

# INTERPRETING URBAN SCENES BY SKYLINE AND WINDOW DETECTION / 3D URBAN SCENE UNDERSTANDING BY SKYLINE AND WINDOW DETECTION

T. Kosteljik  
mailtjerk@gmail.com

March 23, 2012

## Contents

<b>1</b>	<b>Window detection</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Related work . . . . .	3
1.3	Method I: Connected corner approach . . . . .	5
1.4	Method II: Histogram based approach . . . . .	9
1.5	Results . . . . .	14
1.6	Discussion . . . . .	15
1.7	Conclusion and Future work . . . . .	15
1.8	Efficient Projecting . . . . .	16

# 1 Window detection

## 1.1 Introduction

This chapter is about one of the tasks of semantic urban scene interpretation, Window detection. Semantic interpretation of urban scenes is used in a wide range of applications.

**3d City models** Manual creation of 3d models is a time consuming and expensive procedure. Therefor semantic models are used for semi automatic 3d reconstruction/modelling. The semantic understanding is also used in 3d city models which are generated from aerial or satellite imagery. The detected (doors and) windows are mapped to the model to increase the level of detail. Some other applications can automatically extract a CAD-like model of the building surface.

**Historical buildings documentation and deformation analysis** In some field of research, Historical buildings are documented. The complex structures that are contained in the facades are recorded and reconstructed. Window detection plays a central role in this. Another field of research is the analysis of building deformation in areas containing old buildings. Window detection provides information about the region of interest that could be tracked over time for an accurate deformation analysis.

**Interactive 3d models** There are some virtual training applications that are designed for emergency response who require interaction with a 3d model. For the simulation to be realistic it is important to have a model that is of high visual quality and has sufficient semantic detail (i.e. contains windows). This is also the case for a fly-through visualization of a street with buildings. Other applications that require semantic 3d models are virtual tourism, visual impact analysis, driving simulation and military simulation systems.

**Augmented reality** Some mobile platforms apply augmented reality using facade and window detection to make a accurate overlay of the building. An example overlay is the same building but 200 years earlier. Semantical information is used to not only identify a respective building, but also find his exact location in the image. The accuracy and realistic level of the 3d model are vital for a successful simulation. And because the applications are

mobile, very fast building understanding algorithms are required. Window detection plays an important role in these processes.

**Building recognition and urban planning** Building recognition is used in the field of urban planning where the semantic 3d models are used to provide important references to the city scenes from the street level. Building recognition is done using large image datasets where the buildings are mostly described by local information descriptors. Some approaches try to describe the 3D building with laser range data. Some methods fuse the laser data with ground images. However those generated 3D models are a mesh structure which doesn't make the facade structure explicit. For a more accurate disambiguation, other types of contextual information are desired. The semantical interpretation of the facade can provide this need. In this context, window detection can be used as a strong discriminator.

We can conclude that window detection plays an important role in the interpretation of urban scenes and is applied in a wide range of domains. This section is about two developed methods for robust window detection.

We start with discussing related work and putting our work in context. Then we begin with a window detection approach that is invariant to viewing direction. After this we present our second method that assumes orthogonal and aligned windows. Finally we show and discuss results.

## 1.2 Related work

A large amount of research is done on semantical interpretation of urban scenes. First we discuss related work that has a big overlap with our research in detail. After this, we briefly discuss the research that is done on window detection using approaches from a different point of view.

### 1.2.1 Similar approaches

Pu and Vosselman [4] use laser images together with Hough line extraction to reconstruct facade details. They solve inconsistency between laser and image data and improve the alignment of a 3d model with a matching algorithm. In one of the matching strategies they compare the edges of a 3d model to Hough lines of ground images. They match the lines by comparing the angle, location and length difference of the model edges with the extracted Houghlines. These criteria is also used in our approach.

They also detect windows and use them to provide a significant better alignment of the 3d model. The windows are extracted from the holes from laser points of a wall, these results where far from accurate.

To summarize, the work of Pu and Vosselman provides a useful practical application of window detection and it amplifies the need for a robust window detection technique that is independent of laser data.

In [5] Rocky et all developed a window detector that is build on the primary work of Lee and Nevatia [2] (which is discussed next). In order to be able to assume aligned windows they rectify the facade. After this they apply a threshold on an orthogonal projection of the extracted edges. For example they use a vertical edge projection to establish the horizontal division of the windows. This is very similar to our approach, although we only project line segment endpoints.

The next step, labeling the areas containing windows, is however very different as they use color to disambiguate the windows. To be more precise, they convert the image to CIE-Lab color space and use k-means to classify the windows. Although this method is robust, both color transformation and k-means clustering are very computational expensive. In our method we use the same source, edge information, for the window alignment and for the window labeling. As we don't require color transformation and only apply math on line segment endpoints, our algorithm performs in real-time. As in the work of Rocky et all [5] Lee et all [2] perform orthogonal edge projection to find the window alignment. As different shape of windows can exist in the same column, they use the window alignment as a hypothesis. Then, using this hypothesis, they perform a refinement for each window independently. Although this comes with accurate results, the iterative refinement is a computational expensive procedure. Therefor we do not consider the refinement process in our research.

### 1.2.2 Other approaches

Muller et all [3] detect symmetry in the building. The symmetry is detected in the vertical (floors) and horizontal (window rows) direction. The use shape grammars to divide the building wall in tiles, windows, doors etc. The results are used to derive a 3d model of high 3d visual quality.

Using a thermal camera, Sirmacek [6] detects heat leakage on building walls as an indicator for doors or windows. Windows are detected with L-shaped features as set of *steerable filters*. The windows are grouped using *perceptual organization rules*.

Ali et all [1] describe the windows with Haar like features which are fed into a (Ada boost) cascaded decision tree.

### 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

**Edge detection and Houghline extraction** Edge detection is done as is described in chapter (*not included chapter*) From the edge image we extract two different 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. The horizontal group has a range of [-30..0..30] degrees, where 0 presents a horizontal line. The vertical group has a range of [80..90..100] degrees. This ranges seem to work good on an empirical basis for all datasets. The results can be seen in Figure 3 and 4.

**Extract connected corners** Next we pair up horizontal and vertical lines to discard found edges that are not originated from a window. These paired up lines form a connected corner. Often a connected corner is not fully connected or over connected. We consider different types of connected corners, see Figure 5.

To clean up the lines, the algorithm discards intersections between the horizontal and vertical Houghlines that are apart. 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 5(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  $D \leq midT$ , the intersection is close enough to form a connected



Figure 1: Original image



Figure 2: Rectified image



Figure 3: Result edge detection

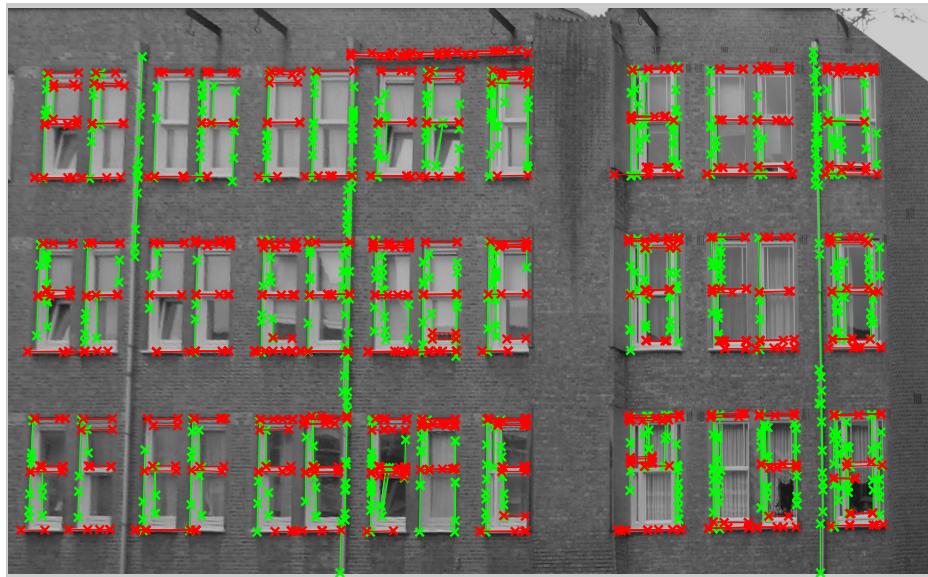


Figure 4: Houghlines with endpoints

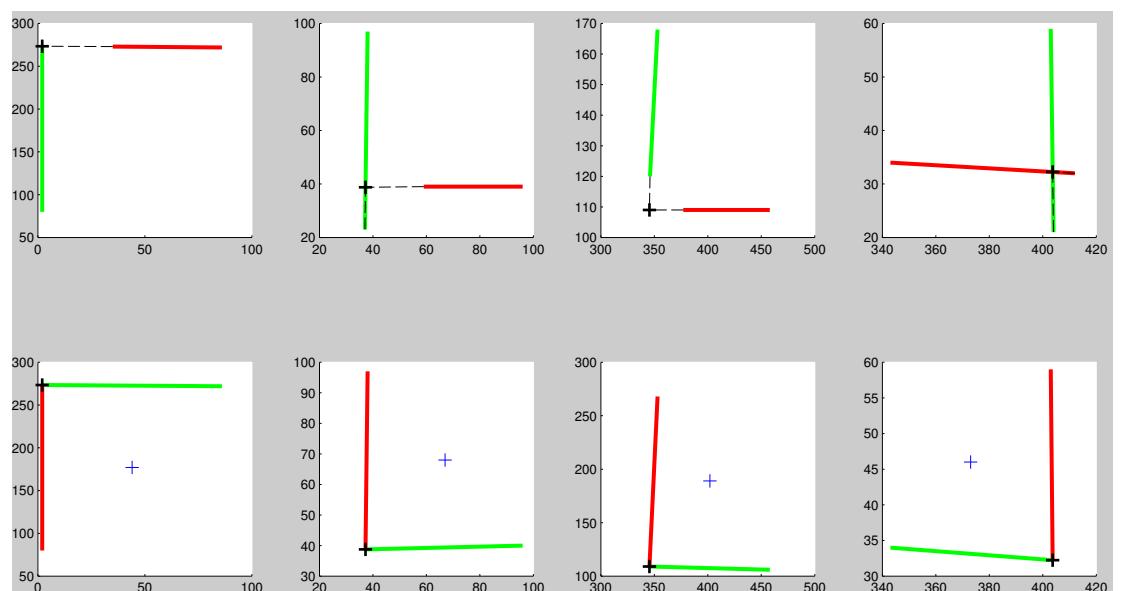


Figure 5: First row: different type of connected corner candidates. Second row: the result the clean connected corner

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 5. In Figure 5(I) the horizontal line is stretched. Figure 5(II) shows that the vertical line is trimmed. In Figure 5(III) both lines are stretched. At last, Figure 5(IV) shows how both lines are trimmed.

**Extract window areas** To retrieve the actual windows, each connected corners is mirrored on its diagonal. The connected corner now contains four sides which form a quadrangle window area. All quadrangles are filled and displayed in Figure 12 and 16.

## 1.4 Method II: Histogram based approach

### 1.4.1 Introduction

From the previous chapter we know that from a series of images, a 3D model of a building can be extracted. Furthermore we saw that using this 3D model the scene could be converted to another viewing point.

For accurate and robust window detection we projected the scene to a frontal view of a building, where a building wall appears orthogonal. This frontal view enables us to assume orthogonality and alignment of the windows. We exploit this properties to build a robust window detector. First we determine the the alignment of the windows and then we label the areas that contain the windows.

### 1.4.2 Situation and assumptions

To be more precise in our assumptions, we assume the windows have orthogonal sides. Furthermore we assume that the windows are aligned. This means that a row of windows share the same height and  $y$  position. For a column of windows the width and  $x$  position has to be equal. Note that this doesn't mean that all windows have the share the same size.

### 1.4.3 Method

The extraction of the windows is done in different steps. First the alignment of the windows is determined, this is based on collecting the Houghlines' start and endpoints. Then we use this alignment to divide the image in window

or not window regions. Finally these regions are classified and combined which gives us the windows.

**Extract Window alignment** We introduce the concept alignment line. We define this as a horizontal or vertical line that aligns multiple windows. In Figure 6 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 we classify as window or non-window areas.

How do we determine this alignment lines? We make use of the fact that among a horizontal alignment line a lot of horizontal Houghlines start and end (see red crosses in Figure 4. For the vertical alignment lines the number of vertical Houghline start and ends is high (see green crosses in Figure 4.

We begin by extracting the coordinates of the endpoints of the Hough transformed line segments. We store them in two groups, horizontal and vertical. We discard the dimension that is least informative by project the coordinates to the axis that is orthogonal to its group. This means that for each horizontal Houghline two coordinates are projected to the X axis and for each vertical Houghline two coordinates are projected to the Y axis. We have now transformed the data in two groups of 1 dimensional coordinates which represent the projected position of the Houghlines.

Next we calculate two histograms  $H(\text{horizontal})$  and  $V(\text{vertical})$ , containing respectively  $w$  and  $h$  bins where  $w \times h$  is the dimension of the image. The histograms are presented as small yellow bars in Figure 6.

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 correlated to the alignment lines of the windows.

It is easy to see that the number of peaks is far more than 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 comes with a price, it decreases the accuracy. Therefor we keep the maximum resolution and, instead, smooth the values using a moving average filter. The result, red lines in Figure 6, is a smooth function which contains the right number of peaks. The peaks are located at the average positions of the window edges. Next step is to calculate the peak areas and after this the peak positions.

Before we find the peak positions we extract the peak *areas* by threshold-

ing the function. To make the threshold invariant to the values, we set the threshold to  $0.5 \cdot \max \text{Peak}$ . (This value works for most datasets but is made up and can be altered). 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  $P = 1$  (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  $P = 0011000011100$ , then two peaks are present. The first peak covers positions (3, 4), the second peak 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 the peaks are often concave. Therefor we extract the peaks by locating the max of each peak area. These locations are used to draw the window alignment lines, they can be seen as dotted red lines and dotted green lines in Figure 6.

The image is now divided in a grid of rectangular areas. The next challenge is to classify the areas as window and non-window areas: the window classification.

**Window classification** Instead of classifying each rectangle independently we classify full rows and columns as window or non-window areas. This approach results in more accurate classification as it uses a full row and column as evidence for a singular window.

The method exploits the fact that the windows are assumed to be aligned. A row that contains windows is remarkable by its high amount of vertical Houghlines, Figure 4 (green). For the columns the number of horizontal Houghlines (red) is high at window areas. We use this property to classify the rows/columns.

For each row the number of vertical Houghline pixels that lie in this row are summed up. (Remark that with this method we take both the length of the Houghlines and amount of Houghlines implicitly into account.)

To prevent the effect that the size of the row influences the outcome, this total value is normalized by the size of the row.

$$\forall R_i \in \{1..numRows\} : R_i = \frac{\text{HoughlinePxCount}}{R_i^{\text{width}} \cdot R_i^{\text{height}}}$$

Leaving us with  $\|R\|$  (number of rows) scalar values that give a rank of a row begin a window area or not. This is also done for each column (using

the normalized horizontal amount of Houghlines pixels) which leaves us with  $C$ .

If we examine the distribution of  $R$  and  $C$ , we see two clusters appear: one with high values (the rows/columns that contain windows) and one with low values (non window rows/columns). For a concrete example we displayed the values of  $R$  in Figure 7. Its easy to see that the high values, row 4,5,7,8,10 and 11, correspond to the six window row areas in Figure 17.

How do we determine which value is classified as high? A straight forward approach would be to apply a threshold, for example 0.5 would work fine. However, as the variation of the values depend on (unknown) properties like the number of windows, window types etc., the threshold maybe classify insufficient in another scene. Hence working with the threshold wouldn't be robust.

Instead we use the fact that a row is either filled with windows or not, hence there should always be two clusters. We use *k-means* clustering (with  $k = 2$ ) as the classification procedure. This results in a set of Rows and Columns that are classified as window an non-window areas.

The next step is to determine the actual windows  $W$ . A rectangular area  $w \in W$  that is crossed by  $R_j$  and  $C_k$  is classified as a window iff *k-means* classified both  $R_j$  and  $C_k$  as window areas. These are displayed in Figure 17 as green rectangles.

The last step is to group a set of windows that belong to each other. This is done by grouping adjacent positively classified rectangles. These are displayed as red rectangles in Figure 17.

As the figure gives a binary representation of the windows it is not possible to see the probabilities behind the classification. Therefor we developed a probabilistic function.

$$P(R_i) = \frac{R_i}{\max(R)}$$

$$P(C_i) = \frac{C_i}{\max(C)}$$

$$P(w) = \frac{P(R_i) + P(C_i)}{2}$$

As you can see  $P$  is normalized, this is to ensure the value of the maximum probability is exactly 1. The results can now be relatively interpreted, e.g. if the rectangle's  $P = 0.5$  then the system nows for 50 percent sure it is a window, compared to its best window ( $P = 1$ ). And, as the normalization implies this, there are always one or more window with  $P = 1$ .

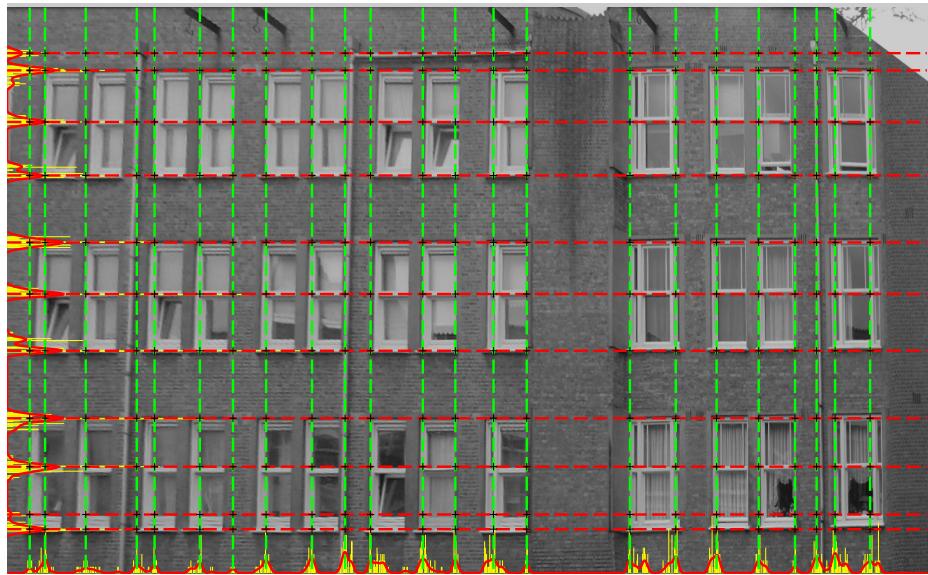


Figure 6: (smoothed) Histograms and window alignment lines

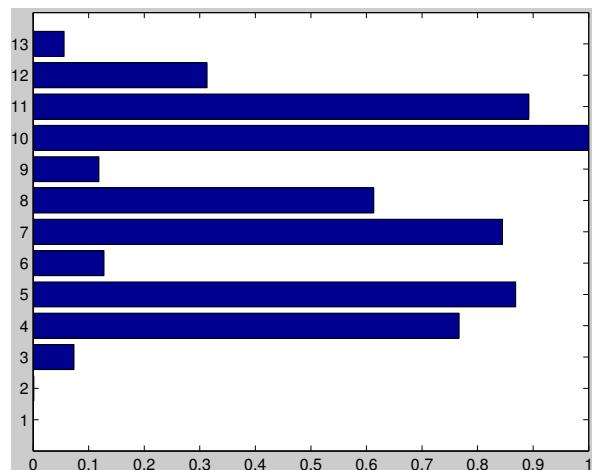


Figure 7: Normalized vertical Houghline pixel count of the rows (R)

To get insight about the probabilities that lie behind the individual rows and columns we designed another representation in Figure 8. The whiter the area the more probable a rectangle is classified as a window.

## 1.5 Results

We tested both methods on different datasets.

*TODO include images other datasets  
todo compair methods and explain differences*

## 1.6 Discussion

*todo discuss results*

**Method I: Connected corner approach** The big advantage of this method is that it doesn't require the windows to be aligned. Furthermore it's robust to a variation in window sizes and types. This makes this approach suitable for a wide range of window scenes where no or few prior information about the windows is known. *TODO*

**Method II: Histogram based approach** One drawback is that the outcome is non-deterministic, as it depends on to the random initialization of the cluster centers.

*TODO*

**Occlusion** If the image isn't the frontal view of the buildingwall we project the image see section ?This projection comes with some difficulties, occlusion. In a few cases an buiding wall extension (middle of figure 1) a drainpipe or the building wall itself is occluding a part of the window. The less frontal the view, the more occlusion negatively effects the cleanliness of the projection. However, this occlusion artefact is in most cases no problem as the system combines the windows probabilities.

## 1.7 Conclusion and Future work

### 1.7.1 Conclusion

We showed that projecting the image to a frontal view is a good preprocessing step of a robust window detector. *TODO*

### 1.7.2 Future work

**Method I: Connected corner approach** It would be nice to have a clustering algorithm that groups connected corners to a window. For this method it would be useful to assume the window size as this correlates directly to the inter-cluster distance.

It would also be nice to incorporate not only the center of the connected corner as a parameter of the cluster space but also the length and position of the connected corners' horizontal and vertical line parts. The inter cluster distance and the number of grouped connected corner could form a good source for the probability that a window is found.

We only developed L-shaped connected corners, it would be nice to connect more parts of the window to form U shaped connected corners or even complete rectangles.

The later is difficult because the edges are often incomplete due to for example occlusion or the angle of viewing.

**Method II: Histogram based approach** It would be nice to investigate the effect of the occlusion and to exploit the robustness of the window detector under extreme viewing angles. For example the viewing angle could be plotted against the percentage of correct detected windows. *TODO*

## 1.8 Efficient Projecting

As we are interested in the frontal view of the building, it would be straight forward to project the original image. However this is computational very expensive as each pixel needs to be projected. 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  $w \times h$  where  $w, h$  are the dimensions of the image. To give an indication, for the *Spil* dataset this means 600 projections in stead of 1572864.

However for the purpose of display we also presented the rectified images.

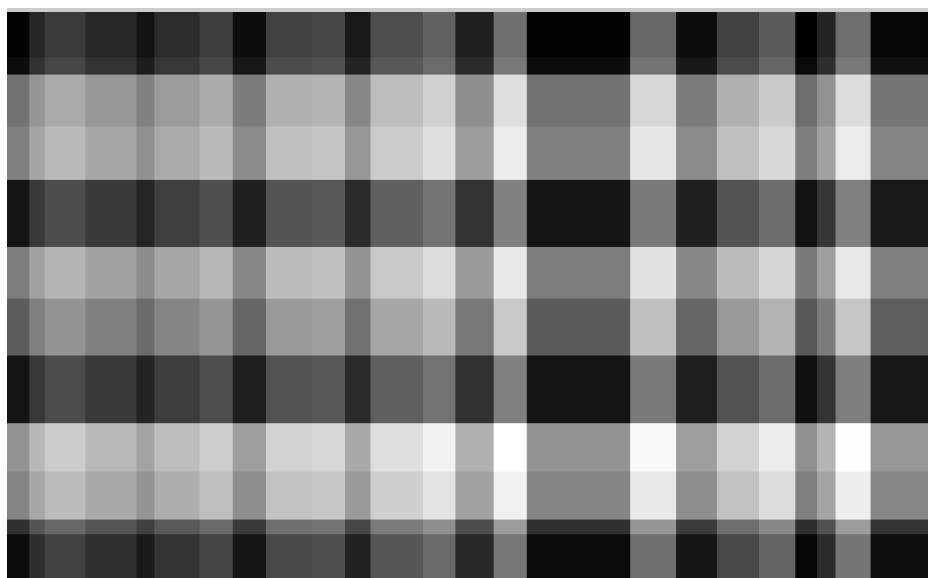


Figure 8: Window classification probabilities, white means high.

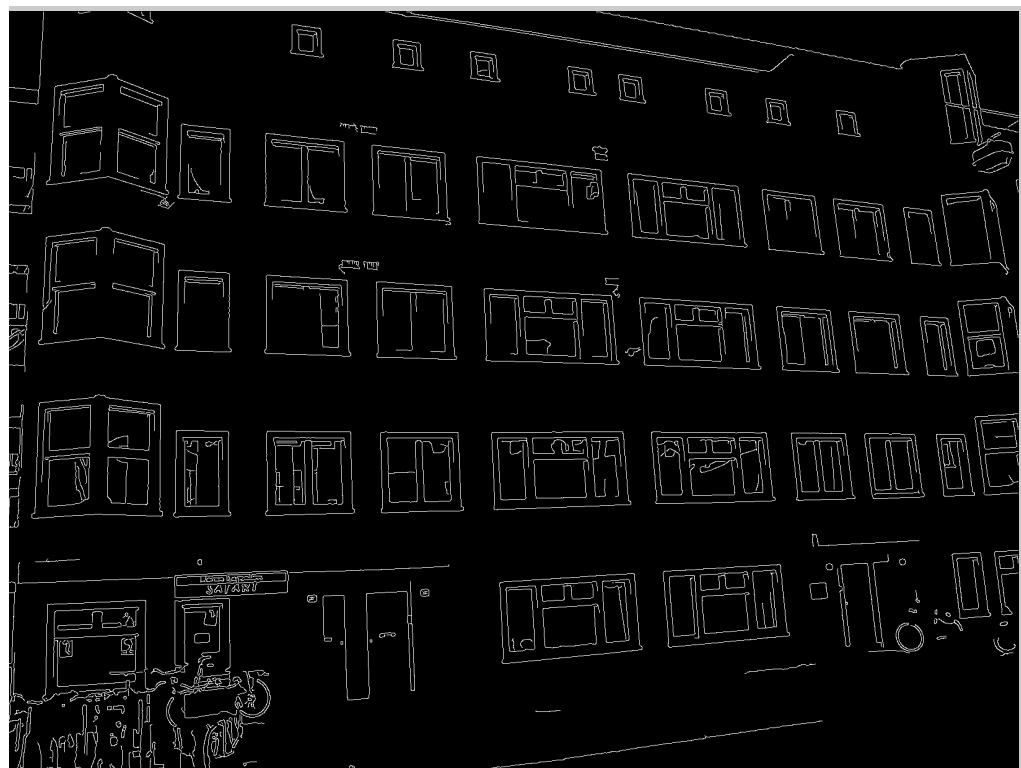


Figure 9: Edge detection

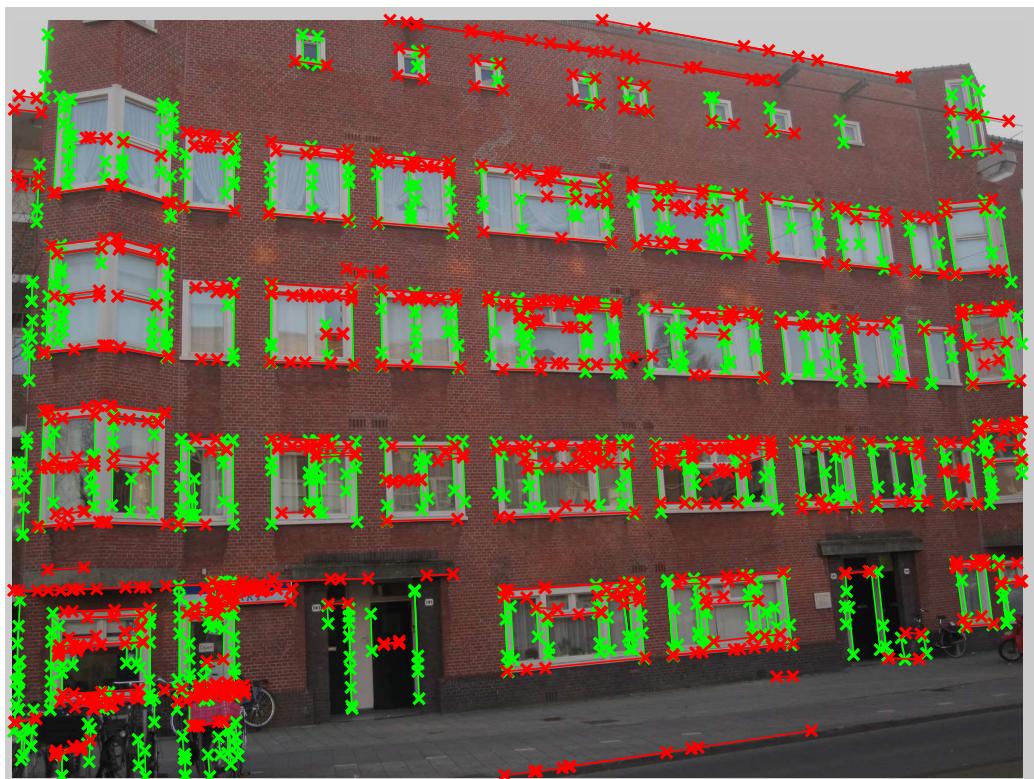


Figure 10: Result of  $\theta$  constrained Hough transform



Figure 11: Found connected corners

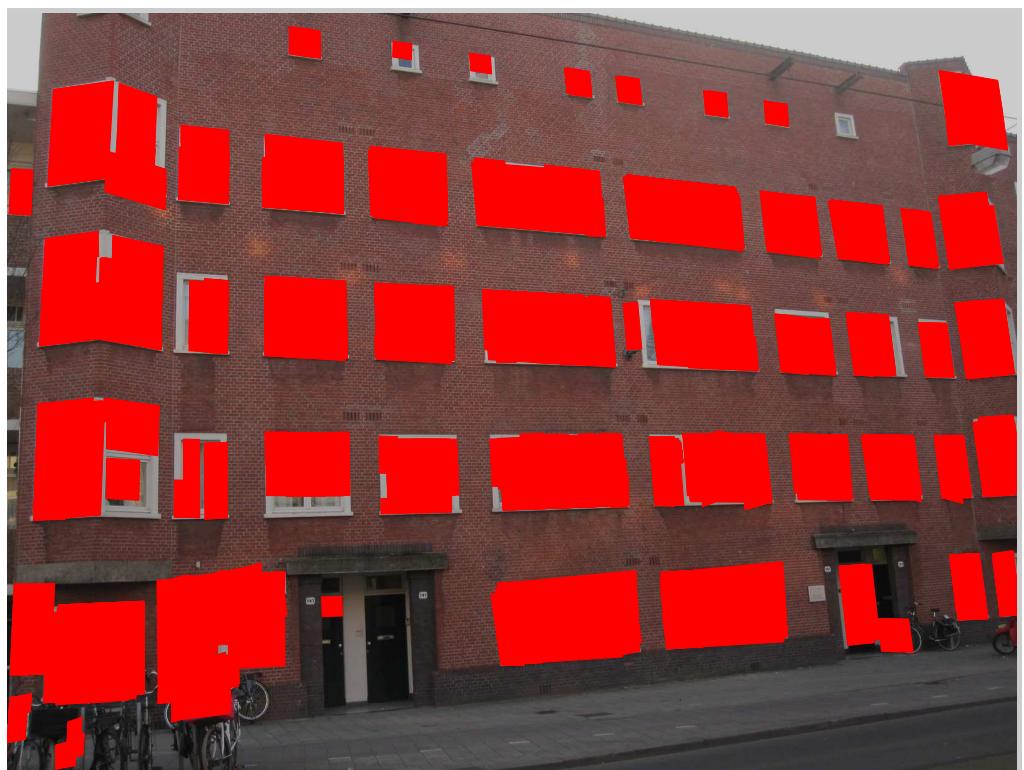


Figure 12: Window regions



Figure 13: Edge detection (with occluding tree)

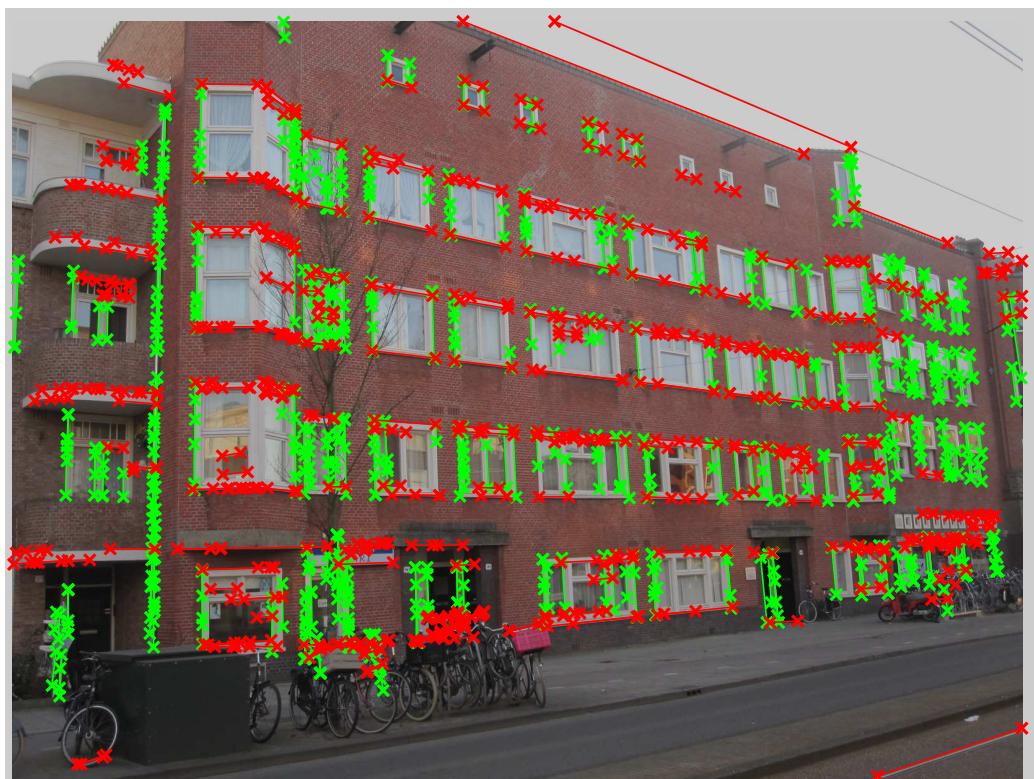


Figure 14: Result of  $\theta$  constrained Hough transform (with occluding tree)

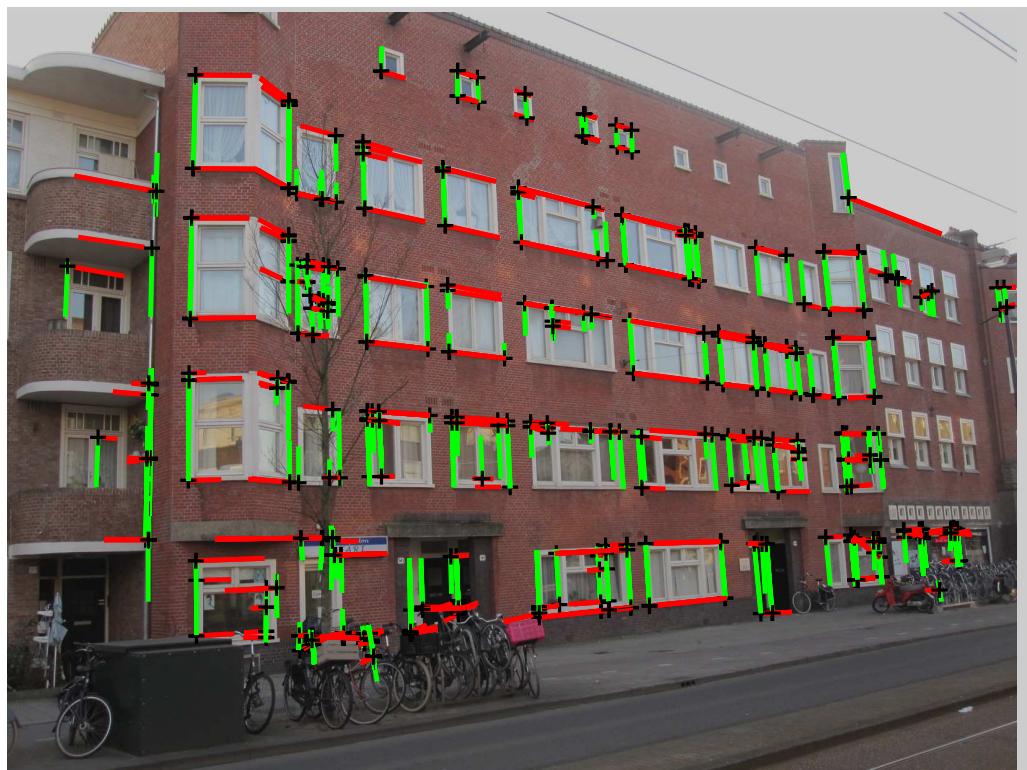


Figure 15: Found connected corners (with occluding tree)

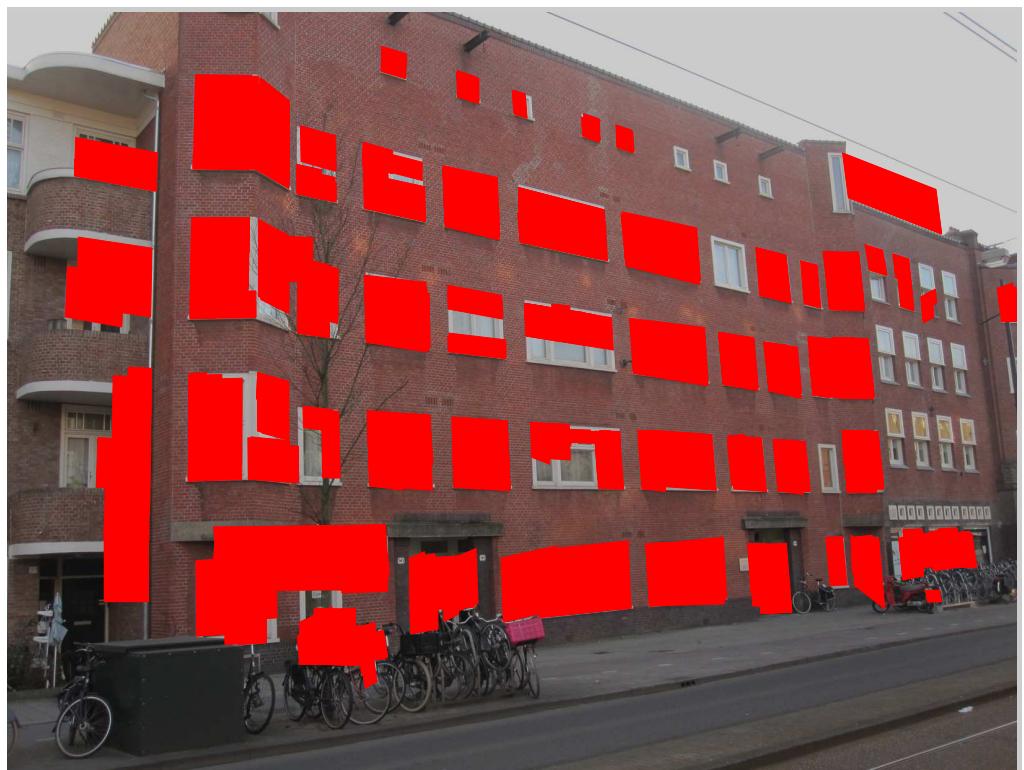


Figure 16: Window regions (with occluding tree)

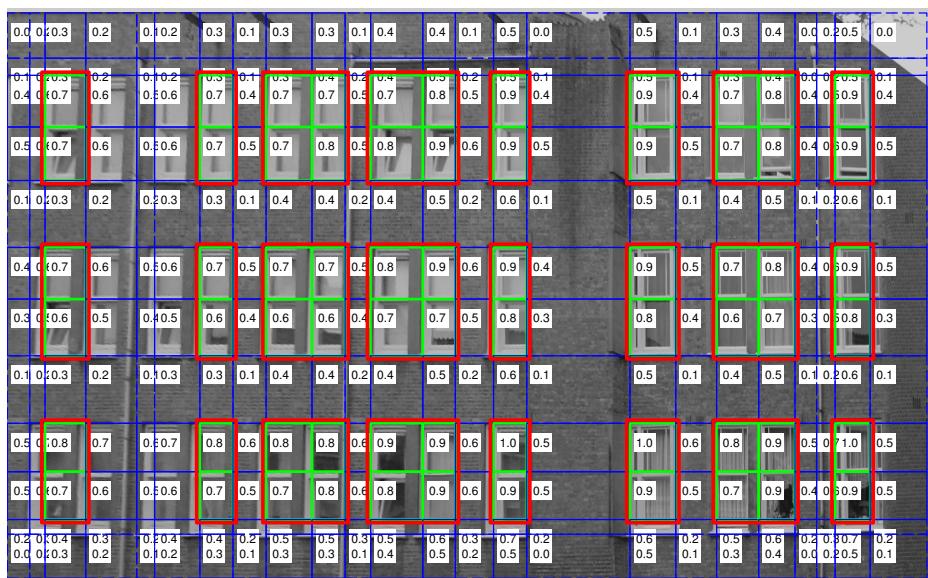


Figure 17: Classified rectangles



Figure 18: Original Image



Figure 19: Window alignment lines and histograms



Figure 20: Classified rectangles

## References

- [1] H. Ali, C. Seifert, N. Jindal, L. Paletta, and G. Paar. Window detection in facades. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 837–842, 2007.
- [2] Sung Chun Lee and Ram Nevatia. Extraction and integration of window in a 3d building model from ground view images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:113–120, 2004.
- [3] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3), July 2007.
- [4] Shi Pu and George Vosselman. Refining building facade models with images.
- [5] Michal Recky and Franz Leberl. Windows detection using k-means in cie-lab color space. In *Proceedings of the 2010 20th International Conference on Pattern Recognition, ICPR '10*, pages 356–359, Washington, DC, USA, 2010. IEEE Computer Society.
- [6] B. Sirmacek, L. Hoegner, and Stilla. Detection of windows and doors from thermal images by grouping geometrical features. In *Proc. Joint Urban Remote Sensing Event (JURSE)*, pages 133–136, 2011.