Semantic notation and 3D reconstruction of urban scenes

T. Kostelijk
mailtjerk@gmail.com

April 20, 2012

Contents

1 Window detection

1.1 Updated/New since 28-3-2012

Meeste feedback verwerkt
Explanation different types of connected corners
Future research - Window alignment refinement
Discussion - nummeric evaluation of connected corner result
Ik werk niet meer met houghline eindpunten maar met de gehele lijn.
Ivm onduidelijkheid rijen en pixelrijen gebruik ik nu ipv rectangular areas, en rows, : blocks en blockrows
Facade rectification
Improved window alignment lines
Window classification, method II
results
Future research
Results

1.2 Q

Ik heb mijn oude methode van window alignment line detectie en classificatie laten staan. Is dit verstandig?

1.3 Introduction

This chapter deals with 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. Therefore 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.

Figure 1: Simulation environment

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 chapter presents two developed methods for robust window detection.
We start with discussing related work and putting our work in context. Then we describe 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.

1.4 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 approach in detail. After this, we briefly discuss the research that is done on window detection using other approaches.

1.4.1 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] Recky 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 which is very similar to our approach,.
The next step, labeling the areas containing windows, is however very dif-

Figure 2: Projection profiles of Lee and Nevatias work

ferent 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. Furthermore we use an advanced window alignment process where we take both horizontal and vertical projection profiles into account. Our window classification is based on a higher level of shape interpretation of these functions.
As in the work of Recky 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. More on this in Future research.

1.4.2 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.

(a) original texture (b) reconstruction

Figure 3: Results of Muller et all

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.

Figure 4: Original image

1.5 Method I: Connected corner approach

1.5.1 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 viewing direction is not required to be frontal. 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.

Figure 5: Rectified image

Figure 6: Result edge detection

1.5.2 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

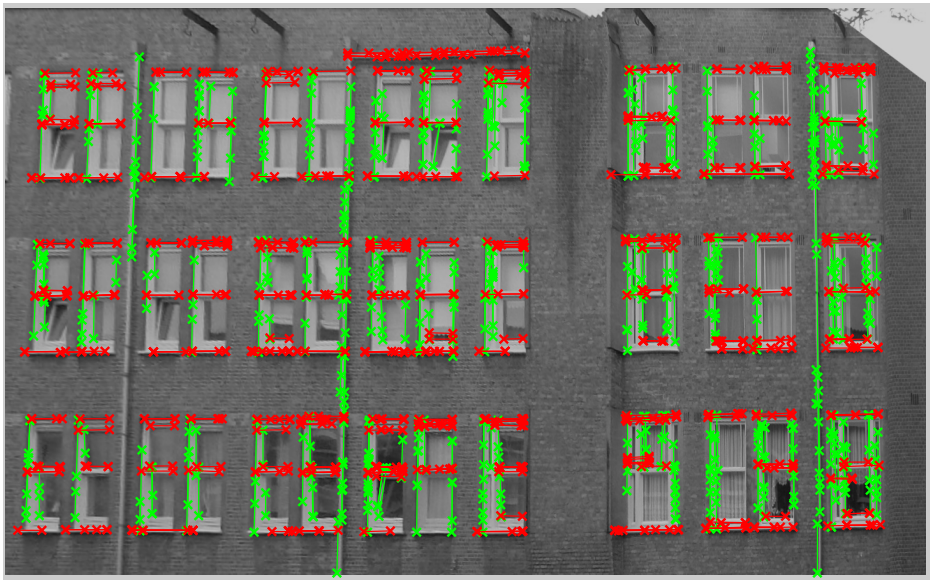

Figure 7: Houghlines with endpoints

vertical. We set the θ 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. These ranges seem to work well on an empirical basis for all datasets. The results of two images can be seen in Figure 6 and 7.

Extract connected corners

As windows contain complex structures the amount of horizontal and vertical Houghlines is large at these locations. A horizontal and vertical line is often connected in a corner of a window. In this approach we pair up these horizontal and vertical lines to determine *connected corners* that indicate a window.
Often a connected corner contains a small gap or an extension which we tolerate, these cases are illustrated in Figure 8 in the top row. A horizontal gap a vertical and horizontal gap and a vertical elongation. The cleaned up corners are given in the bottom row. When the horizontal and vertical lines intersect, the gap distance is $D = 0$. When the lines do not intersect, the distance between the intersection point and the endpoint of the lines is measured, this is illustrated as dotted lines in Figure 8. Next, D is compared

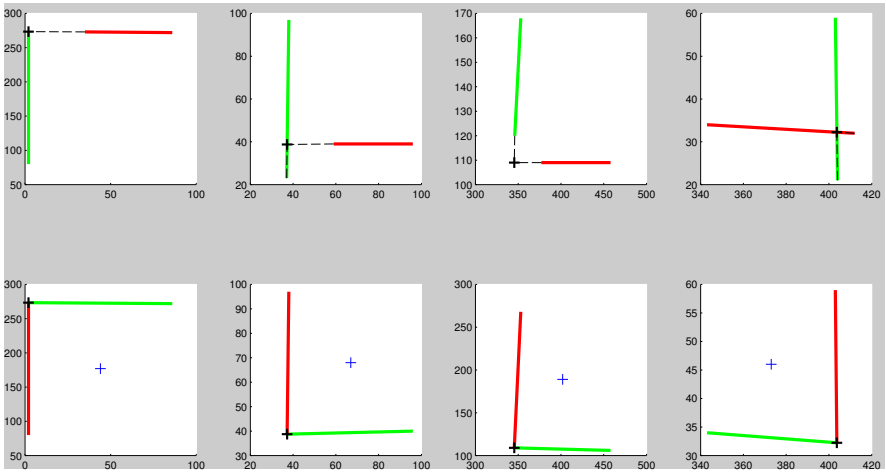

Figure 8: First row: different type of connected corner candidates. Second row: the result the clean connected corner

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 extended or trimmed, depending on the situation. The results are shown in the second row in Figure 8. In Figure 8(I) the horizontal line is extended. Figure 8(II) shows that the vertical line is trimmed. In Figure 8(III) both lines are extended. At last, Figure 8(IV) shows how both lines are trimmed.

Extract window areas

To retrieve the actual windows, each connected corner is mirrored along its diagonal. The connected corner now contains four sides which form a quadrangle window area. All quadrangles are filled and displayed in Figure 15, this result is discussed in section 1.8.

1.6 Facade rectification

1.6.1 Introduction

In order to apply our second method of window detection we need the windows on the facade to be orthogonal and aligned. Therefore we rectify the facade, which can be achieved in a simple or complex way.

The simple rectification method uses point to point correspondences. This requires annotation of the corner points of the facade that are mapped with the corners of a rectangle. This mapping is used to calculate a transformation matrix. The downside of this method is that it isn't very accurate.
Isaac/Frans, do you know how I can explain why?

The second method involves the extraction of a 3D plane of the facade. This method is more complex and gives more accurate results. It involves a comprehensive process and lots of research is done in this area. Given the related work, the automity of this module and our focus on the annotational part we used existing software to apply the window rectification. We used I. Estebans *FIT3D toolbox*, which extracts a 3D model from a series of frames. One of the planes of the 3D model was used to rectify the facade, making it ready for robust window detection.
Using the *FIT3D toolbox* we calculated the motion between a series of frames in order to extract a point cloud of matching features. This point cloud is used to extract a plane. which is used to rectify the facade by a projection process. We now explain the steps in more detail.

1.6.2 Method

We started by taking a series 6 consecutive consequent (steady zoom, lightning, etc. parameters) images of a scene. The images are chronological and have sufficient overlap. We calculated the motion of the camera between the frames in a few steps First we extract about 25k SIFT features of each frame. Then we use SIFT descriptors to describe and match the features within the consecutive frames. Not all features will overlap or match in the frames therefor RANSAC is used to robustly remove the outliers. After this an 8-point algorithm together with a voting mechanism is used to extract the camera motion.
The frames where matched one by one which returns an estimation of the camera motion that is not accurate enough. Therefore a 3-frame match is done which gives more accurate results. Unfortunately this result comes with a certain amount of reprojection error, this error is minimized using a numerical iterative method called *bundle adjustment*. The final result is a very accurate estimation of the camera motion.
The next step is to use this camera motion to obtain a set of 3D points (corresponding to the matching image features). This is achieved using linear triangulation method.
Next a RANSAC based plane fitter is used to accurately fit a plane through

the 3D points. *todo example figure*

Efficient Projecting

Now we extracted the 3D plane of the facade, the next challenge is to use this in order to rectify the facade.

It would be straight forward to rectify the full 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 necessary data. Since we are using Houghlines we project only the endpoints of the found Houghlines. (This is allowed because the projective transformation we apply preserves the straightness of the lines. Note that this means we apply the edge detection and Houghline extraction on the unrectified image.)

If h is the number of Houghlines, the number of projections is $2h$ When we rectify the full image the number of projection is w x h, where w,h are the width and height of the image. To give an indication, for the *Spil* dataset this means we apply 600 projections in stead of 1572864: a factor of almost 3k faster.

The Houghline endpoints are projected to the 3D plane we extracted in the same way as we explained in chapter **??**. To wrap up, we send rays from the camera center trough the houghline endpoints and calculated the intersection with the 3D plane. The result is a 3D point cloud where each point is labeled to there corresponding Houghline.

The next step is to transform the facade (and therefore the Houghlines) are seen upfront. Instead of transforming the facade we rotate and translate the camera. This means the viewing direction (z-axis) of the camera needs to be equivalent to the normal of the facade plane. Lets denote the unit vector spanning the original viewing direction as z and the unit vector spanning the desired viewing direction/normal of the facade z'. We calculated a rotation matrix from the axis-angle representation.

This presentation uses a unit vector u indicating the direction of a directed axis, and an angle describing the magnitude of the rotation about this axis. This axis is orthogonal to z and z' and can therefore be calculated by $u = cross(z, z')$ where cross defines the cross product of two vectors. The magnitude of the rotation α is equivalent to the angle z between z'

given u. E.g. if the images are taken almost upfront (the facade is almost rectified) α is low. α and u are used to create a rotation matrix R which is applied to the 3D point cloud. The result is a set of rectified 3D points that are grouped tot their Houghlines.

1.6.3 Results

For the purpose of representation we also calculated a full rectification of the image, see Figure 5.

1.7 Method II: Histogram based approach

1.7.1 Introduction

From the previous section we saw 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 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 alignment of the windows and then we label and group the areas that contain the windows.

1.7.2 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.

1.7.3 Method

The extraction of the windows is done in different steps. First we rectify the image making the assumptions are valid. The rectification process is done as described in the previous section.
Then the alignment of the windows is determined, this is based on a histogram of the Houghlines'. 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.

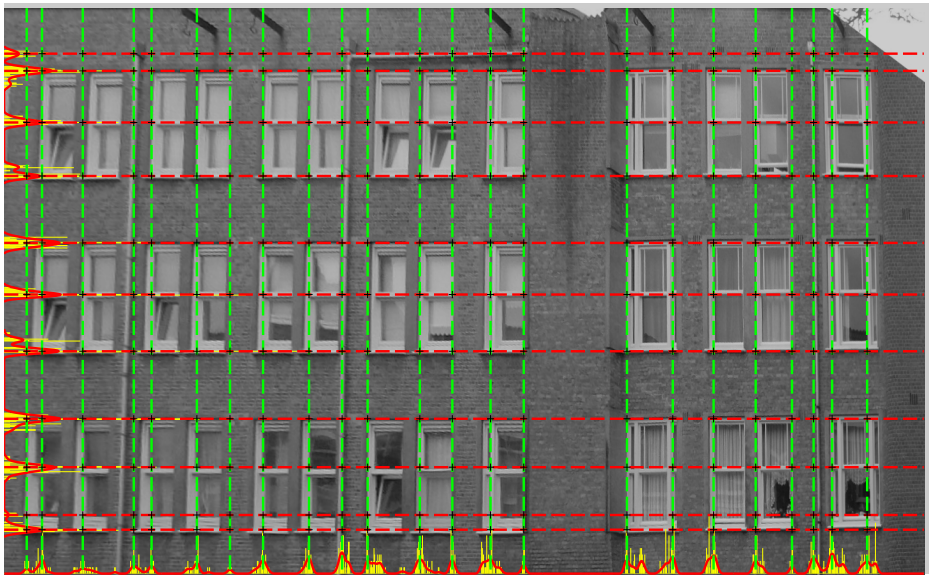

Figure 9: (smoothed) Histograms and window alignment lines

Extract Window alignment

We introduce the concept alignment line. We define this as a horizontal or vertical line that aligns multiple windows. In Figure 9 we show the alignment lines as two groups, horizontal (red) and vertical (green) alignment lines. The combination of both groups give a grid of blocks 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 are present, see Figure 7. For the vertical alignment lines the number of vertical Houghline is high, see green lines in Figure 7.

We begin by extracting the pixel coordinates of Hough transformed line segments. We store them in two groups, horizontal and vertical.We discard the dimension that is least informative by projecting the coordinates to the axis that is orthogonal to its group. This means that for each horizontal Houghline the coordinates on the line are projected to the X axis and for each vertical Houghline the 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(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 9.
The peaks are located at the positions where an increased number of Hough-lines 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 then the desired number of alignment lines. Therefore we smooth the values using a moving average filter. The result, red lines in Figure 9 , is a smooth *projection profile* 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 thresholding the function. To make the threshold invariant to the values, we set the threshold to $0.5 \cdot$ max Peak. (This value works for most datasets but is a parameter that can be changed). Next we create a binary list of peaks P, P 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 segmented 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. Therefore 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 9.

1.7.4 Improved window alignment

As you can see in Figure 20 a few window alignment lines are not found and a few lines are found at wrong locations.
Lets explain the missing alignment lines. The right side of the window frame of the first 4 windows is occluded by the wall. This is because the original image is not a frontal image. The reflection of the window has about the same color as the bricks of the wall, this means that the edge detector doesn't

find a strong edge on positions where the window frame is missing. This artefact can be seen in the edge image Figure 6 few or no edges are present, and at the low height of the peaks in Figure 20 at these positions.
This means we have to find another way to detect the window alignment on these positions. For the vertical alignment we only took vertical lines into account.
In this method we also examine the projection profile of the *horizontal* Houghlines projected on the X axis, X_h, Figure 21. On the positions of the desired vertical alignment lines there appears to be a big decrease or increase of X_h at the window frame. This is because on these positions a window containing (a large amount of horizontal lines) starts or end.
We detect these big decreases or increases by creating a new pseudo peak profile D that takes the absolute of the derivative of X_h, , Figure 21.

$$D = abs(X_h')$$

We apply the same peak location detection as the previous method but combine the results. Sometimes the same window alignment line is found by both methods, Figure 22. We merge the peaks by discarding peaks that are close to each other, the result can be seen in Figure 23.
The image is now divided in a new grid of blocks based on these alignment lines. The next challenge is to classify the blocks as window and non-window areas: the window classification. We developed two different methods.

Window classification, method I

Instead of classifying each block independently, we classify full rows and columns of blocks as window or non-window areas. This approach results in more accurate classification as it combines a full blockrow and blockcolumn as evidence for a singular window.
The method exploits the fact that the windows are assumed to be aligned. A blockrow that contains windows will have a high amount of vertical Houghlines, Figure 7 (green). For the blockcolumns the number of horizontal Houghlines (red) is high at window areas. We use this property to classify the blockrows/blockcolumns.
For each blockrow the overlap of all vertical Houghlines 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 blockrow influences the outcome,

this total value is normalized by the size of the blockrow.

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

Leaving us with $||R||$ (number of blockrows) scalar values that give a rank of a blockrow begin a window area or not. This is also done for each blockcolumn (using the normalized horizontal amount of Houghlines pixels) which leaves us with C.

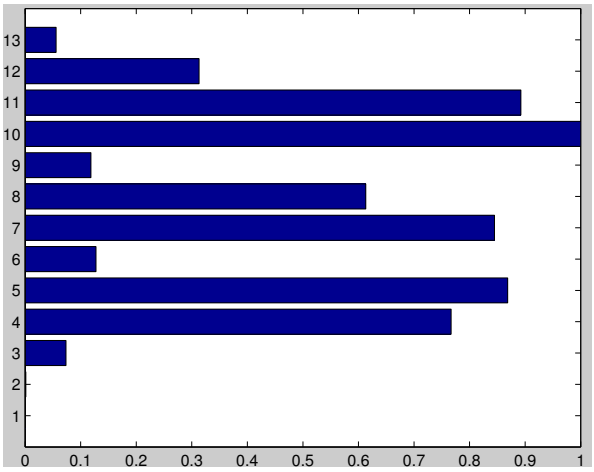

Figure 10: Normalized vertical Houghline pixel count of the blockrows (R)

If we examine the distribution of R and C, we see two clusters appear: one with high values (the blockrows/blockcolumns that contain windows) and one with low values (non window blockrows/blockcolumns). For a specific example we displayed the values of R in Figure 10. Its easy to see that the high values, blockrow 4,5,7,8,10 and 11, correspond to the six window blockrows in Figure 16.
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 blockrow 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 block $w \in W$ 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 16 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 blocks. These are displayed as red rectangles in Figure 16.
To get insight about the probabilities that lie behind the individual blockrows and blockcolumns, we designed another representation in Figure 11 The whiter the area the more probable a block is classified as a window.

Window classification, method II

If we take a look at Figure 23 we see that X_h (the amount of horizontal Houghlines) has two shapes that repeat: At the location where a window is present X_h is concave whereas at non-window areas X_h is convex. This is because lots of horizontal lines are found at certain window positions (the window centers) and few are found at positions that are far away from these certain positions, which are the centers of non-window areas (that lie between windows). The shape type of X_h is used as a cue for our second window classifier.

This shape type of X_h is detected as follows: As in **improved window alignment**, the first step is to examine the derivative of X_h, blue line in Figure 23. We investigate the positions where $D = X_h'$ changes from sign, these are the peaks or valleys of X_h. X_h is concave at the sign changes from positive to negative (+,-) and X_h is convex if the sign changes (-,+).

We expect one sign change per block, however it is possible that multiple sign changes occur. In this case we smooth X_h again and repeat the algorithm until for each block a maximum of one sign change is found.

Now we have detected the shape type (concave or convex), we can directly classify the blockrows and blockcolumns as window areas and non-window areas. The windows are determined as the previous classifier by combining the (positively classified) blockrows and blockcolumns.

For the sake of representation only blockcolumns are presented, the method

for the blockrows is almost the same, the projection profiles are projected to the Y-axis.

1.8 Results

. We tested all methods on different datasets.

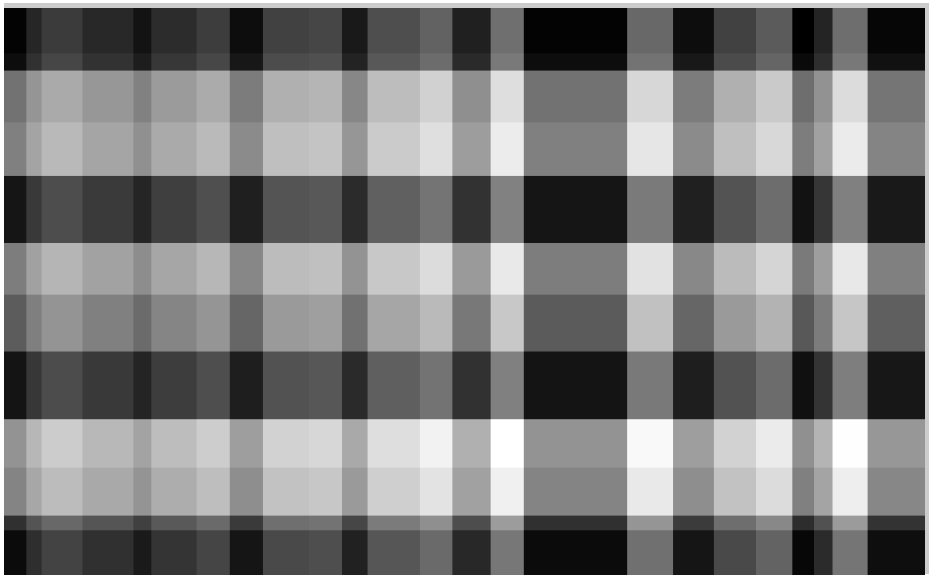

Figure 11: Window classification probabilities, white means high.

todo include images other datasets
todo compare methods and explain differences

1.9 Discussion

todo discuss results

Method I: Connected corner approach

Figure 15 contains 110 windows of which are 109 detected, this is 99%. Furthermore there are some False Positive areas, this is about 3 %. The window on the right top isn't detected, this is because he is smaller then our minimum window width.
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

We described a basic method of window alignment and build a significant improvement. If we compare the basic method (Figure 16) with the improved

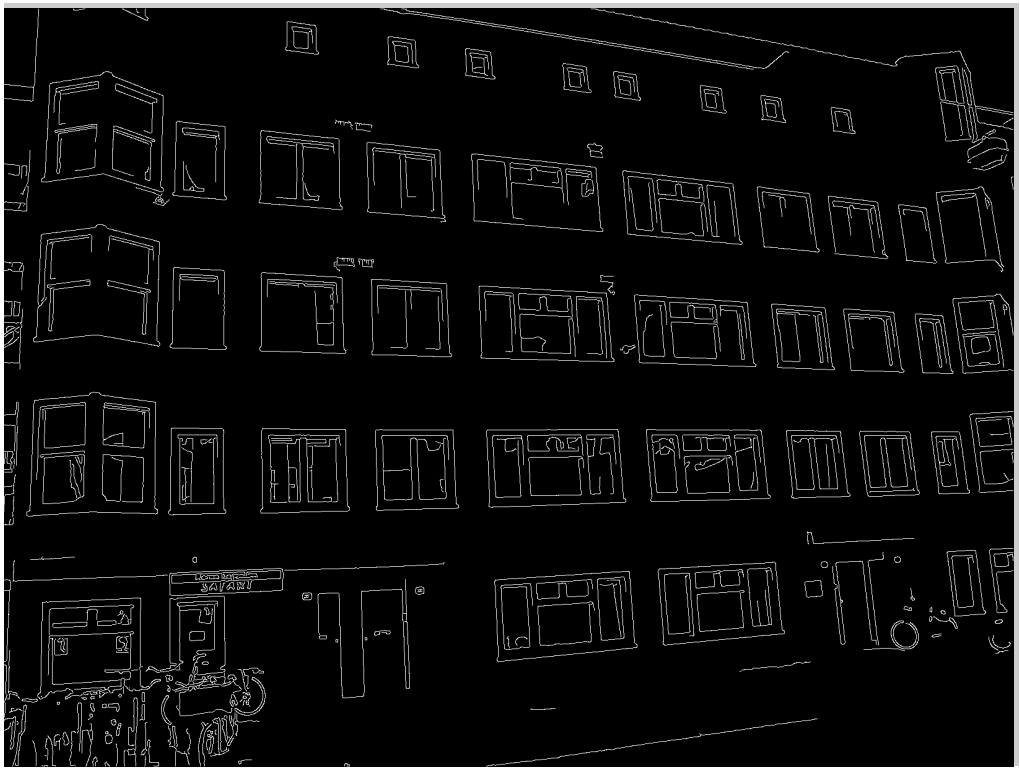

Figure 12: Edge detection

method (Figure 22) we see that the improved method detected each window alignment with an maximum error of 3px whereas the basic method has a 15px error. Its hard to say something about the basic classification method because the window alignment lines had to large error. The improved classification method Figure 25, classified 100 percent of the windows correct.
It could be a drawback that the outcome of method I is non-deterministic, as it depends on to the random initialization of the cluster centers. Our results could be correct by coincidence. To exclude this artefact, we ran the cluster algorithm 10 times, fortunately it resulted in the same classification.
todo

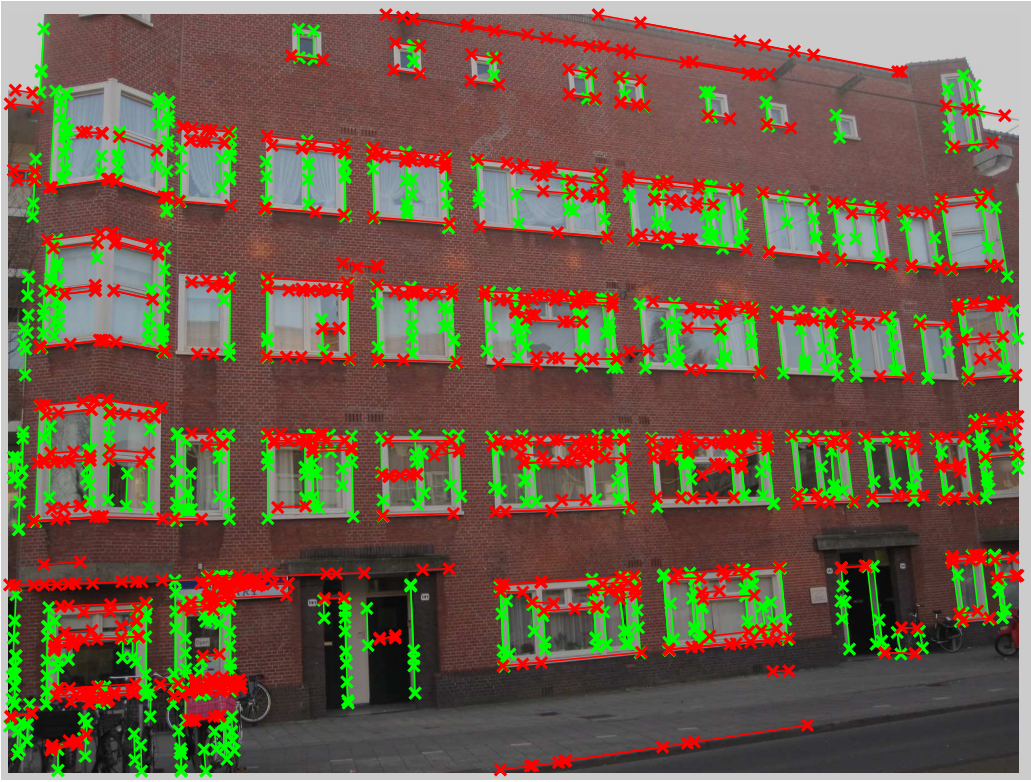

Figure 13: Result of θ constrained Hough transform

Occlusion

If the image isn't the frontal view of the building wall we project the image see section ?This projection comes with some difficulties, occlusion. In a few cases an building wall extension (middle of figure 4) 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 cleanness of the projection. However, this occlusion artefact is in most cases no problem as the system combines the windows probabilities.

1.10 Conclusion

todo

Figure 14: Found connected corners

1.11 Future research

1.11.1 Method I: Connected corner approach

It would be nice to group the connected corner to groups of sub windows. The big window that contains sub windows could be found by calculating the convex hull of the red areas in Figure 15. The sub windows could be found using a clustering algorithm that groups the 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 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 of the sub window.

Figure 15: Window regions

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.

1.11.2 Method II: Histogram based approach

It would be nice to investigate the effect of the occlusion and to examine 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.

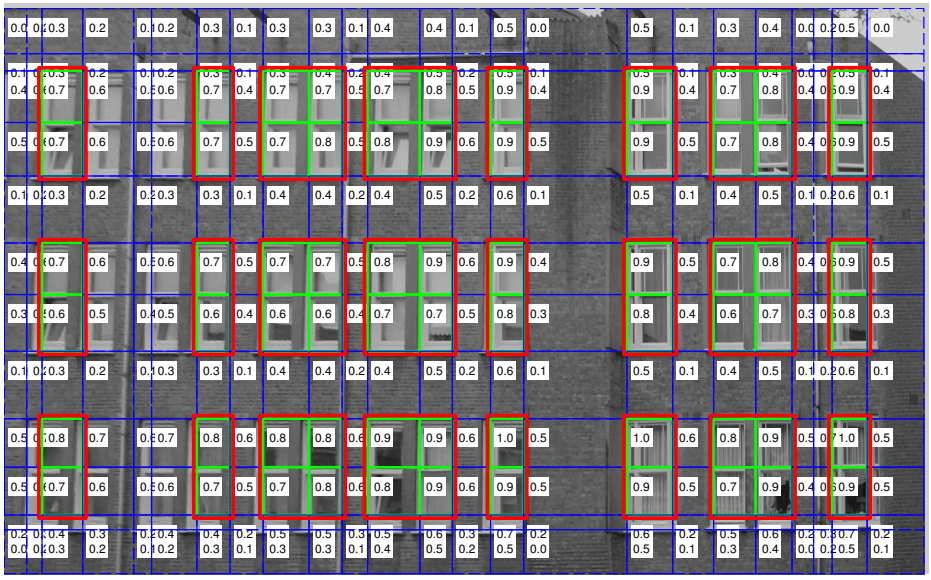

Figure 16: Classified windows

Window alignment: peak merging

We merged the result of the horizontal and vertical Houghlines and discarded peaks that where close. A more accurate result would be achieved if close peaks where averaged. The challenge is to determine if both methods are talking about the same peak. And how to handle multiple close peaks, e.g. if 4 peaks are close then peak 1 could be different then peak 4. Assuming some knowledge, like maximum and minimum window size, would help to solve this problem.

Window alignment refinement

To get more accurate result or to handle scenes with poor window alignment a refinement procedure could be applied. As mentioned in the related work, Lee et all [2] applied window refinement. Although this comes with accurate results, the iterative refinement is a computational expensive procedure. It would be nice to have a dynamic system that is aware of this accuracy and computational time trade of. A system that only refines the results when the recourses are available. For example if a car is driving and uses window detection for building recognition the refinement is disabled. But if the car is lowering speed the refinement procedure could be activated. Resulting in

Figure 17: Original Image

accurate building recognition which opens the door for augmented reality.

Feature fusion

Both window refinement and window alignment steps could use some additional evidence which could be provided by feature based methods. For example a *multi scale Harris corner detector* could help an accurate alignment or refinement of the windows.

Figure 18: Window alignment lines and histograms

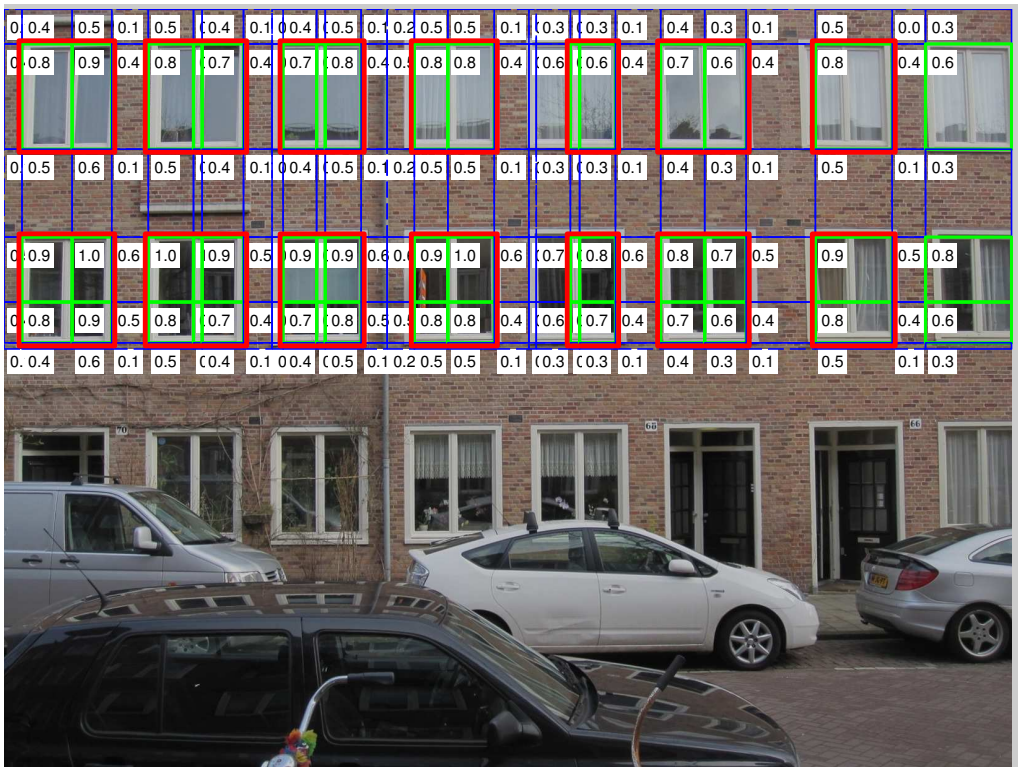

Figure 19: Classified rectangles

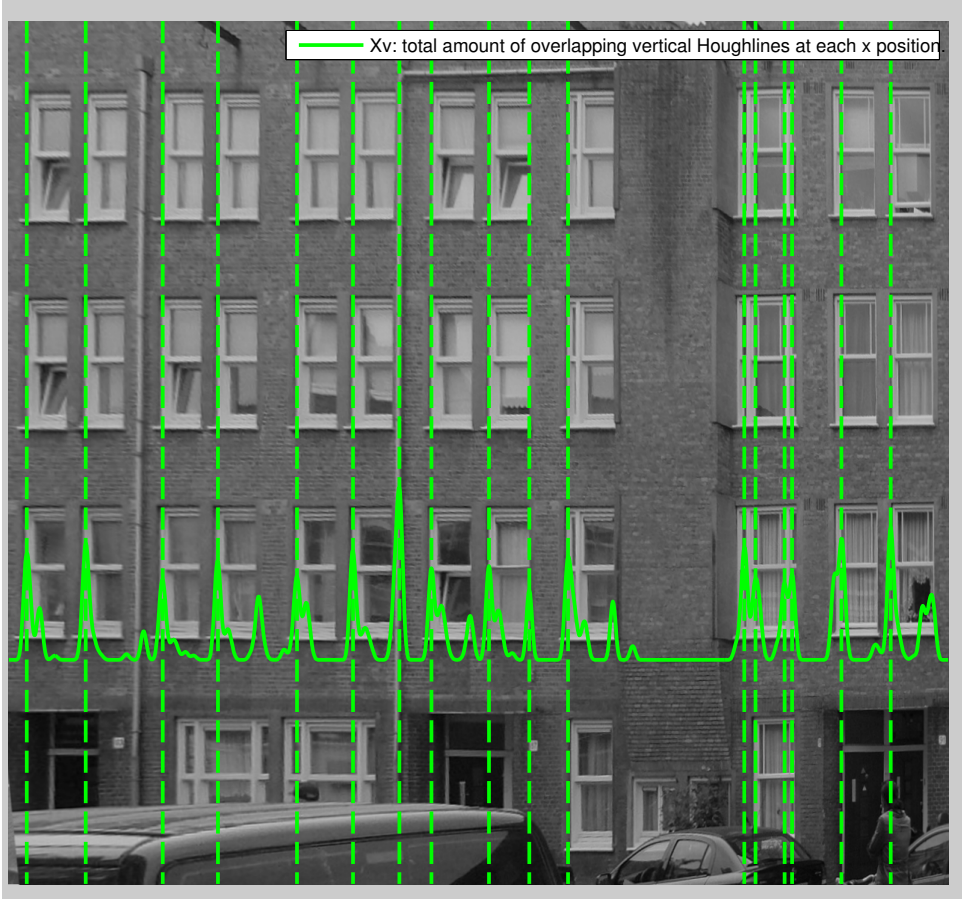

Figure 20: Improved window alignment: Xh: total amount of overlapping horizontal Houghlines at each x position

Figure 21: Improved window alignment: Using the shape of the horizontal Houghlines amount as window alignment

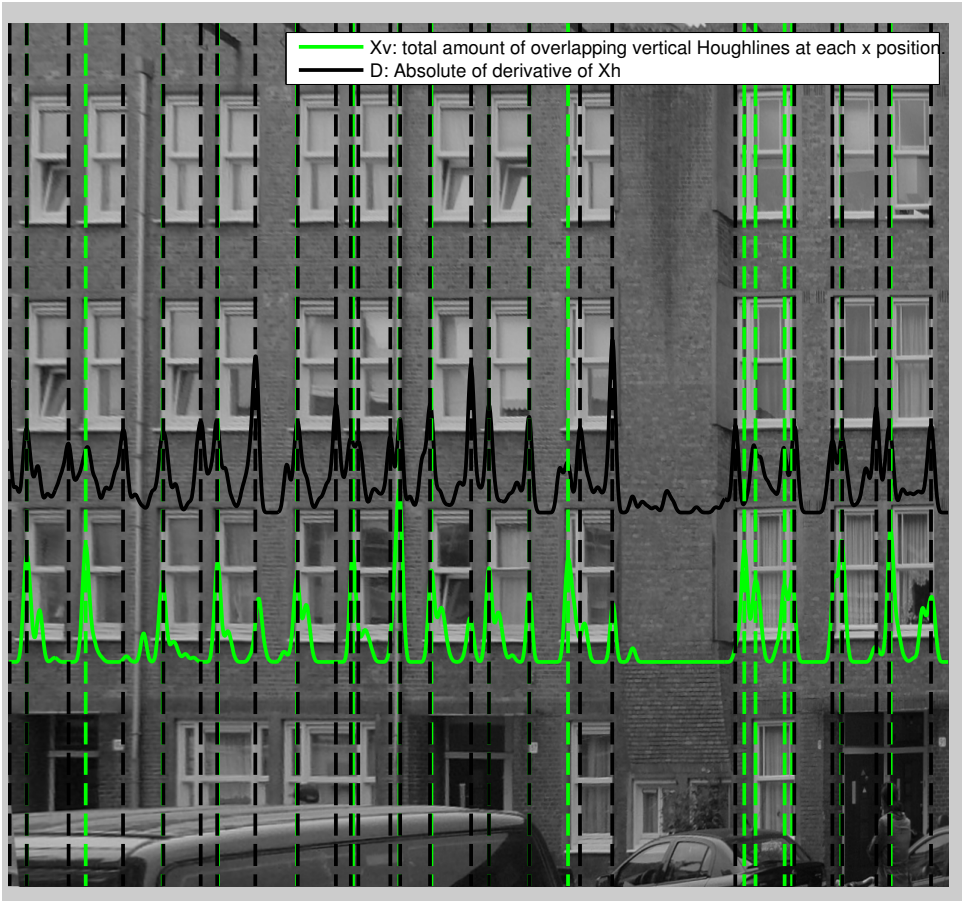

Figure 22: Improved window alignment: Both methods combined

Figure 23: Window classification method II:Concave and convex shapes on window/nonwindow locations

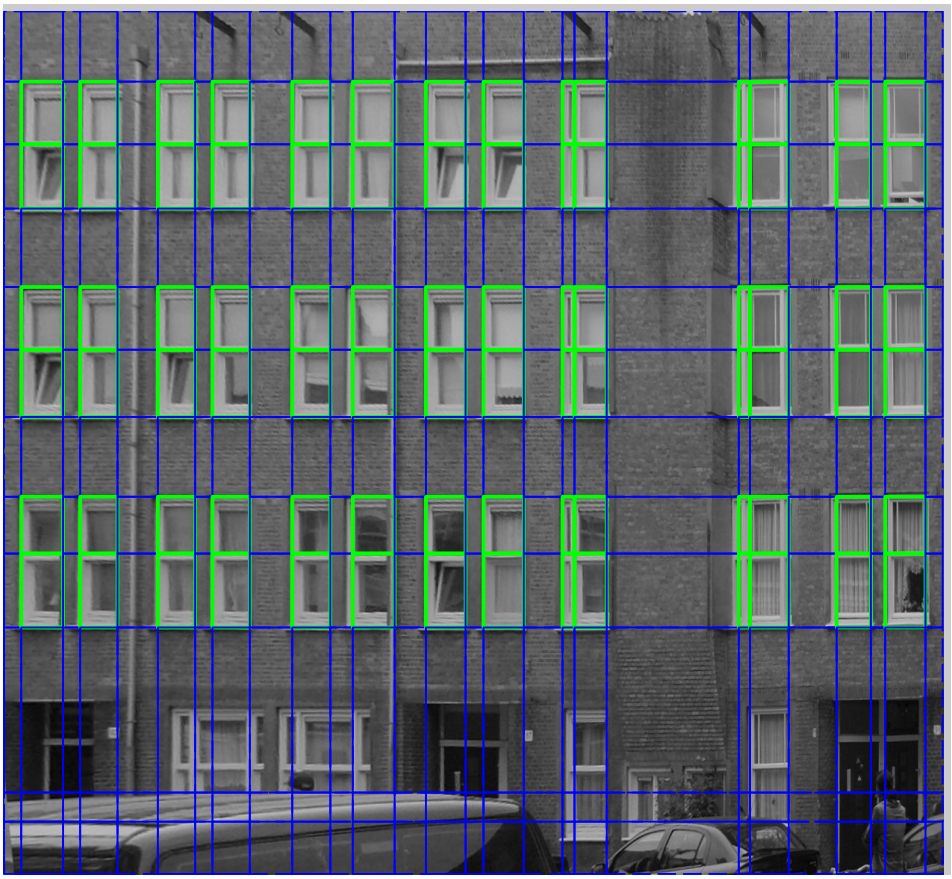

Figure 24: Window classification method II:extracted windows

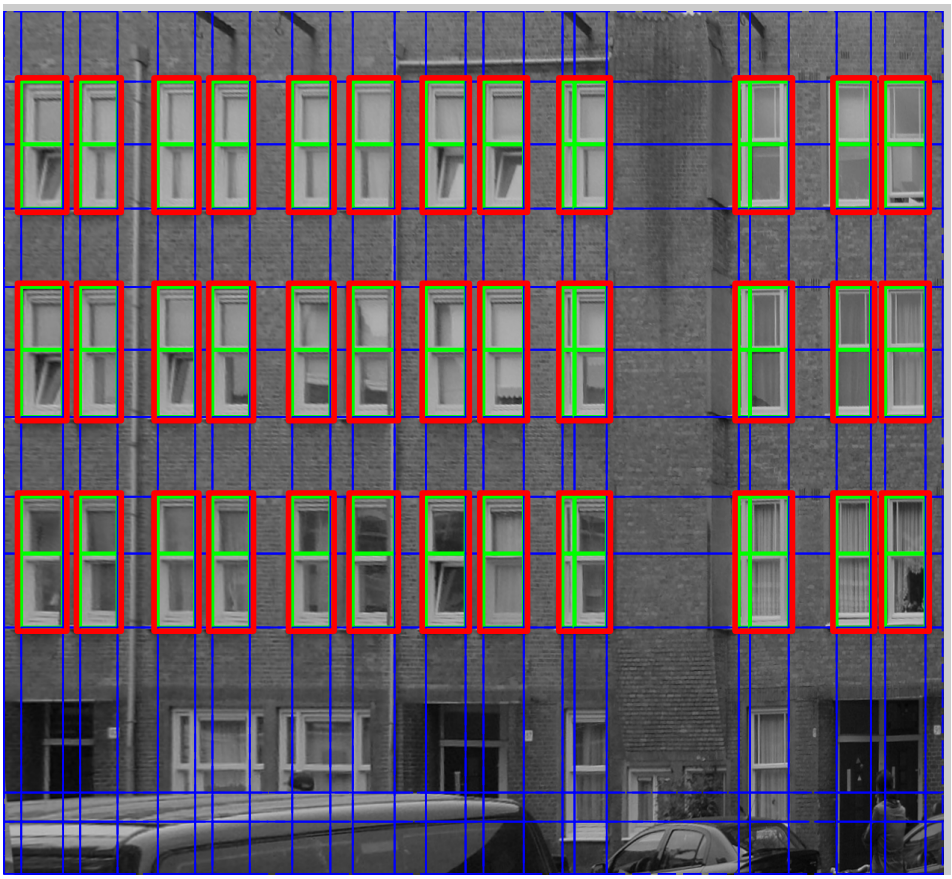

Figure 25: Window classification method II:groupde windows

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.