

3D RECONSTRUCTION AND SEMANTIC ANNOTATION OF URBAN SCENES

T. Kosteljik
mailtjerk@gmail.com

May 25, 2012

Contents

1 comments/questions	5
1.1 questions	5
2 Window detection	6
2.1 Introduction	6
2.2 Application examples	6
3d City models	6
Historical buildings documentation and deformation analysis	6
Interactive 3d models	7
Augmented reality	7
Building recognition and urban planning	8
2.3 State of art window detection	8
2.3.1 Alternative approaches on window detection	8
2.3.2 Similar approaches	10
2.3.3 Our work in context	12
2.4 Method I: Connected corner approach	12
2.4.1 Situation and assumptions	12
2.4.2 Edge detection and Houghline extraction	12
2.4.3 Connected corners extraction	13
2.4.4 Window area extraction	14
2.4.5 Results	15
2.4.6 Conclusion	18
2.4.7 Future research	18
2.5 Facade rectification	19

2.5.1	Introduction	19
2.5.2	3D plane extraction	19
2.5.3	Efficient Projecting	20
2.5.4	Results	22
2.6	Datasets	25
2.6.1	Anne1	25
2.6.2	Anne2	25
2.6.3	Dirk	25
2.7	Method II: Histogram based approach	26
2.7.1	Introduction	26
	Situation and assumptions	26
2.7.2	Extracting the window alignment	26
	Regular window alignment	26
	Alternative window alignment	28
	Fusing the window alignment methods	28
	Results	30
	Discussion	35
	Conclusion	36
	Future research	37
2.7.3	Basic window classification (based on line amount) . .	39
	Results	42
	Discussion	44
	Future research	44
	Conclusion	44
2.7.4	Improved window classification (based on shape of the histogram function)	45
	Results	46
	Conclusion	51
	Future research	52
2.8	Conclusion	53
3	Appendices	54
3.1	Figures	54
3.1.1	K-means bad luck	56

Todo list

TODO: Zorgen dat paragraphs ook 3.5.3.1 nummering krijgt	6
TODO: in state of art een lijn creeren, opbouwen, in context plaatsen, theoretisch afwimplene	8
TODO: research question	10
TODO: loopt niet heeeelemaal lekker	10
TODO: maak analogie met L en H patronen hoe we die herkennen in de hersenen	13
TODO: more datasets..	15
TODO: ref axis angle wikipedia	21
TODO: Met iemand overleggen wat betere volgorde is	24
TODO: image of dataset	25
TODO: ipv realistic alledaags?	25
TODO: uitbouw van de ramen	26
TODO: tabel van datasets en handicaps!	26
TODO: linken	26
TODO: peak merging result misses for 1 dataset?	29
TODO: result1 drain pipe result niet erg ivm vlg stap	36
TODO: make diagram: edge- hough -window alignment - window clas- sificatio - window grouping	37
TODO: diagram maken window alignment - ζ peak merging - ζ etc . . .	37
TODO: DIT PLAATJE eruit en mooie staafdiagrammen erin	39
TODO: ref	40
TODO: include results of binary classification	44
TODO: include table of false positives etc, detection rate etc. etc. misch staat dit nog op github	44
TODO: introduce a little bit	45
TODO: say that anne2 dataset is more rectified	45
TODO: make a diagram, histogram - alignment lines - ζ window clas- sification etc.	46
TODO: discuss drain pipe, dat het goed is dat het resultaat mooi is ondanks drainpipe	46
TODO: IMPROVED result on Anne1 dataset, investigate which combi- nations of dataset andmethods i miss	48
TODO: certainty based (white strokes) classification on anne2 dataset, pas als skyline af is, img zijn wel al klaar	51
TODO: remark that we goedkeuren partieel resultaat van windows helemaal rechts en links dirk dataset	51
TODO: Interpretation of the results of the Anne1 dataset	51

TODO: compaire to other classification methods	51
TODO: (perform both classifications on both datasets (2x3 img), visual concatinate images vertically)	51
TODO: especially the horizontal division of the lowest row of windows is amazing, refer to	51
TODO: Meer over lof spreken	51
TODO: put a opbouwende line in future work	52
TODO: speak about video, or sequence of images, detection + tracking etc	52
TODO: compare classification methods and explain differences	53
TODO: We showed that projecting the image to a frontal view is a good preprocessing step of a robust window detector.	53
TODO: Furthermore our algorithms work in real time. 53	
TODO: check if appendix images not already in ch5!!	54

1 comments/questions

1.1 questions

Ik heb er voor gekozen om de resultaten, discussie en conclusie van de window alignment individueel te bespreken
voor window classificatie methode 1 + 2 heb ik dit samengevoegd. is dit een wijze keuze?

TODO: Zorgen dat paragraphs ook 3.5.3.1 nummering krijgt

2 Window detection

2.1 Introduction

Semantic interpretation of urban scenes is used in a wide range of applications. We deal with an important aspect of semantic urban scene interpretation, Window detection.

This chapter is organised as follows, we start with a variety applications that use semantic interpretation of urban scenes. Then we discuss related work and put our work in context. Next we describe our first window detection approach that is invariant to viewing direction. After this we present a facade rectification method. Next this method is used for our second method that assumes orthogonal and aligned windows. Finally we show and discuss results.

2.2 Application examples

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 an 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 as the size and location of the windows

supply an effective descriptor that can be used for robust and fast building identification. Furthermore it provides an accurate alignment of the overlay.

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.

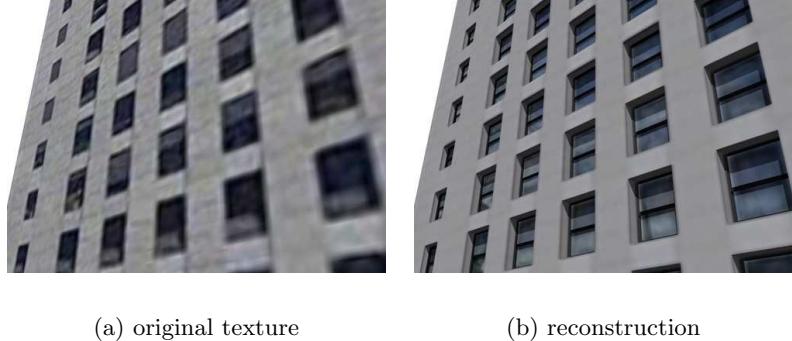
TODO: in state of art een lijn creeren, opbouwen, in context plaatsen, theoretisch afwimpen

2.3 State of art window detection

A large amount of research is done on semantical interpretation of urban scenes. First we briefly discuss the research that is done on window detection using approaches that differ from our approach. After this, we discuss state of the art window detection that has a big overlap with our approach in detail.

2.3.1 Alternative approaches on window detection

Muller et all [4] detect regularity and symmetry in the building. The symmetry is detected in the vertical (floors) and horizontal (window rows) direction. They 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. Although they achieved some interesting results their method has some disadvantages. As their method is fully based on detecting symmetry, they have to assume repeating and aligned windows: this constraints the variety



(a) original texture

(b) reconstruction

Figure 2: Results of Muller et all

of scenes the system can handle. Furthermore they match template window objects which they predefine, this constraints the variety of window types that could be matched. At last they use expensive algorithms that make it impossible for the system to run in real time.

Using a thermal camera, Sirmacek [7] 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*: they search and group intersecting L-shapes to close a window shape. A shape is determined closed if it can separate an inside region from the outside, i.e. determine if it is a window.

Ali et all [1] describe the windows with *Haar* like features, the features are combined using a cascading classifier. The cascading classifier, which acts like a decision tree, is learned using the *Ada boost* algorithm. They also use window detection to determine the region of the facade.

Although this method is used a lot in computer vision, it is not the most promising approach to window detection because it is supervised. This means that it requires a large dataset in which every window must be accurately annotated. As the cascader uses a fixed swiping window it is sensitive to scale: all window sizes must be equal and a size range must be given to the system. Furthermore the system is always overfitted to the learning data, making it hard to generalise (detect windows that are not included in the dataset). A more general descriptor of the window that is size invariant is desirable.

TODO: research question

TODO: loopt niet heeeelemaal lekker

To investigate this, we developed two methods on which one method doesn't require repeating windows nor aligned windows. One of the main targets of our research is a low requirements on the input data. First our system doesn't need a large annotated dataset. Furthermore we took the images with a mobile phone and decided to extract the windows from the image space only, this makes us independent of additional expensive data like heat or laser range images. It doesn't mean the previous work on window detection using laser or heat images isn't of good use. Instead we learned a lot from the previous research as they have to match the laser or heat data to the real image space. This matching process involves a description (semantical annotation) of the facade. Let's explain a method of that kind and discuss other approaches that are more similar to our approach.

2.3.2 Similar approaches

Pu and Vosselman [5] combine laser range images with ground images 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 extracted Hough lines of both ground and laser range images. They match the lines by comparing the angle, location and length differences. These criteria are also used in our approach.

They also detect windows and use them to provide a significant better alignment of the 3d model. As windows have a high reflection, they form hole like shapes in the laser range images. These holes are directly used to extract the windows, unfortunately the results were far from accurate.

The work of Pu and Vosselman [5] provides a useful practical application of window detection and it amplifies the need for a robust window detection technique that is independent of laser range data.

Recky et all [6] developed a window detector that is build on the primary work of Lee and Nevatia [3] (which is discussed next). In order to make clear orthogonal projections they rectify the facade. To determine the alignment

of the windows the edges are projected into their orthogonal direction. For example the horizontal edges are projected in the vertical direction to establish the vertical division of the windows.

A function is developed that counts the amount of projected edges on each



Figure 3: Projection profiles of Lee and Nevatias work

location, this is the *Projection profile*, see Figure 3. A threshold is applied on the projection profile to indicate the window boundaries that is used for the window alignment.

In the next step they use color to disambiguate the window areas from non-window areas. 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. Furthermore the classification based on color is sensitive to change in illumination conditions.

As in the work of Recky et all [6] Lee et all [3] perform orthogonal edge projection to find the window alignment. As different shapes of windows can exist in the same column/row, they only 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. As we want to run our system in realtime this method is not suitable for our application.

2.3.3 Our work in context

The state of the art window alignment procedure in [6] and [3] is very robust. Therefore we decided to use this method as a basis and improved the alignment algorithm, furthermore we build a different window classification method.

Our improvement on the alignment procedure is as follows. In the previous work [6] and [3] they use a single projection profile for each direction. We improved this process by fusing two (more advanced) projection profiles for each direction. E.g. for the determination of the horizontal division of the windows we fuse both horizontal and vertical projection profiles.

Furthermore we build two alternative window classification procedure which are based on a higher level of shape interpretation of these projection profiles. As the classification is based on the projection profiles (edge information) we don't require expensive color transformations and we only apply (rectification) transformations on line segment endpoints. This makes our algorithm invariant to change in illumination and perform in real-time.

2.4 Method I: Connected corner approach

2.4.1 Situation and assumptions

A window consists of a complex structure involving a lot of connected horizontal and vertical lines, we use this property to detect the windows. We introduce the concept *connected corner*, this is a corner that is connected to a horizontal and vertical line. The search for these connected corners is based on edge information. The connected corners give a good indication of the position of the windows,

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. As such the windows are detected individually.

2.4.2 Edge detection and Houghline extraction

Edge detection is done as it is described in section ?? . From the edge images two groups of Houghlines are extracted. The groups fall in the two window directions horizontal and vertical. This is done by controlling the allowed angles, θ bin ranges, in the Hough transform. The horizontal group has a

range of $\theta = [-30..0..30)$ degrees, where $\theta = 0$ presents a horizontal line. The vertical group has a range of $\theta = [80..90..100)$ degrees. We use these ranges because 1) the user hardly ever holds the camera exactly orthogonal. 2) we work with unrectified facades, meaning we deal with perspective distortion. To be more concrete, if the user takes a photo (Figure 4 with a certain Yaw $\bar{\theta}$, the horizontal lines become skew. The range of the vertical group is smaller than the horizontal group as the user often takes photo with a low pitch value and a high yaw.

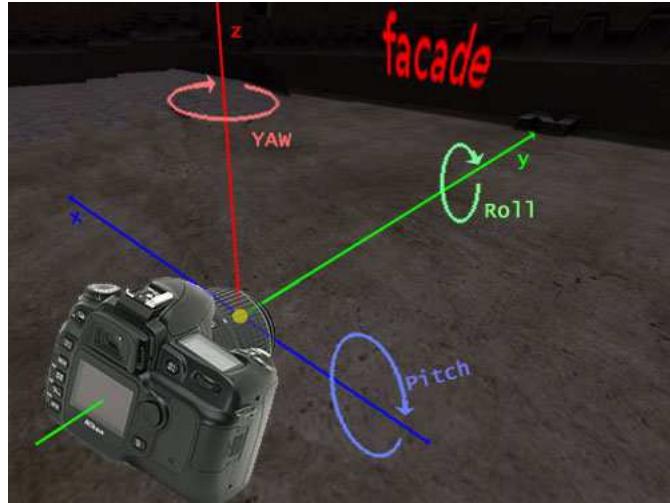


Figure 4: Pitch roll and yaw of the camera

The results of the edge detection and the Hough transform of two images can be seen in Figure 6 and 7.

TODO: maak analogie met L en H patronen hoe we die herkennen in de hersenen

2.4.3 Connected corners extraction

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 5 in the top row. A horizontal

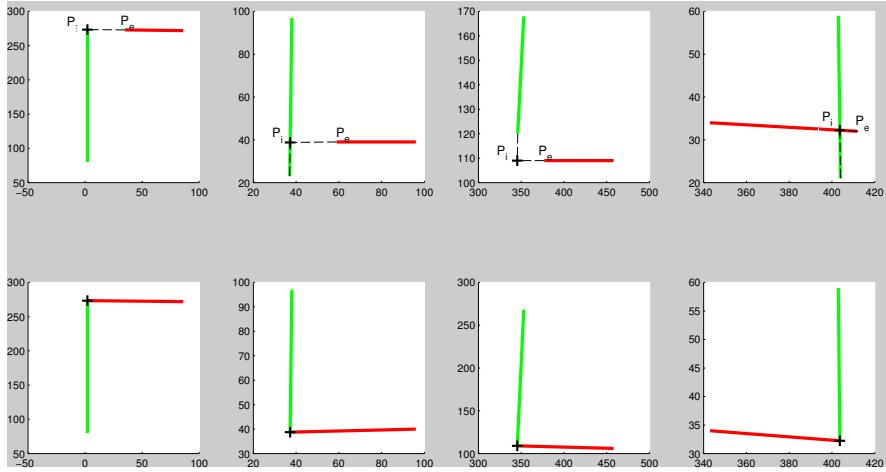


Figure 5: First row: different type of connected corner candidates. Second row: the result the clean connected corner

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 D between the intersection point P_i and the endpoint P_e of the line is measured $D = \|P_i - P_e\|$, this is illustrated as dotted lines in Figure 5. 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 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 5(I) the horizontal line is extended. Figure 5(II) shows that the vertical line is trimmed. In Figure 5(III) both lines are extended. At last, Figure 5(IV) shows how both lines are trimmed.

2.4.4 Window area extraction

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 9, this result is discussed in section 2.4.5.

2.4.5 Results

· TODO: more datasets..



Figure 6: Edge detection

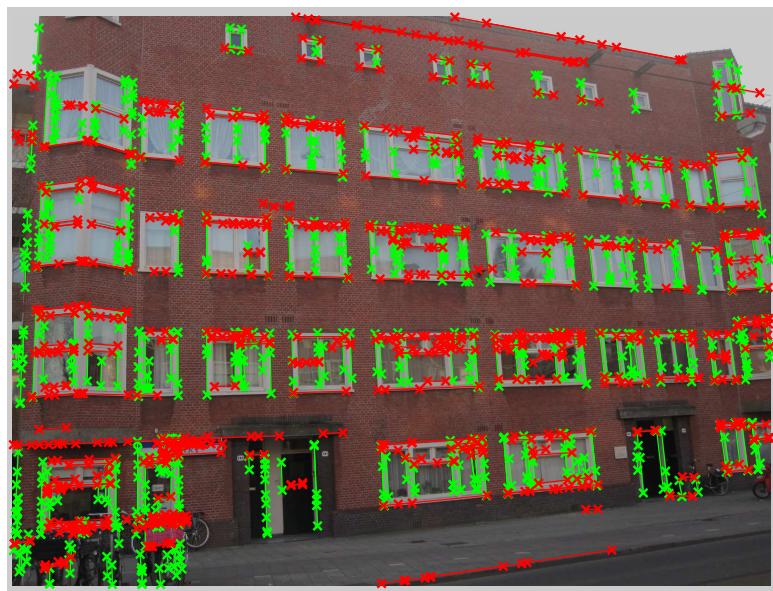


Figure 7: Result of θ constrained Hough transform



Figure 8: Found connected corners

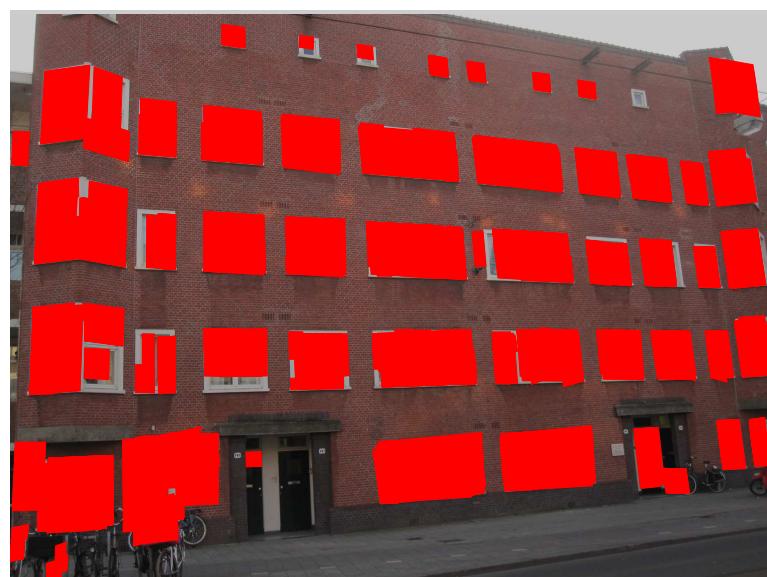


Figure 9: Window regions

Figure 9 displays the result of the connected corner approach on an unrectified scene. It contains 110 windows of which are 109 detected, this is 99%. Furthermore there are some False Positive areas, this is about 3 %. Sometimes a window is not detected, for example the window on the right top isn't detected, this is because its smaller than the minimum window width.

2.4.6 Conclusion

As can be concluded from the results, the connected corner method is suitable for, and robust to, scenes with a variation in window sizes and types. This makes the connected corner approach suitable for a wide range of window scenes where no or few prior information about the windows is known. Furthermore as no image rectification is required, the system has a low input requirement as it is independent of calibrated input data and 3d information about the building.

2.4.7 Future research

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.

Furthermore the next step in this study would be to group the connected corners to windows and sub windows. The big window that contains sub windows could be found by calculating the convex hull of the red areas in Figure 9. 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 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 certainty of the sub window.

2.5 Facade rectification

2.5.1 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. This is because it only uses four point to point correspondences and doesn't take the camera lens distortion into account. Another downside is that it requires (manual) annotation of the corner points. A more accurate and fully automatic method is desired .

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 autonomy of this module and our focus on the annotational part, we used existing software to apply the window rectification. I. Estebans *FIT3D toolbox* [2] comes with an addon 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 this addon of the *FIT3D toolbox* [2] 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.

2.5.2 3D plane extraction

We started by taking a series of 7 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

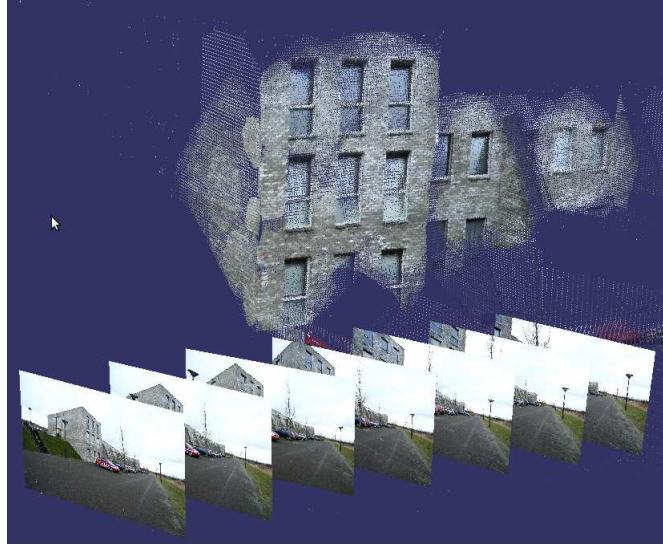


Figure 10: Example series of 7 consecutive frames, (dataset: FIT3D toolbox[2])

with a voting mechanism is used to extract the camera motion. For details please read [2].

The frames were 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*

2.5.3 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 com-

putational 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 $w \times h$, where w, h are the width and height of the image. To give an indication, for the *Anne1* 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 through the houghline endpoints and calculated the intersection with the 3D plane. The result is a 3D point cloud where each point is labeled to their 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. Let's 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.

TODO: ref axis angle wikipedia

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 $u = \text{cross}(z, z')$ where cross defines the cross product of two vectors. The magnitude of the rotation α is equivalent to the angle θ 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 to their Houghlines.

2.5.4 Results

We show the result of two datasets, Anne1 and Dirk. Note that we also rectified the image pixels itself, this is only for the purpose of displaying the projected Houghlines in context.



Figure 11: Dataset: Anne1, Original, (unrectified) image

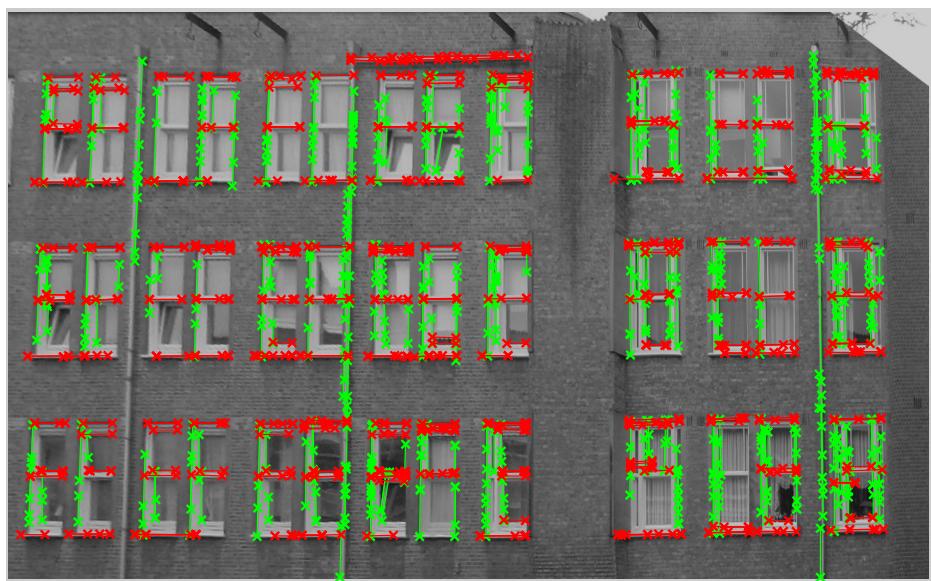


Figure 12: Dataset: Anne1, Projected Houghlines on rectified image



Figure 13: Dataset: Dirk, Original, (unrectified) image,

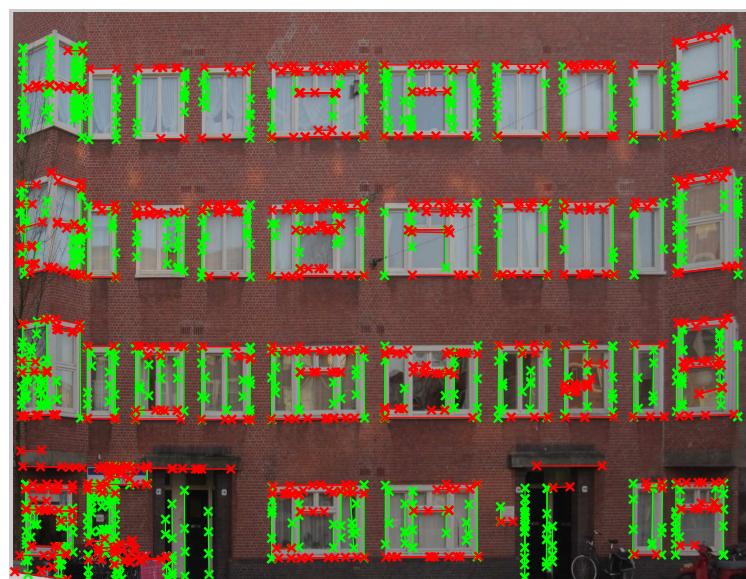


Figure 14: Dataset: Dirk, Projected Houghlines on rectified image

TODO: Met iemand overleggen wat betere volgorde is

2.6 Datasets

To avoid overfitting we used multiple datasets that have a large property variety. Furthermore every dataset has a challenge/handicap. We used three datasets that are recorded in the suburban area 'de Baarsjes' of Amsterdam. All images have a original resolution of 3072x2304 px and are downscaled (using trilinear interpolation) to 1280x1024 px.

2.6.1 Anne1

The challenge of this dataset is that it suffers from rectification errors. This makes the window alignment a challenging task as it assumes the windows to be aligned with the x-axis and y-axis of the image. The rectification error can be seen as the skew window alignment (and skew drainpipe) on the left of the image. Furthermore the yaw value of the camera (horizontal viewing angle) was (relative to the other datasets) high, making parts of the windows occluded. The height of the windows on the right side of the image is 698 px, at the left side this is 320 px: a resolution of more than 2 times smaller making it hard to detect the left windows. Trilinear interpolation is used to minimize this loss. To reduce the number of handicaps (and to focus on the rectification and occlusion error) we cut off the bottom of the image which included cars, unaligned doors and windows.

TODO: image of dataset

2.6.2 Anne2

This dataset contains images of the same scene as Anne1. It has zero rectification error: the windows are perfectly aligned, although the resolution on the left is as in the Anne1 dataset two times smaller. The challenge of this dataset are the occlusion artefacts and the bottom area of the image (cars, unaligned doors and windows).

2.6.3 Dirk

TODO: ipv realistic alledaags?

The Dirk dataset is realistic as it contains light spots on the facade, bicycles, and an occluding tree. It contains zero rectification error but the windows are partially aligned. The windows are very closely located (making it hard to detect non-window areas between them), furthermore they differ in shape,

size and in type. The yaw of the camera is relative low, implicating little or no occlusion artefacts.

TODO: uitbouw van de ramen

TODO: tabel van datasets en handicaps!

2.7 Method II: Histogram based approach

2.7.1 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.

TODO: linken

The extraction of the windows is done in different steps, first we rectify the image 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. We present a regular and alternative window alignment method followed by two different kind of window classifications.

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 same size.

2.7.2 Extracting the window alignment

Regular window alignment

We introduce the concept alignment line. We define this as a horizontal or vertical line that aligns multiple windows. In Figure 15 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 12. For the vertical alignment lines the number of vertical Houghlines is high, see green lines in Figure 12.

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 $H(\text{horizontal})$ 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 15.

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. Therefore we smooth the values using a moving average filter. The result, red lines in Figure 15, 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 $0.5 \cdot \max \text{Peak}$. (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 15.

Alternative window alignment

As you can see in Figure 16 a few window alignment lines are not found and a few lines are found at wrong locations. The right side of the window frame of the first 4 columns of windows is not found. 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 examine the projection profile of the *horizontal* Houghlines projected on the X axis, X_h , Figure 17. 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 17.

$$D = \text{abs}(X'_h)$$

Next we extract the locations of the peaks as the previous method.

Fusing the window alignment methods

We have presented two window alignment methods, next we fuse the methods to gain a robust window alignment. The target is to have as few as possible false positives while detecting all alignment locations.

If a window alignment position is found by both methods, often the peaks are located very close to eachother, see Figure 18. If this is the case we want to fuse the results by grouping the peaks. Most of the times the peaks indicate the same window alignment but have some disparity. This is often the case when horizontal lines stop at the *inside* of a window frame while the vertical edges are located at the *outer* side of the window frame (this is supported by the fact that the disparity is often the size of the windowframe part). Yet, in other cases close peaks indicate different windows that are

just happen to be located closely. To apply a proper grouping of the peaks the challenge is to distinguish these two cases.

First we decrease the total number of found window alignment locations by increasing the individual thresholds (from $0.3 * \text{max peak}$ to $0.5 * \text{max peak}$). Note that this has a positive side effect that the peaks that are found are more certain. After this we group the peaks as follows:

First we calculate the average of the maximum window frame part and the minimum window distance and call this the maximum peak group distance G . Next we compare all peaks and if the distance between two peaks is lower than G , we discard the peak with the least evidence (lowest peak). The result, a set of unique peaks, can be seen in Figure 24.

The advanced peak grouping is only required for the vertical alignment of the windows: The horizontal inter window distance is often big enough to not be mistaken by a window frame part.

TODO: peak merging result misses for 1 dataset?

Results

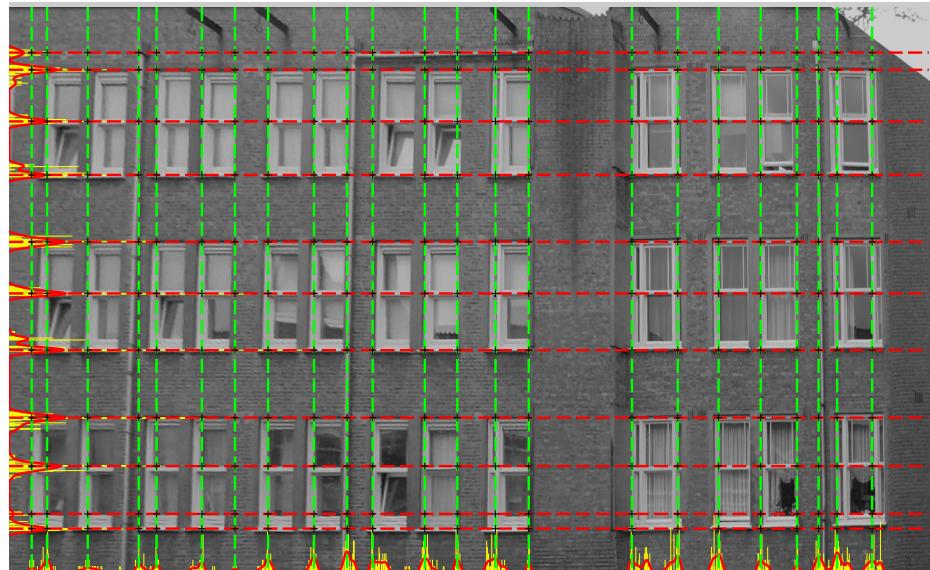


Figure 15: Dataset: Anne1, Regular window alignment (parallel projection): Based on a smoothed histogram (red line) that displays the amount of overlapping Houghlines, for the column division the horizontal houghlines are counted (at each y position), for the row division the vertical houghlines are counted (at each x position)

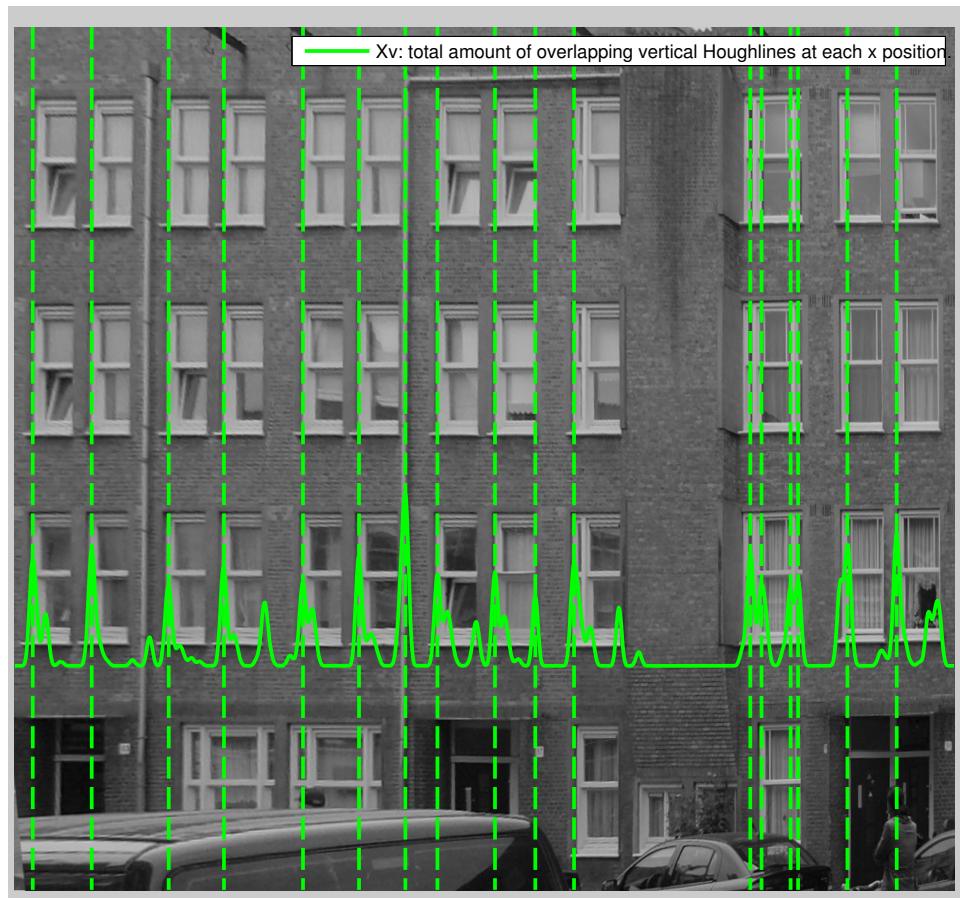


Figure 16: Dataset: Anne2, Regular window alignment: Based on a histogram that displays amount of overlapping vertical Houghlines at each x position



Figure 17: Dataset: Anne2, Alternative window alignment (orthogonal projection): Based on the shape of the smoothed histogram function. For the column division, the number of *horizontal* Houghlines is counted (Note that this is the orthogonal oposite of the regular window alignment method). Peaks (that represent a big decrease or increase of the histogram function) are used for the alignment.



Figure 18: Dataset: Anne2, Regular+Alternative window alignment combined

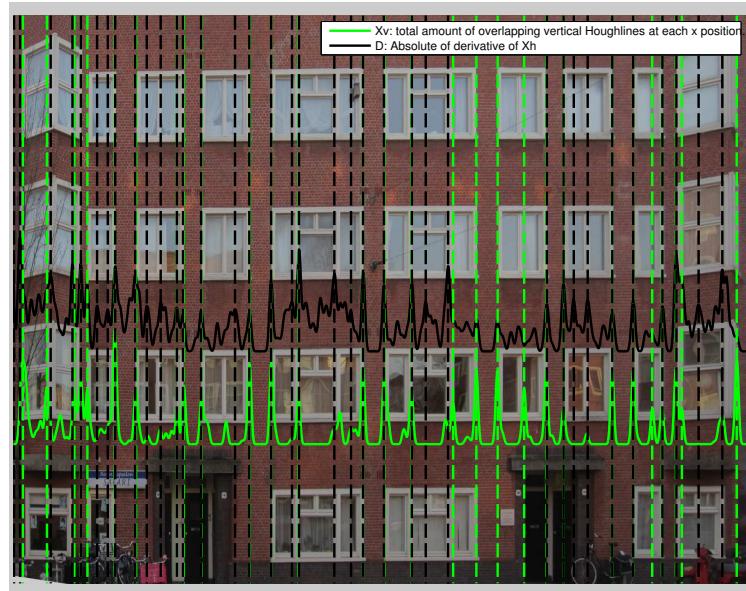


Figure 19: Dataset: Dirk, Regular+Alternative window alignment combined

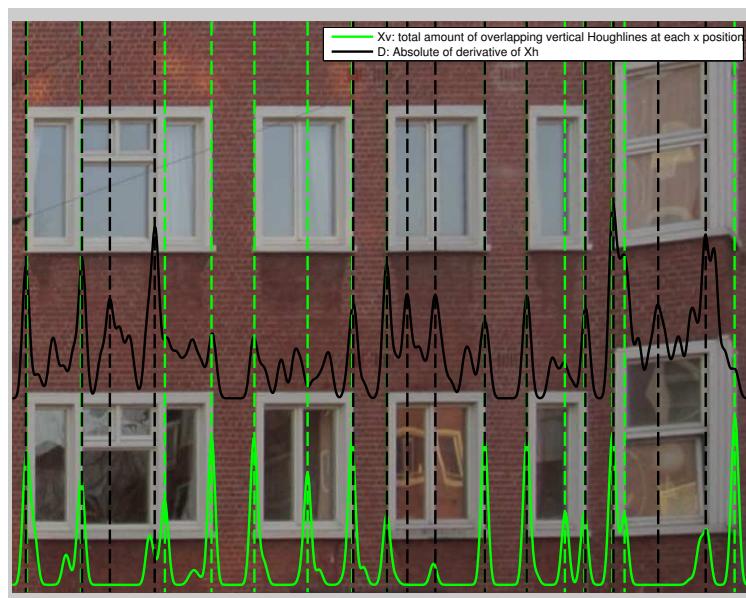


Figure 20: Dataset: Dirk, Regular+Alternative window alignment combined, zoomed

Discussion

The results are promising, as for both datasets 100 % of the alignment lines are positioned either at the boundary of a window area or inside a window area.

Discussion:Many alignment lines

In Figure 19, too many alignment lines detected and not every alignment line is placed correctly. This is mainly because the windows are partially aligned: the two front doors contain many vertical edges that are not aligned with the upper windows therefore a double alignment is found (doors and windows). Another cause of this artefact is that the bicycles on the left (see Figure 30) cause a large amount of found Hough lines which implies a (false) alignment line. Although the causes are clear, we don't have to worry about this additional window alignment lines that lie inside a window area as adjacent window areas are grouped by the classification module.

Fortunately no alignment lines are found at non-window areas.

Discussion:Fusing the methods

If we evaluate the independent window detection results on the Anne2 dataset we see (Figure 17, 16) that in both datasets neither the regular window alignment method nor the improved window alignment detect 100% of the window alignment lines. The effect of fusing the methods is high as, after fusing, 100% of the window alignment is found on both datasets: each window alignment line is detected by at least one method. This is basically explained by the fact that both methods are based on a different Houghline direction. We now discuss the interesting cases, where only one method succeeds in detecting the window alignment.

If we take a look at the regular window alignment in Figure 16 we see that for the first 4 window columns the right side of the window is not detected. This is because the original (unrectified) image (Figure 11) is not a frontal image. This makes building wall extensions (middle of Figure 11) or drainpipes occlude parts of the window. In this case the windows are countersunked into the wall making the building wall itself occluding the right window frame (Figure 16). The color of the reflection of the window is very similar to the bricks and because the window and wall are not separated by the window frame, the edge detector doesn't find a strong edge. This means that on all positions where the window frame is missing, no vertical Houghlines will

be detected and (as the regular window alignment is based on the amount of vertical Houghlines) no window alignment will be found. This artefact can be studied in the edge image Figure 29: few or no edges are present, and at the low height of the peaks in Figure 16 at these positions.

However, the alternative window alignment does find a window alignment on this positions. This is because the method is based on the opposite Houghline direction : for the vertical window alignment the horizontal Houghline direction is taken into account. This occlusion artefact has no effect on the horizontal Houghlines which makes the alternative window alignment method a strong alternative for the alignment of (partially) occluded windows.

In general, the alternative window alignment performs better then the regular window alignment method. This is because this method takes the horizontal window frame parts into account which is a priori stronger as there are more horizontal window parts then vertical window parts present. E.g. Figure 12 every window has (as it has two vertical subwindows) 3 horizontal window frame parts but only 2 vertical window parts). Furthermore the method is more robust because it relies on a higher level of histogram interpretation (by using the derivative). However, in a few cases alternative window alignment method is outperformed by the regular window alignment . For example, in Figure 20, the left side of the second window column is not detected. This is because this window type has no horizontal divisions. Only the top and bottom of the windowframe produce an edge, therefore the derivative of the amount of Houghlines will return a small peak on this position (which is too small to survive the threshold). The threshold is a priori hard to survive because it has a high value as it is determined by taking a fraction of the maximum peak which is located at the windows that do have horizontal divisions (for example the most left or most right window column)).

TODO: result1 drain pipe result niet erg ivm vlg stap

Conclusion

We can conclude that developing histogram functions of the amount of Houghlines is a strong approach towards determination of the window alignment as the results in this work and in previous work a very promising. We proposed two window detection methods and can conclude that the

alternative approach performs better because it is based on a horizontal window division and because it uses a higher level interpretation of the histogram function.

We showed that window alignment becomes a challenging task if the windows are partially occluded. One of the main solutions we proposed is to use multiple window detection approaches. To have a 100% detection rate one needs to be certain that if one approach fails at least one other approach must succeed. We proposed the strong combination of two methods where the first method filled the gap of the second method and the other way around.

We can conclude that a method that fills the gap of the occluded alignment locations must rely on Houghlines that lie in the orthogonal direction of the occlusion. In our case the occluded vertical window parts are supported by a method that detects horizontal window parts.

Conclusion:Relative thresholding

Furthermore we discussed the implication of the use of different window types. Applying a threshold that is relative to a maximum peak doesn't work well on this window types that differ (in for example horizontal divisions) as window alignment lines are missing (Figure 20). This means that if we want to stick with the relative thresholding method, we have to assume a certain equality of the window types.

TODO: make diagram: edge- hough -window alignment - window classification - window grouping

TODO: diagram maken window alignment -*i*, peak merging -*i*, etc

Future research

Future research:Determine window alignment of different window types

A solution of the missing window alignment lines on a scene with different window types (Figure 20) would be to decrease the threshold if multiple window types are found. One could design a measure of variety of the window types. This could be done by taking the variation of the derivative of the amount of Houghlines. This amount of variation will determine the amount of decrease in threshold. Let's explain this by two examples: If there is just a few variation the maximum peak is very representative for a window so the threshold could be for example $0.7 * \text{max peak}$. However

if there is a large variation is found (which means multiple window types are present), the threshold should be lowered to $0.3 * \text{max peak}$ to detect the hard window types (with few horizontal divisions). Another method would be to cluster the amount of Houghlines in $n+1$ values where n is the number of window types (the 1 is for the non-window area). Areas that transcent from a window area cluster to a non-window area cluster (or vice versa) are determined as the window alignment locations. For both methods the challenge is again to detect the window alignment with unknown window types but keep the number of false positives (e.g. a drain pipe) zero.

Future research:Peak grouping

We fused 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 height of the peak could be used as a weight. We used a manual maximum peak group distance (G), this value could also be automatically derived from the image by for example taking a percentage of the window, or by detecting the size of the window frame parts.

It is challenging to handle multiple close peaks, e.g. if 4 peaks are close, then peak 1 could indicate the same window as peak 2 but indicate a different window then peak 4. The location of the close peaks: inside, at the border, or outside the window could add important evidence, some methods of the window classification could be used to detect these values.

Future research: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 [3] 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 off. A system that only refines the results when the resources 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 accurate building recognition which opens the door for augmented reality.

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 align-

ment or refinement of the windows.

2.7.3 Basic window classification (based on line amount)

The image is now divided in a new grid of blocks based on these alignment. The next challenge is to classify the blocks as window and non-window areas: the window classification, we developed two different methods for this.

Instead of classifying each block independently, we classify full rows and columns of blocks as window or non-window areas. This approach results in a 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 12 (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.

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

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 .

TODO: DIT PLAATJE eruit en mooie staafdiagrammen erin

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 21. Its easy to see that the high values, blockrow 4,5,7,8,10 and 11, correspond to the six window blockrows in Figure 23.

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

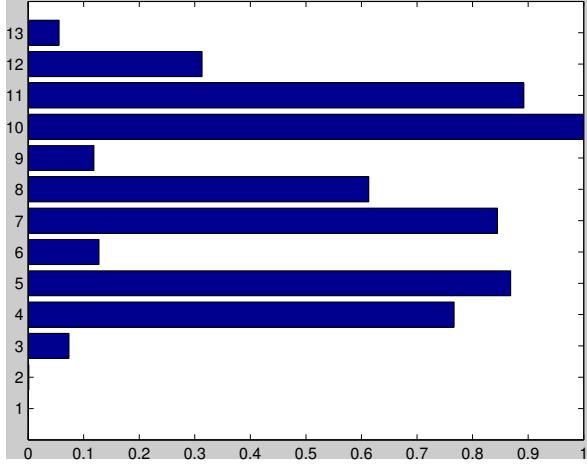


Figure 21: Classification values for window block rows representing the normalized vertical Houghline pixel count of (R)

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.

TODO: ref

This results in a set of Rows and Columns that are classified as window and 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 23 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 23.

As the figure gives a binary representation of the windows it is not possible to see how certain a block is classified. To get insight about this we developed a measure of certainty function.

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

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

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

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

The visualization of the measure of certainty is shown in Figure 22, the whiter the area the higher the measure of certainty.

Results

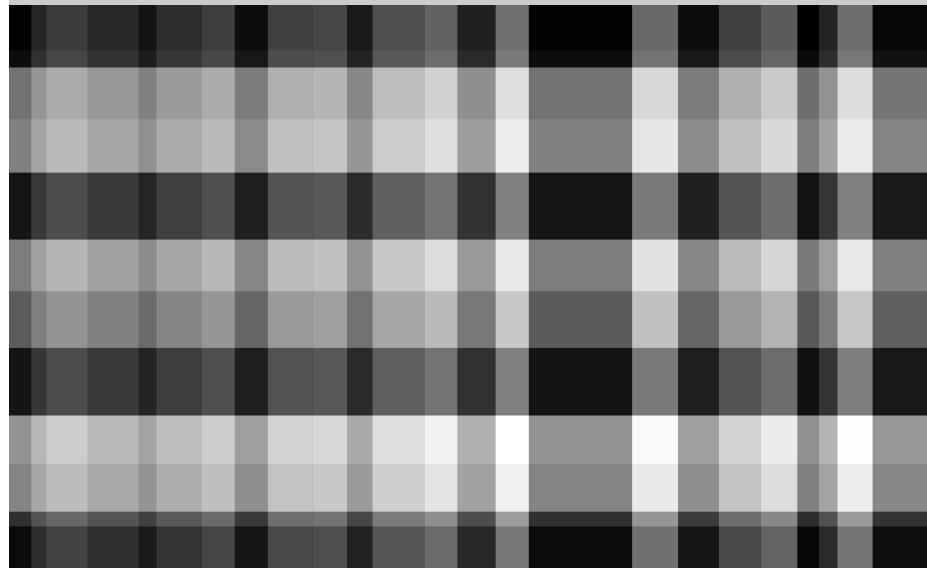


Figure 22: Dataset: Anne1, Basic window classification method:Measure of certainty, the whiter the area the higher the measure of certainty.

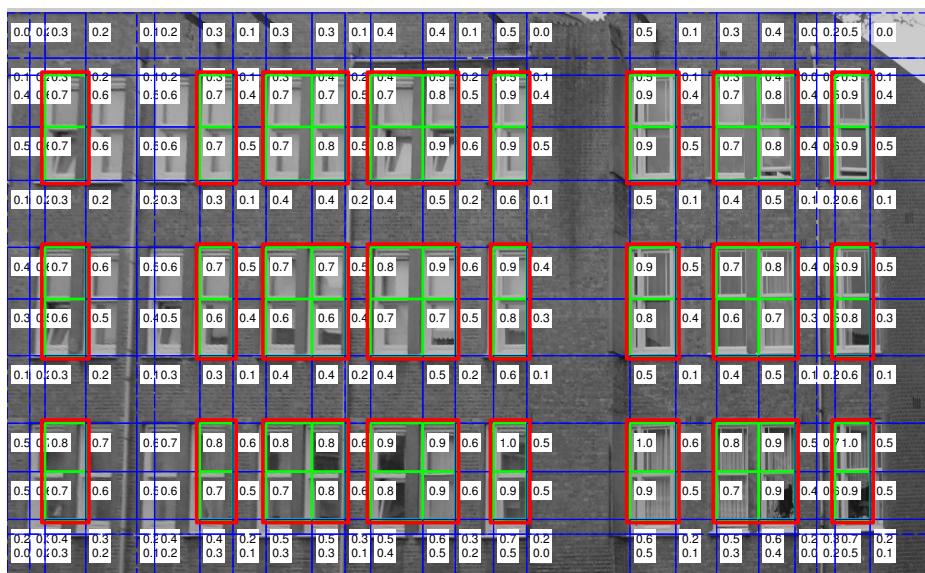


Figure 23: Dataset: Annel1, Basic window classification method: The extracted windows, red:the grouping

TODO: include results of binary classification

The bright rows and columns in Figure 22, indicate window blockrows and blockcolumns, whereas the dark rows and columns indicate non-window areas. The stripe patterns support that the classification process exist of individual row and column classification. The area of window positions is particularly white, as it intersects a bright (positively classified) row and column. One can compare this result to the windows in the original image (Figure ??).

TODO: include table of false positives etc, detection rate etc. etc. misch staat dit nog op github

Discussion

The outcome of this method is non-deterministic, as it depends on to the random initialization of the cluster centers. This means that our results could be correct by coincidence. To exclude this artefact, we ran the cluster algorithm 10 times on all datasets. Unfortunately for the Dirk dataset, 2 of 10 times it resulted in a bad result. This result can be found in Appendix Figures 31, 32, 33 and 34. This result can be explained as follows. The Dirk dataset has window types in the middle of that contain a horizontal subdivision while the others don't. The windows in the middle will cause k-means to drag the cluster center to a value that is too high making windows classified as non-window areas.

Future research

A solution to the bad luck on the initialization of the cluster centers is to increase the number of clustercenters to $n+1$, where n is the number of window types (the 1 is for the non-window area).

Conclusion

We can conclude that this certainty based classification works quite good on this dataset. However, it requires a very well alignment of the windows which is a disadvantage. As it is based on the number of found Houghlines, the errors in the window alignment will propagate to the classification. This makes this classification method inappropriate for a system where one cannot fully rely on the window alignment.

This conclusion amplifies the need for a robust classification method (that is independent, (or at least less sensitive)) to window alignment errors.

2.7.4 Improved window classification (based on shape of the histogram function)

TODO: introduce a little bit

TODO: say that anne2 dataset is more rectified

If we take a look at Figure 24 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 24. 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 the same, the projection profiles are projected to the Y-axis.

Results

TODO: make a diagram, histogram - alignment lines - ζ window classification etc.

TODO: discuss drain pipe, dat het goed is dat het resultaat mooi is ondanks drainpipe

Results: Dirk dataset

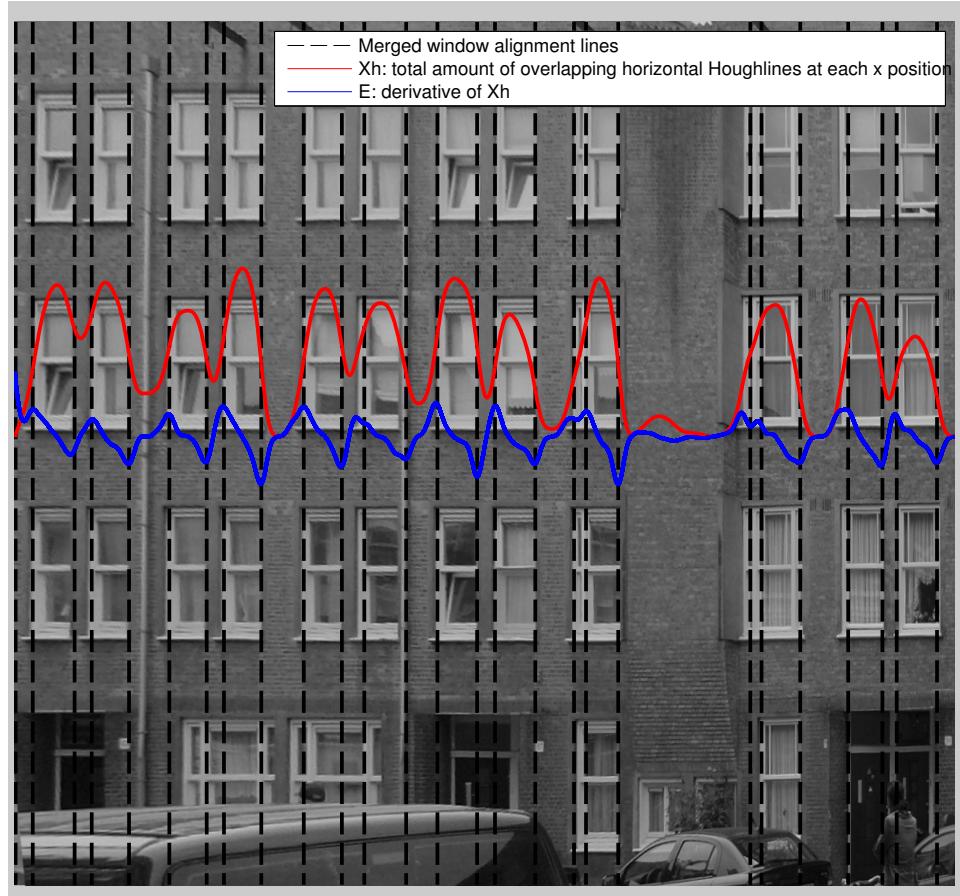


Figure 24: Dataset: Anne2, Improved window classification method: The red line shows concave shapes at window locations and convex shapes at nonwindow locations

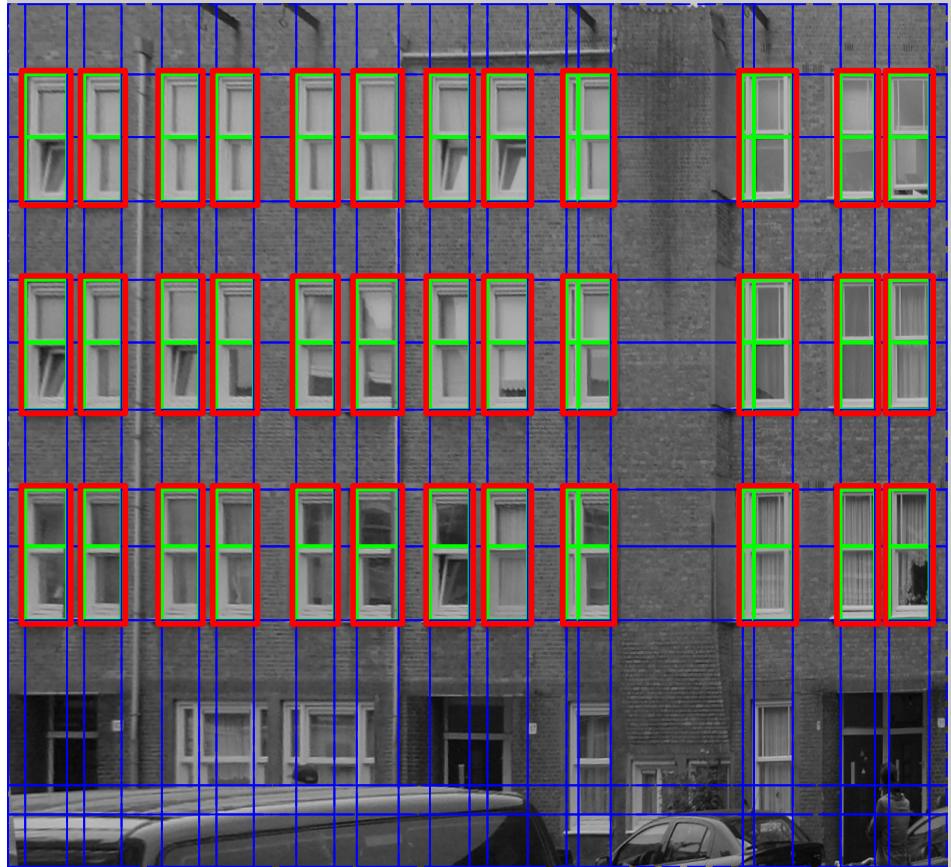


Figure 25: Dataset: Anne2, Improved window classification method: The extracted windows, red:the grouping

Although the scene contains a lot occlusion artefacts and suffers resolution loss, especially on the left side of the image, the results for the Anne2 dataset, a 100% detection and a 100% true positive rate are very well. This is mainly due the high interpretation level of the Histogram function. We used the fact that the histogram function has a very consistent pattern, at every window area its shape is convex and every non-window its shape is concave.

Results: Anne2 dataset



Figure 26: Dataset: Dirk, Rectified image of a realistic scene which is realistic as it contains light spots, bicycles. Note that the windows are partially aligned and the differ in size and type

TODO: IMPROVED result on Anne1 dataset, investigate which combinations of dataset and methods i miss

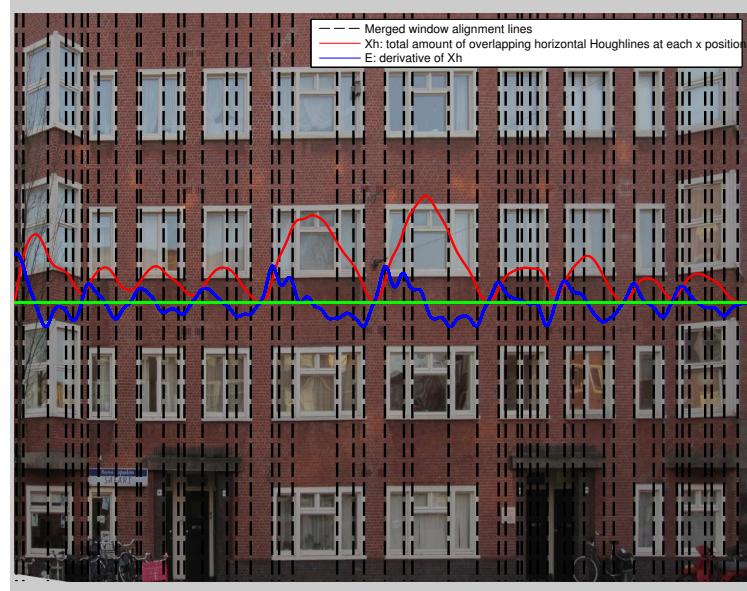


Figure 27: Dataset: Dirk, Improved window classification method: The red line shows concave shapes at window locations and convex shapes at nonwindow locations

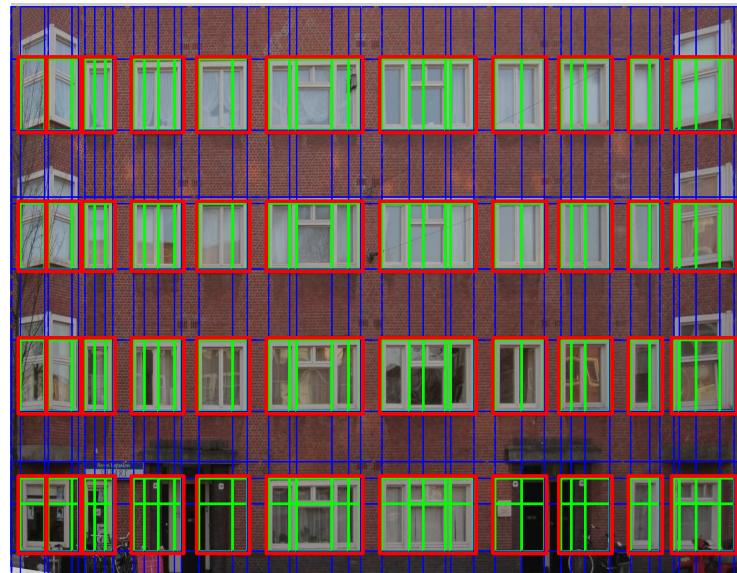


Figure 28: Dataset: Dirk, Improved window classification method: The extracted windows, red:the grouping

Table 1: Improved window classification results on Anne2 and Dirk dataset

Type	Anne2 dataset (Fig 25)	Dirk dataset (Fig 28)
Detection rate	100 %	100 %
True positive rate	100 %	87 %
False positive rate	0 %	11 %
False negative rate	0 %	9 %

Table 2: Grouping results on Anne2 and Dirk dataset

Type	Anne2 dataset (Fig 25)	Dirk dataset (Fig 28)
Grouped correct	100 %	90%
Grouped incorrect	0 %	10%

TODO: certainty based (white strokes) classification on anne2 dataset,
pas als skyline af is, img zijn wel al klaar

TODO: remark that we goedkeuren partieel resultaat van windows helemaal rechts en links dirk dataset

TODO: Interpretation of the results of the Anne1 dataset

TODO: compaire to other classification methods

TODO: (perform both classifications on both datasets (2x3 img), visual concatinate images vertically)

TODO: especially the horizontal division of the lowest row of windows is amazing, refer to

TODO: Meer over lof spreken

We discussed in the section about window alignment that the classification module would handle the redundant window alignment lines. This can be seen in Figure 28 where the classification took advantage of the fact that all alignment lines are located at a boundary of a window or in a window. The peak fusing module discarded many close alignment lines and the residual redundant alignment lines create small green adjacent subwindows which were grouped to clear red big windows.

This grouping result is promising as 90% of the grouping went well. This is a quite good result given the scene contains a large variety in window shape, window size and window type. This result is mostly due an almost perfect window alignment and classification (as the grouping only groups adjacent positively classified subwindows). The first two window rows in Figure 28 however are classified as two groups but this should be one. Normally the peak merging step would handle this problem, but with this dataset we could not use a large *maximum peak distance* value because the windows in the image (especially the first and last columns) are very close. More on this in *Future research on window classification (2.7.4)*.

Conclusion

It is a typically artificial intelligent approach, to look for a high interpretation of the data. We can conclude that the results become very robust if one takes a high level interpretation of the Histogram function.

Furthermore we can conclude that:

- Redundant lines found by the window alignment module cause no

problems as long as they are located at a boundary of a window or in a window.

- The grouping module contained a small weakness, a fix is provided in the next section.
- The improved window classification outperformed the basic window classification.

Future research

TODO: put a opbouwende line in future work

Future work:Extensive evaluation methods

As the classification worked very well it would be interesting to find out on which point it will fail? This could be investigated by using low quality images (which are taken for example with a cell phone), furthermore we could downscale the images and/or add Gaussian noise to the data.

TODO: speak about video, or sequence of images, detection + tracking etc

The results of the classification module depend heavily on the result of the window alignment. It would be nice feature research to make an independent evaluation of the classification modules. This could be done with a random window alignment generator. Some evaluation should be developed and it would help if the windows are annotated.

Future work:Grouping error minimalisation

Although the grouping module gave promising results it can be optimized. The grouping module now groups adjacent positively classified window areas. Some window in Figure areas are false negatively classified, see the left side of Figure 28. This is caused by a small area between the windows that is classified as a non-window area. This could be solved by adding a minimum size constraint of a area to be threatened as a non-window area. In this way small negatively classified areas cannot interupt the adjacent windows.

Performance measure for extreme viewing angles

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.

TODO: compare classification methods and explain differences

2.8 Conclusion

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

TODO: Furthermore our algorithms work in real time.

3 Appendices

3.1 Figures

TODO: check if appendix images not already in ch5!!



Figure 29: Dataset: Anne1, Result edge detection

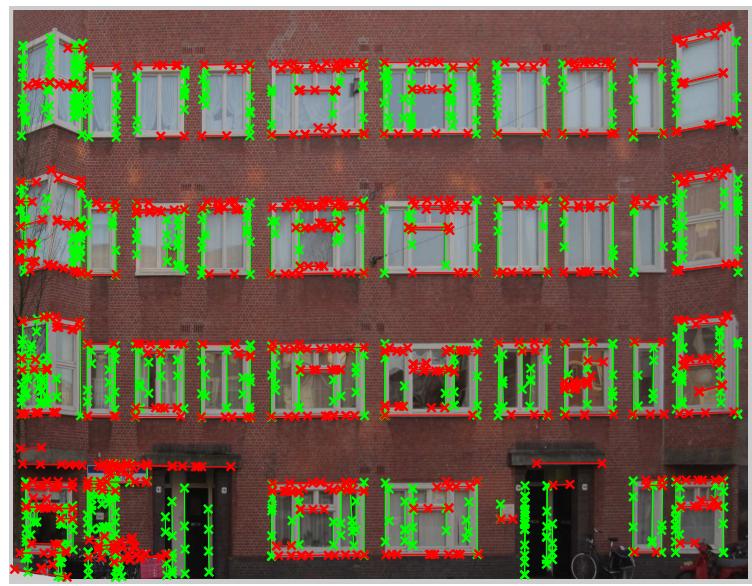


Figure 30: θ -constrained Hough transform

3.1.1 K-means bad luck

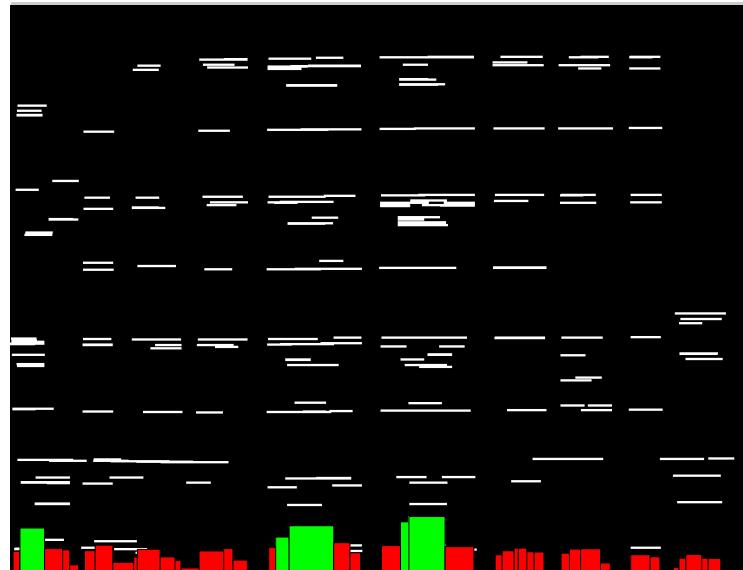


Figure 31: Dataset: Dirk, Horizontal Houghlines

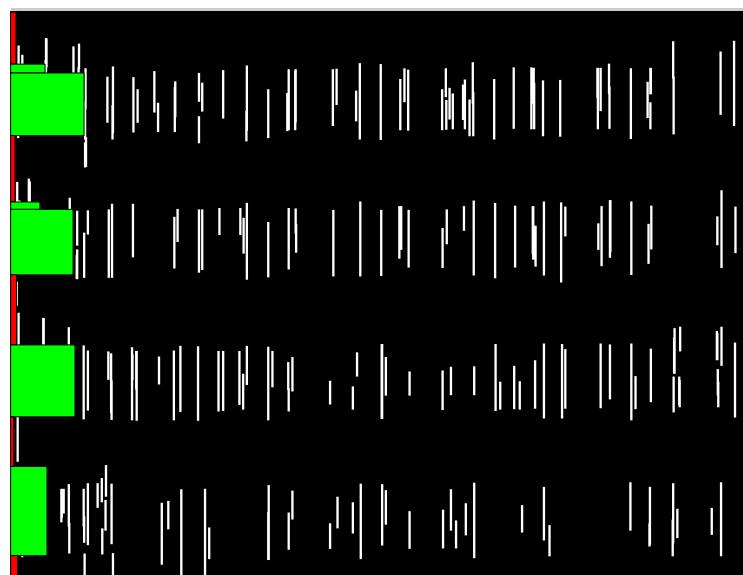


Figure 32: Dataset: Dirk, Vertical Houghlines

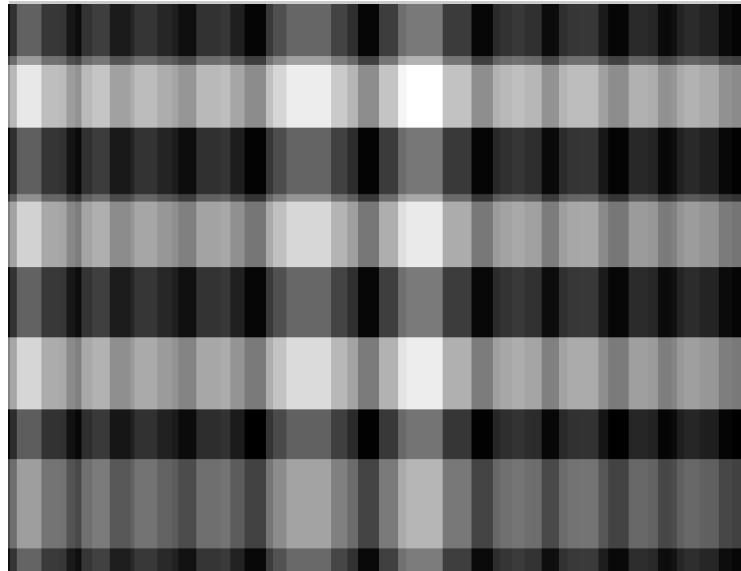


Figure 33: Dataset: Dirk, Basic window classification method:Measure of certainty, the whiter the area the higher the measure of certainty. The values of the window areas in the middle are relative large, this is due the horizontal subwindows.

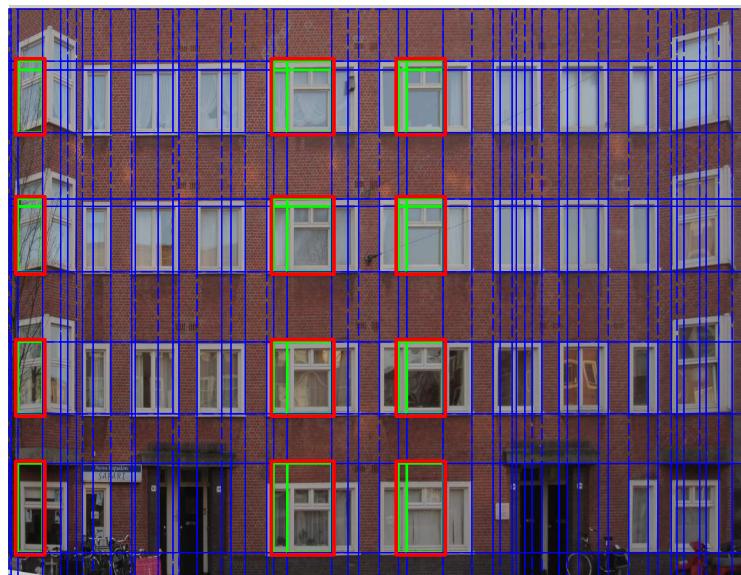


Figure 34: Dataset: Dirk, Extracted windows

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] 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.
- [3] 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.
- [4] Pascal Müller, Gang Zeng, Peter Wonka, and Luc Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3), July 2007.
- [5] Shi Pu and George Vosselman. Refining building facade models with images.
- [6] 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.
- [7] 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.