

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Automatic configuration of vision sensor

Examensarbete utfört i datorseende
vid Tekniska högskolan vid Linköpings universitet
av

Niklas Ollesson

LiTH-ISY-EX--13/4666--SE

Linköping 2013



Linköpings universitet
TEKNISKA HÖGSKOLAN

Automatic configuration of vision sensor

Examensarbete utfört i datorseende
vid Tekniska högskolan vid Linköpings universitet
av


Niklas Ollesson

LiTH-ISY-EX--13/4666--SE

Handledare: **Johan Hedborg**
ISY, Linköping university
Henrik Turbell
SICK IVP

Examinator: **Klas Nordberg**
ISY, Linköping university

Linköping, 23 april 2013

	Avdelning, Institution Division, Department Computer vision Department of Electrical Engineering SE-581 83 Linköping	Datum Date 2013-04-23
---	---	--

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--13/4666--SE Serietitel och serienummer ISSN Title of series, numbering _____
URL för elektronisk version http://ep.liu.se		

Titel Title	Automatisk konfiguration av smartkamera Automatic configuration of vision sensor
Författare Author	Niklas Olsson

Sammanfattning
Abstract

In factory automation cameras and image processing algorithms can be used to inspect objects. This can decrease the faulty objects that leave the factory and reduce manual labour needed. A vision sensor is a system where camera and image processing is delivered together, and that only needs to be configured for the application that it is to be used for. Thus no programming knowledge is needed for the customer. In this Master's thesis a way to make the configuration of a vision sensor even easier is developed and evaluated.

The idea is that the customer knows his or her product much better than he or she knows image processing. The customer could take images of positive and negative samples of the object that is to be inspected. The algorithm should then, given these images, configure the vision sensor automatically.

The algorithm that is developed to solve this problem is described step by step with examples to illustrate the problems that needed to be solved. Much of the focus is on how to compare two configurations to each other, in order to find the best one. The resulting configuration from the algorithm is then evaluated with respect to types of applications, computation time and representativeness of the input images.

Nyckelord Keywords	image processing, vision sensor, factory automation, automatic configuration, multidimensional optimization
------------------------------	---

Abstract

In factory automation cameras and image processing algorithms can be used to inspect objects. This can decrease the faulty objects that leave the factory and reduce manual labour needed. A vision sensor is a system where camera and image processing is delivered together, and that only needs to be configured for the application that it is to be used for. Thus no programming knowledge is needed for the customer. In this Master's thesis a way to make the configuration of a vision sensor even easier is developed and evaluated.

The idea is that the customer knows his or her product much better than he or she knows image processing. The customer could take images of positive and negative samples of the object that is to be inspected. The algorithm should then, given these images, configure the vision sensor automatically.

The algorithm that is developed to solve this problem is described step by step with examples to illustrate the problems that needed to be solved. Much of the focus is on how to compare two configurations to each other, in order to find the best one. The resulting configuration from the algorithm is then evaluated with respect to types of applications, computation time and representativeness of the input images.

Acknowledgments

First of all I would like to thank SICK IVP for supplying the idea for the Master's thesis and an office where I could work. I would also like to thank my supervisor at SICK IVP, Henrik Turbell, for support and ideas on solutions for the algorithm development and help with the creative process of gathering as many sample image sets as possible.

I would also like to thank my supervisor, Johan Hedborg, and examiner, Klas Nordberg, for their support in planning and input on possible solutions for the algorithm development.

Linköping, April 2013
Niklas Ollesson

Contents

Notation	ix
1 Introduction	1
1.1 The Inspector	2
1.2 Problem formulation	2
1.2.1 Configuration	4
1.2.2 Evaluation	4
1.3 Outline	4
2 Background	5
2.1 Object locator	5
2.1.1 Inputs	5
2.1.2 Outputs	6
2.1.3 Score threshold	6
2.2 Detailed inspections	6
2.2.1 Pattern inspection	6
2.2.2 Pixel counter inspection	7
2.2.3 Edge pixel counter inspection	7
2.2.4 Blob counter inspection	7
2.3 Restrictions	8
3 Algorithm	9
3.1 Example images	9
3.2 Overall strategy	11
3.3 Object locator	11
3.3.1 Settings	11
3.3.2 Threshold	11
3.3.3 Marking classified images	12
3.4 Image preprocessing	12
3.4.1 Forbidden areas	13
3.4.2 Flowchart	13
3.5 Detailed inspections	14
3.5.1 Summary of detailed inspection algorithm	16

3.5.2	Simplifications	16
3.5.3	Variables	17
3.5.4	Comparing detailed inspections	17
3.5.5	Classification threshold	22
3.5.6	Search	23
3.5.7	Flowchart	25
3.5.8	Robust scoring	25
4	Evaluation	29
4.1	Data	29
4.2	Evaluation algorithm	35
4.2.1	Data labelling	35
4.2.2	Image subsets	36
4.3	Evaluation results	36
4.3.1	Application types	37
4.3.2	Computation time	39
4.3.3	Representativeness of training images	39
4.3.4	Rotation symmetry	41
5	Conclusions	43
5.1	Applications	43
5.2	Computation time	44
5.3	Future development	44
	Bibliography	47

Notation

ABBREVIATIONS

Abbreviation	Meaning
NCC	Normalized cross-correlation
NPV	Negative predictive value
PPV	Positive predictive value
ROI	Region of interest

1

Introduction

In modern days computers and cameras can often replace the need of manual inspection in factories. The help of machine vision inspections of products for example on conveyor belts can increase production speed and reduce the amount of labour needed.

SICK IVP develops camera systems with embedded image processing. The Inspector product is such a camera with focus on ease-of-use. The functionality is rather limited; it acquires a grayscale VGA image of an object and compares it with a reference image. The processing is done in two steps: first the object is found in the image, then smaller regions of interest (ROI) are inspected relative the found pose of the object. Currently, the Inspector needs to be manually configured before use. The configuration is done by taking a reference image of the object to be inspected, marking the ROIs on the image and setting some image processing parameters. The configuration requires no programming knowledge, but some basic understanding of image processing.

With the focus of the Inspector camera being the ease-of-use, a relevant question to ask is if using it can be made even easier. Many customers have no knowledge in image processing, and a way of configuring it without needing this knowledge would be an improvement to the product. The goal of this thesis is to investigate one way that this could be done.

The idea is that the customer knows his or her product much better than he or she knows image processing. The customer could take images of objects that he or she wants classified as positive and negative respectively, and feed these to the Inspector. If the Inspector could then find the configuration automatically from these manually classified images, there would be no need for image processing knowledge to configure it. A solution like this would take the Inspector one big

step forward when it comes to simplicity.

The configuration derived will give a result on the same form that a manually configured Inspector camera would have. The result is thus easy to examine visually, which could be very useful if one would want to adjust it slightly for some reason. It is also useful if the system does not behave as wanted, and one wants to understand why and how this could be helped.

1.1 The Inspector

The work in this Master's thesis is focused on the Inspector version named I40, where I stands for inspection. When configured the Inspector I40 takes images and outputs one of the three results "pass", "fail" or "object not found". Example applications for the Inspector I40 includes:

- Check that a date code is present
- Check that an electrical component is mounted
- Check that the correct number of holes are drilled
- Check that an object is not upside down
- Check that the flaps of a box are correctly folded
- Check that there is an almond in each chocolate praline

The Inspector is configured by the user through a user interface, where the configuration is done in the following steps:

- Capture an image of the object to be processed, to be used as reference image
- Mark the areas in the reference image that show the object to be processed
- Choose parameters for the object locator, as listed in Table 1.1
- Mark areas where detailed inspection is to be performed
- For each area to inspect, choose a detailed inspection method and parameters for this. Possible methods with corresponding parameters are listed in Table 1.1.

A screen capture from the user interface is presented in Figure 1.1.

1.2 Problem formulation

The problem formulation for this thesis has two main parts:

- Given a number of manually classified images, automatically choose a configuration for the Inspector using its existing toolbox.

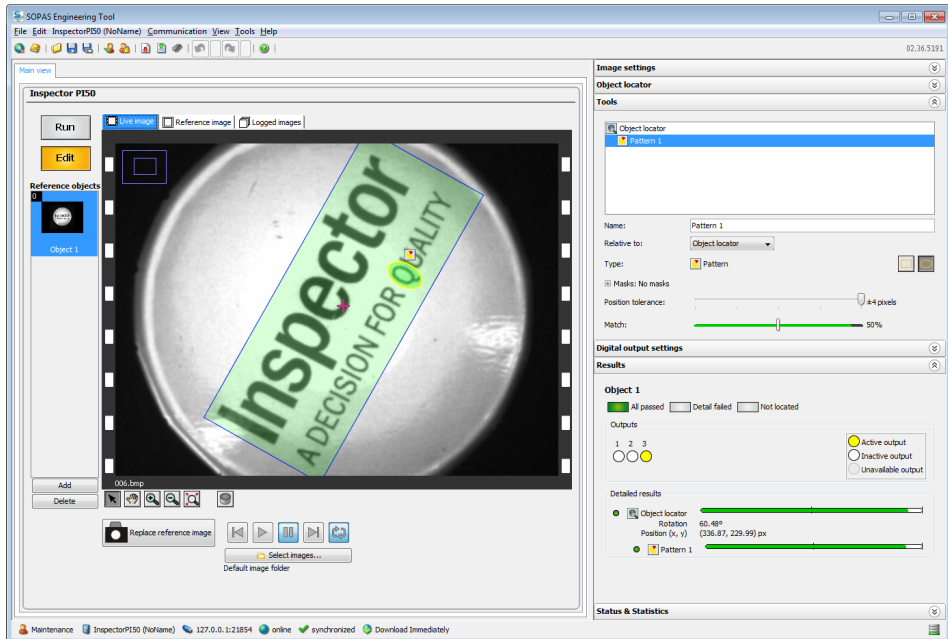


Figure 1.1: Inspector I40 user interface for configuration

Tool	Image processing	Parameters	Example usage
Object locator	Edge-based pattern matching invariant to scale, rotation and translation.	<ul style="list-style-type: none"> - Search area - Score threshold - Maximum rotation - Allow scaled objects - ... 	Locating the reference object in an image.
Pattern inspection	Correlation based comparison with the reference image.	<ul style="list-style-type: none"> - Search range - Correlation threshold 	Check the printing quality of a logo.
Pixel counter inspection	Counts the number of pixels with intensity above a fixed threshold.	<ul style="list-style-type: none"> - Grayscale threshold - Counting threshold 	Check that a ROI has correct intensity.
Edge pixel counter inspection	Counts the number of edge pixels above a fixed threshold.	<ul style="list-style-type: none"> - Edge strength threshold - Counting threshold 	Check if a ROI contains a scratch or not.
Blob inspection	Segmentation on a binarized image.	<ul style="list-style-type: none"> - Binarization threshold - Min/max area - Min/max number of blobs 	Counting the number of items.

Table 1.1: List of available tools for the Inspector

- Evaluate the result to show how well it works on image sets not used when choosing the configuration.

1.2.1 Configuration

A learning algorithm should be developed, that can automatically find a configuration for the Inspector. The input to the algorithm should be images of the object, henceforth called training images, that the user has manually classified as positive or negative. The configuration should consist of object locator parameters and a set of detailed inspections with parameters. Only parameters and kinds of detailed inspections currently available on the Inspector can be used.

1.2.2 Evaluation

The evaluation should be done in a way that the following questions can be answered:

- What kinds of applications the algorithm can handle
- How changing allowed computation time affects the algorithm performance
- How the representativeness of the training images affects the algorithm performance

An example of image representativeness is illumination conditions. In this case the evaluation should show how well the algorithm handles that there are differences in illumination between the training images.

1.3 Outline

Chapter two describes the parts of the Inspector image processing algorithms that are needed to understand this thesis. It also lists the restrictions that are set on the applications and image sets to be used in this thesis.

Chapter three describes the algorithm developed, how it works and what choices were made during development.

Chapter four describes how the evaluation of the configuration algorithm has been done, and presents the evaluation results.

Chapter five presents the conclusions about the work done in this thesis, based on the developed algorithm and the evaluation of this.

2

Background

This chapter describes the details of the Inspector I40 algorithms that are needed to understand the decisions made during algorithm development and evaluation. It also lists the restrictions chosen on applications and input images, to make the problem manageable during the time given for a Master's thesis.

2.1 Object locator

The object locator of the Inspector is used to find an object in the image to be inspected, called live image. To do this it uses a reference image of the object. The object is found independent of rotation and translation, and with up to 20% change in scale. The object locator works with edge images, and is thus dependent on that the edges of the object in the live and reference image match. The inputs to and outputs from the object locator are described in this section.

2.1.1 Inputs

There are many parameters that can be set for the object locator algorithm, where all are not of interest to this Master's thesis. Those of interest are:

- Reference image, grayscale image of the object
- ROI marking what parts of the reference image should be used to locate the object
- Edge strength, value between 0 and 100 corresponding to how sharp edges should be, to be used in the matching
- Allowed rotations, 0 to $\pm 180^\circ$ rotation allowed between reference and live image

- Allowed scale, none or up to 20% scale difference allowed between reference and live image

The parameters that are left out mainly concern the trade-off between live computation time and robustness, and will be set to favour high robustness during this Master's thesis, see restrictions in Section 2.3.

2.1.2 Outputs

The object locator outputs the best match that it finds. Each match contains the following information:

- Match rotation, translation and scale between reference and live images
- Score, value 0 to 100 on how well the edges of the reference and live image match, where a higher score corresponds to a better match

Rotation symmetries

If the object is almost rotation symmetric, the object locator will find more than one match with similar score values, and pick the best one. As a result, the given rotation found can be any of the possible rotations. Details that might be of interest are therefore not guaranteed to be found with the same rotation.

2.1.3 Score threshold

A score threshold can be set for the object locator. When running the Inspector on live images, any image where the object locator score is lower than this threshold will be classified as negative. Images with a score higher than the threshold will be further processed by the detailed inspections configured.

2.2 Detailed inspections

In this Master's thesis detailed inspections of types pattern, pixel counter and edge pixel counter will be used, see restrictions listed in Section 2.3. These all consist of a ROI relative to the object found by the object locator. The ROI is defined as a rotated ellipse or rectangle, optionally masked with additional ellipses and rectangles.

The Inspector can handle multiple detailed inspections. If more than one detailed inspection is used the Inspector will output "pass" if and only if all detailed inspections pass. The detailed inspections require that all of the ROI specified for the inspection are inside the image. If parts of any ROI are outside the image, the inspection will return "fail".

2.2.1 Pattern inspection

The pattern inspection tool performs normalized cross-correlation (NCC) between a ROI in the reference image and corresponding ROI in the live image. The pattern matching performs a search for the highest NCC in a small area in the live image.

This area is set to four pixels in each direction. Using NCC makes the inspection independent of intensity differences in-between the reference and live image, as described in [1].

The parameter that need to be set for a pattern inspection is the correlation threshold. Images where the correlation return less than this threshold will be classified as negative.

2.2.2 Pixel counter inspection

The pixel counter inspection counts the number of pixels in a chosen ROI that has an intensity within a chosen range. The parameters that need to be set for the pixel counter inspection are:

- Grayscale range, lower and upper bound on what intensity values should be counted
- Pixel count threshold range, lower and upper bound for pixel count to give a positive result. Images with pixel counts outside of this range will be classified as negative.

2.2.3 Edge pixel counter inspection

The edge pixel counter inspection counts the number of edge pixels within the chosen ROI in the live image. The edges are found using Canny edge detection. More information on Canny edge detection can be found in [1]. The parameters that need to be chosen for the edge pixel counter are:

- Edge strength. Value in the range 0 to 100 that corresponds to the Sobel thresholds used in the Canny edge detection.
- Edge pixel count threshold range, allowed range for pixel count to give a positive result. Images with edge pixel counts outside of this range will be classified as negative.

2.2.4 Blob counter inspection

The blob counter inspection first creates a binary image from the original image by setting pixels with an intensity within a given range to one, and the others to zero. It then finds connected areas in the binary image, and counts how many of these are within a given area range. The parameters that need to be set for the blob counter inspection are:

- Grayscale range, lower and upper bound on what intensity values should be used when creating the binary image
- Blob area range, lower and upper bound on size of blobs to be counted.

2.3 Restrictions

This section covers the restrictions that have been set on the problem to make it possible to solve within the time of a Master's thesis. The restrictions were chosen after discussions with the supervisor, the product owners and employees at SICK IVP that have experience in configuring the Inspector for customers. The restrictions are listed in order of importance for someone who would want to implement and use the configuration algorithm.

Inspector I40 restrictions

The configuration algorithm will require that the classification between images can be done using the Inspector I40 tool set. This means that applications where someone experienced in configuring the Inspector can not find a working configuration manually, will not be handled by the algorithm either.

Representativeness of negative images

The negative images are assumed to be representative for all negative images. That is, only faults on the object that appear on any of the negative training images will be handled by the configuration. All kinds of faults also have to appear on their own in at least one of the images. Thus the configuration algorithm is restricted to applications where negative examples exists or can be created.

Difference between positive images

The object locator needs to be able to find the object in all positive images using a ROI that is set to the entire image. That is, applications where a more detailed ROI for the object locator would be needed will not be handled by the algorithm.

Detailed inspection shapes

The algorithm will only use detailed inspections consisting of rotated rectangles and ellipses without masks. That is, applications that can not be solved without masks on the rectangles and ellipses will not be solved by the algorithm.

Blob inspection tool

The blob inspection tool will not be considered when choosing configuration. That is, applications where the blob inspection tool is needed to solve the problem will not be handled by the algorithm.

Live computation time

Depending on how the Inspector is configured, the computation time needed to classify one live image will differ. This computation time will not be considered when choosing configuration. Where choices can be made, accuracy will be preferred over live computation time.

3

Algorithm

This chapter describes the algorithm developed in this Master's thesis. It also describes the steps that are taken and the parameters that need to be set for all steps.

3.1 Example images

To describe how the algorithm works, four example image sets will be used. These are chosen because they represent the problems that had to be solved during development in a good way. The image sets are ordered by difficulty, starting with the easiest.

The first image set is shown in Figure 3.1. This is a simple application where the object locator can separate the images on its own, or with one ROI of either type.

The second image set is shown in Figure 3.2, where positive images are ace of spades and negative images are ace of clubs. Here the shadow under the cards falls differently between images, thus changing where edges are found for the ob-

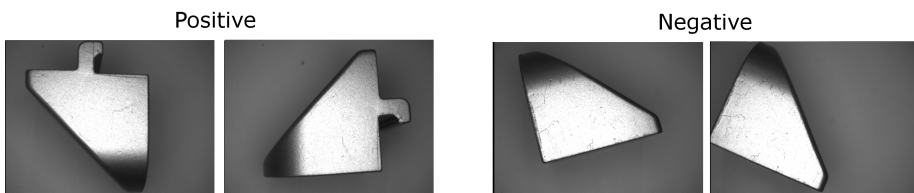


Figure 3.1: Examples of images in application

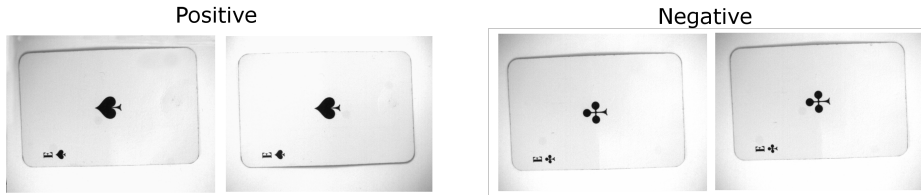


Figure 3.2: Examples of images in application with playing cards



Figure 3.3: Examples of images in application with date stamps

ject locator and making its score behave unreliably. One ROI around the symbol in the middle or corner solves the application.

The third image set is of the lid of yoghurt bottles. Here positive images have complete date stamps somewhere on the lid, and negative image have half-written date stamps or no date stamps at all. This application can not be solved with the object locator alone, but requires one detailed inspection of pixel or edge-pixel type. Examples of this image set is shown in Figure 3.3.

Examples of the fourth image set is shown in Figure 3.4, where positive images have all their light bulbs in place, and negative images miss one or more light bulb. This application can not be solved by the object locator only, and more than one ROI is needed. If this application was to be manually configured the intuitive choice would be to use one ROI per light bulb that can be missing. The ROI could be of either type, but edge pixel counting turns out to give the best result.

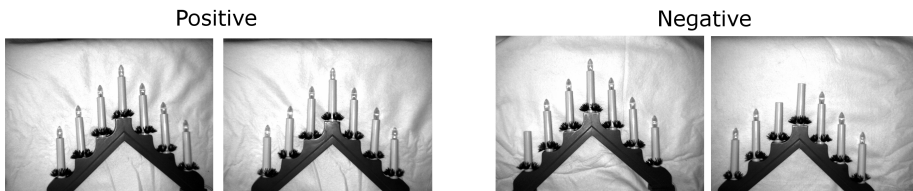


Figure 3.4: Examples of images in application with lamps

3.2 Overall strategy

The configuration algorithm is based on the following reasoning: An object will be classified as positive only if it passes the object locator and all the detailed inspections. An object will be classified as negative if it fails either the object locator or any of the detailed inspections. The goal of each step in the configuration will therefore be to find ways to classify a number of negative images correctly without classifying any of the positive images wrongly.

3.3 Object locator

The first part of the algorithm is to configure the object locator.

3.3.1 Settings

From the restrictions set in Section 2.3, the object locator can be configured with a ROI set to the entire image. With the parameters for the object locator chosen as

- Reference image - the first positive image
- ROI - full image without masks
- Edge strength - 50
- Rotation - $\pm 180^\circ$
- Scale search - on

all positive images in all data sets are correctly matched. This is all that is needed of the object locator, and thus the parameters are left fixed.

3.3.2 Threshold

The object locator gives a score on how good the found match was. This score is based on how similar the images are around the edges found. Given the score for all the images, a threshold value can be chosen to possibly filter out negative images.

The object locator scores for example image sets 1 and 3, together with a possible threshold chosen, is shown in Figures 3.5 and 3.6. In both figures the blue circles represent positive scores and the red crosses represent negative scores. The threshold chosen for the object locator is marked with a vertical solid black line. It can be seen that for example image set 1 it is possible to separate all the positive and negative images using a threshold. For example image set 3 however, no threshold that correctly classifies any negative and all positive images exist.

The images that the object locator score can classify are often simple to find a good detailed inspection in. The object locator score will therefore be used to classify only the images where the object was not found as negative. The negative

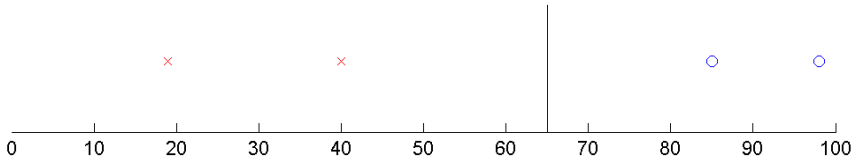


Figure 3.5: Object locator score spread for example image set 1

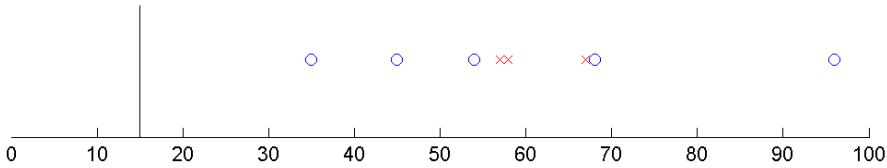


Figure 3.6: Object locator score spread for example image set 3

images where the object locator finds the object are classified by adding a detailed inspection.

With this reasoning, the threshold should be chosen so that images where the object is not found get a score below it. It should also be chosen so that no positive images get a score below it, and therefore gets wrongly classified. The threshold value is chosen based on experience on how the object locator score works, gained during algorithm development. It has been chosen as 20, that is all images that get a score below 20 will be classified as negative. This threshold satisfies the above reasoning for all images sets used for algorithm development.

3.3.3 Marking classified images

Only the images where the object locator could find the object will be used when configuring detailed inspections. Because of this any images that are below the threshold set for the object locator should be marked, so that they are not used for detailed inspection configuration.

If all negative images have been marked, nothing more needs to be done. This is the case when there is a big enough difference between the positive and negative training images, so that the object is not found in any negative image. In this case no detailed inspections will be configured, as the object locator is assumed to be able to correctly classify images.

3.4 Image preprocessing

The object locator gives a transformation that describes how the reference image was transformed in order to match the live image. Given that the object is

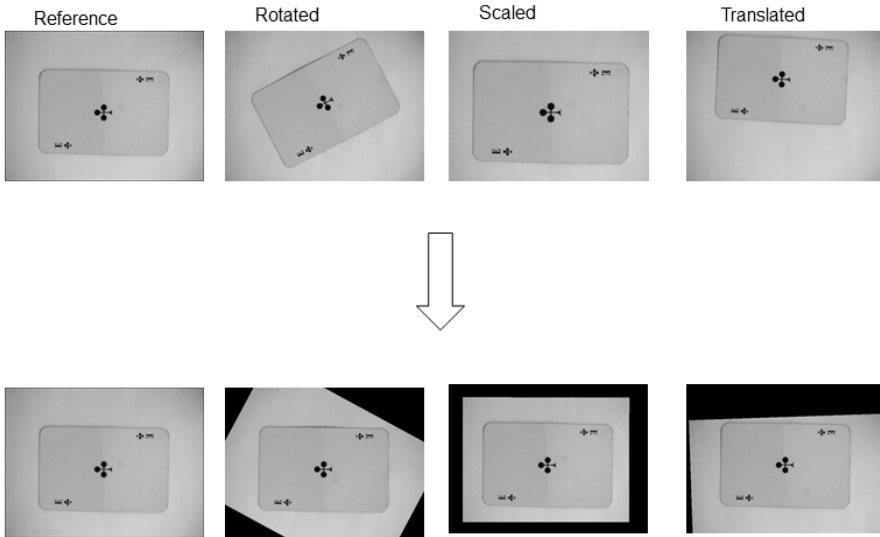


Figure 3.7: Transformation example. Black areas show what parts are outside the live image.

found in all images, the images can be resampled so that the object is in the same position, rotation and scale as in the reference image. An example of this transformation for the cases of pure translation, rotation and scale is shown in Figure 3.7, where the black areas represent parts of the image that were outside before resampling.

3.4.1 Forbidden areas

Given that a detailed inspection can never have any part outside the image, the areas that were outside any image where the object was found before transformation, can not have detailed inspections in them. This area will be referred to as the forbidden area, and used later when detailed inspections are chosen. An example of this is shown in Figure 3.8, where the forbidden area is marked as black.

3.4.2 Flowchart

Figure 3.9 shows a flowchart for setting up the object locator and transforming the images. Teaching the object locator and finding a match in all images is done using code given from SICK IVP. The teaching part is done by giving the settings from Section 3.3.1 to the object locator. The matching is done by giving the image where the object is to be found to the object locator. When this has been done a threshold is chosen for the object locator score, according to Section 3.3.2, and negative images below this score are marked.

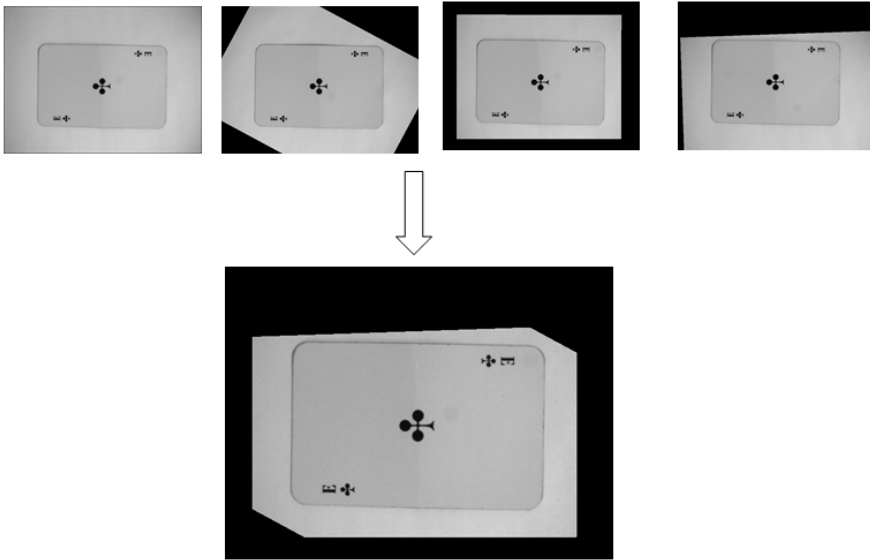


Figure 3.8: *Forbidden areas example. No detailed inspections can be in the black areas of the image*

If all negative images have been marked, nothing else needs to be done and the configuration is finished. If there are still unmarked negative images left, these will be transformed according to Figure 3.7, and detailed inspections will be configured.

3.5 Detailed inspections

Figure 3.10 shows a manually chosen ROI in image set 3. It also shows the values for all types of detailed inspections, with manually well chosen binary and edge strength threshold values. Note the difference in scale on the x-axes in the figures. Values for positive images are marked with blue circles and values for negative images with red crosses. It is clear that for a detailed inspection of type pixel and edge pixel counter, it is possible to chose a threshold that classifies values for positive images from values for negative images. An example of such a threshold is shown as a black dashed line in the figure. For a detailed inspection of pattern type however, this is not possible. This is to be expected since the text is different and in different places in different positive images, and thus the pattern is not the same.

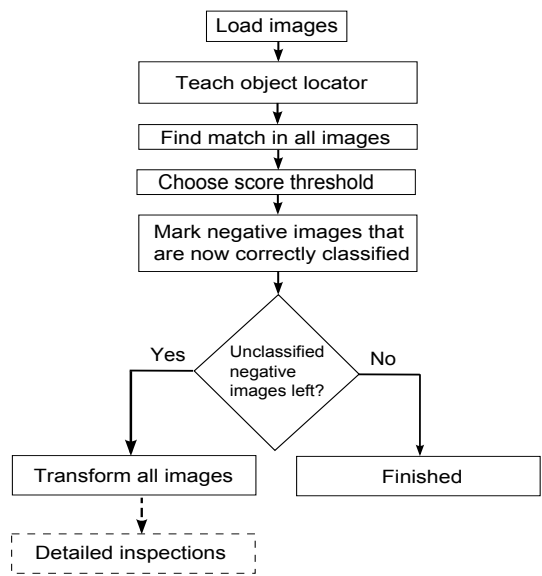


Figure 3.9: Flowchart for object locator and image transformation

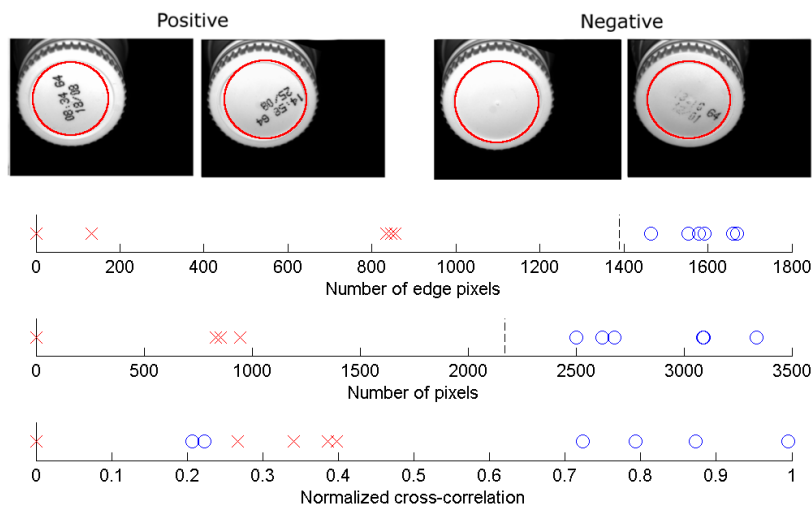


Figure 3.10: Manually chosen region and thresholds with corresponding plots showing values for respective ROI type

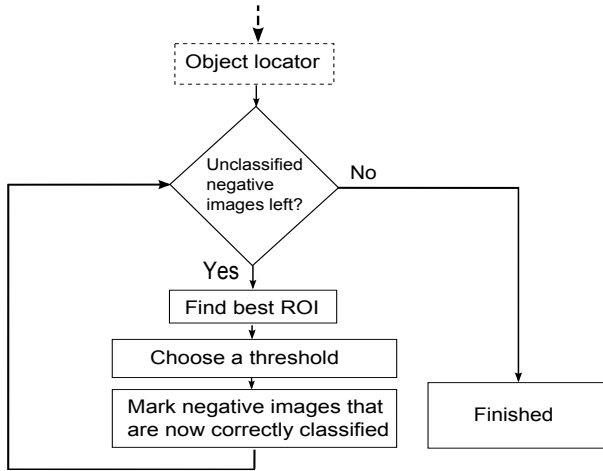


Figure 3.11: Flowchart for detailed inspections

3.5.1 Summary of detailed inspection algorithm

Figure 3.11 shows a flowchart of how the detailed inspections are chosen. The goal is to find the best ROI in the image, mark the negative images that have been classified and then if needed find a new ROI for the remaining images. The problem is split into two parts; first determining how to tell if one detailed inspection is better than another, and then finding the best one.

3.5.2 Simplifications

Some simplifications to the variables that need to be found are made before trying to find the detailed inspections.

Classification threshold

The classification threshold for the pixel and edge pixel counter tool can be set as a range with lower and upper bound. This range will be set as over or under a value instead. This is enough for any of the image sets that have been used for algorithm development, and does reduce the number of parameters per detailed inspection by one. This in turn reduces the computation time needed when searching for detailed inspections, as described in Section 3.5.6.

If an application should require pixels to be within a range in the middle of the interval, the algorithm is assumed to find two detailed inspections at the same place. One of these can then filter negative values below a value and the other negative values above another, thus creating a range.

Binary threshold

The binary threshold for the pixel counter can also be set as a range. In the same way as with the classification threshold, this is set as over or under a value. This

is enough for any of the image sets that have been used for algorithm development, and as for the classification threshold, reduces the parameters per detailed inspection by one.

If an application where an area can not be too dark or too light should exist, the algorithm is assumed to find two detailed inspections at the same place. One detailed inspection can correctly classify negative images that are too dark and one those that are too light.

To avoid confusion and unnecessary computations one can reason about duplicate detailed inspections for the pixel counter tool. Counting pixels under a intensity threshold and setting positive images as over a pixel count, or counting pixels over the intensity threshold and settings positive values under the pixel count gives the same result. That is there are detailed inspections with different parameters that will always give the same classification result. This will lead to unnecessary amount of detailed inspections to search through, and thus unnecessary computations.

This is handled by not including detailed inspections counting pixels over an intensity threshold. This removes the duplicate detailed inspections.

3.5.3 Variables

Each detailed inspection consists of a type, a ROI and a threshold value. Considering finding the optimal ROI of a specific type as a search problem the unknown variables are:

- Center position on horizontal axis
- Center position on vertical axis
- Width
- Height
- Orientation
- Binary / Edge strength threshold (Pixel counter / Edge pixel counter only)

This means that there are six unknown variables that need to be found for each detailed inspection of type pixel and edge-pixel counter, and five unknown variables for each detailed inspection of type pattern.

3.5.4 Comparing detailed inspections

Finding a way of comparing detailed inspections with respect to how well they separate positive and negative images is crucial to finding a configuration algorithm. This will be done starting out from the Mahalanobis distance, which is a way to measure if a data point belongs to a data set [3]. When developing, problems with this measure were noticed, and solutions for these were created based on observations of the algorithm results. That is everything in the detailed inspec-

tion comparison, except for the Mahalanobis distance, is based on observations made during development and not on published methods.

When describing the comparison used it will be assumed that the detailed inspections to be compared have values calculated for each image. This value is the number of pixels/edge pixels for the pixel/edge pixel counter inspections and the NCC for the pattern inspection. The following notation will be used:

- v_{pos} : Set of values for all positive images for a detailed inspection
- v_{neg} : Set of values for all negative images for a detailed inspection
- $v_{neg,i}$: Values for negative image i for a detailed inspection. Sum over all i represent a sum over all negative images.
- μ_{pos} : Mean of all values for positive images
- σ_{pos} : Standard deviation of all values for positive images

Mahalanobis distance

The Mahalanobis distance in one dimension between a value x and a set of data with mean μ and standard deviation σ is defined as

$$d_i = \frac{|\mu - x|}{\sigma}. \quad (3.1)$$

Using the Mahalanobis distance to measure the distance of a value for a negative image to the set of values for positive images, with the above notation, gives

$$d_i = \frac{|\mu_{pos} - v_{neg,i}|}{\sigma_{pos}}. \quad (3.2)$$

Summing these values up as

$$s = \sum_i d_i \quad (3.3)$$

gives a value on how well the detailed inspections would work as a classifier. This way of setting score does have some drawbacks though, and modifications are needed.

Small positive spread

Using the Mahalanobis distance, if all positive images give the same or close to the same value, the standard deviation goes to zero and the distance measurement blows up. This is solved by adding a constant to the standard deviation, thus eliminating the cases where it the distance blows up. The resulting measure of d_i is

$$d_i = \frac{|\mu_{pos} - v_{neg,i}|}{\sigma_{pos} + c}. \quad (3.4)$$

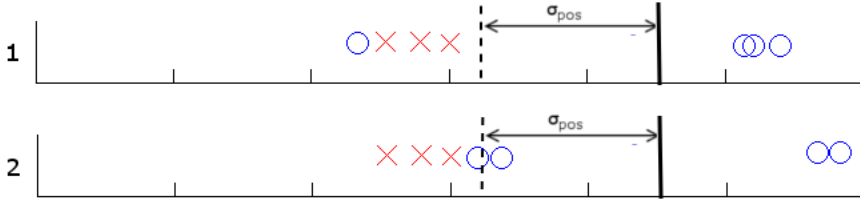


Figure 3.12: Example of when problem with Mahalanobis distances. The solid black line marks the mean value of the positive images, and the dashed black line marks one standard deviation from the mean.

Unclassifiable images

One drawback of the Mahalanobis distance for this application is that it measures from the mean of the positive values, and thus gives good distances to images that may not be possible to classify. This is shown for two cases in Figure 3.12, where the mean value of the positive images is shown as a solid black line, and one standard deviation from the mean as a dashed black line. Using the Mahalanobis distance as it is will give the same score to these detailed inspections, as μ_{pos} , σ_{pos} and all v_{neg} are the same. For detailed inspection 1, no classification threshold can be chosen that can separate positive and negative images. For detailed inspection 2, however, this is possible. This shows that the Mahalanobis distance as it is does not give the desired results.

The mean value in the calculation is therefore changed to the maximum or minimum value, depending on whether threshold is set over or under the positive values. How this decision is made is described later on in Section 3.5.5. The absolute value is also removed, so that images that are not possible to classify are given negative distances.

This measure of distance can be written as

$$d_i = \frac{\min(v_{pos}) - v_{neg,i}}{\sigma_{pos} + c} \quad (3.5)$$

for the case when classification threshold is set as under positive values, and

$$d_i = \frac{v_{neg,i} - \max(v_{pos})}{\sigma_{pos} + c} \quad (3.6)$$

for the case when classification threshold is set as over positive values. Section 3.5.5 describes how the choice of threshold direction is made.

Distance weights

One unwanted property of this measure is that it favours ROI that can classify one image very well, even if the other images are poorly classified. Also, with unclassifiable images getting bigger negative scores the further from classifiable

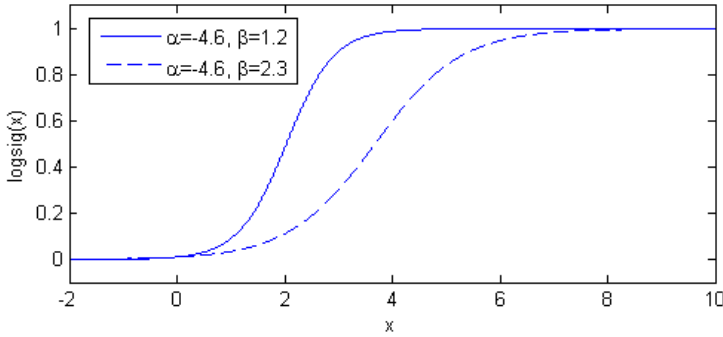


Figure 3.13: Two log-sigmoid functions with different slope

they are, it will favour moving these away. The goal is to get all images well enough separated, and not caring much about images that are not classifiable. To get a measure where score increases faster for negatives closer to the positives than those far away, a log-sigmoid function is used to get ROI score as

$$s = \sum_i \text{logsigmoid}(d_i). \quad (3.7)$$

The log-sigmoid function is defined as

$$\text{logsigmoid}(x) = \frac{1}{1 + e^{-\beta * x - \alpha}}. \quad (3.8)$$

It has two parameters that can be tuned to give it different shape. The choice of these two parameters will affect how the scoring is done. Two examples of log-sigmoid functions with different slope are given in Figure 3.13. One wants the slope of the log-sigmoid-function to be in the area where changes in distance measure should be weighted the highest. The log-sigmoid function with the steep slope in Figure 3.13 is used for the algorithm. This choice was made by trying different parameters and running the algorithm on the image sets used for algorithm development. A lot of different parameters were tried and this gave the best results.

An example of how the scoring works for the date stamp example is shown in Figure 3.14. It shows how the values are recalculated into distances, and how these distances are recalculated to a score using the log-sigmoid function. As can be seen, the pixel and edge pixel counter gets scores close to one for all negative images. The pattern matching however gets close to zero score for all negative images, which is what is expected.

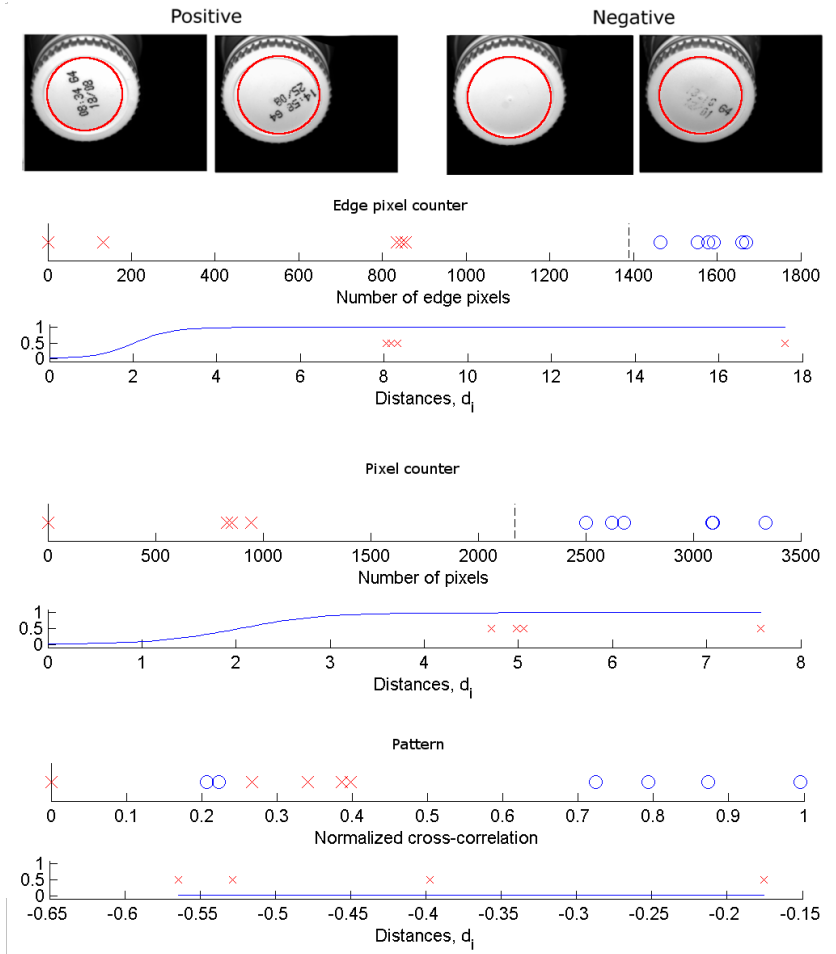


Figure 3.14: Manually chosen region and thresholds with corresponding plots showing values for respective type, the resulting distances d_i and the log-sigmoid function for scoring

Area normalization

An alternative to the normalization described above was also tried during development. This normalization was done as

$$d_{i, relative} = \frac{v_{neg,i} - \max(v_{pos})}{A} \quad (3.9)$$

where A is the ROI area for pixel and edge pixel counter type. Detailed inspections of pattern type were not altered. The main problem with this kind of scoring is that it favours small ROI over big ones, even when big ones would be preferred. Because of this problem the normalization with standard deviation was chosen instead.

3.5.5 Classification threshold

When the ROI position, width, orientation and binary / edge threshold is chosen, a classification threshold for the ROI value needs to be set. This threshold should be set with some safety distance to all positive values, and the negative values that this ROI classifies.

The threshold is chosen in two steps. First a decision is made on which negative images are to be correctly classified. This is done by deciding that it only classifies negative images with a big enough d_i value, that is the d_i value is bigger than a constant. A small value on this constant will let more negative images be classified and thus require less amount of detailed inspections to be chosen. The margin between threshold and values for training images will be smaller though, and assuming that the training images are not entirely representative for all possible images possibly less robust. Choosing a bigger value for this constant will make each detailed inspection classify less negative images. This will give bigger robustness for each ROI, but could lead to fitting unnecessarily many ROI to the data.

In the special case that no ROI is found that can classify any new negative image with a big enough d_i value, this limit is removed. The ROI is instead set to classify only the image with the highest d_i value.

The threshold is then chosen by taking the middle point between the closest positive and negative values. This gives a classification threshold T as

$$T = \frac{\min(v_{neg,classified}) + \max(v_{pos})}{2} \quad (3.10)$$

for ROI with negative values higher than positive, and

$$T = \frac{\min(v_{pos}) + \max(v_{neg,classified})}{2} \quad (3.11)$$

for ROI with negative values lower than positive.

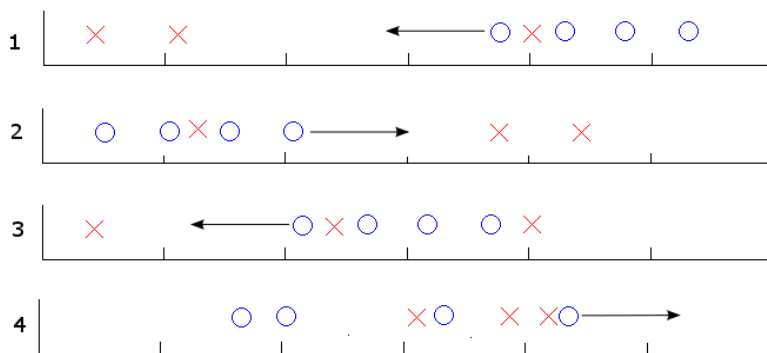


Figure 3.15: Four cases of ROI values. The arrow shows the chosen threshold direction relative to the positive values.

Threshold direction

The comparison between detailed inspections requires that a decision is made if the threshold should be set over or under the positive values. For the pattern tool the threshold always has to be under positive values, so there is nothing more that needs to be done. For the pixel and edge pixel counter tool however, a decision needs to be made.

There are four cases to be considered, illustrated in Figure 3.15. The first two are the simple cases when only one threshold direction makes sense. This is when a threshold on one side of the positive values can separate any negative values, and a threshold on the other side can not.

The third case is when it is possible to separate negative values with a threshold on either side of the positive values. In this case the direction that has the biggest distance to the first classifiable negative is chosen.

The fourth case is when no negative images can be separated by the detailed inspection. This is of little interest but will be handled by choosing the direction that minimizes the distance to when a negative value can be correctly classified.

3.5.6 Search

With a method to compare two detailed inspections with each other, the next step is to find the one with the best score. With six parameters to vary, using an exhaustive search approach would be very time consuming. Since the comparison score derived is not an analytical function, but can only be sampled at specific points, an analytical derivative can not be derived. In other words, it will not be possible to use a standard gradient decent algorithm. Instead the choice has been made to make use of the Nelder-Mead search algorithm. The choice of this algorithm over others is made because there was an implementation available in

the programming environment used. It works well on the image sets used for algorithm development.

As with most search methods available the Nelder-Mead search can get stuck on local minima [6]. To reduce this risk the searching is done in two steps:

- Generate a fixed number of detailed inspections, M , with different position, size and threshold over the images and calculate their score
- Pick the k detailed inspections with top scores, $k \ll M$, and optimize them using Nelder-Mead searching

Generating a lot of detailed inspections with different parameters and searching around these reduces the risk of getting stuck in a local minima far away from the optimum value. Optimizing the best of the generated detailed inspections instead of trying all possibilities will reduce the computation time.

The main assumption made in this approach is that the optimal detailed inspection can be found by optimization of one of the top generated ones. As a consequence, if M is too small, the optimal one might not be found. Also, if k is too small, the optimal might not be found.

Nelder-Mead

The Nelder-Mead search algorithm uses the concept of simplexes to search for a local optimum of a function. A simplex is a generalization of a triangle from two to arbitrary dimensions. The algorithm starts out by generating points in the parameter space in a simplex structure around the starting point, and calculates the function value and the center of gravity for these points. The algorithm then reflects points and expands or contracts the simplex with respect to the center of gravity depending on the function values in the points evaluated. It stops when the difference in function value in the simplex points and the size of the simplex are both below given thresholds. More specifics on how the Nelder-Mead search works can be found in [6].

Search limitations

The forbidden area calculated in Section 3.4 together with the image border is used to limit where detailed inspections can be placed. Since a detailed inspection can never be in the forbidden area or outside the reference image, no detailed inspections are generated with any parts there. The Nelder-Mead search algorithm is also altered so that it does not optimize a detailed inspection into the forbidden area or outside the reference image.

It turns out that a very thin detailed inspection in well-chosen places can give very good score values for areas that should not be inspected. To overcome this no detailed inspections with too small ROIs are generated and the Nelder-Mead search is altered not to allow too small values for ROI width and height. The limit is chosen as 20 pixels.

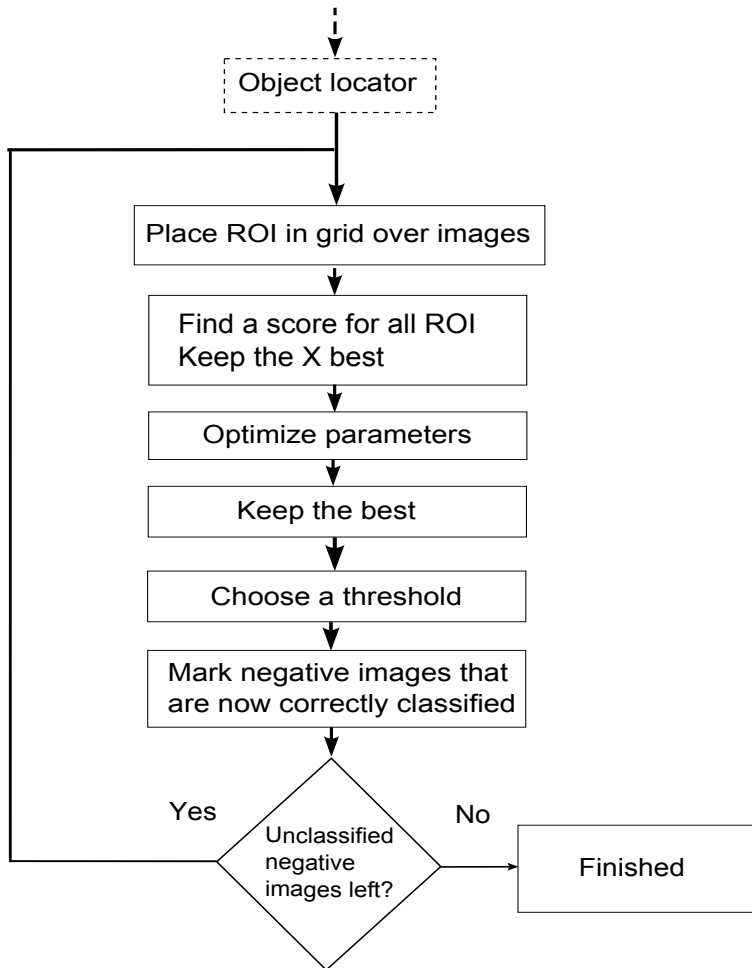


Figure 3.16: Flowchart for finding detailed inspections

3.5.7 Flowchart

The algorithm for detailed inspections is shown in the flowchart in Figure 3.16. The details on generating detailed inspections and optimizing these are described in section 3.5.6. The details on scoring and choosing threshold are described in Sections 3.5.4 and 3.5.5. The algorithm runs until all negative images are marked as correctly classified.

3.5.8 Robust scoring

When evaluating this way of searching for detailed inspections it does give the desired results for most cases considering only the training images. The goal however is to find detailed inspections that work well with images not used for train-

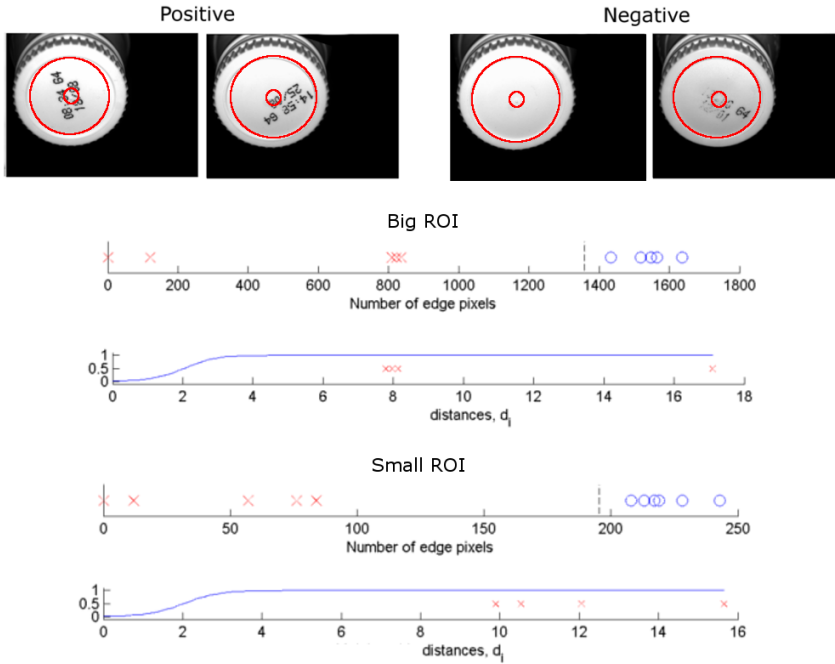


Figure 3.17: Two ROI in the date stamp example image with corresponding edge pixel count and d_i values

ing as well, and therefore some robustness in the ROI scoring is needed. Three things that could improve the robustness were found during development, and are described with examples in this section.

ROI too small

Figure 3.17 shows the spread plots and distances d_i for two different detailed inspections in the example image set 3. Comparing these two they both give close to perfect scores for all negative images, where the small detailed inspection has higher distance values for three out of four images. With both detailed inspections being scored as perfect, the search algorithm will pick one of them seemingly randomly.

The problem is that the training images are not representative for all possible input images. In this case, there is no negative image that has ink where the small ROI is, and therefore that ROI gets good scores. This problem occurs in many data sets and exists for ROI of pixel counter type as well.

The solution used is to assume that a ROI score should be stable for small changes in position. This can be seen as creating extra positive images from the original ones, that are translated slightly. The scoring is then run as before but with the extra positive images.

Illumination conditions

Figure 3.18 shows a pixel counter detailed inspection placed in the lamp image set. In this ROI none of the positive images have pixels below the threshold, but many of the negative images do. The score given to this ROI will therefore be big, and chosen before a ROI around a light bulb.

One likely reason that this problem arises is that the the positive and negative images are gathered separately. It can for example be more convenient to first take images of all the positive samples, and then of all negatives. It is then possible that the illumination conditions change between the classes.

This is handled by globally adding or removing intensity to the positive images, i.e. adding or removing a constant to all pixels. Creating new positive images where all pixels are increased or decreased by a constant gives three times as many positive images. This new set of images is then used to compare ROI in the same way as before.

Pattern inspection

For none of the gathered image sets, a pattern type detailed inspection would work better than an edge or pixel counter detailed inspection. Work has been put into finding such an application with no success. Therefore it has been decided not to look for detailed inspections of pattern type.

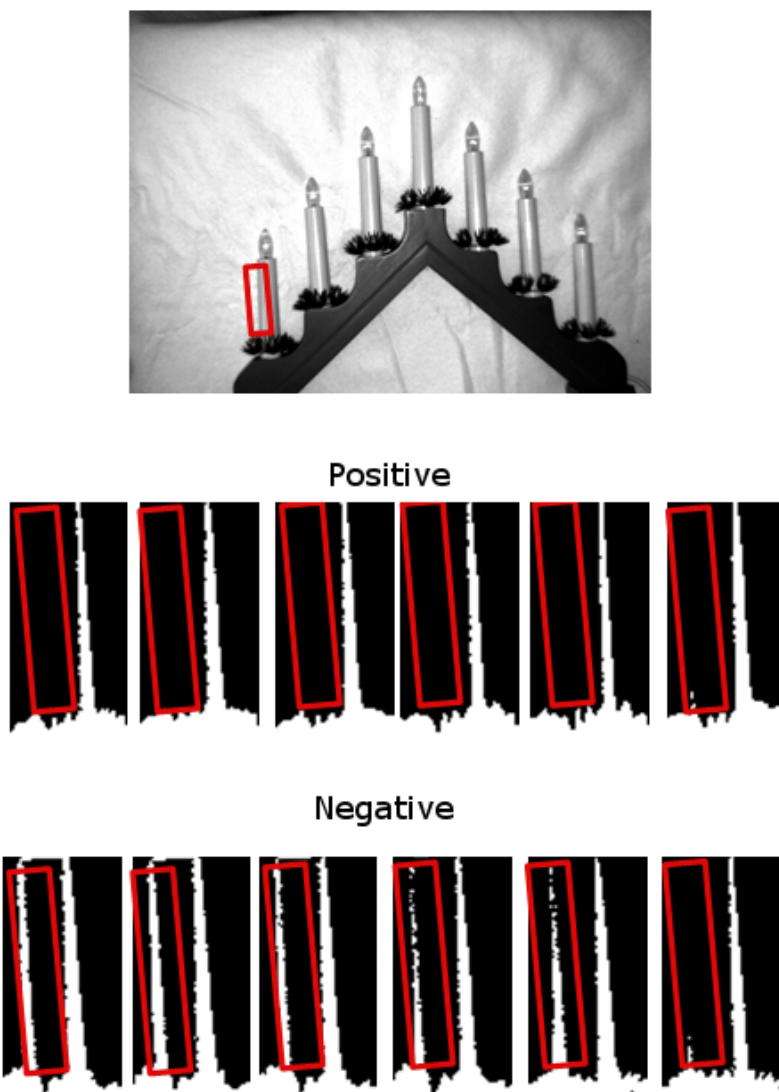


Figure 3.18: Pixel counter ROI where lighting is different between positive and negative images

4

Evaluation

This chapter covers the evaluation of the algorithm and the results of that evaluation. The goal of the evaluation is to draw conclusions as to how well the algorithm works for different kinds of objects, training image representativeness and allowed computation time.

In this chapter the term evaluation will be used when discussing the overall performance of the algorithm. The term validation will be used when comparing a result against ground truth. That is validation will tell if a configuration automatically derived from training images gives the correct classification compared to ground truth.

4.1 Data

For evaluation to be performed image sets are needed. All image sets have to be solvable with the restrictions given in section 2.3. They should also represent the kinds of applications for which the configuration algorithm should work. Lastly they should be of different difficulty with respect to disturbances such as lighting conditions, reflections and shadows to be able to tell how well the algorithm handles these.

Most image sets gathered are of the kind where something on the object can be removed or changed. The kinds of applications where something is broken, a stamp is missing or similar are harder to generate, and thus not as well represented. Care was taken when gathering the data so that no two images would be exactly alike, as this would have unwanted impact on the evaluation. The main way of achieving this was to move and rotate the object in the image. With light coming either from the internal light source of the camera, a window or a ceil-

ing lamp, reflections and shadows thus change between images. The number of images differ from about 15 to 40 images per image sets. There is a total of 979 images in all sets.

All images gathered are manually classified as positive or negative to use for training and ground truth in validation. For image sets where there are different kinds of positive or negative images, these are manually given different labels. Take for example the image set with advent candlesticks, presented in Section 3.1. Here there are seven different classes of negative images, one for each missing light bulb, and they are therefore given different labels. For images where more than one light bulb is missing, multiple labels are given.

The images are split into groups depending on their difficulty. These groups are:

Group 1:

- There is only a single kind of negative images
- The application can be solved with one ROI .
- Small changes in light conditions.
- Contrast between positive and negatives where inspection is to be performed is big.
- No ROI smaller than 50 pixels is needed.

Group 2:

- There are many kinds of negative images
- The application can be solved with one ROI .
- Small to big changes in light.
- Contrast between positive and negatives where inspection is to be performed is big.
- No ROI smaller than 50 pixels is needed.

Group 3:

- There are many kinds of negative images
- More than one ROI is needed to solve the application
- Small to big differences in lighting.
- Contrast between positive and negatives where inspection is to be performed can be small. ROI down to 30 pixels in size can be needed.

Group 4: Others. The lamp image set is used for illustration, but there are too few images to live up to the restrictions set on image sets in Section 2.3. The image set with four chips is used to illustrate problems with rotation symmetry.

Samples of all image sets are shown in Figures 4.1, 4.2, 4.3 and 4.4. The column to the left shows two positive images and the column to the right shows four negative. The images are chosen so that if there are different kinds of positive or negative images, as many of these as possible are shown.

The images are ordered depending on what group they belong to, starting with the first group and ending with the last. An empty line is left between each group to separate them.

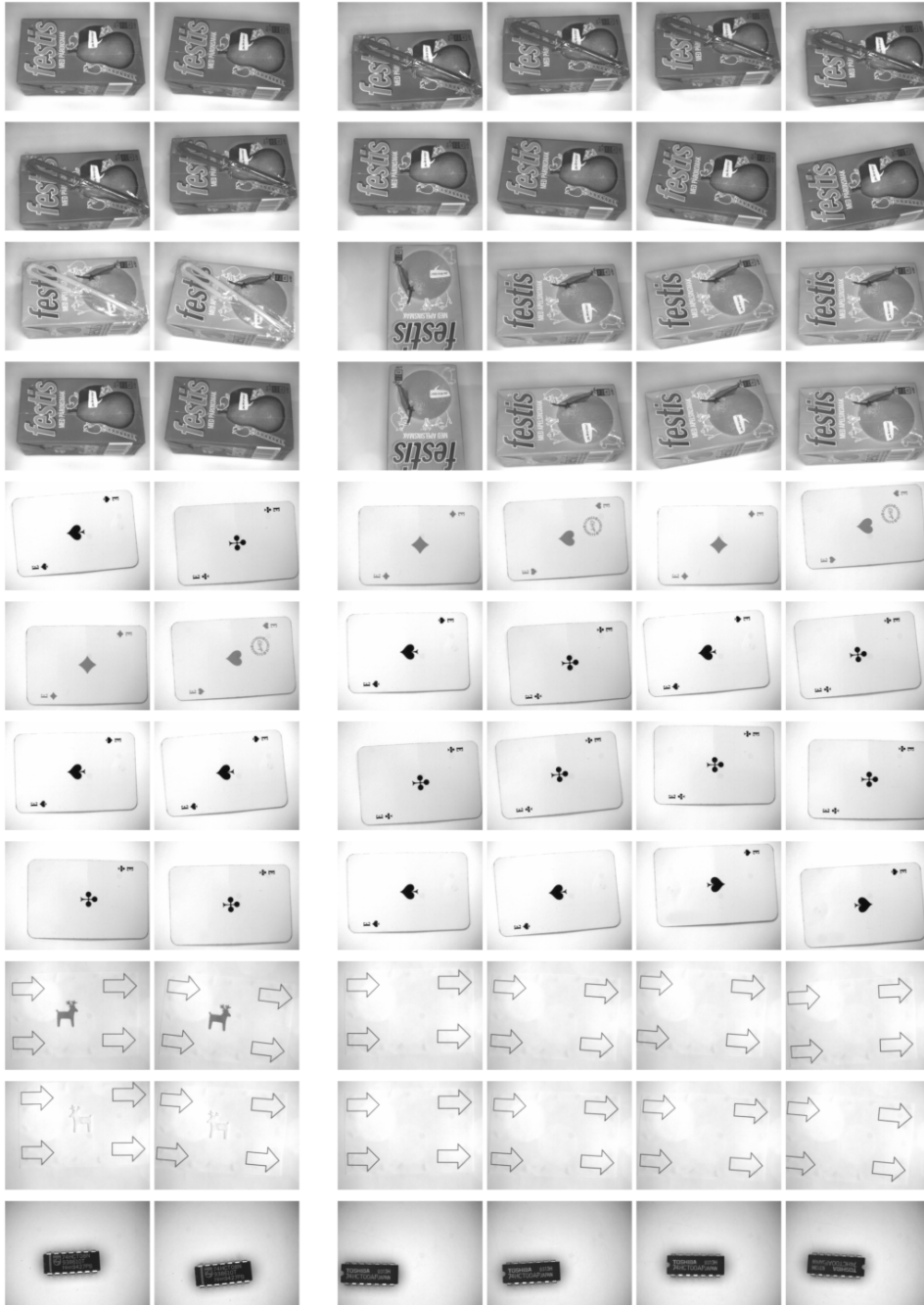


Figure 4.1: Summary of images used for evaluation of algorithm. Images in the left column are positive and images in the right column are negative

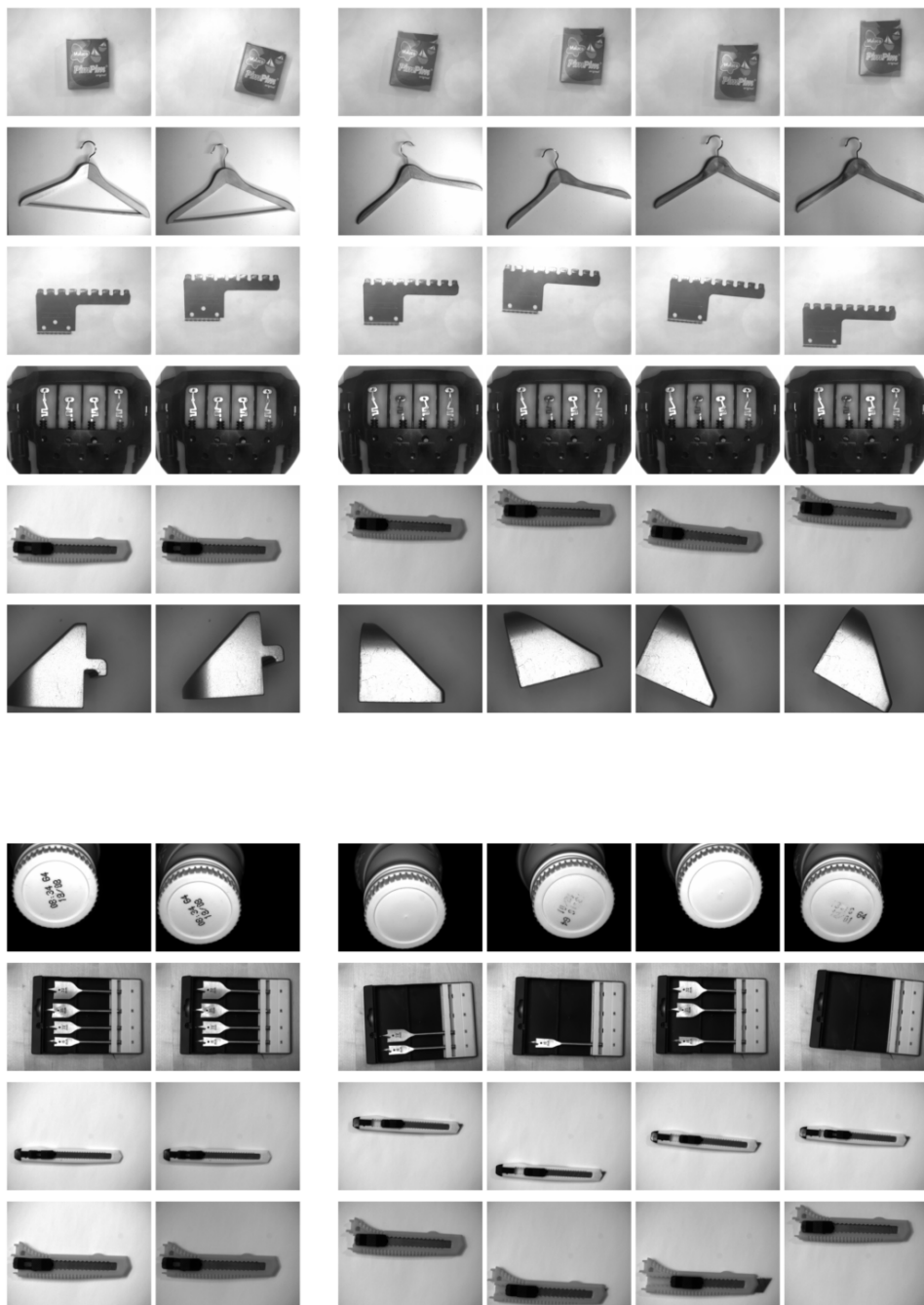


Figure 4.2: Summary of images used for evaluation of algorithm. Images in the left column are positive and images in the right column are negative

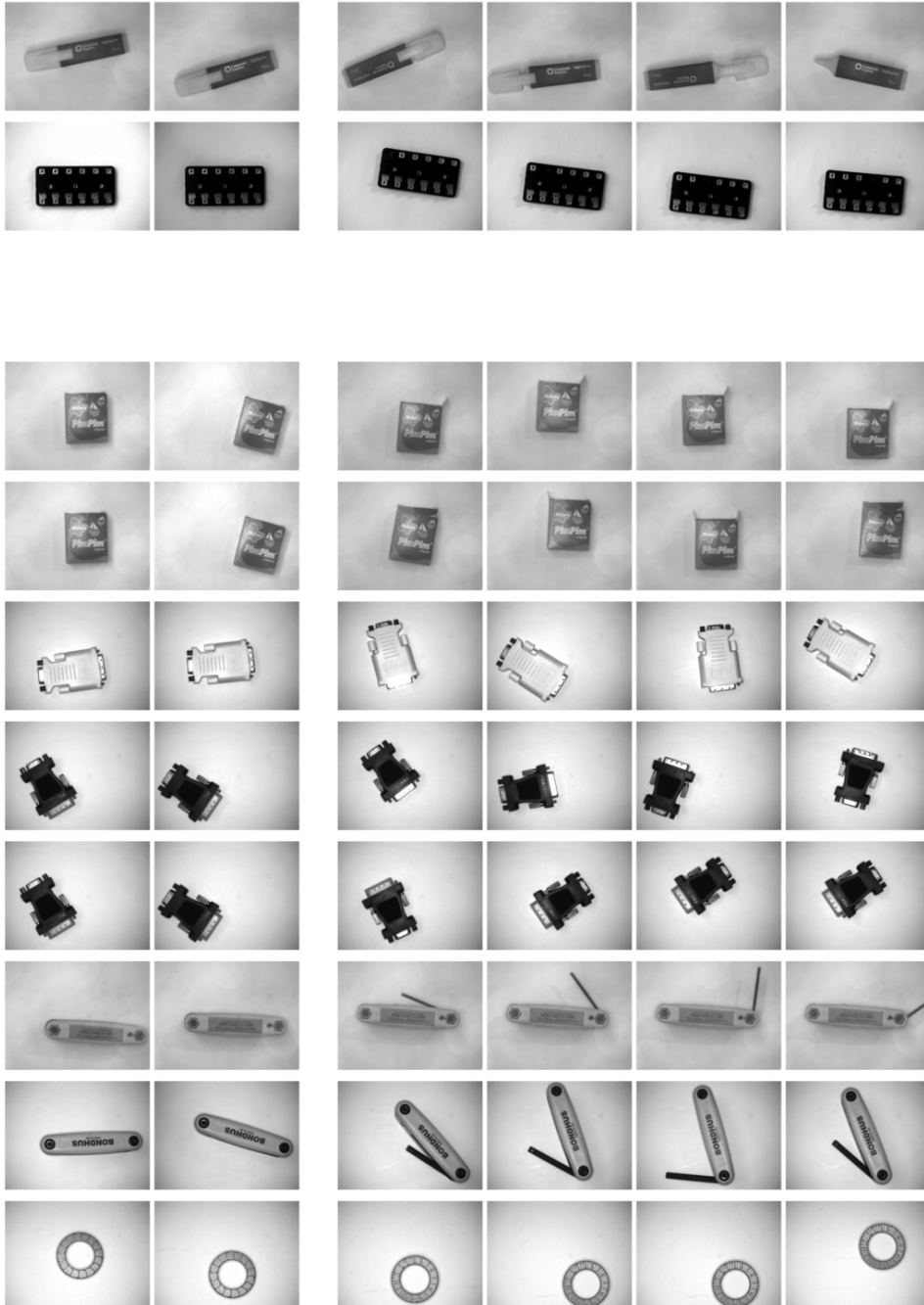


Figure 4.3: Summary of images used for evaluation of algorithm. Images in the left column are positive and images in the right column are negative



Figure 4.4: Summary of images used for evaluation of algorithm. Images in the left column are positive and images in the right column are negative

4.2 Evaluation algorithm

When evaluation is performed it is important that it is done on data that is not used when creating the configuration. One way of doing this is to split the available data into two sets, one for training and one for validation. In this project the amount of images for each set is very limited, and removing too many training images will have a bad effect on the algorithm performance.

One way to overcome this problem is to use cross validation, see [2]. This requires that the data is split into mutually exclusive subsets of approximately equal size. All but one of these subsets are then used for training, and the last one for validation. This is then rotated so that another subset is used for validation, and the classifier is retrained and revalidated.

4.2.1 Data labelling

There are two special cases that need to be considered when choosing how to split the image sets.

The first one is big variations in positive images. Take for example an application where one would want to separate black playing card aces from red ones, that is clubs and spades are positive. This application can be seen in Figure 4.1. If one splits the data set so that all positive images are clubs, there is a big chance that the configuration will not classify spades as positive. This big non-representativeness in training images should not affect the evaluation results, and needs to be considered when choosing how to split the image sets for validation.

The second case is different kinds of negative images. Take the lamp application in Figure 4.4, and consider that one of the restrictions from Section 2.3 is that the

negative images need to be representative. In this case, all the light bulbs that are missing in the validation set need to be missing in at least one image in the training set. More specifically every light bulb has to be the only missing light bulb in at least one training image. If this is not the case the training images does not live up to the restrictions, and this should not affect the evaluation results.

The solution to this is that all positive and negative images are labelled with what class or classes they belong to. When the data subsets are chosen for validation, it will have to be assured that all subsets contain at least one image with each label on its own.

4.2.2 Image subsets

These restrictions make it difficult to split the image sets with multiple kinds of positive or negative images into the kinds of subsets required for cross validation. Instead a variant is used where two positive and two negative images, randomly chosen, are used for validation and the rest is used for training. This is then repeated until a certain amount of validation results have been produced. When the images to be used for validation are chosen this is done so that:

- At least one of each class of positive and negative images exist in the training set
- This split into training and validation images has not been used before

A bigger amount of validation results should give a better accuracy on how well the algorithm works. Every training of images takes quite some time, and the total computation time available for validation is limited. The amount of validation results used for each image set will thus be limited by the available computation time.

4.3 Evaluation results

Evaluation has been performed with respect to application type, computation time and representativeness of images. The problem with rotation symmetry in the objects is also explained.

The evaluation is done with respect to accuracy and computation time. With the computation time being measured in seconds, it is of interest what hardware is used for computation, and what programming language is used. The algorithm is run on a dual core Intel processor with 1.86GHz clock frequency. The algorithm is not parallelized, and thus runs only on one of the two processor cores. The code is written in Mathworks Matlab, with care taken to vectorize the computationally heavy parts of the code to increase speed. More information on Matlab with vectorized Matlab code can be found in [4].

4.3.1 Application types

The evaluation should show how well the trained configuration works for different kinds of applications. To do this, image sets are split into groups depending on difficulty, as described in Section 4.1, and each group is evaluated separately. In this way it is possible to tell what applications are well handled by the algorithm, and what applications are not. Parameters are chosen for the algorithm to favour accuracy over computation time.

The validation is performed as described above, i.e. using two positive and two negative images for validation and the rest for training. The algorithm will be trained 15 times for each image set, which gives 60 validation images per set.

The results of the validation with respect to application group is presented in Table 4.1. For each group of images positive and negative predictive values (PPV and NPV) are calculated as

$$\text{PPV} = \frac{\text{Number of true positives}}{\text{Number of positive calls}} \quad (4.1)$$

and

$$\text{NPV} = \frac{\text{Number of true negatives}}{\text{Number of negative calls}}. \quad (4.2)$$

Here "number of positive calls" refers to the number of validation images that the configuration classified as positive. "Number of true positives" refers to the number of positive calls that were correctly classified, when comparing to ground truth. The same applies for NPV. This is to show if positive or negative values are more represented among the errors. A total amount of correct classifications is also presented for each group, as well as the average computation time for the image sets in the group.

Group	Positive calls	True positives	PPV	Negative calls	True negatives	NPV	Total	Computation time
1	512	510	0.996	508	508	1.00	0.998	116 seconds
2	177	177	1.00	183	180	0.984	0.992	153 seconds
3	280	266	0.950	260	256	0.984	0.966	203 seconds

Table 4.1: Validation results showing how well the configuration algorithm works for different kinds of applications

The resulting configurations for some of the image sets run are presented in Figure 4.5. The figure shows the reference image of each set overlaid with the resulting detailed inspection(s) used. Blue rectangles represent pixel counter inspections and green ones represent edge pixel counter inspections. The box in the top left corner of each image is green when all validation images were correctly classified, and red when not.



Figure 4.5: Samples of resulting configurations. Blue rectangles represent pixel counter inspections and green ones represent edge pixel counter inspections

4.3.2 Computation time

The evaluation should show how changing allowed computation time for the configuration algorithm changes its performance. The two main time consuming parts of the algorithm is the generation of detailed inspections in grid and the optimization of the top detailed inspections found. The grid generation depends mainly on how dense the grid is, and the optimization on how many of the top detailed inspections found are optimized.

To evaluate the effects of computation time the grid density and number of detailed inspections optimized will be varied. Nine different parameter settings for grid density and number of detailed inspections optimized will be manually chosen. The one with highest computation time will be the one used to evaluate performance with respect to application type. The others are then chosen so that each one has a lesser dense grid and optimizes fewer of the top generated detailed inspections. Because the validation has to be run once for each parameter setting, and computation time is still limited, 16 validation images per image set were used for this evaluation.

The resulting total accuracy for each group and parameter is plotted in Figure 4.6 against the group average computation time. The graphs show that accuracy tends to increase with computation time, but at some points accuracy is lower even though allowed computation time increases. Visually examining these extra misclassifications that occur shows that the algorithm has failed because there are areas where the training images are not representative. This is further discussed in the conclusions, Section 5.3.

4.3.3 Representativeness of training images

It is of importance for a user of the algorithm to know how the representativeness of the training images affects the results. It is, however, difficult to quantify, and there are important cases where it does not show in Table 4.1. Instead the algorithm has been run on the image sets with different amount of training images used, and the results have been visually evaluated. The representativeness problems that were found are presented below.

Missing data

An example of when the representativeness of training data is not sufficient is given in Figure 4.7. Here, the goal of the application is to detect if the left contact is leaning to the left. The second contact from the left is however always very much darker in the negative images than in the positive. This will lead to the algorithm picking a ROI around the second contact instead of the first one. Since this is the case in all available images of this set, the application still gets 100% accuracy in the above evaluation.

Illumination conditions

Differences in illumination conditions do have an effect on the result. This is the same problem as described in Section 3.5.8 where shadows fall in the lamp

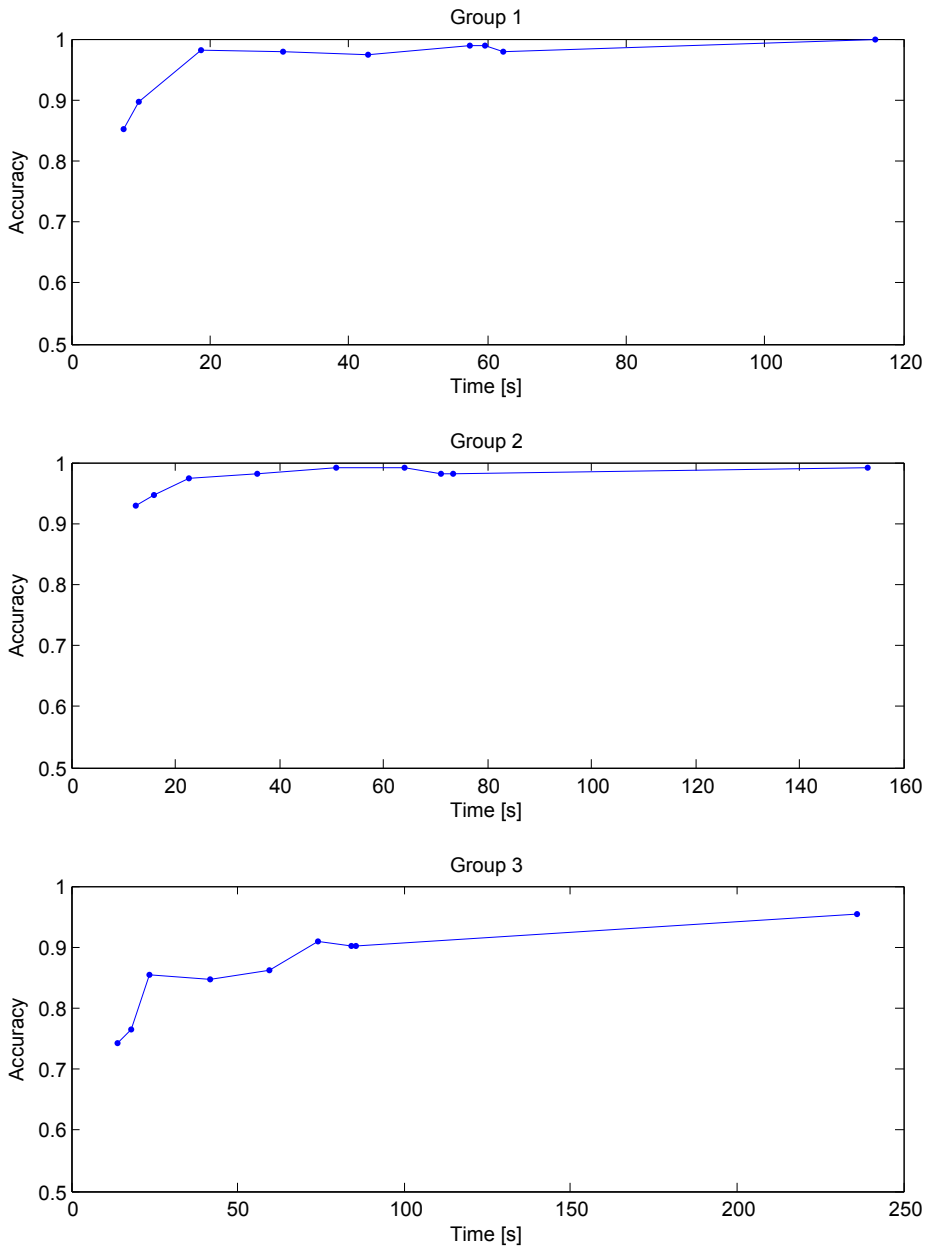


Figure 4.6: Accuracy for each group of image sets with respect to computation time



Figure 4.7: Examples of a representativeness problem. The goal is to detect if the left contact is leaning to the left, but the second contact from the left is a lot darker in all negative images and thus inspected instead.

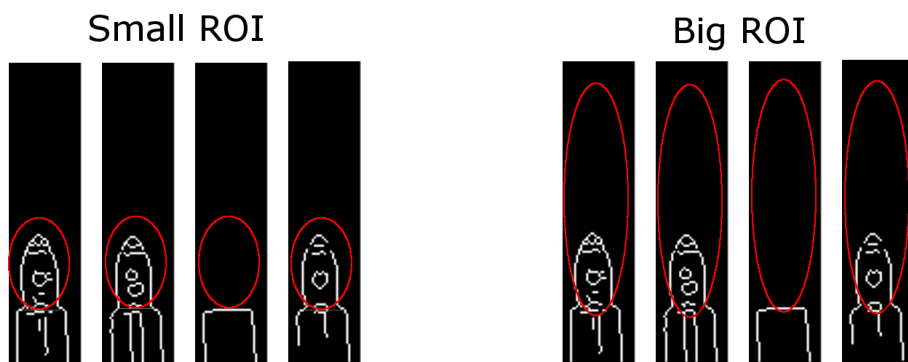


Figure 4.8: Two edge pixel ROI with different sizes

example image set. The solution to this problem that is presented does help in the cases where the differences are small. If they are too big though, the algorithm still fails.

ROI too big

Figure 4.8 shows a zoomed in image of the edges around a light bulb from the lamp example image, see Section 3.1. Since there are no edges in the background above the lamp in the training images, the search algorithm has no reason to choose the smaller ROI over the big one. The smaller one, however, is less sensitive to possible disturbances in the background, and would be preferable over the bigger. The same problem occurs for detailed inspections of pixel counter type as well. This problem does not show up in the accuracy estimations for the algorithm above, since no of the images gathered have background disturbances.

4.3.4 Rotation symmetry

Rotation symmetric objects are not handled by the algorithm, as illustrated by Figure 4.9. The positive images in this application have all four chips present, and the negatives have one or more chip missing. The object rotated 180° is very similar to itself, thus the object is rotation symmetric.



Figure 4.9: *Example of rotation symmetry problem*

Assume that the first positive image is used as reference image for the object locator. All other images will then be transformed so that the objects match the object in this. In the figure, the first negative chip can be found in two ways, either with the bottom right chip missing, or the top left. The same is true for the second negative image.

Because of this, the resulting images for which the algorithm is trained can either both have the top left and the bottom right chip missing, or just one of them. If only one chip is missing in the training images, the configuration will not handle both cases, and thus not give a good result.

5

Conclusions

This chapter covers the conclusions drawn from development and evaluation of the configuration algorithm. It focuses on how well the algorithm works, based on the evaluation results. It also describes the problems that still exist, and suggests future developments.

5.1 Applications

The evaluation shows that the algorithm works well on the first two groups of images evaluated, given that it is allowed a generous amount of computation time, see Section 4.3.1. The third group of images has an increased computation time and a lower accuracy.

It should be noted that the gathering of representative image sets for the more complex applications is not easy. One needs to be careful so that light falls in the same way in at least one positive image as it does in the negative. Where there are many kinds of negative images, one also needs to be sure that each kind is represented on its own in at least one training image.

This means that the more complex the application, the more understanding the customer needs. With the goal of this automatic configuration being the increased easy-of-use of the product, this is a big issue. At some point it might be easier to learn the image processing needed to configure the product in the "normal" way, instead of learning how to take good training images.

5.2 Computation time

The evaluation of accuracy at different computation times is described in Section 4.3.2. It shows that, for the image sets gathered in group 1 and 2, the computation time can be reduced a lot without losing accuracy. Choosing parameters so that the computation time is around 40 seconds gives good accuracy for the two groups. For image group 3 however, the accuracy falls a lot if computation time should be decreased that far. Thus when using the algorithm, a choice has to be made on what kinds of applications the algorithm should be able to handle, and what computation times are reasonable. The evaluation indicates that the more computation time that is allowed, the more difficult applications can be handled.

5.3 Future development

Two things were mentioned as important for this algorithm in the introduction of this report. Firstly the ease-of-use, and secondly the possibility to visually examine the results. To come all the way with these two parts there are some things that could be improved.

Computation time

The computation time needed for the algorithm is quite long. This is not a major problem, but the algorithm would be much nicer to use if it was faster. The implementation of the algorithm is currently done in MathWorks MATLAB, and an implementation in a faster programming language would probably reduce the computation time.

The computation time of a MATLAB program does depend very much on how it is written, and thus it is very difficult to determine how much faster it would be in another language. It would be possible to move the time critical parts of the algorithm to MEX-files [5], to get a possibly lowered computation time without having to rewrite all the code.

Algorithm accuracy

With the goal of the project to increase the ease-of-use of the product for the customer, it is important that the trained configuration works in as many cases as possible. Thus, increasing the accuracy of the algorithm would be a good improvement. The most important issues are listed in the evaluation chapter, Section 4.3.3.

Algorithm robustness

In the evaluation of how computation time affects accuracy, Section 4.3.2, it is noted that some parameter settings give less accuracy than those where both lower and higher computation time allowed. Why this happens is not known, but it indicates that there are robustness problems with the algorithm. This would be an important detail to investigate if the algorithm is further developed.

Visually logical results

Solving the problem with too big ROIs would be an important improvement. First it would, as mentioned, improve the accuracy of the algorithm in presence of background disturbances. Second the resulting configuration, if loaded into the user interface and inspected visually, would look a lot more logical. This would benefit the visual inspection of the configuration results a lot.

Bibliography

- [1] R.C Gonzalez and R. E. Woods. *Digital image processing*. Pearson International Edition, third edition edition, 2007. Cited on page 7.
- [2] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. Morgan Kaufmann, 1995. Cited on page 35.
- [3] R. De Maesschalck, D. Jouan-Rimbaud, and D.L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1 – 18, 2000. ISSN 0169-7439. doi: 10.1016/S0169-7439(99)00047-7. URL <http://www.sciencedirect.com/science/article/pii/S0169743999000477>. Cited on page 17.
- [4] Mathworks. Vectorization - matlab. http://www.mathworks.de/de/help/matlab/matlab_prog/vectorization.html, . [Online; accessed 2-May-2013]. Cited on page 36.
- [5] Mathworks. Mex - matlab. http://www.mathworks.de/de/help/matlab/matlab_external/introducing-mex-files.html, . [Online; accessed 14-May-2013]. Cited on page 44.
- [6] J. A. Nelder and R. Mead†. A simplex method for function minimization. *The Computer Journal*, (7), 1965. ISSN 308-313. doi: 10.1093/comjnl/7.4.308. URL <http://comjnl.oxfordjournals.org/lt.ltag.bibl.liu.se/content/7/4/308>. Cited on page 24.

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/>