Pós-Graduação em Ciência da Computação

SAULO ANDRADE PESSOA

A ROBUST TECHNIQUE FOR DETECTING CUSTOM PATTERNS OF ROUNDISH FEATURES

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Saulo Andrade Pessoa

A Robust Technique for Detecting Custom Patterns of Roundish Features

This thesis has been submitted to the Informatics Center of the Federal University of Pernambuco as a partial requirement to obtain the degree of Doctor in Computer Science.

ADVISOR: **Judith Kelner**

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

P475r Pessoa, Saulo Andrade
A robust technique for detecting custom patterns of roundish features / Saulo Andrade Pessoa. – 2017.
133 f.: il., fig., tab.

Orientadora: Judith Kelner.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Ciência da computação. 2. Visão computacional I. Kelner, Judith (orientadora). II. Título.

004 CDD (23. ed.) UFPE- MEI 2017-139

Saulo Andrade Pessoa

A Robust Technique for Detecting Custom Patterns of Roundish Features

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutora em Ciência da Computação.

Aprovado em: 10/03/2017.

_____

Orientadora: Profa. Dra. Judith Kelner

BANCA EXAMINADORA

_____

Prof. Dr. Silvio de Barros Melo
Centro de Informática / UFPE

_____

Prof. Dr. Carlos Alexandre Barros de Mello
Centro de Informática / UFPE

_____

Prof. Dr. Hansenclever de França Bassani
Centro de Informática / UFPE

_____

Prof. Dr. Alberto Barbosa Raposo
Departamento de Informática/PUC-Rio

_____

Prof. Dr. Alecio Pedro Delazari Binotto
IBM Research Brazil

ACKNOWLEDGEMENTS

Firstly, I would like to thank my family members and friends for all the support given throughout this thesis development and for concerning about me. Not to mention for having the patience to deal with me during this stressful period.

Many thanks also to my advisor Judith Kelner for all the encouraging words and for not giving up on me even when I was ready to give up on myself.

My sincere gratefulness to the coworkers of GRVM/GPRT for all the productive discussions conducted in many situations and for the patience spent in hours of manual labor to create the ground-truth images for the experiments.

Special thanks to Rafael G. de Mesquita and Leo H. Botler, who have gently helped me with the hypothesis tests performed to validate the results of the experiments.

Finally, I thank for the financial support received from FACEPE and Petrobras for the trustworthy collaboration.

"Do not subordinate fundamental principles to minor details"
(James E. Counsilman)

ABSTRACT

A fundamental task in computer vision is extracting low-level features from the image. Since this is one of the first tasks executed by most vision-based systems, imprecisions and errors committed during its execution are propagated to the next stages thus affecting the system overall performance. Therefore, robust and precise feature extractors are mandatory in computer vision. In the literature, two kinds of low-level features are commonly used: natural features, and artificial patterns of features. Natural features are extractable only from scenarios rich in textured elements. On the other hand, artificial patterns of features can be easily crafted by using commodity printers, which permits its application in a diversity of scenarios. Moreover, since the real dimensions of the pattern are known beforehand, the usage of artificial patterns allows the construction of metric systems. This thesis presents a new detection technique for patterns formed by roundish features. The new technique is composed of two stages: the extraction of candidates for features of the pattern; and the searching for the elements (among the candidates) that actually constitute the pattern. Differently from the techniques found in the related literature, the proposed one does not restrict the patterns to be rectangular grids of regularly-spaced features, but it allows the creation of a variety of patterns through the use of graphs (the pattern template). Experimental results collected from two case studies evidence that the new technique is robust to uneven and low-lighting conditions.

Keywords: Detection. Pattern. Roundish Features. Blobs.

RESUMO

Em visão computacional, uma tarefa fundamental é a extração de características da imagem. Por essa ser uma das primeiras etapas a serem realizadas na maioria dos sistemas computacionais baseados em visão, imprecisões e erros cometidos durante sua realização são propagados para as demais etapas afetando o resultado final obtido pelo sistema. Dessa forma, extratores de características que sejam robustos e precisos são uma necessidade em visão computacional. Na literatura, dois tipos de características são amplamente utilizados: características naturais; e padrões artificiais de características. Características naturais são extraíveis apenas de cenários ricos em elementos texturizados. Já padrões artificiais de características podem ser facilmente confeccionados com impressoras domésticas, permitindo sua aplicação em diversos cenários. Além disso, o uso de padrões artificiais possibilita que as medidas reais entre as características sejam previamente conhecidas (informação essencial à construção de sistemas métricos). Esta tese apresenta uma nova técnica para detecção de padrões artificiais formados por características arredondadas, sendo composta de dois estágios: a extração de elementos candidatos a característica do padrão; e a busca para encontrar quais elementos (dentre os candidatos) constituem o padrão de interesse. Diferentemente das técnicas encontradas na literatura, a técnica proposta não é restrita a detectar padrões retangulares formados por características uniformemente espaçadas; o usuário é livre para criar o padrão da sua escolha através da construção de um grafo (o gabarito do padrão). Experimentos realizados com imagens reais comprovam que a técnica proposta é robusta a iluminação não uniforme e a baixo contraste.

Palavras-chave: Detecção. Padrão. Características Arredondadas. Regiões de Interesse.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Advances in computer science and robotics allowed the development of modern systems which are able to perform daily human activities (talk, listen, smell, see, walk, handle objects, etc.). Despite many of these activities being effortlessly fulfilled by humans, they may be remarkably difficult for computers. In special, the human visual system can accomplish feats such as easily locating and recognizing elements of interest among a myriad of others, whilst a modern computer system would not be so efficient or effective performing the same task (LI, 2015). This visual skill disparity (humans vs. computers) is even bigger when modifications in the scene parameters/characteristics are allowed (i.e., when the scene is dynamic in some way). For example, even small changes in camera viewpoint, illumination, or in objects placement would be enough to confuse a vision-based computer system performing an image understanding task, whereas an untrained individual would hardly be impaired by such changes.

Recent breakthroughs in computer vision and image processing fields allowed even commodity computers equipped with webcams to mimic humans in some visual tasks. These regard both theoretical (methods, techniques, and algorithms) and hardware (improved optical devices/sensors, CPUs with more processing power, etc.) advances (BROWN, 2014), which caused the field of computer vision applications to increase over the last years. Some modern applications are 3D reconstruction, autonomous cars, robot guidance, tracking of objects, automated optical inspection for industrial purposes (also known as machine vision), and recognition of faces, irises, fingerprints, and printed characters.

Computer vision developments also support some applications of well-researched computer science areas such as augmented reality (AZUMA et al., 2001) and virtual reality (EARNSHAW, 2014). In augmented reality applications, both intrinsic (calculated by a camera calibration procedure) and extrinsic camera parameters (calculated by a pose estimation algorithm, which can be aided by fiducial markers or not) are required for coherently registering virtual objects into the real scene. In immersive virtual reality systems, HMDs (Head Mounted Displays) are used for presenting virtual worlds to the user. These systems must be aware of the user's head pose for properly setting up the virtual camera pose, which may be achieved by an optical head tracking system. Particularly for the head tracking problem, since it has many practical applications, a number of optical commercial solutions are available (ART, 2016b; PS-TECH, 2016a; NATURALPOINT, 2016; TRACKHAT, 2016; PINTO et al., 2008).

A fundamental task in computer vision is the extraction of *low-level features* (or simply *features*) from the image. This task is performed in the early stages of the

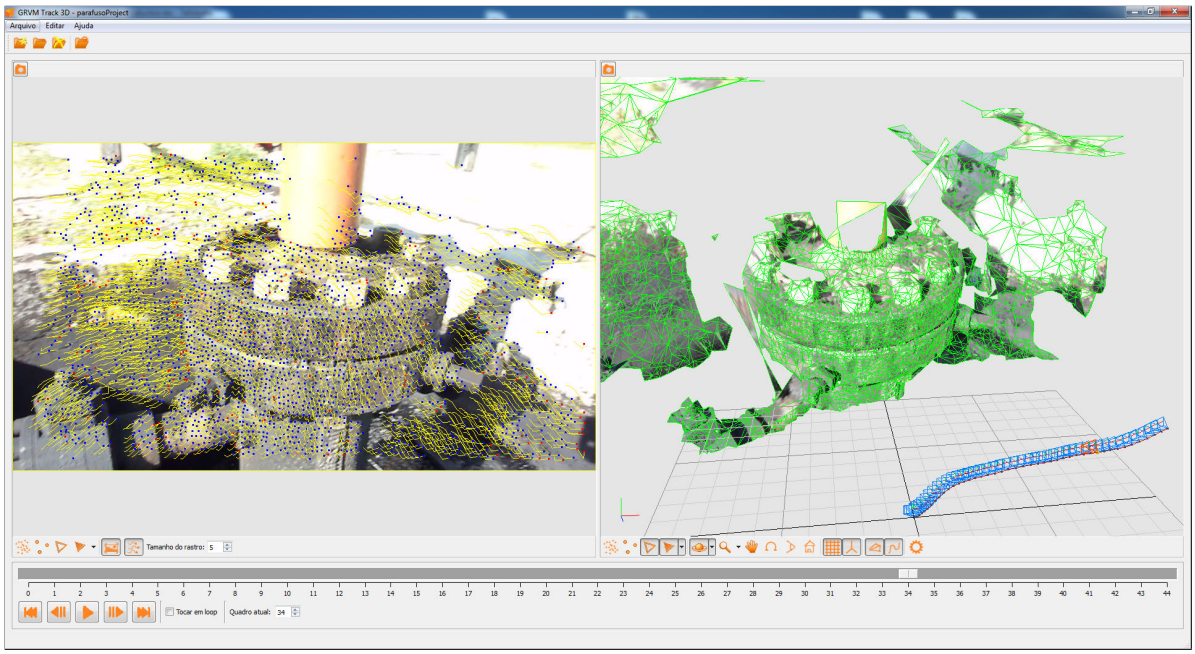
application execution providing summarized abstractions of image information; thereby the following stages are benefited by focusing only on refined data of the image. A feature is any piece of information extractable from images that is relevant for solving the problem of interest. Ideally, a good feature must have a well-defined position in image space and can be detected robustly. In the literature, diverse elements are used as features: edges, corners, points, lines, circles/ellipses, rings, blobs, and ridges (KANGAS, 2011). Recently, some authors also employed the more general term *interest points* when referring to corners and blobs (the feature concept is further detailed in Section 2.1).

Many applications make use of features that are naturally present in the scene. This is possible mainly when the scene is rich in terms of local information contents (i.e., it contains significant edges, corners, textured objects, or any other kind of detectable feature). However, in such scenarios, there are no guarantees that the selected features can be spatially related to each other in world space. For example, given that two features were detected in image space, the real world distance between them is not necessarily known. This lack of information has a notable impact on the system capabilities — it precludes metric[1] measurements of the scene elements. For example, the 3D reconstruction system shown in Figure 1 can only produce 3D models up to a scale factor. A real world measurement provided by the user would solve this impairment, although lessening the system autonomy. In addition, since the natural features have not been specially designed for the most robust detection, their extraction may not be very reliable.

On the other hand, there are scenarios where manufactured patterns of features are intentionally placed into the scene. These patterns arise frequently in computer vision because their structured geometry is well-suited for algorithmic detection and processing. In addition, some techniques used in the patterns manufacturing favor their precise/accurate detection, such as the usage of contrasting colors (usually black and white) and the employment of a color hue that are not present in the scene. Figure 2 presents a scenario where the usage of manufactured patterns is beneficial.

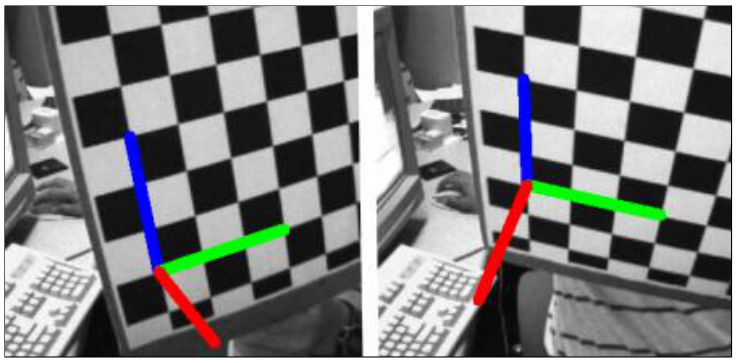
Despite the clear advantages of preferring manufactured patterns to natural features (better detectability and the possibility of previously knowing the features measurements), their usage is not always feasible. For example, in scenarios where intrusive elements are not allowed or when the scene scale is huge, manufactured patterns usage is prohibitive. In short, despite manufactured patterns not being always applicable, for some scenarios their benefits prevail against their drawbacks, thus they yet play an important role in computer vision.

[1] Throughout this chapter, the term “metric” refers to measurements expressed in real-world units. Therefore, a metric 3D reconstruction means that it includes the actual dimensions of objects.



Source: The author

Figure 1 – The R3D (GRVM, 2016) 3D reconstruction system which is based on natural features. On the left panel, the result of the features extraction stage is shown. Since no metric information can be obtained from the extracted features, the produced 3D model (right panel) is up to a scale factor.



Source: (OPENCV, 2016)

Figure 2 – A pose estimation algorithm based on manufactured patterns. The chessboard pattern printed over the board makes its detection easier than if it had a "solid" color. For augmented reality applications, this approach would produce more precise/accurate pose estimations, which in turn reduce jitter effects.

1.1 Motivation

Since feature extraction is one of the first tasks of most computer vision systems, any subsequent computation will be impacted by its precision/accuracy. From the practical standpoint, this means that errors made by the feature extractor will be propagated through the next execution stages, which, in turn, harms the overall quality of the system. Therefore, a precise/accurate feature extraction algorithm is necessary for any computer

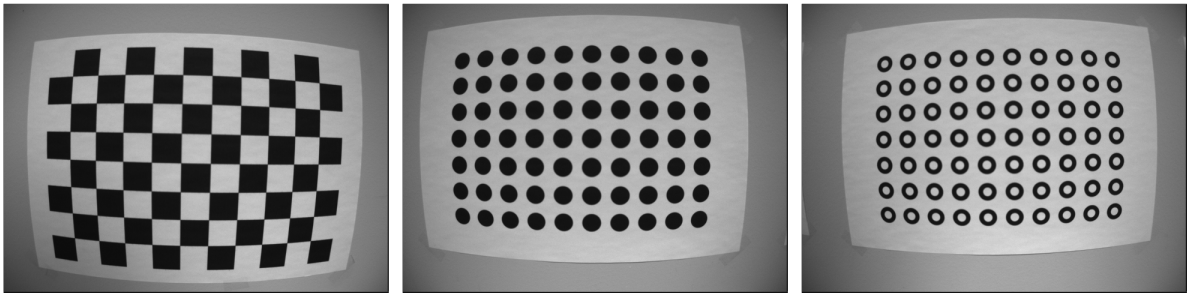
vision system.

As previously mentioned, using manufactured patterns has advantages over natural features. Particularly, the possibility of achieving a better feature detectability and the awareness of the features measurements. For example, if one has to develop a photogrammetry system (SCHENK, 2005) to perform a metric 3D reconstruction of a scene, manufactured patterns would probably be the best option for calibrating the camera. Beyond intrinsic and extrinsic camera calibration (ZHANG, 2001; ESCALERA; ARMINGOL, 2010; CARON; EYNARD, 2011), detecting patterns of features in images has a number of other applications, such as medical (BRUYANT et al., 2005; WANG; KOBAYASHI; SAKUMA, 2015), head tracking (MULDER; JANSEN; RHIJN, 2003), object pose estimation (OPENCV, 2016), and oil exploration (PESSOA et al., 2015).

Since planar patterns are inexpensive and easy to craft by using commodity printers, they are by far the most common kind of pattern. However, there are more complex patterns whose features are not restricted to be disposed over a planar surface (ART, 2016a). In addition, for increasing features contrast and consequently improving their overall detectability, they may be coated with special reflective materials; or active solutions based on LEDs may be used (PS-TECH, 2016b). Finally, approaches based on non-visible light, such as infrared, also exist. The advantage of working with non-visible light is that undesirable elements of the scene may be effortlessly suppressed from the final image, making the features easier to detect. The usual infrared approach consists in illuminating the scene with infrared light, coating the elements of interest (the features) with retroreflective materials, and using an IR-pass filter mounted in front of the camera lens (MEHLING, 2006).

Planar patterns are usually formed by primitives such as squares, circles, or rings (Figure 3). Patterns formed by squares present features such as corners, edges, and lines. At the same time, patterns formed by circles and rings will most likely present ellipses due to the inherent projective distortions (unless the pattern is perfectly facing the camera). Positional data from circles (and consequently from rings) are extracted by evaluating their centroids.

Given the importance of detecting patterns of features for some computer science fields, this Thesis proposes a new technique for achieving so. In special, the patterns of interest here are formed by roundish features (such as the one depicted in Figure 3b). The term “roundish” was propositionally employed because the pattern features are not required to be perfect circles/ellipses. As it will be detailed in next chapters, the technique makes no strict restriction about the shape of the features. This characteristic is what makes the technique suitable for scenarios wherein patterns cannot be precisely crafted, such as the one presented in Chapter 6 (in this scenario, the pattern was hand marked over a cylindrical surface).



(a) (b) (c)

Source: (HIGUCHI; DATTA; KANADE, 2016)

Figure 3 – Printed patterns of features formed by different primitives. Squares in (a), circles in (b), and rings in (c).

Experiments (Chapter 5 and Chapter 6) evidence the proposed technique is robust to uneven illumination conditions as well as to low contrast images. In addition, differently from most techniques found in literature, the proposed one supports features arrangements distinct from the conventional uniform rectangular grid, giving more freedom for creating customized patterns.

1.2 Problem definition

The problem tackled in this Thesis is summarized as follows. Given a gray-scale input image and a user-provided template of a pattern of features, one has to find in the input image the features that constitute the pattern. Users are not allowed to provide any hint about the pattern localization. Therefore, the system must be autonomous enough for finding the pattern by its own means.

More formally, let an image be a mapping $\mathrm{I} : D \to L$. For a typical 8-bit gray-scale digital image, the domain is $D = \{(x, y) \in \mathbb{Z}^2 : 0 \leq x < W \text{ and } 0 \leq y < H\}$ and its range is $L = \{l \in \mathbb{Z} : 0 \leq l \leq 255\}$, where W denotes the image width and H its height (GONZALEZ; WOODS, 2006a).

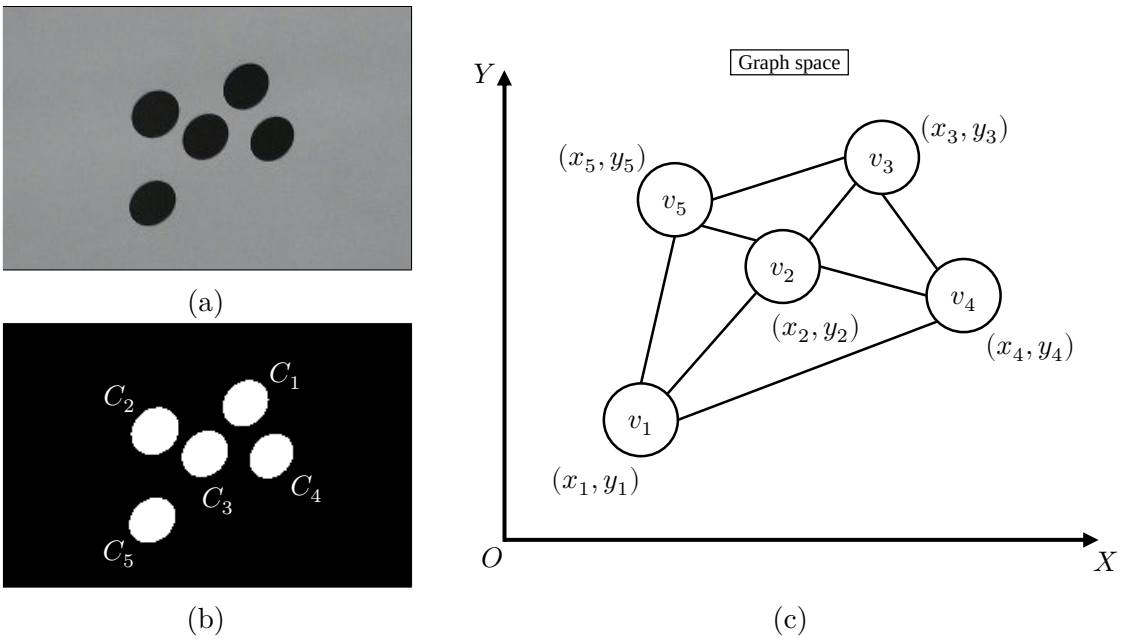
The neighbors of a pixel $p \in D$ with coordinate (x, y) are given by $\mathrm{N}(p) = \{(x+1, y), (x-1, y), (x, y+1), (x, y-1), (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$ (i.e., eight-neighborhood is adopted). Notice that pixels in the border of the image have some neighbors lying outside the image domain. These pixels are excluded from the final neighbors set.

For two pixels p and q, both in D, $p \in \mathrm{N}(q)$ if and only if $q \in \mathrm{N}(p)$ — if so, p and q are denominated adjacent (denoted by pAq). A sequence of pixels $(p, a_1, a_2, \ldots, a_n, q)$ is a path from p to q if and only if pAa_1, a_iAa_{i+1} for $i \in \{1, \ldots, n-1\}$, and a_nAq (i.e., it is a sequence of adjacent pixels). A connected component C is a subset of D such that for

all p and q in C there is a path $(p, a_1, a_2, \ldots, a_n, q)$ and $\{a_1, a_2, \ldots, a_n\} \subset C$.

The template of a pattern is represented by a finite, connected, unweighted, and simple graph $G = (V, E)$. Since it is a simple graph, loops and multiple edges are not allowed. This graph has the set of vertices V as the set of pattern features and the set of edges E as the set of immediate neighborhood relations. In addition, each vertex $v_i \in V$ has a position, which is given by $\mathrm{P}(v_i) = (x_i, y_i)$.

The problem consists in finding a mapping $\mathrm{M} : V \to A$ that satisfies the geometrical restrictions[2] imposed by G, where A is the set of all possible connected components over the image domain. See Figure 4 for an illustration of the problem. Because of the nature of the patterns of interest, which prohibits features to overlap each other (be it a total or partial overlapping), some remarks on this mapping can be made. Firstly, the mapping M is an injective non-surjective function. Secondly, be the subset $A' \subset A$ the image of M, for all C_i and C_j in A' such that $i \neq j$, $C_i \cap C_j = \emptyset$.



Source: The author

Figure 4 – Illustration of the problem tackled in this Thesis. In (a), an input image exhibiting a pattern of five circular features. In (c), an illustration of the template of the pattern of interest (a graph where each vertex has a position). For this example, a solution is the mapping that gives $\mathrm{M}(v_1) = C_5$, $\mathrm{M}(v_2) = C_3$, $\mathrm{M}(v_3) = C_1$, $\mathrm{M}(v_4) = C_4$, and $\mathrm{M}(v_5) = C_2$, where C_i is a connected component depicted in (b).

For most applications, it only makes sense to find the pattern in an all-or-nothing

[2] Note that since a position is attached to each vertex of G, not only topological but also geometrical information is embedded into G. This geometrical data is essential for the algorithm operation and it is what restricts wrong patterns to be detected.

fashion[3]. However, in scenarios such as the one presented in Chapter 6, the partial detection of the pattern is required, so this case will be also addressed by this Thesis (up to some extent). This means that for such scenarios, the mapping sought is actually a partial function (i.e., not all graph vertices will have a connected component mapped to).

As a final statement, it is worth emphasizing that the problem described here cannot be solved by a singly binarization, thresholding, interest region extractor, or any other kind of segmentation technique. While these techniques can indeed detect a region in the input image for each feature of the pattern, they cannot establish the required correspondences between the regions and the vertices of the pattern template.

1.3 Objectives

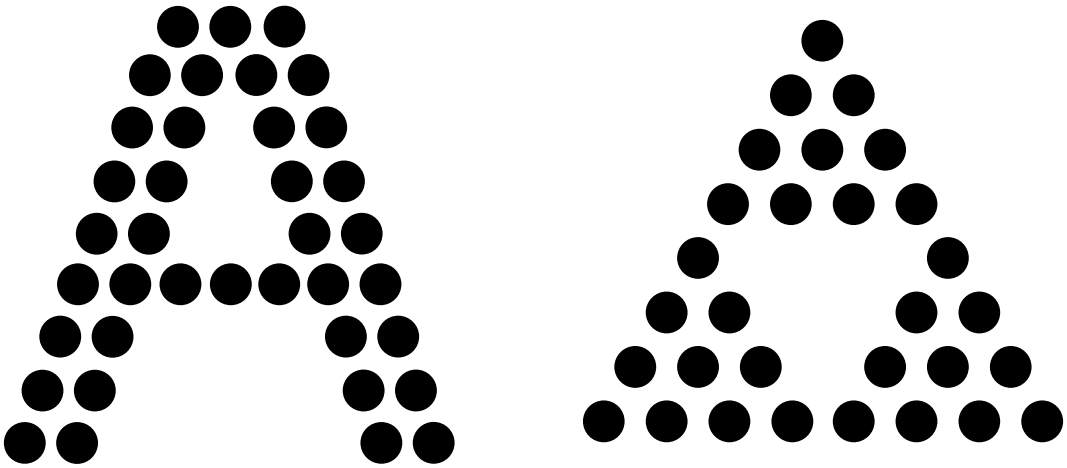
Given the aforementioned motivations, the general objective of this work is to propose a new technique for detecting patterns of roundish features. In contrast to the state-of-the-art, the proposed technique employs a more general mechanism for the pattern specification, allowing users to create their own patterns. In addition, aspects such as robustness to restricted lighting conditions and tolerance to lens distortion must also be taken into account. The specific objectives of this work are:

1. To design the overall pattern detection technique;
2. To develop a blob extractor that is able to succeed under uneven and low-lighting conditions;
3. To propose a suitable data structure for representing different-shaped patterns; and
4. To compare the proposed technique with techniques from the state-of-the-art under different case studies.

1.4 Contributions

This Thesis presents a new detection technique for patterns formed by roundish features. It does not restrict the patterns to be rectangular grids of regularly-spaced features, but it allows the creation of a variety of patterns through the usage of graphs. Therefore, regarding this aspect, the proposed technique is more general than the state-of-the-art ones (refer to Figure 5 to see some examples of unusual patterns that can be detected by the proposed technique).

[3] For instance, if part of the pattern is not visible due to occlusion or a bad framing, it is not desired to detect the non-occluded part of this pattern, even though it was visible.



(a) (b)

Source: The author

Figure 5 – Examples of unusual patterns that can be detected by the proposed technique.

Experimental results collected from two case studies evidence that the new technique is robust to uneven and low-lighting conditions. In addition, by relying on short-range geometrical constraints during the searching stage, the proposed technique is able to detect deformed patterns.

Another contribution of this work is a new blob formulation which was built upon the ideas of two blob extractors from the literature. This formulation has the advantage of producing fewer outliers while at the same time creating representative blobs even when the pattern is formed by non-perfectly circular features.

1.4.1 Publications

The following papers were published during this Thesis development.

- Saulo Pessoa, Vinicius Cesar, Bernardo Reis, Judith Kelner, and Ismael Santos. **A Segmentation Technique for Flexible Pipes in Deep Underwater Environments**. British Machine Vision Conference (BMVC), 2015 (PESSOA et al., 2015).

- Ismael Santos, Eduardo Vardaro, Emerson de Goes, Volney Lopes, Alyson Vaillant, Andre Palmeiro, Judith Kelner, Vinicius Cesar, Saulo Pessoa, and Bernardo Reis. **Real Time Radius of Curvature Measurement During DVC Operations Based on Flexible Pipe 3D Reconstruction**. Offshore Technology Conference (OTC), 2015 (SANTOS et al., 2015).

1.5 Thesis structure

This Thesis is structured as follows. Related work is presented in Chapter 2. This chapter starts by covering the extraction of low-level features (giving special attention to blob extractors) and, at the end, it addresses works that targeted the overall problem of detecting patterns formed by circular features. Chapter 3 introduces the proposed pattern detection technique providing in-depth technical details (a formal definition, as well as implementation aspects, are discussed). The methodology employed in the experiments is described in Chapter 4. Next, the proposed technique is assessed through comparative experiments, which are conducted within two distinct case studies (Chapter 5 and Chapter 6). Finally, Chapter 7 draws the conclusions and presents open problems left as future work.

2 RELATED WORK

This chapter presents the works that are related to the technique proposed in this Thesis. The first covered topic is concerned with the initial stage of the proposed technique — the extraction of low-level features from the input image. For the context of this Thesis, this stage comes down to extracting elements termed as *blobs*. The last section of this chapter addresses the overall problem of detecting patterns formed by roundish features.

2.1 Low-level feature extraction

A typical computer vision system is composed of multiple sequential stages wherein specific operations are performed. The initial stages are usually in charge of extracting primitive visual elements that constitute the image — they are termed as *low-level features* or simply as *features*. These elements are important because they are transformed into entities of higher semantic level in the subsequent stages of the system (ARORA, 2007). For instance, regarding the problem of detecting the pattern shown in Figure 3b, the black circles are the low-level features and the pattern (as a whole) is the element of higher semantic level.

Extracting low-level features from images is a broad problem. Not surprisingly, different kinds of feature extractors have been proposed and used in the literature (KANGAS, 2011). Regarding the problem of detecting manufactured patterns, the most pervasive types of extractors are the ones based on corners, edges, or blobs.

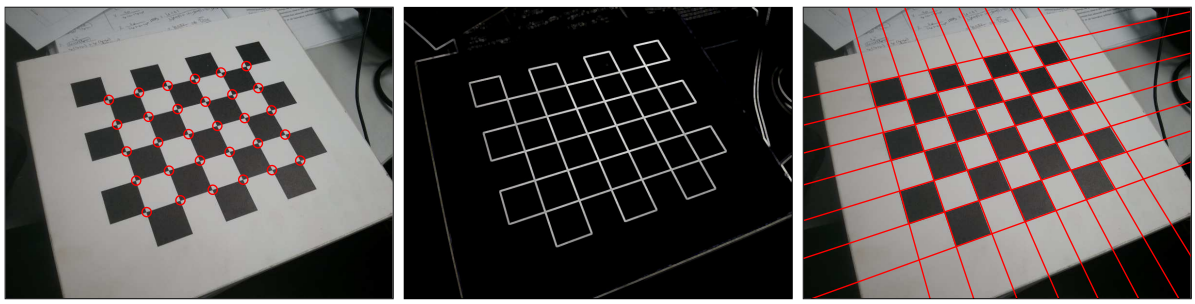
Corners can be imagined as regions of the image that have large intensity changes in more than one direction, such as in the intersection of two lines. The basic notion of a corner is analog to a point (i.e., a geometric element without dimensional attributes). One of the earliest corner detectors was presented by Moravec, which proposed determining the changes that resulted from shifting a local window in the image by a small amount in various directions (MORAVEC, 1980). This approach suffered from anisotropic responses, which was later addressed by Harris and Stephens. Harris and Stephens also improved the noise response of the operator, making this new detector known as the Harris Corner Detector (HARRIS; STEPHENS, 1988).

Edges are elongated elements extracted from regions at which the image brightness has discontinuities. They are analog to lines and represented by sets of connected pixels which delimit the boundaries between disjoint regions. Edge detectors have many practical applications in the computer vision field. A recent work evaluated edge detectors and concluded that among Prewitt, Robert, Sobel, and Canny, Canny's algorithm performs

better for noisy image and additionally has a lower probability of extracting false edges (KUMAR; SAXENA et al., 2013).

In any event, corners and edges are not the best kinds of feature for underlying the detection of patterns formed by roundish features. Essentially, corners are zero-dimensional elements, so they cannot capture the notion of area. The same applies for edges[1], which are one-dimensional elements.

On the other hand, corners and edges suit well for detecting chessboard patterns. In a chessboard pattern, the regions wherein squares intersect each other are easily identified by corner detectors (Figure 6a). Similarly, edge detectors are sensitive to square edges (Figure 6b). If the image does not suffer from significant radial distortions, edges can further be transformed into straight lines (Figure 6c), something that can be accomplished by a Hough Transform algorithm (HART, 2009). For some scenarios, directly storing lines is advantageous because it is a more compact representation (while edges are stored on a per-pixel basis, each line requires only two scalar parameters).



(a) (b) (c)

Source: The author

Figure 6 – In (a), the corners of a chessboard pattern are highlighted in red. In (b), the edges extracted from the same pattern. Finally, in (c), edges are transformed into straight lines.

Under those circumstances, this work does not go further reviewing corners and edge detectors. For more detailed reviews, as well as comparative evaluations, refer to some papers of reference (SCHMID; MOHR; BAUCKHAGE, 2000; MOKHTARIAN; MOHANNA, 2006; TUYTELAARS; MIKOLAJCZYK, 2008; GAUGLITZ; HÖLLERER; TURK, 2011; PATEL; PANCHAL, 2014; SUBBAN; PRABAKARAN, 2015).

2.1.1 Blob extraction

More relevant to this Thesis are blob extractors. A blob can be defined as a set of connected pixels which share some property in common. Differently from corners and

[1] As a matter of fact, edges can capture the notion of area if they form closed contours. However, from the practical standpoint, the usage of edge detectors for extracting regions is limited because they are subject to produce discontinued edges, and in such a situation contours would not be closed anymore.

edges, blobs are regional entities (i.e., they extend over the image domain). Due to this inherent characteristic, blobs are also referred to as *regions of interest*. The notion of a roundish feature previously exposed in this Thesis (the low-level element of interest) is formalized by the blob concept.

An important concern about blob extraction algorithms regards their ability to tolerate changes in blobs size. A blob extractor is said to be *scale invariant* if it is able to detect blobs over a range of scales without requiring a new parameterization for each scale level. This characteristic is important because in many practical applications patterns are allowed to be moved closer and away from the camera, and in such a situation the size (in image space) of the features changes due to the various distortions involved in the image formation process. For instance, during a camera calibration process features can considerably increase in size when the pattern approaches the camera; conversely, distancing the pattern from the camera has the opposite effect in features apparent size.

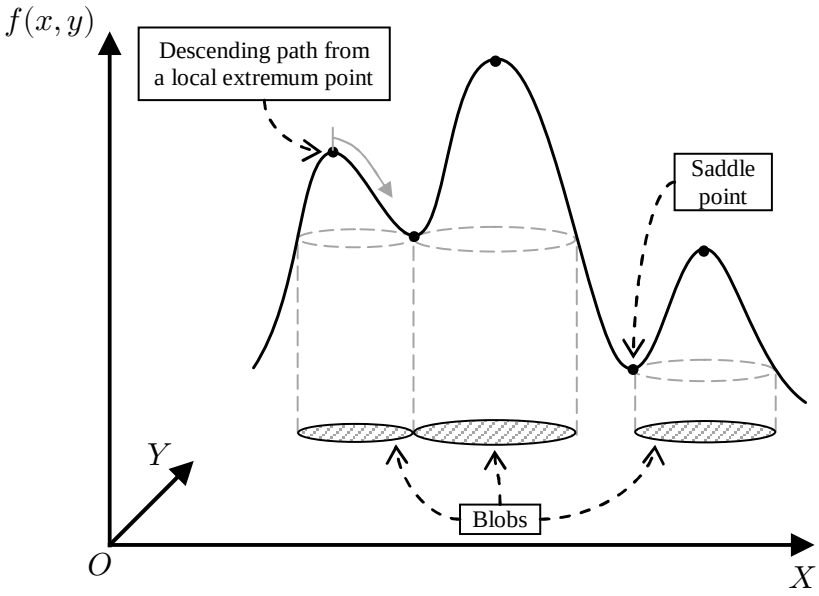
Several approaches have been proposed for extracting blobs. Many of them proceed by analyzing the responses given by differential filters at different regions of the image. If the filter produces a strong response at a region of the image, then the region is extracted as a blob. One remark about this approach is that not only the operator location but also its scale affects the overall extraction performance (a small windowed operator placed at the center of a big feature does not produce a strong response, and vice versa). Thereby, it is common to perform a multiple scale analysis of the image for ensuring differently sized blobs are extracted (LINDEBERG, 1994).

The Laplacian of Gaussian (LoG) operator has been largely used for extracting blobs. In particular, approximations of this operator were mostly aimed since they are faster to compute. For instance, the Difference of Gaussians (DoG) operator is a good approximation and is computationally less demanding. This operator consists in filtering the original image with differently sized Gaussians and evaluating the difference between consecutive smoothed images. If scale invariance is desired, then successive smoothing operations are required for ensuring the whole range of blob sizes is covered. However, these operations are known to decrease the signal amplitude, which in turn bias the operator responses obtained from different scale levels. An appropriate normalization mechanism compensates this decreasing (LINDEBERG, 1998).

Another operator found in the literature for extracting blobs is the Determinant of the Hessian matrix (DoH). This operator is also coupled with a multiple scale analysis and a normalization mechanism for achieving scale-invariant behavior (LINDEBERG, 1998). Blobs extracted by the DoH operator have better scale selection (as compared to the Laplacian-based ones) when image undergo non-Euclidean affine transformations. Additionally, a recent publication (LINDEBERG, 2015) compared different operators and concluded that the DoH achieved the best features matching performance under

perspective image transformations.

On the other hand of the blob extractors based on differential filters, there are the ones that proceed by finding the local maxima and minima of the image function. In particular, the notion of blob exposed by Lindeberg and Eklundh (LINDEBERG; EKLUNDH, 1990) is intimately related to the one presented in this Thesis (Section 3.2). By imagining the surface plot of a gray-scale image one can intuitively define this blob as a region associated with a local extremum point. This region is permitted to grow, as a descending path is followed, until the path assumes an ascending inclination toward another local extremum point (i.e., a saddle point is reached). Figure 7 illustrates this notion.



Source: The author

Figure 7 – Illustration of the Lindeberg and Eklundh's blob concept. The blobs at the bottom of illustration are merely projections of the upper isolines (the dashed elliptical lines in gray).

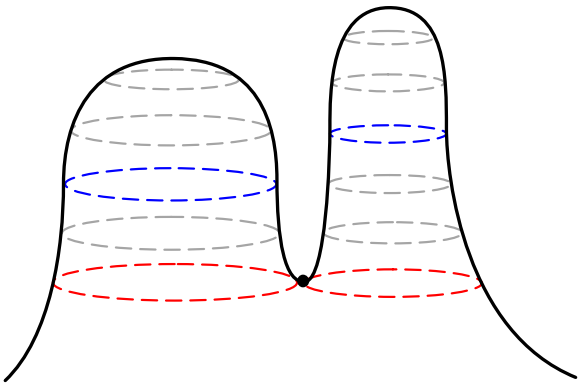
This blob notion was later formalized and an algorithmic procedure for its extraction was presented (refer to the paper (LINDEBERG; EKLUNDH, 1991) and to the Lindeberg's Thesis (LINDEBERG, 1991)). It is worth noting that, in these works, the authors were interested in something broader than the simple blob concept — the extraction of salient structures from images without a priori information about the structures (i.e., how to determine the scale of an object and where to search for it before knowing its size and where it is located). For achieving so, they presented a multi-scale representation for gray-scale images, which was termed the Scale-space Primal Sketch. This representation permitted them to successively inspect structures/blobs from fine to coarse levels of scale. The most salient ones were selected as being important regions of the image. Additional attention to

the behavior of blobs in scale space was given in a posterior paper (LINDEBERG, 1992).

To the best knowledge of the author, no further developments regarding the Lindeberg and Eklundh's blob extractor were published. The posterior works mostly comprised experiments with real imagery (LINDEBERG; EKLUNDH, 1992). The Scale-space Primal Sketch concept was also employed in problems such as edge detection and automatic peak detection in histograms (LINDEBERG, 1993), as well as PET data analysis for cerebral blood flow measurements (LINDEBERG; LIDBERG; ROLAND, 1999).

Another important work related to blob extraction was presented by Matas et al. (MATAS et al., 2004). They studied the problem of establishing correspondences between elements extracted from a pair of images taken from different viewpoints. For such a thing, they introduced a new set of image elements/features denominated as *maximally stable extremal regions* (MSERs). Despite this different formulation and the absence of references to the Lindeberg and Eklundh's former work (LINDEBERG; EKLUNDH, 1990), MSERs are very alike the blobs presented in Figure 7.

In essence, both works proceed by analyzing the isolines of a surface plot at different levels and selecting the appropriate ones for extracting blobs. What differs them is basically the selection criterion employed. While Lindeberg and Eklundh extract blobs from the isolines that occur at saddle points, Matas et al. select the maximally stable isolines (Figure 8). By "stable isoline" one means an isoline that is alike to others over a range of nearby levels (more details about this topic are covered in Subsection 3.2.4).



Source: The author

Figure 8 – Multiple isolines at different levels of a surface plot. Essentially, these are the elements analyzed by both the Lindeberg and Eklundh's blob extractor; and the Matas and coworkers'. Isolines in red are the ones selected by the former blob extractor. The latter extractor selects the blue ones.

The blob extractor proposed in this Thesis was inspired by both Lindeberg and Eklundh's work (LINDEBERG; EKLUNDH, 1990); and Matas and coworkers' (MATAS et al., 2004). As is detailed in Section 3.2, the proposed extractor combines the best of both worlds for achieving better results.

As a final observation, there are many circle detection methods which are variants of the Hough Transform that could be used as blob extractors (MUKHOPADHYAY; CHAUDHURI, 2015). These methods are commonly used in the literature because they have the advantage of being relatively unaffected by the presence of image noise. However, since they demand considerable computational time and large storage (in particular, three-dimensional accumulators are necessary for detecting circles with an arbitrary radius), their application to some scenarios is limited.

2.1.2 Image segmentation techniques as blob extractors

Despite not being formally presented as blob extractors, many image segmentation techniques (ZHU et al., 2016) can be used as so. These techniques work by partitioning the image into disjoint sets of similar pixels. Since these sets contain pixels that share a common characteristic, from the practical standpoint they can be considered as blobs.

However, at least for the scenarios analyzed in this Thesis, image segmentation techniques are unlikely to produce comparable results to those obtained by blob extractors. The main problematic is that most image segmentation techniques are too much general-purpose and do not explore the specificities of the problem in hand to reach good results.

In an earlier presentation of the work presented in this Thesis (PESSOA et al., 2015), some image thresholding techniques were comparatively evaluated against the proposed blob extractor. Among the four assessed thresholding techniques, there was one that uses a global threshold for all pixels (OTSU, 1975) and the other three were locally adaptive methods (BERNSEN, 1986; BRADLEY; ROTH, 2007; SAUVOLA; PIETIKÄINEN, 2000). The global method was not able to segment the features of interest in any of the tested images. The adaptive methods achieved better results, although they did not match up in performance with the proposed technique.

The main difficulty with the adaptive methods was that they require a specific parameterization for each input image in order to achieve good results — if a small window size is used then blobs with holes are produced, on the other hand, large windows fail to separate blobs brought forth from close features. On top of these issues, it must be noted that most of the thresholding techniques were created in the context of the document image binarization problem, so there is an excuse for their deficient performance while extracting blobs.

One general-purpose image segmentation technique considered for experimentation (see Chapters 4, 5, and 6) is the P-Algorithm (BEUCHER; MARCOTEGUI, 2009; BEUCHER, 2013). In spite of P-Algorithm standing on the watershed transformation, it does not suffer from the typical over-segmentation problem presented by implementations based on the watershed principle (GONZALEZ; WOODS, 2006b). Another advantage

of P-Algorithm is that it is self-adaptable to the changes that occur from one input image to other (P-Algorithm is denominated as a parameterless technique). This last advantage, in particular, makes P-Algorithm suitable for experimentation because this is the way that blob extractors are expected to work — without requiring to be constantly reparameterized.

2.2 Detection of patterns formed by roundish features

This section presents the works found in the literature that approached the problem of detecting patterns formed by roundish features. Two different strategies have been used for collecting these works: 1) backtracking, which consists in searching for the "cited by" papers of a publication of reference; and 2), hand searches in the proceedings of relevant conferences (IEEE, 2017a; IEEE, 2017b; IEEE, 2017c) and journals (ELSEVIER, 2017a; ELSEVIER, 2017b), which comprehended the period from 2005 to 2016.

The OpenCV software library (BRADSKI, 2000) implements a blob extractor as well as a circular pattern detector (the latter is powered by the former). The blob extractor works by firstly applying consecutive ordinary thresholding operations to produce a set of binary images. Afterward, in each binary image, connected white pixels are grouped together creating a set of blobs. Next, close blobs from different thresholding levels are merged producing the final set of feature candidates. Finally, the centroids and radii of these candidates are evaluated. The pattern detector of the OpenCV can recognize two types of patterns: the symmetric (a rectangular grid of circles) and the asymmetric (similar to the symmetric one, although the odd rows of the pattern are offset). However, experiments testify that the OpenCV detector is not suitable for scenarios with limited lighting conditions (see Chapter 5).

Smith and Smith have used thresholding and regression analysis for detecting arrays of equally spaced dots (SMITH; SMITH, 2005). The overall detection procedure begins by extracting blobs and then computing their center of mass. After that, the correspondences between the blobs and the dots of the array are established by a process that starts by the nearest blob to the center of the image (the region with minimal radial distortion). This process repeatedly utilizes regression analysis to predict the position of the blob to correspond to the next dot of the array; the process ends by reaching the edge of the image. Despite not discussed by the authors, the approach proposed by Smith and Smith has issues. Firstly, since they have used a simplistic thresholding technique for extracting blobs, the proposed detector can only succeed in evenly illuminated scenarios. Additionally, the procedure used for establishing correspondences does not consider that some of the extracted blobs may be outliers. By the way, the sample image used for experimentation (refer to the paper) seems to be acquired from a carefully crafted scenario so that the

mentioned issues were never a problem.

Kang et al. have detected regular patterns of circular features for camera calibration purposes (KANG; HA; JEONG, 2008). They utilized the Otsu global thresholding technique (OTSU, 1975) for selecting feature candidate regions. Since their system executed in an industrial environment with uneven illumination and with the presence of occluders, the image binarization stage was subject to miss many inliers. To overcome this limitation, they supported partial detections of the pattern, which was achieved by selecting a reliable subset of the features and estimating a transform from the pattern plane to the camera projection plane (this transform has ultimately aided the detection of the remaining features). It is worth noting that this approach was only possible because of the pattern regularity (an evenly spaced rectangular grid). Later, Kang et al. built upon their work (KANG; LEE, 2010) by employing the Sauvola's adaptive thresholding technique (SAUVOLA; PIETIKÄINEN, 2000). They accelerated the Sauvola's technique by utilizing integral images.

Yu et al. have detected patterns of circular features displayed in LCD monitors (they focused on the asymmetric pattern of the OpenCV library). Their pattern detection procedure can be roughly divided into three stages (YU et al., 2011). Firstly, it extracts feature candidates by smoothing the input image with a Gaussian filter for reducing noise; thresholding the smoothed image with an adaptive algorithm; obtaining enclosure regions; calculating several statistics from the regions (such as area, radius, etc.); and filtering the regions to obtain the final set of feature candidates. Secondly, the vicinity information is used to evaluate a similarity metric between candidates and features, which produces an initial correspondence set. Finally, a RANSAC (RANdom Sample And Consensus) algorithm refines the initial correspondence set while the camera parameters are estimated. The technique presented by Yu et al. has some issues that must be highlighted. First, since the estimated camera parameters are used for detecting new features, the pattern detection is dependent on the camera calibration process. In addition, in spite of the testing scenarios always depicting patterns with good contrast (after all, LCD monitors have light of their own), their technique was not always able to detect every feature of the pattern.

Wang et al. have presented a coarse-to-fine approach (WANG; KOBAYASHI; SAKUMA, 2015) for detecting dot array markers. Each marker is framed by a rectangular shape for facilitating their initial detection (the goal of the coarse stage). The first step performed during the coarse stage is to smooth the input image with a Gaussian filter. Next, a thresholding operation is used for binarizing the smoothed image (the authors did not expound on the thresholding operation, they limited to mention that a "simple threshold" was used). Afterward, as an attempt to remove outliers, morphological operations (opening and closing) are performed in the binarized image. The contours of the remaining blobs

are then extracted in a hierarchical manner, which permits identifying holed blobs and knowing what blobs are inside them. The marker is finally detected by searching for the holed blob that contains the same amount of internal blobs as the number of dots in the marker. Given the initial estimation of the dots contours obtained by the coarse stage, the fine stage finds out the dots edges with sub-pixel precision by searching the zero-crossing in the convolution of the marker image with a LoG (Laplacian-of-Gaussian) kernel. The fine stage ends by fitting ellipses to the edge points and evaluating their centers. The chief issue with the Wang and coworkers' approach is that its coarse stage is clearly subject to fail in more challenging scenarios (because of the simplistic feature extraction procedure). Not to mention that the patterns used by this approach have the downside of requiring some specificities (such as the surrounding frame).

Bergamasco et al. have proposed a pattern characterized by a circular arrangement of dots that are at fixed angular positions in one or more concentric rings (BERGAMASCO et al., 2011). This design has allowed the projective properties of both the atomic dots and the rings to be beneficially exploited. In addition, Bergamasco et al. have chosen to employ the mathematical framework of coding theory for empowering the patterns with error-correcting capabilities. Experiments conducted with synthetic images have shown that both of these features allowed the technique to achieve a strong occlusion resilience. The main concern with the Bergamasco and coworkers' approach is that it does not take into account non-linear effects such as radial and tangential distortions. Since these effects invalidate the assumed pinhole camera projective properties, it is supposed that such distortions were previously removed from images for the experiments conducted with real cameras (another option is that the distortions were negligible).

Habacher et al. have designed a pattern wherein the features (which are circular) are disposed on two parallel planes (HABACHER; HARKER; O'LEARY, 2014). This arrangement has enabled them to calibrate a camera from a single image. However, the scenario presented in their work is overly simplistic — they conveniently placed LED lamps close to the camera and utilized retro-reflectors for creating the features of the pattern. As might be imaged, these circumstances made the features highlight from other regions of the image turning the features extraction problem trivial.

2.3 Summary

This chapter presented the most relevant works that are related to the technique proposed in this Thesis. For the extraction of blobs (the type of low-level feature in which this Thesis is interested), the most relevant works that have been found in the literature are the one presented by Lindeberg and Eklundh (LINDEBERG; EKLUNDH, 1990); and the one introduced by Matas and coworkers (MATAS et al., 2004).

Considering the overall problem of detecting patterns formed by roundish features, none of the techniques found in the literature allows the same level of customization of the pattern as the proposed one — most of them are restricted to rectangular shaped patterns formed by regularly-spaced features. Another limitation observed in these techniques is that they utilize generic approaches for extracting low-level features (adaptive thresholding procedures are the preferred ones), thus prohibiting their application to scenarios with limited lighting conditions.

3 PROPOSED TECHNIQUE

This chapter presents the proposed detection technique in details. This technique achieved significant results at detecting patterns in images captured under low-light conditions (which reduces the image contrast) and with uneven illumination distribution. It also succeeds when the pattern is visually deformed due to radial and perspective distortions. These virtues were not achieved by chance, but they are the result of the conscious decisions taken during the technique development, which are presented here.

3.1 Overview

In the literature, there is a myriad of scenarios where researchers approached the pattern detection problem. Since each scenario has its own characteristics, unsurprisingly distinct solutions have been proposed over the years. At first glance, developing specific solutions for different instances of the same problem would not seem the ideal manner of facing a problem. However, this is the natural way of achieving the best results because it permits the peculiarities of each scenario to be beneficially exploited. Note that this does not mean that developing solutions that are more general is less important than developing specialized ones, neither vice versa, but that there is a trade-off between the generality and the overall performance of a technique.

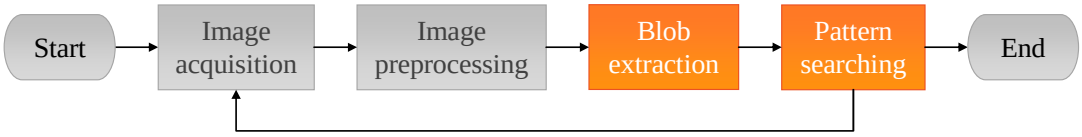
This Thesis addresses the pattern detection problem by prioritizing the performance side of the trade-off (i.e., it relies on some specificities of the scenario in order to achieve the best results). In addition, if performance was not impaired, advances in generality are also targeted as a second priority. For example, instead of relying on the usual rectangular grids of evenly spaced features, this work employs a more generic mechanism for representing the patterns (Section 1.2), which enables the creation of patterns with more diverse forms.

As stated in the introductory chapters, this Thesis is focused on patterns formed by roundish features. These patterns are usually man-made objects with known dimensions. However, in opposition to what most would think, these patterns are not restricted to be printed out on paper, as will be shown in the case study presented in Chapter 6.

The proposed detection technique is composed of two sequential stages. In the first stage, regions that are potential features of the pattern (the so-called blobs) are extracted from the image. In the second one, a search is performed for finding the blobs that actually compose the pattern of interest.

The diagram shown in Figure 9 illustrates a fictional computer vision system that implements the proposed technique. The boxes highlighted in orange represent the

detection technique stages. The two initial stages of the system (in gray) are in charge of preparing the input image for the subsequent stages. The image acquisition stage may rely on different methods for obtaining images. For instance, it may access a frame grabber for acquiring a live stream of video or it may load a recorded video from a file. While the image acquisition stage is mandatory for this system to work, the image preprocessing one is optional as the detection technique can operate over raw images. However, since even simple preprocessing procedures (such as a low-pass filter) may benefit the overall system operation, they are recommended. Results presented in Chapter 5 and in Chapter 6 were obtained by employing a minimalist preprocessing procedure for enhancing the image signal-to-noise ratio (see Section 4.1).



Source: The author

Figure 9 – The execution diagram of a system that implements the proposed technique. After the system is started, it enters a loop and repeatedly processes images until the execution ends.

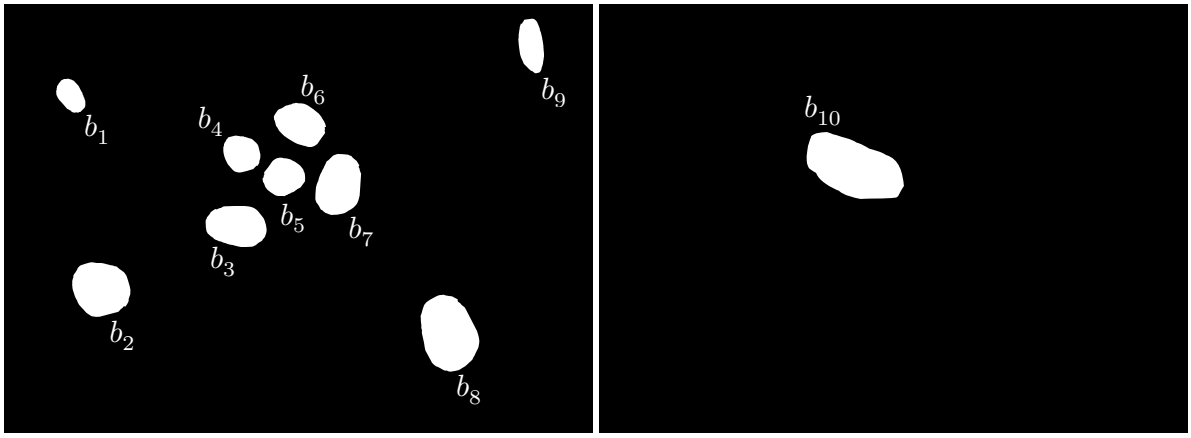
3.2 Blob extraction

The goal of this stage is to extract from the input image elements that can represent pattern features. These elements are called *blobs* and are considered candidates for the pattern features.

Ideally, this stage should only extract the blobs that arise from features of the pattern (these blobs are termed as *inliers* or as *true positives*). However, since at this stage the only known information is that the elements of interest are rounded shapes, it is natural to extract some blobs caused by elements other than the pattern features; these are termed as *outliers* or as *false positives*. On top of these, a central idea concerning this stage is to admit *false positive* errors in favor of avoiding *false negative* ones. In other words, it is preferred extracting outliers than discarding inliers. This choice is justified by the fact that, after this stage, the technique does not have mechanisms for recovering missed inliers. Consequently, any overlooked inlier will be permanently lost.

The result of this stage is denoted by the set of blobs $\mathfrak{B} = \{b_i\}_{i=1}^{n} = \{b_1, b_2, \ldots, b_n\}$, where each blob b_i is a connected component over the image domain. As previously mentioned, it is acceptable that $\mathfrak{B}$ contains outliers. Additionally, the set $\mathfrak{B}$ may contain overlapping blobs (i.e., for two distinct blobs b_i and b_j it is possible that $b_i \cap b_j \neq \emptyset$). As it will be seen later in this section, for any two overlapping blobs b_i and b_j either

$b_i \subset b_j$ or $b_j \subset b_i$. From the practical standpoint, the set $\mathfrak{B}$ can be represented by binary images. If the set contains only non-overlapping blobs, then a single image is enough for representing the set; otherwise, multiple images are required. These definitions are illustrated in Figure 10.



Source: The author

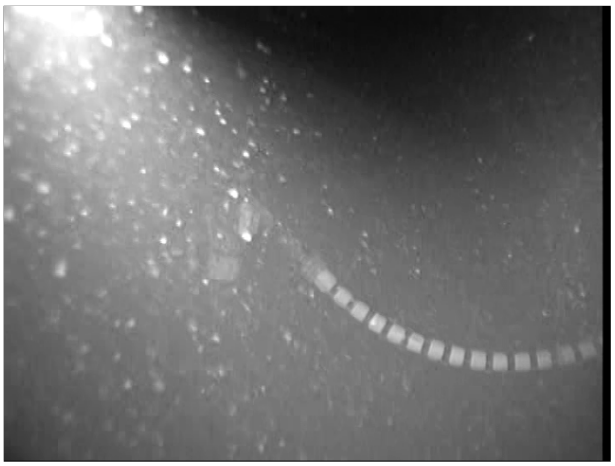
Figure 10 – Illustration of the set of blobs $\mathfrak{B} = \{b_1, b_2, \ldots, b_{10}\}$. Since b_{10} overlaps b_4 and b_5, b_{10} has to be represented in a second binary image.

The technique used in the blob extraction stage was conceived to be robust, addressing non-uniform illumination. This characteristic was set as a goal because the natural lighting conditions of some scenarios are not enough to evenly illuminate the scene. Moreover, these scenarios may also restrict the placement of artificial illuminators as a countermeasure against the natural lighting deficit. For example, in the case study presented in Chapter 6 the scenario is a deep underwater environment wherein no natural light can reach. Additionally, due to its high depth, remotely operated vehicles equipped with special camera systems have to be used. Illumination comes exclusively from spotlights attached to these vehicles, which concentrates light energy in the regions near the camera. All these limitations make the captured images lack in quality, as depicted in Figure 11.

Another important characteristic of the proposed blob extraction technique is that it tends to extract each feature as a "disconnected blob". In other words, any two distinct features of the pattern are unlikely to be extracted as a unique blob. This trend is observed even for close features separated by "shallow valleys" and is what makes the technique suitable for low-contrast images.

3.2.1 Features and peaks

Before formally defining the concepts that underlie the blob extraction technique, some intuitive ideas about the features of a pattern and the impact they cause in a 1D image are presented. At the top of Figure 12 a 2D gray-scale image depicts four side-by-side



Source: Petrobras

Figure 11 – Image captured in a deep underwater environment with limited lighting conditions.

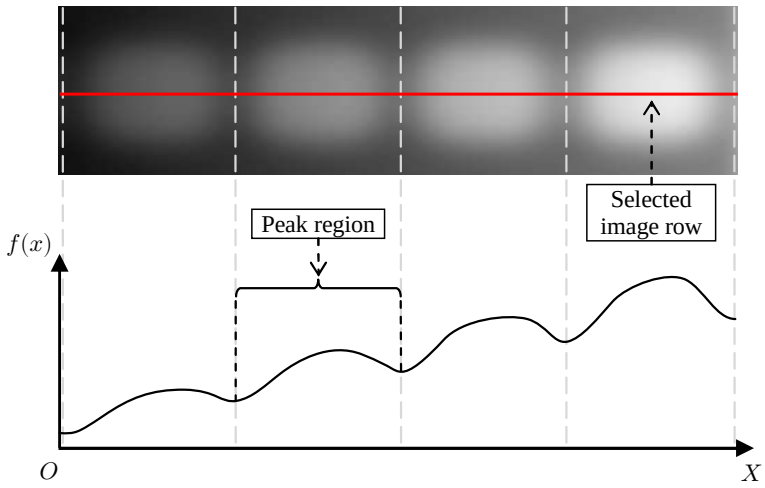
features (bright regions) under a non-uniform illumination condition. In this image, the red line highlights the chosen row to be used as the 1D image. The intensity of the pixels of this row can be modeled by a function whose plot is illustrated at the bottom of Figure 12.

The basic concept behind the proposed technique consists in finding peaks in this plot. Notice that wherever there is a feature of the pattern in the image, a peak appears in the plot. Therefore, one can assume that by collecting every peak from the image plot, every feature will be collected as well. As suggested by the example presented in Figure 12, this statement also stands true for images captured under uneven illumination conditions. It is evident that some peaks will arise from elements other than pattern features, but this is the preferred behavior at this stage (recall that false positive errors are tolerated in favor of avoiding false negative ones).

3.2.2 Component trees for 1D images

The blob extractor presented in this Thesis operates over a data structure referred to as *component tree* (JONES, 1997). This data structure is a tree-based representation for gray-scale images, which links the image connected components from different gray-levels. For a sake of simplicity, this data structure is firstly defined for 1D images. The 2D case is covered in Subsection 3.2.3. Finally, in Subsection 3.2.4, it is presented how blobs are extracted from component trees.

A 1D image can be denoted by the continuous function $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ (notice that pixels can assume only non-negative values). The non-negative region that is under the plot of this function is denoted by the set of points $\mathfrak{S} = \{(x, y) \in \mathbb{R}^2 : 0 \leq y \leq f(x)\}$. Let also a parallel line to the x-axis be $l_b = \{(x, y) \in \mathbb{R}^2 : y = b\}$, where b is its level. Since the image function is strictly non-negative, only lines such that $b \geq 0$ are considered.



Source: The author

Figure 12 – Illustration of how the features of a pattern translate into peaks. Notice that from left to right the overall pixel intensity gradually increases, suggesting the image was captured in a scenario with uneven illumination conditions.

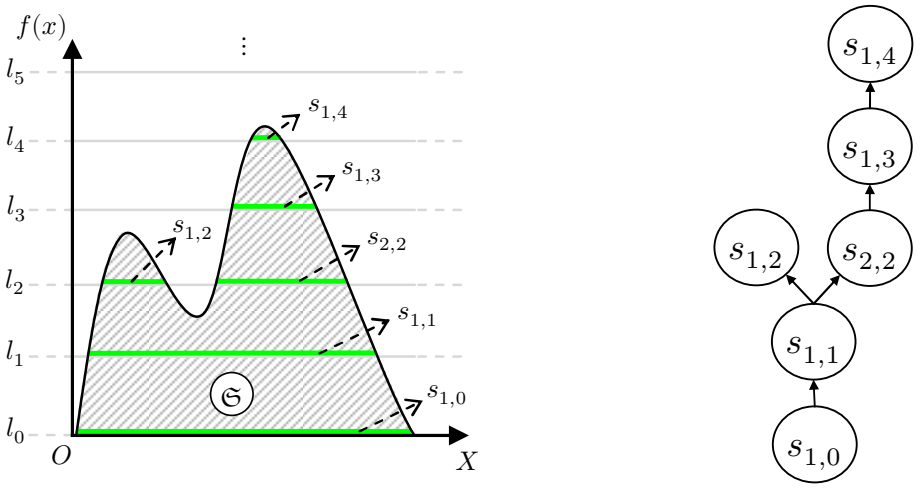
These lines slice the region $\mathfrak{S}$ by applying the slicing operation denoted by $\mathrm{S}(l_b) = \mathfrak{S} \cap l_b$, which produces a set of $n \geq 0$ connected components[1] $\mathrm{S}(l_b) = \{s_{1,b}, s_{2,b}, \ldots, s_{n,b}\}$ such that $s_{i,b} = \{(x, y) \in \mathbb{R}^2 : y = b\}$. The connected components generated by the slicing operation are denominated *slices*; therefore, one may say that $s_{i,b}$ is the i-th slice produced by line l_b. By considering the previous definitions, one can also say that $\mathrm{S}(l_0) = \{s_{1,0}\}$, where $s_{1,0} = l_0$ (i.e., the lowest line produces only a single slice that is equivalent to this line). Finally, let the projection of slice $s_{i,b}$ be $s'_{i,b} = \{x \in \mathbb{R} : (x, y) \in s_{i,b}\}$.

The presented slicing process has an interesting characteristic — for any $s_{i,b}$ with $b > 0$, there is one and only one $s_{j,b-1}$ such that $s'_{i,b} \subseteq s'_{j,b-1}$. For example, in Figure 13a: $s'_{1,1}$ is a subset of $s'_{1,0}$; $s'_{1,2}$ and $s'_{2,2}$ are subsets of $s'_{1,1}$; $s'_{1,3}$ is a subset of $s'_{2,2}$; and so on. Since this relationship can always be established between slices produced by consecutive lines, one can construct a tree for representing this hierarchy (Figure 13b). The root of this tree is always the slice $s_{1,0}$, which is produced by l_0. As might be imagined, this tree is the aforementioned component tree.

3.2.3 Component trees for 2D images

The definitions for 2D images are similar to the ones introduced for 1D images. A 2D image can be denoted by the continuous function $f : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$. The non-negative region that is under the surface plot of this function is denoted by the set of points $\mathfrak{S} = \{(x, y, z) \in \mathbb{R}^3 : 0 \leq z \leq f(x, y)\}$. The 2D case uses planes instead of lines for slicing the region $\mathfrak{S}$. A plane is denoted by $p_b = \{(x, y, z) \in \mathbb{R}^3 : z = b\}$, such that $b \geq 0$ is its level. The same previously defined chopping operation for 1D images is used in the 2D case, except

[1] For the 1D case, the connected components produced by the slicing operation are line segments.



Source: The author

Figure 13 – In (a), the region $\mathfrak{S}$ (hatched area) is sliced with a set of horizontal lines (slices produced are depicted in green). The lines l_0, l_1, l_3, and l_4 produce only one slice, each. The line l_2 produces two. Finally, the line l_5 does not produce any slice. In (b), the respective component tree.

that the 2D version of this operation receives a plane instead of a line, giving $\mathrm{S}(p_b) = \mathfrak{S} \cap p_b$. The result of this operation[2] is a set of $n \geq 0$ slices $\mathrm{S}(p_b) = \{s_{1,b}, s_{2,b}, \ldots, s_{n,b}\}$ such that $s_{i,b} = \{(x, y, z) \in \mathbb{R}^3 : z = b\}$. The property $\mathrm{S}(p_0) = \{s_{1,0}\}$, such that $s_{1,0} = p_0$, also applies here. Finally, the projection of a slice $s_{i,b}$ is $s'_{i,b} = \{(x, y) \in \mathbb{R}^2 : (x, y, z) \in s_{i,b}\}$.

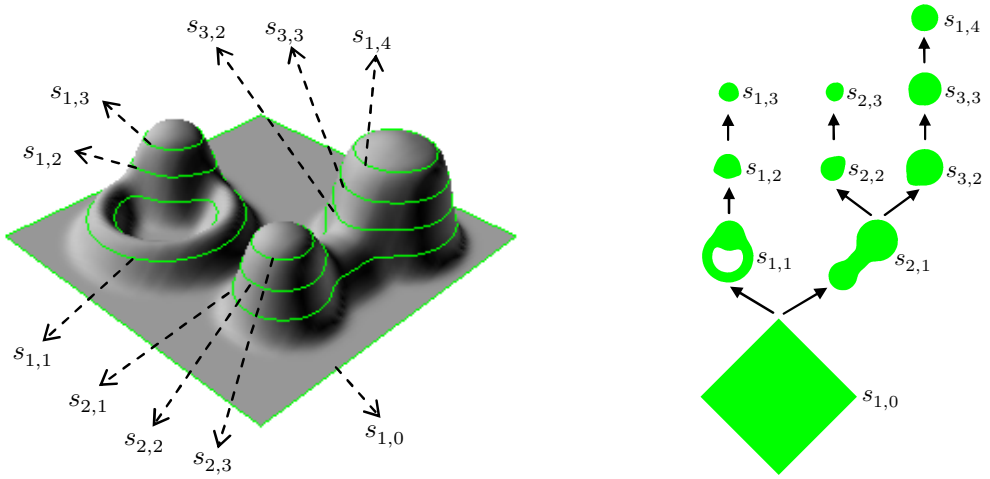
The slices produced from 2D images have the same properties of the ones produced from 1D images (i.e., the projection of upper slices are subsets of the projection of lower ones). Therefore, the component tree of 2D images can be constructed by following the same steps previously described for 1D images. Figure 14 illustrates the component tree construction of a 2D image.

3.2.4 Extracting blobs from component trees

For now, an alternative representation of the input image was built — the component tree. This representation allows the image to be traversed in a fashion that goes beyond a merely per-pixel sampling. Such a traversing method is more appropriate for the feature extraction problem as it makes the peaks extent explicit during the image analysis. Details of how the notion of a peak is translated into a blob are presented next.

In short, the goal of this subsection is to establish a selection criterion for choosing the nodes (or more precisely, the slices associated with them) of the component tree to be transformed into blobs. This criterion is expected to select appropriate nodes so that

[2] For the 2D case, the connected components produced by the slicing operation are closed regions.



Source: The author

Figure 14 – In (a), the surface plot of a 2D image. The slices highlighted in green were produced by applying the slicing operation with the planes p_0, p_1, p_2, p_3, and p_4 (these planes are not portrayed in the figure to make a cleaner view of the slices). In (b), the constructed component tree.

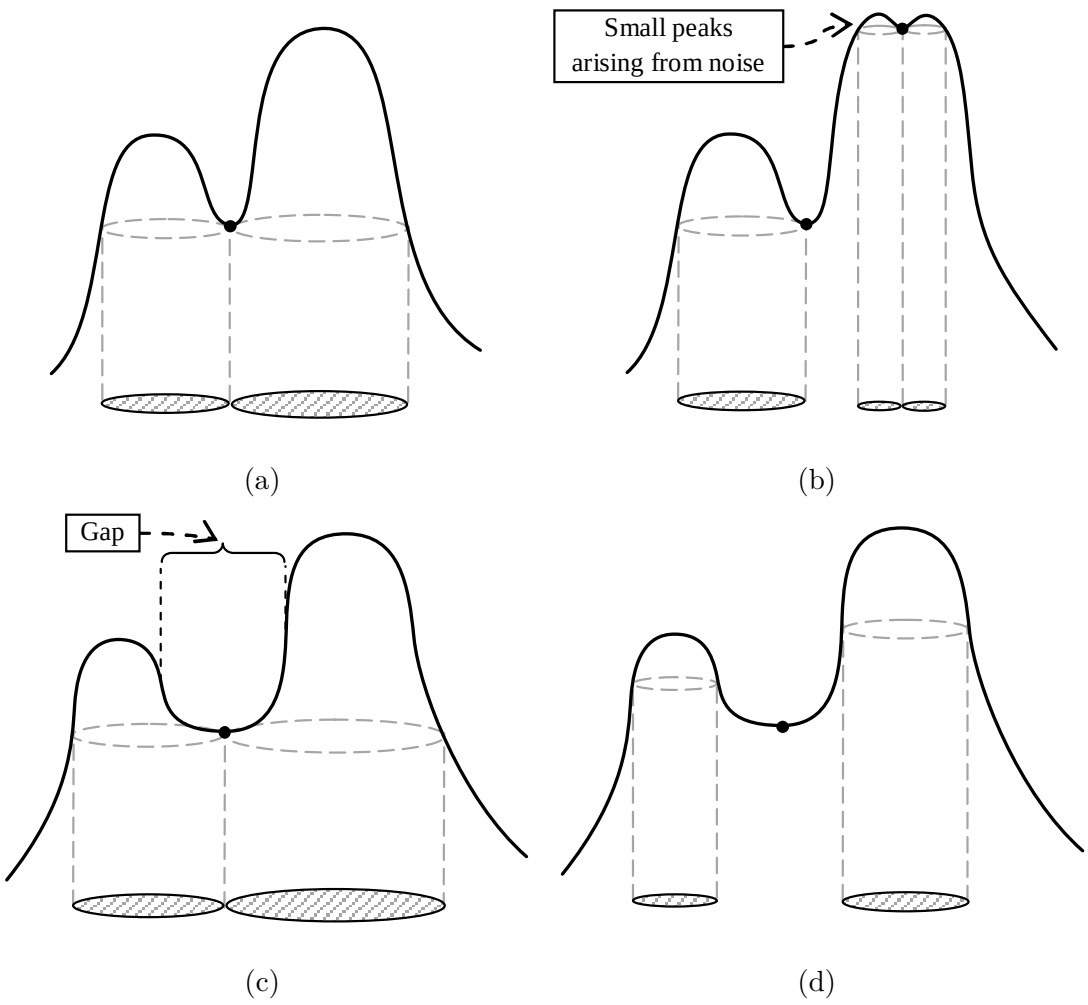
representative blobs are created.

The proposed technique combines concepts from both Lindeberg and Eklundh's work (LINDEBERG; EKLUNDH, 1990); and Matas and coworkers' (MATAS et al., 2004). While the former proposed extracting the base slices of the topmost peaks (following a top-down path, the base slice of a peak is the slice that occurs immediately before a merge between peaks), the latter preferred the maximally stable ones.

Lindeberg and Eklundh's approach has two remarkable drawbacks. 1) It tends to miss inlier blobs (especially if images are noisy). In a noise-free image, the first peaks to be traversed (the topmost ones) are probably the peaks produced by the pattern features, thereby inlier blobs are extracted in such a situation (Figure 15a). However, when noise comes on the scene undesirable results may occur. Close peaks arising from noise neutralize each other preventing the traversal from reaching deeper levels wherein inlier blobs are (recall that the Lindeberg and Eklundh's approach regards only the topmost peaks). Consequently, outlier blobs are extracted in place of inliers (Figure 15b). 2) The Lindeberg and Eklundh's approach also produces oversized blobs (Figure 15c) when the gaps between peaks are comparable to or greater than their diameter — a possible situation considering that a pattern with more spaced features may be required depending on the application. Better representative blobs could be extracted if the upper slices were selected instead (Figure 15d).

As previously stated, Matas et al. proposed selecting the maximally stable slices[3]

[3] In their original work the term *extremal regions* was used instead of the word *slice*.

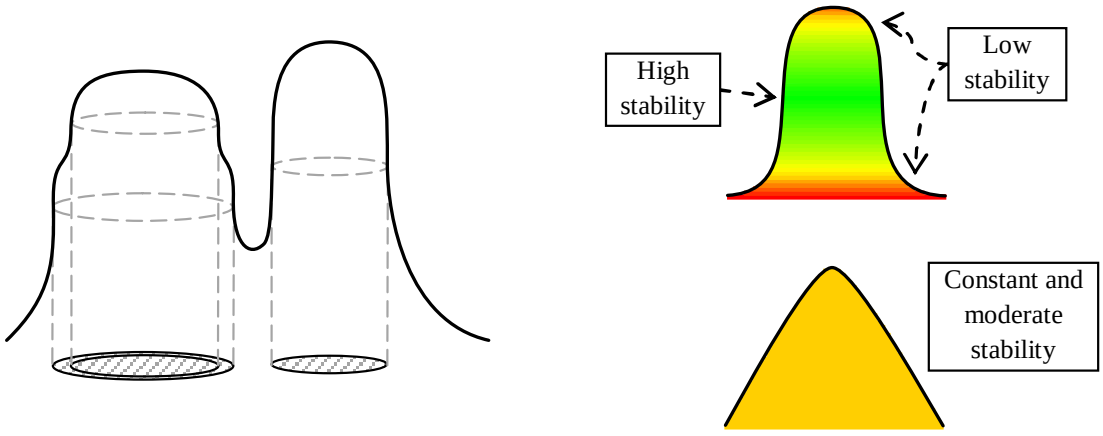


Source: The author

Figure 15 – In (a), an image surface plot depicting two peaks that arise from features of a pattern. The hatched regions beneath the plot illustrate the blobs extracted by the Lindeberg and Eklundh's approach. In (b), noise is added producing two small peaks upon the bigger peak on the right. Notice that in this situation two outlier blobs were extracted in place of the inlier one. Now in (c), the size of the gap between the two peaks is substantially greater. As it turns out, oversized blobs are extracted even though better representative ones could be taken instead (d).

of the component tree. By a "stable slice" one means a slice that is alike to others over a range of nearby levels. In other words, the stability of a slice is directly proportional to the similarity of this slice with its upper and lower neighbors (Figure 16b). This approach follows extracting blobs whenever a slice presents local maximum stability. As it turns out, multiple blobs may be extracted from a single peak even though they are alike each other (Figure 16a). The drawback here is that multiple equivalent blobs are output causing the number of false positive results to increase.

The proposed selection criterion combines the best of the previously described



(a) (b)

Source: The author

Figure 16 – The right peak in (a) has only one slice with local maximum stability, resulting in a single blob extraction. However, in the left peak, there are two slices with such a property. Notice that the two slices extracted are sufficiently similar to justify the discarding of one of them. In (b), the general notion of stability for slices from different portions of two peaks.

works for achieving better results (at least for the kind of pattern being tackled by this Thesis).

The first borrowed idea comes from the Lindeberg and Eklundh's approach, which relies on the occurrence of merges between peaks for signalizing the extraction of blobs. This is a good approach as merges between peaks are good hints about the features extent (in the intensity domain) and this notion can be used for preventing the extraction of "redundant blobs" (such as in the example presented in Figure 16a). However, in order to overcome the noise sensibility problem presented by the Lindeberg and Eklundh's approach, the technique proposed in this Thesis does not stop the component tree traversal after reaching a merge node (merge nodes are depicted in Figure 17), that is, traversal continues towards the root of the tree for extracting blobs from lower levels. The minimum level reached during the traversal depends on a user-provided parameter that limits the number of merge nodes permitted to be overleapt.

The other idea comes from the Matas and coworkers' approach, which uses a stability measurement for picking out those slices that can produce better representative blobs. This measurement is useful for reducing the number of outliers as well. The adoption of this stability measurement relied on the fact that the peaks arising from the features of a pattern are likely to have a region wherein high stability slices exist (the upper peak in Figure 16b portrays a typical peak that arises from pattern features). This is an expected outcome since the patterns of interest of this Thesis are constructed with contrasting colors giving sharp edges to the features. Therefore, if a peak does not have at least a

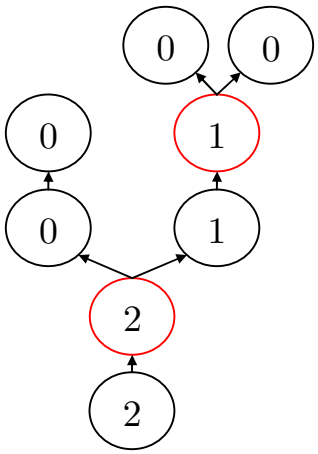
short region of stability (such as the one at the bottom of Figure 16b), than it is likely to produce an outlier blob and thereby may be discarded.

The *stability* of a slice $s_{i,b}$ is formally defined by the expression

$$\mathrm{Q}(s_{i,b}) = \frac{\textsc{Area}(s'_{j,b+\Delta})}{\textsc{Area}(s'_{i,b})}. \tag{3.1}$$

From the component tree perspective the slice $s_{j,b+\Delta}$ is a descendant node of the slice $s_{i,b}$ (i.e., $s'_{j,b+\Delta} \subseteq s'_{i,b}$). The parameter Δ determines how far these slices are from each other in the level dimension. As might be imagined, the function Area evaluates the area of a slice projection. The stability metric Q assumes values in the range $[0, 1]$ (zero for minimum stability and one for maximum stability). Additionally, leaf and merge nodes are assumed to have stability equal to zero by definition.

Another metric used for restricting blobs extraction is the so-called *mergewise height*, which is denoted by M. This is the metric in charge of controlling how deep the component tree traversal will reach. This metric is evaluated for each node and, as its name suggests, it is related to the height of a node. The height of a node (not the mergewise height) is defined as the number of edges on the longest path between that node and a descendant leaf. The mergewise height metric is similarly defined; however, it relies on a per merge node analysis instead of a per edge analysis. The mergewise height of a node is the maximum number of merge nodes found in a path from that node to a descendant leaf. Figure 17 illustrates this definition.



Source: The author

Figure 17 – Tree exhibiting the mergewise height of all nodes. Merge nodes are in red.

From the practical standpoint, blobs are extracted from the component tree while it is traversed using a *depth-first search*. Nodes are processed in the post-order manner (i.e., it is as if traversal was from leaves towards the root) and during this processing the stability (Q) and the mergewise height (M) metrics are evaluated for each slice/node. Whenever traversal visits a merge node, the extraction of blobs is triggered. In this situation, there are

at least two segments of branch from which blobs can be extracted. Each of these segments starts with a descendant of the merge node and ends either: in the closest descendant node that is also a merge node; or in a leaf node. A maximum of one blob is produced per branch segment. However, two conditions must hold true for the extraction of a blob to materialize:

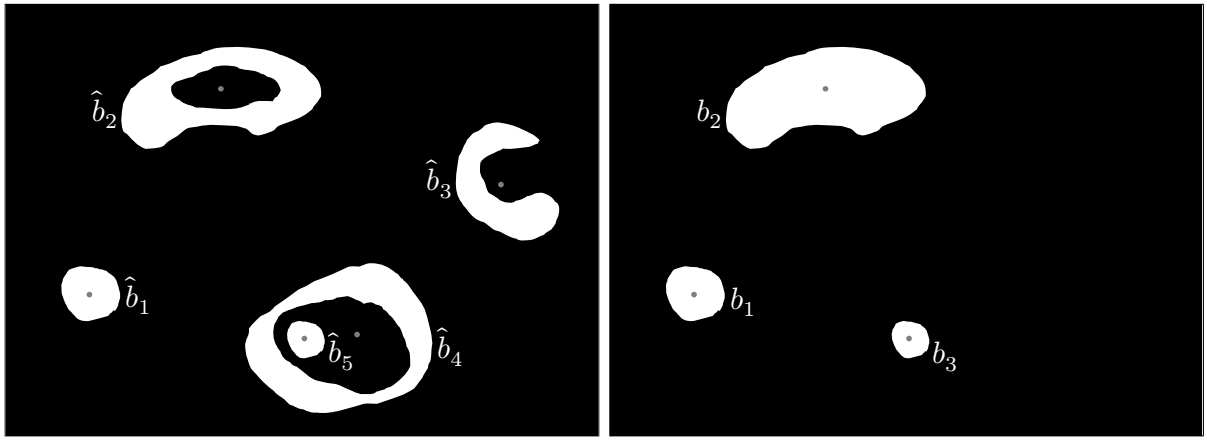
1. The mergewise height of the merge node shall not be greater than the user parameter kMaxM; and,
2. The stability of the maximally stable slice in the branch segment shall not be lesser than the user parameter kMinQ.

Provided that these conditions are satisfied, a blob is derived from the maximally stable slice.

As a final observation, two tests regarding the shape of the derived blob are performed before it is inserted into the final set. Until now, shape-related tests were avoided throughout the blob extraction procedure. However, some pieces of evidences regarding the blobs shape can be reliably used for further mitigating the occurrence of false positives and improving the quality of the extracted elements.

The first test consists of evaluating whether the centroid of the blob is outside. If so, this blob is not inserted into the final set. Something to notice here is that this test regards only the outer contour of the blob. That is, even if a blob has inner contours (i.e., it has holes) and its centroid is inside one of these contours, the centroid will be considered to be inside that blob (see $\hat{b}_2$ in Figure 18 for an example of such a blob). In addition, this blob has its holes filled before it is inserted into the final set. The following reasoning motivated these decisions: the features of the pattern do not have holes and thereupon they are expected to produce blobs without holes; however, if a blob with holes is produced, then this blob may be the result of a pattern feature that suffered from undesirable artifacts (such as noise); in that case, that blob is inserted into the final set for not risking discarding inliers and its holes are filled so that its (predicted) original shape is restored.

The second test is concerned with holed blobs that possess another blob inside them ($\hat{b}_4$ in Figure 18 exemplifies such a blob). These blobs are also discarded because they are considered overly complex to be produced by pattern features; a typical peak produced by a pattern feature will hardly produce such blobs arrangement. A pseudocode of the proposed blob extraction technique is in Algorithm 1.



(a) (b)

Source: The author

Figure 18 – The slices depicted in (a) were selected during the tree traversal for deriving blobs. However, due to the two shape-related tests, some of the derived blobs are not introduced into the final set shown in (b). In this example, blob $\hat{b}_4$ was discarded because it has an inner blob; $\hat{b}_3$ was discarded because of its outer centroid. Notice also that $\hat{b}_2$ had its hole filled before its insertion into the final set.

3.3 Search for the pattern of features

This section addresses the second (and final) part of the problem. It was originally stated (see Section 1.2) as the creation of a mapping $\mathrm{M} : V \to A$ that relates each vertex of the pattern template with a connected component of the input image (V denotes the set of vertices of the template and A denotes the set of all connected components over the image domain). The main issue with this definition is that it poses a complex problem because it involves checking all possible connected components of the image. Simply put, the set A is a too large search space to be explored in feasible time, and that is the reason why the proposed solution was divided into two stages — while the blob extractor (in the first stage) resorts to a fast heuristic for discarding connected components that clearly fail in representing pattern features, the search (in the second stage) is benefited by having a reduced set of elements to analyze. All things considered, the final mapping to be created has a much smaller searching space, resulting in $\hat{\mathrm{M}} : V \to \mathfrak{B}$.

The blobs in $\mathfrak{B}$ do not have sufficient local information (e.g., shape and size) for securely guiding the creation of the desired mapping — after all, pattern features are equal-sized circles impossible to be differentiated. This limitation suggests that a "more global" approach is required instead. The proposed solution takes into account the relative placement between blobs as well as their relative sizes, but since only close blobs are regarded, it cannot be considered as a complete global approach. This intermediate approach turned out to produce the best trade-off; it restricts the pattern shape sufficiently

Algorithm 1 The component tree traversal algorithm that extracts blobs.

Input: *root* ▷ The component tree root node.
Output: $\mathfrak{B}$ ▷ The final set of blobs.

```
1: TRAVERSE(root)

2: procedure TRAVERSE(node)
3:   if node.DEGREE = 1 then ▷ Handles leaf nodes.
4:     node.Q ← 0 ▷ By definition leaf nodes have stability equal to 0.
5:     node.M ← 0

6:   else if node.DEGREE = 1 then ▷ Handles single-child nodes.
7:     TRAVERSE(node.CHILD)

8:     node.Q ← Q(node)
9:     node.M ← child.M

10:  else ▷ Handles merge nodes.
11:    max_m ← 0
12:    for all child ∈ node.CHILDREN do
13:      TRAVERSE(child)

14:      if max_m > child.M then
15:        max_m ← child.M

16:      if child.M ≤ kMaxM then
17:        max_q_node ← FINDMAXQNODE(child)
18:        if max_q_node.Q ≥ kMinQ then
19:          blob ← NODETOBLOB(max_q_node)
20:          if TESTSHAPE(blob) then
21:            𝔅.INSERT(blob)

22:    node.Q ← 0 ▷ By definition merge nodes have stability equal to 0.
23:    node.M ← max_m + 1

24: function FINDMAXQNODE(node) ▷ Finds maximally stable slice in the branch segment starting with node.
25:   result ← NULL

26:   if node.DEGREE = 1 then
27:     result ← FINDMAXQNODE(node.CHILD)
28:   else
29:     result ← node

30:   if node.Q ≥ result.Q then ▷ The lower node is preferred when stabilities are equal.
31:     result ← node

32:   return result
```

to prevent mistaken detections, while it tolerates the deformities caused by radial and perspective distortions.

The searching approach used for creating the mapping $\hat{\mathrm{M}} : V \to \mathfrak{B}$ relies on a backtracking algorithm. This searching is constrained by the graph that represents the pattern of features; topological and geometrical aspects of the graph are regarded during the search.

3.3.1 Backtracking

Intuitively speaking, a backtracking algorithm works by incrementally building solution candidates, which are abandoned (backtracked) as soon as it determines that a candidate cannot be completed to a valid solution. A pseudocode of the implemented backtracking algorithm is in Algorithm 2.

The backtracking proceeds by visiting all vertices in V; for each vertex v_j visited it searches for a blob b_i in $\mathfrak{B}$ to be mapped to. A blob b_i is mapped from a vertex v_j if two conditions are satisfied: 1) b_i is not yet mapped from other vertex and 2) b_i respects the constraints imposed by v_j and its neighbors that already have blobs mapped to. Notice that, despite the constraints are ultimately determined by the topological and the geometrical characteristics of the graph that represents the pattern (which are statically defined), there is also a dynamic factor playing a role — the mapping state of the vertices (more details about these constraints are in next subsection).

After mapping v_j to b_i, the algorithm follows to the next vertex and repeats the same blob searching process for the new vertex. On the case that there are no blobs that could be mapped from a given vertex (i.e., all blobs failed in satisfying the above conditions), the algorithm steps back (backtracks) to the previous vertex and the searching for another satisfiable blob continues from where it had stopped. Under favorable conditions, this process is repeated until all vertices are mapped; that is, the pattern is detected in its entirety. Otherwise, the algorithm concludes with the best partial solution found during the search[4].

Despite the backtracking algorithm would succeed by visiting the vertices in V in any order, by carefully choosing this order one may spare some computing time. The idea is to choose an order that favors premature backtrack events when non-promising solution paths are taken, such as by first visiting vertices with stronger constraints. At the earliest stage of the searching (i.e., while selecting the first vertex for visiting) all vertices are equally good choices because all their neighbors are not mapped to blobs yet. This condition also nullifies the tested constraints making the first selected vertex mappable to any blob, regardless of the blob positioning and shape.

On the other hand, for all vertices other than the first one, the visiting order affects performance; significant improvements can be observed even with the adoption of non-strict ordering relations. For example, considering that the first vertex is already selected, a good option for the second vertex would be any neighbor of the first vertex, which would force the candidate blobs to satisfy the constraints imposed by the first vertex before they would be mapped from that vertex. The same reasoning applies for choosing the third vertex to be visited — the best option in this situation would be to opt for

[4] Partially detecting patterns is useless for most scenarios, but since it is essential for the case study described in Chapter 6, it was considered in this Thesis.

any vertex that is simultaneously neighbor of the first and the second vertices, which maximizes the number of constraints tested. Given these points, a BFS (Breadth-first Search) was used for guiding the backtracking.

3.3.2 Constraints

Except by the first vertex (which is trivially mapped to any blob), the mapping between a vertex and a candidate blob is conditioned to the successful meeting of some constraints by this candidate. These constraints are tested regarding the vertices that are at maximum two edges of distance from the current vertex being mapped (this short-range inspection is the reason why it was considered a semi-global approach). Measurements of angles and distances are evaluated for those vertices (using the space wherein the graph was defined) and for the blobs related to them (using the image space). The mapping is materialized if and only if all these measurements are compatible.

In the pseudocode of Algorithm 2 all the constraints that permit (or not) the creation of a mapping from vertex v to blob b are encapsulated in the function TestConstraints(v, b). The first three constraints tested take into account only the current vertex being mapped and its neighbors. The last two constraints additionally inspect the neighbors of the neighbors of the current vertex being mapped. Figure 19 depicts a convenient example for illustrating the used constraints.

The first constraint limits the distance between the centroid of the current candidate blob and the centroid of the blobs mapped from the neighbors of the vertex being mapped. For two blobs b_i and b_j this distance is denoted by the function Distance(b_i, b_j), which is also used for measuring the distance between two vertices in the pseudocode (refer to Figure 20a for an illustration of this measurement). The goal of this constraint is to discard prematurely candidate blobs that are too apart from the ones already mapped to. The parameter kMaxDist (specified in pixels) controls this constraint. In the example shown in Figure 19, if the distance from b_4 to b_5 or the distance from b_4 to b_3 is greater than kMaxDist, then b_4 is promptly discarded and the algorithm iterates over the next blob.

The second constraint forbids that a candidate blob separated by a large gap be mapped to the current vertex. This constraint regards the fact that close features form the patterns of interest of this Thesis. The measurement used by this constraint is the ratio of the length of the gap in between two blobs to their (centroids) distance. This ratio is expected to be lesser than the parameter kMaxGapDistRatio, which ranges in the interval $[0, 1]$. The inter-blob gap between blobs b_i and b_j is denoted by Gap(b_i, b_j) (refer to Figure 20a). In the example shown in Figure 19, if the ratio measurement for b_4 and b_5 or the ratio measurement for b_4 and b_3 is greater than kMaxGapDistRatio, then b_4 is discarded.

Algorithm 2 Pseudocode of the pattern searching algorithm.

Input: $\mathfrak{B}, G$ ▷ The set of blobs and the graph that represents the pattern.
Output: $\hat{\mathrm{M}}$ ▷ The resultant mapping.

1: *temp* ▷ A temporary mapping to be worked on.

2: **for all** $b_i \in \mathfrak{B}$ **do**
3: $v_1 \leftarrow G$.FirstVertex
4: *temp*.Map(v_1, b_i) ▷ Starts the temporary mapping building by mapping v_1 to b_i.
5: *pattern_complete* $\leftarrow$ Process(G.NextVertexBfs(v_1)) ▷ Get the next vertex following a breadth-first search order and processes it.
6: **if** *pattern_complete* **then break**
7: *temp*.Unmap(v_1)

8: **function** Process(v_j)
9: **if** $v_j = G$.End **then return TRUE** ▷ If all vertices were mapped, the pattern was completely detected.

10: **for all** $b_i \in \mathfrak{B}$ **do**
11: **if** *temp*.IsMapped(b_i) **then continue**

12: *success* $\leftarrow$ TestConstraints(v_j, b_i);
13: **if** *success* **then**
14: *temp*.Map(v_j, b_i)
15: *pattern_complete* $\leftarrow$ Process(G.NextVertexBfs(v_j)) ▷ Returning from this function is the same as backtracking.
16: **if** *temp*.Size $>$ $\hat{\mathrm{M}}$.Size **then**
17: $\hat{\mathrm{M}} \leftarrow$ *temp* ▷ Updates output with the best solution found so far.
18: **if** *pattern_complete* **then return TRUE**
19: *temp*.Unmap(v_j)

20: **return FALSE**

21: **function** TestConstraints(v, b)
22: **for all** $v_i \in G$.Neighbors(v) **do**
23: $b_k \leftarrow$ *temp*.GetMapped(v_i)
24: **if** $b_k =$ **NULL then continue**

25: *dist* $\leftarrow$ Distance(b, b_k)
26: **if** *dist* $>$ kMaxDist **then return FALSE**

27: *gap* $\leftarrow$ Gap(b, b_k)
28: **if** *gap*/*dist* $>$ kMaxGapDistRatio **then return FALSE**

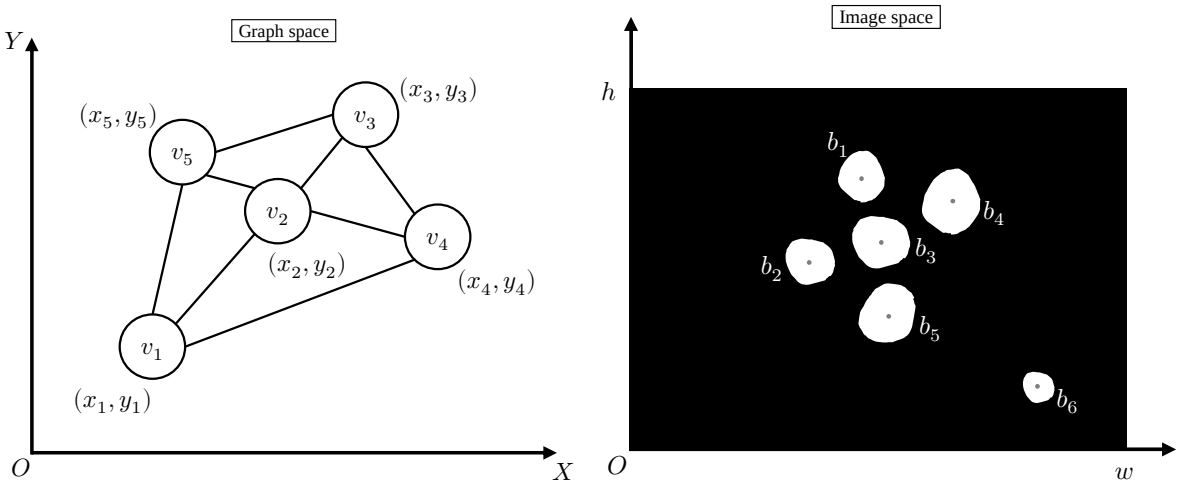
29: *radius_b_b_k* $\leftarrow$ Radius(b, b_k)
30: *radius_b_k_b* $\leftarrow$ Radius(b_k, b)
31: *radius_ratio* $\leftarrow$ *radius_b_b_k*/*radius_b_k_b*
32: *max_radius_ratio* $\leftarrow 1.0 +$ kMaxRadiusRatioPercent
33: *min_radius_ratio* $\leftarrow 1.0/(1.0 +$ kMaxRadiusRatioPercent$)$
34: **if** *min_radius_ratio* $<$ *radius_ratio* $<$ *max_radius_ratio* **then return FALSE**

35: **for all** $v_j \in G$.Neighbors(v_i) **do**
36: $b_l \leftarrow$ *temp*.GetMapped(v_j)
37: **if** $b_l =$ **NULL then continue**

38: *blobs_dist_ratio* $\leftarrow$ Distance(b, b_k)/Distance(b_k, b_l)
39: *vertices_dist_ratio* $\leftarrow$ Distance(v, v_i)/Distance(v_i, v_j)
40: *dist_ratio* $\leftarrow$ *blobs_dist_ratio*/*vertices_dist_ratio*
41: *max_dist_ratio* $\leftarrow 1.0 +$ kMaxDistRatioPercent
42: *min_dist_ratio* $\leftarrow 1.0/(1.0 +$ kMaxDistRatioPercent$)$
43: **if** *min_dist_ratio* $<$ *dist_ratio* $<$ *max_dist_ratio* **then return FALSE**

44: *blobs_angle* $\leftarrow$ Angle(b, b_k, b_l) ▷ Angle measured in the range $[0, 360)$ degrees.
45: *vertices_angle* $\leftarrow$ Angle(v, v_i, v_j)
46: *angle_diff* $\leftarrow$ AngleDiff(*blobs_angle*, *vertices_angle*)
47: **if** *angle_diff* $>$ kMaxAngleDiff **then return FALSE**

48: **return TRUE**



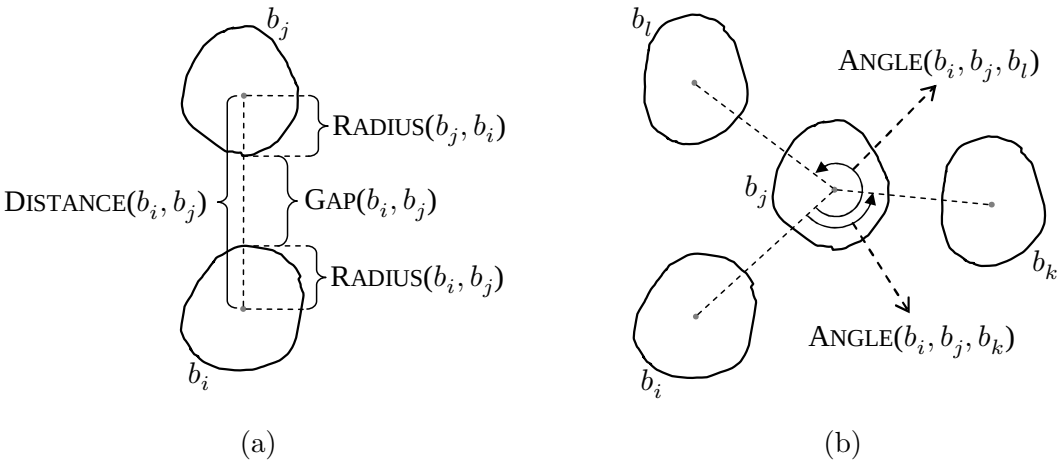
(a) (b)

- Visiting order: v_1, v_2, v_4, v_5, v_3
- Mapping state: $v_1 \rightarrow b_5$
  $v_2 \rightarrow b_3$
  $v_3 \rightarrow$ nil
  $v_4 \rightarrow$ nil
  $v_5 \rightarrow$ nil
- Current vertex being mapped/visited: v_4
- Neighbors of the current vertex that are mapped: v_1 and v_2
- Current candidate blob: b_4

Source: The author

Figure 19 – Example for illustrating the backtracking searching algorithm operation. The graph that represents the pattern is in (a). The pattern is considered detected when each vertex of this graph is mapped to a blob that is in (b). The constraints imposed by that graph restricts this mapping. The frame at the bottom shows the overall state of the searching algorithm at a given moment.

The third constraint determines a superior and an inferior limit for the radius of the candidate blob. The radius of a blob is a relative measurement taken from a blob of origin b_i to a blob of destiny b_j; it is denoted by $\textsc{Radius}(b_i, b_j)$ (see Figure 20a). The output of this function can be read as "the radius of b_i relative to b_j", so the measurement $\textsc{Radius}(b_i, b_j)$ is not the same as $\textsc{Radius}(b_j, b_i)$. The third constraint is controlled by the parameter kMaxRadiusRatioPercent, which is a percentage value ranging in the interval $[0, \infty)$. In the example shown in Figure 19, if the radius of b_4 relative b_5 is not compatible with the radius of b_5 relative to b_4; or if the radius of b_4 relative b_3 is not compatible with the radius of b_3 relative to b_4, then b_4 is discarded.



Source: The author

Figure 20 – The measurements DISTANCE, GAP, and RADIUS in (a) regards only two blobs (or vertices, when vertices apply). The ANGLE measurement in (b) regards three.

The fourth constraint checks the proportion between distance measurements taken from vertices and blobs. These measurements regard not only the neighbors of the current vertex being visited, but also the neighbors of the neighbors of it. This constraint should be clear from the example shown in Figure 19 — it is condition for the creation of the mapping $v_4 \rightarrow b_4$ and it is tested twice:

1. DISTANCE(b_4, b_3) must be to DISTANCE(b_3, b_5) as DISTANCE(v_4, v_2) is to DISTANCE(v_2, v_1); and
2. DISTANCE(b_4, b_5) must be to DISTANCE(b_5, b_3) as DISTANCE(v_4, v_1) is to DISTANCE(v_1, v_2).

The parameter kMaxDistRatioPercent is a percentage value that controls how permissive this constraint is to disproportionalities.

The fifth (and last) constraint takes into account the difference in angles formed by vertices and blobs. The function ANGLE(b_i, b_j, b_k) is used for evaluating the oriented angle subtended by b_i and b_k at b_j (it is also used for measuring angles subtended by vertices). This function outputs results in the range $[0, 360)$, as depicted in Figure 20b. This fifth constraint ultimately regards the shortest angular length between the angles a_b (formed by blobs) and a_v (formed by vertices), which is denoted by ANGLEDIFF(a_b, a_v). The parameter kMaxAngleDiff (which ranges in the interval $[0, 180]$ degrees) limits the maximum angular difference permitted. In the example shown in Figure 19, the creation of the mapping $v_4 \rightarrow b_4$ is conditioned to: ANGLE(b_4, b_3, b_5) be similar to ANGLE(v_4, v_2, v_1); and ANGLE(b_4, b_5, b_3) be similar to ANGLE(v_4, v_1, v_2).

3.4 Summary

This chapter introduced the proposed detection technique and presented some general ideas that directed its development. Next, the two stages of the technique were detailed; they are executed sequentially for each input image.

The first stage is in charge of extracting candidates for features of the pattern. At this stage, the main goal is to retrieve the maximum number of inliers possible, even if it means producing some outliers. Ideas borrowed from two works of the literature were used for developing a tailored blob extractor that suits better the nature of the problem tackled in this Thesis.

The second stage establishes the elements that actually constitute the pattern. It proceeds by traversing a graph data structure that represents the pattern of interest while a number of constraints are tested to ensure the correct pattern is detected. These constraints were made intentionally localized and "loose" so that allowing global deformations of pattern to occur.

4 METHODOLOGY OF THE EXPERIMENTS

This chapter is concerned with the initial part of the experiments — the methodology used for assessing the proposed technique. The presented methodology addresses the blob extraction stage in isolation as well as the overall pattern detection algorithm. The covered topics include: an image preprocessing step, which is applied to every input image beforehand; the hardware used for experimenting; the ground-truth creation method; and, the metrics used for measuring the techniques performance.

Experiments were conducted in two distinct case studies. The first case study (Chapter 5) deals with the usual scenarios in which printed patterns have to be detected. Therefore, it is related to common computer vision tasks, such as camera calibration and pose estimation.

The second case study (Chapter 6) covers a more atypical scenario, which consists in a deep underwater environment with limited lighting conditions (the setting where offshore oil exploration takes place). In this case, the elements of interest are flexible pipes used for transporting fluids (oil, gas, and water).

The results obtained from both case studies were generated by using the same computing device — a desktop computer fitted out with a Core i7-3960X CPU and a GeForce GTX 560 Ti GPU, 24 GB of RAM, and running a Windows 7 64-bit operating system. Since different camera systems were used in each case study, their specificities were left to be detailed in the respective case study chapters.

4.1 Image preprocessing

A typical approach for improving the overall quality of a vision based system is preprocessing input images before they are submitted to the system core functionality. A preprocessing method is satisfactory if it improves images overall quality (for instance by removing undesirable artifacts) without demanding many computational resources. This preprocessing is illustrated as the second stage of the execution diagram depicted in Figure 9.

More formally, the purpose of this preprocessing is to increase the signal-to-noise ratio in the input image. One can achieve this by either amplifying the signal itself (the portion of the image related to the features of the pattern) or by reducing the noise (the undesired portion of the image). Despite existing many enhancement techniques that

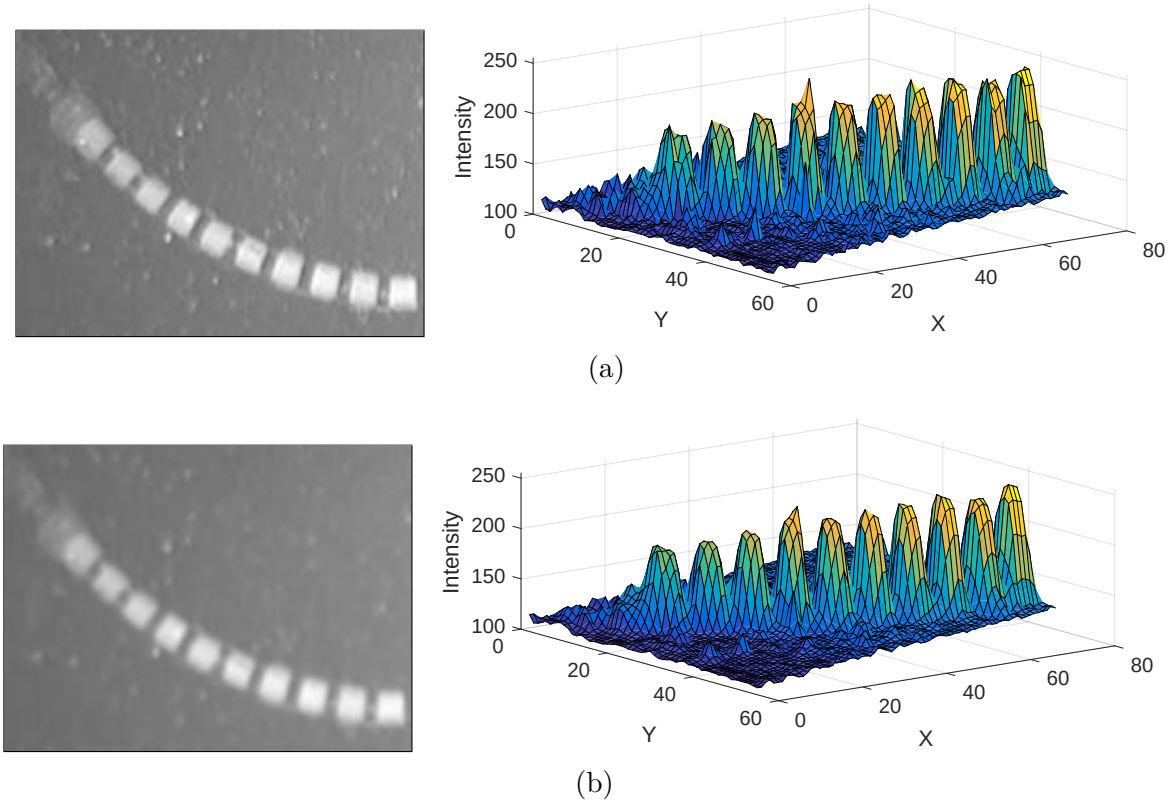
serve this purpose (GONZALEZ; WOODS, 2006c), the scenario-to-scenario specificities produce distinctive artifacts over the captured images. For example, the model of the camera system used, the lighting conditions, and the scene cluttering level surely have an impact on the input image visual aspect. Consequently, this problem does not have a single well-defined solution, but it requires a tailored one be developed for each scenario (especially when the best results are desired).

In spite of the particularities inherent to each scenario, some primal characteristics of most camera systems may be taken into account for developing a basic preprocessing procedure that is beneficial for a variety of situations. Images captured by any camera system are subject to noise at some degree. Most of these artifacts arise during the image acquisition and/or its transmission (YAN, 2013; VERMA; ALI, 2013). While acquiring images, the performance of an imaging sensor is affected by factors such as the area size of the sensor and its temperature. Small sensors are remarkably prone to produce noisy images when the scene is poorly illuminated. This happens because the sensor tries to counterbalance the low-level signal by increasing the gain, which ends up also accenting noise. Longer exposure times mitigate the effects of limited lighting conditions, although this option is only applicable to still images acquisition. Due to interferences in the channel, analogical transmissions are also subject to add noise to the image. The incessant presence of these noise sources suggests that a noise reduction filter would be a beneficial preprocessing method independently of the circumstance.

The noise that arises during the acquisition and/or transmission of an image can be approximated by a Gaussian model (CATTIN, 2015). The Gaussian model is also broadly used for noise removal because of its mathematical tractability in both spatial and frequency domains. In many practical situations, a Gaussian filter is preferred over others because of its well-behaved frequency response (FISHER et al., 2003). As any smoothing filter, it removes high-frequency components of the image. However, its effect is more gradual over the range of frequencies. From the lowest to the highest frequency, Gaussian filters progressively attenuate bands[1], whereas others have an oscillating behavior. In addition to the noise removal effect, Gaussian filters can also smooth out small details of the image.

Given that the image acquisition mechanism of cameras inevitably suffers from some degree of noise, the image preprocessing approach adopted by this work is applying a Gaussian filter. This filter suppresses high-frequency information from image making it more easily tractable. For not risking losing valuable information, a conservative and small kernel size was adopted. Empirical experiments evidence that a 3×3 Gaussian kernel is enough for achieving the desired results as well as it is also a size recurrently used in the literature. The effect of this preprocessing can be observed in Figure 21.

[1] The shape of a frequency response function of a Gaussian filter is itself a Gaussian.



Source: The author

Figure 21 – In (a), a raw input image (on the left) and its surface plot (on the right). In (b), the result of filtering the upper image with a 3×3 Gaussian kernel. Notice that the overall shape of the features (the white markings) are preserved while tiny and irrelevant details (possibly caused by noise) are suppressed.

Despite the techniques evaluated in this chapter do not require images to be preprocessed, employing such a method is justified because it portrays a more trustworthy usage scenario for the techniques (since many real applications employ it). Moreover, once the same preprocessing method is used throughout all experiments, if this method is affecting the techniques performance, then at least all techniques are equally affected, be it positive or negatively. As an outcome, the techniques evaluation remains on a par with each other.

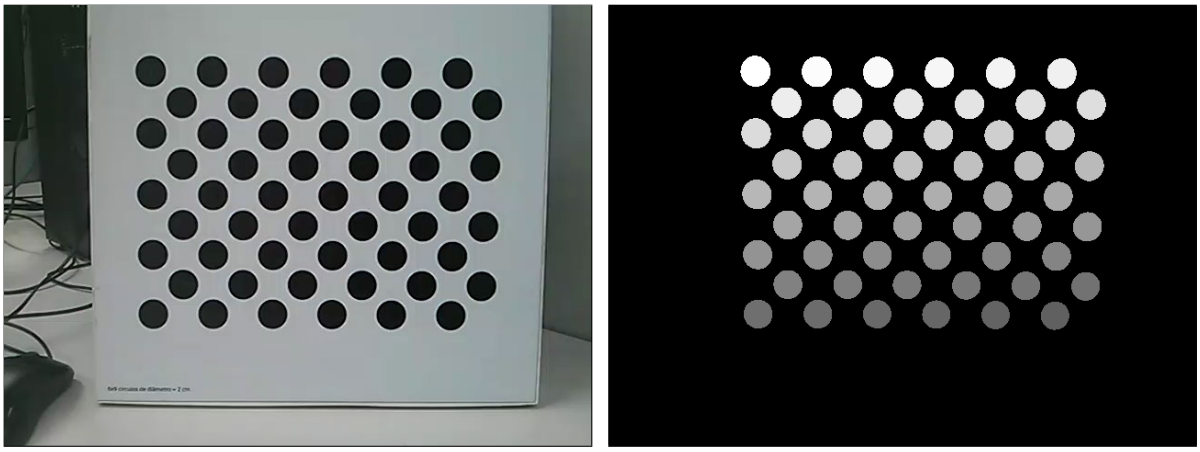
4.2 Hand-labeled ground-truth

Experimenting with images captured from real-life scenarios implicates some difficulties. One of them is that there is not an easy way to produce reliable ground-truths for such images. In that case, the ground-truths used in the experiments had to be manually crafted. This process was performed by different individuals; at its end, one hand-labeled image was produced for each input image.

In the context of this Thesis, two pieces of information are enough for evaluating a pattern detector (and consequently a blob extractor):

1. A set of connected components specifying the regions covered by the features of the pattern; and
2. A one-to-one set of correspondences that relates those connected components to the pattern features.

Both of these are stored in the hand-labeled images. For example, Figure 22 depicts an input image and its hand-labeled ground-truth. The ground-truth is a gray-scale image that portrays the regions not covered by pattern features in black. Each non-black connected component in the ground-truth image is filled with a constant color that uniquely identifies the pattern features.



(a) (b)

Source: The author

Figure 22 – An input image (a) and its hand-labeled ground-truth (b).

4.3 Evaluating blob extractors

This section introduces the blob extractors that are evaluated in the next two chapters and the metrics used for that. But before that, it is important to mention the filtering mechanism employed by the evaluator in order to make the assessment equitable.

The mentioned filtering mechanism is simple and direct — it discards blobs that are out of the size range of 21 to 3019 pixels in area (all tested images are 640×480 sized). These values were empirically defined so that no inlier blobs are risked to be rejected; the minimum value is small enough for accepting the smallest inliers possible and the maximum value is big enough for accepting the largest ones. Under those circumstances,

the assessment of the techniques that do not have a built-in filtering mechanism is put on a par with the assessment of the other techniques that have.

4.3.1 Algorithms

This subsection presents the blob extractors considered for evaluation. Additionally, the configuration parameters used by each technique are detailed.

The basic concept that guided the techniques parameterization was to use the same set of parameters for the maximum number of scenarios possible. The reason for the adoption of this approach is two-fold. First, it mitigates the overwhelming burden of selecting specific parameters for each scenario. At the same time, by proceeding so an important characteristic of the techniques is put on test — how well they can handle different conditions without user intervention.

In essence, both these reasons are related to the usability of each technique. After all, a technique that requires being continuously fine-tuned is not as convenient as a "self-adaptable" one (especially in scenarios subject to dynamic changes). In summary, for each of the case studies presented next, all the experiments were conducted by using the same set of parameters (i.e., the techniques parameterization was the same for all images of a given case study). The parameters used in each case study were defined empirically so that the techniques overall performance was maximized.

The first extractor evaluated was a Difference of Gaussian based one (this extractor is referred to as *dog* henceforth). The implementation used was the one provided by the package scikit-image (WALT et al., 2014) version 0.12.3 available for the Python programming language. The parameters[2] used with *dog* are in Table 1.

Table 1 – Parameters used with the blob extractor *dog*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Parameters	min_sigma	1.83	1.83
	max_sigma	21.92	21.92
	sigma_ratio	1.3	1.02
	threshold	0.01	0.001
	overlap	1.0	1.0

An extractor based on the Laplacian of Gaussian operator was also evaluated (referred to as *log*). The implementation provided by the same previous package was used with the parameters shown in Table 2.

The package scikit-image also provides an implementation of the Determinant of Hessian method, which was initially considered for evaluation. However, the results

[2] Refer to the libraries reference pages for a complete description of the parameters of each technique.

Table 2 – Parameters used with the blob extractor *log*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Parameters	min_sigma	1.83	1.83
	max_sigma	21.92	21.92
	num_sigma	10	10
	threshold	0.007	0.015
	overlap	1.0	1.0

produced by this implementation/method were not satisfactory and for this reason it was discarded. In short, different parameterizations were tried but this extractor always produced a disproportionate amount of outlier blobs (some hundreds), making the evaluation process impracticable to be completed in a timely interval.

The open-source computer vision library VLFeat (VEDALDI; FULKERSON, 2008) supplied the Maximally Stable Extremal Regions implementation (referred to as *mser*). The version used was the 0.9.20 coupled with the API (Application Programming Interface) for MATLAB. The parameters used are in Table 3.

Table 3 – Parameters used with the blob extractor *mser*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Parameters	Delta	1	1
	MaxArea	9.8277×10^{-3}	9.8277×10^{-3}
	MinArea	6.9131×10^{-5}	6.9131×10^{-5}
	MaxVariation	0.5	0.5
	MinDiversity	0.7	0.7
	BrightOnDark	1	1
	DarkOnBright	0	0

The blob extractor provided by the OpenCV (BRADSKI, 2000), which is referred to as *ocv*, was parameterized with the values shown in Table 4. The version 2.4.10.1 through its native API (C/C++) was used.

The P-Algorithm segmentation technique was evaluated through the implementation provided by the version 2.0.1 of the Mamba Image library (BEUCHER; BEUCHER, 2016). This library is open-source and its functionalities are accessible through Python. Since the technique is parameterless, only the following auxiliary parameters had to be specified: gain = 2.0 and grid = SQUARE.

Additionally to the techniques found in the literature, two versions of the proposed blob extractor were tested. The first one consists in the exact algorithm as presented in Chapter 3; this is referred to as *proposed*. The other is a version adapted to work as the Lindeberg extractor; it is referred to as *mll* (an acronym to Multi-level Lindeberg). The

Table 4 – Parameters used with the blob extractor *ocv*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Parameters	minThreshold	0	0
	maxThreshold	255	255
	thresholdStep	1	1
	filterByArea	true	true
	minArea	21.23	21.23
	maxArea	3019.07	3019.07
	filterByCircularity	false	false
	filterByColor	false	false
	filterByConvexity	false	false
	filterByInertia	false	false

single change made to derive *mll* from *proposed* is in the slice selection criterion — while *mll* selects the slice from the base of the peaks to create blobs, *proposed* prefers the most stable one. The parameters used by both versions of the proposed extractor are in Table 5.

Table 5 – Parameters used with the blob extractors *mll* and *proposed*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Parameters	kMinArea	21.23	21.23
	Δ	1	1
	kMaxM	4	1
	kMinQ	0.695	0.695

4.3.2 Metrics

The performance of blob extractors are evaluated by using different metrics. The two first metrics regard the number of blobs extracted (inliers and outliers) as well as the number of pattern features neglected during the extraction process. Essentially, they measure how well a blob extractor performs in numbers of extracted elements.

The first metric is the Precision. It evaluates the fraction of extracted blobs that are inliers. Letting the inlier blobs be referred to as true positives (tp) and the outlier ones be referred to as false positives (fp), one has that

$$\text{Precision} = \frac{tp}{tp + fp}. \tag{4.1}$$

The second metric is the Recall, which evaluates the fraction of relevant elements that are extracted. Therefore, this metric takes into account the number of false negative (fn) results produced by the extractor (i.e., the number of pattern features missed). This

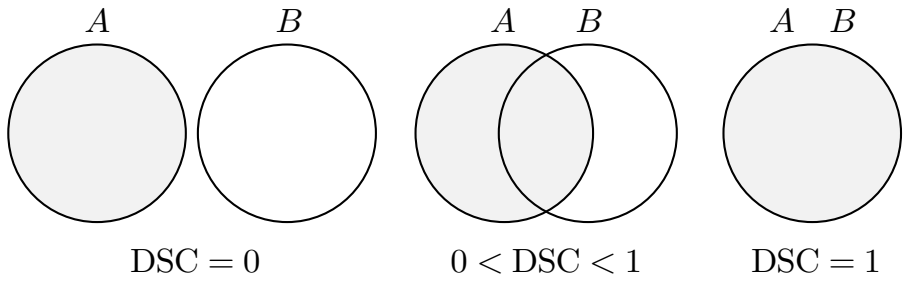
metric is denoted by the expression

$$\textsc{Recall} = \frac{tp}{tp + fn}. \tag{4.2}$$

Before evaluating these metrics it is necessary to classify the extracted blobs either as true positives or as false positives. Basically, a blob is considered a true positive if it attains to a minimum spatial overlapping score with a connected component of the ground-truth image. The overlapping score used is the Dice Similarity Coefficient (DSC) (ZOU et al., 2004), which is expressed as

$$\text{DSC} = \frac{2 \times \textsc{Area}(A \cap B)}{\textsc{Area}(A) + \textsc{Area}(B)}, \tag{4.3}$$

where A is a connected component of the ground-truth image and B is a blob (Figure 23).



Source: The author

Figure 23 – The range of scores produced by the DSC. From left to right: it produces 0 when the elements do not overlap at all; when the elements partially overlap the score is in the range $(0, 1)$; and finally, under a complete overlapping 1 is produced.

The overall procedure that evaluates the number of true positives, false positives, and false negatives is as follows. For each connected component in the ground-truth image, the DSC score is evaluated against all blobs extracted. If none of the blobs produced a score greater than 0.5 at the end of the evaluation for a given connected component, then a false negative result is accounted. Otherwise, a true positive is accounted and the blob that produced the highest score is removed from the set of available blobs. At the end of this procedure, the elements remaining in the set of available blobs determine the number of false positives.

The last metric used is the arithmetic mean of the DSC scores obtained by the connect components that attained to the minimum score requirement, which is denoted by AMDSC. This metric can be imagined as a measurement of the inlier blobs quality.

4.4 Evaluating pattern detectors

This section presents the pattern detectors considered for evaluation and the metrics used for such a thing.

4.4.1 Algorithms

Besides the proposed pattern detector, the one provided by the OpenCV was the only detector considered for evaluation (the same terms used for identifying the corresponding blob extractors are used here; *proposed* and *ocv*). Two reasons justify this option. Firstly, there are not many implementations of pattern detectors based on circular features out there to be used. Secondly, in spite of using the same kind of feature (circles), most of the detectors available are not compatible with each other. Simply put, it is common that the features arrangement adopted by a detector is unique, which in turn does not permit another detector to recognize that pattern. The proposed detector overcomes this limitation by utilizing a graph data structure for representing the pattern of interest, which enables it to be used with a variety of different patterns.

In addition to its availability, another reason that motivated selecting the *ocv* pattern detector was that it supports a kind of pattern that can be detected unambiguously — the asymmetrical circle pattern (more details in section Section 5.2). On the other hand, *ocv* is not able to detect the pattern marked over the pipe used in the second case study (Section 6.2), and no other detector with such an ability was found in the literature. For this reason, the sole pattern detector evaluated in the second case study was *proposed*.

The *ocv* pattern detector was parameterized by using an instance of the blob extractor constructed with the parameters show in Table 6. This instance is considerably less sensitive than the one used for assessing that extractor (the reason for not using the same parameterization is detailed in Section 5.5). For the pattern detector itself, in addition to the flag CALIB_CB_ASYMMETRIC_GRID (which determines the kind of pattern desired), it was also used the flag CALIB_CB_CLUSTERING to make the detection more robust to perspective distortions. Without this last flag, it was noticed that *ocv* occasionally failed in correctly detecting the pattern under oblique viewpoints.

The proposed pattern detector was configured with the data presented in Table 7. Its blob extraction stage was parameterized in the same manner as while assessing the *proposed* extractor (refer to Table 5). For its pattern searching stage, although the same parameters were used in both case studies, different pattern templates were utilized in each scenario (see them in Figures 26 and 70).

Table 6 – Parameters used with the pattern detector *ocv*.

		Case Study
		1 (printed pattern)
Extractor Parameters	minThreshold	0
	maxThreshold	255
	thresholdStep	10
	filterByArea	true
	minArea	21.23
	maxArea	3019.07
	filterByCircularity	false
	filterByColor	true
	filterByConvexity	true
	filterByInertia	true
Detector Flags		CALIB_CB_ASYMMETRIC_GRID | CALIB_CB_CLUSTERING

Table 7 – Parameters used with the pattern detector *proposed*.

		Case Study	
		1 (printed pattern)	2 (pipe in deep sea)
Extractor Parameters		Table 5	
Detector Params.	kMaxDist	100	100
	kMaxGapDistRatio	0.66	0.66
	kMaxRadiusRatioPercent	2.0	2.0
	kMaxDistRatioPercent	1.5	1.5
	kMaxAngleDiff	40	40

4.4.2 Metrics

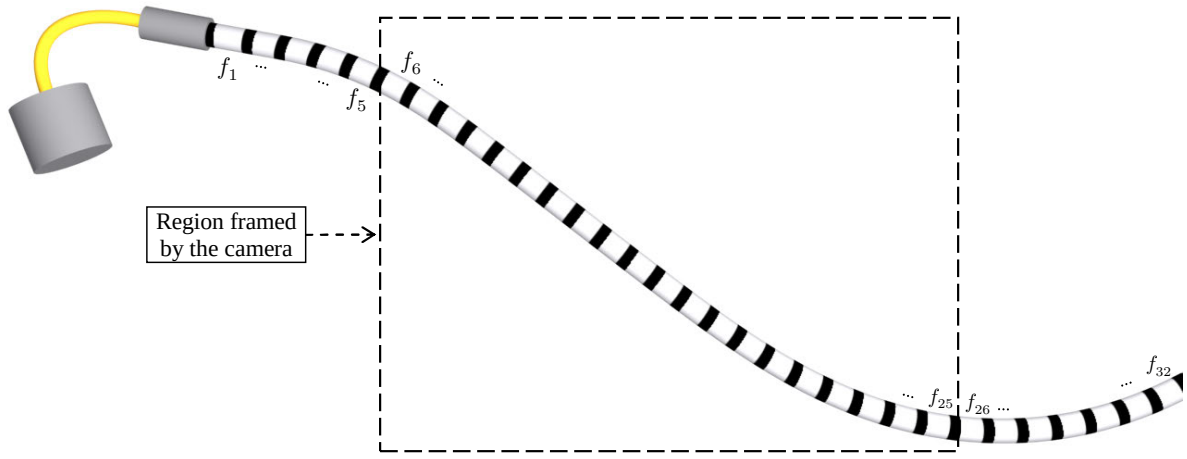
The detectors assessment is straightforward for the first case study. The fundamental idea behind it consists in accounting the number of complete detections achieved by them. That is, for a given number of frames processed, it is evaluated the total amount of frames in which detectors succeeded in retrieving the whole pattern (partial detections[3] are accounted as failures).

However, a different measurement is used for asserting true positive results while assessing pattern detectors — the distance between centroids of blobs and centroids of ground-truth regions. This measurement was preferred instead of the DSC score mainly because of a characteristic of the *ocv* blob extractor, which is the tendency to underestimate the real size of features (more details about it in Section 5.4). On top of that, the first case study is more related to problems such as camera calibration and pose estimation, which (for most cases) ultimately depend on positional information rather than on the overall shape of the extracted features.

[3] Since *ocv* can only detect the pattern as a whole, partial detections are pertinent only to the *proposed* pattern detector.

All things considered, for the first case study the evaluation procedure is as follows. For a given input frame, it is estimated the centroid of each ground-truth region and the centroid of the blobs resulting from the pattern detection (i.e., the blobs mapped as representative of the pattern features). Next, the evaluation procedure utilizes the color of each ground-truth region (which is unique) for identifying its corresponding blob; their centroid distance is evaluated afterward. A successful detection happens if all ground-truth regions are no more than three pixels distant from their respective corresponding blobs. Otherwise, a detection failure is reported for that frame.

The second case study is evaluated differently because of its odd characteristics. Firstly, the pattern marked over the pipe is not always expected to be detected in its entirety — while the whole pattern is constituted of some tens of features, just a fraction of them remain framed by the camera at some moments. Additionally, the pattern simplicity does not permit it to be unambiguously detected (e.g., for any portion of the pipe being framed at a given moment, the detector cannot distinguish if that portion is of the beginning, of the means, or of the end of the pipe). As a final outcome, the *proposed* detector usually does not produce an "exact" mapping between the vertices of the pattern template and the features marked over the pipe (Figure 24).



Source: The author

Figure 24 – Example in which the ambiguity of the pattern prevents the *proposed* detector from producing an "exact" mapping between the vertices of the pattern template and the features marked over the pipe. Observe that the detector does not have means to know that the blob extracted from the first feature framed by the camera is the sixth feature marked over the pipe (f_6) instead of the first one (f_1). As a result, the first vertex of the pattern template will be mapped to the blob extracted from f_6, the second vertex will be mapped to the blob extracted from f_7, and so on.

Given these points, the assessment of the second case study is based on the evaluation of the metrics Precision and Recall. However, opposed to the universe of

all extracted blobs (as was with the blob extractors assessment), the metrics PRECISION and RECALL are evaluated regarding the universe of the blobs mapped by the detector.

4.5 Summary

This chapter presented the methodology that will be used for assessing the proposed technique in the next two chapters (the case studies). This methodology involves a comparative evaluation of existing algorithms from the literature against the proposed one. For a better assessment, the evaluation is divided into two parts: first, the blob extractor is evaluated in isolation; second, the pattern detector is evaluated as a whole. The algorithms are fine-tuned for each case study by adjusting their parameters, although no special adjustments are permitted to particular images from a given case study.

An image preprocessing procedure, which was implemented with a 3×3 Gaussian kernel, was included in the methodology for removing high-frequency noise from input images. Ground-truth images hand-labeled by different individuals are used for verifying whether the results output by the algorithms are correct. Three metrics (PRECISION, RECALL, and DSC) are the basic indexes used for assessing blob extractors and pattern detectors. Whenever necessary, these metrics were adapted to address the specificities of each case study.

5 CASE STUDY 1 (PRINTED PATTERN)

This case study regards an usual computer vision problem — the detection of manufactured patterns formed by circular features. These patterns can be easily crafted with commodity printers and some important applications of them include camera calibration and pose estimation. In order to find out the range of scenarios in which the evaluated techniques can perform well, different image sets featuring distinct conditions (some of them being purposely made severe) are used.

5.1 Scenario description

Most of the typical scenarios wherein a manufactured pattern has to be detected involve arbitrarily positioning the camera so that images of the pattern are captured from different viewpoints. For example, both of the aforementioned applications requires so — while the optimization process of a camera calibration algorithm demands distinct views of the pattern for properly finding the camera intrinsic parameters, it is natural that in a pose estimation application the object of interest appears from different viewpoints.

However, this freedom of positioning can pose additional challenges for blob extractors as well as for pattern detection algorithms — the distortions caused by the cameras image formation process. In a trivial case where the pattern is perfectly facing the camera, perspective distortions change the pattern apparent size (it increases or decreases when the pattern comes closer or farther from the camera, respectively). Distortions that are more aggressive may occur when pattern is observed from oblique views. Since these viewpoints produce non-uniform deformities over the pattern surface, they severally affect the pattern shape. Additionally, perspective distortions affect not only the overall pattern shape, but also the features themselves (circles appear as ellipses when observed from oblique views). As shown above, a pattern observed from different viewpoints may not seem the same, and under such a circumstance, the algorithms are yet expected to properly detect patterns.

Another factor that may impair algorithms performance is the scene lighting condition. The overall lighting condition of a scene is influenced by both the light sources (positioning and intensity) and the existence of occluding objects. It is true that the user has a reasonable control over these elements in many habitual scenarios, but there are others wherein such a control is limited. For instance, in industrial settings (KANG; HA; JEONG, 2008) and in offshore environments (see Chapter 6) it is not always possible to freely rearrange scene elements. As can be seen, every real-life scenario has its own restrictions regarding the elements placement, therefore, it is expected that algorithms are robust enough to handle some degree of lighting limitation.

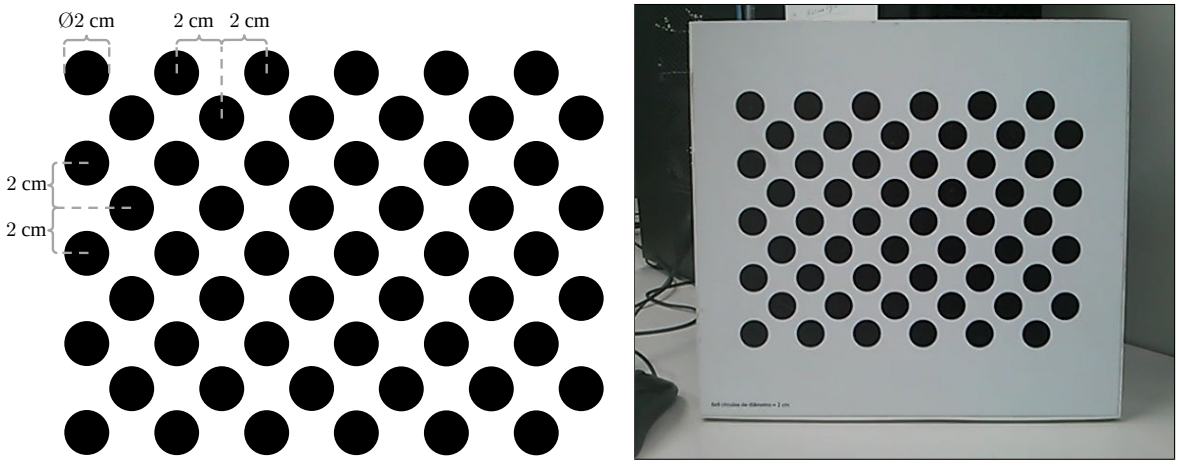
Finally, there are camera systems equipped with fisheye lens, which can capture wide angle views of a scene. In addition to the distortions caused by the perspective projection, these systems also feature radial and tangential distortions. These distortions become more pronounced with an increase in the lens aperture angle and towards the image edges.

All the previously mentioned issues were taken into account for creating the image sets presented in Section 5.3. By doing so, a wide range of real-life scenarios is covered in this case study.

5.2 Pattern of features

The pattern used in this case study has 54 circular features arranged in a specific fashion. The pattern overall shape is rectangular and the features spacing is regular (Figure 25a presents the details). The reason for this arrangement is to do not permit the pattern to be detected in more than one way. More precisely, the intent is to make the pattern unambiguous under rotation transforms.

The tangible pattern was crafted by using a printer and a flat wood board (Figure 25b). The wood board is intended for convenience of handling while maintaining the pattern planarity.



(a) (b)

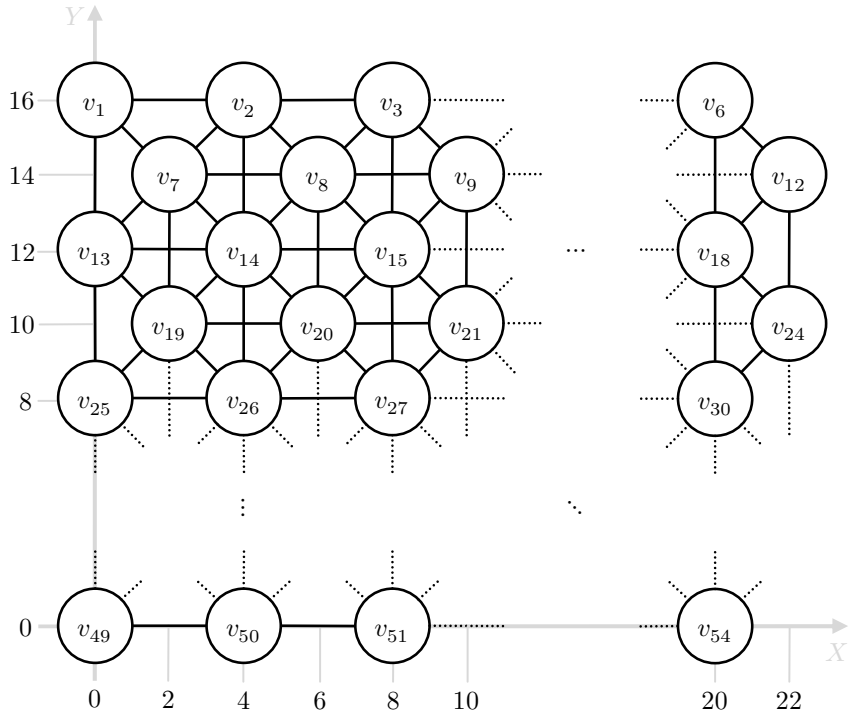
Source: The author

Figure 25 – In (a), details of the pattern used throughout this case study. For achieving the best image contrast, features are in black while the pattern background is in white. This pattern is the same as a 6×9 sized asymmetric circle pattern provided by the OpenCV library. In (b), the final palpable pattern.

As previously stated in Section 3.3, the definition of a pattern template is invariant under 2D transforms such as translations, rotations, and uniform scaling. Therefore, there

are infinite ways to arrange the template vertices that are equivalent to each other. For a sake of convenience, it was preferred to center the vertex at the bottom left corner of the template at the coordinate $(0, 0)$ and to align the longest edges of the template with the x-axis (Figure 26). Vertices spacing was set to the same values presented in Figure 25a.

The vertices immediate neighborhood relations were established adopting an eight-neighborhood-like connectivity (although border vertices have lower cardinality). Since the detector algorithm envisages an edge as an additional restriction that must be met by a candidate blob, having more edges leads to a greater likelihood of blobs not being satisfactory. Therefore, this creates a more restrictive template than a four-neighborhood-based one would do. As a natural outcome, more reliable detection results can be expected by using such a template.



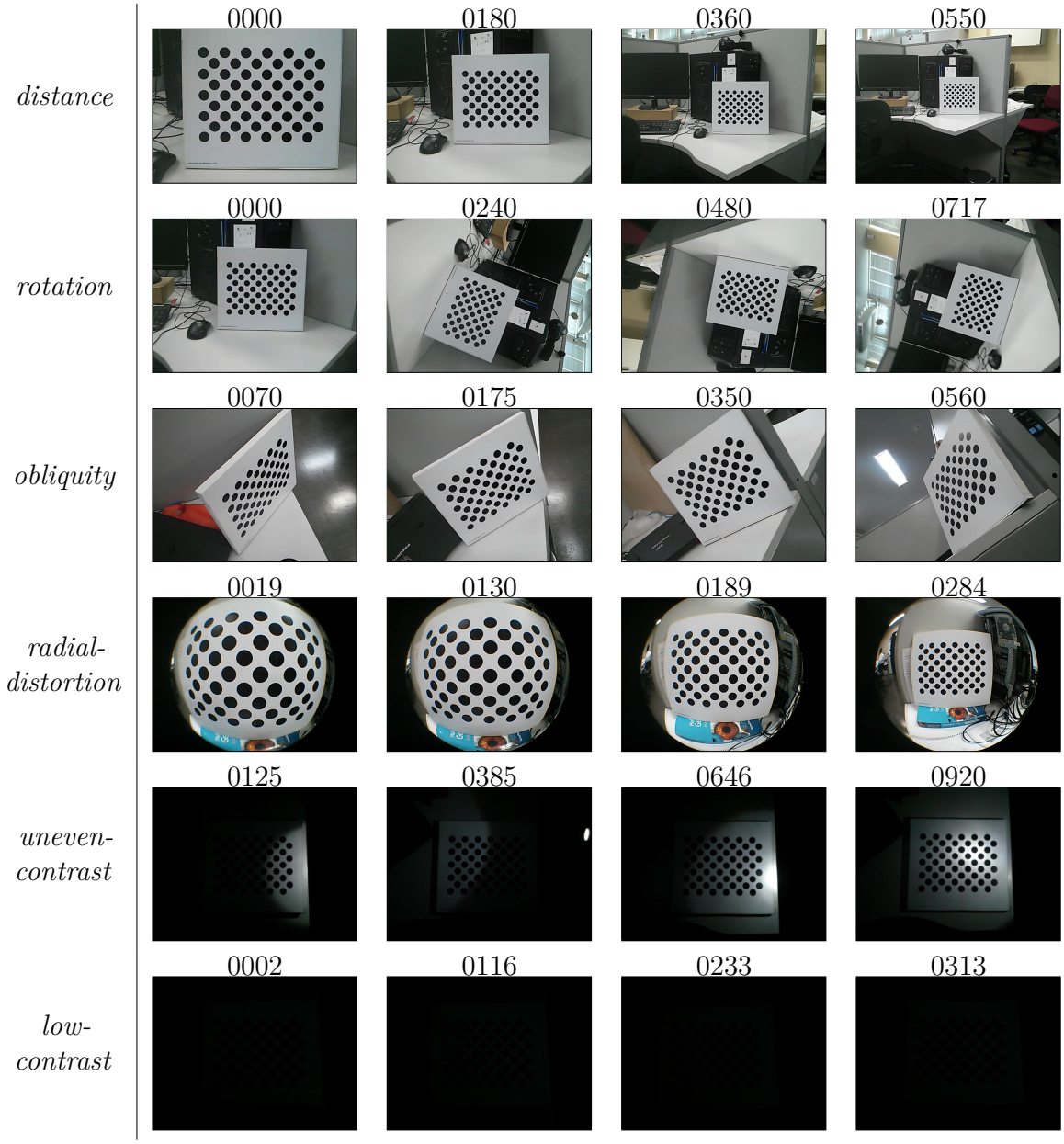
Source: The author

Figure 26 – Template of the pattern used in this case study. Vertices are connected to each other so that they have at maximum eight neighbors.

5.3 Image sets

Six image sets are used in this case study (refer to Figure 27 for some sample images). They depict different conditions that applications may face in real-life scenarios. Each set contains 20 images, resulting in a total of 120 images.

Images were captured using the rear-facing camera of an LG Nexus 4 E960 smartphone. The acquisition process of each image set was as follows. 1) Recording a video of



Source: The author

Figure 27 – Each row contains images sampled from one image set used in this case study. The images are labeled with four-digit names drawn from their original frame numbers in the video footage.

about 10 to 30 sec (approximately 300 to 900 frames) with the camera configured to SD 480p (the resolution of 720×480 pixels). 2) Evenly sampling 20 images from the video footage. 3) And finally, cropping the sampled images to the final size of 640×480 pixels. This size was preferred just for a consistency sake (the camera system used in the next case study natively captures 640×480 sized images).

The first image set is termed *distance*. It explores viewpoints from a close to mid-range distance with the pattern facing the camera. Since the pattern appears to reduce gradually in size over the images, this image set can also be imagined as the product of a

uniform downscaling transform of the pattern. The outcome is that the pattern dimensions in the last image is around one-fourth of the same measurements in the first image. The features diameter also reduces in a similar manner, which in turn makes this image set a good scenario for evaluating whether blob extractors are scale invariant.

The next image set is named *rotation*. As its name suggests, it regards a rotation transform of the camera (more precisely, a complete rotation around the camera optical axis). Similarly to the first image set, the pattern is also kept facing the camera over all images. However, since in this case the camera to pattern distance is constant, pattern dimensions (as well as features diameter) are preserved over images. This creates a condition where, from the blob detectors perspective, all images of this set are equally good (or bad) for processing. Therefore, this image set is more intended for evaluating the pattern detection in overall (a rotated pattern may be ambiguous for some pattern detectors).

The *obliquity* image set explores viewpoints from where perspective distortions are more pronounced. This set starts by a severe oblique viewpoint and reaches a neutral one at the middle of the set. In the second half of the set, the camera follows towards the antipodal viewpoint increasing obliquity again. The final camera trajectory is similar to an (partial) orbit. In this image set distortions not only affect the pattern overall shape, but also the features (they are distorted from circles into ellipses). Therefore, it poses a challenge for pattern detectors as well as for blob extractors.

The fourth image set delve into radial-distortion effects (not surprisingly this set was named *radial distortion*). Differently from perspective distortions, radial distortions do not necessarily preserve straight lines (only vertical and horizontal lines passing through the image center are preserved). This deformation transforms the pattern overall shape from a perfect quadrilateral into a form with rounded sides (although yet resembling a quadrilateral). Radial distortions are more apparent in images captured by cameras equipped with a wide-angle lens and the effect increases from the image center to its border. Therefore, for achieving the desired degree of distortion a 37 mm 0.25× Super Fisheye Lens with +12.5 Macro was attached to the camera prior to starting the recording.

The goal of the last two image sets is to explore changes in lighting conditions. With the intent of isolating the impact of such changes, no viewpoint changes were made during both recording sessions (except by the natural hand movements while holding a camera). In the set *uneven-contrast* a non-uniform lighting condition was forced. It was achieved by turning off the light sources present in the scenario and using a flashlight for illuminating just some regions of the pattern. As a result, a single image of this set features high contrast regions (the regions aimed by the flashlight central light beams) as well as low to moderate contrast regions (the regions not directly illuminated and the regions aimed by the peripheral light beams).

The final image set is named *low-contrast*. The scenario wherein the images of this

set were captured was made as dark as possible. There was not any artificial light source turned on and it was night by the time recording was made. These circumstances created a challenging condition for blob detectors in which pattern features are barely visible in the captured images (adjust the brightness/contrast of your monitor for a better view of these images).

The complete sets of images (including the hand-labeled ground-truth images) are in Appendix B. Note that these images were converted to a gray-scale format and filtered with a 3×3 Gaussian kernel (as described in Section 4.1) before experimentation begins.

5.4 Blob extraction results

This section evaluates the blob extractors introduced in Subsection 4.3.1 by using the six image sets previously detailed. It starts by evaluating the metrics PRECISION, RECALL, and AMDSC on a per-image basis for each image set. At its end, the extractors overall performance are summarized by regarding the mean of intra-set results. Supplementary tables are provided in Appendix A.

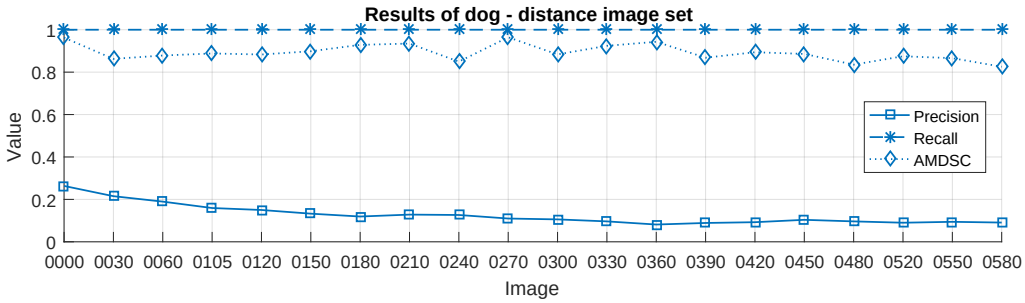
5.4.1 Image set *distance*

The *distance* image set is characterized by the reduction in features apparent size with the images progress. Refer to Table 9 for the per-image results obtained by extractors in numerical values.

In overall, the *dog* extractor performed well over all images of the *distance* set (Figure 28). It maxed out RECALL at 1 (indicating that all pattern features were properly extracted) while maintaining AMDSC high (values above 0.75 indicate the extracted blobs are good representatives). The drawback of this extractor is that it is prone to respond positively to regions of the image devoid of features, which made PRECISION low. Another important conclusion that can be drawn from the results is that this extractor did not suffer from the decreasing in features size (see the consistent RECALL and AMDSC results over the images). The reduction in PRECISION indicates only the camera is moving to a more cluttered viewpoint.

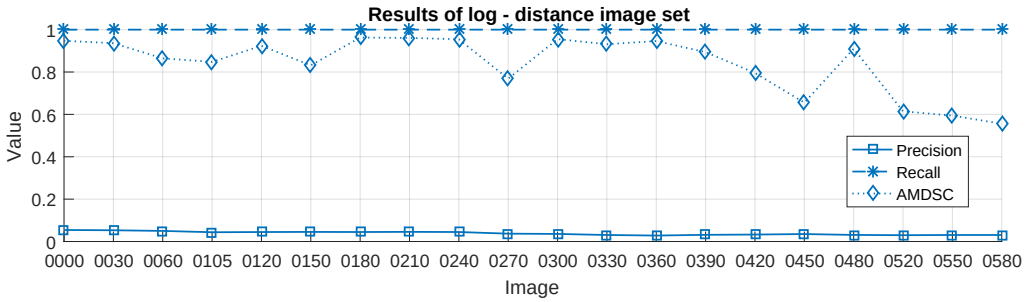
The *log* extractor performed similarly to *dog* (Figure 29). There are only two observations worthy of remark here. First, *log* is more prone to produce false positives results than *dog* (lowering its PRECISION). And second, the noticeable drop in AMDSC for the last images of the set (indicating that from this point the quality of blobs degrades with the reduction in features size).

The *mser* extractor also retrieved all pattern features over all images (Figure 30). However, this time the PRECISION results were superior to the ones obtained with the two



Source: The author

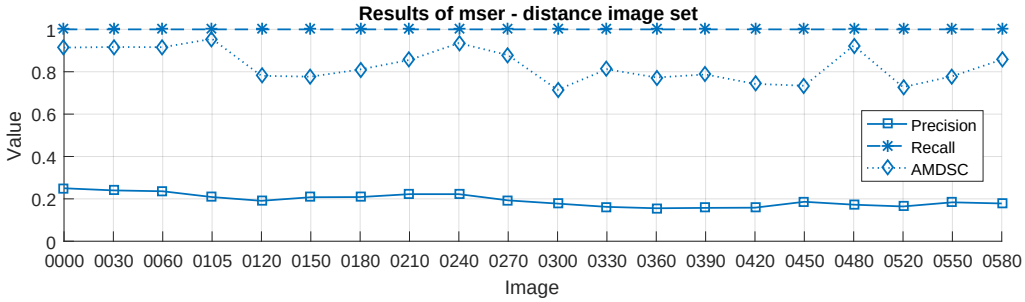
Figure 28 – Results of blob extractor *dog* for image set *distance*. The result of each metric is presented on a per-image basis. Images are identified by four-digit names drawn from their original frame numbers in the video footage.



Source: The author

Figure 29 – Results of blob extractor *log* for image set *distance*.

last extractors (the inferior standard deviation indicates a greater consistency as well). Yet regarding the two last extractors, the quality of the inlier blobs extracted by *mser* was inferior though.



Source: The author

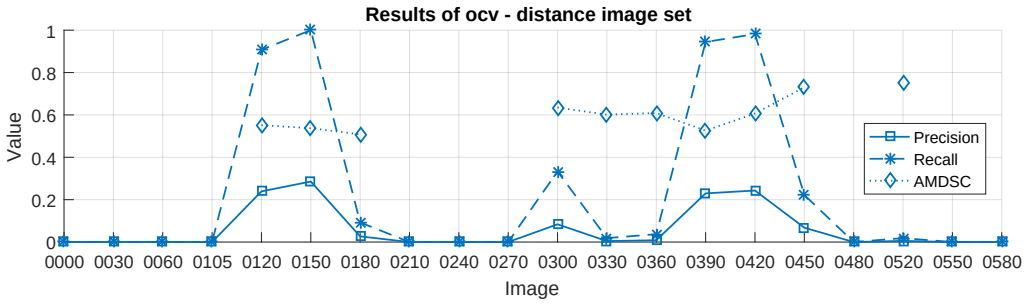
Figure 30 – Results of blob extractor *mser* for image set *distance*.

The *ocv* extractor performed poorly for most images of this set (Figure 31). Essentially, the reason for such result is that this extractor underestimates the real size of pattern features. In this case, the minimum score required for a blob to be considered a true positive is not attained (Figure 32). As might be expected, all the tree metrics are impacted when no true positive results are produced. While PRECISION and RECALL drop

to zero, AMDSC is absent. Another interesting remark about this extractor is that its results alternate creating a pseudo-periodic wave-shaped chart. Two reasons are attributed to this fact:

1. There is a variance in the size of the regions hand-labeled by individuals (i.e., while creating ground-truth images, some individuals drew slightly bigger regions, others drew slightly smaller regions[1]); and

2. The blobs produced by *ocv* have, in average, a deficit in size which produces DSC scores compatible with the adopted cutoff value of 0.5 (notice that *ocv* AMDSC results, when existent, are mostly about 0.5).



Source: The author

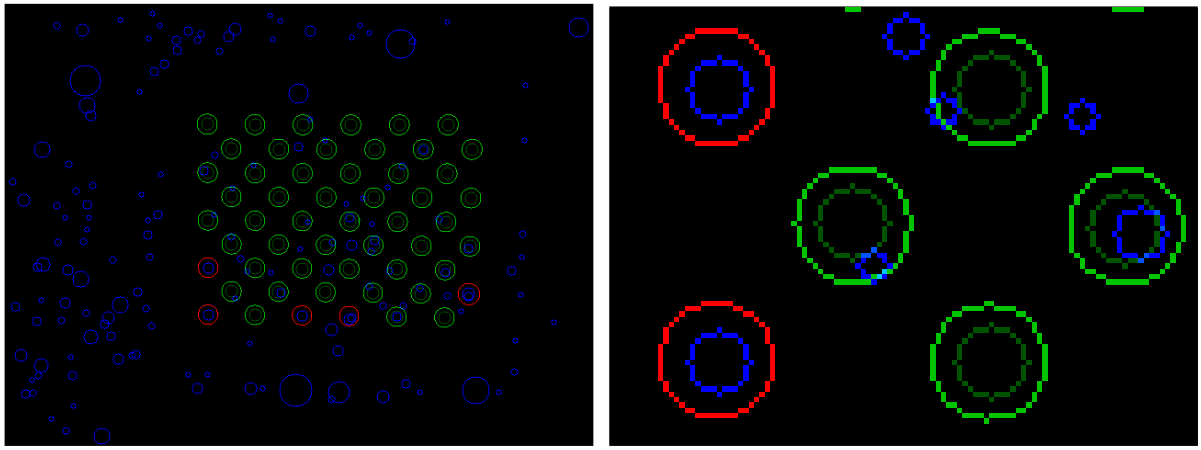
Figure 31 – Results of blob extractor *ocv* for image set *distance*.

These two factors combined create a circumstance in which even a small variance in the size of the ground-truth regions can change blobs classification from true positive to false positive (and vice versa). In special for the *ocv* extractor, it is evident that smaller hand-labeled regions bring benefits (it improves the likelihood of the minimum DSC score be attained producing more true positive results), and it was what caused this extractor to produce sporadic positive results.

The *p-algo* extractor achieved the best PRECISION results so far, although it produced an inconsistent RECALL reaction with only two images getting maximum response (Figure 33). By visually inspecting the images output by this extractor it is easy to identify the cause of the missing true positive results that lowered RECALL. For some reason, a number of extracted blobs are connected with their outer regions making these blobs part of bigger connected components (i.e., disfigured blobs that cannot be recognized as true positives anymore). Even a small failure in the encompassing contour of a blob (e.g., an one pixel wide "crack") is enough to ruin its DSC score. Figure 34 exemplifies this problem. Despite all these issues, the inlier blobs produced by *p-algo* have good quality.

The *mll* extractor retrieved all features over all images at the same time that false positive results were avoided (Figure 35). However, contradicting what was expected,

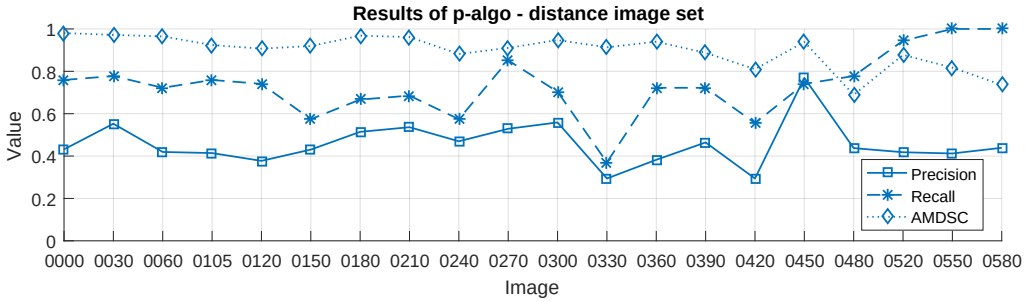
[1] This is a natural outcome since distinct individuals are expected to have different levels of skills.



(a) (b)

Source: The author

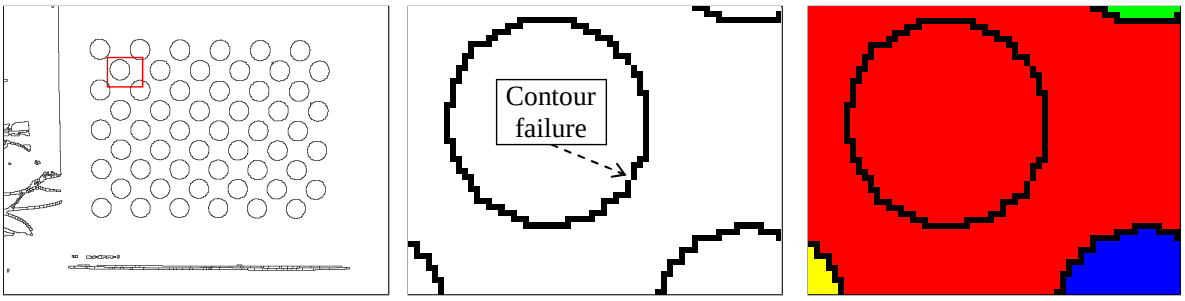
Figure 32 – The image in (a) shows how the blobs extracted by *ocv* from input image 0120 were classified by the analyzer. Contours in blue represent false positive results, contours in red represent false negatives, and contours in green represent true positives (dark ones are the extracted blobs and bright ones are the matched ground-truth regions). In (b), the bottom left portion of the pattern is shown in detail. Note that inside each false negative contour (in red) there is a false positive one (in blue) that did not attain to the minimum DSC score.



Source: The author

Figure 33 – Results of blob extractor *p-algo* for image set *distance*.

this extractor did not perform significantly worse than others considering the AMDSC metric (remember that this extractor was expected to produce oversized blobs). This fact is curious and has a tricky explanation. This time the explanation comes from a close inspection of the input images, which reveals that the intensity of pixels at features edges changes in an unexpected way (considering how a black to white transition, and vice versa, should look like). By imagining the plot of a row of pixels that cross a feature through its center, it is as if there were artifacts similar to road bumps exactly at the crossing points (Figure 36). The author's best guess is that these artifacts are unnaturally produced by the camera built-in software, which supposedly applies a sharpening operator in an attempt to enhance images overall quality. It has been noted that these "bumps" create

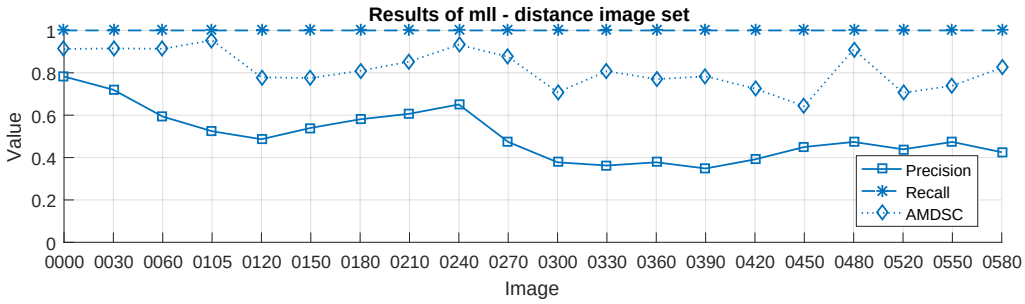


(a) (b) (c)

Source: The author

Figure 34 – In (a), image output by *p-algo* from input image 0000. The red rectangle in (a) is shown in detail in (b). Note that there is a one-pixel wide failure connecting inner and outer regions (eight-neighborhood is considered here). In (c), colors identify the resultant connected components.

barriers limiting blobs growth to a fraction of the expected size. As a final observation, the Gaussian filter used to preprocess input images is not enough to wipe these artifacts away.



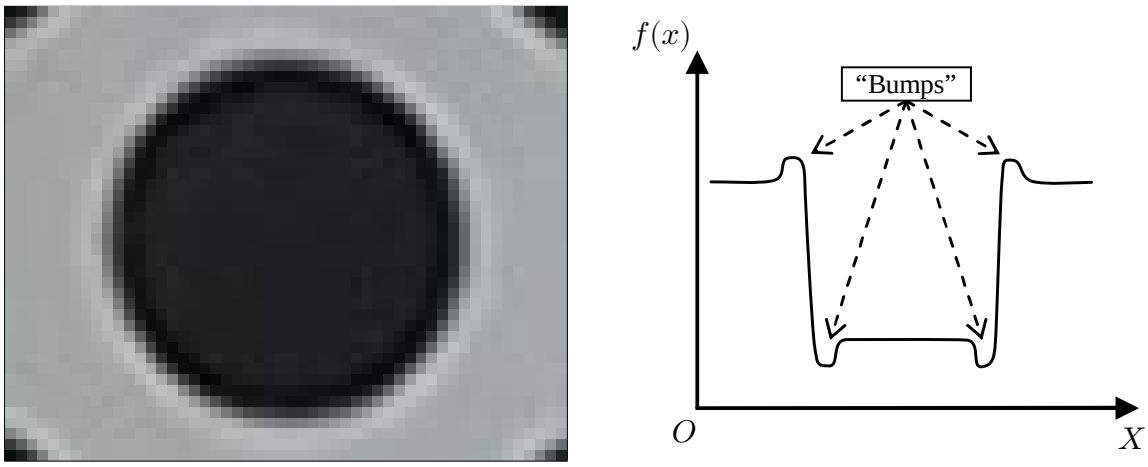
Source: The author

Figure 35 – Results of blob extractor *mll* for image set *distance*.

The *proposed* extractor performed virtually identical to *mll* (Figure 37). As a matter of fact, either selecting slices that are at the base of the peaks or selecting the maximally stable ones (the only difference between both extractors) did not turn out to produce major differences in AMDSC. Given these points, the two extractors are also expected to perform similarly for the most part of the next image sets, since the same camera system was used to produce them.

5.4.2 Image set *rotation*

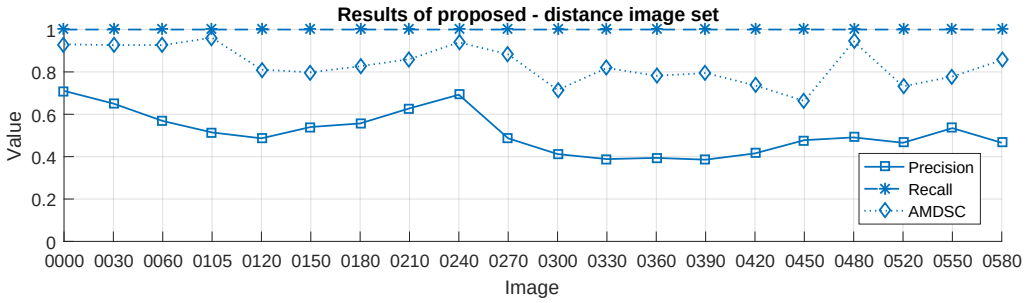
The images of this set were captured with the camera placed at a fixed distance from the pattern and by progressively rotating it around its optical axis (in fact, the camera was unwittingly moved away from pattern during the recording, but it was subtle). These circumstances made images alike each other from the blob extractors perspective (after all, circular features are invariant under such a rotation transform). Therefore, consistent



(a) (b)

Source: The author

Figure 36 – In (a), a pattern feature in detail. Note the weird changes in pixels intensity at feature edge. In (b), plot of a row of pixels depicting the "bumps" that limit blob growth.



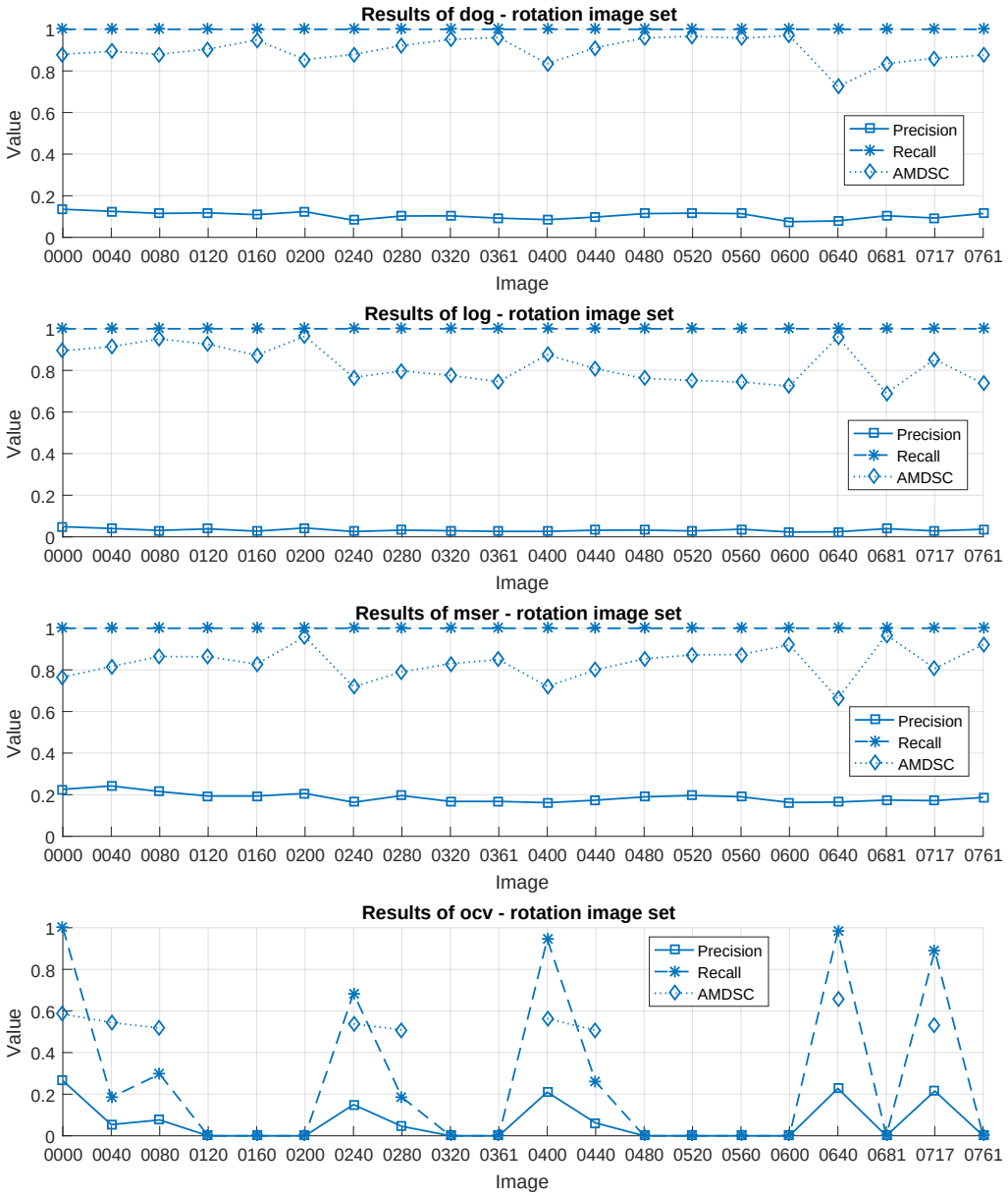
Source: The author

Figure 37 – Results of blob extractor *proposed* for image set *distance*.

results over the images of this set are expected. In addition, it is plausible to argue that images of the *rotation* set are alike the ones at the middle of the *distance* set, so it is also possible to anticipate that extractors will perform similarly with both sets. The per-image results obtained by each extractor with the *rotation* set are in Figure 38 and in Table 10.

As can be observed in those results, all extractors performed similarly for the image sets *rotation* and *distance*, as expected. For this reason, no further detailing is needed on the results obtained with the *rotation* set. The next paragraphs draw some noteworthy observations about the variance in size of the regions hand-labeled by different individuals.

By closely inspecting the ground-truth images relative to input images 0000, 0240, 0400, and 0640 it is possible to testify that the hand-labeled regions do not cover the pattern features in their entirety. It is important to make two points clear here though. First, the non-covered portions of the features are small and usually not visible through "naked eyes" (they are in general ring-shaped regions visible only under a magnifier).

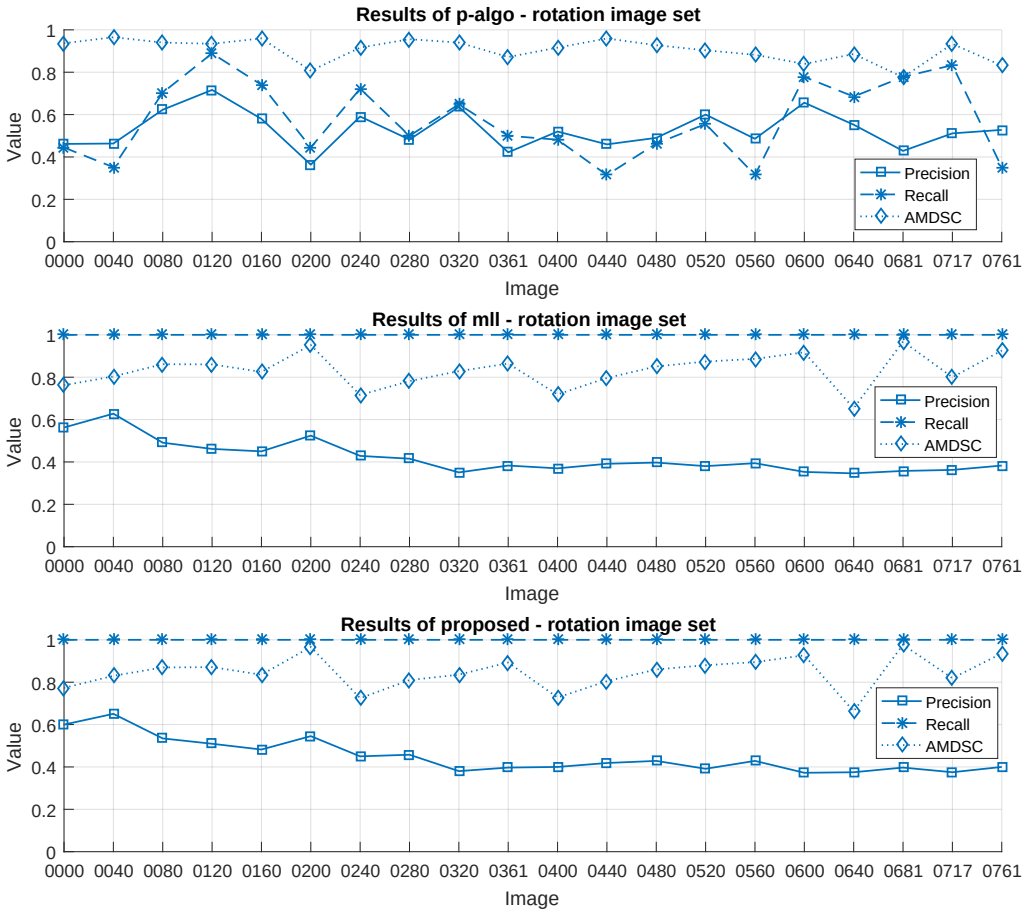


Source: The author

Figure 38 – Results of blob extractors for image set *rotation*.

Second, the boundaries of the features are not clear, permitting multiple interpretations (even under intense magnification they may be nebulous). Regarding these points and considering that individuals are allowed to make their own judgments about the features extent, this variance issue is something practically impossible to be completely prevented.

Naturally, not only the *ocv* extractor is affected by the variance in size of the hand-labeled regions, but all others as well. However, the impact over the other extractors are not as pronounced as it was with *ocv*, which made RECALL results cover the whole range of possible values. To see the impact caused by the variance of the hand-labeled regions on the extractors *mser*, *mll*, and *proposed* refer to the AMDSC results obtained with the images 0000, 0240, 0400, and 0640 (Figure 38); the deviations in blobs quality



Source: The author

Figure 38 – (cont.) Results of blob extractors for image set *rotation*.

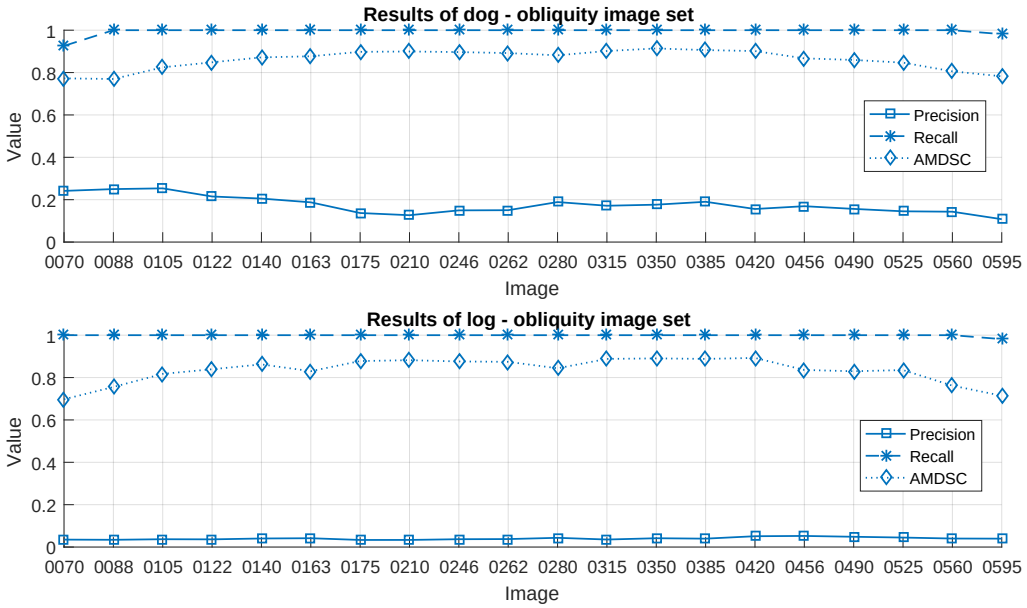
observed for these images are actually fruit of the stated variance issue.

5.4.3 Image set *obliquity*

The *obliquity* image set is intended to evaluate the impact caused by distorted features over the extractors. The hypothesis to be checked here is that extractors based on circular blobs (*dog*, *log*, and *ocv*) will not perform well with this image set. In this set, the effects of perspective distortions are more pronounced at the beginning and at the end of the image sequence. Approaching the middle of the sequence pattern features are almost free of distortions. Refer to Table 11 for the per-image results obtained by extractors in numerical values.

The extractors *dog* and *log* did not maximize RECALL over all images of this set (Figure 39). Not surprisingly these extractors failed in extracting features exactly while processing images of the sequence extremities (i.e., the images captured from severe oblique viewpoints). When the increased obliquity did not make extractors produce false negative results, it lowered the quality of true positive ones (notice that AMDSC decreases with

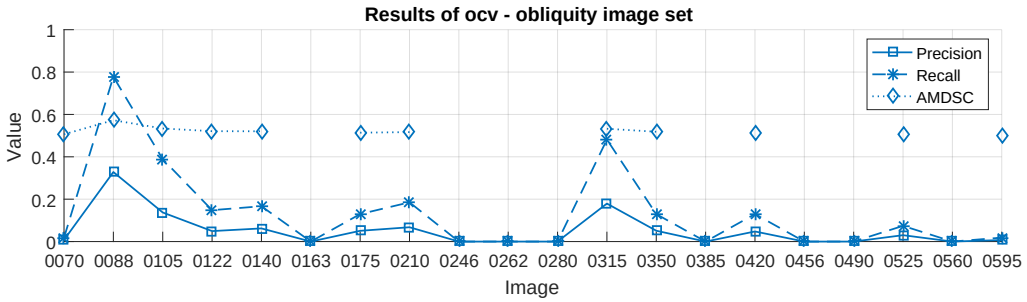
the increase in obliquity). Therefore, the results corroborate the hypothesis previously raised.



Source: The author

Figure 39 – Results of blob extractors *dog* and *log* for image set *obliquity*.

There is not much to add about *ocv* except by one observation, it would suffer from the same problem as the last two extractors if it would not be impaired by its inability to estimate the features real size in first place. Results of *ocv* are in Figure 40.

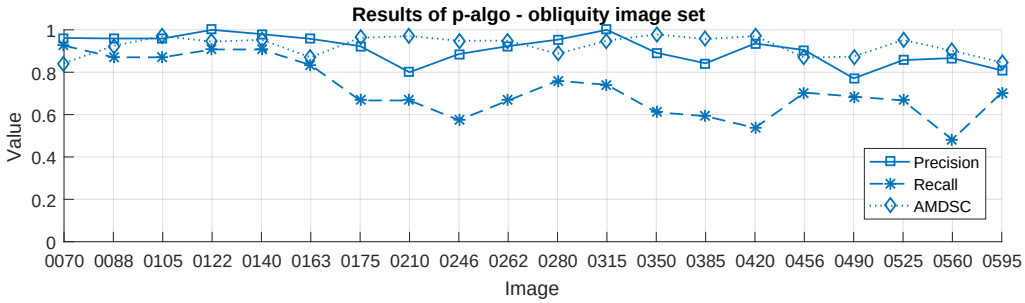


Source: The author

Figure 40 – Results of blob extractor *ocv* for image set *obliquity*.

Differently from the last three extractors, *p-algo* can produce blobs with arbitrary shapes. This characteristic made this extractor to achieve good AMDSC results (the best among all extractors). However, it did not perform well regarding RECALL (Figure 41) because of the same reasons detailed in the previous image set subsections.

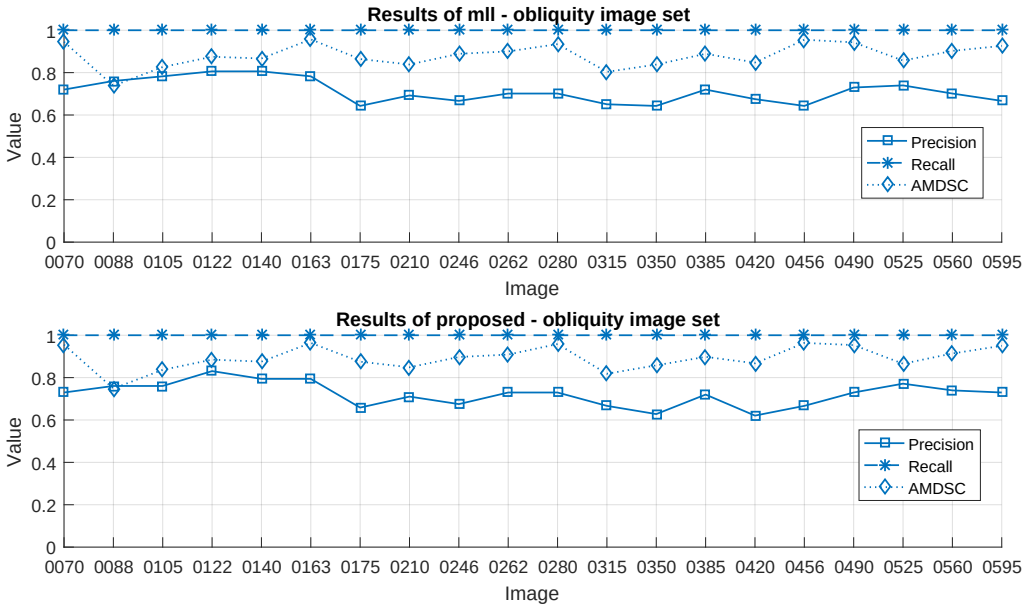
The extractors *mll* and *proposed* once again presented virtually identical results (Figure 42), as expected. However, a noteworthy observation regarding the subtle difference in their PRECISION results (also observable in others image sets) is made here. It was stated



Source: The author

Figure 41 – Results of blob extractor *p-algo* for image set *obliquity*.

that the slice selection criterion employed by each extractor is the only difference between them — something that should not be enough to make these extractors to produce distinct amounts of blobs. Therefore, provided that the extractors produced identical RECALL results, they should also produce the same PRECISION output, something that did not turn out to be true. The reason for such divergence is attributed to two factors that affect the amount of blobs considered by the evaluator. The first one is the culling mechanism described at the end of Subsection 3.2.4, which may discard some blobs depending on their shape. The second factor is the filtering mechanism used by the evaluator, which may also discard blobs if their size is out of the stipulated range (Section 4.3).

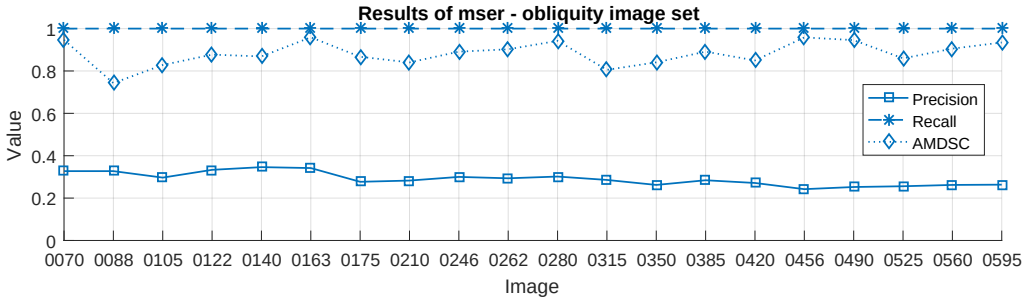


Source: The author

Figure 42 – Results of blob extractors *mll* and *proposed* for image set *obliquity*.

The *mser* extractor performed similarly to *mll* and *proposed* regarding RECALL and AMDSC (Figure 43). Although, these two last extractors outperformed the former by a reasonable margin when considering PRECISION. While *mll* and *proposed* achieved

results invariantly greater than 0.6 for this metric, *mser* did not surpass 0.4. In average this difference is even bigger, with the two better extractors achieving results above 0.7 and the worse below 0.3.



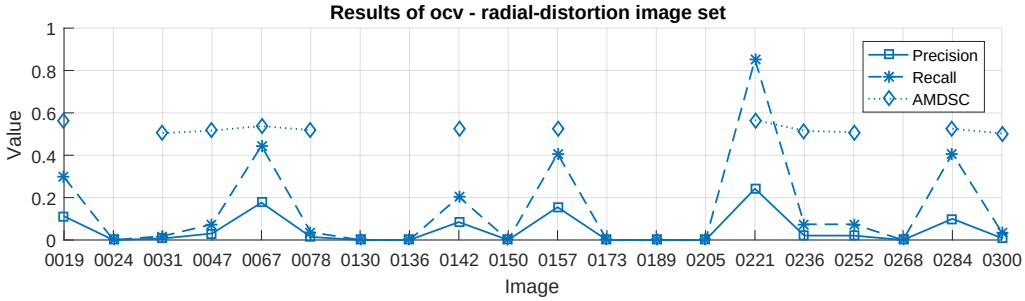
Source: The author

Figure 43 – Results of blob extractor *mser* for image set *obliquity*.

5.4.4 Image set *radial-distortion*

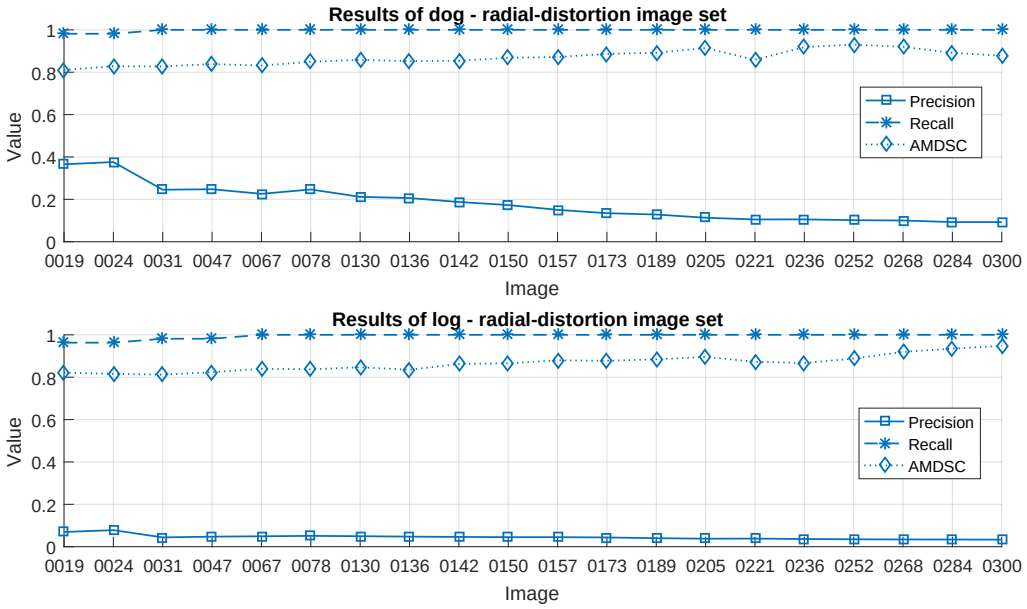
This image set explores the effects of radial distortions on the extractors performance. In this set, the first images were captured from a close viewpoint producing high distorted images. As the camera moved away from the pattern distortions reduced and the scene background was made more evident. Strictly speaking, radial distortions are not the same as perspective distortions, but they cause alike impacts over the blob extractors. In effect, the results obtained with this image set are anticipated to be similar to those presented in last subsection. Refer to Table 12 for the per-image results of this set in numerical values.

For the extractors that can only produce perfect circular blobs (*dog* and *log* are included here, but not *ocv* because of its chronic problem (Figure 44)) it is possible to notice an overall improvement in blobs quality with the decreasing of distortion (Figure 45). Not to mention that *dog* and *log* once again failed in extracting some features from the most distorted images. See Figure 46 for an illustration of how different from the ground-truth regions are the blobs extracted by *dog*.



Source: The author

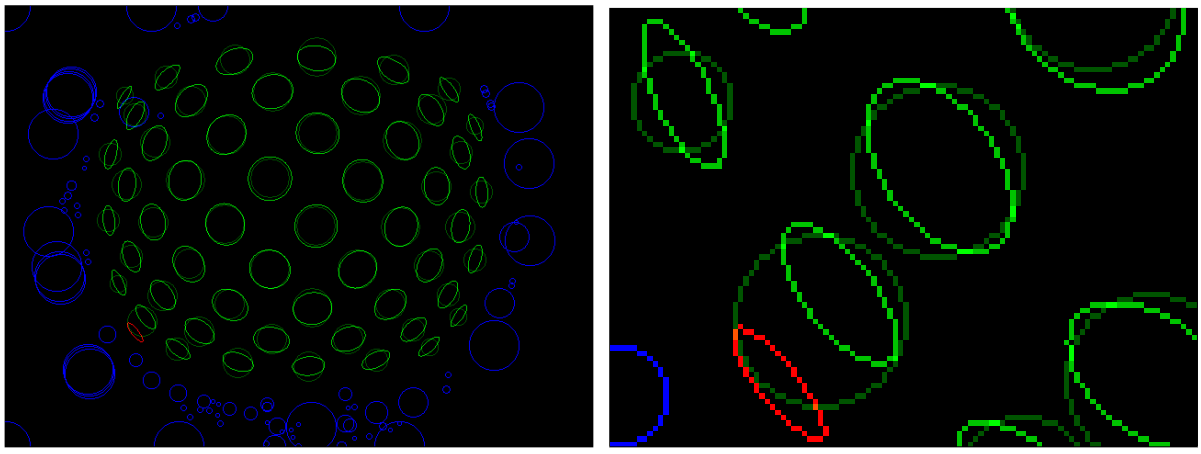
Figure 44 – Results of blob extractor *ocv* for image set *radial-distortion*.



Source: The author

Figure 45 – Results of blob extractors *dog* and *log* for image set *radial-distortion*.

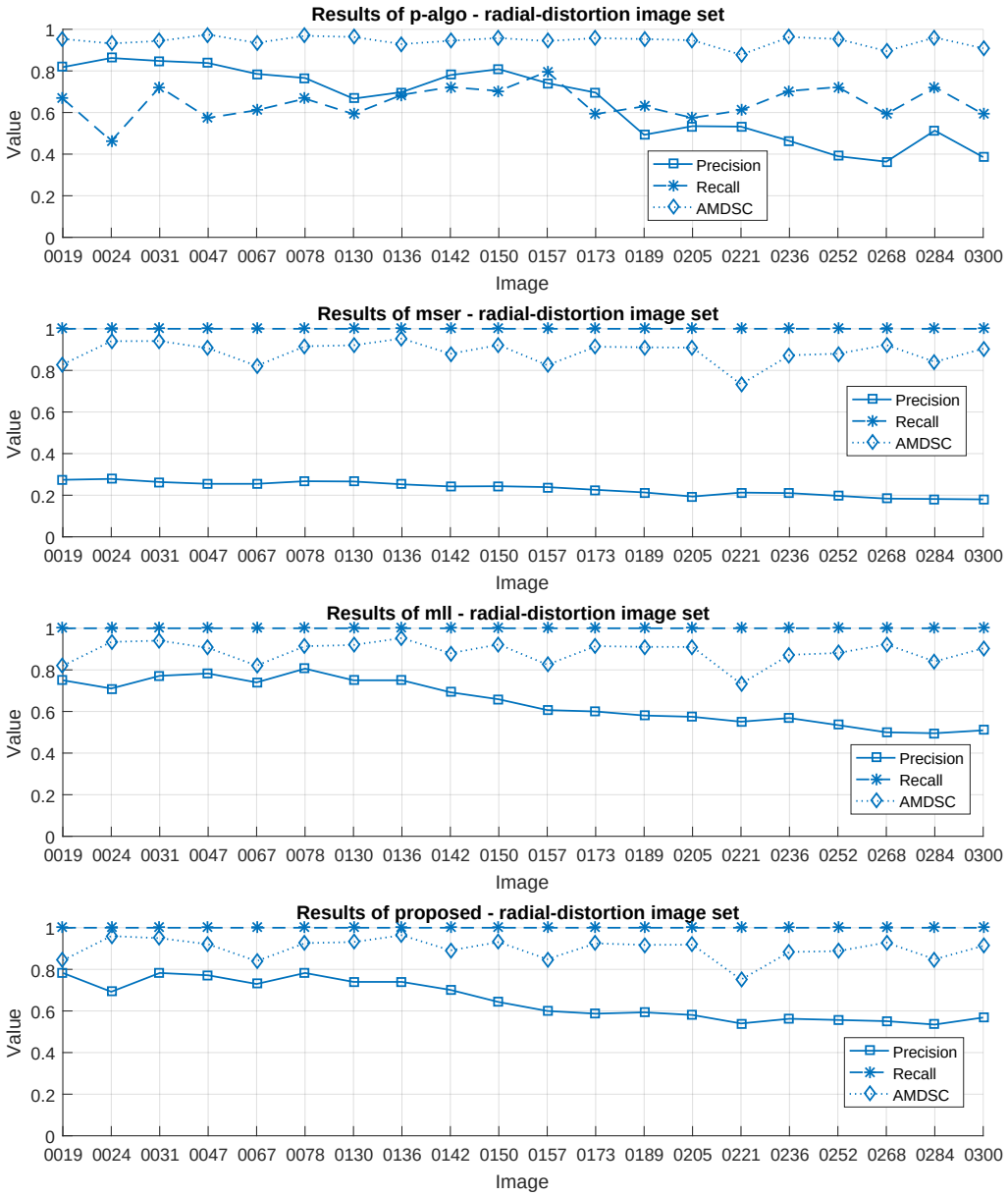
Regarding the other extractors (Figure 47), *p-algo* performed poorly once more (low RECALL results). The extractors *mser*, *mll*, and *proposed* maximized RECALL by finding all pattern features. When comparing these three last extractors, *mser* produced the worst PRECISION results, as usual.



(a) (b)

Source: The author

Figure 46 – The image in (a) shows how the blobs extracted from input image 0019 by *dog* were classified by the analyzer. In (b), the bottom left portion of the pattern is shown in detail. Note how loosely the true positive blobs (circles in dark green) overlap with the ground-truth regions (ellipsis in bright green). Observe also that distortions mislead *dog* to extract the two features at the bottom left corner of the pattern as a single blob.



Source: The author

Figure 47 – Results of blob extractors *p-algo*, *mser*, *mll*, and *proposed* for image set *radial-distortion*.

As a final comment, all extractors (except *ocv*) presented a decreasing trend in PRECISION that does not correlate with the decreasing in distortion. This behavior is in fact related to the cluttered scene background, which is made more evident over the images of this set.

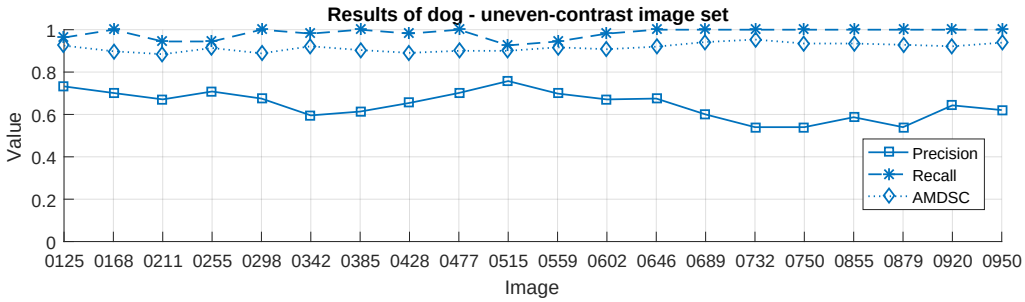
5.4.5 Image set *uneven-contrast*

The two next image sets are characterized by limited lighting conditions. Despite these conditions have been intentionally created for the purpose of the experiments, they make good scenarios for assessing how well the extractors can perform in harsher

environments. It is worth mentioning that, among all image sets, these are the ones that most influenced the parameterization of the extractors driving it to a more sensitive configuration. After all, features in poorly-lit images are less salient than in well-lit ones requiring more acuity from the extractors.

In particular, the *uneven-contrast* set presents images captured from a non-uniform illuminated scenario, which was created by using a flashlight. Despite all images of this set depict the pattern features with variable contrast, the first half[2] images were captured with a better overall illumination than the second half. In the first half, some parts of the pattern are illuminated by the peripheral light beans of the flashlight while the others are not directly illuminated at all. In the second half, some parts are illuminated by the central light beams while the others are illuminated by the peripheral beans. The per-image results (in numerical values) obtained by each extractor with this image set are presented in Table 13.

The *dog* extractor failed in extracting a few pattern features from some images of the first half of this set (Figure 48). Therefore, the darker appearance of these images impacted in some way on this extractor. For the second half images (images with a slight better illumination) all features were extracted. The AMDSC metric does not seem to be affected by the peculiar lighting condition of this set (in comparison with the other better-lit scenarios). On the other hand, PRECISION was notably higher for this set than it was for others (an expected outcome since poorly-lit images have fewer saliences that can be extracted).

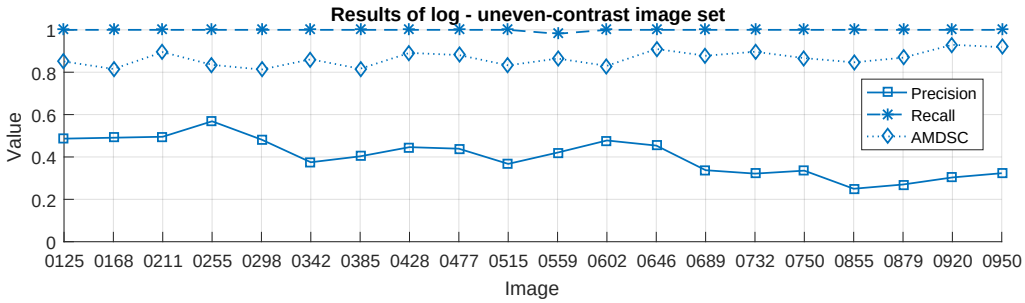


Source: The author

Figure 48 – Results of blob extractor *dog* for image set *uneven-contrast*.

Regarding the *log* extractor, the inferior PRECISION results observed in other image sets (in comparison with *dog*) are also observable in this set (Figure 49). However, in this set *log* achieved slightly better RECALL results missing just one feature from the image 0559.

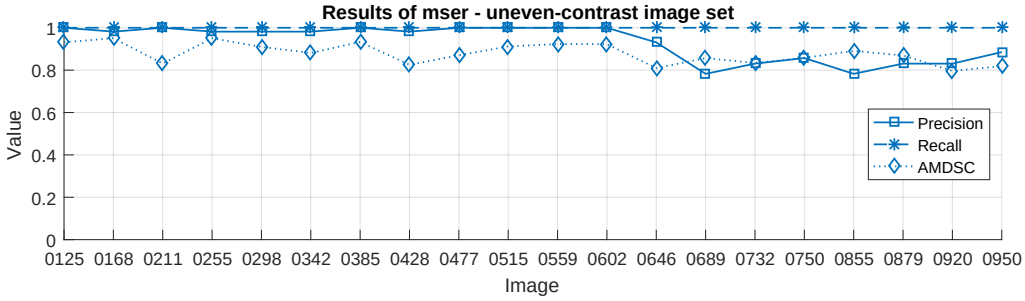
[2] The term "half" is used just for convenience of designation because it is not precisely at the middle of the image sequence that the overall illumination condition changes.



Source: The author

Figure 49 – Results of blob extractor *log* for image set *uneven-contrast*.

The *mser* extractor obtained remarkable results regarding the three metrics (Figure 50). As usual, RECALL was maximized over all images and AMDSC was high. Although, differently from the last image sets, this time PRECISION was high as well (in average above 0.9). The *mser* extractor also turned out to be sensible to the illumination change that occurs around the middle of the image sequence (notice the decreasing in PRECISION from this point indicating that more false positive blobs were extracted).



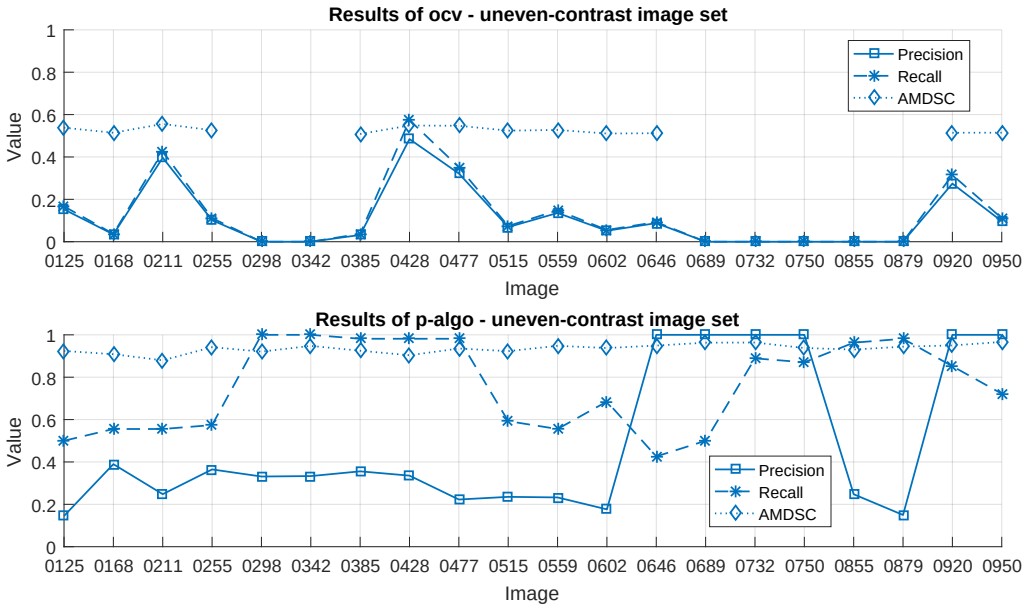
Source: The author

Figure 50 – Results of blob extractor *mser* for image set *uneven-contrast*.

As in other image sets, the extractors *ocv* and *p-algo* obtained inconsistent results due to same reasons previously addressed (Figure 51).

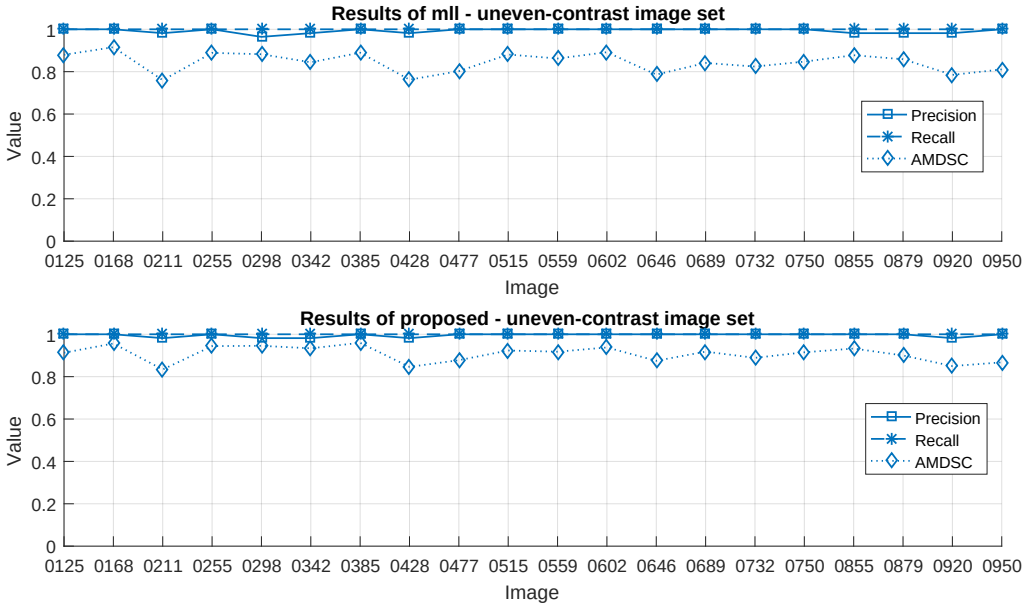
The *proposed* extractor performed the best between all extractors. It maximized RECALL over all images and at the same time produced the best PRECISION results for this set. The AMDSC was high as well (above 0.9 in average).

The *mll* extractor also did this set well. However, it is important to draw the attention to the fact that in this set the difference between *mll* and *proposed* regarding AMDSC was larger (Figure 52), with the latter extractor outperforming the former (more details about it in next subsection).



Source: The author

Figure 51 – Results of blob extractors *ocv* and *p-algo* for image set *uneven-contrast*.



Source: The author

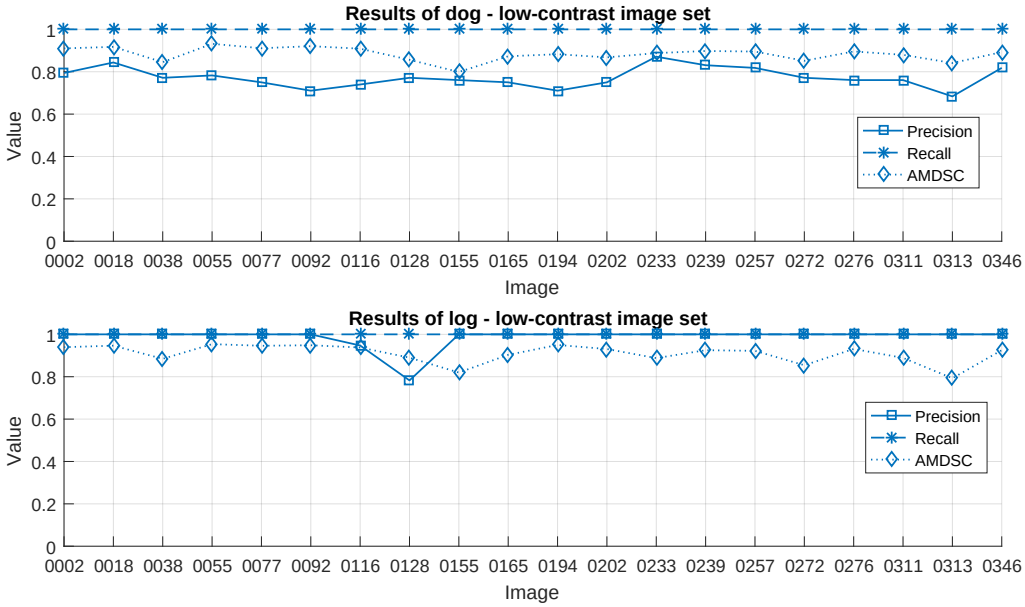
Figure 52 – Results of blob extractors *mll* and *proposed* for image set *uneven-contrast*.

5.4.6 Image set *low-contrast*

The *low-contrast* image set is characterized by the constant and deficient lighting condition set for its recording. This condition caused insufficient contrast even between the features and the base of the pattern, which were made in highly contrasting colors (black and white, respectively). As a result, this set posed difficulties not only for the extractors but also for the individuals in charge of the images hand-labeling.

Interestingly enough, this image set made extractors to produce the best PRECISION results over all image sets. At a first glance this outcome seems contradictory (after all, this set is supposedly the most challenging one), but, as previously stated, poorly-lit images have fewer saliences that can be extracted as blobs. In that case, the amount of false positive results is reduced, consequently improving PRECISION. Refer to Table 14 for the per-image results of this set in numerical values.

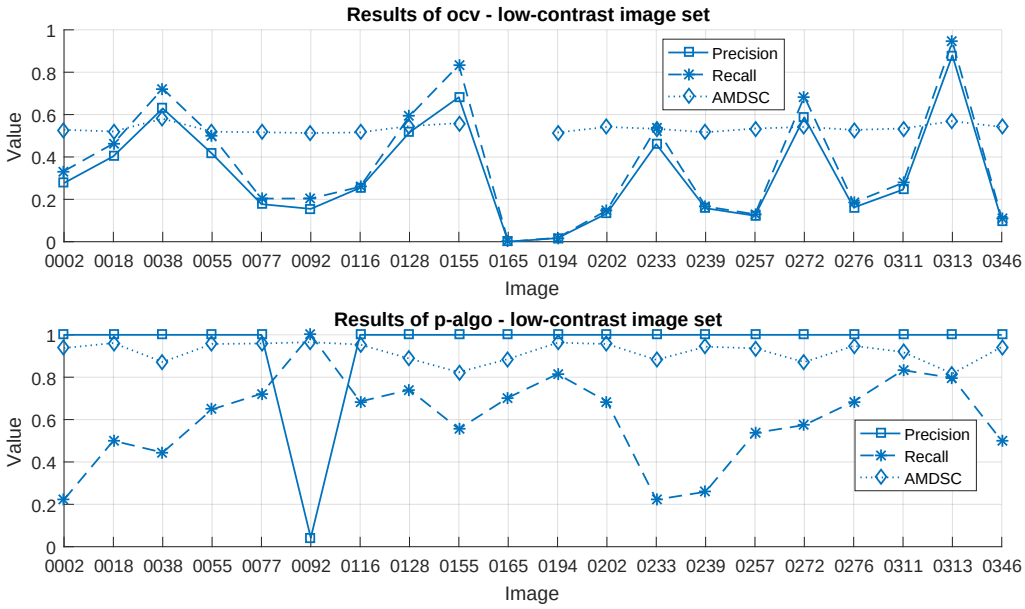
The extractors *dog* and *log* maximized RECALL over all images, therefore performing better than in the last image set (Figure 53). This outcome suggests that more salient features positioned in inclined areas (i.e., the way that some pattern features are presented in the last image set) are harder to be extracted than less salient features positioned in planar areas (the way that every feature of this image set is presented). The AMDSC metric was also high for both the extractors.



Source: The author

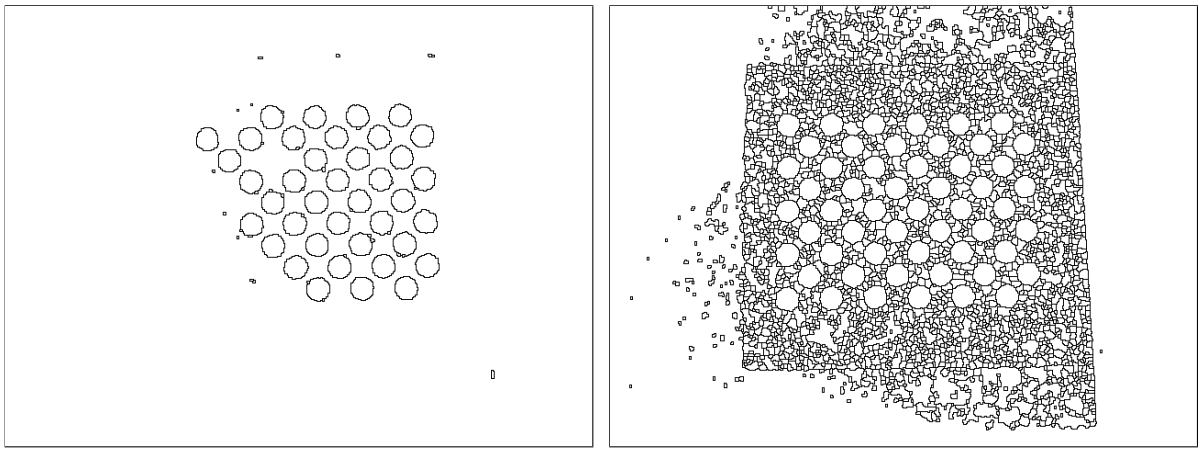
Figure 53 – Results of blob extractors *dog* and *log* for image set *low-contrast*.

The extractors *ocv* and *p-algo* performed inconsistently as usual (Figure 54). In special for *p-algo*, it was noticed that the uncanny results produced with image 0092 (maximum RECALL and PRECISION close to zero) is fruit of the different number of hierarchical level output by the technique. As a general rule, an inferior number of hierarchical levels means that the image will be segmented into more regions. While image 0092 caused the extractor to output level one, the others caused it to output two. This unitary difference turned out to produce drastic changes in the segmentation degree achieved by each image, which ranged from an under-segmented image (without some important contours) to an over-segmented one (with lots of non-significant contours). Refer to Figure 55 to visualize this variation.



Source: The author

Figure 54 – Results of blob extractors *ocv* and *p-algo* for image set *low-contrast*.



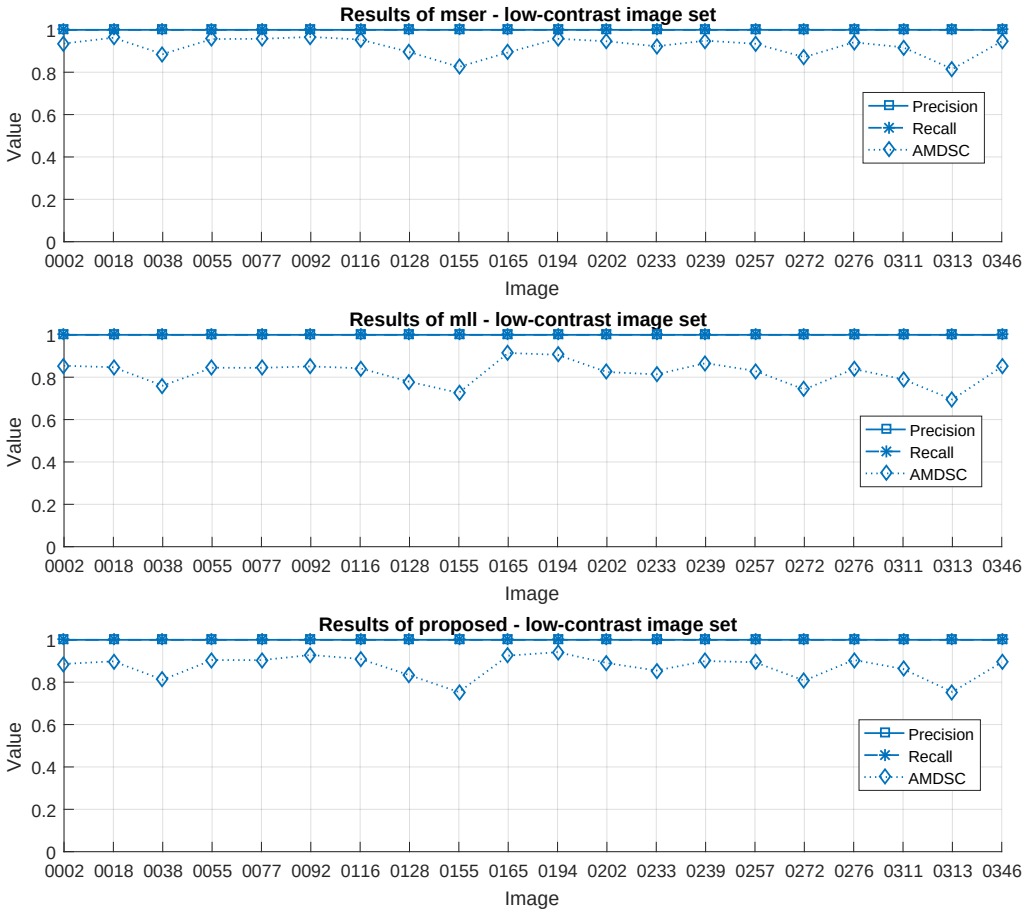
(a) (b)

Source: The author

Figure 55 – Respectively in (a) and in (b), outputs produced by *p-algo* for input images 0077 and 0092. Even though these input images are consecutive (and consequently similar to each other), one was under-segmented while the other was over-segmented.

The only extractors to achieve maximum Recall and Precision over all images of this set were *mser*, *mll*, and *proposed* (Figure 56). These extractors also obtained high AMDSC results, with *mser* performing the best, *proposed* in second, and *mll* in third.

Similarly to what happened with the *uneven-contrast* set, the *low-contrast* set also caused the extractors *proposed* and *mll* to produce more distinctive results, regarding the AMDSC metric. This fact evidences that the (supposed) sharpening operator applied



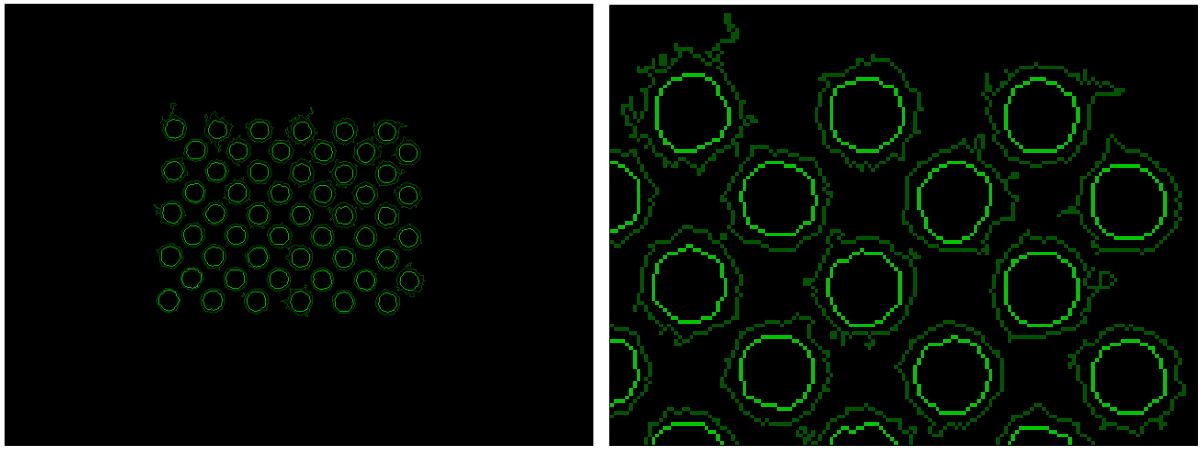
Source: The author

Figure 56 – Results of blob extractors *mser*, *mll*, and *proposed* for image set *low-contrast*.

by the camera is not as effective in preventing *mll* from overestimating the features real size for images captured under low-lighting conditions as it was for well-lit scenarios. This hypothesis sounds reasonable because many enhancing mechanisms are characterized by being proportional to the image contrast ratio. In other words, the more contrasting regions of an image are the more enhanced ones with greater "bumps". Figure 57 shows how the blobs extracted by *mll* compare with the ground-truth regions. Finally, it is worth mentioning that the difference in AMDSC would be more pronounced if a pattern with a greater ratio between features distance and features radius was used (e.g., if the pattern used had its features reduced in size causing larger gaps between them).

5.4.7 Overall analysis

This subsection summarizes the results presented in the last subsections. This final analysis is based on the per-set average results obtained by each extractor, which are available in Figure 58 and in Table 15. At the end, hypothesis tests are conducted by using the whole set of images.



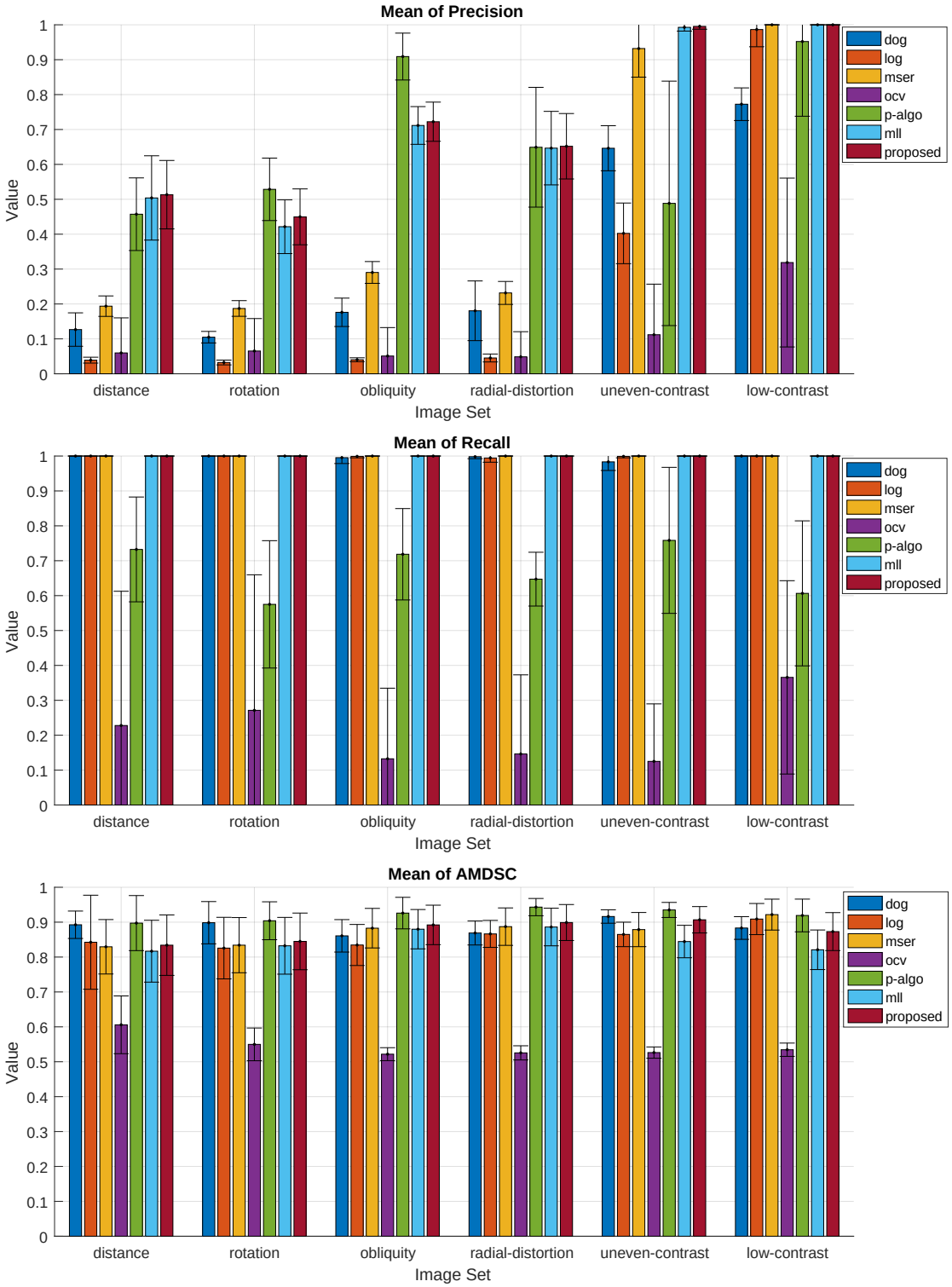
(a) (b)

Source: The author

Figure 57 – The image in (a) shows how the blobs extracted from input image 0272 by *mll* were classified by the analyzer. In (b), the top right portion of the pattern is shown in detail. Note that despite all blobs (regions in dark green) have attained to the minimum score required to produce true positive results, they are oversized when compared with the ground-truth regions (in bright green).

But before analyzing the extractors, it is important to detail some considerations taken into account about the relevance of each metric. The Recall was considered the foremost metric for the extractors assessment and therefore prevails over the others. This decision is justified because, for most computer vision applications, the faults committed during the feature extraction stage are not repaired by the subsequent ones. And in such a circumstance, the system overall performance is inevitably impacted (e.g., in a pattern detection system the inlier features overlooked by the extractor would make it impossible to detect the pattern). The metrics Precision and AMDSC are of less importance than Recall because of their reduced impact over a hypothetical system. However, these are the metrics that ultimately permit extractors that equated in Recall to be ranked. Finally, the metrics Precision and AMDSC were considered equal in importance to each other.

The extractors *dog* e *log* performed well in overall, although they missed a few features from some images of the sets *obliquity*, *radial-distortion*, and *uneven-contrast*. Both these extractors were also characterized by low Precision results (in special for well-lit images). It is true that this last issue was caused by the way that the extractors were parameterized, which made them extremely sensitive to the images saliences. It is also worth noting that it was a "necessary evil" to enable *dog* and *log* to work within the poor-lit scenarios (but still not achieving maximum Recall over all images). Another important remark about *dog* and *log* is that they can only produce circular blobs. This characteristic makes them not suitable for scenarios in which the elements of interest



Source: The author

Figure 58 – Mean of the three metrics evaluated for all blob extractors for all image sets of the printed pattern experiments.

have arbitrary shapes or even when these elements are circles and severe distortions apply (recall image sets *obliquity* and *radial-distortion*).

Despite results pointing out *ocv* as the worst extractor among all, by closely inspecting the images output by the analyzer one can notice that most of the pattern features were properly extracted by this extractor (regarding all image sets). As previously stated, the problem with *ocv* is its propensity to underestimate the features real size leading the analyzer to make poor evaluations. If it had not occurred, *ocv* would have achieved a much better evaluation.

The *p-algo* extractor achieved remarkable results regarding PRECISION and AMDSC (the best for most image sets), but its performance was not considered satisfactory because of the poor RECALL results. Not to mention that *p-algo* at times produced inconsistent results for similar images, with some images being under-segmented, others over-segmented.

The most succeeded extractors were *proposed*, *mll*, and *mser* — they were the only extractors to maximize RECALL over all images of all sets. Among these three extractors, *proposed* can be considered the best because of its superior PRECISION (in comparison with *mser*) and higher AMDSC (in comparison with *mll*). For all image sets captured from well-lit scenarios, *proposed* outperformed *mser* in PRECISION by at least 140% and as much as 181%. For the others sets (the ones captured under limited lighting conditions) these two extractors presented less discrepant PRECISION results. Regarding the quality of blobs extracted by *proposed* and *mll*, it is known that the camera used in the experiments produced artifacts that benefited *mll*. Therefore, the author believes that the results obtained with this case study do not reflect the real difference that exists between the two extractors. A case study conducted with a camera devoid of enhancing capabilities and using a pattern with more scattered features would certainly cause *mll* to output inferior AMDSC results, in which the reason for this would be oversized blobs.

Statistical hypothesis tests have been conducted for a better quantitative analysis of the extractors *mser*, *mll*, and *proposed*. Since these extractors achieved identical RECALL results, only the metrics PRECISION and AMDSC were regarded for this purpose. The images of the six sets were put together to achieve an appropriate sample size, resulting in a single set of 120 images.

Before testing the techniques against each other, it must be verified whether the data of each metric come from a population that is normally distributed. This hypothesis was checked by using the Lilliefors test. As it turned out, for the both metrics PRECISION and AMDSC, the normality hypothesis was rejected at the 1% significance level for at least one of the extractors. Therefore, since the data normality cannot be assumed in all cases, a non-parametric test was chosen; the Wilcoxon signed rank test for paired samples.

The first test is concerned with PRECISION results of *mser* and *proposed*; the null hypothesis is that the median of PRECISION differences of *mser* and *proposed* is 0, which is tested against the alternate that it is less than 0. The null hypothesis was rejected at the 1% significance level (p-value was 6.47×10^{-17}). Therefore, there is enough statistical evidence to conclude that the median of PRECISION obtained by *mser* is less than the median of PRECISION obtained by *proposed*. A similar test was performed regarding the AMDSC metric. Once again, it was concluded that, at the 1% significance level, the median of AMDSC obtained by *mser* is less than the median of AMDSC obtained by *proposed* (p-value was 1.54×10^{-4}).

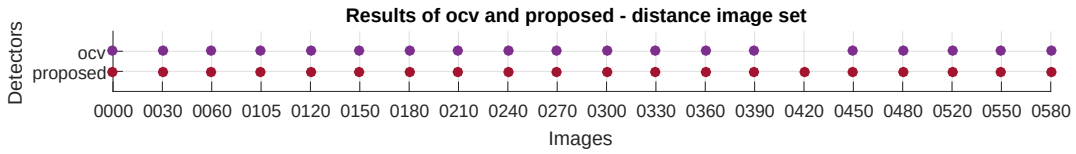
Concerning PRECISION results of *mll* and *proposed*, the null hypothesis is that the median of PRECISION differences of *mll* and *proposed* is 0; the alternate hypothesis is that it is less than 0. The null hypothesis was rejected at the 1% significance level (p-value was 7.21×10^{-7}), which means that there is enough statistical evidence to conclude that the median of PRECISION obtained by *mll* is less than the median of PRECISION obtained by *proposed*. A similar test was performed regarding the AMDSC metric; p-value was 9.98×10^{-22} indicating that, at the 1% significance level, the median of AMDSC obtained by *mll* is less than the median of AMDSC obtained by *proposed*.

5.5 Pattern detection results

This section presents the results obtained by the two pattern detectors introduced in Subsection 4.4.1 (*ocv* and *proposed*). The same six image sets previously used for evaluating blob extractors are used here.

5.5.1 Image set *distance*

The *proposed* pattern detector succeeded over all images of the *distance* set (Figure 59). Therefore, the reduction in features apparent size which occurs over the images of this set did not affect this detector.

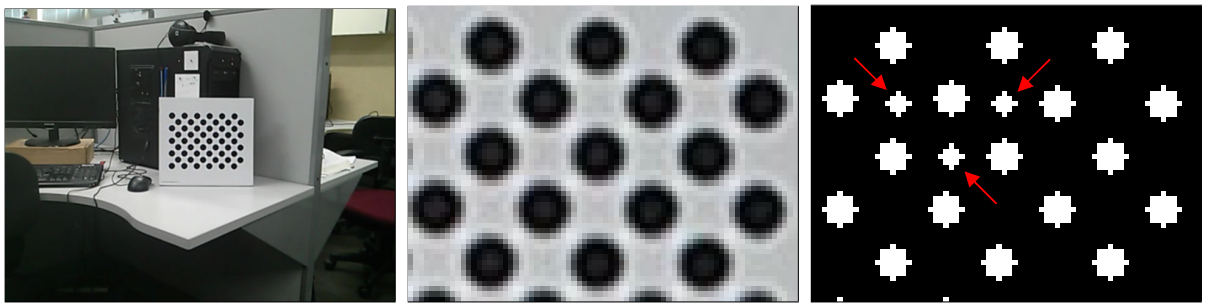


Source: The author

Figure 59 – Pattern detectors results for image set *distance*. The result of each detector is presented on a per-image basis; the presence of a circle indicates that the detector succeeded in that frame. Images are identified by four-digit names drawn from their original frame numbers in the video footage.

On the other hand, *ocv* failed in detecting the pattern from one image of this set (image 0420). Interestingly enough, the failure occurred not in an image from the extremities of the set, but from its means. The explanation for this fact is curious. By inspecting the blobs extracted from the image that caused the failure, it is possible to notice the presence of outlier blobs among the inlier ones (Figure 60c). And as was observed while experimenting with the *ocv* detector, by some reason it can only tolerate the presence of such undesirable elements when they are outside the pattern region. The presence of a single outlier blob among inlier ones ultimately cause *ocv* detector to fail.

Still concerning the *ocv* detector failure, it was investigated why the image in Figure 60a caused this problem but not others. It was noticed that the features apparent size reaches a critical point from that image — from this size the "bumps" surrounding neighbor features (recall it from Figure 36) start connecting to each other, creating a favorable condition to the emergence of outlier blobs in the regions between these features (Figure 60b). The images subsequent to 0420 have not caused this problem because of a simple reason; the regions in between the pattern features reduce to a point of not permitting the emergence of blobs with area superior to 21 pixels. In other words, the outlier blobs that should appear in those regions are actually discarded because they do not attain the minimum area requirement (recall the *ocv* extractor parameterization from Subsection 4.4.1). For this reason, the last images of this set are free of such unwelcome artifacts.



(a) (b) (c)

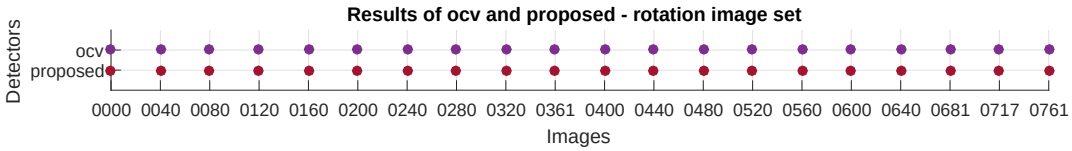
Source: The author

Figure 60 – The image 0420 of the *distance* set is in (a). In (b), the top right portion of the pattern is shown in detail. Finally, the blobs extracted from that portion of the image is shown in (c). Notice the presence of three outlier blobs (pointed by red arrows) among the inliers.

As a final observation, the *proposed* blob extractor is as susceptible to produce outlier blobs among the inliers as the *ocv*. However, since the pattern searching stage of the proposed solution is robust to these unwanted elements, the *proposed* pattern detector fared better than its competitor.

5.5.2 Image set *rotation*

The images of the *rotation* set have been flawlessly detected by both the detectors (Figure 61). Therefore, the hypothesis that the images in which the pattern appears upside-down could "confuse" detectors was not confirmed (i.e., detectors turned out to be equally robust to such a rotation transform).

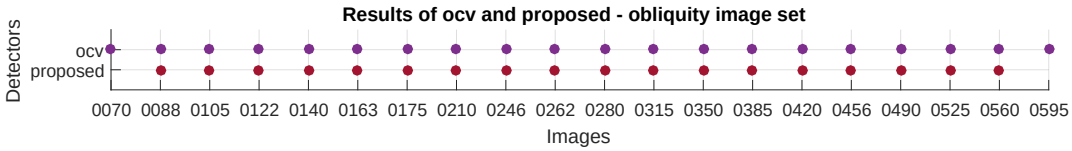


Source: The author

Figure 61 – Pattern detectors results for image set *rotation*.

5.5.3 Image set *obliquity*

While the *ocv* detector has succeeded over all images of the *obliquity* set, *proposed* has failed in the first and in the last image (Figure 62). It is worth noting that these two failures were not caused by the absence of inlier blobs (after all, the *proposed* blob extractor has maximized RECALL for both the images), but by the pattern searching stage which did not prosper in mapping all inlier blobs to the vertices of the pattern template.



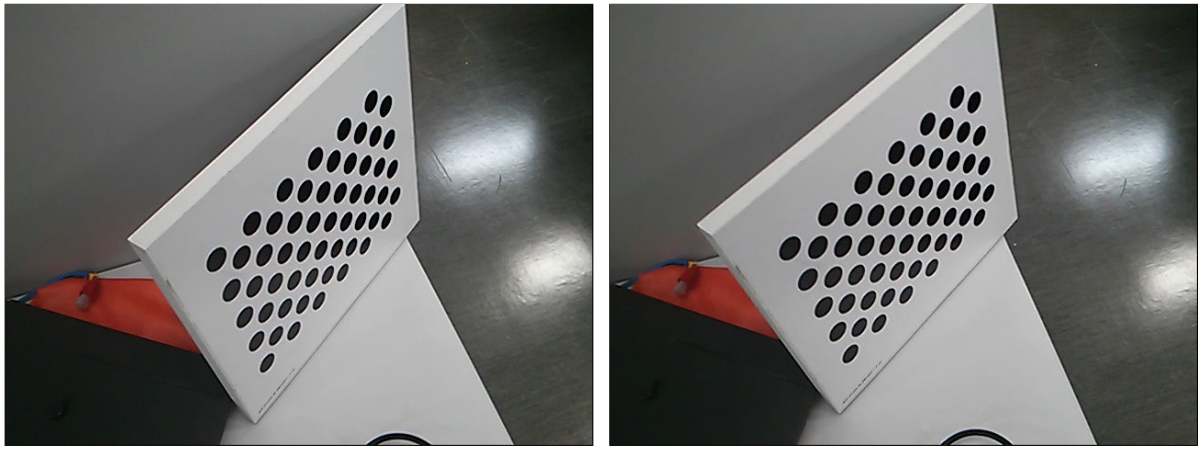
Source: The author

Figure 62 – Pattern detectors results for image set *obliquity*.

Those failures indicate that the parameterization used with the pattern searching procedure was not permissive enough to accommodate the perspective distortions suffered by those images. Relaxing parameters would make the pattern searching succeed, although, for not risking it to produce wrong detections in other situations this idea was abandoned. Refer to the images in Figure 63 for getting the notion of the level of distortion tolerated by *proposed*.

5.5.4 Image set *radial-distortion*

The *ocv* detector failed in detecting the pattern from most images of the *radial-distortion* set (as seen in Figure 64, it succeed in only 7 out of 20 images). Once more it was investigated whether these failures were caused by the absence of inlier blobs, but this

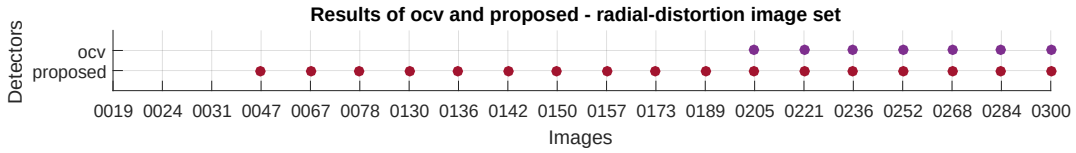


(a) (b)

Source: The author

Figure 63 – The input image 0070 (the first of the *obliquity* set) is in (a); it has not been detected by the *proposed* detector because of its distortion. The next image of the set (the image 0088) is in (b); from this image *proposed* succeed.

assumption was not confirmed (i.e., the problem resides in the pattern searching procedure itself). An interesting observation that must be made here is that, while perspective distortions seems to be innocuous to the *ocv* detector (recall that *ocv* succeed over all images of the *obliquity* set), even small degrees of radial distortions can prevent *ocv* from detecting the pattern (the first image of the *radial-distortion* set in which *ocv* succeed is in Figure 65b, which shows a not very deformed pattern).



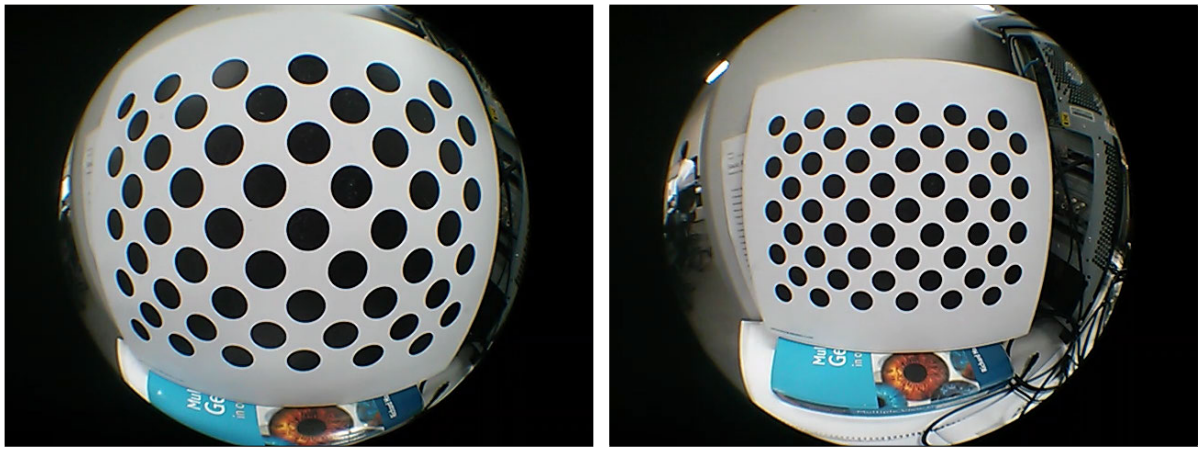
Source: The author

Figure 64 – Pattern detectors results for image set *radial-distortion*.

The origin of the *ocv* detector failures seems to be related to the particularities of the radial distortions and how they ultimately affect the shape of the imaged pattern. Differently from perspective distortions, radial distortions do not preserve straight lines. These effects become more evident as the object of interest approaches the image edges, such as in the first images of the *radial-distortion* set (notice in Figure 65a how the rows and columns of the pattern of features, which are straight, become curved close to the image edges). The author's best guess is that the *ocv* detector was not designed to bear the effects of radial distortions unless they are mild deformities over the pattern extent (such as in the last images of the set). The final observation about the *ocv* pattern detector (and consequently about the OpenCV camera calibration functionality) is that it may not

suit scenarios in which cameras are equipped with wide angle lens.

The *proposed* pattern detector achieved better results than *ocv* (Figure 64), although not perfecting it (17 out of 20 images were detected). The *proposed* detector has succeeded from the image in Figure 65a, which has a much greater degree of distortion than the image in Figure 65b. Under those circumstances, it is possible to state that the proposed searching procedure tolerates well the effects of radial distortions. The next case study (Chapter 6) presents a scenario in which the pattern undergoes more severe arching; not because of the camera lens optical characteristics, but because of actual changes in the pattern/pipe shape due to its flexibility.



(a) (b)

Source: The author

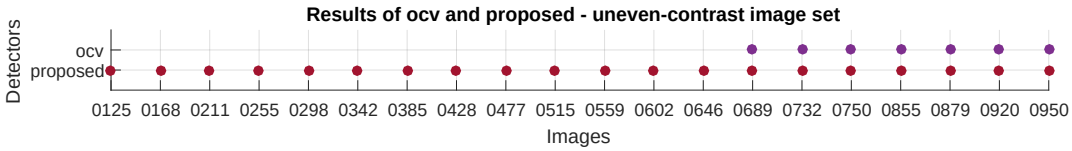
Figure 65 – The input image 0047 is in (a); it was the first image of the *radial-distortion* set in which the *proposed* detector succeeded. The image 0205 is in (b); the first one to be detected by *ocv*.

5.5.5 Image set *uneven-contrast*

Differently from the last four image sets, the sets *uneven-contrast* and *low-contrast* have been recorded by keeping the camera at a fixed viewpoint from the pattern of features and with the pattern perfectly facing the camera. At first glance, this creates a convenient circumstance for any pattern detector because the shape of the imaged pattern do not change over the images and it is free of distortions. However, the lighting conditions forced while recording these sets drastically affect the blob extractors, which ultimately impact detectors overall performance.

While the *proposed* detector succeeded over all images of the *uneven-contrast* set (Figure 66), *ocv* could only achieve the same feat with the last seven images of the set (the better-lit ones). By inspecting the debug images output by the *ocv* detector, it is possible to confirm that its failures have been caused by the lack of inlier blobs in the

darker regions of the input images (i.e., the *ocv* blob extractor has missed the pattern features not directly lit by the flashlight).



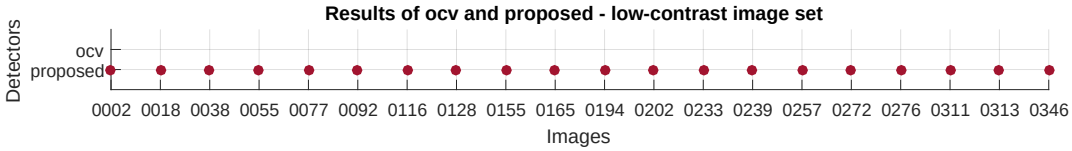
Source: The author

Figure 66 – Pattern detectors results for image set *uneven-contrast*.

It has been tried to increase the *ocv* extractor sensitivity in the hope of overcoming this problem, but it has caused another difficulty — the emergence of outlier blobs among the inlier ones in the well-lit regions of the image (as was presented in Subsection 5.5.1). In addition, if this increased sensitivity had used with the better-lit image sets, *ocv* detector performance would have worsened for that sets.

5.5.6 Image set *low-contrast*

In view of the poor performance of the *ocv* detector with the darker images of the *uneven-contrast* set, it is easy to predict that this detector is fated to be unsuccessful with the *low-contrast* set as well. The *ocv* worst performance has been achieved with the *low-contrast* set indeed — it failed with every image of this set. All these failures were caused by the absence of inliers in the set of extracted blobs. On the other hand, the *proposed* detector succeed over all images once more (Figure 67).



Source: The author

Figure 67 – Pattern detectors results for image set *low-contrast*.

5.5.7 Overall analysis

No pattern detector perfected over all images. However, in the final quantitative analysis *proposed* outperformed *ocv* by achieving a superior number of detections — while *proposed* succeeded in 96% of the images processed, *ocv* achieved the mark of 61%.

The major characteristic that has harmed *ocv* performance is its inability to deal with outlier blobs among the inliers. This creates a delicate circumstance in which the benefit of increasing the extractor sensitivity (for maximizing the extraction of inliers)

is easily outweighed by the risk of producing some intrusive outliers. This limitation is naturally more harmful to images captured under non-uniform lighting conditions, although it has impacted in trivial scenarios as well (such as in the *distance* image set). The *ocv* pattern detector has also struggled with images radially distorted, as was evidenced with the *radial-distortion* set.

According to the experiments, the single unfavorable point of the *proposed* detector is its inferior tolerance to perspective distortions, although it was not possible to quantify this disadvantage since the *ocv* failing point is unknown (*ocv* succeed over all images of the *obliquity* set). For all other scenarios *proposed* has turned out to be a better detector than its competitor. In special for scenarios with limited lighting conditions (the sets *uneven-* and *low-contrast*), *proposed* presented valuable results by succeeding over all images.

As a final observation, the reference implementation of the detector *proposed* spent 2.96 seconds (in average) to process each of the well-lit images (the first four image sets) and 0.92 seconds to the poor-lit ones (the last two sets). This difference in execution time is due to the greater amount blobs extracted from the well-lit images (note in Figure 58 that the extractor *proposed* achieved inferior PRECISION results for these images).

5.6 Summary

This chapter presented the first part of the assessment of the technique proposed in this Thesis. The case study used to serve this purpose regards the detection of a printed pattern, which is commonly used in problems involving camera calibration and/or pose estimation. To cover a broad range of scenarios, six image sets were utilized, totalizing 120 images. Each image set addresses a distinct circumstance that can occur in real-world situations, such as the pattern viewed from different distances, the pattern rotated, the pattern deformed by perspective or radial distortions, and the pattern under deficient lighting conditions.

The analysis of the blob extractors indicated that the most succeeded ones were *proposed*, *mll*, and *mser*. Among these extractors, *proposed* was considered the best because of its superior PRECISION and AMDSC marks (confirmed by means of hypothesis tests). Additionally, it is worth noting that the good results achieved by *mll* were only possible because of the image enhancement feature of the camera system utilized in this case study. Therefore, *mll* would not equally succeed if a camera without such a feature had been used instead.

Regarding the evaluation of the pattern detectors, *proposed* outperformed its only competitor (*ocv*) for most of the image sets tested. In special, for the last two image sets, in which the lighting conditions were made as harsh as possible, *proposed* succeeded over all images while *ocv* failed entirely.

6 CASE STUDY 2 (PIPE IN DEEP SEA)

The purpose of this case study is to evaluate the proposed technique in a real scenario with odd characteristics. As will be detailed next, one of the most distinctive characteristics of the offshore scenario (regarding the pattern detection problem) is that pipes are non-rigid structures. This fact has a significant impact on the problem addressed — the pattern of interest changes over time. In addition, this case study also testifies that the applicability of the technique is not limited to printed patterns.

6.1 Scenario description

Offshore operations are conducted in deep underwater environments (at times surpassing 2000 m in depth). Consequently, ROVs (Remotely Operated Vehicles) assist most parts of the operations (WHITCOMB, 2000). There is no natural illumination in this kind of environment and the only light sources available are the ones attached to the ROV. Since illumination is scarce, ROVs are equipped with special underwater camera systems, which can capture images in low-light conditions (10^{-3} lux). These images are monochromatic, low resolution (usually NTSC or PAL standards), and tend to be blurred due to the underwater light scattering.

In this environment devoid of natural lighting, suspended particles are everywhere. Animals of the marine biome such as tiny fish and squid are also present, mainly in the shallower depths. Besides, some equipment such as floaters may be sparsely attached to the pipe. Buoyancy forces naturally compel floaters to position themselves strictly above the pipe, but nothing is placed behind it. The emptiness behind the pipe and the restrictive lighting conditions of the environment make the images background range from medium gray to black. Hence, in the deep ocean, it is as if there was a permanent black backdrop wherever you go.

In this harsh scenario, special flexible pipes are used for transporting oil and gas. They are permanently oscillating due to waves and ocean currents, and, if the oscillation reaches a critical amplitude, equipment installed at the pipe endings may be damaged (LI, 2012).

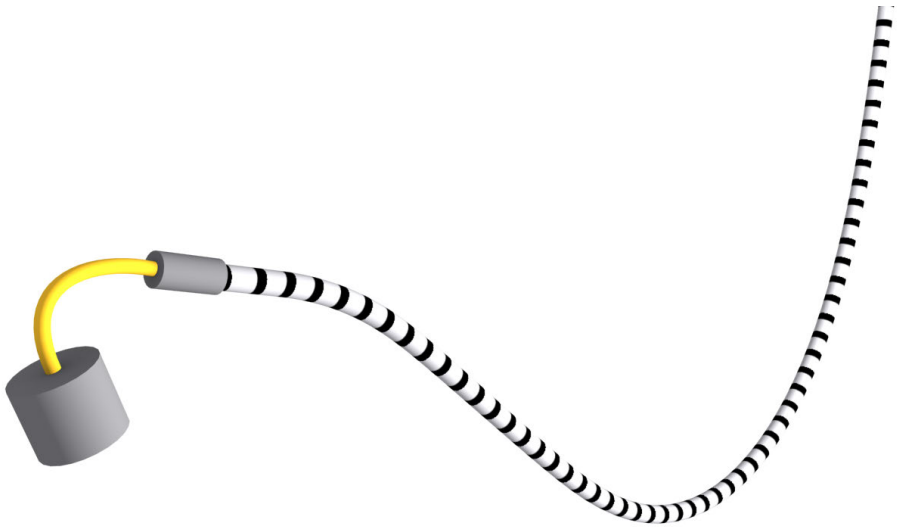
In order to mitigate the risks involved in such situations, a pipe monitoring tool based on 3D reconstruction is being developed by Santos et al. (SANTOS et al., 2015). This tool uses the proposed technique for detecting the pipe in images. Since the pipes are non-rigid objects, a calibrated stereo rig is used. This setup requires that features are extracted and matched across both images. Therefore, for each pair of captured images,

the detection technique is independently executed for each image.

6.2 Pattern of features

The purpose of the detection technique in this case study is to provide features that will be posteriorly matched across a pair of stereo images. However, since the pipes are coated with a plastic of uniform color, it is almost impossible to distinguish features over their surface.

In order to overcome this issue, regularly-spaced longitudinal sections of the pipe are marked with an alternating black and white[1] pattern (Figure 68). This can be easily achieved by using colored adhesive tapes or by painting the pipe surface. Since the region of the pipe that shall be monitored is short (about 15 m), this manual task is affordable (resulting in about 29 markings for a 35 cm diameter pipe). In this case, the white sections marked over the pipe (the elements that will be extracted as features by the detection technique) are also referred to as *vertebrae*. The reason for this is the resemblance of a marked pipe with a vertebral column.



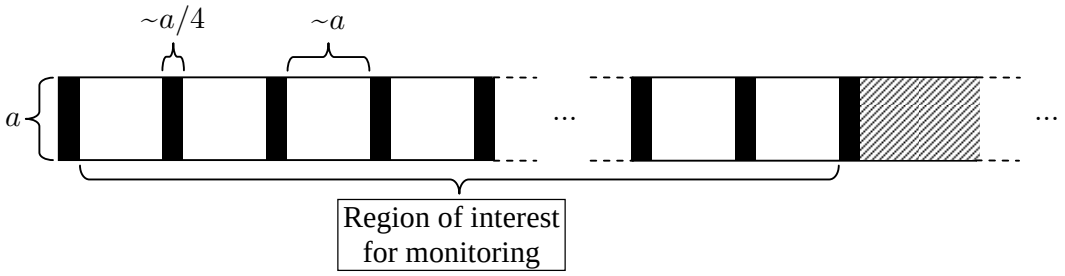
Source: The author

Figure 68 – Illustration of a flexible pipe marked with the suggested pattern of alternating black and white sections.

The dimensions of the pipe markings are illustrated in Figure 69. These dimensions have been determined for creating features as similar to circles as possible; the length of the white markings is about the pipe diameter so that they look like squares when the ROV is laterally observing the pipe (the usual viewpoint when the pipe is intended to

[1] For a sake of convenience, some markings may be made in yellow instead of white. This change does not alter the essence of the feature being marked (i.e., it is yet a bright region surrounded by a dark rim), but reduces its contrast somewhat (white over black is more contrasting than yellow over black).

be monitored); the Gaussian filter of the preprocessing stage (Section 4.1) gives the final touch by rounding the corners of the white markings. The length of the black markings has been set to be about quarter of the length of the white ones, which is long enough for producing gaps when the pipe is about the intended monitoring distance[2].



Source: The author

Figure 69 – The dimensions of the markings for a pipe with a diameter equal to a. The white markings have a length of about a and the black ones have a length of about $a/4$.

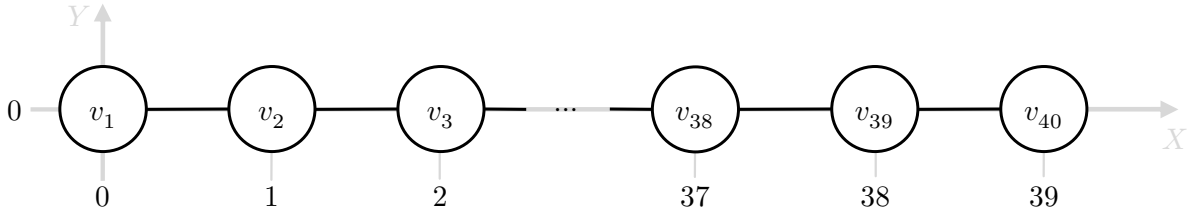
This scenario has two interesting characteristics that would affect the technique execution. Firstly, since a pipe is an elongated body, its markings create a one-dimensional pattern. Nevertheless, this is naturally tackled by the proposed technique, provided that a suitable pattern template is given.

Secondly, the pipes are non-rigid structures, thereby the patterns marked over them are subject to change over time. Basically, pipes are incompressible and unexpansive structures (at least not significantly), thus the size of the markings and the distances between them are preserved. However, pipes are allowed to bend, and consequently the pattern marked over them curves accordingly. The proposed technique is able to tackle these changes just by relaxing the angular constraint parameter of the backtracking in the searching stage. Thus, no special workarounds are required, such as creating multiple pattern templates for covering all bending levels.

The pattern template used for generating the results of this case study is in Figure 70. It has a total of 40 vertices sequentially connected to each other. This superior number of vertices was purposely set for having room for all pipe vertebrae as well as for some additional blobs expected to be included as outliers. Differently from the first case study in which the pattern was expected to be detected in its entirety, in the pipe experiments the pattern detector does not have reliable means for asserting such a thing. Therefore, the detector has been configured with a different modus operandi causing it to

[2] This case study has a noteworthy characteristic — solely marking the pipe is actually not enough for detecting it with the proposed technique. In fact, it becomes detectable only when the dark background of the image come on the scene. One can notice in Figure 69 that while the white markings are laterally bounded by the black ones, their upper and bottom sides have no bounds. The dark background is what creates the missing bounds.

output the best partial mapping found during its searching stage (i.e., the longest chain of blobs which satisfies the constraints imposed by the pattern template).



Source: The author

Figure 70 – The one-dimensional pattern template used for detecting flexible pipes. Despite describing a straight pattern, it is also able to detect bent pipes.

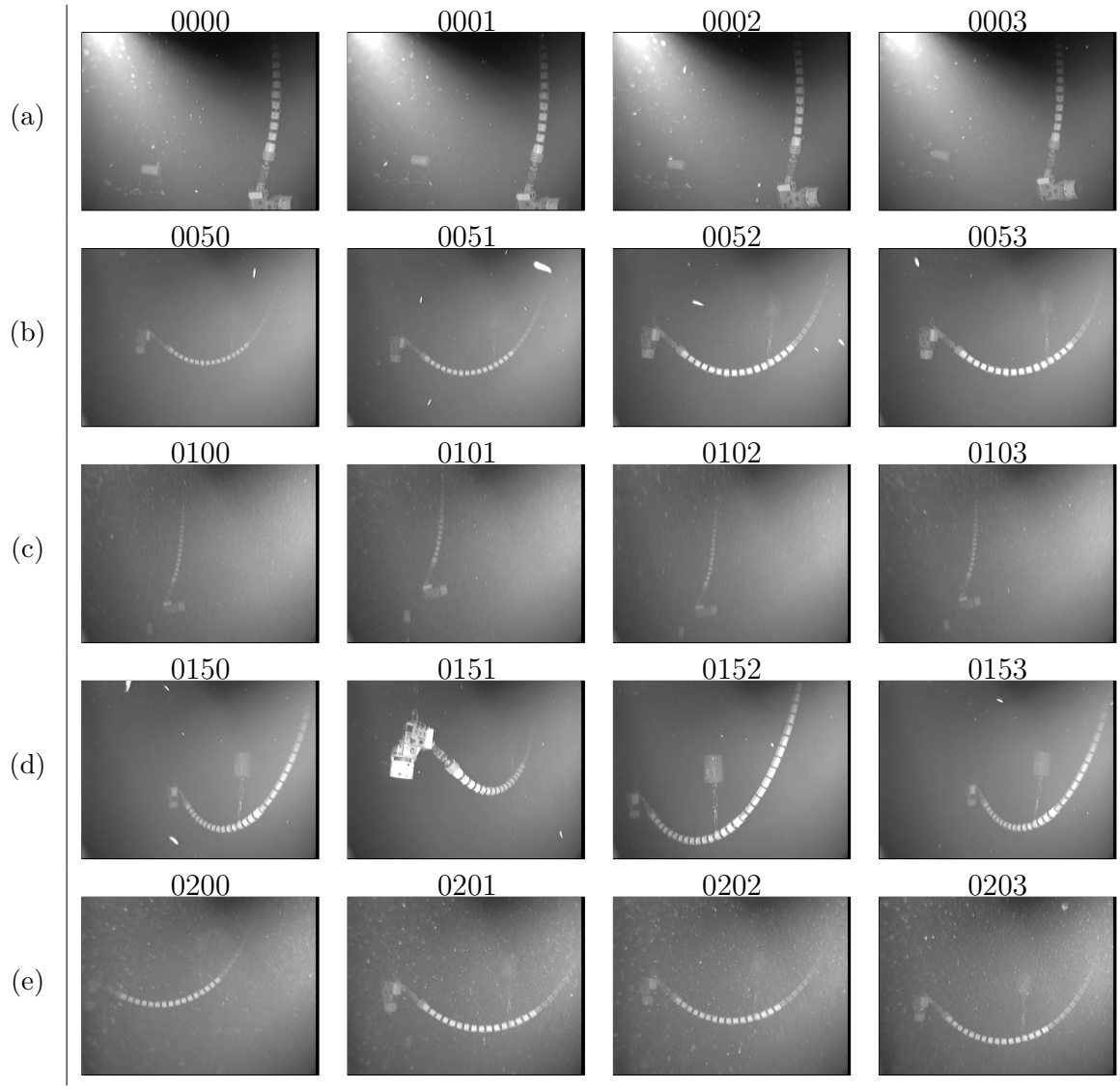
6.3 Image set

The image set used in this case study is termed as *pipe* and contains 20 images captured during a real offshore operation. These images were sampled from five different moments of the operation for a more comprehensive experiment. The camera system used was the low-light underwater Kongsberg OE15-101c, which captures monochrome images (8-bit depth) with 640×480 pixels in size. Additionally, the ROV was equipped with independently adjustable spotlights. The images of the *pipe* set are shown in Figure 71, the ground-truth images of this set are in Figure 82.

The first four images of the *pipe* set (Figure 71a) are characterized by the strong presence of undesirable artifacts (mainly fish and floating particles) and by an uneven illumination of the pipe surface (observe that the pipe markings are less contrasting at the top of the image). In particular, for these images, the proximity of a spotlight to the camera system is what makes the presence of undesirable artifacts so pronounced (notice the intense shaft of light coming from the top left corner of the image and how the particles in its path are brighter than others).

In the next four images (Figure 71b) there are fewer floating particles in comparison with the previous ones (although some fish are yet visible). However, in spite of the overall better lighting condition at the moment these images were captured, the second half of the pipe surface (from left to right) has low contrast. This contrast reduction is caused by the yellowish tone of that pipe region and ultimately reaches the point in which the pipe completely fade out.

The images in Figure 71c are characterized by a low contrast and by a moderate presence of floating particles. However, notice that the reduced contrast in these images is not caused by a cutback in spotlights power, but by the lighting attenuation caused by



Source: Petrobras

Figure 71 – The image set used in the pipe experiments. The images are labeled with four-digit names and each row contains images captured from a specific moment of the operation.

the superior distance that light has to travel in that circumstance (pipe was at a farther distance from the camera by the time images were captured).

The images in Figure 71d explore oblique viewpoints of the pipe. It was noticed that these images have distinctive characteristics that may affect algorithms performance. Firstly, the features (the white markings over the pipe) are deformed into concave shapes possibly creating difficulties for the blob extractors that are based on circles. Secondly, for the images captured from viewpoints in which the second half of the pipe is closer to the camera (images 0150, 0152, and 0153), a more uniform contrast over the whole pipe extension is observed; the explanation for this fact is that the inferior distance from the camera to that part of the pipe make up for the reduced contrast between the colors used

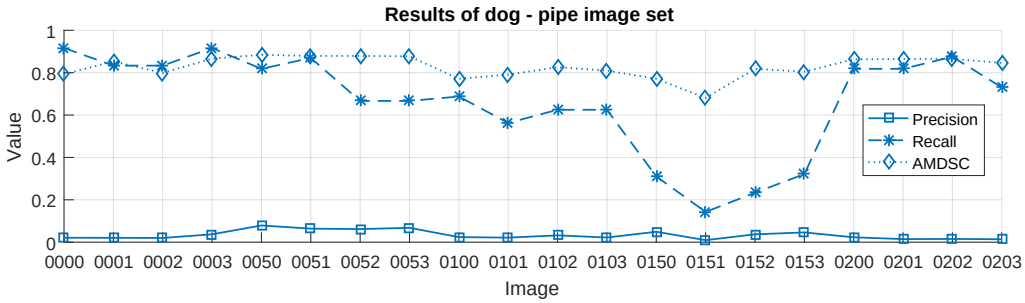
to mark it (yellow and black).

Regarding the camera viewpoint and the lighting condition, the last four images (Figure 71e) are similar to ones shown in Figure 71b. However, the images in Figure 71e present a higher density of floating particles creating a harsher scenario for the algorithms being tested.

6.4 Blob extraction results

The same methodology used for evaluating the blob extractors in the case study of the printed pattern (Section 5.4) is used here. Refer to Table 16 for the per-image results obtained by each extractor in numerical values.

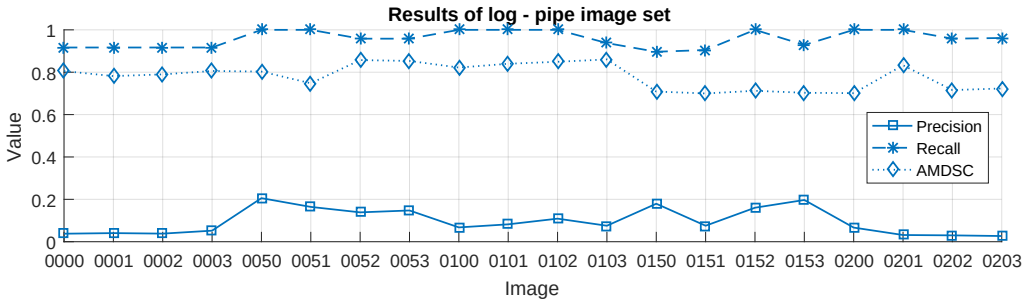
The *dog* extractor achieved ordinary results overall (Figure 72). Firstly, Precision was low indicating that many outliers were extracted (lesser than 0.1 over all images of the set). Secondly, since *dog* can only produce perfect circular blobs, it did not output the best representatives for the features marked over the pipe (although achieving an AMDSC mark superior to 0.8 in average). Finally and foremost, *dog* did not achieve Recall results as high as other extractors, such as *log*, *mser*, *mll*, and *proposed*. In special for the images captured from oblique viewpoints (0150, 0151, 0152, and 0153), *dog* did not extract more than a third part of the features, which suggests that this extractor is not robust to that kind/level of distortion.



Source: The author

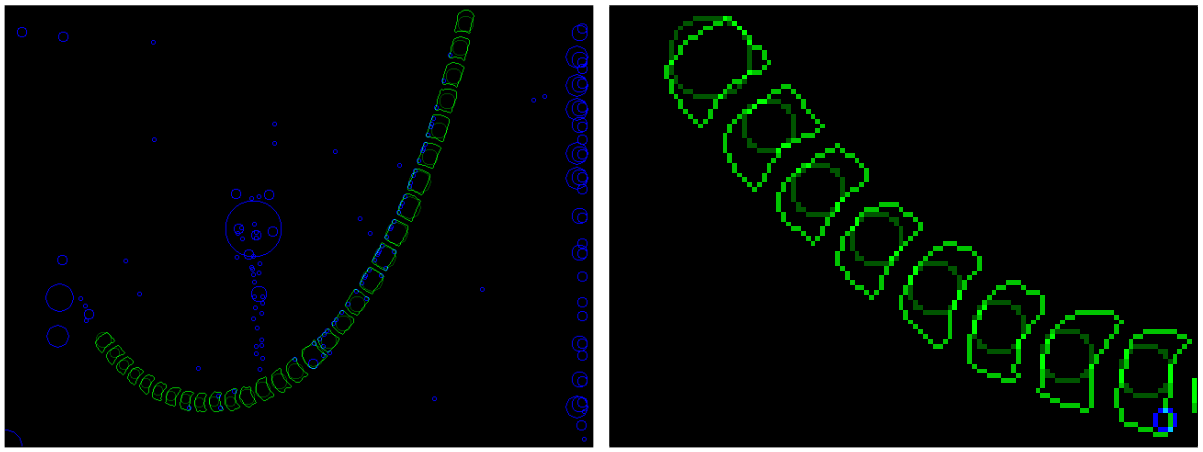
Figure 72 – Results of blob extractor *dog* for image set *pipe*. The result of each metric is presented on a per-image basis. Images are identified by four-digit names drawn from their original frame numbers in the video footage.

The extractor *log* performed noticeably better than *dog* for the metrics Precision and Recall (Figure 73). This is a curious fact since in the first case study the results obtained by *log* were comparable to that obtained by *dog*. More interesting yet is that *log* actually achieved the best Recall mark (in average) among all extractors — *log* managed to maintain high Recall scores even for the images in Figure 71d (the ones that contain the most distorted features). However, the quality of the blobs extracted by *log* was not equally high (see Figure 74) reaching AMDSC values as lower as 0.7 for some images.



Source: The author

Figure 73 – Results of blob extractor *log* for image set *pipe*.



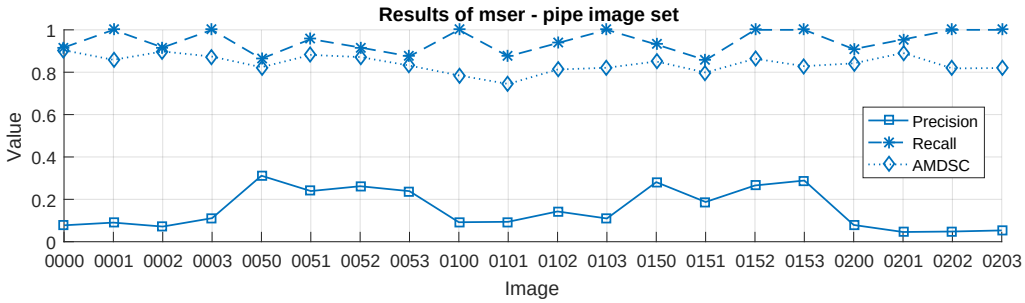
(a) (b)

Source: The author

Figure 74 – The image in (a) shows how the blobs extracted by *log* from input image 0152 were classified by the analyzer. Despite *log* extracted all features marked over the pipe, it is clear from (b) that there is a lack of quality in the inlier blobs; notice that most inlier blobs (contours in dark green) are significantly smaller than their respective ground-truth regions (contours in bright green).

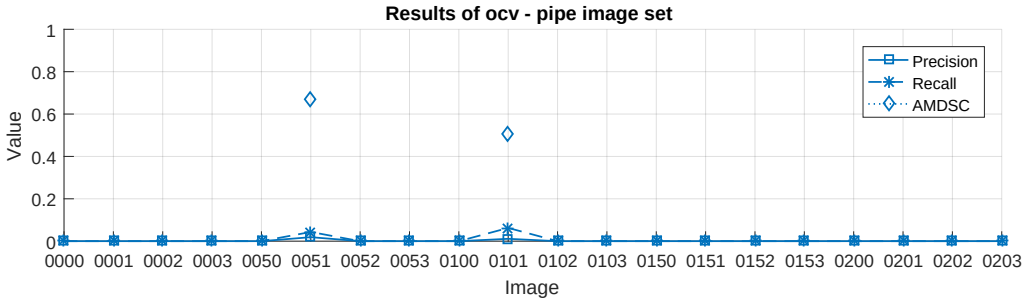
The *mser* extractor performed well regarding the three metrics (Figure 75). Its PRECISION is visibly superior in comparison with *log* results, indicating a cleaner extraction. However, for the images with more undesirable artifacts (Figures 71a, 71c, and 71e) it is possible to notice a drop in PRECISION. The RECALL results achieved by *mser* was comparable with *log* results, although *mser* produced blobs with superior quality rising its AMDSC marks.

The *ocv* extractor retrieved only one feature from the image 0051 and another from the image 0101; this was its worst performance among all image sets (Figure 76). The same problem regarding undersized blobs that was observed while experimenting with the previous image sets was the cause of such a poor performance (i.e., *ocv* extracts too small blobs that do not attain to the minimum DSC score required for producing true positive results).



Source: The author

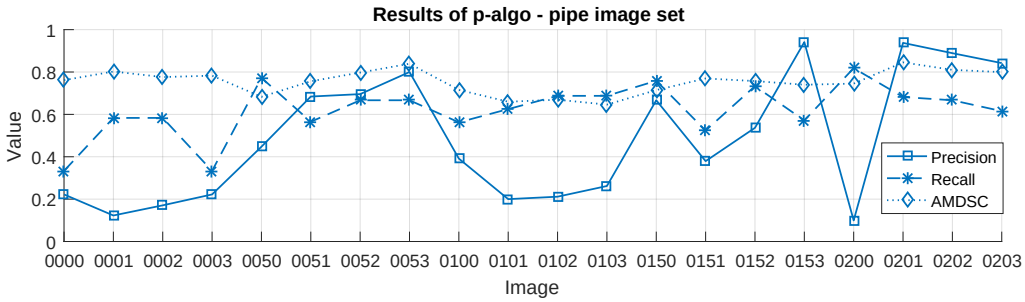
Figure 75 – Results of blob extractor *mser* for image set *pipe*.



Source: The author

Figure 76 – Results of blob extractor *ocv* for image set *pipe*.

As seen in Figure 77, another extractor that performed poorly was *p-algo* (once more). In spite of achieving the best PRECISION mark (in average) from among all extractors, *p-algo* failed in extracting many of the features marked over the pipe (more than a third of the features were missed overall). Not to mention that PRECISION was inconsistent over the images and AMDSC was not satisfactory as well.

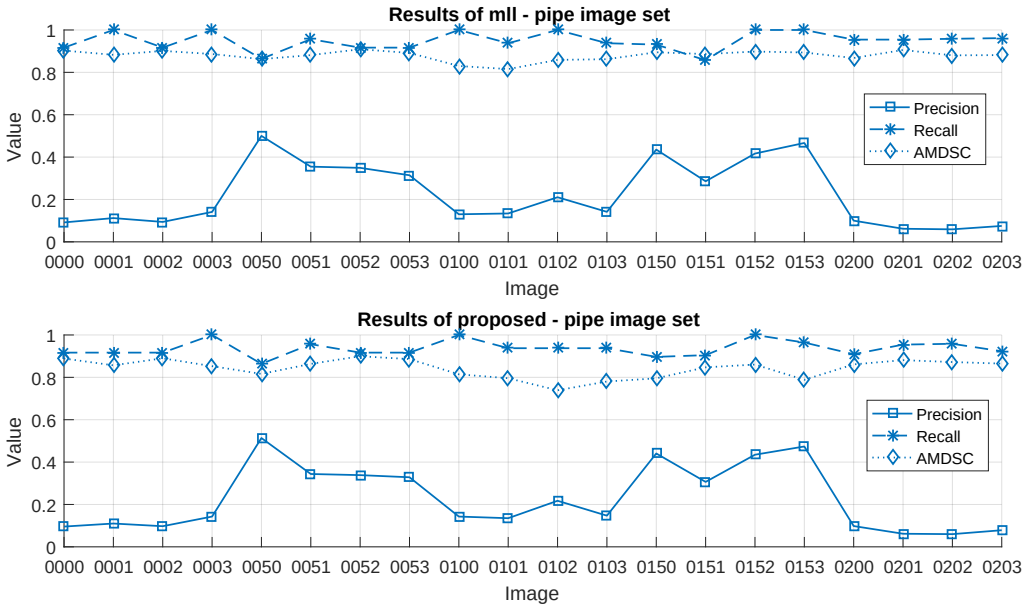


Source: The author

Figure 77 – Results of blob extractor *p-algo* for image set *pipe*.

The extractors *mll* and *proposed* performed virtually identical (Figure 78), with a marginal advantage to the former (in average, *mll* achieved slightly superior results for the metrics RECALL and AMDSC, but not for the PRECISION). It is curious that the natural tendency of *mll* to extract oversized blobs has given it an advantage over *proposed*

in this case study. This fact is attributed to some characteristics of pattern marked over the pipe and how it ultimately looks like from the viewer perspective.



Source: The author

Figure 78 – Results of blob extractors *mll* and *proposed* for image set *pipe*.

Firstly, the pattern created for the case study of the pipe has closer features than the pattern used in the previous case study — for the pipe scenario, the ratio of the features extent to the gaps separating them is $4:1$ (Figure 69), for the printed pattern this ratio is about $2.44:1$ (Figure 25a). This condition favor *mll* because it reduces the amount of space that blobs have to grow beyond the features bounds consequently preventing the production of oversized blobs.

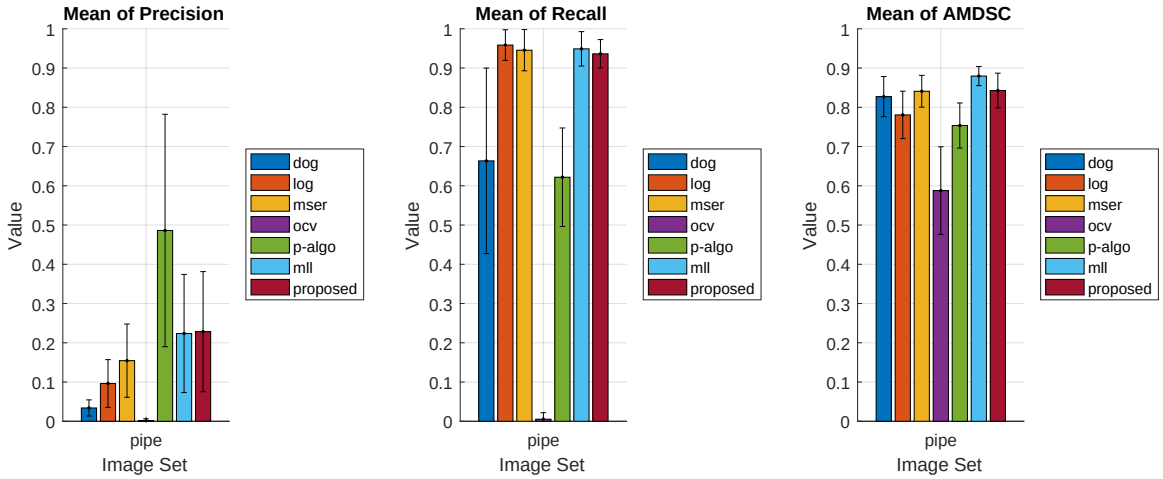
On the other hand, the imperfections of the white markings painted over the pipe, coupled with the diffusion caused by the dense medium in which the pipe is immersed, blurs the features edges lowering the acuteness of the stability measurement used by *proposed* (recall from Figure 16). From the practical standpoint, this circumstance increases the likelihood of extracting undersized blobs, which may either simply reduce the quality of the blobs extracted (lowering AMDSC) or make the analyzer to consider some blobs extracted from features as false positive results (lowering RECALL).

6.4.1 Overall analysis

The blob extractors average results over the whole set of images are presented in Figure 79. Refer to Table 17 to see these results in numerical values.

The *ocv* extractor performed the worst among all the extractors — it extracted only two features over all images of the set. The ultimate conclusion that can be drawn

about *ocv* is that it is not effective at extracting representative regions of interest, but it is only suitable for retrieving the centroid of these regions.



Source: The author

Figure 79 – Mean of the three metrics evaluated for all blob extractors for the pipe experiments.

The extractors *dog* and *p-algo* achieved marginal to poor results (each of them missed at least 33% of the features marked over the pipe, in average). While *dog* struggled with images captured from oblique viewpoints, *p-algo* presented inconsistent results independently of image characteristic.

Much better performances were achieved by *log* and *mser*, which have extracted nothing less than 94% of the features. In particular, *log* achieved a slight better mark regarding Recall. On the other hand, *mser* has had the advantage of extracting better representative blobs (higher AMDSC) and producing fewer false positive results (superior Precision).

As *log* and *mser*, *mll* and *proposed* also achieved valuable results (Recall marks were virtually identical for these four extractors). However, the latter have the additional benefit of producing less false positive results — the average Precision results show that *mll* and *proposed* outperformed *log* in about 133% and *mser* in about 45%. Finally, regarding AMDSC, *mll* achieved a slightly superior performance in comparison with *proposed*.

Hypothesis tests have been performed for the Recall results obtained by *log*, *mser*, *mll*, and *proposed*; the results of *log* (the highest in average) were tested against the results of other extractors to verify whether this superiority is statistically significant. The null hypothesis is that the median of Recall differences of *log* and one of the other extractors is 0; the alternate is that it is greater than 0. The p-values of the tests involving *mser*, *mll*, and *proposed* were respectively 0.1758, 0.2754, and 0.0190; therefore, at the 1%

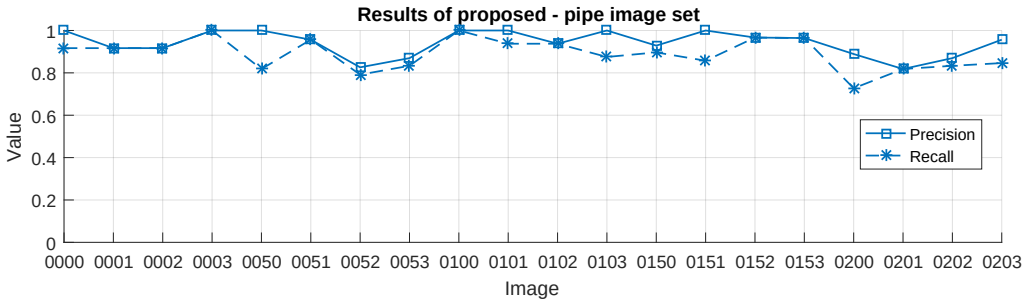
significance level, there is not enough statistical evidence to conclude that *log* was superior to *mser*, *mll*, or *proposed* regarding the RECALL metric.

In respect to the tiny advantage of *proposed* over *mll* regarding the PRECISION metric (in average), it was tested whether this superiority is statistically significant. The null hypothesis is that the median of PRECISION differences of *mll* and *proposed* is 0; the alternate hypothesis is that it is less than 0. The null hypothesis was rejected at the 1% significance level (p-value was 0.0093), which means that there is enough statistical evidence to conclude that the median of PRECISION obtained by *mll* is less than the median of PRECISION obtained by *proposed*.

Finally, it was also tested whether the superior AMDSC result achieved by *mll* (in comparison with *proposed*) is statistically significant. The null hypothesis is that the median of AMDSC differences of *mll* and *proposed* is 0; the alternate hypothesis is that it is greater than 0. Since p-value was 4.78×10^{-5}, the null hypothesis was rejected indicating that there is enough statistical evidence to conclude that *mll* outperformed *proposed* regarding the AMDSC metric.

6.5 Pattern detection results

This section presents the results obtained by the *proposed* pattern detector (as previously stated, *ocv* was excluded from this evaluation because it is not able to detect the kind of pattern marked over the pipe). The per-image results obtained by *proposed* are presented in Figure 80 and in Table 18. The average results are in Table 8.



Source: The author

Figure 80 – Proposed pattern detector results for image set *pipe*. Images are identified by four-digit names drawn from their original frame numbers in the video footage.

The *proposed* detector retrieved most of the features marked over the pipe (89% in average). This result represents a drop of only 4.6% in comparison with the results obtained by the respective blob extractor (recall that since the pattern searching stage of the proposed detection algorithm cannot recover the features missed during the blob

Table 8 – Mean of the proposed pattern detector results for the *pipe* image set. Results are presented as $x \pm y$, where x is the mean value of the metric over all images of the set and y is the standard deviation.

		Image Set
		pipe
proposed	PRECISION	0.941±0.060
	RECALL	0.890±0.074

extraction stage, the RECALL measured at the end of the pattern searching stage is impossible to be greater than the one measured at the end of the blob extraction stage).

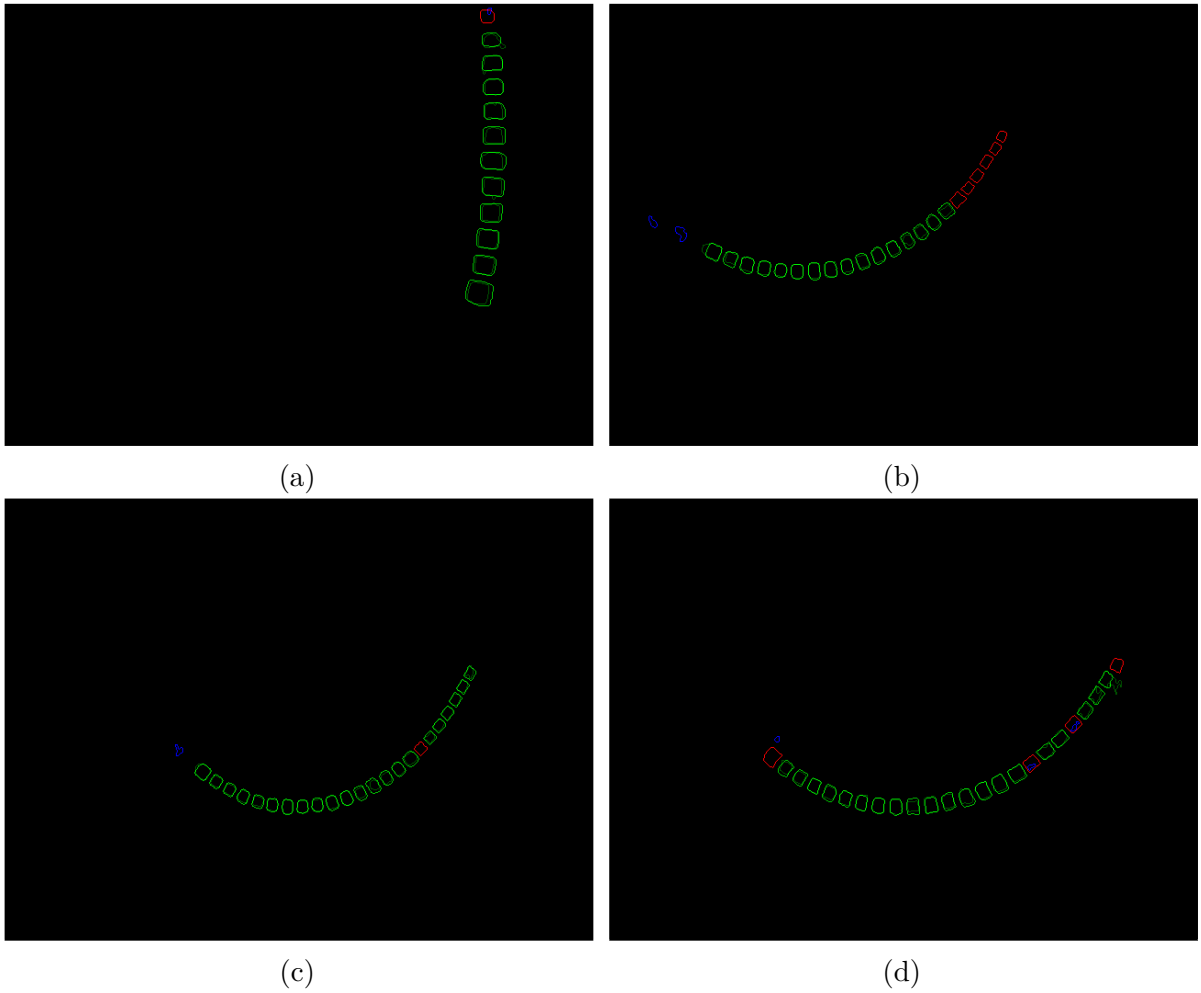
Additionally, just a small fraction of the elements detected by *proposed* were classified as outliers (5.9% in average). This result indicates that the majority of the outliers produced by the blob extractor were discarded during the pattern searching stage (PRECISION results improved from 0.228 to 0.941).

Regarding execution time, *proposed* spent 1.1 seconds (in average) to process each of the pipe images.

In spite of the valuable results achieved by the *proposed* pattern detector, some problems were observed while processing specific images; these are detailed next.

There are basically two circumstances in which the *proposed* pattern detector produces outliers; they are both related to false positive errors produced by the blob extractor. The first circumstance occurs when undersized blobs are extracted from the features marked over the pipe. Notice in Figure 81a that an undersized element at the top of the image (contour in blue) was accounted as outlier because of its inferior size. It is equally important to observe that a false negative result (contour in red) was also produced in detriment of this fact. The second circumstance involves the blobs extracted from the regions of the pipe not marked with the pattern of interest (Figure 81b). In this case, two blobs that were extracted from that regions were detected by *proposed* and accounted as outliers by the analyzer (contours in blue).

Another problem observed with *proposed* is related to the occurrence of false negative results while extracting blobs. The image in Figure 81a shows a case in which the *proposed* detector missed one feature of the pattern because of a failure of the blob extractor at one of the pipe extremities. As might be expected, this kind of failure may also occur at medial regions of the pipe creating discontinuities over the pipe extent, which may implicate in graver consequences for the pattern detector. For example, in Figure 81b only two out of the six false negative results were caused because of extraction failures (the leftmost and the rightmost ones). The others (the intermediate ones) were actually due to the discontinuity created by one of the extractor failures (the leftmost one), which split the pipe into two parts. This problem could be mitigated by relaxing some of the



(a) (b)

(c) (d)

Source: The author

Figure 81 – The debug images output by the pattern detector analyzer for the input images 0001 (a), 0200 (b), 0051 (c), and 0202 (d). The analyzer draws in green the regions confirmed as true positive results (ground-truth regions, in bright green, are matched with elements detected by *proposed*, in dark green), in blue are the false positive results, and in red are the false negative ones.

parameters that control the constraints tested by the pattern searching procedure, but this could have the side effect of producing more false positive results. It is also relevant to mention that in some situations the absence of a single blob (at medial regions of the pipe) does not produce a significant discontinuity that could cause the mentioned problem, such as in Figure 81c.

The last problem observed with *proposed* can be visualized in Figure 81d. Notice that, in that occasion, the pattern searching procedure "preferred" a solution that has an outlier blob as the leftmost element (the region in blue at the left extremity of the pipe). The term "preferred" was used here because there was the option of choosing the correct blob and preventing the false negative result (the region in red) at that pipe extremity. That is, by inspecting the image output by the blob extractor analyzer, it is possible to see

that the blob relative to that false negative result was correctly extracted. This problem occurred because the searching procedure does not have means to judge which of two equal-sized solutions is the best (see line 16 in Algorithm 2). Therefore, both solutions seemed equally good and the algorithm opted for the first found.

6.6 Summary

This chapter presented the second and final part of the assessment of the proposed technique. This time, the case study happened within an offshore scenario with flexible pipes, which has some distinctive characteristics when compared with the scenario presented in the previous case study. Among these characteristics, the most remarkable ones are: the pattern is subject to deform over time, the pattern is one-dimensional, the pattern is meant to be detected partially, and the features that constitute the pattern are not perfectly circular. The image set created for the experiments has images captured at five different moments of a real offshore operation, totalizing 20 images.

Since the features marked over the pipe are not perfectly circular, the blob extractors based on circles clearly had a disadvantage here (in special with the images captured from oblique viewpoints). Another observation made about the extractors was that *mll* achieved slightly superior results in comparison with *proposed*. The superior performance of *mll* was explained by a specificity of the pattern marked over the pipe — the nearness of the features that did not permit neighbor blobs to overgrow.

The only pattern detector tested in the pipe case study was *proposed* because *ocv* is not able to detect one-dimensional patterns. In overall, *proposed* achieved valuable results by detecting most parts of the pipe with few outliers in the diverse situations. However, a noteworthy problem was encountered with its pattern searching procedure — it may opt for outlier blobs even if the correct inlier blob is available for selection.

7 CONCLUSION

This Thesis presented a detection technique for patterns formed by roundish features. The new technique is composed of two sequential stages; while the first is in charge of selecting candidates for features of the pattern (the blob extraction stage), the second finds out the candidates that actually constitute the pattern (the pattern searching stage).

The blob extractor utilized in the first stage was built on concepts introduced by two extractors of the literature. This tailored solution turned out to suit well the problem of interest by producing fewer outliers and extracting blobs from multiple levels of scale.

The pattern searching stage was implemented as a backtracking procedure that relies on the constraints imposed by a graph data structure (the pattern template). As main advantage, it allows the proposed technique to detect patterns with a diversity of shapes, something not observed in other detectors. Therefore, regarding this aspect, the proposed technique is more general and applicable to a wider variety of scenarios when compared with the state-of-the-art ones.

Experimental results collected from two case studies evidence that the new technique is robust to uneven and low-lighting conditions. Particularly, for the last two image sets of the printed pattern case study, in which the lighting conditions were made as harsh as possible (individuals in charge of hand-labeling ground-truth images could barely see the pattern features because of the images low contrast), the proposed technique succeeded over all tested images.

Moreover, it is important to highlight the fact that the new technique is able to detect patterns that have undergone deformations, which was possible through the employment of a short-range constraint analysis strategy while executing the pattern searching stage. The deformation tolerance was verified not only with a non-rigid object (the flexible pipe), but also with distortions caused by camera optics (it included radial and perspective distortions, which are relevant to the purpose of intrinsic camera calibration). Regarding execution time, the reference implementation of the new technique achieved at best 1.08 frames per second, which is not fast enough for real-time requirements but would be satisfactory for most interactive applications.

The main contributions of this Thesis are:

1. The pattern detector algorithm;
2. The new blob formulation that was built upon the ideas of two existing blob extractors;

3. The utilization of a graph data structure for representing patterns; and
4. The comparative evaluation of some techniques from the literature (blob extractors and pattern detectors) in scenarios relevant to the purpose of camera calibration.

7.1 Challenges faced

One challenge was in making the output of the techniques evaluated as blob extractors compatible each other. Since most of the techniques output results in distinct formats, this was a fundamental requirement to enable their comparison. For instance, while blob extractors may output sets of circles (circles are mostly represented by a position and a radius) or sets of contours (contours are usually represented by connected groups of pixels), image segmentation techniques usually output binary images. An additional complication regarding the adaptation of these techniques was the necessity of rewriting code for three different programming languages — C/C++, Python, and MATLAB.

Another difficulty was related to the creation of the ground-truth images. During the hand-labeling process, it was common for some individuals to commit mistakes that could affect the experiments, such as: filling a single connected component with multiple colors; repeating the same color for different connected components; accidentally painting some non-black pixels in the background region; creating RGB images instead of gray-scale ones. Therefore, all the ground-truth images had to be double-checked for ensuring their proper condition.

In addition to the effort of creating the ground-truth images, the time spent by the analyzer for executing the tests also interfered in the experiments. This fact not only impacted the process of fine-tuning the techniques parameters (because repeating the experiments used to consume too much time) but also influenced the decision of reducing the number of images utilized in the second case study.

7.2 Future work

Despite the main objectives of this Thesis have been successfully achieved, there are some advances that have been left for future developments. These include the addressing of the issues observed while experimenting with the technique in the case studies as well as some general improvements.

Regarding the blob extractor, a useful functionality that could be incorporated is the ability to extract holed blobs, which would enable the proposed detector to identify patterns formed by ringed features (such as the one in Figure 3c). In addition to the benefit of supporting a different kind of pattern/feature, another advantage of working with ringed features is that the extractor would produce fewer outliers (there is a natural

tendency in most scenarios to the number of holed elements be much less than the number of non-holed ones).

As noticed while experimenting with the images captured from the pipe scenario, the pattern searching stage may opt for outlier blobs even if the correct inlier blob is available for selection (it turned out that, although these blobs are outliers, they are compatible with the constraints tested by the searching procedure). The ultimate effect of this issue is that two distinct but equal-sized detection results (one containing an outlier blob wherein the other contains an inlier) are indistinguishable regarding correctness. It is worth mentioning that, in spite of being observed only in the pipe case study, this issue could also happen with the printed pattern used in the first case study. An idea that could overcome this problem is to take into account the residual error measured for each constraint while testing candidate blobs; the hypothesis is that the accumulated residual error can be used as metric for ranking equal-sized detections.

Concerning the usability aspect of the new detection technique, a valuable addition would be to have a visual tool for assisting the creation of the pattern templates (i.e., a tool for creating graphs). Whilst it is true that for the both case studies that have been presented the pattern templates were easy to create (they were created procedurally), for patterns with non-geometric shape the same approach would be impracticable.

Finally, there is room for improvements in the execution time of the proposed technique. Regarding the blob extraction stage, it is known that the most time-consuming operation of this stage is the component tree construction; traversing the tree for extracting blobs is fast as each node of the tree needs to be visited at most once. Efficient algorithms for constructing component trees have been proposed and evaluated comparatively (CARLINET; GÉRAUD, 2014). Another rewarding research would be to investigate whether the pattern searching stage can be modeled as a CSP (Constraint Satisfaction Problem). If so, the A* (A-star) search algorithm could be used to solve it since all CSPs can be viewed as graphs (POKORNY; VINCENT, 2013).

REFERENCES

ARORA, H. *Modeling objects using interest points, edges, and regions*. [S.l.]: University of Illinois at Urbana-Champaign, 2007. 24

ART, A. R. T. *Passive - Targets - Markers & Targets - Products - ART Advanced Realtime Tracking*. 2016. <http://www.ar-tracking.com/products/markers-targets/targets/passive/>. Accessed on January 2, 2016. 18

ART, A. R. T. *Systems Overview - Tracking Systems - Products - ART Advanced Realtime Tracking*. 2016. <http://www.ar-tracking.com/products/tracking-systems/systems-overview/>. Accessed on January 2, 2016. 15

AZUMA, R. et al. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, IEEE, v. 21, n. 6, p. 34–47, 2001. 15

BERGAMASCO, F. et al. Rune-tag: A high accuracy fiducial marker with strong occlusion resilience. In: IEEE. *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. [S.l.], 2011. p. 113–120. 32

BERNSEN, J. Dynamic thresholding of grey-level images. In: *International conference on pattern recognition*. [S.l.: s.n.], 1986. p. 1251–1255. 29

BEUCHER, N.; BEUCHER, S. *Mamba Image (Mathematical Morphology Library Image)*. 2016. <http://www.mamba-image.org>. Accessed on November 1, 2016. 58

BEUCHER, S. Towards a unification of waterfalls, standard and p algorithms. 2013. 29

BEUCHER, S.; MARCOTEGUI, B. P algorithm, a dramatic enhancement of the waterfall transformation. 2009. 29

BRADLEY, D.; ROTH, G. Adaptive thresholding using the integral image. *Journal of graphics, gpu, and game tools*, Taylor & Francis, v. 12, n. 2, p. 13–21, 2007. 29

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 30, 58

BROWN, C. *Advances in computer vision*. [S.l.]: Psychology Press, 2014. v. 1. 15

BRUYANT, P. P. et al. A robust visual tracking system for patient motion detection in spect: Hardware solutions. *Nuclear Science, IEEE Transactions on*, IEEE, v. 52, n. 5, p. 1288–1294, 2005. 18

CARLINET, E.; GÉRAUD, T. A comparative review of component tree computation algorithms. *IEEE Transactions on Image Processing*, IEEE, v. 23, n. 9, p. 3885–3895, 2014. 115

CARON, G.; EYNARD, D. Multiple camera types simultaneous stereo calibration. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. [S.l.: s.n.], 2011. p. 2933–2938. ISSN 1050-4729. 18

CATTIN, D. P. *Lecture notes in Image Restoration: Introduction to Signal and Image Processing*. [S.l.]: MIAC, University of Basel, 2015. <https://miac.unibas.ch/SIP/pdf/SIP-06-Restoration.pdf>. Accessed on November 20, 2015. 54

EARNSHAW, R. *Virtual Reality Systems*. Elsevier Science, 2014. ISBN 9781483296579. Disponível em: <https://books.google.com.br/books?id=gEOjBQAAQBAJ>. 15

ELSEVIER. *Proceedings of the Computer Vision and Image Understanding*. 2017. <http://www.sciencedirect.com/science/journal/10773142>. Accessed on January 23, 2017. 30

ELSEVIER. *Proceedings of the Image and Vision Computing*. 2017. <http://www.sciencedirect.com/science/journal/02628856>. Accessed on January 23, 2017. 30

ESCALERA, A. De la; ARMINGOL, J. M. Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration. *Sensors*, Molecular Diversity Preservation International, v. 10, n. 3, p. 2027–2044, 2010. 18

FISHER, R. et al. *Spatial Filters - Gaussian Smoothing*. 2003. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>. Accessed on November 20, 2015. 54

GAUGLITZ, S.; HÖLLERER, T.; TURK, M. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, Springer, v. 94, n. 3, p. 335–360, 2011. 25

GONZALEZ, R. C.; WOODS, R. E. Digital image fundamentals. In: *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. cap. 2. ISBN 013168728X. 19

GONZALEZ, R. C.; WOODS, R. E. Digital image fundamentals. In: *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. cap. 10. ISBN 013168728X. 29

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Fundamentals*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X. 54

GRVM. *R3D*. 2016. <https://www.gprt.ufpe.br/grvm/?page_id=1307>. Accessed on January 4, 2016. 17

HABACHER, M.; HARKER, M.; O'LEARY, P. Self-calibrating optical 3d pose measurement device for offshore sites. In: IEEE. *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*. [S.l.], 2014. p. 1364–1367. 32

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: CITESEER. *Alvey vision conference*. [S.l.], 1988. v. 15, p. 50. 24

HART, P. E. How the hough transform was invented [dsp history]. *IEEE Signal Processing Magazine*, IEEE, v. 26, n. 6, p. 18–22, 2009. 25

HIGUCHI, M.; DATTA, A.; KANADE, T. *Robotics Institute: Software Package for Precise Camera Calibration*. [S.l.]: The Robotics Institute at Carnegie Mellon University, 2016. <http://www.ri.cmu.edu/research_project_detail.html?project_id=617&menu_id=261>. Accessed on January 2, 2016. 19

IEEE. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017. <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000147>. Accessed on January 23, 2017. 30

IEEE. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017. <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000149>. Accessed on January 23, 2017. 30

IEEE. *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 2017. <http://ieeexplore.ieee.org/xpl/conhome.jsp?punumber=1000545>. Accessed on January 23, 2017. 30

JONES, R. Component trees for image filtering and segmentation. In: *Proceedings of the 1997 IEEE Workshop on Nonlinear Signal and Image Processing, Mackinac Island*. [S.l.: s.n.], 1997. 37

KANG, D.-J.; HA, J.-E.; JEONG, M.-H. Detection of calibration patterns for camera calibration with irregular lighting and complicated backgrounds. *International Journal of Control, Automation, and Systems*, v. 6, n. 5, p. 746–754, 2008. 31, 65

KANG, D.-J.; LEE, W.-H. Automatic circle pattern extraction and camera calibration using fast adaptive binarization and plane homography. *International Journal of Precision Engineering and Manufacturing*, Springer, v. 11, n. 1, p. 13–21, 2010. 31

KANGAS, V. *Comparison of Local Feature Detectors and Descriptors for Visual Object Categorization*. Dissertação (Mestrado) — Lappeenranta Univ. of Technology, 2011. 16, 24

KUMAR, M.; SAXENA, R. et al. Algorithm and technique on various edge detection: A survey. *Signal & Image Processing*, Academy & Industry Research Collaboration Center (AIRCC), v. 4, n. 3, p. 65, 2013. 25

LI, F.-F. *How we're teaching computers to understand pictures*. 2015. <https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures?language=en>. Accessed on December 31, 2015. 15

LI, H. Flexible pipe stress and fatigue analysis. Institutt for marin teknikk, 2012. 99

LINDEBERG, T. *Discrete scale-space theory and the scale-space primal sketch*. Tese (Doutorado) — Royal Institute of Technology, 1991. 27

LINDEBERG, T. Scale-space behaviour of local extrema and blobs. *Journal of Mathematical Imaging and Vision*, Springer, v. 1, n. 1, p. 65–99, 1992. 28

LINDEBERG, T. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, Springer, v. 11, n. 3, p. 283–318, 1993. 28

LINDEBERG, T. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, p. 224–270, 1994. 26

LINDEBERG, T. Feature detection with automatic scale selection. *International journal of computer vision*, Springer, v. 30, n. 2, p. 79–116, 1998. 26

LINDEBERG, T. Image matching using generalized scale-space interest points. *Journal of Mathematical Imaging and Vision*, Springer, v. 52, n. 1, p. 3–36, 2015. 26

LINDEBERG, T.; EKLUNDH, J.-O. Scale detection and region extraction from a scale-space primal sketch. In: IEEE. *Computer Vision, 1990. Proceedings, Third International Conference On*. [S.l.], 1990. p. 416–426. 27, 28, 32, 40

LINDEBERG, T.; EKLUNDH, J.-O. On the computation of a scale-space primal sketch. *Journal of Visual Communication and Image Representation*, Elsevier, v. 2, n. 1, p. 55–78, 1991. 27

LINDEBERG, T.; EKLUNDH, J.-O. Scale-space primal sketch: Construction and experiments. *Image and Vision Computing*, Elsevier, v. 10, n. 1, p. 3–18, 1992. 28

LINDEBERG, T.; LIDBERG, P.; ROLAND, P. Analysis of brain activation patterns using a 3-d scale-space primal sketch. *Human Brain Mapping*, v. 7, n. 3, p. 166–94, 1999. 28

MATAS, J. et al. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, Elsevier, v. 22, n. 10, p. 761–767, 2004. 28, 32, 40

MEHLING, M. *Implementation of a Low Cost Marker Based Infrared Optical Tracking System*. Tese (Doutorado) — Master's thesis, Hochschule der Medien, Fachhochschule Stuttgart, 2006. 18

MOKHTARIAN, F.; MOHANNA, F. Performance evaluation of corner detectors using consistency and accuracy measures. *Computer Vision and Image Understanding*, Elsevier, v. 102, n. 1, p. 81–94, 2006. 25

MORAVEC, H. P. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. [S.l.], 1980. 24

MUKHOPADHYAY, P.; CHAUDHURI, B. B. A survey of hough transform. *Pattern Recognition*, Elsevier, v. 48, n. 3, p. 993–1010, 2015. 29

MULDER, J. D.; JANSEN, J.; RHIJN, A. van. An affordable optical head tracking system for desktop vr/ar systems. In: *Proceedings of the Workshop on Virtual Environments 2003*. New York, NY, USA: ACM, 2003. (EGVE '03), p. 215–223. ISBN 1-58113-686-2. Disponível em: <http://doi.acm.org/10.1145/769953.769978>. 18

NATURALPOINT. *TrackIR5 - In Depth | TrackIR*. 2016. <http://www.naturalpoint.com/trackir/trackir5/>. Accessed on January 2, 2016. 15

OPENCV. *OpenCV: Pose Estimation*. 2016. <http://docs.opencv.org/master/d7/d53/tutorial_py_pose.html#gsc.tab=0>. Accessed on January 2, 2016. 17, 18

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, v. 11, n. 285-296, p. 23–27, 1975. 29, 31

PATEL, T. P.; PANCHAL, S. R. Corner detection techniques: An introductory survey. Citeseer, 2014. 25

PESSOA, S. et al. A segmentation technique for flexible pipes in deep underwater environments. In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2015. p. 135.1–135.12. ISBN 1-901725-53-7. Disponível em: <https://dx.doi.org/10.5244/C.29.135>. 18, 22, 29

PINTO, F. et al. Bratrack: A low-cost marker-based optical stereo tracking system. In: *ACM SIGGRAPH 2008 Posters*. New York, NY, USA: ACM, 2008. (SIGGRAPH '08), p. 131:1–131:1. ISBN 978-1-60558-466-9. Disponível em: <http://doi.acm.org/10.1145/1400885.1401026>. 15

POKORNY, K. L.; VINCENT, R. E. Multiple constraint satisfaction problems using the a-star (a*) search algorithm: Classroom scheduling with preferences. *J. Comput. Sci. Coll.*, Consortium for Computing Sciences in Colleges, USA, v. 28, n. 5, p. 152–159, maio 2013. ISSN 1937-4771. Disponível em: <http://dl.acm.org/citation.cfm?id=2458569.2458602>. 115

PS-TECH. *Optical Trackers | PS-Tech*. 2016. <http://www.ps-tech.com/optical-trackers>. Accessed on January 2, 2016. 15

PS-TECH. *Optical Tracking Markers | PS-tech*. 2016. <http://www.ps-tech.com/optical-trackers/accessories/optical-tracking-markers>. Accessed on January 2, 2016. 18

SANTOS, I. et al. Real time radius of curvature measurement during dvc operations based on flexible pipe 3d reconstruction. In: OFFSHORE TECHNOLOGY CONFERENCE. *OTC Brasil*. [S.l.], 2015. 22, 99

SAUVOLA, J.; PIETIKÄINEN, M. Adaptive document image binarization. *Pattern recognition*, Elsevier, v. 33, n. 2, p. 225–236, 2000. 29, 31

SCHENK, T. Introduction to photogrammetry. *The Ohio State University, Columbus*, 2005. 18

SCHMID, C.; MOHR, R.; BAUCKHAGE, C. Evaluation of interest point detectors. *International Journal of computer vision*, Springer, v. 37, n. 2, p. 151–172, 2000. 25

SMITH, L. N.; SMITH, M. L. Automatic machine vision calibration using statistical and neural network methods. *Image and Vision computing*, Elsevier, v. 23, n. 10, p. 887–899, 2005. 30

SUBBAN, R.; PRABAKARAN. Corner detection methods. *Middle-East Journal of Scientific Research*, IDOSI Publications, v. 23, n. 10, p. 2521–2532, 2015. 25

TRACKHAT. *TrackHat head tracking*. 2016. <http://www.trackhat.org/>. Accessed on January 2, 2016. 15

TUYTELAARS, T.; MIKOLAJCZYK, K. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, Now Publishers Inc., v. 3, n. 3, p. 177–280, 2008. 25

VEDALDI, A.; FULKERSON, B. *VLFeat: An Open and Portable Library of Computer Vision Algorithms*. 2008. <http://www.vlfeat.org/>. Accessed on November 1, 2016. 58

VERMA, M. R.; ALI, D. J. A comparative study of various types of image noise and efficient noise removal techniques. *International journal of advanced research in computer science and software engineering*, v. 3, n. 10, p. 617–622, 2013. 54

WALT, S. van der et al. scikit-image: image processing in Python. *PeerJ*, v. 2, p. e453, 6 2014. ISSN 2167-8359. Accessed on November 1, 2016. Disponível em: <http://dx.doi.org/10.7717/peerj.453>. 57

WANG, J.; KOBAYASHI, E.; SAKUMA, I. Coarse-to-fine dot array marker detection with accurate edge localization for stereo visual tracking. *Biomedical Signal Processing and Control*, Elsevier, v. 15, p. 49–59, 2015. 18, 31

WHITCOMB, L. L. Underwater robotics: Out of the research laboratory and into the field. In: IEEE. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. [S.l.], 2000. v. 1, p. 709–716. 99

YAN, M. Restoration of images corrupted by impulse noise and mixed gaussian impulse noise using blind inpainting. *SIAM Journal on Imaging Sciences*, SIAM, v. 6, n. 3, p. 1227–1245, 2013. 54

YU, X. et al. A rapid and automatic method for camera calibration based on lcd circle pattern. In: IEEE. *Image Analysis and Signal Processing (IASP), 2011 International Conference on*. [S.l.], 2011. p. 383–388. 31

ZHANG, L. *Camera calibration*. [S.l.]: Aalborg University. Department of Communication Technology, 2001. 18

ZHU, H. et al. Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, Elsevier, v. 34, p. 12–27, 2016. 29

ZOU, K. H. et al. Statistical validation of image segmentation quality based on a spatial overlap index 1: Scientific reports. *Academic radiology*, Elsevier, v. 11, n. 2, p. 178–189, 2004. 60

APPENDIX

APPENDIX A – TABLES

This appendix presents supplementary tables.

Table 9 – Blob extractors results for image set *distance* in numerical values.

		Image																			
		0000	0030	0060	0105	0120	0150	0180	0210	0240	0270	0300	0330	0360	0390	0420	0450	0480	0520	0550	0580
dog	Precision	0.26	0.22	0.19	0.16	0.15	0.13	0.12	0.13	0.13	0.11	0.11	0.10	0.08	0.09	0.09	0.10	0.10	0.09	0.09	0.09
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.96	0.86	0.88	0.89	0.88	0.90	0.93	0.93	0.85	0.96	0.88	0.92	0.94	0.87	0.89	0.89	0.84	0.87	0.87	0.83
log	Precision	0.05	0.05	0.05	0.04	0.04	0.05	0.04	0.05	0.04	0.04	0.04	0.03	0.03	0.03	0.03	0.04	0.03	0.03	0.03	0.03
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.95	0.94	0.86	0.85	0.92	0.83	0.96	0.96	0.95	0.77	0.96	0.93	0.95	0.89	0.80	0.66	0.91	0.61	0.59	0.56
mser	Precision	0.25	0.24	0.24	0.21	0.19	0.21	0.21	0.22	0.22	0.19	0.18	0.16	0.16	0.16	0.16	0.19	0.17	0.16	0.18	0.18
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.91	0.92	0.92	0.96	0.78	0.78	0.81	0.86	0.94	0.88	0.71	0.81	0.77	0.79	0.74	0.73	0.92	0.73	0.78	0.86
ocv	Precision	0.00	0.00	0.00	0.00	0.24	0.29	0.03	0.00	0.00	0.00	0.08	0.00	0.01	0.23	0.24	0.07	0.00	0.01	0.00	0.00
	Recall	0.00	0.00	0.00	0.00	0.91	1.00	0.09	0.00	0.00	0.00	0.33	0.02	0.04	0.94	0.98	0.22	0.00	0.02	0.00	0.00
	AMDSC	-	-	-	-	0.55	0.54	0.51	-	-	-	0.64	0.60	0.61	0.53	0.61	0.73	-	0.75	-	-
p-algo	Precision	0.43	0.55	0.42	0.41	0.38	0.43	0.51	0.54	0.47	0.53	0.56	0.29	0.38	0.46	0.29	0.77	0.44	0.42	0.41	0.44
	Recall	0.76	0.78	0.72	0.76	0.74	0.57	0.67	0.69	0.57	0.85	0.70	0.37	0.72	0.72	0.56	0.74	0.78	0.94	1.00	1.00
	AMDSC	0.98	0.97	0.96	0.92	0.91	0.92	0.97	0.96	0.88	0.91	0.95	0.91	0.94	0.89	0.81	0.94	0.69	0.88	0.82	0.74
mll	Precision	0.78	0.72	0.59	0.52	0.49	0.54	0.58	0.61	0.65	0.47	0.38	0.36	0.38	0.35	0.39	0.45	0.47	0.44	0.47	0.43
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.91	0.92	0.91	0.95	0.78	0.78	0.81	0.85	0.93	0.88	0.71	0.81	0.77	0.78	0.73	0.64	0.91	0.71	0.74	0.82
proposed	Precision	0.71	0.65	0.57	0.51	0.49	0.54	0.56	0.63	0.69	0.49	0.41	0.39	0.39	0.39	0.42	0.48	0.49	0.47	0.53	0.47
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.93	0.93	0.93	0.96	0.81	0.80	0.83	0.86	0.94	0.88	0.71	0.82	0.78	0.79	0.74	0.66	0.94	0.73	0.78	0.86

Table 10 – Blob extractors results for image set *rotation* in numerical values.

		Image																			
		0000	0040	0080	0120	0160	0200	0240	0280	0320	0361	0400	0440	0480	0520	0560	0600	0640	0681	0717	0761
dog	Precision	0.14	0.13	0.11	0.12	0.11	0.12	0.08	0.10	0.10	0.09	0.08	0.10	0.11	0.12	0.11	0.08	0.08	0.10	0.09	0.11
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.88	0.89	0.88	0.91	0.95	0.85	0.88	0.92	0.95	0.96	0.84	0.91	0.96	0.97	0.96	0.97	0.72	0.83	0.86	0.88
log	Precision	0.05	0.04	0.03	0.04	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.02	0.02	0.04	0.03	0.04
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.89	0.91	0.95	0.93	0.87	0.96	0.77	0.80	0.78	0.75	0.88	0.81	0.76	0.75	0.74	0.72	0.96	0.69	0.85	0.74
mser	Precision	0.23	0.24	0.22	0.19	0.19	0.21	0.16	0.20	0.17	0.17	0.16	0.17	0.19	0.20	0.19	0.16	0.16	0.17	0.17	0.19
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.77	0.82	0.86	0.86	0.83	0.96	0.72	0.79	0.83	0.85	0.72	0.80	0.85	0.87	0.87	0.92	0.66	0.97	0.81	0.92
ocv	Precision	0.26	0.05	0.08	0.00	0.00	0.00	0.15	0.05	0.00	0.00	0.21	0.06	0.00	0.00	0.00	0.00	0.23	0.00	0.22	0.00
	Recall	1.00	0.19	0.30	0.00	0.00	0.00	0.69	0.19	0.00	0.00	0.94	0.26	0.00	0.00	0.00	0.00	0.98	0.00	0.89	0.00
	AMDSC	0.59	0.54	0.52	-	-	-	0.54	0.51	-	-	0.56	0.51	-	-	-	-	0.65	-	0.53	-
p-algo	Precision	0.46	0.46	0.62	0.72	0.58	0.36	0.59	0.48	0.64	0.42	0.52	0.46	0.49	0.60	0.49	0.66	0.55	0.43	0.51	0.53
	Recall	0.44	0.35	0.70	0.89	0.74	0.44	0.72	0.50	0.65	0.50	0.48	0.31	0.46	0.56	0.31	0.78	0.69	0.78	0.83	0.35
	AMDSC	0.94	0.97	0.94	0.93	0.96	0.81	0.92	0.96	0.94	0.87	0.92	0.96	0.93	0.90	0.88	0.84	0.89	0.78	0.93	0.83
mll	Precision	0.56	0.63	0.49	0.46	0.45	0.52	0.43	0.42	0.35	0.38	0.37	0.39	0.40	0.38	0.39	0.35	0.35	0.36	0.36	0.38
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.76	0.80	0.86	0.86	0.83	0.95	0.72	0.78	0.83	0.86	0.72	0.80	0.85	0.87	0.89	0.92	0.65	0.97	0.80	0.93
proposed	Precision	0.60	0.65	0.53	0.51	0.48	0.55	0.45	0.46	0.38	0.40	0.40	0.42	0.43	0.39	0.43	0.37	0.38	0.40	0.38	0.40
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.77	0.83	0.87	0.87	0.84	0.97	0.72	0.81	0.84	0.89	0.73	0.80	0.86	0.88	0.89	0.93	0.66	0.98	0.82	0.94

Table 11 – Blob extractors results for image set *obliquity* in numerical values.

		Image																			
		0070	0088	0105	0122	0140	0163	0175	0210	0246	0262	0280	0315	0350	0385	0420	0456	0490	0525	0560	0595
dog	Precision	0.24	0.25	0.25	0.22	0.21	0.19	0.14	0.13	0.15	0.15	0.19	0.17	0.18	0.19	0.16	0.17	0.16	0.15	0.14	0.11
	Recall	0.93	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
	AMDSC	0.77	0.77	0.82	0.85	0.87	0.88	0.90	0.90	0.90	0.89	0.88	0.90	0.91	0.91	0.90	0.87	0.86	0.85	0.81	0.78
log	Precision	0.04	0.03	0.04	0.04	0.04	0.04	0.03	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.05	0.05	0.05	0.04	0.04	0.04
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98
	AMDSC	0.70	0.76	0.82	0.84	0.86	0.83	0.88	0.88	0.88	0.87	0.84	0.89	0.89	0.89	0.89	0.84	0.83	0.84	0.76	0.71
mser	Precision	0.33	0.33	0.30	0.33	0.35	0.34	0.28	0.28	0.30	0.29	0.30	0.29	0.26	0.28	0.27	0.24	0.25	0.26	0.26	0.26
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.95	0.74	0.83	0.88	0.87	0.96	0.87	0.84	0.89	0.90	0.94	0.81	0.84	0.89	0.85	0.96	0.95	0.86	0.91	0.93
ocv	Precision	0.01	0.33	0.14	0.05	0.06	0.00	0.05	0.07	0.00	0.00	0.00	0.18	0.05	0.00	0.05	0.00	0.00	0.03	0.00	0.01
	Recall	0.02	0.78	0.39	0.15	0.17	0.00	0.13	0.19	0.00	0.00	0.00	0.48	0.13	0.00	0.13	0.00	0.00	0.07	0.00	0.02
	AMDSC	0.51	0.57	0.53	0.52	0.52	-	0.51	0.52	-	-	-	0.53	0.52	-	0.51	-	-	0.51	-	0.50
p-algo	Precision	0.96	0.96	0.96	1.00	0.98	0.96	0.92	0.80	0.89	0.92	0.95	1.00	0.89	0.84	0.94	0.90	0.77	0.86	0.87	0.81
	Recall	0.93	0.87	0.87	0.91	0.91	0.83	0.67	0.67	0.57	0.67	0.76	0.74	0.61	0.59	0.54	0.70	0.69	0.67	0.48	0.70
	AMDSC	0.84	0.92	0.97	0.94	0.95	0.87	0.96	0.97	0.95	0.95	0.89	0.95	0.98	0.96	0.97	0.87	0.87	0.95	0.90	0.85
mll	Precision	0.72	0.76	0.78	0.81	0.81	0.78	0.64	0.69	0.67	0.70	0.70	0.65	0.64	0.72	0.68	0.64	0.73	0.74	0.70	0.67
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.95	0.74	0.83	0.87	0.87	0.96	0.86	0.84	0.89	0.90	0.93	0.80	0.84	0.89	0.85	0.96	0.94	0.86	0.90	0.93
proposed	Precision	0.73	0.76	0.76	0.83	0.79	0.79	0.66	0.71	0.68	0.73	0.73	0.67	0.63	0.72	0.62	0.67	0.73	0.77	0.74	0.73
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.96	0.75	0.84	0.88	0.88	0.96	0.88	0.85	0.90	0.91	0.96	0.82	0.86	0.90	0.87	0.96	0.95	0.87	0.91	0.95

Table 12 – Blob extractors results for image set *radial-distortion* in numerical values.

		Image																			
		0019	0024	0031	0047	0067	0078	0130	0136	0142	0150	0157	0173	0189	0205	0221	0236	0252	0268	0284	0300
dog	Precision	0.37	0.38	0.25	0.25	0.23	0.25	0.21	0.21	0.19	0.17	0.15	0.13	0.13	0.11	0.11	0.11	0.10	0.10	0.09	0.09
	Recall	0.98	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.81	0.83	0.83	0.84	0.83	0.85	0.86	0.85	0.85	0.87	0.87	0.89	0.89	0.91	0.86	0.92	0.93	0.92	0.89	0.88
log	Precision	0.07	0.08	0.04	0.05	0.05	0.05	0.05	0.05	0.05	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.03	0.03	0.03
	Recall	0.96	0.96	0.98	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.82	0.82	0.81	0.82	0.84	0.84	0.85	0.84	0.86	0.87	0.88	0.88	0.88	0.90	0.87	0.87	0.89	0.92	0.93	0.95
mser	Precision	0.27	0.28	0.26	0.25	0.25	0.27	0.27	0.25	0.24	0.24	0.24	0.23	0.21	0.19	0.21	0.21	0.20	0.18	0.18	0.18
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.83	0.94	0.94	0.91	0.82	0.92	0.92	0.95	0.88	0.92	0.82	0.91	0.91	0.91	0.73	0.87	0.88	0.92	0.84	0.90
ocv	Precision	0.11	0.00	0.01	0.03	0.18	0.01	0.00	0.00	0.08	0.00	0.15	0.00	0.00	0.00	0.24	0.02	0.02	0.00	0.10	0.01
	Recall	0.30	0.00	0.02	0.07	0.44	0.04	0.00	0.00	0.20	0.00	0.41	0.00	0.00	0.00	0.85	0.07	0.07	0.00	0.41	0.04
	AMDSC	0.56	-	0.50	0.52	0.54	0.52	-	-	0.53	-	0.52	-	-	-	0.57	0.52	0.51	-	0.53	0.50
p-algo	Precision	0.82	0.86	0.85	0.84	0.79	0.77	0.67	0.70	0.78	0.81	0.74	0.70	0.49	0.53	0.53	0.46	0.39	0.36	0.51	0.39
	Recall	0.67	0.46	0.72	0.57	0.61	0.67	0.59	0.69	0.72	0.70	0.80	0.59	0.63	0.57	0.61	0.70	0.72	0.59	0.72	0.59
	AMDSC	0.95	0.93	0.94	0.97	0.94	0.97	0.96	0.93	0.94	0.96	0.94	0.96	0.95	0.95	0.88	0.96	0.95	0.90	0.96	0.91
mll	Precision	0.75	0.71	0.77	0.78	0.74	0.81	0.75	0.75	0.69	0.66	0.61	0.60	0.58	0.57	0.55	0.57	0.53	0.50	0.50	0.51
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.82	0.93	0.94	0.91	0.82	0.91	0.92	0.95	0.88	0.92	0.82	0.91	0.91	0.91	0.73	0.87	0.88	0.92	0.84	0.90
proposed	Precision	0.78	0.69	0.78	0.77	0.73	0.78	0.74	0.74	0.70	0.64	0.60	0.59	0.59	0.58	0.54	0.56	0.56	0.55	0.53	0.57
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.84	0.96	0.95	0.92	0.84	0.93	0.93	0.96	0.89	0.93	0.85	0.92	0.92	0.92	0.75	0.88	0.89	0.93	0.85	0.91

Table 13 – Blob extractors results for image set *uneven-contrast* in numerical values.

		Image																			
		0125	0168	0211	0255	0298	0342	0385	0428	0477	0515	0559	0602	0646	0689	0732	0750	0855	0879	0920	0950
dog	Precision	0.73	0.70	0.67	0.71	0.68	0.60	0.61	0.65	0.70	0.76	0.70	0.67	0.68	0.60	0.54	0.54	0.59	0.54	0.64	0.62
	Recall	0.96	1.00	0.94	0.94	1.00	0.98	1.00	0.98	1.00	0.93	0.94	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.92	0.90	0.88	0.91	0.89	0.92	0.90	0.89	0.90	0.90	0.92	0.91	0.92	0.94	0.95	0.93	0.93	0.93	0.92	0.94
log	Precision	0.49	0.49	0.50	0.57	0.48	0.38	0.40	0.45	0.44	0.37	0.42	0.48	0.45	0.34	0.32	0.34	0.25	0.27	0.30	0.32
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.85	0.82	0.89	0.83	0.81	0.86	0.82	0.89	0.88	0.83	0.86	0.83	0.91	0.88	0.90	0.87	0.85	0.87	0.93	0.92
mser	Precision	1.00	0.98	1.00	0.98	0.98	0.98	1.00	0.98	1.00	1.00	1.00	1.00	0.93	0.78	0.83	0.86	0.78	0.83	0.83	0.89
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.93	0.95	0.83	0.95	0.91	0.88	0.93	0.83	0.87	0.91	0.92	0.92	0.81	0.86	0.83	0.86	0.89	0.87	0.80	0.82
ocv	Precision	0.15	0.03	0.40	0.10	0.00	0.00	0.03	0.48	0.32	0.07	0.14	0.05	0.09	0.00	0.00	0.00	0.00	0.00	0.27	0.10
	Recall	0.17	0.04	0.43	0.11	0.00	0.00	0.04	0.57	0.35	0.07	0.15	0.06	0.09	0.00	0.00	0.00	0.00	0.00	0.31	0.11
	AMDSC	0.54	0.51	0.56	0.53	-	-	0.51	0.55	0.55	0.52	0.53	0.51	0.51	-	-	-	-	-	0.51	0.51
p-algo	Precision	0.15	0.39	0.25	0.36	0.33	0.33	0.36	0.34	0.22	0.24	0.23	0.18	1.00	1.00	1.00	1.00	0.25	0.15	1.00	1.00
	Recall	0.50	0.56	0.56	0.57	1.00	1.00	0.98	0.98	0.98	0.59	0.56	0.69	0.43	0.50	0.89	0.87	0.96	0.98	0.85	0.72
	AMDSC	0.92	0.91	0.88	0.94	0.92	0.95	0.92	0.90	0.94	0.92	0.95	0.94	0.95	0.96	0.96	0.94	0.93	0.94	0.95	0.97
mll	Precision	1.00	1.00	0.98	1.00	0.96	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.98	0.98	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.88	0.92	0.76	0.89	0.88	0.84	0.89	0.76	0.80	0.88	0.86	0.89	0.79	0.84	0.82	0.85	0.88	0.86	0.78	0.81
proposed	Precision	1.00	1.00	0.98	1.00	0.98	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.91	0.96	0.83	0.94	0.95	0.93	0.96	0.85	0.88	0.92	0.92	0.94	0.87	0.92	0.89	0.91	0.93	0.90	0.85	0.87

Table 14 – Blob extractors results for image set *low-contrast* in numerical values.

		Image																			
		0002	0018	0038	0055	0077	0092	0116	0128	0155	0165	0194	0202	0233	0239	0257	0272	0276	0311	0313	0346
dog	Precision	0.79	0.84	0.77	0.78	0.75	0.71	0.74	0.77	0.76	0.75	0.71	0.75	0.87	0.83	0.82	0.77	0.76	0.76	0.68	0.82
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.91	0.92	0.85	0.93	0.91	0.92	0.91	0.86	0.80	0.87	0.88	0.87	0.89	0.90	0.90	0.85	0.90	0.88	0.84	0.89
log	Precision	1.00	1.00	1.00	1.00	1.00	1.00	0.95	0.78	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.94	0.95	0.88	0.95	0.95	0.95	0.94	0.89	0.82	0.90	0.95	0.93	0.89	0.93	0.92	0.85	0.93	0.89	0.79	0.93
mser	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.94	0.96	0.89	0.96	0.96	0.97	0.95	0.90	0.82	0.89	0.96	0.95	0.92	0.95	0.93	0.87	0.94	0.92	0.81	0.95
ocv	Precision	0.28	0.40	0.63	0.42	0.18	0.15	0.25	0.52	0.68	0.00	0.02	0.13	0.46	0.16	0.12	0.59	0.16	0.25	0.88	0.10
	Recall	0.33	0.46	0.72	0.50	0.20	0.20	0.26	0.59	0.83	0.00	0.02	0.15	0.54	0.17	0.13	0.69	0.19	0.28	0.94	0.11
	AMDSC	0.53	0.52	0.58	0.52	0.52	0.51	0.52	0.55	0.56	-	0.52	0.54	0.53	0.52	0.53	0.54	0.53	0.53	0.57	0.54
p-algo	Precision	1.00	1.00	1.00	1.00	1.00	0.04	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Recall	0.22	0.50	0.44	0.65	0.72	1.00	0.69	0.74	0.56	0.70	0.81	0.69	0.22	0.26	0.54	0.57	0.69	0.83	0.80	0.50
	AMDSC	0.94	0.96	0.87	0.96	0.96	0.97	0.95	0.89	0.82	0.89	0.96	0.96	0.88	0.94	0.94	0.87	0.95	0.92	0.82	0.94
mll	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.85	0.85	0.76	0.84	0.84	0.85	0.84	0.78	0.73	0.91	0.91	0.82	0.81	0.87	0.83	0.74	0.84	0.79	0.69	0.85
proposed	Precision	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Recall	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	AMDSC	0.89	0.90	0.81	0.90	0.90	0.93	0.91	0.83	0.75	0.93	0.94	0.89	0.86	0.90	0.89	0.81	0.91	0.86	0.75	0.89

Table 15 – Mean of the three metrics evaluated for all blob extractors for all image sets of the printed pattern experiments in numerical values. Results are presented as $x \pm y$, where x is the mean value of the metric over all images of the set and y is the standard deviation.

		Image Set					
		distance	*rotation*	*obliquity*	*radial-distortion*	*uneven-contrast*	*low-contrast*
dog	Precision	0.127±0.048	0.105±0.017	0.176±0.041	0.180±0.086	0.646±0.065	0.772±0.047
	Recall	1.000±0.000	1.000±0.000	0.995±0.017	0.998±0.006	0.983±0.025	1.000±0.000
	AMDSC	0.893±0.039	0.898±0.061	0.861±0.046	0.869±0.034	0.916±0.019	0.883±0.033
log	Precision	0.039±0.008	0.032±0.007	0.040±0.006	0.045±0.011	0.402±0.087	0.986±0.049
	Recall	1.000±0.000	1.000±0.000	0.999±0.004	0.994±0.012	0.999±0.004	1.000±0.000
	AMDSC	0.842±0.135	0.826±0.088	0.835±0.059	0.866±0.039	0.865±0.035	0.909±0.045
mser	Precision	0.194±0.029	0.187±0.022	0.290±0.031	0.232±0.033	0.932±0.082	1.000±0.000
	Recall	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
	AMDSC	0.829±0.078	0.834±0.079	0.883±0.057	0.887±0.054	0.879±0.049	0.922±0.044
ocv	Precision	0.060±0.100	0.065±0.093	0.051±0.081	0.049±0.072	0.112±0.145	0.319±0.242
	Recall	0.228±0.385	0.271±0.388	0.132±0.202	0.146±0.227	0.125±0.165	0.366±0.277
	AMDSC	0.606±0.083	0.550±0.047	0.522±0.019	0.525±0.020	0.526±0.016	0.534±0.019
p-algo	Precision	0.457±0.104	0.529±0.090	0.909±0.067	0.649±0.172	0.488±0.350	0.952±0.214
	Recall	0.732±0.150	0.575±0.182	0.719±0.131	0.647±0.077	0.758±0.209	0.606±0.208
	AMDSC	0.897±0.079	0.904±0.054	0.926±0.045	0.943±0.025	0.935±0.022	0.919±0.047
mll	Precision	0.504±0.121	0.421±0.077	0.712±0.054	0.647±0.105	0.993±0.011	1.000±0.000
	Recall	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
	AMDSC	0.817±0.089	0.832±0.081	0.880±0.056	0.886±0.054	0.844±0.047	0.821±0.057
proposed	Precision	0.513±0.098	0.450±0.080	0.722±0.056	0.652±0.094	0.995±0.008	1.000±0.000
	Recall	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
	AMDSC	0.834±0.087	0.845±0.081	0.892±0.057	0.899±0.052	0.907±0.038	0.873±0.054

Table 16 – Blob extractors results for image set *pipe* in numerical values.

		Image																			
		0000	0001	0002	0003	0050	0051	0052	0053	0100	0101	0102	0103	0150	0151	0152	0153	0200	0201	0202	0203
dog	Precision	0.02	0.02	0.02	0.04	0.08	0.06	0.06	0.07	0.02	0.02	0.03	0.02	0.05	0.01	0.04	0.05	0.02	0.01	0.02	0.01
	Recall	0.92	0.83	0.83	0.92	0.82	0.87	0.67	0.67	0.69	0.56	0.63	0.63	0.31	0.14	0.23	0.32	0.82	0.82	0.88	0.73
	AMDSC	0.79	0.85	0.80	0.87	0.88	0.88	0.88	0.88	0.77	0.79	0.83	0.81	0.77	0.68	0.82	0.80	0.86	0.86	0.87	0.85
log	Precision	0.04	0.04	0.04	0.05	0.21	0.16	0.14	0.15	0.07	0.08	0.11	0.08	0.18	0.08	0.16	0.20	0.07	0.03	0.03	0.03
	Recall	0.92	0.92	0.92	0.92	1.00	1.00	0.96	0.96	1.00	1.00	1.00	0.94	0.90	0.90	1.00	0.93	1.00	1.00	0.96	0.96
	AMDSC	0.81	0.78	0.79	0.81	0.80	0.75	0.86	0.85	0.82	0.84	0.85	0.86	0.71	0.70	0.71	0.70	0.70	0.83	0.72	0.72
mser	Precision	0.08	0.09	0.07	0.11	0.31	0.24	0.26	0.24	0.09	0.09	0.14	0.11	0.28	0.19	0.27	0.29	0.08	0.05	0.05	0.05
	Recall	0.92	1.00	0.92	1.00	0.86	0.96	0.92	0.88	1.00	0.88	0.94	1.00	0.93	0.86	1.00	1.00	0.91	0.95	1.00	1.00
	AMDSC	0.90	0.86	0.90	0.87	0.82	0.88	0.87	0.83	0.78	0.74	0.81	0.82	0.85	0.80	0.86	0.83	0.84	0.89	0.82	0.82
ocv	Precision	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Recall	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	AMDSC	-	-	-	-	-	0.67	-	-	-	0.51	-	-	-	-	-	-	-	-	-	-
p-algo	Precision	0.22	0.12	0.17	0.22	0.45	0.68	0.70	0.80	0.39	0.20	0.21	0.26	0.67	0.38	0.54	0.94	0.10	0.94	0.89	0.84
	Recall	0.33	0.58	0.58	0.33	0.77	0.57	0.67	0.67	0.56	0.63	0.69	0.69	0.76	0.52	0.73	0.57	0.82	0.68	0.67	0.62
	AMDSC	0.76	0.80	0.78	0.78	0.68	0.76	0.80	0.84	0.72	0.66	0.67	0.65	0.71	0.77	0.76	0.74	0.75	0.85	0.81	0.80
mll	Precision	0.09	0.11	0.09	0.14	0.50	0.35	0.35	0.31	0.13	0.13	0.21	0.14	0.44	0.29	0.42	0.47	0.10	0.06	0.06	0.08
	Recall	0.92	1.00	0.92	1.00	0.86	0.96	0.92	0.92	1.00	0.94	1.00	0.94	0.93	0.86	1.00	1.00	0.95	0.95	0.96	0.96
	AMDSC	0.90	0.88	0.90	0.89	0.86	0.88	0.91	0.89	0.83	0.82	0.86	0.86	0.89	0.89	0.90	0.89	0.87	0.91	0.88	0.88
proposed	Precision	0.10	0.11	0.10	0.14	0.51	0.34	0.34	0.33	0.14	0.14	0.22	0.15	0.44	0.31	0.43	0.47	0.10	0.06	0.06	0.08
	Recall	0.92	0.92	0.92	1.00	0.86	0.96	0.92	0.92	1.00	0.94	0.94	0.94	0.90	0.90	1.00	0.96	0.91	0.95	0.96	0.92
	AMDSC	0.89	0.86	0.89	0.85	0.82	0.86	0.90	0.89	0.81	0.80	0.74	0.78	0.80	0.85	0.86	0.79	0.86	0.88	0.87	0.86

Table 17 – Mean of the three metrics evaluated for all blob extractors for the pipe experiments in numerical values. Results are presented as $x \pm y$, where x is the mean value of the metric over all images of the set and y is the standard deviation.

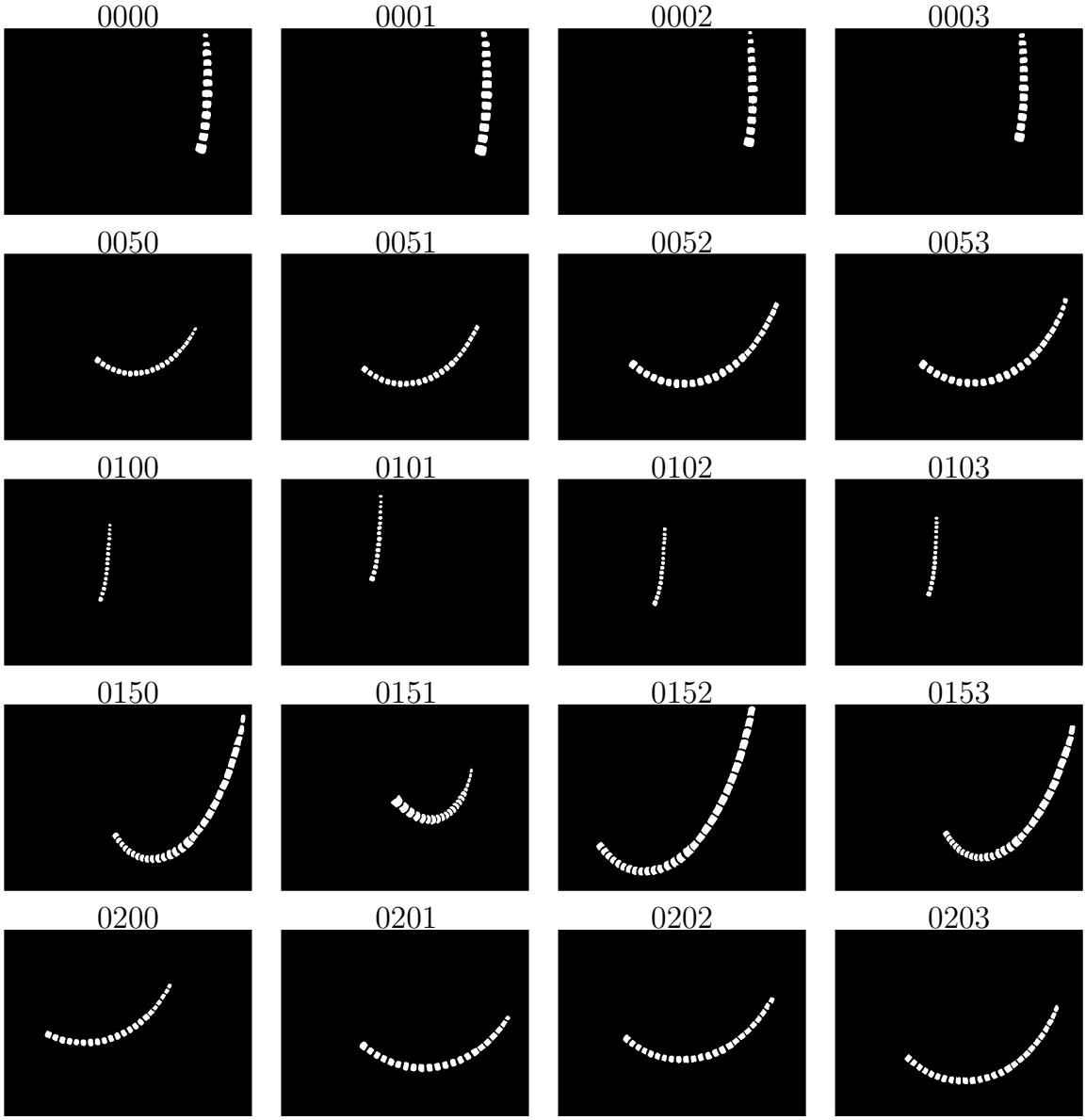
		Image Set
		pipe
dog	Precision	0.034±0.021
	Recall	0.664±0.236
	AMDSC	0.827±0.051
log	Precision	0.096±0.061
	Recall	0.959±0.039
	AMDSC	0.781±0.060
mser	Precision	0.154±0.093
	Recall	0.945±0.053
	AMDSC	0.841±0.041
ocv	Precision	0.001±0.005
	Recall	0.005±0.017
	AMDSC	0.588±0.112
p-algo	Precision	0.486±0.296
	Recall	0.622±0.125
	AMDSC	0.754±0.057
mll	Precision	0.224±0.151
	Recall	0.949±0.044
	AMDSC	0.880±0.024
proposed	Precision	0.228±0.153
	Recall	0.936±0.036
	AMDSC	0.843±0.044

Table 18 – Proposed pattern detector results for image set *pipe* in numerical values.

		Image																			
		0000	0001	0002	0003	0050	0051	0052	0053	0100	0101	0102	0103	0150	0151	0152	0153	0200	0201	0202	0203
proposed	Precision	1.00	0.92	0.92	1.00	1.00	0.96	0.83	0.87	1.00	1.00	0.94	1.00	0.93	1.00	0.97	0.96	0.89	0.82	0.87	0.96
	Recall	0.92	0.92	0.92	1.00	0.82	0.96	0.79	0.83	1.00	0.94	0.94	0.88	0.90	0.86	0.97	0.96	0.73	0.82	0.83	0.85

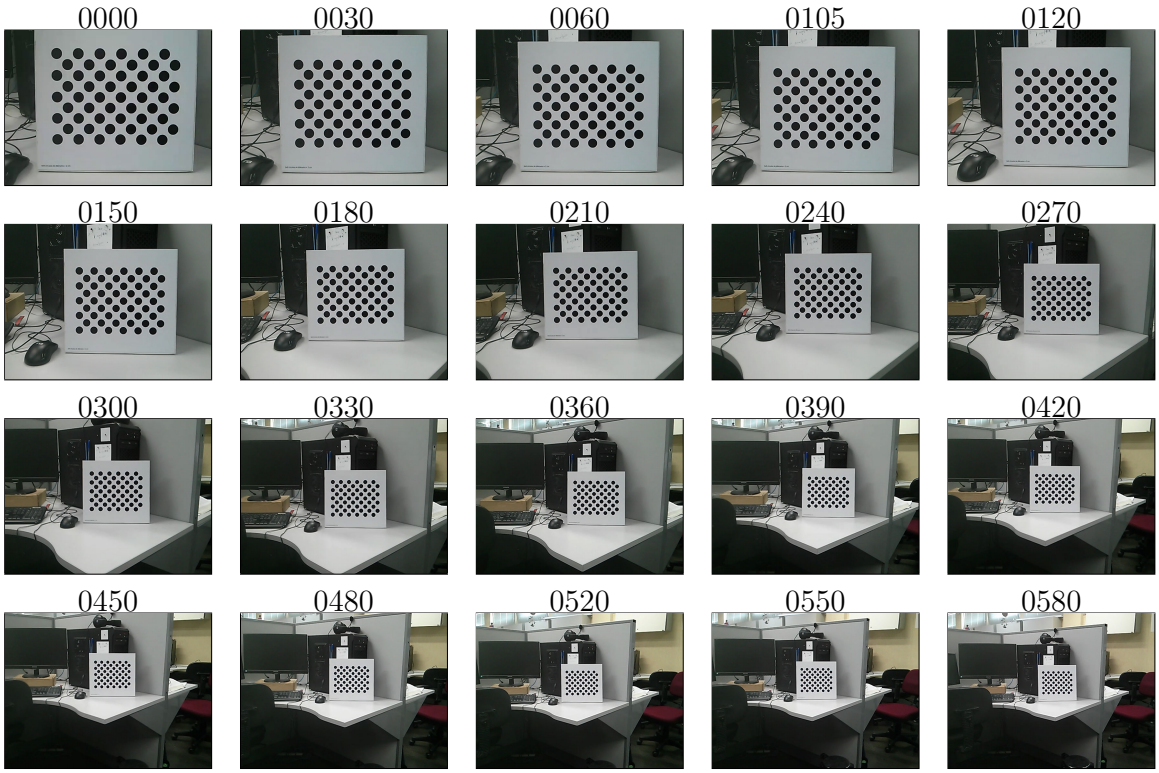
APPENDIX B – FIGURES

This appendix groups supplementary figures that were subtracted from the main body of the Thesis for a better readability.



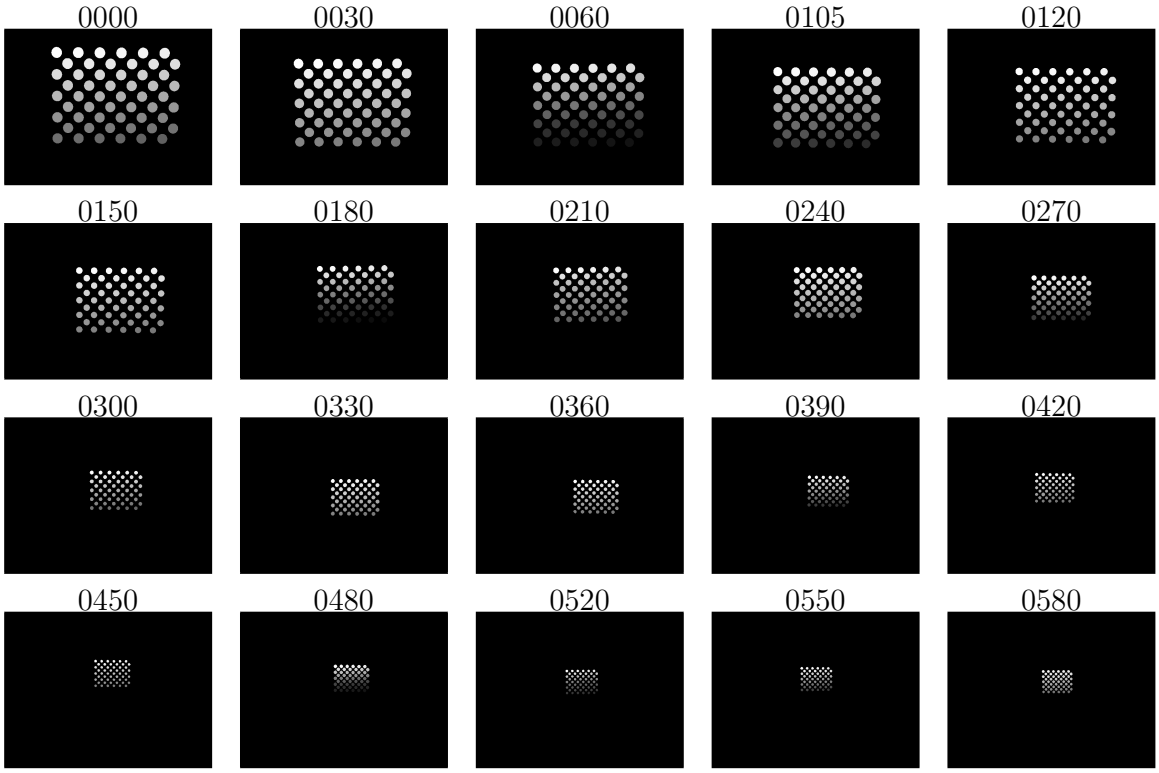
Source: The author

Figure 82 – The hand-labeled ground-truth images of the *pipe* image set.



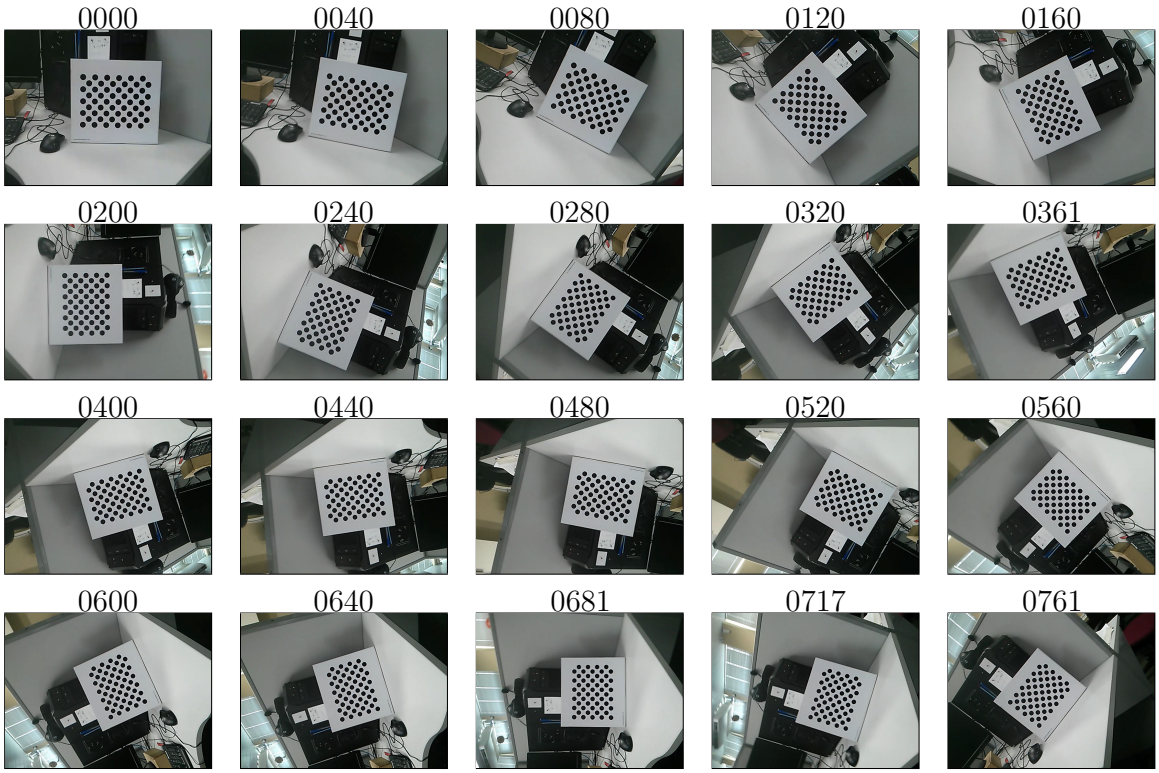
Source: The author

Figure 83 – The input images of the *distance* image set.



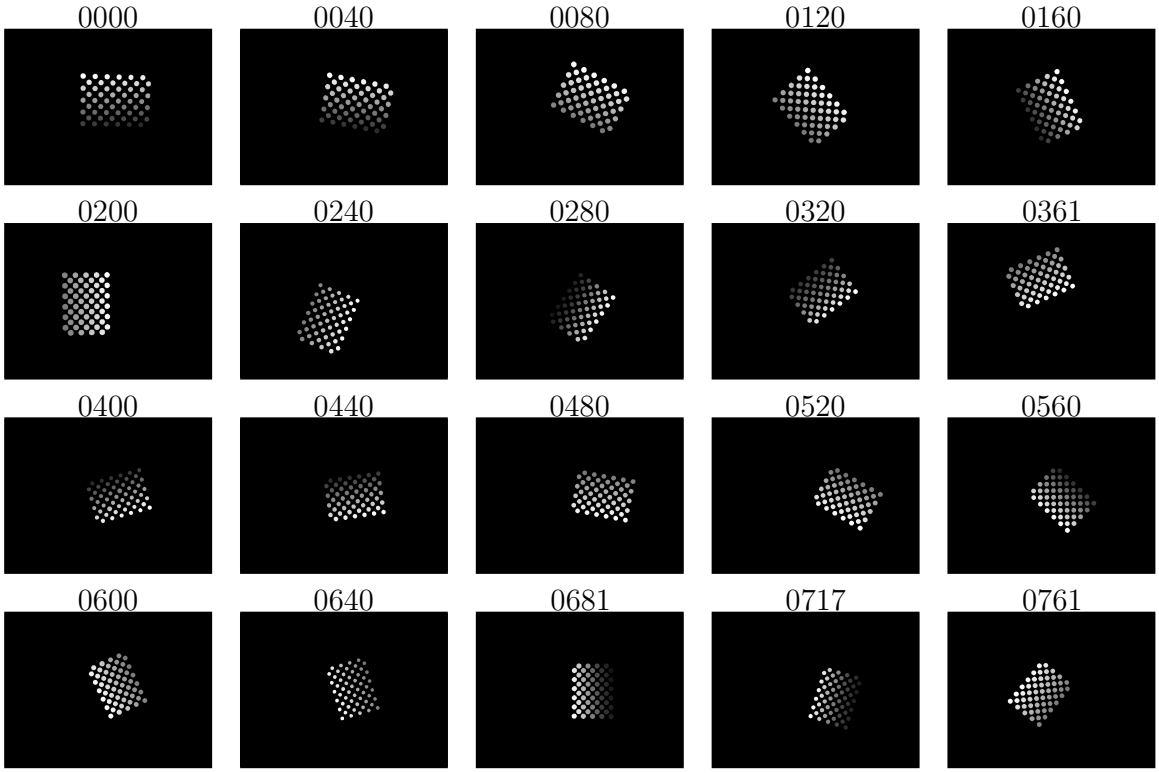
Source: The author

Figure 84 – The hand-labeled ground-truth images of the *distance* image set.



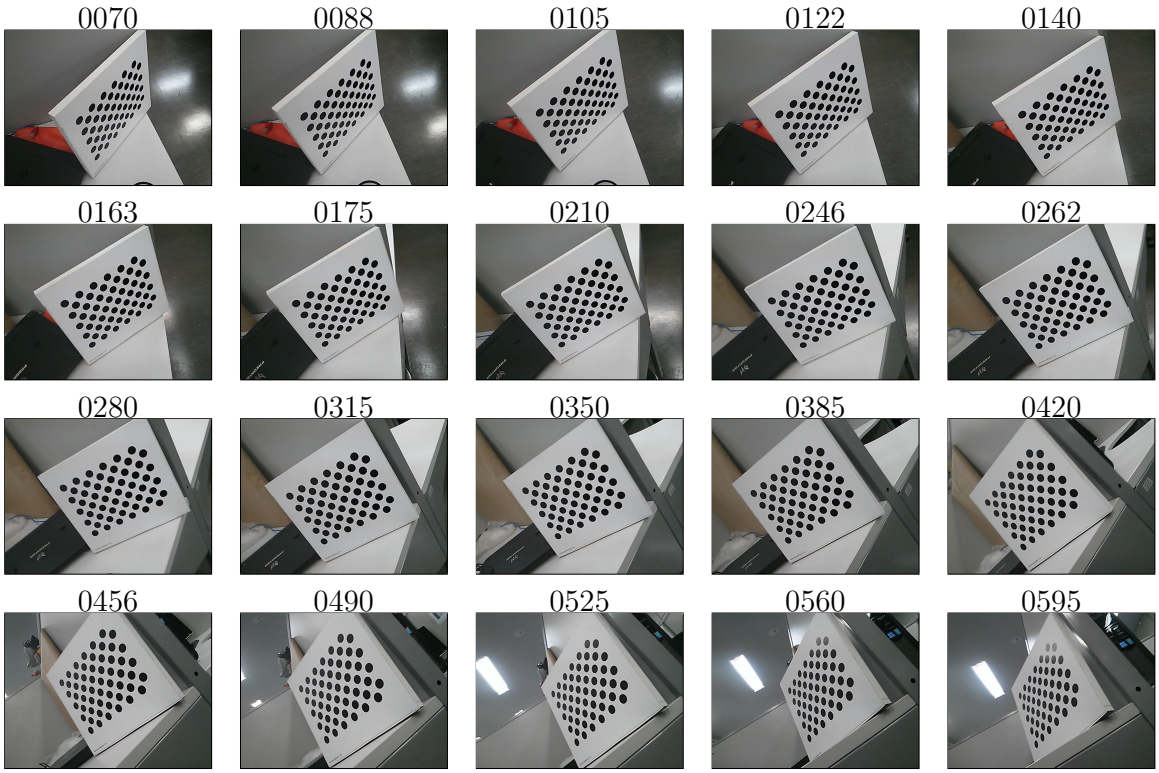
Source: The author

Figure 85 – The input images of the *rotation* image set.



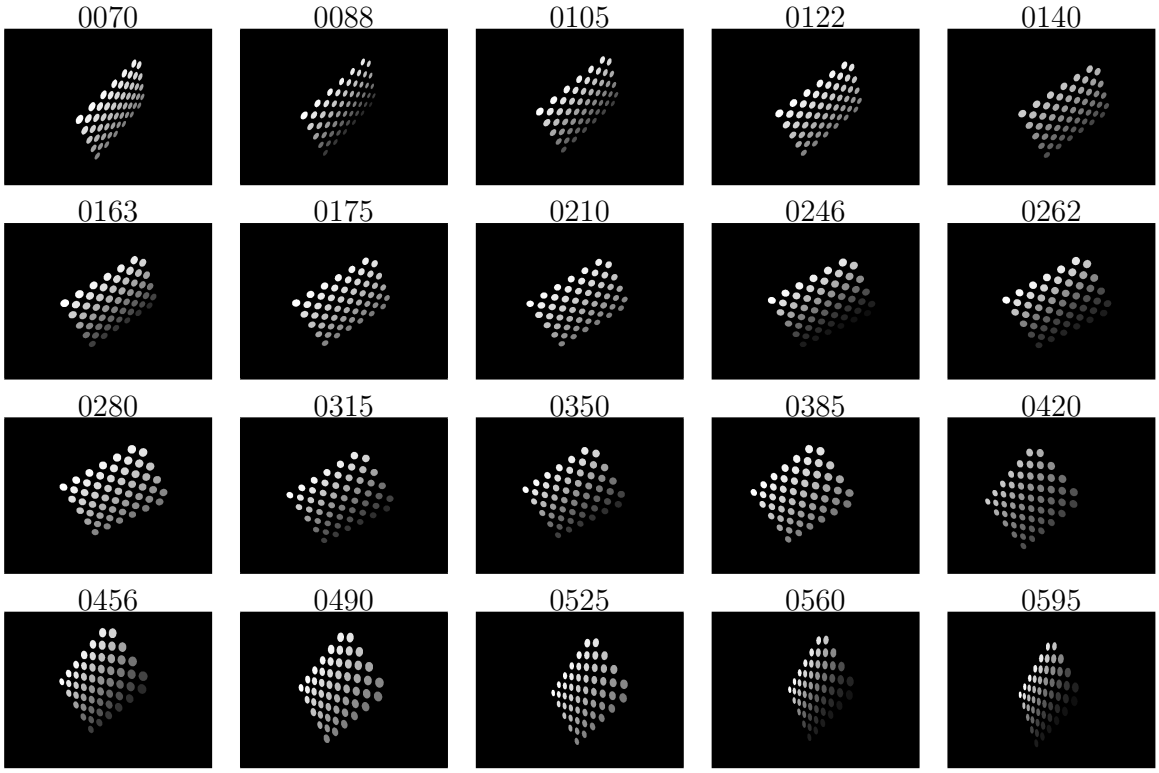
Source: The author

Figure 86 – The hand-labeled ground-truth images of the *rotation* image set.



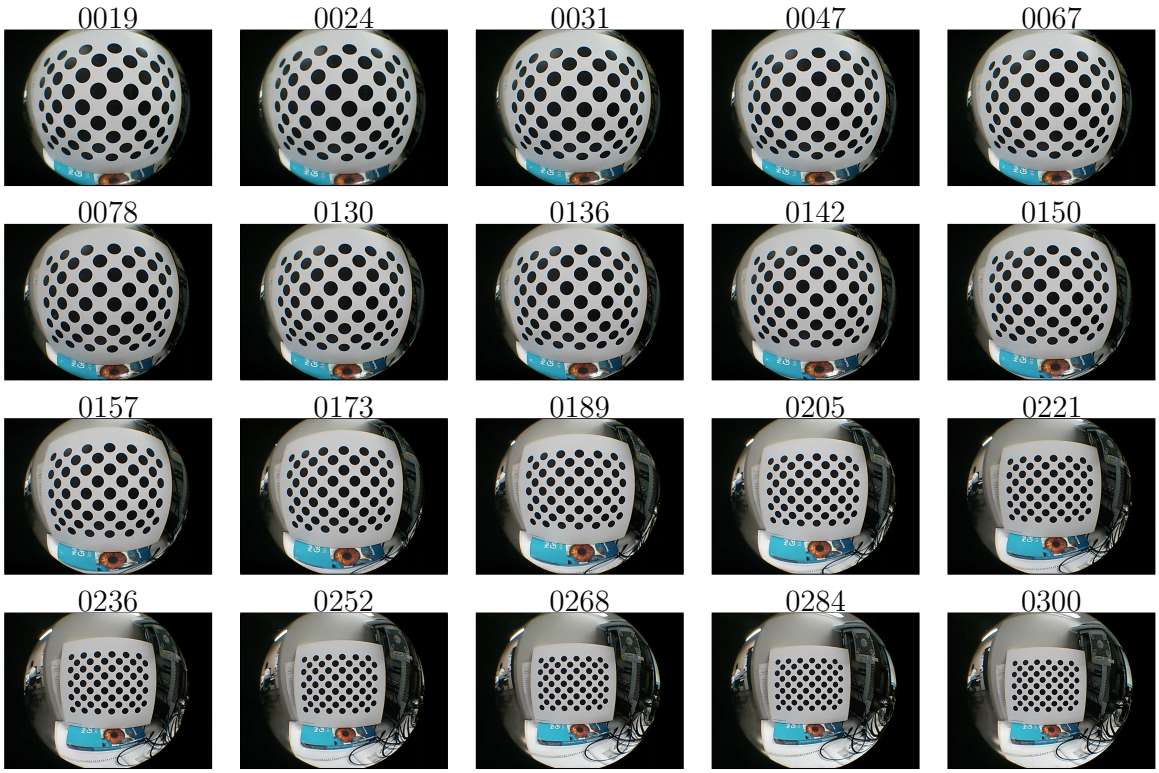
Source: The author

Figure 87 – The input images of the *obliquity* image set.



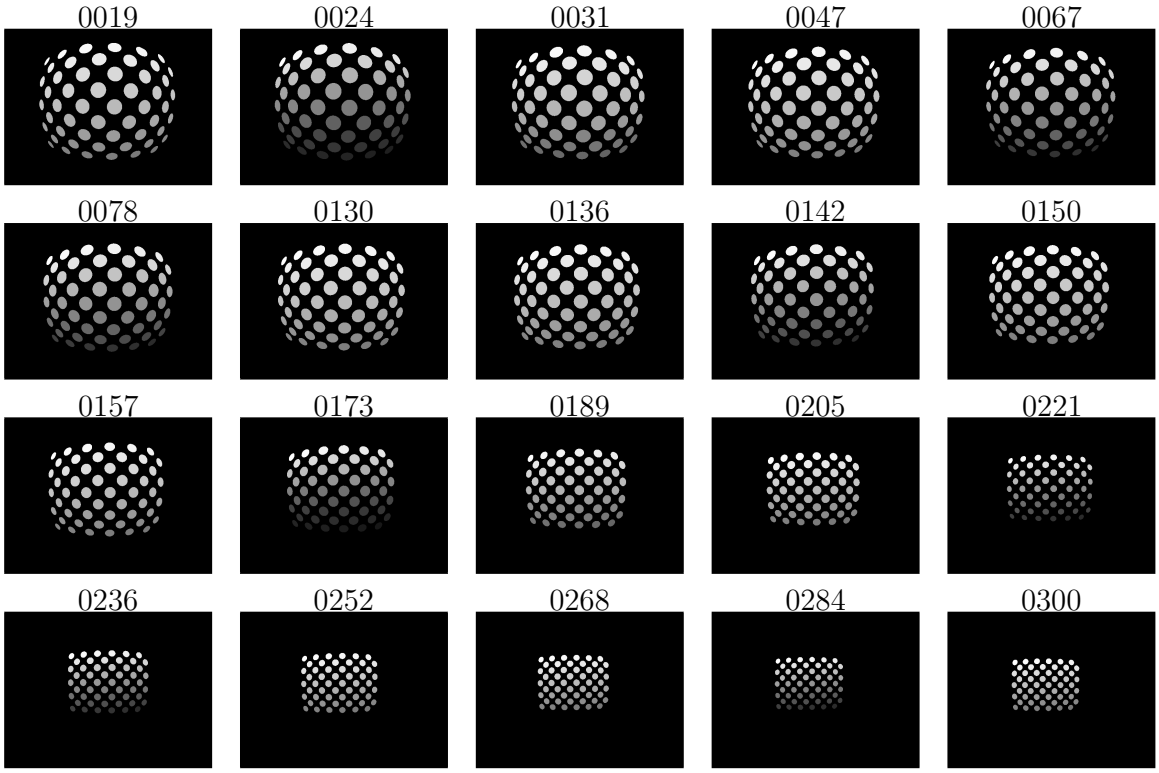
Source: The author

Figure 88 – The hand-labeled ground-truth images of the *obliquity* image set.



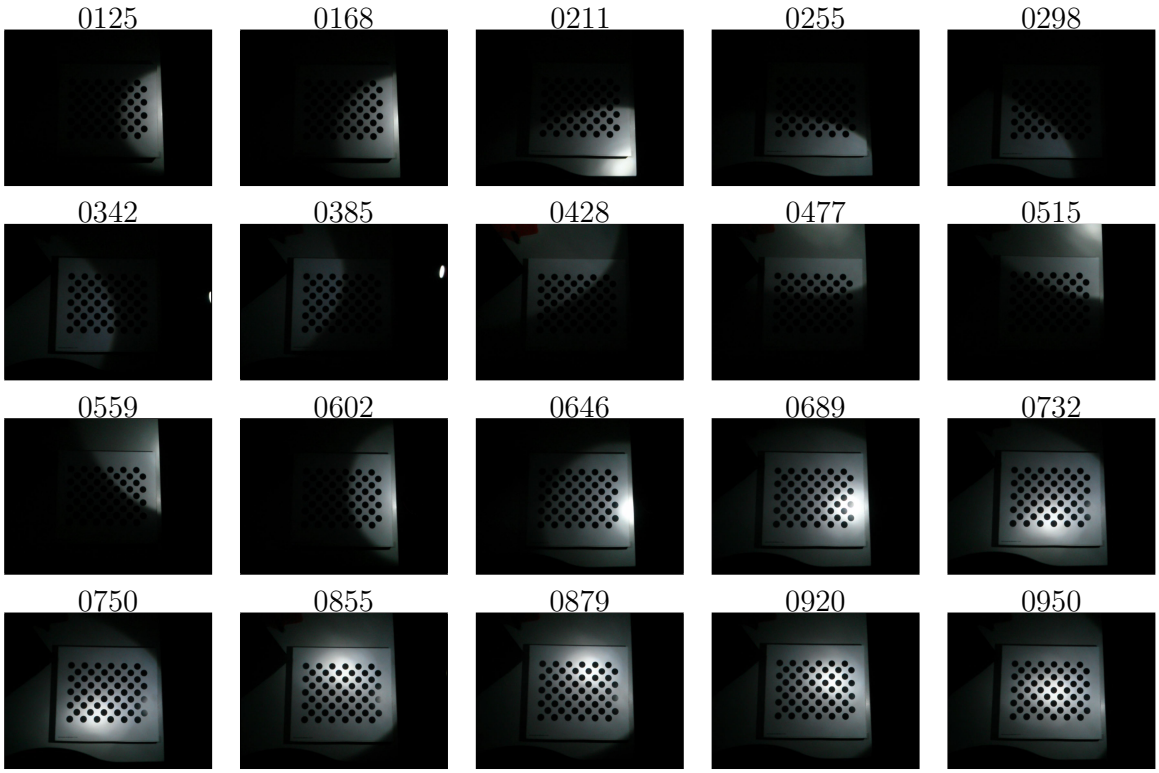
Source: The author

Figure 89 – The input images of the *radial-distortion* image set.



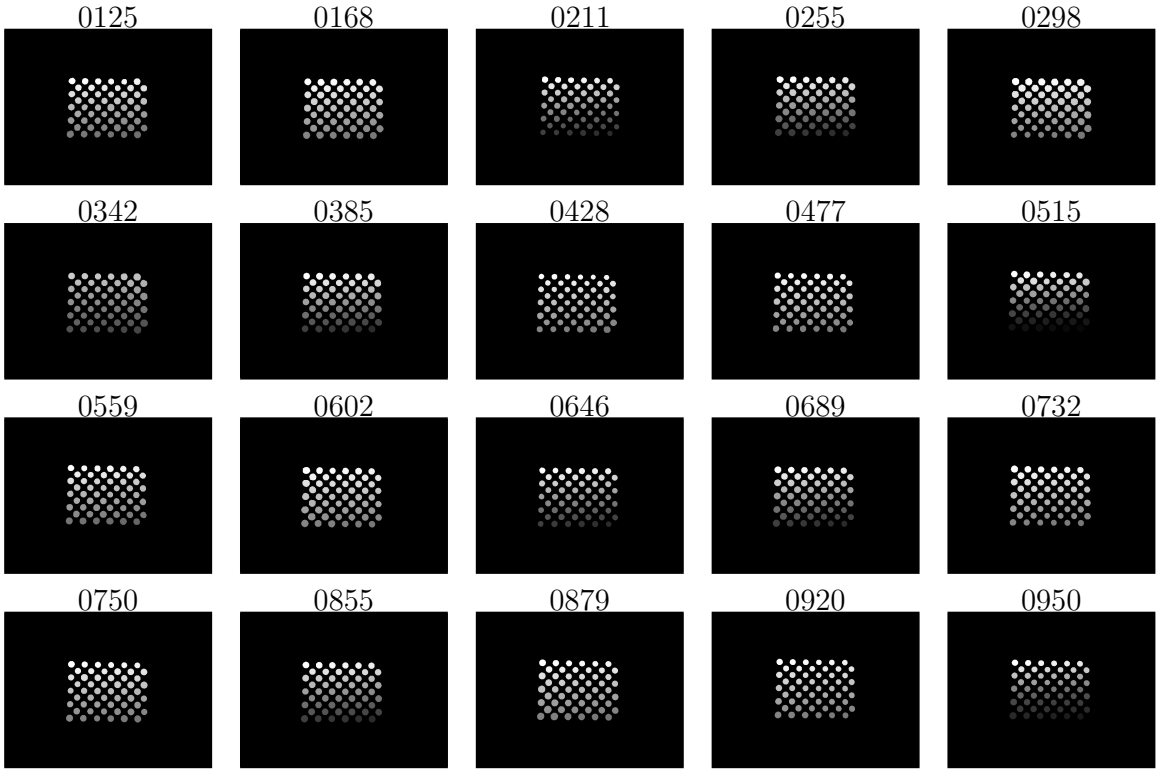
Source: The author

Figure 90 – The hand-labeled ground-truth images of the *radial-distortion* image set.



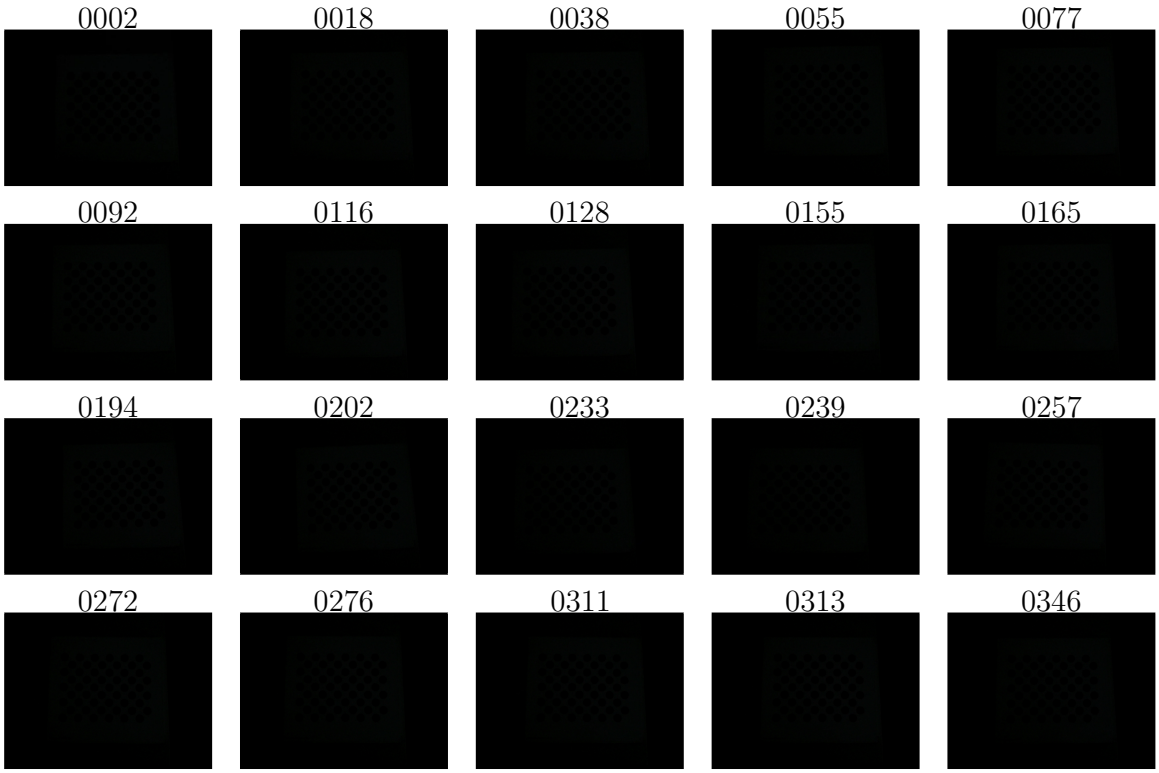
Source: The author

Figure 91 – The input images of the *uneven-contrast* image set.



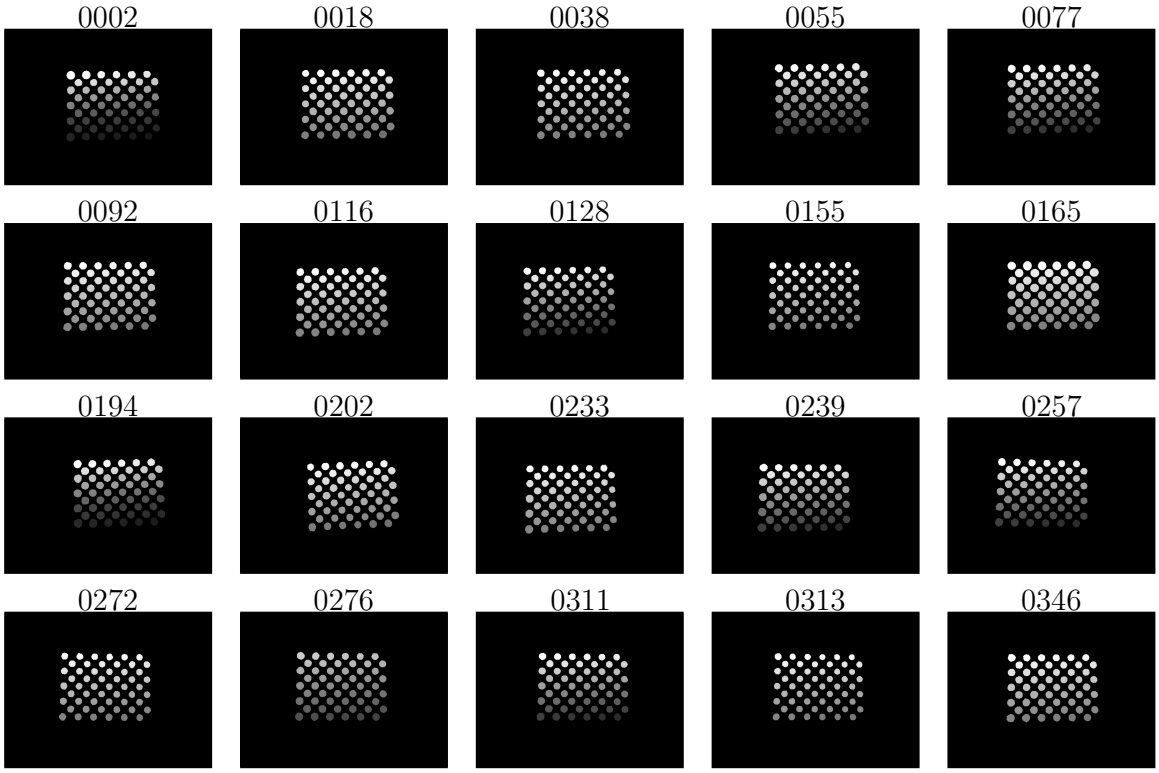
Source: The author

Figure 92 – The hand-labeled ground-truth images of the *uneven-contrast* image set.



Source: The author

Figure 93 – The input images of the *low-contrast* image set.



Source: The author

Figure 94 – The hand-labeled ground-truth images of the *low-contrast* image set.