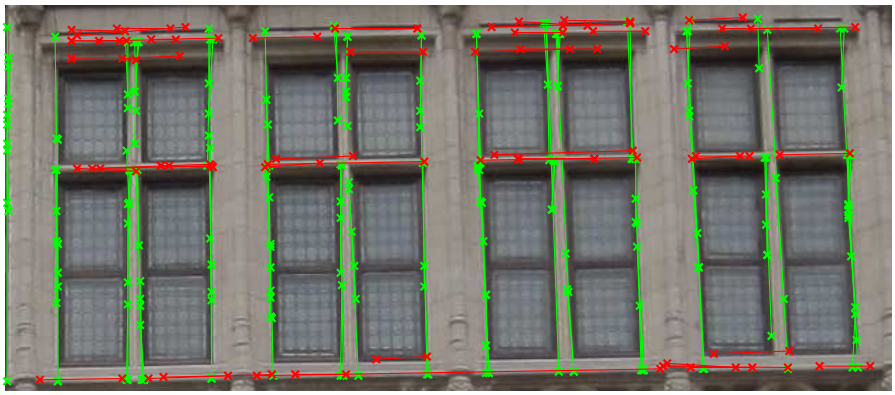

Figure 5: Houghlines with endpoints

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. Leaving the projection done on 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 wxh 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 consist of a complex structure involving a lot of horizontal and vertical lines.
In this approach the windows could be arbitrarily located and don't need to be aligned.

1.3.2 Method

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

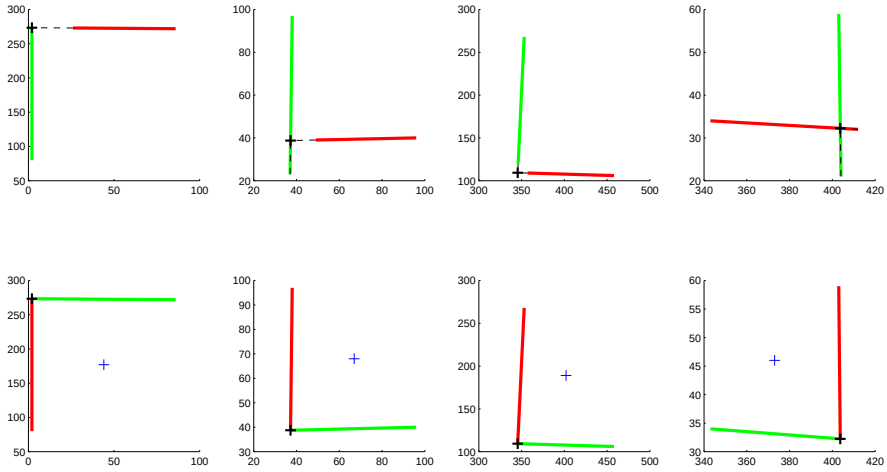

Figure 6: First row different type of connected corner candidates. Second row the result the clean connected corner

The algorithm searches for intersections between the horizontal and vertical Houghlines. If the intersection is near the endpoints of the Houghlines the lines are ready to form a connected corner. The lines are first stretched or trimmed dependent of the situation. Then a clean connected corner is created by connecting the line segment endpoints to the intersection. The result is shown in the second row Figure 6. *Frans: shall I put a pseudo code algorithm here?*
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 coordinate is retrieved using the X coordinate of the mile point of the horizontal line and the Y coordinate of the mile point of the vertical line of the connected corner. This is represented as a blue cross in Figure 6.

1.3.3 Results

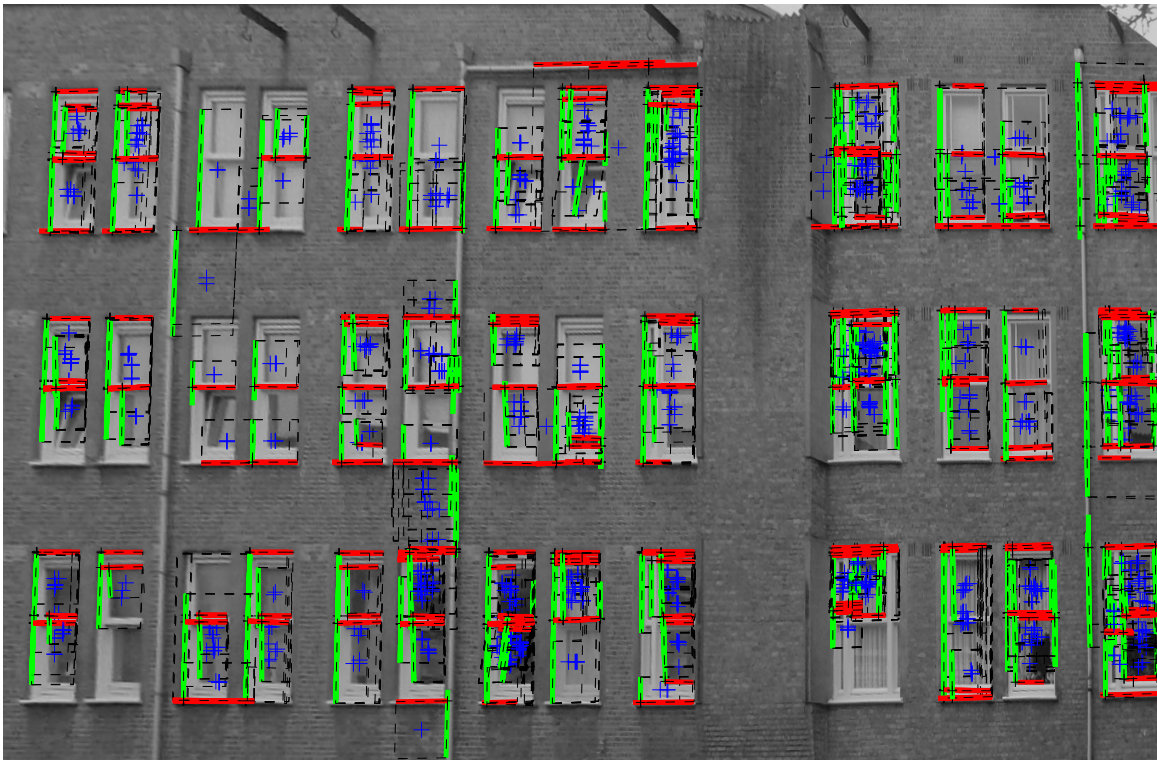

Figure 7: TODO

1.4 Method II: Histogram based approach

1.4.1 Situation and assumptions

In this method we assume the wall containing the windows to be rectified. To be more precise we assume the windows to have orthogonal sides. Furthermore we assume the windows to be 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.

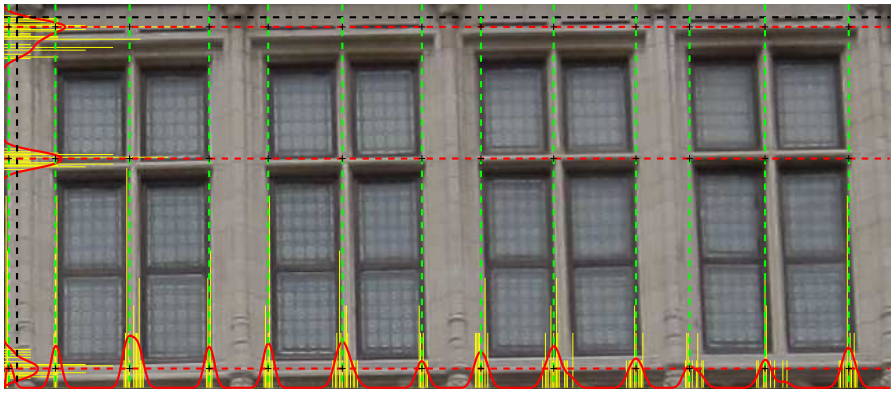

Figure 8: (smoothed) Histograms and window alignment lines

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 represent the alignment lines as two groups, horizontal (red) and vertical (green) alignment lines. The combination of both groups give a grid of rectangles that have to be classified as window or non-window. First we explain the extraction of the alignment lines which consist of several steps. 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, leaving 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 dimensions of the image. The graphs of the histograms are presented as small yellow bars in Figure 8

The peaks are located at the positions where an increased number of Hough-lines start or end. These are the interested positions as they are highly correlated to the alignment lines of the windows.
As can be seen 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 smooth the function using a using 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, see the black dotted line in Figure 8 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 ends on e if P(e-1),P(e)=1,0. I.e. if 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, each peak area has a concave shape. Consequently we can easily extract the peaks by locating the max of each peak area. On these locations we have drawn the window alignment lines, dotted red and yellow 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

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

1.5 Method III: Feature detection approach

TODO explain contribution and method of multi scale harris corner detector

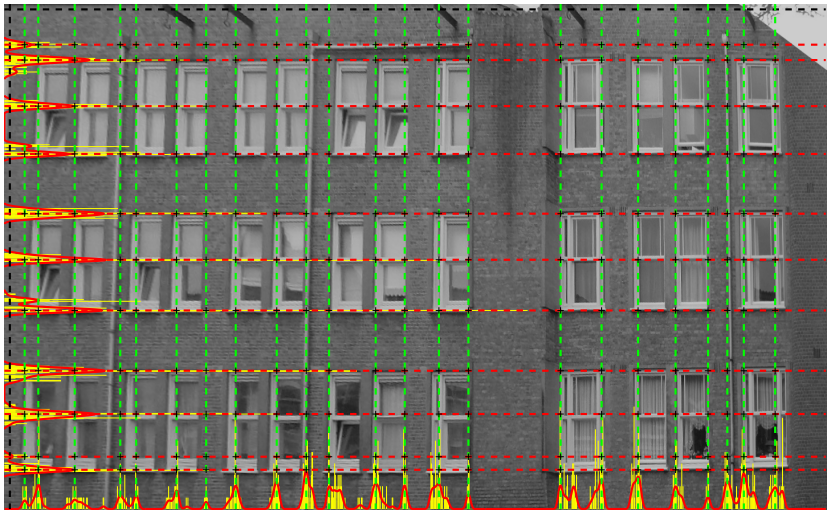

Figure 9: TODO

1.6 Fusing the methods

TODO

1.7 Results

1.8 Discussion

1.9 Conclusion and Future work