DETECTION OF WINDOWS IN FACADES USING IMAGE PROCESSING ALGORITHMS

Milos Miljanovic
Institute of Information Systems,
Technical University of Vienna
Karlsplatz 13, Vienna 1040 Austria
milos@kr.tuwien.ac.at

Thomas Eiter, Uwe Egly
Institute of Information Systems,
Knowledge Based Systems Group
Favoritenstrasse 9-11 1040 Vienna Austria

Abstract
In this paper, a method for window detection in images of facades is presented. Windows play an important role in the context of deformation analysis of facades, and automatic detection from images is needed in novel, reflector-free measurement setups. The method we present is capable of detecting windows at several different orientations and scales, and yields good results for practical settings. The performance is demonstrated by an evaluation of the method using the Vienna and Zurich database sets of window images, which shows that window coordinates can be reliably detected.
***Keywords*:** façade, window detection, image processing.

1. Introduction

The windows of a building are considered as objects of great interest, since they provide information on the texture of the building to the user. In this paper, we will more specifically focus, as motivated below, on the facades of buildings. The goal is a method which gives us accurate window coordinates in the image of a facade. We develop such a method which uses an engineering approach rather than a machine learning approach. The argument is that such a method is more reliable and efficient than the machine learning approach. Furthermore, the method we develop may also provide the user with a more general description of objects which can be found in the image.

Motivation Detection of windows in facades and buildings can be of great use in monitoring deformations and rigid body movements. Traditionally, the deformations have been monitored by measuring reflectors attached to the buildings. However, novel measurement setups avoid such reflectors (e.g., for esthetical reasons), and measurement has to be done on image-based data alone, as possible with modern video-theodolite. As for deformation monitoring, the coordinates of windows in a time series of images are identified as reference points for assessing whether, and which, deformations are happening. Currently, the engineer has to determine the window coordinates at each epoch manually. The idea is an automated detection using interest operators (IOPs), most notably Harris[1][15], and Foerstner[2][14], which single out points with special properties from the image. In order to effectively apply this approach, one needs an implemented method for automatic detection of windows from an image respectively a series of images.

Problem description We consider the basic problem where the method is given a single image of a facade. The outcome of the method should be the (exact) coordinates of windows found in the image. The windows in the image are not necessarily rectangular, but also might have other shapes due to the perspective or different shape of windows. The method should be robust with respect to different lightning conditions, and also with respect to shadows as met in practice.

Fig. 1. Examples of images containing windows.

Fig. 2. Examples of Foerstner and Harris operators.

Current Methods In a paper by Haider et al [3], an algorithm for window detection is presented. The algorithm uses machine learning techniques, to find the coordinates of the window. Other than this method, algorithms have been designed which try to capture rectangular-shaped objects in an image, using the Hough Transform [5]. Although this method is very effective, it is not suitable for window detection, since it captures all objects in an image which resemble to rectangles. Applied to the image of a facade, far too many objects will be classified as windows. However, we can use this method to find, besides windows, a number of other objects of interest in the image (like balconies, bars, etc). Still we need an algorithm which can filter windows from other objects, and which is also applicable under the conditions mentioned above.

Contribution Our main contribution is a method which takes, as input, an image of a facade, and provides, as output, for each of the windows in the image a bounding box. The method combines several image preprocessing algorithms, but also employs two new algorithms, which have been specifically developed. The first performs thresholding of a façade image, and the second segmentation of objects in a facade image and window separation from other objects. Furthermore, we present two numeric measures for the quality of windows detection, which are motivated by the application, and evaluate an implementation of our method against these measures.

2. Preliminaries

The images which we use here are gathered from the Vienna[8] and Zurich[9] public domain databases, which are available on the web and provide us with a very diverse set of images for testing. The pictures in the database give a clear view of the facade, but not of the building.

2.1 Assumptions

One of the assumptions is that the picture should not be taken at a small angle (less than 30$^\\circ$). Another important aspect is that the user should use a high quality instrument, such as a video-theodolite, in order to deal with the problems encountered with depth of focus, saturation, motion blur, and lens distortion. Images which are not taken in this manner do not provide the user with good information. However, these assumption apply in practice, even with the traditional methods, and thus are no limitation.

2.2 Problem constraints

The main constraint which is set in this paper is that facades should be clearly visible, and not covered with excessive objects. In particular, the images do not contain additional objects, such as cars or trees (which are also hindering manual measurement). The performance of our method depends on the quality of the image. If additional objects are present, manual (as currently done) or (semi-)automated segmentation has to be done. On the other hand, shadows do not cause problems, because of the thresholding algorithm incorporated in our method.

3. Method

This section describes our method, and the image processing techniques which are used by it. The method works in four steps. In the first step, the facade in the image is classified into one of several types. After that, appropriate image preprocessing algorithms, are applied. In the third step, thresholding techniques are applied, and in the fourth and final step the image is segmented using Histogram Median Filtering (HMF).

3.1 Classification

In image processing, it is quite common to use simple statistical descriptions of images and subimages. The notion of a statistics is connected to the concept of probability distribution, generally the distribution of signal amplitudes. For a given region which could conceivably be a window, the probability distribution function of the brightness in that image is defined. The algorithms need to have prior knowledge as to what kind of façade they are segmenting. There are three different types of images in the database: classical, modern, and ordinary. The probability distribution function of the brightness has been used to differentiate between three types of facades. This function behaves differently when dealing with different types of facades (modern facades have a very low value, classical have a high value due to the texture, and the rest are classified as ordinary facades).

3.2 Preprocessing

Before the image of a facade can be segmented, it needs to be preprocessed, so that the noise can be discarded. The choice of preprocessing algorithms depends on the type of facade. For example, normal buildings need very few preprocessing algorithms to be applied, such as normalization and equalization, whereas classical buildings need contrast stretching. The description of these algorithms can be found below. Modern buildings require unsharp masking, because the windows are too bright.

Contrast Stretching Frequently an image is scanned in such a way that the resulting brightness values do not make full use of the available dynamic range[6]. By stretching the histogram over the available dynamic range we attempt to correct this situation. If the image is intended to go from brightness 0 to brightness $2^B - 1$ then one generally maps the 0 percent value to the value 0 and the maximum to the value $2^B - 1$. The appropriate transformation is given by:

$$b[m,n] = (2^B - 1)\frac{a[m,n] - min}{max - min} \quad (1)$$

This formula can be sensitive to outliers and a less sensitive and more general version is given by:

$$f(x) = \begin{cases} 0, & a[m,n] \leq p_{low} \\ (2^B - 1)\dfrac{a[m,n] - p_{low}}{p_{high} - p_{low}}, & p_{low} < a[m,n] \leq p_{high} \\ (2^B - 1), & p_{high} \leq a[m,n] \end{cases} \quad (2)$$

where p_{low} and p_{high} are dynamic ranges of the percentages in which the contrast of the picture is to be stretched, usually set to 10% and 90% respectively.

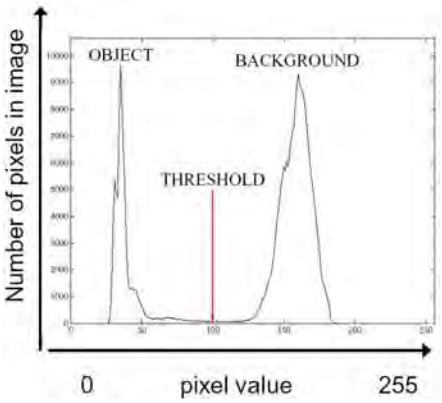

Fig. 3. Selection of a threshold.

Unsharp masking A well-known technique from photography to improve the visual quality of an image is unsharp masking, i.e., to enhance the edges of the image. This means isolating the edges of an image, amplifying them, and then adding them back into the image[2]. This leads immediately to the technique:

$$a^{i}[m,n] = a[m,n] - (k \times \Delta^{2} a[m,n]) \quad (3)$$

where the term *k* is the amplifying term, and k>0.

3.3 Tresholding

Once the image has been preprocessed, thresholding can begin. In order to separate an object from the background, in this case, a window from the facade, an appropriate threshold needs to be chosen. When this is done, bimodal distribution is exploited. Distributions are broad, and may overlap. Possible problems include shadows because they are dark and may be classified as an object. Figure 3 shows an example of how the threshold should be chosen, it is located between the object and the background. In order to get rid of the shadows, which are sometimes included as part of the object, the histogram has to be smoothed.

The process of thresholding is the most important because the windows need to be preserved, and excessive objects have to be removed. Several thresholding algorithms from the literature have been tried (such as isodata, and background symmetry algorithms)[6], but the results have not been robust. Therefore, we have developed a special algorithm for thresholding, described below.

Local Adaptive Thresholding 4 (LAT-4) This algorithm has been specifically designed for window detection. First, the horizontal and vertical histograms are calculated. The 1D Convolution[6] is applied to smooth the histogram. After that, the image is partitioned according to the position of local minima. These partitions are then used as sub-images, which are later on divided into four parts. Therefore, four different thresholds are used for each subimage, or in total $4 \times N_{min}$, where N_{min} is the number of minima. This algorithm has been chosen, as it gives also good results when the light is not uniformly distributed in the image.

3.4 Segmentation

Once the image has been thresholded, the analysis of the histograms comes into place. The main core of the method for window detection is Histogram Mean Filtering (HMF) developed specifically for the detection of windows in facades.

3.4.1 Histogram Mean Filtering

Windows in facades follow a particular pattern, and they have similar pixel values (due to their similarity). After thresholding, the user is left with a black and white image (0,1) which holds the information on the objects present in the building. The next step is to calculate horizontal and vertical histograms H_h and H_v respectively, given by

Fig. 4. Image before and after applying LAT-4 (white lines in the first image represent local minima).

$$H_h = \sum_{i=1}^{W} I_t(h,i), \qquad h = 1, \ldots, H \tag{4}$$

$$H_v = \sum_{i=1}^{HW} I_t(i,v), \qquad v = 1, \ldots, W \tag{5}$$

where (*W*, *H*) are the width and height of an image, respectively, and I_t is the thresholded image. Windows are the most numerous objects to be found in the I (depicted as white pixels in $I_t(x,y)$) and are aligned in rows and columns i.e., they form a pattern. The values of the two histograms will have high volume when processing an area with windows in them. Therefore, after the calculation of the two histograms, the values which are greater than the medians M_h, M_v will hold the position of the window along the horizontal and vertical axes. After filtering the histogram values, the user is left with a series of numbers in the range of (1..W) and (1..H) which actually represent values along the (h,v) axes that hold window coordinates.

3.4.2 Histograms at an angle

The limitation of Histogram Median Filtering (HMF) is that it can only work when the windows are aligned at a near 90° angle. To overcome this limitation, histograms are taken at an angle α, and new histograms $H_h(\alpha)$ and $H_v(\alpha)$ are calculated respectively. The new algorithm HMF(α) works exactly the same as HMF, except that it does not know the exact value of α. The algorithm starts by trying with an angle 10°, finishing at 100°, with a step of 10°. The results obtained in this way are suitable in many cases (see Section~\ref{sec:results} for a further discussion). The quality can be improved by iterating with a step of 1°, which incurs high computation time, or by dynamic step adjustment, which decreases the step if the matching gets close.

Figure 5 (right) shows lines that actually represent the histograms which were taken at an angle. The algorithm stores this information, and finds the best fitting line which will capture the window by comparing histogram values (the best one is minimal in this case). The final segmentation can be seen in Figure 6 (right).
The idea behind angled histograms is that they will capture much more information than the “regular”, histograms, because they can capture more information in the thresholded image. HMF works first by calculating regular histograms. If the results are poor, then the histograms will be taken at an angle. The values are stored, and then later on compared to see which one has captured most information.

4. Evaluation

In order to measure the quality of an algorithm for window detection numerically, different criteria may be considered, depending on particular application. Three natural criteria for assessing the similarity between the captured window *W* and the target window *T* (i.e., the manually determined ground truth) are structure (i.e., shape), the area extent, and the location in the image. The desire is to have the captured window include the real window, in particular when doing the segmentation of the image.

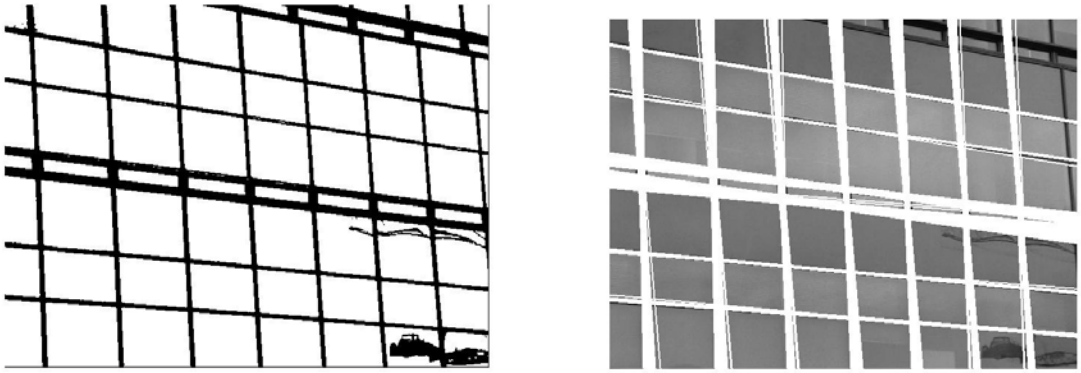

Fig. 5. Tresholded image (left) and histogram "candidates" for capturing the window.

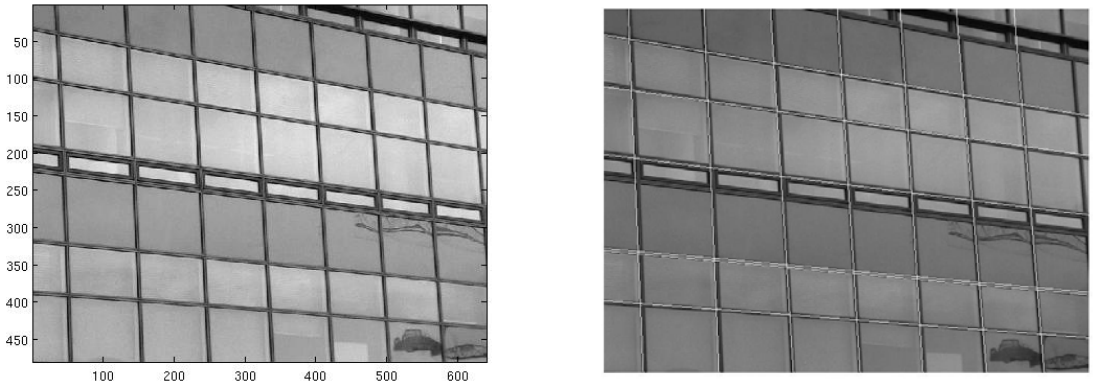

Fig. 6. Original image (left) and final segmentation (right)

4.1 Structure Similarity

There are various ways to measure similarity between two structures[7], but there is no precise methodology for determining similarity between two rectangles (windows). Rectangles can be observed as polygons, but this is not the most reliable way. Only a degree of similarity is possible to measure using existing methods, which are in fact only classification algorithms as they can tell the user whether two structures are either very similar or completely different.

A I approach for measuring structure similarity would be to use the width/height ratio of the target and real window. In order to measure a degree of similarity of two windows, it is appropriate to use distances between corners. For the purposes[10][11][12] of our application, the similarity of shapes is measured by comparing the distances between the coordinates of windows, i.e. their corners.

4.2 Area Extent

A natural measure for the similarity of the area extents of the captured and the target window is the ratio between their common and total extents, i.e., the fraction of overlap. We refer to this as *Area Coverage (AC)* which is formally given by

$$AC = \frac{1}{n} \times \sum_{i=1}^{n} \frac{A(T_i) \cap A(W_i)}{A(T_i) \cup A(W_i)} \quad (6)$$

where *n* is the number of windows, and $A(W_i)$ respectively $A(T_i)$ is the area of the captured window W_i respectively the target window T_i.

In particular if it holds that $T_i \subseteq W_i$, i.e. the ground truth is always contained in the captured window, for all i=1..n (as it occurred in our experiments), then the above expression simplifies to

$$AC = \frac{1}{n} \times \sum_{i=1}^{n} \frac{A(T_i)}{A(W_i)} \quad (7)$$

which measure coverage. The quality of the result *AC* is actually the mean of the ratios between the bounding box and the window size. It represents a good quality measurement for the detection of windows, as it provides

the user with information on how well the windows have been bounded by the given coordinates from the output. The goal is to have *AC* as close to 100% as possible. Indeed, when applying the Harris or Foerstner operator on the image to single out the window corners precisely, the captured windows W_i (which are used to filter the results of the Harris or Foerstner operator), should cover only the target windows T_i. If *AC* is close to 100%, no other objects will fit into the captured windows W_i.

4.3 Location in the image

For measuring the similarity of location between the captured and the target windows, we use *Center of Mass (CoM)*. For each captured window W_i and target window T_i, the center of mass CoM(W_i) resp. CoM(T_i) is evaluated. Then, in order to assess how much W_i resembles T_i, the vector from CoM(T_i) to CoM(W_i) calculated to see in which cardinal direction the captured window W_i is moving. This CoM vector is calculated for all windows W_i and T_i in all images, in order to produce results that can provide information on how well the location of the captured window fits with reality.

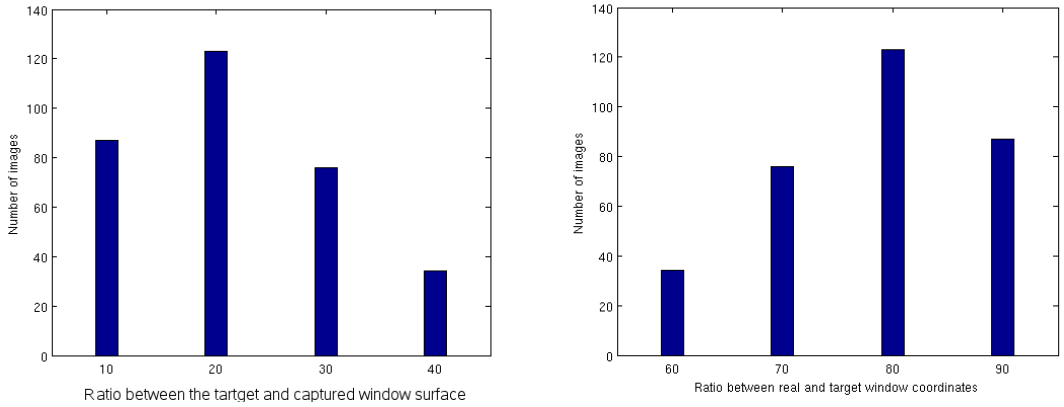

Figure 7. Performance of the algorithm

4.4 Distance between Corners

For measuring shape similarity, distance between corners d_c is used. d_c is another way to measure similarity between to structures. This measurement indicates how the target and the real window are aligned, and it can cover structure similarity very well. The desire is to minimise the distance. d_c is calculated by averaging distances between corners in one picture. This measurement methodology is used to determine by what margin the real window fits into the captured. The results of d_c should correlate with the results of *AC* under the assumption that there is a bounding box present, and can also be used as an alternative to area extent, by normalizing this value with the real height and weight of the window. However, the same method does not correlate with the center of mass, because the distances cannot provide the information on the alignment of the center of masses of the real and target window. Nevertheless, this method represents a good addition when doing the evaluation, as not only does it cover shape similarity, but also area extent.

5. Results

We have tested an implementation of our method on a database containing 500 images in total selected from the Vienna and Zurich public domain[8][9]. The database is full of a diverse set of images, with classical, modern, and ordinary facades, taken from different perspectives. The images used for the experiments had a resolution of 640x480 pixels, and the windows had an average of 50x100 pixels.
On visual inspection, the method yields good results. In all cases it singles out reliably bounding boxes for the windows in the images. These bounding boxes properly contain the windows. When determining the coordinates of the bounding box, the desire is to have a small margin of three pixels present, for the purposes of our application. The small margin has to be present in order to properly calculate the Foerstner and Harris operators.

Area Coverage The performance results of our algorithm with respect to area coverage AC are displayed in Figure~\ref{fig:AC}. As mentioned above, in total 500 images have been used for testing, and the algorithm always returned true bounding boxes of the target windows.

Out of the 500 images, the algorithm classified 127 images with an AC value for target and captured windows of at least 90% (mean of 92.67%, 197 with a ratio between 80% and 90% (mean of 84.8%), and 138 images with a ratio between 70% and 80% (mean 77.4%). The remaining 78 images had a ratio between 60% and 70%.

Center of Mass The dislocation of the CoM had been evaluated with respect to the window width. Out of 500 images, 107 images had the CoM displaced for 5% of the window width, 218 images had the center of mass displaced for 10\%, and 88 images had the CoM displaced for 15\%, and finally 87 images had a displacement of 20%.

The results have shown that the algorithm can find the exact coordinates of the windows with high accuracy with respect to AC, d_c and CoM. The algorithm will always capture any window in an image. Furthermore, 31% of the images have been segmented with very high accuracy AC>90%, whereas the images with 70%<AC<90% have given very good segmentations of the facade.

With respect to subsequent detection of window corners by using Harris and Foerstner operators, an AC quality of less than 70% for an image in the example database did not lead to good results. Even if the windows were captured, the corners of the bounding boxes were too far away from the window corners since the bounding boxes were too big. For the remaining images (87.85%), the result was good.
The experimental results for CoM have shown that in worst case scenarios, the center of mass of the target window is moving upwards, or north. This occurs in particular in facades of classic style. In such facades, other objects are found just above a window. When the algorithm is trying to fit a box around the window, the extra objects intervene in the calculation of the target coordinates. The results were less satisfactory in images where the windows were at an angle.

Distance between corners The results have shown that in 158 images, distance was between 5 and 15 pixels, whereas in 257 images, the distance was between 15 and 25. The remaining 85 images had the distance over 25 pixels. Furthermore, if one was to measure shape similarity for each rectangular side, that is, to average distances between each of the two corners, and to exclude the images with classical windows, the results would improve greatly, as this distance would never be above 20 pixels. Thus, the results have shown that there is high similarity between the captured and real window in most pictures. For the purposes of our application, the extra distance between the corners of actual and real window is negligent, as in a lot of cases, it is desired to have some extra coverage of the real window, when calculating the Foerstner and Harris operators, in order to get the maximum number of interest points. The results of this methodology have shown that this extra coverage does not produce a lot of unnecessary points, and that all interest points are covered.

5.1 Comparison Study

Haider et al [3] describes an algorithm for window detection which uses machine learning techniques. The algorithm uses images from Vienna and ZuBud public domains. Evaluation is carried out using positive true and false based samples, in the frame of the pattern based detector. Furthermore, the algorithm is evaluated using Single Window (SW), and Window Region of Interest (WROI). For SW evaluation, positive true samples are counted if one detection rectangle would be found inside a mask rectangle or at the maximum, would overlap it only by a few pixels in each direction. For WROI evaluation, positive true samples are counted whenever the mask rectangle is covered by a certain percentage. The algorithm has performed well using these testing methodologies, but it is notable to say that our algorithm would yield 100 \% on WROI and SW if applied, due to the nature of our algorithm. However, the algorithm from Haider et al, works better when there are additional objects present in the picture.

The Hough transform can be used to detect rectangular shapes in an image, but these rectangles would have to be filtered to locate the real coordinates of the window. This method can be used as an addition to our algorithm, but the results would not improve significantly. We have developed a method for window detection in images of facades, which is needed in the context of deformation monitoring and analysis of buildings. There, the challenge is to (re-)identify the corners of windows from images of facades automatically, by the use of interest operators.

6. Conclusions and Future Work

The method which we have developed combines image processing algorithms but also involves new algorithms, which we developed specifically for thresholding of images containing facades, and for segmentation of such an image using bounding boxes, which can then be passed on to window corner detection and deformation

analysis. The experimental evaluation of the method with respect to two quality measures has shown very good results.

There are several directions for enhancing the current method in future work. One would be enabling it to work when there are other objects present on the image, such as cars and trees. The current method cannot perform well in these cases, because the additional objects generate too much noise, and the information is then lost. One possible attempt would be to use machine learning to discard these objects from an image, and then post-process it and forward it to the algorithm. Another direction would be to extend the method from a single image to a time series of images. Here, the results for past epochs may be exploited in order to improve the result. Finally, another direction would be to extend the method to facades under external conditions such as curved buildings, or irregular structure.

References

[1] W. Foerstner and E. Guelch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In ISPRS Intercommission Workshop, Interlaken.
[2] David A. Forsyth and Jean Ponce. Computer Vision: A Modern Approach. Prentice Hall, 2003.
[3] Haider Ali, Nitin Jindal, Lucas Paletta, Gerhard Paar, Christin Seifert. Window detection in facades. 2007.
[4] C. Harris and M. Stephens. A combined corner and edge detection. In Proceedings of The Fourth Alvey Vision Conference, pages 147{151, 1988.
[5] Claudio Rosito Jung and Rodrigo Schramm. Rectangle detection based on a windowed hough transform. In SIBGRAPI, pages 113{120. IEEE Computer Society, 2004.
[6] J.R. Parker. Algorithms for Image Processing and Computer Vision. John Wiley & Sons Inc, 1996.
[7] Remco C. Veltkamp. Shape matching: Similarity measures and algorithms. In Shape Modeling International, pages 188{199. IEEE Computer Society, 2001.
[8] http://info.tuwien.ac.at/ingeo/research/FWF_Projekt_CogniMess/FdbV2006/.
[9] http://www.vision.ee.ethz.ch/showroom/zubud/index.en.html.
[10] Alexander Reiterer, Martin Lehmann, Milos Miljanovic, Haider Ali, Gerhard Paar, Uwe Egly, Thomas Eiter and Heribert Kahmen, A 3D optical deformation measurement system supported by knowledge-based and learning techniques,
[11] Alexander Reiterer, Martin Lehmann, Milos Miljanovic, Haider Ali, Gerhard Paar, Uwe Egly, Thomas Eiter and Heribert Kahmen, Deformation Monitoring using a new kind of optical 3D Measurement System: Components and Perspectives.Lisbon 2008
[12] Lehmann, M., Reiterer, A. (2007) Online-Chrakterisierung von Deformationen in 3DPunktmengen– erste konzeptionelle Ansätze, in: Photogrammetrie Laserscanning Optische 3D-Messtechnik
[13] Reiterer, A. (2004) Knowledge-Based Decision System for an On-Line Videotheodolite-Based Multi-Sensor System, PhD thesis, Vienna University of Technology, 2004.
[14] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In Proceedings of ISPRS Intercommission Workshop, Interlaken, pages 281–305, 1987.
[15] C. Harris and M. Stephens. A combined corner and edge detection. In Proc.of The Fourth Alvey Vision Conference, pages 147–151, 1988.