

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Detection of Barcode using Machin Learning

Examensarbete utfört i Datorseende
vid Tekniska högskolan vid Linköpings universitet
av

Olle Fridolfsson

LiTH-ISY-EX--YY/NNNN--SE

Linköping 2014



Linköpings universitet
TEKNISKA HÖGSKOLAN

Detection of Barcode using Machin Learning

Examensarbete utfört i Datorseende
vid Tekniska högskolan vid Linköpings universitet
av


Olle Fridolfsson

LiTH-ISY-EX--YY/NNNN--SE

Handledare: **Freddie Åström**
ISY, Linköpings universitet
Ola Friman
SICKIVP

Examinator: **Lasse Alfredsson**
ISY, Linköpings universitet

Linköping, 9 april 2014

	Avdelning, Institution Division, Department	Datum Date
	Avdelningen för ditten Department of Electrical Engineering SE-581 83 Linköping	2014-04-09

Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	ISBN _____ ISRN LiTH-ISY-EX--YY/NNNN--SE Serietitel och serienummer Title of series, numbering ISSN _____
URL för elektronisk version http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX		

Titel Title	Maskininlärning för detektion av streckkod Detection of Barcode using Machin Learning
Författare Author	Olle Fridolfsson

Sammanfattning Abstract
<p>Det här som vi har hållit på med är jätteviktigt faktiskt och det vi gjort blev bara sååå bra. Kanske inte helt otippat, men det glass är sååå gott!</p> <p>Förresten har vi blivit bäst på att skriva rapporter, så nu ska ska vi inte gå in närmare på några detaljer såhär i sammanfattningen.</p>

Nyckelord Keywords	problem, lösning
------------------------------	------------------

Sammanfattning

Det här som vi har hållit på med är jätteviktigt faktiskt och det vi gjort blev bara sååå bra. Kanske inte helt otippat, men det glass är sååå gott!

Förresten har vi blivit bäst på att skriva rapporter, så nu ska ska vi inte gå in närmare på några detaljer såhär i sammanfattningen.

Abstract

If your thesis is written in English, the primary abstract would go here while the Swedish abstract would be optional.

Acknowledgments

Vi tycker alla har varit så himla goa hela den här långa och tuffa tiden i våra liv.

Linköping, Januari 2020

NN och MM

Contents

Notation	xi
1 Introduction	1
2 Overview	3
2.1 System overview	3
3 Preprocessing of data	5
4 Machine learning methods	7
4.1 AdaBoost	7
5 Features used to detect barcodes	9
5.1 Standard deviation	9
5.2 Structure tensor	9
5.3 FAST corner detection	10
5.4 Distance map	10
5.5 Local binary pattern	10
6 Cascade	13
7 Evaluation	15
7.1 Calculation of ground truth	15
7.2 Evaluation of features	16
7.2.1 Standard deviation	16
7.2.2 Structure tensor	17
7.2.3 Distance map	17
7.2.4 FAST corner detection	18
7.2.5 Local binary pattern	18
7.3 Evaluation of cascade	19
7.4 Conclusions	22
8 Further investigations	25

Bibliography

27

Notation

NÅGRA MÄNGDER

Notation	Betydelse
\mathbb{N}	Mängden av naturliga tal
\mathbb{R}	Mängden av reella tal
\mathbb{C}	Mängden av komplexa tal

FÖRKORTNINGAR

Förkortning	Betydelse
ARMA	Auto-regressive moving average
PID	Proportional, integral, differential (regulator)

1

Introduction

2

Overview

Text...

2.1 System overview

To increase the speed a good way is to reduce the amount of data. One way to do this is to use a cascade system, where some amount of data is discarded in each step, illustrated in figure 1. This method can be used both during training and testing.

A great starting point when you are new to \LaTeX is to read [1].

There are many interesting things about \LaTeX found in the standard references by Lamport [2] and Gossens et al. [3]. These describe most everything one needs to know about creating documents with \LaTeX 2_ε. Gossens et al. has also written a book dealing with graphics in \LaTeX , mostly Post-Script based, [4]. Of course there exists many other good references to \LaTeX out there too.

2.1 Example: An example of an example

In this example please note that there is a substantial difference between [5] and the first edition of the book [6].

3

Preprocessing of data

A big amount of gray scale images containing different kinds of code will be available. For each image the corresponding ground truth will also be available. One part of the images will be used as training data and the rest will be used as test data. The idea is to divide each image into blocks of same size. The amount of training data

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

will then be the number of blocks in each image times the number of training images. The blocks can either overlap each other or just lay next each other. Overlapped blocks will lead to higher accuracy but more data have to be processed. Each data will consist of a feature vector:

Each feature will in some way describes the corresponding block. The only information that is available is the intensities of the pixels in each block; consequently all features will always, in some way, be based on the pixel values.

4

Machine learning methods

4.1 AdaBoost

The machine learning method that will be tried out first is Boosting which is described in [2]. The basic idea is to train a number of weak classifiers which during the testing will be combined to a strong classifier. To each data in the training dataset there are corresponding weights which are equal for all data at the beginning. The weak classifiers are trained sequentially and after each step the weights are adjusted depending if they were correctly or incorrectly classified.

There exist several variants of Boosting algorithms. The one that will be tried out first is discrete AdaBoost. The weak classifiers in discrete AdaBoost are split functions which simply classifies the data as true or false. The split functions consist of a number of different parameters. The most basic function, which will be tried out first, only has one parameter, a threshold. The function search for a threshold in one dimension at a time and choose the one which best separates the data.

5

Features used to detect barcodes

5.1 Standard deviation

One good method to exclude tiles that don't contain any code is to compute the standard deviation of each tile. Tiles which contain code will have a high standard deviation; hence all data with standard deviation under a certain threshold can be discarded. This fast method to reduce the amount of data before applying other more complicated features.

5.2 Structure tensor

The structure tensor is an effective way to detect 1D-codes and to distinguish between 1D- and 2D-codes. The structure tensor for a tile is produced by first computing the gradients, this can be done by convolving the tile with a sobel filter. The structure tensor is then calculated in the following way:

$$T = \begin{pmatrix} (I_x)^2 & I_x I_y \\ I_x I_y & (I_y)^2 \end{pmatrix}$$

The structure tensor can be used in several ways to estimate the structure inside the tile. Areas that contain 1D-codes will have a one dimensional structure, i.e. the gradient in this area will only vary in one direction. On the other hand, areas that contain 2D-codes the variation will be fairly equal in both directions. The amount of variation of the gradient is measured by studying the eigenvalues of the structure tensor. The following feature is calculated for each tile:

$$f = \frac{(\lambda_1 - \lambda_2)^2}{\lambda_1^2 + \lambda_2^2}$$

This feature is a measure of how likely the tile contains 1D-codes.

5.3 FAST corner detection

FAST (Features from Accelerated Segment Test) is described in [Rosten and Drummond, 2006] and [Rosten and Drummond, 2005]. This is a method to detect corners in images, which is a good way to distinguish between 1D- and 2D-codes. For a pixel p with intensity I , consider a circle of 16 pixel and a threshold t . The pixel is considered to be a corner if there exist a set of n contiguous pixels in the circle which are all brighter than $I+t$ or darker than $I-t$. The algorithm is described in 5.1.

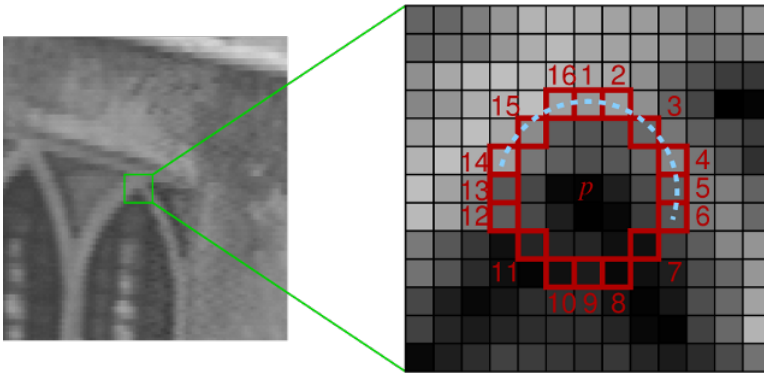


Figure 5.1: Image describing FAST corner detection

5.4 Distance map

A method for barcode detections using a distance map is described in [P. Bodnar]. The objective of this method is to first calculate the edges in the image by using the canny edge detection algorithm. After that a distance transform is used which in each pixel calculates the closest distance to an edge. There after the mean and standard deviation for the distance map are calculated which are then used as features. The distance map has in this system been used only for detection of 1D-codes.

5.5 Local binary pattern

Local binary pattern, which is described in [Pietikainen, 2010], is a method that can be used for detection of 2D-codes. The basic idea is to compute a binary code for every pixel, based on the difference of the intensity between the pixel and the surrounding pixels, illustrated in 5.2. The binary code will then be transformed to a decimal scalar value. If a 3x3 neighborhood is used there will be 256 different

possible values. For each block a histogram will be calculated for all these values. Every bin in the histogram will then be used as a feature. If there is a bin for every possible value, there will be 256 features.

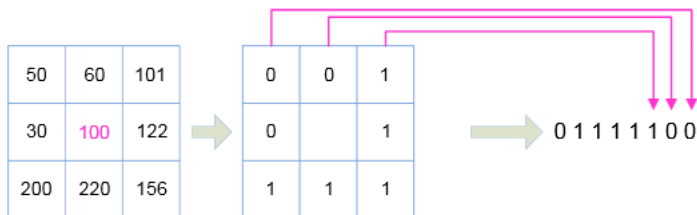


Figure 5.2: Image describing local binary pattern

6

Cascade

7

Evaluation

Here is a compilation of the evaluation that has been made for different methods for detections of barcodes. First some results from the different features are evaluated individually regarding their accuracy. Then the accuracy and the speed of the whole system is evaluated.

To get a measure of the accuracy of the system the number of true detections and the number of false detections are calculated and plotted. There are a lot of parameters and variation of the preprocessing of the tiles that will affect the result:

- Size of the tiles
- Use of overlapping tiles or not
- Use of down sampling
- ϵ from
- Use of Laplace filtering

In the evaluation 265 images has been used containing 1D-codes and 2D-codes. Of them 100 has been used for training and 165 for testing.

7.1 Calculation of ground truth

One of the problems that can occur with too big tiles and when if the down sampling is too high is that the ground truth is not calculated correctly. Since 70% of the tile has to be inside the code area there is a chance some codes will be missed out entirely. This is primarily the case for the 1D-codes since some of them are

very thin. There are not really any good way to evaluate this except pick out some difficult images and look at the result. If the the tile size is too big to detect some of the codes one solution is to make them overlap with each other.

In the table below some tile sizes have been tried out with different down sampling. The values written in the table is how much overlap of the tiles which is necessary to achieve an acceptable result. The places in the table which is empty are cases when it's not possible to get a good result.

tile size	no down sample	down sample 2	down sample 3	down sample 4
24x24	1	1-2	3	
32x32	1	2-3		
48x48	2			
64x64	3			

Table 7.1: Necessary overlap of the tiles to calculate ground truth

7.2 Evaluation of features

Here is an evaluation of each feature that are used in the cascade. The objective is to keep as many true tiles as possible in each step of the cascade and in the same time reduce the amount of data as much as possible. In this evaluation the objective has been to preserve at least 98% of the true tiles in each step. Here different tile sizes and different amount of down sampling have been tried and best value of ϵ has been calculated to preserve 98% of the true tiles and as few false tiles as possible. Here only cases which gave a good result in the evaluation of the ground truth above are tested.

7.2.1 Standard deviation

When using standard deviation the images have been preprocessed with Laplace filtering and no overlapping tiles. It has been trained on both 1D- and 2D-codes.

tile size	no down sample	down sample 2	down sample 3
24x24	4	4	4
32x32	4	4	
48x48	3		
64x64	3		

Table 7.2: Lowest value of ϵ that keeps at least 98% of the true tiles when using standard deviation

tile size	no down sample	down sample 2	down sample 3
24x24	280	12.23	33.6
32x32	96.27	163	
48x48	77		
64x64	69		

Table 7.3: The average number of false detection per image using standard deviation

7.2.2 Structure tensor

When using structure tensor the images have been preprocessed with Laplace filtering and no overlapping tiles. It has been trained only on 1D-codes.

tile size	no down sample	down sample 2	down sample 3
24x24	1.5-2	2.5	bad result
32x32	1	2	
48x48	1.5		
64x64	1		

Table 7.4: Lowest value of ϵ that keeps at least 98% of the true tiles when using structure tensor

tile size	no down sample	down sample 2	down sample 3
24x24	226	121	bad result
32x32	92	228	
48x48	23		
64x64	28		

Table 7.5: The average number of false detection per image using structure tensor

7.2.3 Distance map

The distance map is trained only with 1D-codes and without Laplace filtering.

tile size	no down sample	down sample 2	down sample 3
24x24	1	1.5	2
32x32	1	1.5	
48x48	1		
64x64	1.2		

Table 7.6: Lowest value of ϵ that keeps at least 98% of the true tiles when using distance map

tile size	no down sample	down sample 2	down sample 3
24x24	90	15	71
32x32	35	23	
48x48	45		
64x64	52		

Table 7.7: The average number of false detection per image using distance map

7.2.4 FAST corner detection

The FAST corner detection feature is trained only with 2D-codes and without Laplace filtering. When down sampling with 3 the result seems to drop significantly, then it seems like it detects a lot of corners in the 1D-code.

tile size	no down sample	down sample 2	down sample 3
24x24	1.5	4	3
32x32	3	4	
48x48	2.2		
64x64	3.		

Table 7.8: Lowest value of ϵ that keeps at least 98% of the true tiles when using FAST corner detection

tile size	no down sample	down sample 2	down sample 3
24x24	75	17	138
32x32	59	21	
48x48	80		
64x64	44		

Table 7.9: The average number of false detection per image using FAST corner detection

7.2.5 Local binary pattern

The LBP feature is trained only with 2D-codes and without Laplace filtering. The evaluation was in this case done together with the cascade. The reason for this is that it takes too long to test this features for all data. When using the cascade the data is reduced a lot before the step where the LBP is used. Also the training was done with 50 weak classifiers instead of 100.

tile size	no down sample	down sample 2	down sample 3
24x24		3	
32x32		2	
48x48	2		
64x64	2		

Table 7.10: Lowest value of ϵ that keeps at least 98% of the true tiles when using LBP

7.3 Evaluation of cascade

Here are some conclusions that can be drawn from the evaluation of the features before evaluating the cascade.

- There is no reason to use tiles with sizes 24x24 and 32x32 without down sampling with 2, since all feature seems to work in this case.
- To down sample with more than 2 seems to give really bad results with structure tensor.
- The cascade will be a lot faster with down sampling with 2 and tile sizes 24x24 and 32x32. This is because the amount of false tiles when using standard deviation are very low.

From these conclusions the evaluation of the cascade will be done for four different cases regarding the tile size, the down sampling and the overlap of the tiles. Also the cascades has been evaluated for the two different variants which was described in 6 have been evaluated.

	ms/image	true detections in percent	false tiles per image
tile size 24x24 down sampling 2 overlap 1	75	1D: 98 2D: 95	1D: 0.7 2D: 0.14
tile size 32x32 down sampling 2 overlap 2	190	1D: 98 2D: 93	1D: 0.26 2D: 0
tile size 48x48 down sampling 1 overlap 2	420	1D: 96 2D: 94.5	1D: 0.3 2D: 0.097
tile size 64x64 down sampling 1 overlap 3	631	1D: 0.96 2D: 0.93	1D: 0.35 2D: 0.93

Table 7.11: Result from evaluation of cascade

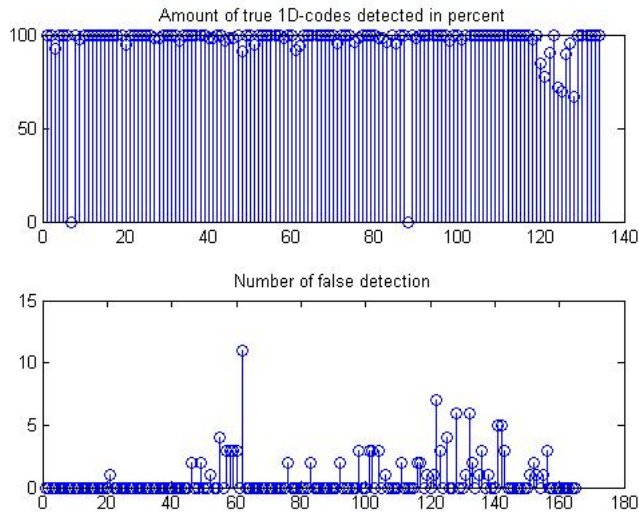


Figure 7.1: Result for 1D-codes from evaluation of cascade using tile size 24x24, down sampling 2, and no overlap

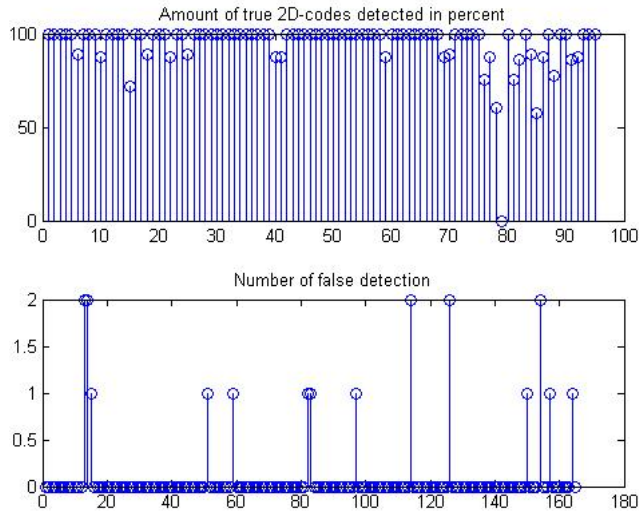


Figure 7.2: Result for 2D-codes from evaluation of cascade using tile size 24x24, down sampling 2, and no overlap

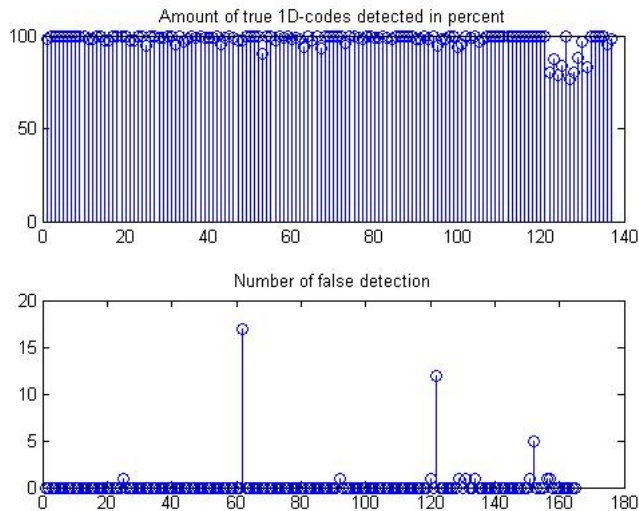


Figure 7.3: Result for 1D-codes from evaluation of cascade using tile size 32x32, down sampling 2, and overlap 2

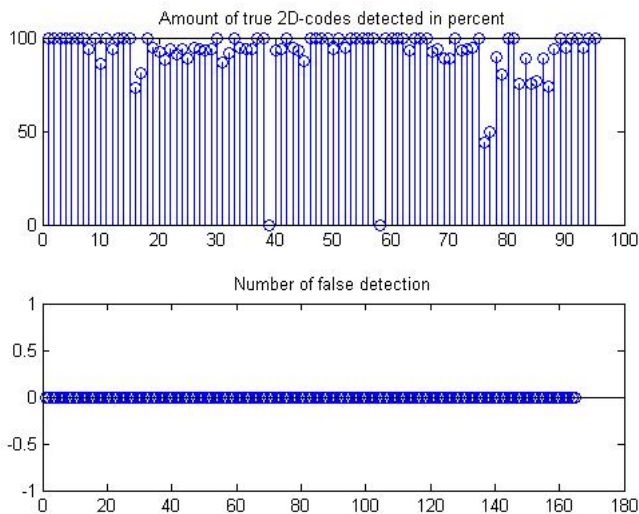


Figure 7.4: Result from evaluation of cascade using tile size 24x24, down sampling 2, and no overlap

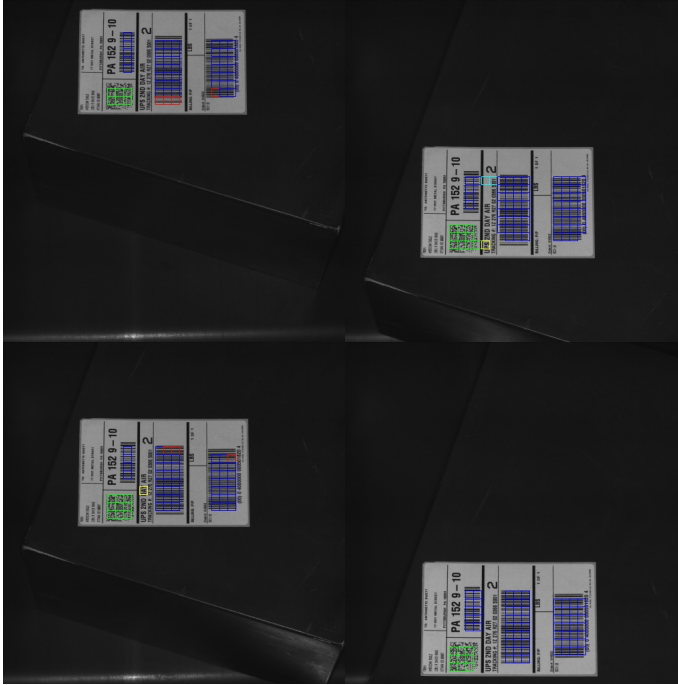


Figure 7.5: Result for 2D-codes from evaluation of cascade using tile size 32x32, down sampling 2, and overlap 2

7.4 Conclusions

The two variants of the cascade gave rather similar results. Both the structure tensor and the FAST corner detection are rather good at distinguish between 1D- and 2D-codes.

The first two cases (the two first rows in table 7.11) gives overall better result, especially regarding the speed. Which one of them that is best depend depend of what is required of the system, regarding speed and accuracy.

One alternative is to omit the distance map feature in the cascade and only use standard deviation and structure tensor for detection of 1D-codes. The result is still rather good, the amount of true detections is even higher but the amount of false tiles will increase.

The amount of true tiles detected can in some cases seem a bit weak, especially for 2D-codes. In the graphs one can see that in some images the there are no detections at all. This is the case when the codes are at the border of the image and only a small part are inside. Since in this case only a few tiles will cover the codes there is a big chance that these will be lost in the post-processing. The alternative is to reduce the amount of post-processing but this will instead lead

to more false tiles.

8

Further investigations

There are a lot of different 2D-codes that are currently used. The system has so far only been tested on Maxicodes. Since other kinds of 2D-codes have similar structure they will presumably give a very similar result. Maxicodes should be more difficult to detect than most other types of 2D-codes since it consists of small circular areas. Most other types of 2D-codes, e.g. QR-codes consist instead of small squares, this should make them even more distinguishable than Maxicodes.

The system could also be tested on a more complicated data set. If one would use a data set that contains more details in the background the system would probably detect a lot more false tiles. However many of the data sets that are available look very similar to the one that has been used.

To speed up the system even further one possibility could be to train a cascade for the local binary pattern alone. Since the LBP has 256 features it is likely that some of these features are more common than others for 2D-codes. If the system is trained to use these features in a certain order it would probably speed up the system. However since the LBP features have been used in the last step of the cascade and the amount of data already has been reduced a lot it is uncertain if this will make much impact.

Postprocessing

Bibliography

- L. Nyul P. Bodnar. Barcode detection with uniform prtitioning and distance transformation. *Univerity of Szeged*. Cited on page 10.
- M. Pietikainen. Local binary pattern. *Scholarpedia*, 2010. Cited on page 10.
- E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. *IEEE International Conference on Computer Vision*, 2005. Cited on page 10.
- E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *European Conference on Computer Vision*, 2006. Cited on page 10.

Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>