# DETECTOR DE PONTOS DE INTERESSE BASEADO EM CARACTERÍSTICAS VISUAIS E DE PROFUNDIDADE

LEVI OSTERNO VASCONCELOS

# DETECTOR DE PONTOS DE INTERESSE BASEADO EM CARACTERÍSTICAS VISUAIS E DE PROFUNDIDADE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: ERICKSON RANGEL DO NASCIMENTO
COORIENTADOR: MARIO FERNANDO MONTENEGRO CAMPOS

Belo Horizonte
Março de 2015

LEVI OSTERNO VASCONCELOS

# A KEYPOINT DETECTOR BASED ON VISUAL AND DEPTH FEATURES.

Dissertation presented to the Graduate Program in Ciência da Computação of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Ciência da Computação.

ADVISOR: ERICKSON RANGEL DO NASCIMENTO
CO-ADVISOR: MARIO FERNANDO MONTENEGRO CAMPOS

Belo Horizonte
March 2015

Vasconcelos, Levi Osterno

V278d Detector de pontos de interesse baseado em características visuais e de profundidade / Levi Osterno Vasconcelos. — Belo Horizonte, 2015
xx, 49 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Erickson Rangel do Nascimento
Coorientador: Mario Fernando Montenegro Campos

1. Computação - Teses. 2. Visão por computador. 3. RGB-D, 4. Keypoints. I. Orientador. II. Coorientador. III Título.

CDU 519.6*82.10(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

# FOLHA DE APROVAÇÃO

Detector de pontos de interesse baseado em características visuais e de profundidade

**LEVI OSTERNO VASCONCELOS**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ERICKSON RANGEL DO NASCIMENTO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. MARIO FERNANDO MONTENEGRO CAMPOS - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. ALEXEI MANSO CORREA MACHADO
Departamento de Ciência da Computação - PUCMG

PROF. WILLIAM ROBSON SCHWARTZ
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 27 de abril de 2015.

# Acknowledgments

I am truly thankful for everyone who helped me, directly and indirectly, towards the conclusion of this work.

First, I would like to thank my family; my parents José Sidney and Carmen Angela. My brother Jansen and my sister Tainá and last, but not least, my wife Rebeca for all the given support during the whole process. For you all, my sincere gratitude and love.

To my advisors Erickson Rangel do Nascimento and Mario Fernando Montenegro Campos (Co-orientation), I would like to register my admiration and sincere gratitude for all the lessons, orientation and knowledge that were directed towards me. It was a great and huge experience of self-improvement and learning. Thank you.

Also, the amazing laboratory that has built an incredible community, I would like to thank the VeRLab crew: Paulo Drews, Elerson Rubens, Jhielson Pimentel, Omar Pino, David Saldana, Hector Azpurua, Daniel Balbino, Rafael Colares, Ramon Melo, Samuel Servulo, Igor Campos, Vinicius Santos, Claudio Fernandes and many others that i am forgetting and, for sure, a will be ashamed of once I remember.

For my personal friends, for all the support and fun we had during the whole process: Rogerio Fonteles, Samuel Servulo, Vladimir Portela, Paulo Drews, Igor Campos, Omar Pino, Elerson Rubens, Jhielson Pimentel, Clayson Celes, Caio Rodrigues, Gustavo Malkomes, Rodrigo Viana.

For the financial and infra-structure support, my sincere thanks for CNPQ, VeRLab and UFMG.

# Resumo

Em sistemas de visão computacional, é prática comum representar partes de imagens por um conjunto de pontos cuidadosamente selecionados, tais pontos são identificados de acordo com características convenientes à aplicação. Exemplos de características são: arestas, quinas, *blobs* e *ridges*. Tais regiões são denominadas como *pontos de interesse* ou *keypoints*.

Inúmeras aplicações em visão computacional necessitam da extração de pontos de interesse. Apesar da grande quantidade de trabalhos presentes na literatura de detectores de pontos de interesse em imagens, poucas metodologias combinam, de forma eficiente, informações visuais e geométricas.

Neste trabalho, apresentamos *Keypoints from Visual and Depth Data* (KVD), um novo detector de pontos de interesse invariante a escala que combina informação de textura e geometria utilizando operações de baixo custo e uma árvore de decisão. Também apresentamos a análise de duas abordagens para a fusão de informação para dados RGB-D: a abordagem aditiva e a concatenação.

Graças à fusão de informação, o algoritmo é capaz de encontrar *keypoints* confiáveis mesmo em cenários desafiadores, como em ambientes escuros ou com pouca textura. Apresentamos vários resultados que mostram que nossa abordagem produz o melhor resultado quando comparado com outros métodos da literatura, mantendo boas taxas de repetibilidade para: rotação, translação e escala, bem como robustez a ruídos tanto na informação visual, quanto na geométrica. Nosso método, quando comparado a algoritmos com tipos de entrada equivalentes, possui o menor gasto de tempo.

**Palavras-chave:** KVD, pontos de interesse, RGB-D, detector.

# Abstract

In computer vision systems, its a common practice to represent patches of an image using a set of carefully chosen points, those points are identified based on desired local features that are convenient to the application. Examples of such features are edges, corners, blobs and ridges. Those areas are commonly denoted as *keypoints*.

One of the first steps in numerous computer vision tasks is the extraction of keypoints. Despite the large number of works proposing image keypoint detectors, only a few methodologies are able to efficiently use both visual and geometrical information.

In this work we introduce Keypoints from Visual and Depth Data (KVD), a novel keypoint detector which is scale invariant and combines intensity and geometrical data through low-cost operations and a decision tree. We also present an evaluation of two different methods for combining and extracting information from RGB-D data: the additive and the concatenation approach.

Thanks to the information fusion, the algorithm is capable to find reliable keypoints even under challenging scenarios, such as dark rooms and textureless environments. Results from several experiments that show that our methodology produces the best performing detector when comparing to state-of-the-art methods are presented, with good repeatability scores for rotations, translations and scale changes, as well as robustness to corruptions on either visual or geometric information. When compared to algorithms with equivalent input type, our algorithm yields the best time results.

**Keywords:** keypoint, detector, KVD, RGB-D.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

Over the years, selecting a set of points of interest in images has been an omnipresent step in a large number of computer vision methodologies. A careful choice of points of interest in an image may avoid the effect of noisy pixels and identify regions rich in information, which allows for an effective description of the regions containing these points. Also, the use of an image subset makes it possible to tackle cluttered backgrounds and occlusions in object recognition [Lowe., 2004; Chen and Bhanu, 2004] and scene understanding applications, while greatly reduces the search space and the computational burden required, which further can be focused on more likely relevant areas for the problem.

Moreover, the ever growing volume of data, such as high resolution images, RGB-D data (composed of visual and three dimensional data) and the massive image repositories available in the web, make the creation of keypoint detectors crucial for a large number of computer vision techniques, specially by reducing the data search space, thus making the processing of such data a manageable task.

The detection and selection of a set of points of interest (Figure 1.1), to which we will henceforth refer as *keypoints*, consist in looking for unique points located in discriminative


**Figure 1.1.** Keypoints being applied to object identification within different images. Each line marks the correspondence between keypoints from both images. Figure taken from Rusu [2010]

regions of the image, that will account for good *repeatability*, which in turn may lead to less ambiguity. There is a vast body of literature on keypoint detectors, of which [Harris and Stephens, 1988; Lowe., 2004; Rosten et al., 2010; Rublee et al., 2011] are well known representatives.

Broadly speaking, the main task of a keypoint detector is to assign a saliency score to each pixel of an image. This score is then used to select a smaller subset of pixels that presents the following properties (Tuytelaars and Mikolajczyk [2008]):

- Repeatability: The selected pixels should remain stable under several image perturbations;
- Distinctiveness: The neighborhood around each keypoint should present an intensity pattern with strong variations;
- Locality: The features should be a function of local information;
- Accurately localizable: The localization process should be less error-prone with respect to scale and shape;
- Efficiency: Low processing time.

## 1.1 Motivation

From the time the first corner detection was proposed in the late 70's, the quest for the optimal keypoint detector has been marked by an impressive advancement. Scale invariance [Lowe., 2004; Rublee et al., 2011], estimation of canonical direction of the keypoints and robustness to noise [Lowe., 2004], reduction in processing time [Rosten et al., 2010; Rublee et al., 2011], are some of the improvements seen throughout the years. Keypoint detection based on visual and geometrical data has been thoroughly investigated, and every year new techniques are debuted. In spite of all the advances in keypoint detection, there is a gap that needs to be further addressed, namely, the efficient use of distinct type of data, such as visual and geometrical.

The richness of information being constantly engrafted in images has naturally pushed the envelope for several image based keypoint detection techniques. The vision literature presents numerous works that use different cues for keypoint detection based on pixel intensity [Harris and Stephens, 1988; Lowe., 2004; Rublee et al., 2011; Rosten et al., 2010]. Nearly all of those techniques are based on the analysis of local gradients. Keypoint detectors based on images alone seldom use other information such as the scene's geometry.

**Figure 1.2.** A point cloud created from a RGB-D image and the keypoints detected (red points) by our methodology.

As a consequence, common issues concerning real scenes, like variation in illumination and textureless objects, may dramatically decrease the performance of such techniques.

Nowadays 3D data abounds, such as those obtained from range images. Compared to 2D cameras, the newer 3D sensors are less sensitive to illumination and provide the scale associated to each point. Steder et al. [2011] and Zaharescu et al. [2009] are some examples of approaches that use 3D information.

Although 3D sensing techniques have been largely available, such as those based on LIDAR, time-of-flight (Canesta), and projected texture stereo, they are still very expensive and demand substantial engineering effort to be acquired. With the recent introduction of fast and inexpensive RGB-D sensors, the integration of synchronized intensity (color) and depth has become easier to obtain. The growing availability of inexpensive, real time depth sensors, has made depth images increasingly popular, inducing many new computer vision techniques.

RGB-D systems output color images and the corresponding pixel depth information enabling the acquisition of both depth and visual cues in real-time. These systems have opened the way to obtain 3D information with unprecedented trade-off between richness and cost. One such system is the Kinect, a low cost commercially available system that produces RGB-D data in real-time for gaming applications.

The 3D information provided by such technologies, allows computer vision systems to approach more challenging environments such as Figure 1.3. One can extract geometric cues from 3D information to cope with illumination and texture absence. There is an increasing demand for systems capable of mapping and monitoring complex environments such as caves and mines which usually alternates between absence of illumination and wide sky-opened natural structures. Those tasks nowadays relies on the usage of expensive and heavy equipments, such as 3D laser scans and entire lighting structures to provide the necessary illumination.

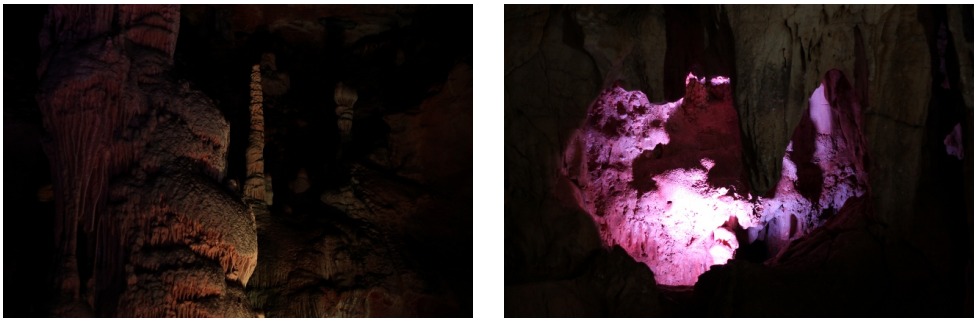


**Figure 1.3.** Images from the inside of a cave, we can see that most of the image is lost due to illumination issues.

For this reason, RGB-D cameras can be of great help for such complex environments. Although the relatively low range of approximately five meters, this low cost device can easily be carried by mobile platforms, or even by humans. Nevertheless, we need a robust keypoint detector algorithm to cope with these complex scenarios, such an algorithm needs to be capable of function in both dark and illuminated environments, as well as take advantage of geometry information to deal with textureless surfaces.

## 1.2 Objectives

We aim at building a novel RGB-D keypoint detection technique combining both intensity and 3D information, capable of handling lack of illumination as well as dealing with substantial noise on both intensity and depth data, while maintaining a good computational efficiency. The detector should also be robust to motion disturbances, such as rotation, scale, and translational motion.

## 1.3 Contributions

The main contribution of this work is the creation and analysis of a keypoint detector called KVD (Keypoints from Visual and Depth data) that efficiently combines intensity and depth data. Besides addressing the properties set in Tuytelaars and Mikolajczyk [2008]:

- repeatability,
- distinctiveness/informativeness,
- locality,
- quantity,

- accuracy and
- efficiency,

the proposed methodology also produces the best performing detector by using both visual and geometrical data, and that still presents a good performance and graceful degradation even in the absence of either one of them.

The results of this work were published at:

- *Vasconcelos, L. O., Nascimento, E. R., and Campos, M. F. M. (2015). A scale invariant keypoint detector based on visual and geometrical cues. In Iberoamerican Congress on Pattern Recognition. IEEE.*

## 1.4 Organization

We have organized this document into the following chapters:

- Chapter 2 - Related Work: This chapter we discuss important works in the literature which is related with ours.
- Chapter 3 - Methodology: A thorough explanation of the method developed through this work is given.
- Chapter 4 - Experiments: Explains the purposes and shows the results of several different experiments performed to access and understand the behaviour of our method.
- Chapter 5 - Conclusion: Closes this document with a conclusion, also discussing future works.

# Chapter 2

# Related Work

Several terms are used in the literature to describe a point of interest within an image which presents high variation in its vicinity and it is stable for affine, rotational, scale and illumination transformation. Hereafter we will refer to such a point of interest as a *keypoint*. We also use the term *invariant* from the literature to characterize keypoint detectors which present stability for certain types of transformations, for example: we say a algorithm is scale invariant if the keypoints detector repeatability rate[1] does not deprecate harshly under scale transformations.

In this section, we summarize several popular keypoint detector methods which relate to our work. They have been organized by their handled data type where extra attention has been given to the methods we thought comparable to our developed technique. Although we compare our method with several state-of-the-art approaches, we address two methods as main concurrent: Harris 6D and ORB (Sections 2.1 and 2.3 respectively), which will be presented and justified along this section.

## 2.1 Intensity 2D Images

The detection of keypoints in images is an essential component in a myriad of applications in pattern recognition and computer vision algorithms. Since the seminal paper of Morevec [1977], where he presented one of the first corner detectors, a large number of keypoint detectors have been proposed. The basic idea of Moravec's work was to select the points that have large intensity variations in specified directions. This variation is measured by centering a rectangular window at the queried point and monitoring the average changes of

[1] repeatability rate is used to evaluate the performance of a keypoint detector under a specific transformation, it will be thoroughly explained at Chapter 4

image intensity $E$ while shifting the window towards a given $(x, y)$ direction:

$$E_{x,y} = \sum_{u,v} w_{u,v}[I(x+u, y+v) - I(u, v)]^2, \tag{2.1}$$

where $w$ specifies the window: it is one within a specified rectangular region, and zero elsewhere. The function $I(u, v)$ stands for the $(u, v)$ pixel intensity. Intuitively, if the point lies in a flat region, then all shifts will result in low intensity variation. If it lies on an edge, moving across the edge will yield low intensity variation. However, if we shift perpendicularly to the edge, a high intensity variation will be observed. Thus, to find corner pixels, one can search for points with high intensity variation in every direction, or, in a more formal way, when the minimum change produced by a shift is larger than a threshold.

### 2.1.1 The Harris Corner Detector

The above described methodology was later improved by Harris and Stephens [1988], where they replaced the discrete shifted windows with the partial derivatives of the Sum of Squared Differences (SSD). The authors noted that Equation 2.1 could be seen as:

$$E(x, y) = (x, y)\mathbf{M}(x, y)^T, \tag{2.2}$$

with the $2 \times 2$ matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{X}^2 \otimes w & (\mathbf{XY}) \otimes w \\ (\mathbf{XY}) \otimes w & \mathbf{Y}^2 \otimes w \end{bmatrix}, \tag{2.3}$$

for:

$$\begin{aligned} \mathbf{X} &= \mathbf{I} \otimes (-1, 0, 1) = \frac{\delta \mathbf{I}}{\delta x}, \\ \mathbf{Y} &= \mathbf{I} \otimes (-1, 0, 1)^T = \frac{\delta \mathbf{I}}{\delta y}, \end{aligned} \tag{2.4}$$

where $\mathbf{I}$ denotes the intensity image and $\otimes$ is the convolution operator. Intuitively, this expands Moravec's work to consider small shifts in every possible direction, a thorough explanation of this fact is shown in Harris and Stephens [1988].

Analysing the matrix $\mathbf{M}$ as an auto-correlation matrix, its eigenvalues and eigenvectors play an important role in estimating whether the target point is a corner. Intuitively, the eigenvectors point toward the directions of most significant intensity variations (maximum and minimum). Thus, similarly to Moravec's reasoning, a corner can be detected if the smallest eigenvalue is above a given threshold.

In order to quantify this cornerness property, the authors tailored a response function

upon the eigenvalues of the correlation matrix $\mathbf{M}$. To avoid the explicit eigenvalue decomposition, which is an expensive operation, the response function $R$ is given as:

$$R = Det(\mathbf{M}) - k \times Tr(\mathbf{M}), \tag{2.5}$$

where $Det(\cdot)$ and $Tr(\cdot)$ stands for the Determinant and Trace of the matrix $\mathbf{M}$ respectively.

The Harris corner detector, as this approach is known in the literature, comprehends an intuitively and powerful methodology for keypoints detection, since the method has several desirable properties such as: rotational invariance, robustness to illumination changes, and easily extendable for different data types, as we will present later. The main drawbacks of Harris-like detectors is that they demand relatively high computational efforts to calculate the correlation matrix, exhausting it for real time applications.

The Harris detector searches for keypoints in only one scale (defined by the window's size), which makes the method sensible to scale transformations. In order to tackle this issue, a modified version of the Harris detector, known as Harris-Laplacian, was proposed by Mikolajczyk and Schmid [2004], the authors exploited the response function of the Harris detector to make it invariant to scale changes. To achieve this, the Harris cornerness response is accessed in a set of different scales for each queried point. The automatic scale selection is performed by following the methodology proposed by Lindeberg [1998], where the correct scale is chosen according to the local maximum of the cornerness response within the scale space. Even though this new version presents some invariance to changes in scale, it returns low distinctive keypoints and considerably increases the computational time.

### 2.1.2 The SIFT Detector

One of the most popular modern algorithm that detect distinctive, small illumination changes as well as camera viewpoint changes is the SIFT algorithm Lowe. [2004]. This technique extracts features using local gradients and estimates a characteristic orientation and the scale of the keypoint's neighborhood to provide rotational and scale invariance.

The algorithm firstly builds a scale pyramid by repeatedly applying Gaussian filters and sub-sampling the original image. Once the pyramid is built, the authors approximate the scale-normalized Laplacian of Gaussian filter by performing Difference of Gaussians (DoG) among subsequent pyramid levels. To select one point as a keypoint candidate, the point needs to be an extrema (maximum or minimum) among its neighbouring pixels, both in the image and scale space.

Later, the keypoint candidates are tested for high contrast and cornerness. For the contrast test, they performed the fitting of a 3D quadratic function centered at the keypoint

candidate, and depending on the quality of the fit, the point is rejected. For the cornerness test, the keypoint is tested similarly to the Harris cornerness function response. The main drawback of this algorithm is the high need of computational resources.

### 2.1.3 Machine Learning Techniques

A recent approach that has become popular is based on machine learning techniques. Dias et al. [1995] deployed a Neural Network based Sub-Image Classifier (NNSIC) to detect corners. Their algorithm receives an $8 \times 8$ window from the input image and outputs a binary image indicating the presence or absence of a corner. One of the key contributions of using an artificial neural network was to show the capability of this approach in providing a way to compute a generalized representation for a corner.

Following the machine learning approach, Rosten and Drummond [2005] proposed the Features from Accelerated Segment Test (FAST) detector, an interest point extractor from intensity images, which considers a circle around the queried point and analyses the intensity differences between the pixels at the arc and the central point to decide whether the queried point is a keypoint or not. In Rosten and Drummond [2006], the authors extended its previous work by training a decision tree, which takes the intensity differences as features, to learn the FAST detector, improving severely the efficiency of the method. Finally, in Rosten et al. [2010] was applied simulated annealing to optimize the decision tree. The search is performed by randomly modifying an initially trained decision tree and monitoring how well the modified tree behaves concerning repeatability and efficiency.

Although FAST algorithm was designed to be a low-cost algorithm, it shows reliable results concerning repeatability under illumination change, noise corruption and translational motion. As drawbacks, the method shows sensitivity regarding scale transformation and rotational motion. The method also lacks a reliable keypoint response function, relying on a simple average of the pixels lying at the ring.

In order to cope with the above mentioned issues, the work of Rublee et al. [2011] presented a novel detector called Oriented FAST and Rotated BRIEF (ORB) that builds upon FAST's methodology. To deal with the scale sensitivity problem, the authors deployed the method suggested by Lindeberg [1998], adopting the Harris cornerness response instead of FAST's response. A scale pyramid is employed and the correct scale is chosen as the one maximizing the Harris response within the pyramid space.

The scale pyramid is carried out by subsequently smoothing and sub-sampling the image. For each pyramid level, the authors ran the FAST keypoint detector and evaluate them using the Harris response. For each detected keypoint, the chosen scale will then be the one maximizing the response through the scale space (pyramid).

To account for rotational invariance, the authors were inspired by Rosin [1999]. They defined the corner orientation based on intensity centroids. The moments of a given patch are defined as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y). \tag{2.6}$$

The centroid is then calculated as:

$$c = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \tag{2.7}$$

The corner orientation is then given by the vector $\vec{\mathbf{oc}}$. Where $\mathbf{o}$ stands for the corner's center. The corner orientation is now calculated as:

$$\theta = atan2(m_{01}, m_{10}), \tag{2.8}$$

where atan2 is the quadrant-aware version of arctan.

The ORB detector is a state-of-the-art algorithm, albeit designed only to 2D intensity images, it is quite robust being invariant to rotation, illumination changes and scale transformation, while maintaining a low computational cost. Due to this robustness we chose this algorithm for a more detailed comparison with our method.

A similar approach to ours is carried by Hasegawa et al. [2014], where the authors enhanced the FAST methodology implementing a cascade of decision trees to classify a target point as a corner. The idea is to avoid badly classified corners through analysing a bigger vicinity and checking the corner consistency among the consecutive pixels.

The region analysed consists of three concentric Bresenham's circles of growing radii ($r_1 = 3, r_2 = 4 and r_3 = 5$). For each radius, a decision tree is trained, just like the FAST methodology. A point is classified as a keypoint if considered as a true keypoint by all three decision trees and if the corner orientation matches in all analysed circles.

The corner orientation for each analysed Bresenham's circle is calculated based on the sequence of all brighter or all darker consecutive pixels as shown in Figure 2.1.3. A positive keypoint is considered if, besides being classified by all decision trees as positive, all those orientation directions point approximately toward the same direction. The algorithm is then applied at a scale pyramid to add scale invariance.

## 2.2 Point Clouds and Range Images

Extracting data from images can usually provide rich information on the object features, but geometrical information produced by 3D sensors based on structured lighting or time of flight is less sensitive to visible light conditions. Three-dimensional data has been successfully exploited by algorithms such as NARF Steder et al. [2011], which has been proposed to extract features from 3D point cloud data for object recognition and pose estimation. NARF's approach is based on a set of points classified as stable, and the explicit use of border information.

The benefits of using three-dimensional data have led to the proposal of several 3D detectors implementations derived from 2D approaches, some examples of these are SIFT3D and HARRIS3D Rusu and Cousins [2011]. In general, these three-dimensional variations consist in replacing gradient images by surface normals. These methodologies are useful mainly for unordered 3D point clouds.

A machine learning based method for keypoint extraction from depth maps is detailed at Holzer et al. [2012]. The authors make use of a random forest to approximate a specially tailored response function which takes advantage of the curvature and the repeatability of each point. Each tree of the forest implements several queries concerning the depth comparison between different points in a given vicinity centred at the queried point. The response function is constructed by combining the curvature response and the repeatability of each point, such repeatability is computed by the reconstruction of the training set images sequence.

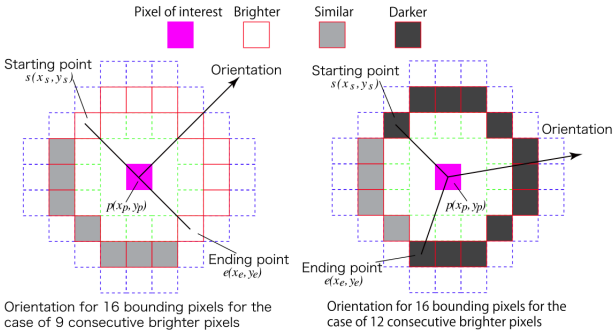



**Figure 2.1.** Figure extracted from Hasegawa et al. [2014] representing the considered vicinity and the computation of the keypoint orientation.

## 2.3 Multiple Cues

### 2.3.1 Descriptors

The use of multiple cues, such as visual and geometric features, is growing popular among descriptor techniques. Kanezaki et al. Kanezaki et al. [2011] combined depth and visual into a global descriptor, called VOSCH, in order to increase the recognition rate. Local descriptors such as BRAND and BASE Nascimento et al. [2013], Color SHOT Tombari et al. [2011] and MeshHOG Zaharescu et al. [2009] also combines the visual information to improve matching quality.

### 2.3.2 Keypoint Detectors

A keypoint detector that goes in the same fashion, making use of both visual and geometric informations, is the Harris 6D. As far as we know, it is an unpublished work that is implemented by Rusu and Cousins [2011]. The Harris 6D methodology follows the same reasoning of the original Harris corner detector, the extension relies on the construction of the correlation matrix, which takes into consideration both RGB (red, green and blue) and 3D Euclidian spaces. Each instance on the RGB space stands for an image pixel color, while the 3D Euclidian space is crowded by the normal vectors of each 3D point composing the point cloud. This detector inherits all properties from the Harris corner detector and adds illumination independence, since it operates also on the point cloud, which is independent from the intensity image. It is a very stable detector in spite of its sensitivity to scale changes. The Harris 6D merges both visual and geometry information and is a commonly used keypoint detector. For those reasons we select this method as the main competitor to our proposed approach.

## 2.4 Summary

The previously presented information is summarized at Table 2.4

In this work, we follow a similar approach to Harris 6D for the keypoint detection problem. Our technique leads to the synthesizing of a detector which simultaneously consider both visual and geometrical information to classify points as a keypoint. This detector excels both in its low computational cost and its high repeatability rate with no loss in performance of the other desirable features set out by properties defined in Tuytelaars and Mikolajczyk [2008].

| Method | Scale inv. | Rotation inv. | Illumination | Intensity | Geometry | Resources |
|---|---|---|---|---|---|---|
| Harris 2D | | • | • | • | | M |
| SIFT | • | • | • | • | | H |
| FAST | | | • | • | | L |
| ORB | • | • | • | • | | L |
| SIFT 3D | • | • | • | | • | H |
| HARRIS 3D | | • | | | • | M |
| NARF | | • | | | • | M |
| HARRIS 6D | | • | • | • | • | M |
| PROPOSED METHOD | • | | • | • | • | M |

**Table 2.1.** A summary of the presented information, the symbols filling the right most column stands for High, Medium and Low for resource consumptions.

# Chapter 3

# Methodology

In this chapter we detail the methodology of the proposed KVD keypoint detector, which is based on a machine learning approach for keypoint detection, similar to other works in the literature (Rosten et al. [2010]; Rublee et al. [2011]). However, differently from their work, we assume that both 3D and visual data are available to the detection process, which enables our detector to work even in the absence of image data.

The input of our algorithm is a data pair $(\mathbf{I}, \mathbf{D})$, which denotes the output of a typical RGB-D device, where $\mathbf{I}$ and $\mathbf{D}$ are the intensity and the depth matrices, respectively. Let $\mathbf{x} = (i, j)$ denotes a pixel's coordinates, $I(\mathbf{x})$ the pixel's intensity, $D(\mathbf{x})$ is the depth for that pixel, $P(\mathbf{x})$ is the corresponding 3D point, and $N(\mathbf{x})$ is its normal vector.

As stated, our technique is built upon a supervised learning approach, with a training step where a decision tree is created. This decision tree plays a key role in the detection procedure, since it is used to classify points into keypoints. There are four steps in this classification process: the surface's normal estimation, the feature vector extraction, the model training, and the non-maximal suppression. An overview of the process can be seen in Figure 3.1.

## 3.1 Surface's Normal Estimation

To calculate an approximation of the surface's normal for each point, we need to analyse the relationship of the target point with nearby points from its vicinity. Once the surface's normal vector is estimated we can evaluate geometrical information of the surface in local patches. This information is leveraged to improve the quality of the selected keypoints.

There are several methods to estimate the normal vectors from a point cloud [Klasing et al., 2009]. One accurate approach consists in performing Principal Component Analysis (PCA) on the $3 \times 3$ covariance matrix $\mathbf{C}$ of a small vicinity $\mathcal{K}$ around the target point.

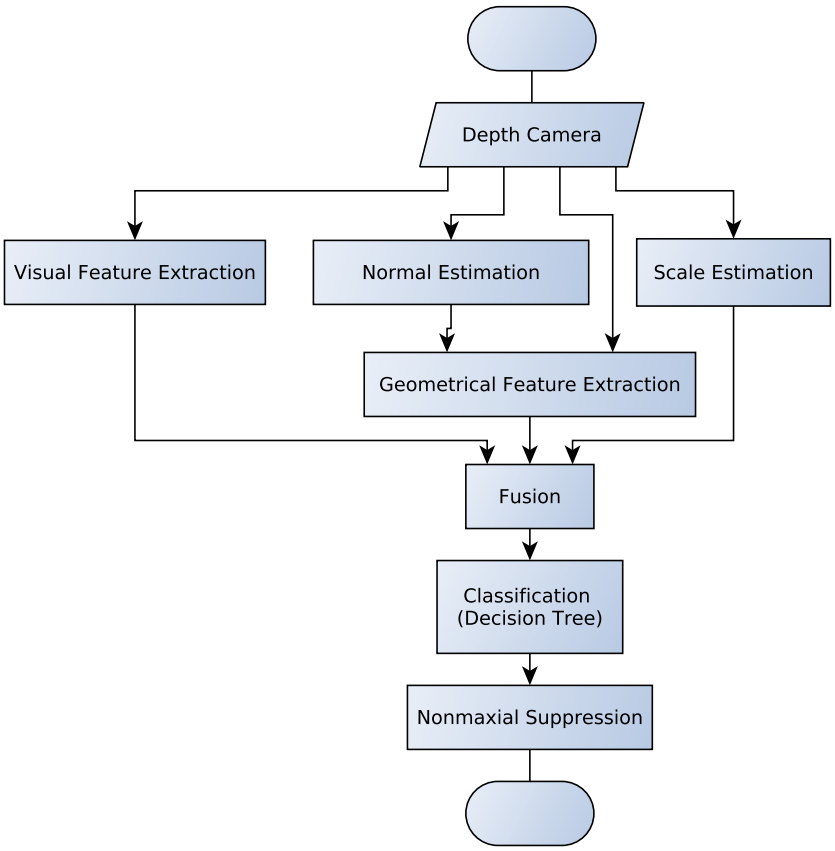



**Figure 3.1.** The algorithm's flow chart. After estimating a scale and extracting both visual and geometric features, the algorithm assembles this information in a final feature vector which is further classified by a decision tree into keypoints candidates or not. Finally the algorithm performs nonmaximal suppression to avoid multiple detections in a small patch.

The vicinity $\mathcal{K}$ plays an important role on this process. The analysed vicinity should be large enough to cope with noisy points, and small enough to represent the plane within the required details. Thus, the sensor's resolution must be taken into account. As a golden rule, we should pick the vicinity just large enough to handle the necessary quality of details for the application. The covariance matrix $\mathbf{C}$ is computed as:

$$\mathcal{C} = \frac{1}{|\mathcal{K}|} \sum_{\mathbf{p}_i \in \mathcal{K}} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T, \tag{3.1}$$

where $\bar{\mathbf{p}}$ is computed as $\bar{\mathbf{p}} = \frac{1}{|\mathcal{K}|} \sum_{\mathbf{p}_i \in \mathcal{K}} \mathbf{p_i}$. We take the eigenvector that is assigned to the smallest eigenvalue of $\mathbf{C}$ as the surface's normals estimation. This vector represents the direction towards the data that has the lowest variance. For example, in a planar surface with low noise the variance in the perpendicular direction tends to be zero. Thus this process can

be seen as a plane fitting among the neighbouring points.

The computation of the neighbouring points is of extreme importance for the method's accuracy. Since poorly selected points may not properly represent the targeted plane. An accurate solution is to select the vicinity as the points lying within a sphere of radius $k$, centered at the queried point. Although this provides good estimates for the normal vectors, the search for neighbouring points in non-indexed point clouds is an computationally expensive operation, even when using of spacial data structures such as octrees, kd-trees, etc.

In order to speed the vicinity search, it is possible to exploit the matrix structure of a depth image and represent the vicinity as the pixels lying within a rectangular region centered at the queried point. This method, although much faster, results in estimated normal vectors of lower quality.

Both methods are implemented in Rusu and Cousins [2011]. We performed experiments using both normal estimation methods, which can be found in Chapter 4.

## 3.2 Feature Vector Extraction

The first step of the detection process is to create a feature vector for every point, which will be fed into a decision tree for classification. Figure 3.2 shows a representation of the feature vector construction.

To decide if a given pixel is a keypoint, we analyze a circular vicinity of the given point in increasing radii, as depicted in Figure 3.2. The figure refers to the radii set $\mathcal{S} = \{3, 5, 7, 9\}$, which is the settings used in our experiments. It is worth noting that the algorithm is defined for any arbitrary set $\mathcal{S}$.

Given an image pixel's coordinates $\mathbf{c} \in \mathbb{R}^2$, we consider its vicinity as the image patches containing the circles centred at $\mathbf{c}$ with radii varying in $r \in \mathcal{S}$. Each circle is defined by the function $B(r, \mathbf{c})$ which we denote as $B_r(\mathbf{c})$:

$$B_r(\mathbf{c}) : \mathbb{R}^3 \rightarrow \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}. \tag{3.2}$$

The $B_r(\mathbf{c})$ map function outputs all pixels $\mathbf{p_i}$ lying within the Bresenham's circle [Pitteway, 1967] with radius equal to $r$. Thus, the total vicinity considered consists of the concatenation of all vectors $B_r(\mathbf{c}), \forall r \in \mathcal{S}$. We define the vicinity of a central pixel $\mathbf{c}$ as

$$\mathcal{V}_c = \{B_{r_1}(\mathbf{c}), B_{r_2}(\mathbf{c}), \ldots, B_{r_{|\mathcal{S}|}}(\mathbf{c})\}, \quad \forall r_i \in \mathcal{S}. \tag{3.3}$$

For instance, in our implementation we use four fixed radii of sizes $\mathcal{S} = \{3, 5, 7, 9\}$. Then, the concatenation of all map functions yields a total vicinity of $|\mathcal{V}| = 136$ pixels to be

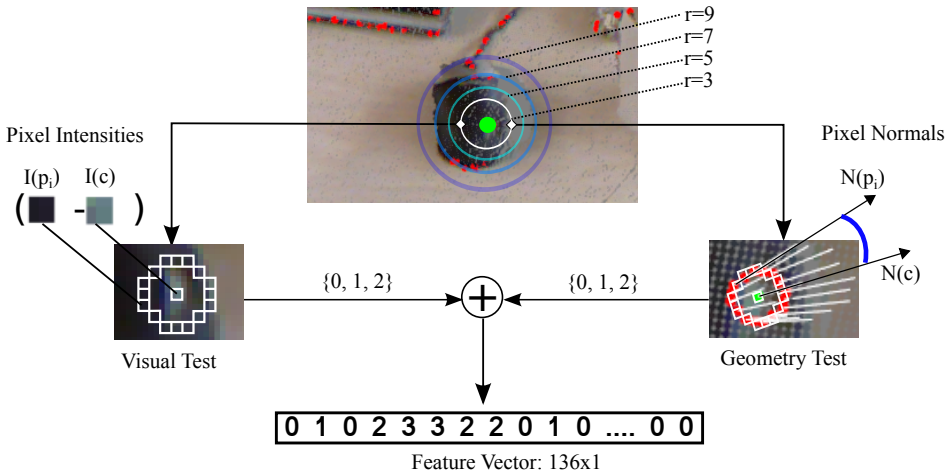



**Figure 3.2.** Visual and geometrical feature extraction for a keypoint. The highlighted squares correspond the Bresenham's circle.

analysed, consisting of the circles computed by $B_3$,$B_5$,$B_7$,$B_9$, each with 16, 28, 40 and 52 pixels. Figure 3.2 shows the four Bresenham's circles. The circle with radius 3 is denoted by red squares. For each vicinity element $\mathbf{p} \in \mathcal{V}_c$, we compute visual and geometric features.

### 3.2.1 Feature Computation

The visual features are computed based on simple intensity difference tests among a carefully chosen vicinity, reducing the memory consumption and processing time. For each pixel $\mathbf{p} \in \mathcal{V}_c$ we evaluate

$$\tau_v(\mathbf{c}, \mathbf{p}) = \begin{cases} 2, & \text{if } I(\mathbf{p}) - I(\mathbf{c}) < -t_v \\ 1, & \text{if } I(\mathbf{p}) - I(\mathbf{c}) \geq t_v \\ 0, & \text{otherwise,} \end{cases} \tag{3.4}$$

where $t_v$ represents a toleration threshold ($t_v = 20$ in our settings). This function analyses the intensity relationship between both pixels $\mathbf{c}$ and $\mathbf{p}$, encoding whether the pixel intensity $I(\mathbf{p})$ is darker, lighter or at similar intensity regarding the pixel intensity $I(\mathbf{c})$, respectively.

The visual feature extraction implemented by the function $\tau_v$ is similar to the one described in Rosten et al. [2010], however we embed geometric cues into our feature vector to increase robustness to illumination changes and to the lack of texture in the scenes.

Geometric feature extraction is performed by the $\tau_g(.)$ function. This process is depicted in Figure 3.2. It is based on two invariant geometric measurements:

- the normal displacement, and

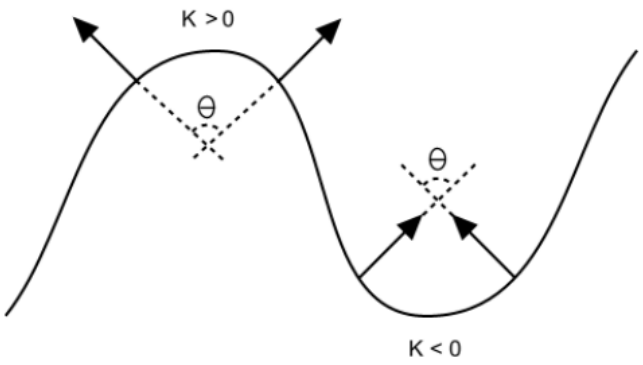



**Figure 3.3.** The dot product between surface normals is ambiguous, but they have opposite curvature signs, which is captured by the function $\kappa(\cdot)$.

- the surface's convexity.

While the normal displacement test is performed to check whether the dot product between the normals $N(\mathbf{c})$ and $N(\mathbf{x_i})$ is smaller than a given displacement threshold $t_g$, the convexity test is accomplished by the local curvature indicator, $\kappa$, estimated as:

$$\kappa(\mathbf{c}, \mathbf{p}_i) = \langle P(\mathbf{c}) - P(\mathbf{p}_i), N(\mathbf{c}) - N(\mathbf{p}_i) \rangle, \tag{3.5}$$

where $\langle . \rangle$ is the dot product, and $P(\mathbf{c})$ is the $3D$ spatial point associated with pixel $\mathbf{p}$ and depth $D(\mathbf{p})$. The $\kappa$ function is used to capture the convexity of geometric features and also to unambiguously characterize the dot product between surface normals. The sign of the function $\kappa$ denotes whether the surface between the two points accessed is convex (positive signed) or concave (negative signed), as shown in Figure 3.3.

Thus, the geometrical features are computed by analysing the behaviour of the surface between two points:

$$\tau_g(\mathbf{c}, \mathbf{p}_i) = \begin{cases} 2, & \text{if } \langle N(\mathbf{p}_i), N(\mathbf{c}) \rangle < t_g \wedge \kappa(\mathbf{c}, \mathbf{p}_i) > 0 \\ 1, & \text{if } \langle N(\mathbf{p}_i), N(\mathbf{c}) \rangle < t_g \wedge \kappa(\mathbf{c}, \mathbf{p}_i) < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Intuitively, it encodes whether the surface between the central pixel $\mathbf{c}$ and the queried point $\mathbf{p_i}$ is convex, concave or plane, respectively. We consider a plane if $\langle N(\mathbf{p}_i), N(\mathbf{c}) \rangle \geq t_g$. With $t_g = 0.97$ in our settings.

### 3.2.2 Scale Invariance

Scale invariance is endowed to our detector by taking advantage of the geometry information available, in order to weight the Bresenham's circles of different radii. We analyze the geometrical vicinity encompassed by each Bresenham's circle $B_r(\mathbf{c})$ in the 3D scene, by computing the minimum Euclidean distance among all vectors $\mathbf{v} = P(\mathbf{c}) - P(\mathbf{p_i})$, with $\mathbf{p_i} \in B_r(\mathbf{c})$:

$$d_r = \min_{\mathbf{p_i}} \|P(\mathbf{c}) - P(\mathbf{p_i})\|, \tag{3.7}$$

where $P(\mathbf{c})$ and $P(\mathbf{p_i})$ are the 3D points corresponding to the central pixel $\mathbf{c}$ and the pixels composing the Bresenham's circle $\mathbf{p}_i \in B_r(\mathbf{c})$.

The minimal distance $d_r$ is taken as an estimative of the circle's radius in the 3D scene. It is then weighted by the unidimensional-Gaussian function, in order to penalize circles which its estimated radii in the 3D scene are distant from $\mu = 0.02$ meters, a predefined radius which seems as a reasonable value for our application.

$$\mathbf{w_r} = \exp\left(-\frac{(\mathbf{u} - \mathbf{d_r})^2}{2\sigma^2}\right), \tag{3.8}$$

the standard deviation $\sigma = 0.011$ was empirically chosen, targeting a sufficiently narrow shape around the mean value. This curve can be seen in Figure 3.4.

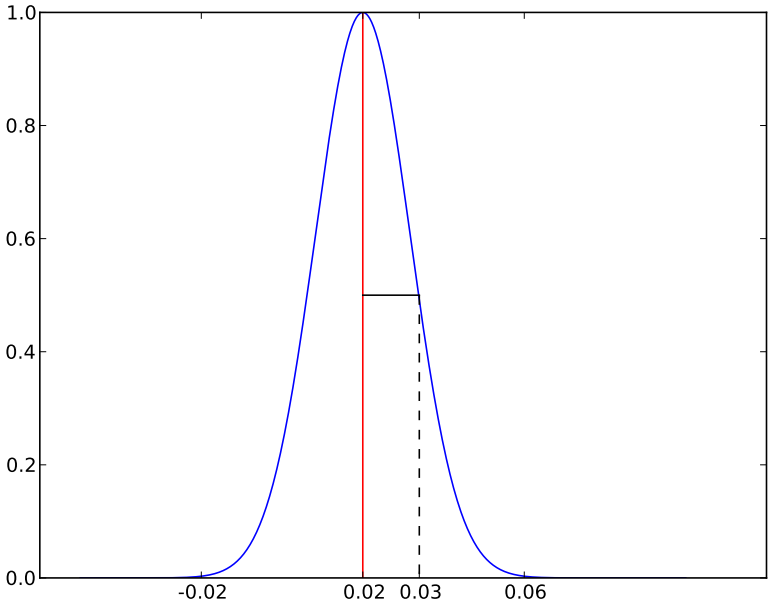



**Figure 3.4.** Gaussian curve penalizing 3D scene radii lying further from the mean $\mu = 0.02$.

The weighting procedure avoids the addition of noise by circles covering non-interesting areas, e.g. points too far from the camera, larger circles might have to be strongly penalized.

### 3.2.3 Final Feature Vector

In order to combine the geometric and visual information in one final feature vector, we deployed two different approaches:

#### 3.2.3.1 The Additive Approach

This method combines both cues by adding both visual and geometric feature vectors. We extract a feature vector from a Bresenham's circle of radius $r$ centered at $\mathbf{c}$ as a row vector $\vec{\mathbf{v}}_r = \begin{bmatrix} f_1 & f_2 & \dots & f_{|B_r(c)|} \end{bmatrix}$ where each $f_i \in \vec{\mathbf{v}}_r$ is given by:

$$f_i(\mathbf{c}, r) = w_r * (\tau_v(\mathbf{c}, \mathbf{p_{i,r}}) + \tau_g(\mathbf{c}, \mathbf{p_{i,r}})), \tag{3.9}$$

where $\mathbf{p_{i,r}}$ is the $i$th element of the respective Bresenham's circle $B_r(\mathbf{c})$.

The final feature vector $\vec{\mathbf{F}}$ is generate by concatenating all the feature vectors $\mathbf{v_r}$ as:

$$\vec{\mathbf{F}} = \begin{bmatrix} \vec{\mathbf{v}}_{r_1} & \vec{\mathbf{v}}_{r_2} & \cdots & \vec{\mathbf{v}}_{r_{|\mathcal{S}|}} \end{bmatrix} \quad \forall r \in \mathcal{S}. \tag{3.10}$$

#### 3.2.3.2 The Concatenation Approach

To combine both informations in this approach, we concatenate two vectors, where one encodes visual features and the other geometrical features. We will depict the visual vector $\vec{\mathbf{F}}_v$ construction, while the geometric vector $\vec{\mathbf{F}}_g$ follows a similar process.

For each Bresenham's circle, a row feature vector $\vec{\mathbf{v}}_r = \begin{bmatrix} f_1 & f_2 & \dots & f_{|B_r(c)|} \end{bmatrix}$, is computed where each $f_i \in \vec{\mathbf{v}}_r$ is calculated as:

$$f_i(\mathbf{c}, r) = w_r * \tau_v(\mathbf{c}, \mathbf{p_{i,r}}), \tag{3.11}$$

then, the visual feature vector $\vec{\mathcal{F}}_v$ is defined as:

$$\vec{\mathbf{F}}_v = \begin{bmatrix} \vec{\mathbf{v}}_{r_1} & \vec{\mathbf{v}}_{r_2} & \cdots & \vec{\mathbf{v}}_{r_{|\mathcal{S}|}} \end{bmatrix} \quad \forall r \in \mathcal{S}. \tag{3.12}$$

The geometric feature vector $\vec{\mathbf{F}}_g$ follows the same logic, but applying the function $\tau_g(\cdot)$ instead of $\tau_v(\cdot)$ in Equation 3.11. Finally, the final feature vector $\vec{\mathbf{F}}$ consists of the concatenation of both visual and geometric vectors:

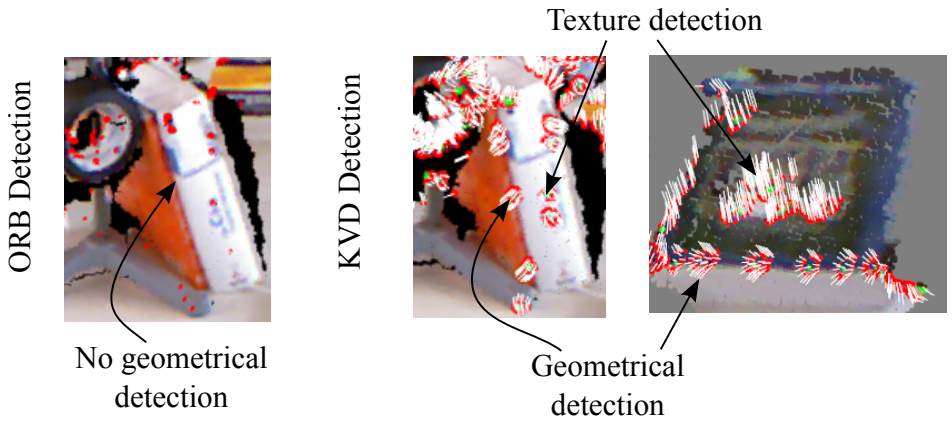



**Figure 3.5.** Comparison between keypoints detected by our methodology (right images) and ORB algorithm (left image). One can note the absence of keypoints around the border of the book in the right image. Since KVD captures both visual and geometrical features to find keypoints it is capable to detect such keypoints.

$$\vec{\mathbf{F}} = \left[\vec{\mathbf{F}}_v, \vec{\mathbf{F}}_g\right]. \tag{3.13}$$

### 3.2.4 Fusion Processes Discussion

The concatenation approach yields a final feature vector of length equals to $2|\mathcal{V}|$. From the one hand, this approach maintain all information extracted during the feature extraction process, on the other hand, the final feature vector is twice as long as the final vector of the additive approach, which impacts on the computational time performance. We performed experiments to compare both approaches which is shown in Chapter 4.

Due to the fusion process, our method is able to detect keypoints using both visual and geometrical data. Figure 3.5 depicts detected keypoints exploiting the fused information.

## 3.3 Decision Tree Training

The decision tree is a fundamental part of the algorithm, it is used to decide whether a target point should be considered as a keypoint candidate or not. In order to do that, we select a sample of training points. This sample contains both keypoints and non-keypoints examples properly labeled. Further, we use this set as a keypoint-model to train a decision tree for predicting whether an addressed point is a keypoint.

The decision tree algorithm was chosen because, differently from other classical machine learning algorithms, the decision tree can be straightforwardly converted into a hard-coded simple if-then-else algorithm to speed the computational time performance.

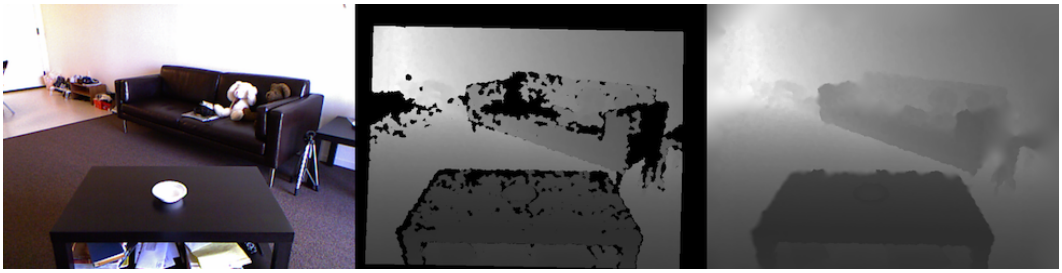


**Figure 3.6.** Example of images from the RGB-D Berkeley 3-D Object Dataset (B3DO), where the right images shows the smoothed depth map. Figure taken from http://kinectdata.com/

**Table 3.1.** Confusion matrix of keypoint classification based on a decision tree.

| | Keypoint | Non-Keypoint |
|---|---|---|
| Keypoint | 0.89 | 0.11 |
| Non-Keypoint | 0.08 | 0.92 |

In the training step, we created a training set by extracting a total of $160,144$ points from the RGB-D Berkeley 3-D Object Dataset (B3DO) Janoch et al. [2011]. This dataset is publicly available[1] and it is composed of a large number of real world scenes with several different objects, geometry and visual data. The images were captured with a Kinect™sensor at a resolution of $640 \times 480$ pixels and all the depth maps had been smoothed. An example of the dataset is shown in Figure 3.6.

## 3.4 Non-Maximal Suppression

The last step in our methodology is to perform a non-maximal suppression. This step avoids the algorithm from finding redundant keypoints. To achieve this we rank the keypoint candidates (classified as positive by the decision tree) lying within a small image patch. Selecting as the true keypoint the candidate with highest score among the patch.

Although the visual and the geometrical tests do not provide a response value, it is still possible to perform a non-maximal suppression based on the values of the features. In order to do so we need to define the set

$$\mathcal{X}_{rc_k} = \{\mathbf{p_i} : \ \mathbf{p_i} \in B_r(c) \ \wedge \ (\tau_v(\mathbf{c}, \mathbf{p_i}) = k \ \vee \ \tau_g(\mathbf{c}, \mathbf{p_i}) = k)\}, \tag{3.14}$$

which contains all the pixels within a Bresenham's circle $B_r(\mathbf{c})$ whose geometry or visual

[1] http://kinectdata.com

feature has value $k$.

For each radius $r \in \mathcal{S}$, we calculate the partial response $\mathrm{R_p}(\mathbf{c}, r)$ as:

$$\mathrm{R_p}(\mathbf{c}, r) = \max_{\mathcal{X} \in \{\mathcal{X}_{rc_1}, \mathcal{X}_{rc_2}\}} \left( \frac{1}{|\mathcal{X}|} \sum_{\mathbf{x_i} \in \mathcal{X}} D_v(\mathbf{c}, \mathbf{x_i}) + \lambda D_g(\mathbf{c}, \mathbf{x_i}) \right), \tag{3.15}$$

where

$$\begin{aligned} D_v(\mathbf{c}, \mathbf{x}) &= |I(\mathbf{x}) - I(\mathbf{c})| \\ D_g(\mathbf{c}, \mathbf{x}) &= 1 - \langle N(\mathbf{x}), N(\mathbf{c}) \rangle \end{aligned} \tag{3.16}$$

give the visual and geometrical responses respectively. The factor $\lambda$ is used to define the contribution of the geometrical information into the response.

The final response is then defined as the maximum response among all radii:

$$\mathrm{R_f}(\mathbf{c}) = \max_{r} \big( \mathrm{R_p}(\mathbf{c}, r) \big), \forall r \in \mathcal{S}. \tag{3.17}$$

Equation 3.17 uses both the absolute difference between intensities and the normal surface angles for the pixels in the contiguous arc of the Bresenham's circle as well as the keypoint candidate to rank the maximal points.

Next, we divide the image into smaller patches with size $w \times w$ (in this work we use $w = 5$). For each patch we select the pixel with the largest final response.

# Chapter 4

# Experiments

This chapter details how we evaluate and compare our approach to other literature from the field. We performed several experiments to evaluate the behaviour of our detector. In order to analyse its repeatability, distinctiveness, robustness and time performance we compared our approach against standard ones for two-dimensional images, SIFT Lowe. [2004], ORB Rublee et al. [2011], and SIFT3D (using all three color channels), for geometric data Harris3D (a 3D version of Harris corner detector) and NARF Steder et al. [2011], and the Harris6D, which similarly to our methodology, uses both visual and geometrical data to detect keypoints. The implementation of the later three methods can be found in Rusu and Cousins [2011].

## 4.1 Dataset

We performed experiments with two different datasets. We used the RGB-D SLAM Dataset and the Benchmark presented in Sturm et al. [2011] to evaluate the behaviour of the methods regarding image changes in translation, scale, and rotation in both the image (roll) and horizontal (yaw) planes. This dataset is publicly available[1] and contains several real world sequences of RGB-D data captured with a Kinect™sensor. Images were acquired at a frame rate of $30$ Hz and a resolution of $640 \times 480$ pixels. Each sequence in the dataset provides the ground truth for the camera pose estimated by a motion capture system, which allows the computation of the homographies relating each image pair by a plane projective transformation.

We separated a different sequence of images for each camera motion type, as can be seen in Figure 4.1. The translational changes Figure 4.1 (a) were produced by shifting the

[1] https://cvpr.in.tum.de/data/datasets/rgbd-dataset

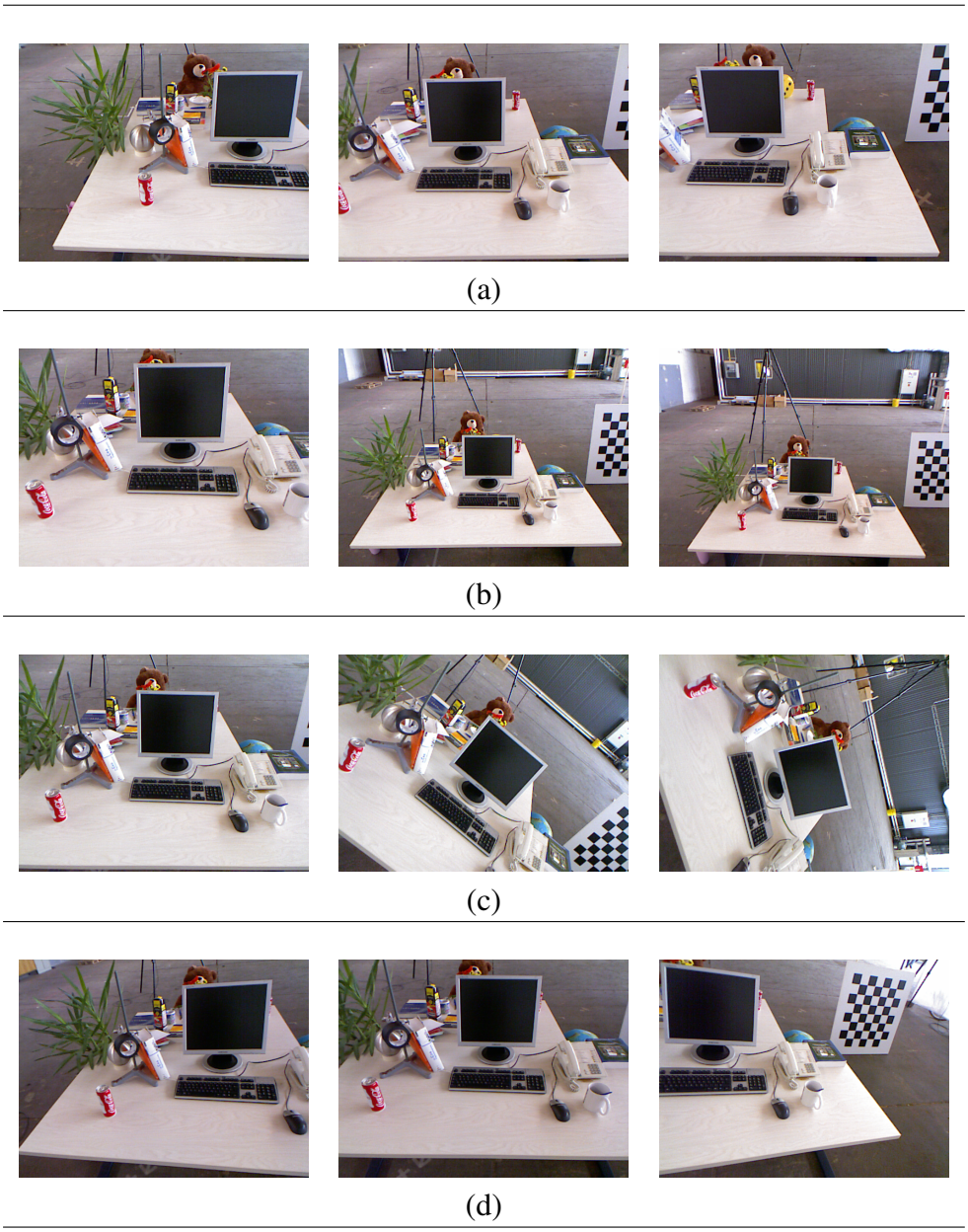



**Figure 4.1.** Samples of the image sequences of the different transformation changes, where (a) is the translational , (b) is the scale, (c) is the rotation in the image plane, and (d) is the rotation in the horizontal plane.

sensor along the horizontal axis with an maximum offset of $0,75$ meters. The scale changes (b) were acquired by moving the camera away from the scene where the maximum shift is approximately $0,40$ meters. The image rotations (c) and (d) vary from $0,5$ to $180$ degrees for the roll axis, and $0,5$ to $35$ degrees around the yaw axis.

In order to test the algorithms for illumination changes, we built a dataset by capturing a total of 104 images of a cluttered room starting at dusk (partial illumination) at an interval

**Figure 4.2.** Samples of the illumination sequence.

of one minute between acquisitions. The images were captured using a Kinect™sensor with the resolution setted to $640 \times 480$ pixels standing at a fixed position, as shown in Figure 4.2. We will furthermore refer to this dataset as *Verlab_Dusk*.

## 4.2 Decision Tree Training

For the Decision tree training, we assigned $66\%$ of the points from the training set and the remaining points $(54, 449)$ to test the quality of the final decision tree. Both sets were equally divided into positive and negative samples. In order to define the threshold of the curvature for a positive keypoint, we computed the curvature of several, manually selected, positive keypoints. We have found the value of $0.09$ based on the average of these curvatures. Thus, all the points with curvature larger than $0.09$ were labeled as positive samples for the keypoint class. To take visual features into account, we also add keypoints detected by ORB as positive examples. The final confusion matrix is shown in Table 3.1.

## 4.3 Evaluation and Criteria

We evaluate and compare our method to others from literature regarding three concepts: Robustness, Distinctiveness and Time performance. In order to assess the Robustness of the methods, we use the Repeatability criterion. In the graphics, both KVD and Concatenation represents our methods which respectively implements the additive and concatenation fusion approaches.

### 4.3.1 Repeatability

One of the most important properties of a keypoint detector is its ability to find the same set of keypoints on images acquired of a scene from different view points. Thus, to evaluate and compare the robustness of KVD detector with other approaches, we applied the same criterion used by Mikolajczyk et al. [2005].

The repeatability score for a pair of images is computed as the ratio between corresponding regions of the keypoints in two frames and the smaller number of keypoints in these frames. To calculate the number of corresponding regions between frames $i$ and $i+\Delta$, let $\mathcal{K}_i$ be the set of keypoints found in image $i$. To compute the correspondence we define

$$\mathcal{K}^G_{i\rightarrow(i+\Delta)} = \{\mathbf{H}_{i\rightarrow(i+\Delta)}\mathbf{k}_i; \forall \mathbf{k}_i \in \mathcal{K}_i\} \tag{4.1}$$

as the ground-truth set which represents the projection of the keypoints $\mathbf{k}_i \in \mathcal{K}_i$ (keypoints found in frame $i$) onto the frame $i+\Delta$, where $\mathbf{H}$ is the homography matrix relating both frames.

Thus, for each keypoint $\mathbf{k}^G_i \in \mathcal{K}^G_{i\rightarrow(i+\Delta)}$ we define an elliptic region $R_{\mathbf{k}^G_i}$ centered at $\mathbf{k}^G_i$ with radius proportional to its normalized scale (we refer the reader to Mikolajczyk et al. [2005]). A correspondence is found if the overlap error between $R_{\mathbf{k}^G_i}$ and $R_{\mathbf{k}_{i+\Delta}}$ is sufficiently small:

$$1 - \frac{R_{\mathbf{k}^G_i} \cap R_{\mathbf{k}_{i+\Delta}}}{R_{\mathbf{k}^G_i} \cup R_{\mathbf{k}_{i+\Delta}}} < \epsilon,$$

where $R_{\mathbf{k}_{i+\Delta}}$ is the elliptic region centered at location of keypoint $\mathbf{k}$ in the frame $i+\Delta$. In our experiments, $\epsilon = 0,4$.

An important factor to be considered is the coverage area, which is the area covered by all keypoints found. For a fair comparison, we adequately adjusted the threshold of all detectors since they return the closest approximation to $500$ keypoints per frame.

### 4.3.2 Robustness

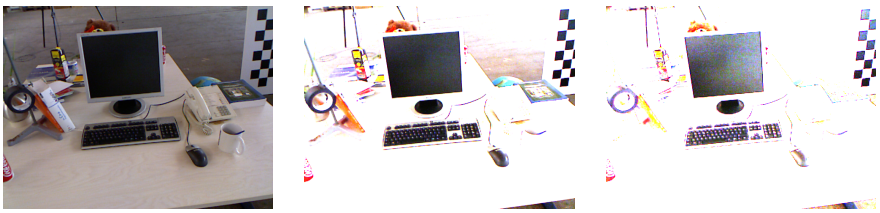
To test for robustness we evaluate the repeatability of the methods over several transformations: translational, scaling, rotational and illumination as well as artificial corruptions to contrast, brightness and added noise in both the visual image and the normal estimation.

In order to corrupt the contrast and brightness of an image we apply multiplication and addition with constants of increasing values:

$$\bar{I}(x,y) = \alpha I(x,y) + \beta,$$

where $\bar{I}$ is the corrupted image, $\alpha$ is the gain, which adjusts image contrast, and $\beta$ is the bias that controls brightness. For the added noise we used a Gaussian distribution with zero mean. Examples of these corruptions are represented in Figure 4.3.

For each transformation test, we compute the repeatability score under a specific image sequence $\mathcal{Q}_t = \{q_0, q_1, ..., q_n\}$, where each frame $q_i \in \mathcal{Q}_t$ along the sequence is increasingly



(a) Contrast

(b) Brightness

(c) Noise

**Figure 4.3.** Examples of the impact of the corruptions applied to an image

affected by the targeted transformation. The repeatability score is calculated among all sequence pairs of the form $(q_0, q_i)$, where $q_0, q_i \in Q_t$.

In order to avoid occlusion effects in motion related transformations, for every sequence pair $(q_0, q_i)$ we consider only the portion of the image that is common for both frames $q_0$ and $q_i$. To achieve this, we build a binary mask by projecting the first frame $q_0$ onto the camera position relative to frame $q_i$. Then, for a pixel $(i, j)$ of the mask: it is one if any projected point lies on it, and zero otherwise. We apply the keypoint detectors only on pixels where its corresponding (same coordinates) mask pixel has value equal to one.

The mask step is performed to prevent the repeatability rate from dropping due to unrelated issues (like object occlusion), instead of from the analysed transformation. A similar precaution was taken regarding the image corrupted tests (contrast, brightness and noise). To make sure the experiment is evaluating the algorithm's robustness to image corruption, we compute for each sequence position $q_n$ the average of the repeatability score of $10$ consecutive pairs $(q_n, q_i)$, where $i \in [n + 1, n + 10]$, thus we can assure that the repeatability rate is not dropping only because of unsuccessful matches due to lost keypoints.

To account for normal estimation noise we use two different methods, as discussed in Chapter 3. In order to assess how sensible each method is regarding the normals quality, we selected from the *freighburg_xyz2* dataset 50 sequences of 10 consecutive images $\mathcal{Q}_x = \{q_{x_1}, q_{x_2}, \cdots, q_{x_{10}}\}$ at random, totalling 500 elements. We run KVD, Harris3D and Harris6D with both coarse and accurate normal estimations, and calculate the repeatability scores over all pairs $(q_{n_0}, q_{n_i})$, where $q_{n_0}, q_{n_i} \in \mathcal{Q}_n$ for each of the 50 sequences. Further we calculate the differences between both coarse and accurate repeatability results for each method. A robust method should offer a lower variance when compared to itself using a different normal estimation method. Thus we evaluate the normal noise robustness by assessing the variance of this difference.

### 4.3.3 Distinctiveness

The keypoints distinctiveness evaluates the detector capability to find good features for a matching task for 2D descriptors (ORB and BRIEF Tombari et al. [2010]), 3D descriptors (FPFH Rusu et al. [2009] and SHOT Tombari et al. [2010]) and 2D + 3D descriptors BASE and CSHOT.

In order to evaluate the discriminative power of KVD detector, and to compare it against other approaches, we matched pairs of keypoints from several pairs of different images by using a brute force algorithm and feature vectors extracted by all these descriptors.

### 4.3.4 Time Performance

For time performance, we assess the processing time of the compared methods by averaging the detection time of $900$ runs over entire images. The detection time was measured while the experiments were running on an Intel Core i7 3.5GHz (using only one core) processor running Ubuntu 12.04 ($64$ bits).

## 4.4 Parameter Settings

In this section, we analyse the best parameter values to be used by our detector.

We chose the radii set $\mathcal{S} = \{3, 5, 7, 9\}$ as it represents well the scale spectrum, given that the Kinect's reach for the depth image ranges from 0.3 to 5 meters. It seems as a good balance between time performance and robustness.

In order to choose a value for the geometric threshold $t_g$, we ran the learning and testing phases for $15$, $30$ and $60$ degrees. As shown in Table 4.1, the value which returned the greatest accuracy in keypoint detection was the one with threshold for $15$ degrees for the

**Table 4.1.** Accuracy for different geometrical thresholds.

| Detector | $t_g = 15^o$ | $t_g = 30^o$ | $t_g = 60^o$ |
|---|---|---|---|
| Additive (KVD) | 0.90 | 0.82 | 0.87 |
| Concatenation | 0.91 | 0.86 | 0.90 |
| Geometric only | 0.83 | 0.78 | 0.85 |
| Visual only | 0.75 | 0.78 | 0.76 |

additive and concatenation combination. We chose to use the additive combination, since the concatenation spends twice the memory as the additive and the difference in their test accuracy was of one percentage point, and also shows better robustness to image corruptions noise (noise, brightness and contrast) as we will show in Section 4.5.

We experimented with different methods for combining the geometric and visual features. Figure 4.4 shows the scale robustness evaluation for four different methods: additive (KVD), concatenation, texture only, and geometric only. We can see that fusing information yields a stronger keypoint detector. We further examine KVD versus concatenation by testing for different transformation.

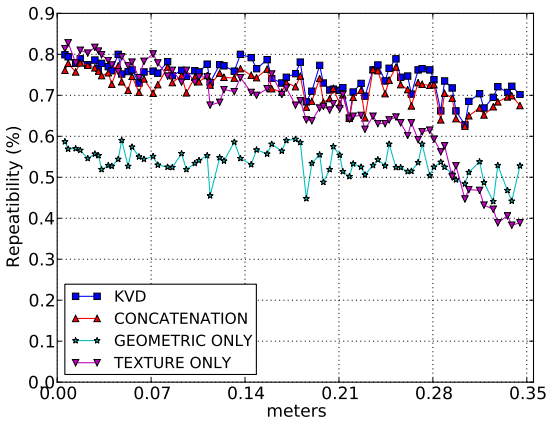
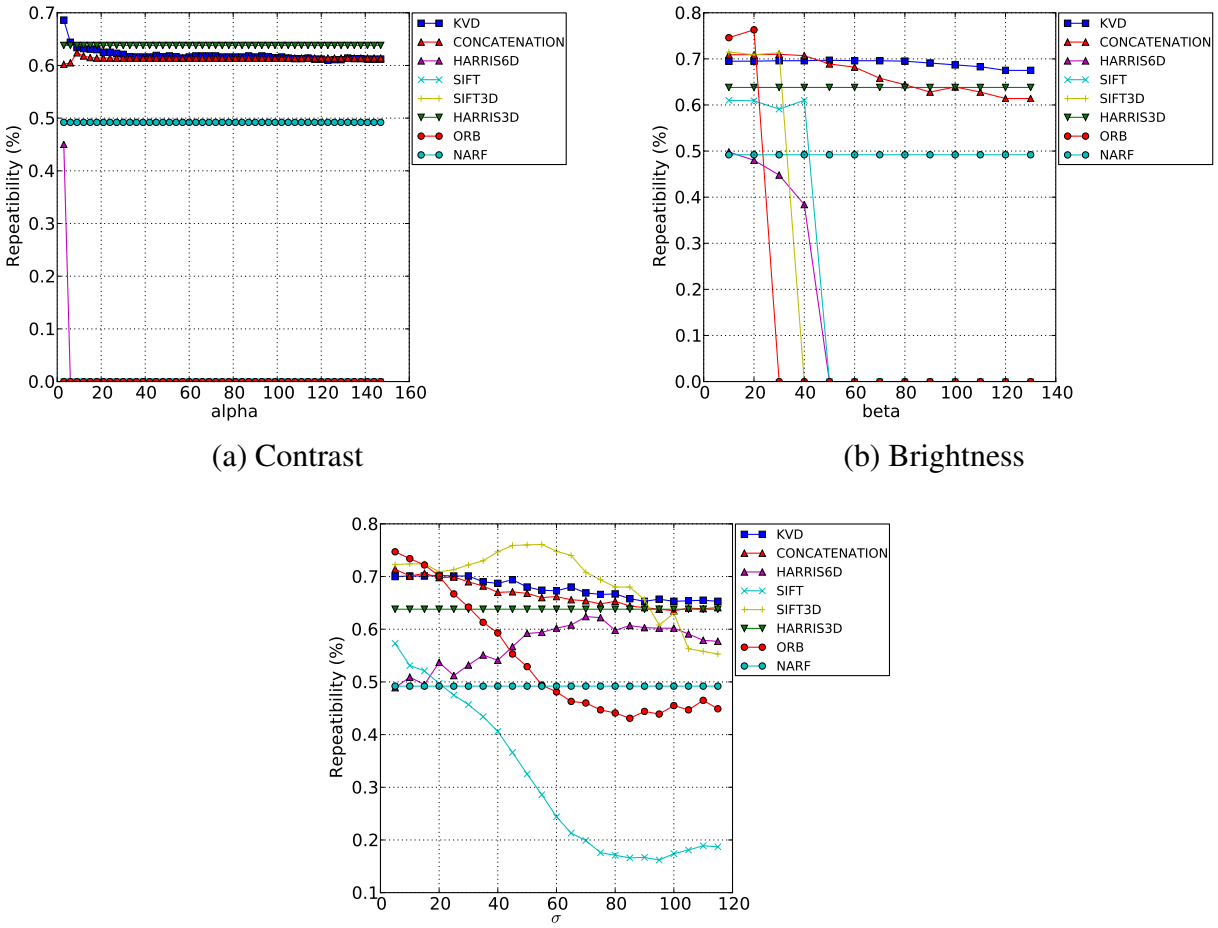



**Figure 4.4.** Scale test evaluating our method for four different feature vector extractions. The image shows that combining both visual and geometric information yields a stronger keypoint detector.

## 4.5 Results

In this section we show the results of the experiments explained in the previous section.




(a) Contrast

(b) Brightness

(c) Noise

**Figure 4.5.** Detector robustness experiments for (a) Contrast; (b) brightness changes and (c) Gaussian Noise.

### 4.5.1 Robustness

We evaluated each detector for robustness to translational, scaling, rotational and illumination as well as artificial corruptions to contrast, brightness and added noise.

We selected four sequences from the datasets to be used in our experiments, as previously explained in Section 4.1:

- *freiburg2_xyz*: Kinect sequentially moved along the x/y/z axes. We used a sequence with difference from 3 mm to 0.75 meters (horizontal direction) for translational tests. For scale tests we selected another set of frames with the camera moving away from the scene up to 0.35 meters;

- *freiburg2_rpy*: Kinect sequentially rotated around the three axes (roll, pitch and yaw). We used two different sequences for the roll and yaw tests. Rotating the relative pose

| Detector | $\mu$ | $\sigma$ |
|---|---|---|
| KVD | 0.00515 | 0.00055 |
| HARRIS6D | 0.00523 | 0.00063 |
| HARRIS3D | 0.2 | 0.00395 |

**Table 4.2.** Table summarizing the mean $\mu$ and variation $\sigma$ of the difference when using an accurate and a coarse normal estimation, we can see that KVD and HARRIS6D had the best performance.

to $35$ and $180$ degrees around the yaw and roll axes, respectively.

- *Verlab_Dusk*: Standing still Kinect capturing images at a rate of one per minute, starting at dusk (partial illumination) totalling $104$ captures.

Figure 4.6 shows the results of the motion tests: translational (a), scale (b) and both rotations (c) and (d), as well as the illumination change experiment. Our detector performs better than other approaches in most of the presented sequences. It provides the highest repeatability rate when there are large translational movements ($0.7$ meters) and large yaw angular rotations ($35$ degrees). A summary of the performance for all detectors is shown in Table 4.3. We can see that for most of the experiments the KVD detector presents the higher repeatability rate.

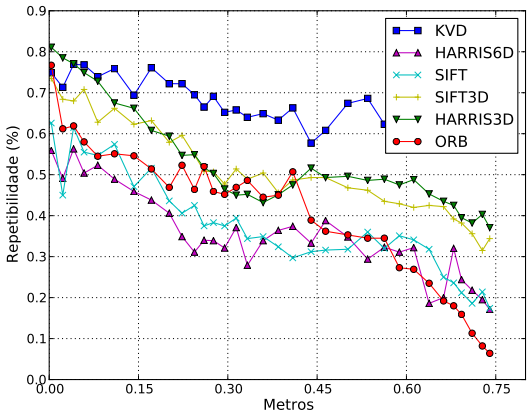
The results for the illumination change experiment can be seen in Figure 4.6 (e). The figure shows that only KVD and HARRIS3D (which uses only 3D data), were capable to continue to provide keypoints under low illumination. Although HARRIS6D combines both geometric and visual cues like KVD, it reveals much more dependence on visual features than our proposed method.

In Figure 4.5 shows that our detector presented the largest repeatability rate in the majority of the tests, followed by HARRIS3D. Since our detection methodology considers visual and geometric information, we were able to detect keypoints even in the absence of visual data ($\alpha > 10$) or when the image was completely saturated ($\beta > 45$). Once again we can notice the HARRIS6D visual dependence.

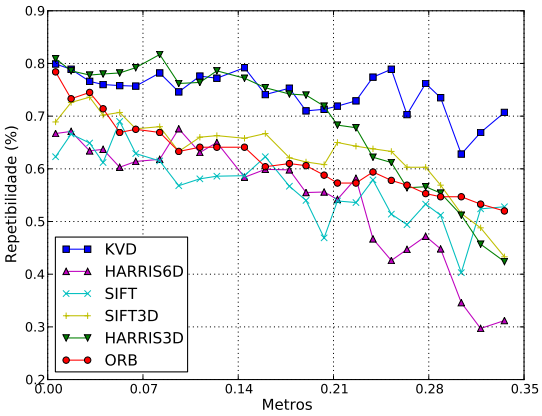
For the normal robustness test, we summarized the results in Table 4.2, we can see that all three methods presented a good performance, with variations close to zero. But KVD and HARRIS6D outperformed HARRIS3D, in our metrics, by one order of magnitude.

## 4.6 Discriminative Power and Time Performance

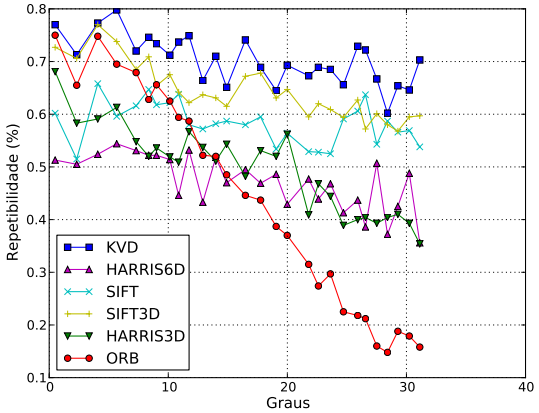
Finally, in this section we present the results for the keypoints distinctiveness experiments. We tested the detectors capabilities to find good features for a matching task for 2D descrip-



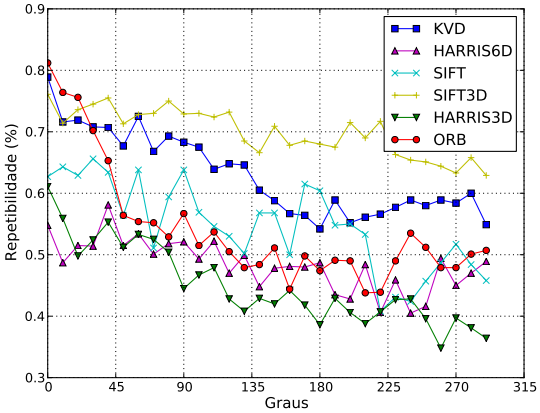

(a) Translation



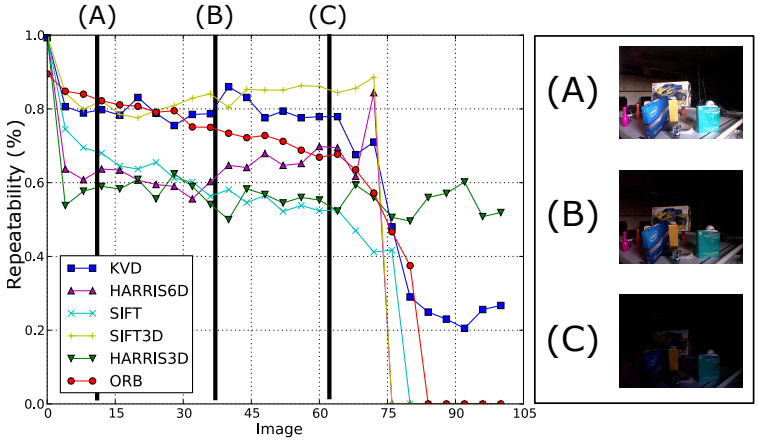

(b) Scale




(c) Rotation (yaw)




(d) Rotation (roll)




(e) Illumination change

**Figure 4.6.** Results for some of the robustness experiments: (a) Horizontal translation came motion; (b) Scale changing; (c) Rotation around the yaw axis; (d) Rotation around the roll axis and (e) illumination change.

| Detector | Translation | Scale | Yaw R. | Roll R. | Light | Contrast | Bright | Noise | Mean |
|---|---|---|---|---|---|---|---|---|---|
| KVD | **1,00** | **1,00** | **1,00** | **1,00** | **1,00** | **1,00** | **1,00** | **1,00** | **1,00** |
| Harris 3D | 0,79 | 0,91 | 0,69 | 0,74 | 0,85 | 1,03 | 0,92 | 0,94 | 0,86 |
| CAT | 1,00 | 0,96 | 1,00 | 0,99 | 1,00 | 0,99 | 0,96 | 0,98 | 0,98 |
| SIFT 3D | 0,76 | 0,84 | 0,91 | 1,07 | 0,94 | 0,00 | 0,20 | 1,02 | 0,72 |
| SIFT | 0,55 | 0,75 | 0,81 | 0,87 | 0,69 | 0,00 | 0,26 | 0,45 | 0,55 |
| Harris 6D | 0,51 | 0,72 | 0,66 | 0,76 | 0,72 | 0,00 | 0,19 | 0,84 | 0,55 |
| ORB | 0,60 | 0,83 | 0,61 | 0,85 | 0,88 | 0,00 | 0,15 | 0,79 | 0,59 |

**Table 4.3.** Comparative table of the different experiments. The values were computed as ratio between the KVD Area Under the Curve (AUC) and the target detectors AUC. Thus, values below $1,00$ mean that KVD performed better, while values above $1,00$ indicate a better performance for the competitor method.

tors (ORB and BRIEF Tombari et al. [2010]), 3D descriptors (FPFH Rusu et al. [2009] and SHOT Tombari et al. [2010]) and 2D + 3D descriptors BASE and CSHOT. Thus, in order to evaluate the discriminative power of KVD detector, and to compare it against other approaches, we matched pairs of keypoints from several pairs of different images by using a brute force algorithm and feature vectors extracted by all these descriptors. When compared against HARRIS6D, which also combines texture and geometrical features, KVD obtained an inlier rate of $0.41\%$ and HARRIS6D a total of $0.25\%$ of inliers, as can be seen in Figure 4.7.

## 4.7 Comparison

In order to find the best algorithm and to evaluate the performance of our proposed method against others, we run a performance analysis of the algorithms with a statistical system comparison technique (Jain [1991]). The following discussion is limited to the comparison of one against one algorithms, using the same workload for each one of them.

All experiments were executed on the same computer. For each transformation we used the same dataset sequence from the experiments for the comparison, where only a specified transformation varies in each sequence. These were the factors which we compared the algorithms:

- Motion related:
  - Translation,
  - Scale,

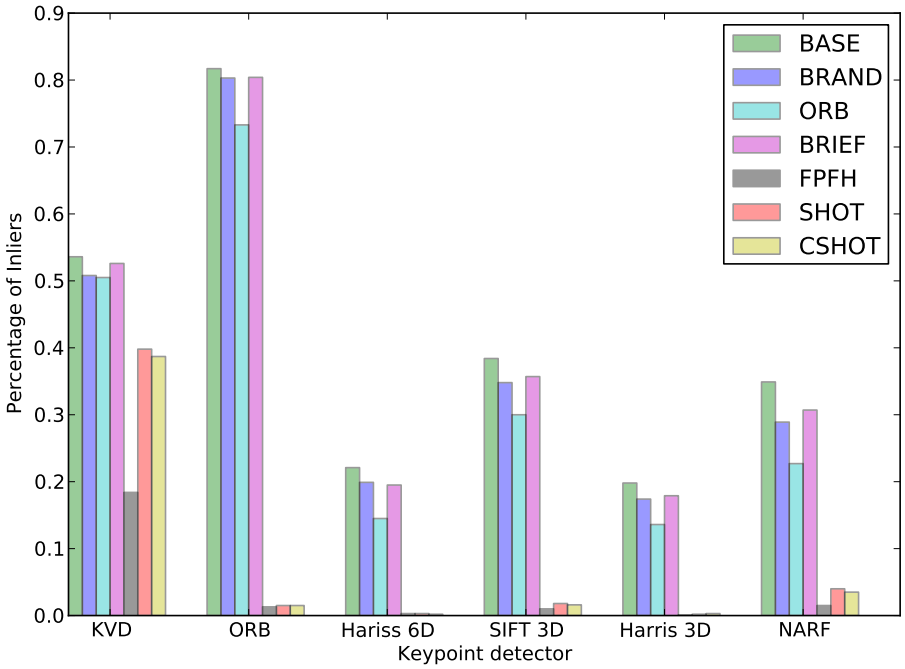



**Figure 4.7.** Distinctiveness: we can notice that KVD provides reliable feature for both geometric and visual descriptors.

  - Rotation (yaw and roll axes).

- Image Corruption:
  - Contrast,
  - Brightness,
  - Noise,
  - Normal Estimation.

- Illumination Changes.

For each compared factor, we have the pairwise repeatability score for each algorithm computed from the performed experiments. Thus, we can establish a one-to-one relationship between each algorithm with the repeatability results for each of the experimental observations. This method is called *observations paired*. The statistical procedures to compare two systems with observations paired is an extension of the test for a zero mean.

The repeatability scores for each algorithm are computed for a sample of the images. The first step is to generate a new sample containing the observations for each comparison, the sample consisting of the differences of the repeatability between both algorithms being

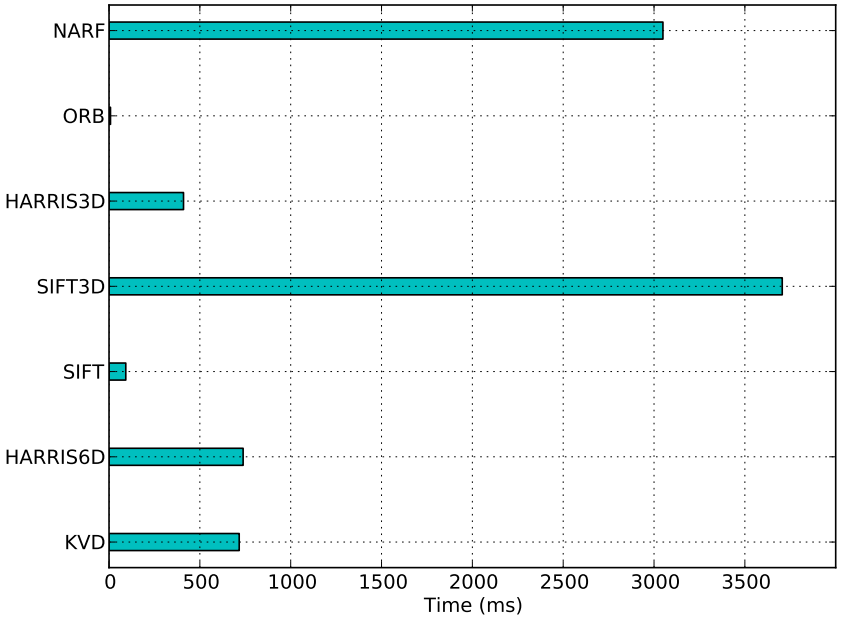



**Figure 4.8.** Time performance: We can see that, although KVD does not benefits from an optimized implementation, its time performance is one of the best among the descriptors using the same type of input data (Harris6D, NARF, Harris3D).

compared, for each element of the dataset, for example Figure 4.9 (a) show the differences for the scale factor.

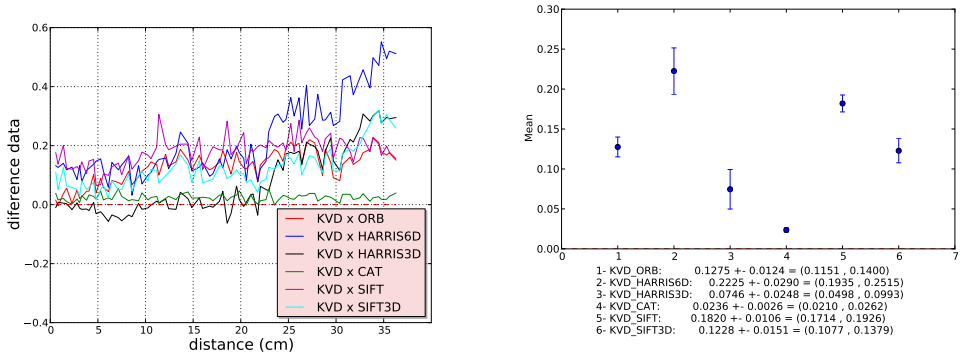
Then, we compute statistics of this new sample of differences: the *sample mean* $\bar{\mu}$, the *sample variance* $\bar{\sigma}$, the *sample standard deviation* $\sqrt{\sigma}$, and the *confidence interval* (CI) $I_z$. Let $\mathcal{Q} = \{q_0, q_1, \cdots, q_n\}$ be the analysed sample, we compute these statistics as

$$\begin{aligned} \bar{\mu} &= \frac{1}{n}\sum_{q_i \in \mathcal{Q}} q_i \\ \bar{\sigma} &= \frac{1}{n-1}\sum_{q_i \in \mathcal{Q}} (q_i - \bar{\mu})^2 \\ I_z &= \bar{\mu} \pm z\frac{\sqrt{\sigma}}{\sqrt{n}}. \end{aligned} \tag{4.2}$$

For the CI computation we consider the distribution of the means of the differences sample, which follows a normal distribution according to the central limit theorem. Thus, for a 95% confidence interval $z = 1.96$, in our experiments we used the sample size $n$ as shown in Table 4.4. The behaviour of the samples for the scale factor are shown in Figure 4.9.

| Transformation type | Sample size |
|---|---|
| Scale | 80 |
| Translation | 198 |
| Yaw | 110 |
| Roll | 300 |
| Illumination Change | 104 |
| Noise | 23 |
| Contrast | 25 |

**Table 4.4.** Sample size used to compute the confidence interval of each transformation.




(a) Differences (b) Confidence intervals

**Figure 4.9.** (a) Image showing the difference for the scale factor between the repeatability scores for KVD and the other methods. (b) Confidence intervals. We can see that for all methods, the confidence interval lies above the zero line, which means that with 95% confidence, KVD is more robust to scale.

The 95% confidence interval (CI) is then calculated for each sample. The test of zero mean is used to verify whether the confidence interval is above, below or includes the zero. When the confidence interval includes the zero, it means that we can not infer which is the better method. Nonetheless, if the CI is completely above or below the zero line, we can affirm with 95% confidence that the KVD is either better or worse than the compared method, respectively. The complete results are shown on Figure 4.7 and pairwise detailed in Figures 4.12, 4.13, 4.14 and 4.15.

A simpler comparison, considering only the repeatability means of each sequence evaluated at Section 4.5 is shown in Figure 4.11. Each vertex represents the mean of the repeatability score, achieved by each evaluated detector, in a different transformation type.

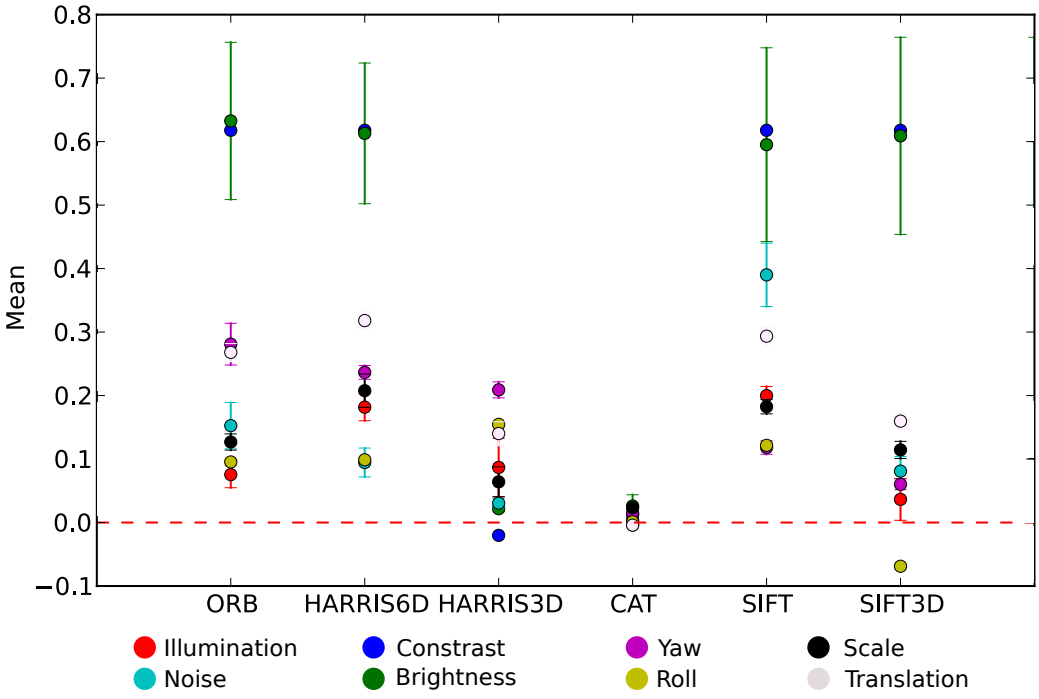



**Figure 4.10.** We can see that our method outperforms the others in the majority of the realized tests.

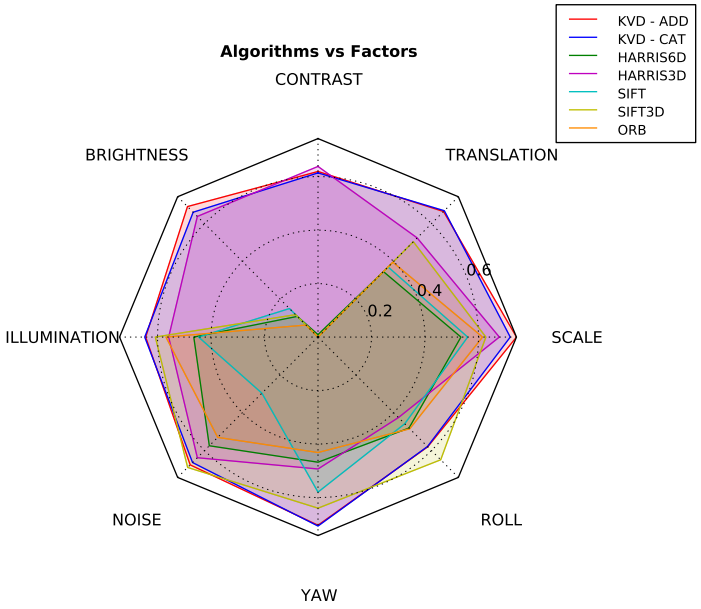
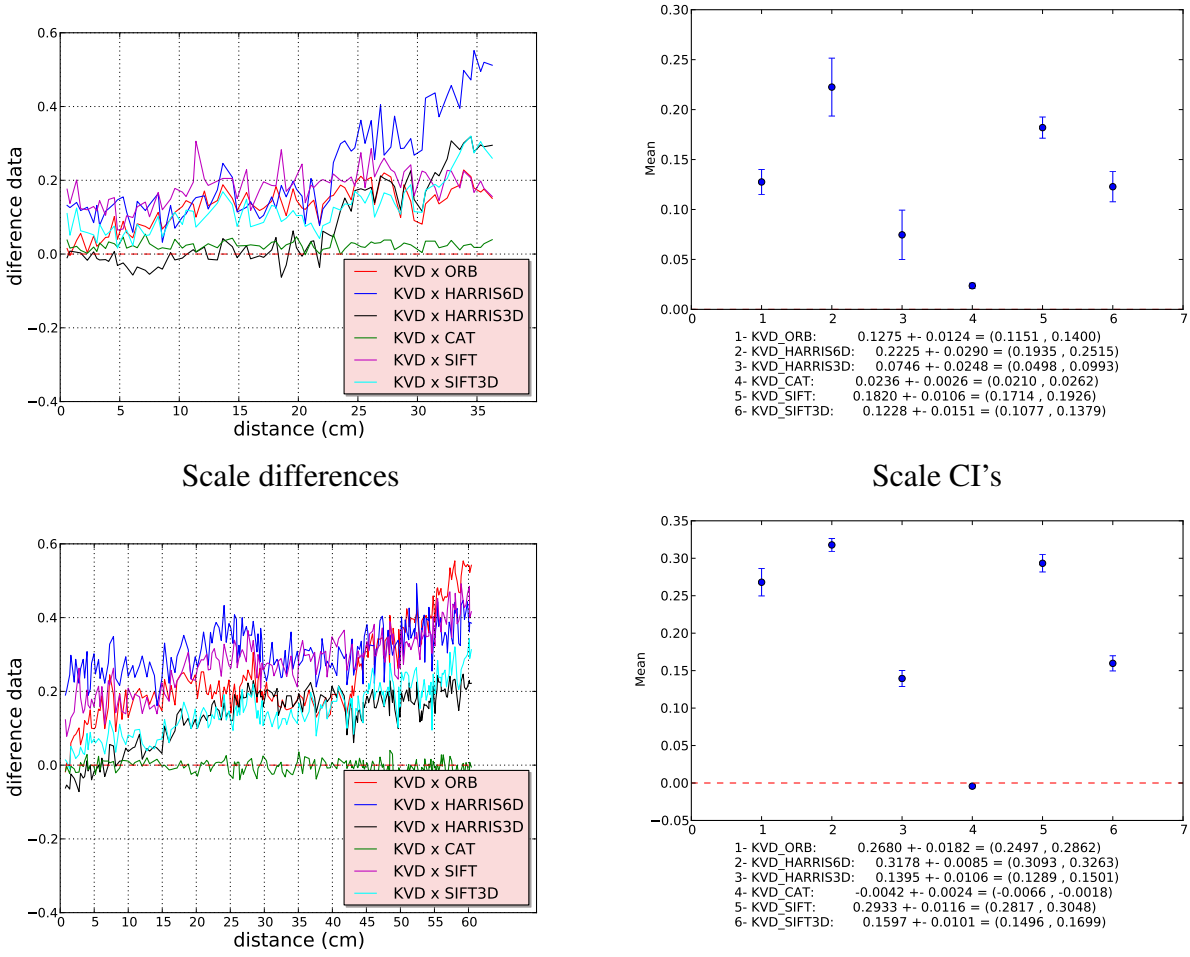



**Figure 4.11.** Comparison among different keypoint detectors, one can see that the area of the polygon representing KVD is the biggest among the evaluated detectors.




Scale differences | Scale CI's

Translation differences | Translation CI's

**Figure 4.12.** Difference samples and Confidence Interval for the scale and translation transformations.

## 4.8 Concluding Remarks

As the experiments and comparisons show, the KVD outperformed several state-of-the-art methods regarding numerous of different transformations, as well as the proposed methodology using the concatenation fusion type.

As drawbacks, our method is surpassed by SIFT3D and HARRIS3D in the roll rotation and the contrast transformations, respectively. For the contrast transformation, the explanation is that the Harris3D method does not makes use of visual data, thus contrast transformation adds no noise for its perspective. For the roll rotational transformations, the developed methodology does not deploy any rotation invariance mechanism, and this is addressed as a future work.

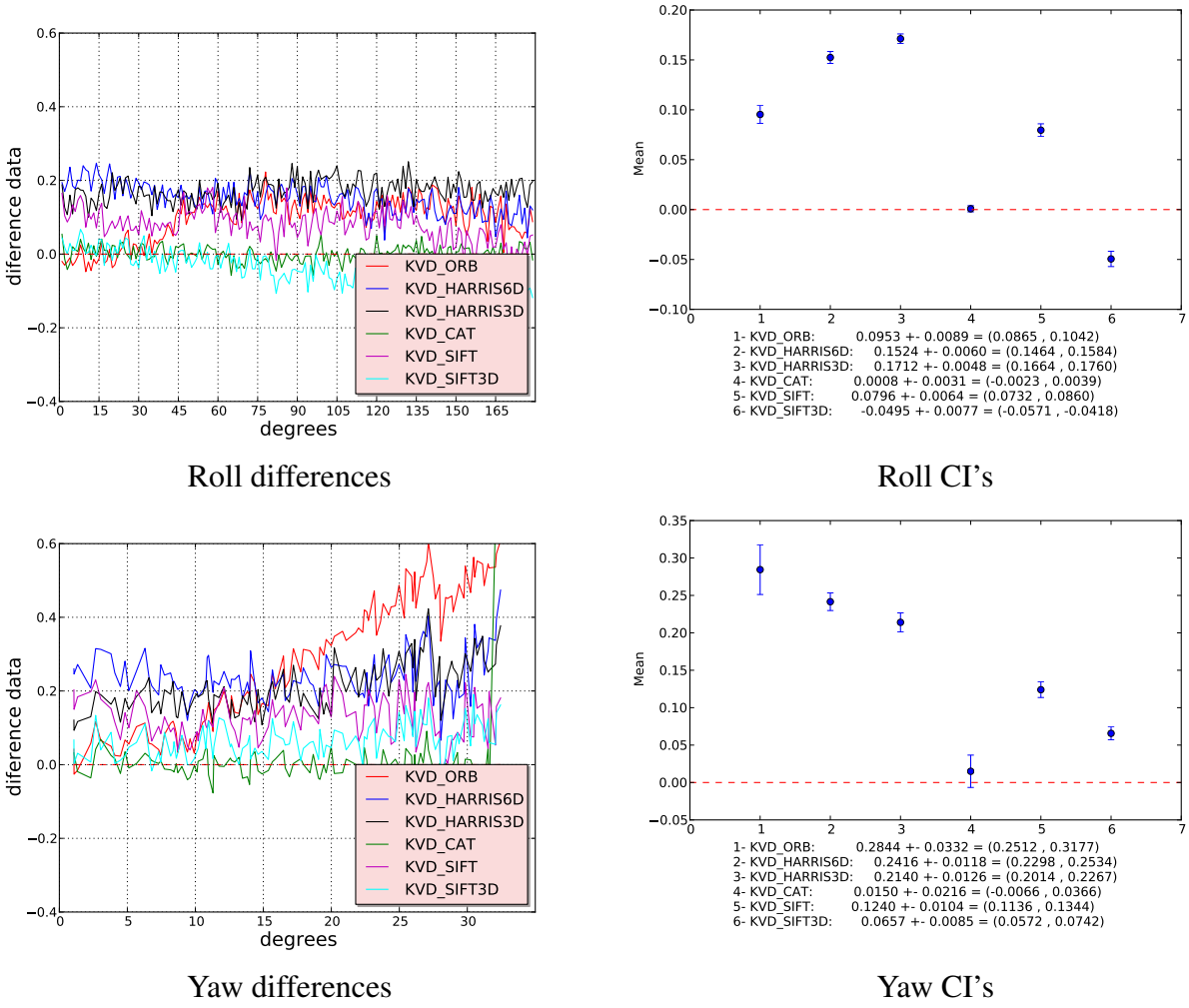
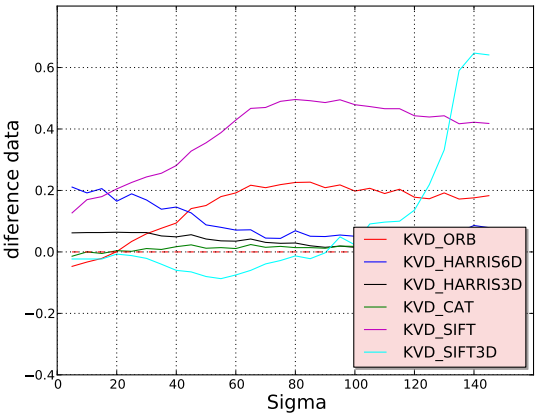
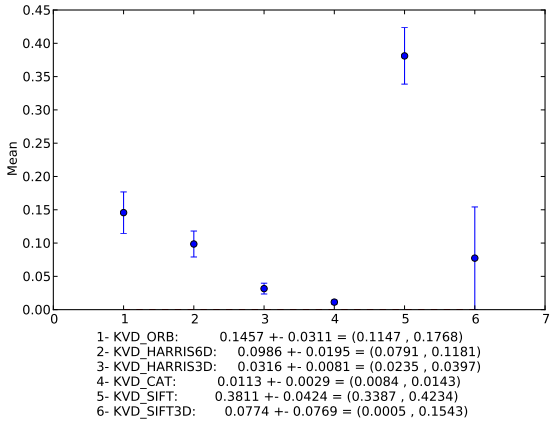


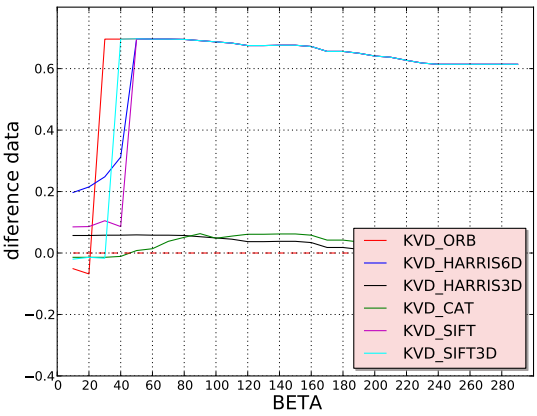

**Figure 4.13.** Difference samples and Confidence Interval for the row and yaw rotational transformations.



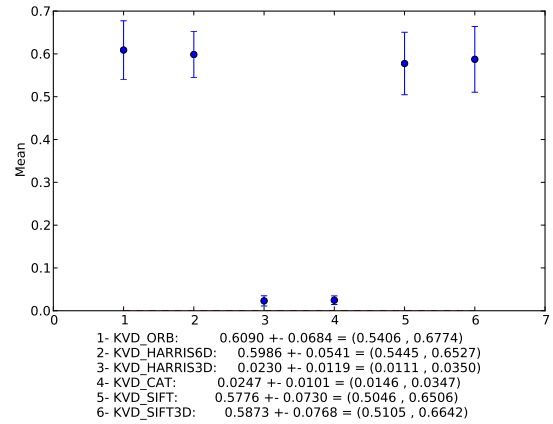

Noise differences




Noise CI's

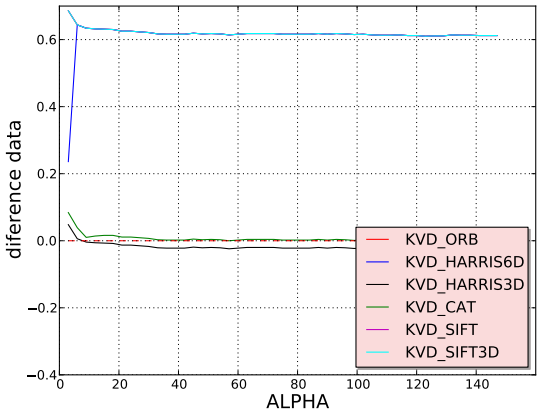



Brightness differences

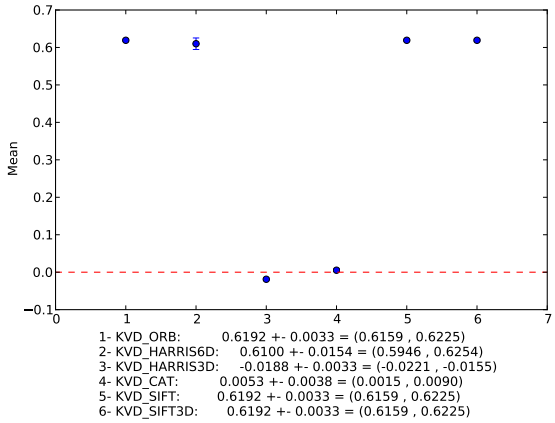



Brightness CI's

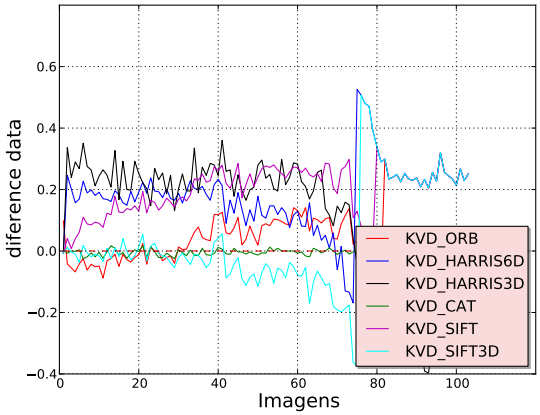
**Figure 4.14.** Difference samples and Confidence Interval for the noise and brightness corruptions.



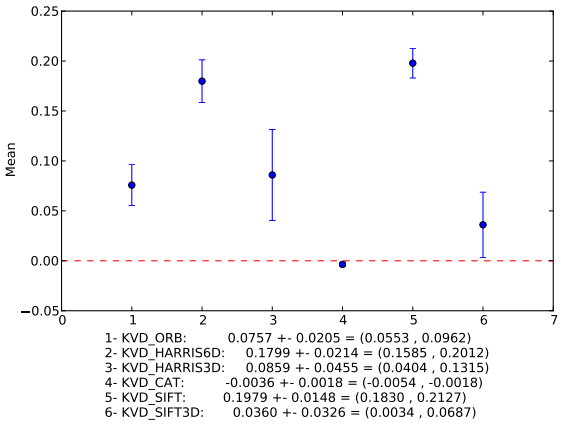

Contrast differences




Contrast CI's




Illumination differences




Illuminations CI's

**Figure 4.15.** Difference samples and Confidence Interval for the contrast and illumination corruptions.

# Chapter 5

# Conclusion

## 5.1 Conclusion

In this work we proposed KVD, a keypoint detector that is capable to work with texture and geometrical data. We presented the results of several experiments that were conducted to show the behavior of our methodology. A comparative analysis in terms of robustness to affine transformations, processing time and distinctness was conducted against the standard detectors in the literature for appearance and geometric information. In those experiments our detector outperformed all other approaches, including a version of Harris corner detector which also fuses visual and geometry information.

Thanks to the strategy of combining different cues our detector was more stable in matching experiments. As it is also shown in the experiments the combination of appearance and geometry information indeed leads to a significantly better performance when compared to using either information alone. Moreover, our detector had similar processing performance and presented high repeatability scores for images severely corrupted with noise, images with low contrast, saturated images, and several transformations like translation, scale, and rotation due to camera motion.

## 5.2 Future Work

Regarding future work, we enumerate several important points that need improvements.

- **Rotational invariance** Since we identify a keypoint through a machine learning algorithm, our method is heavily dependent on the training set composition. One way to decrease such dependence would be to incorporate a mechanism to add rotational invariance in the feature extraction phase.

- **Time performance** An optimized implementation and converting the trained decision tree into nested if-then-else statements would significantly improve the time performance.
- **Decision tree optimization** Optimization methods such as simulated annealing could be applied to improve the classification of the decision tree.
- **Other information fusion mechanisms** A research for studying other methods of fusing both visual and geometric information would be a good concern. Another front to explore would be the training of independent decision trees for visual and depth data then combining their decision to classify whether a certain point is a keypoint or not. This kind of approach might allow a level of flexibility for the algorithm, making it able to adapt better for different environment types.

# Bibliography

Chen, H. and Bhanu, B. (2004). 3D Free-Form Object Recognition in Range Images Using Local Surface Patches. *Pattern Recognition, International Conference on*, 3:136–139. ISSN 1051-4651.

Dias, P. G. T., Kassim, A., and Srinivasan, V. (1995). A neural network based corner detection method. In *IEEE Intl. Conf. on Neural Networks*, volume 4, pages 2116–2120.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147--151.

Hasegawa, T., Yamauchi, Y., Ambai, M., Yoshida, Y., and Fujiyoshi, H. (2014). Keypoint detection by cascaded fast. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 5676–5680.

Holzer, S., Shotton, J., and Kohli, P. (2012). Learning to Efficiently Detect Repeatable Interest Points in Depth Data. In *Proc. of the Europ. Conf. on Comp. Vision*, ECCV'12, pages 200--213, Berlin, Heidelberg. Springer-Verlag.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience.

Janoch, A., Karayev, S., Jia, Y., Barron, J., Fritz, M., Saenko, K., and Darrell., T. (2011). A Category-Level 3-D Object Dataset: Putting the Kinect to Work. In *ICCV Workshop on Consumer Depth Cameras in Computer Vision*.

Kanezaki, A., Marton, Z.-C., Pangercic, D., Harada, T., Kuniyoshi, Y., and Beetz, M. (2011). Voxelized Shape and Color Histograms for RGB-D. In *IROS Workshop on Active Semantic Perception*.

Klasing, K., Althoff, D., Wollherr, D., and Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. In *IEEE Intl. Conf. on Robotics and Automation*, pages 3206 –3211.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79--116.

Lowe., D. G. (2004). Distinctive image features from scale-invariant keypoints. *Internationl Journal of Computer Vision*, pages 91--110.

Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86. ISSN 0920-5691.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43--72.

Morevec, H. P. (1977). Towards automatic visual obstacle avoidance. In *Proceedings of the 5th international joint conference on Artificial intelligence*, volume 2 of *IJCAI'77*, pages 584--584, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nascimento, E. R., Oliveira, G. L., Vieira, A. W., and Campos, M. F. (2013). On the development of a robust, fast and lightweight keypoint descriptor. *Neurocomputing*, 120(0):141–155. ISSN 0925-2312.

Pitteway, M. L. (1967). Algorithm for drawing ellipses or hyperbolae with a digital plotter. *The Computer Journal*, 10(3):282--289.

Rosin, P. L. (1999). Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291--307.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1508--1515. IEEE.

Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Computer Vision–ECCV 2006*, pages 430--443. Springer.

Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119. ISSN 0162-8828.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: an efficient alternative to SIFT or SURF. In *IEEE Int. Conf. on Comp. Vision*.

Rusu, R., Blodow, N., and Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In *IEEE Intl. Conf. on Robotics and Automation*.

Rusu, R. and Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. ISSN 1050-4729.

Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345--348.

Steder, B., Rusu, R. B., Konolige, K., and Burgard, W. (2011). Point Feature Extraction on 3D Range Scans Taking into Account object boundaries. In *IEEE Intl. Conf. on Robotics and Automation*.

Sturm, J., Magnenat, S., Engelhard, N., Pomerleau, F., Colas, F., Burgard, W., Cremers, D., and Siegwart, R. (2011). Towards a benchmark for RGB-D SLAM evaluation. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*.

Tombari, F., Salti, S., and Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *Proc. of the Europ. Conf. on Comp. Vision*, pages 356--369.

Tombari, F., Salti, S., and Stefano, L. D. (2011). A combined texture-shape descriptor for enhanced 3D feature matching. In *IEEE Intl. Conf. on Image Processing*.

Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177--280. ISSN 1572-2740.

Zaharescu, A., Boyer, E., Varanasi, K., and Horaud, R. P. (2009). Surface Feature Detection and Description with Applications to Mesh Matching. In *IEEE Conf. on Comp. Vision and Pattern Recog.*