UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LEONARDO PERDOMO

# c-M2DP: A Fast Point Cloud Descriptor with Color Information to Perform Loop Closure Detection

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Edson Prestes e Silva Junior

Porto Alegre
November 2019

**CIP — CATALOGING-IN-PUBLICATION**

Perdomo, Leonardo

c-M2DP: A Fast Point Cloud Descriptor with Color Information to Perform Loop Closure Detection / Leonardo Perdomo. – Porto Alegre: PPGC da UFRGS, 2019.

62 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2019. Advisor: Edson Prestes e Silva Junior.

1. Point Cloud Descriptor, Loop Closure Detection, SLAM. I. Prestes e Silva Junior, Edson. II. Título.

*“[...] it is such a *simple* thing, not at all what you thought it might be, and you feel yourself suddenly comforted. In knowing your name, your true name, you know that you have gained back perhaps the most important part of yourself. In knowing your name, you know yourself, and you know, now, there is very little you cannot do.”*

— THE NAMELESS ONE, “PLANESCAPE: TORMENT”, 1999

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my parents, Helena and Antonio, and my brother, Maurício, for their unconditional support, especially during this journey.

Also, I would like to thank my advisor Edson Prestes and professors Mariana Kolberg and Renan Maffei for their support, advices, insightful commentaries, and for giving me this incredible opportunity.

Finally, I would like to thank my colleagues and friends, Matheus Ritter, Bruno Bittarello, Eduardo Conter, Ricardo Westhauser, Fernanda Rodrigues, Diego Pittol, Mathias Mantelli and Renata Neuland for their companionship, support, advices, patience, and joy shared during this journey.

**ABSTRACT**

We present c-M2DP, a fast global point cloud descriptor that combines color and shape information, and perform loop closure detection using it. Our approach extends the M2DP descriptor by incorporating color information. Along with M2DP shape signatures, we compute color signatures from multiple 2D projections of a point cloud. Then, a compact descriptor is computed using SVD to reduce its dimensionality. We performed experiments on four public available dataset sequences, with point clouds generated using either camera-LIDAR fusion or stereo depth estimation. Our results show an overall accuracy improvement over M2DP while maintaining efficiency, and are competitive in comparison with another color and shape descriptor.

**Keywords:** Point Cloud Descriptor, Loop Closure Detection, SLAM.

**c-M2DP: Um Descritor Rápido de Nuvem de Pontos com Informações de Cor para Efetuar Detecção de Fechamento de Loop**

## RESUMO

Esta dissertação apresenta o c-M2DP, um descritor global de nuvem de pontos rápido e que combina informações de cor e forma, além de sua utilização na detecção de fechamento de *loop*. A abordagem é uma extensão do descritor M2DP que incorpora informações de cor. Em conjunto com as assinaturas da forma originais do M2DP, são computadas assinaturas de cor de multiplas projeções 2D de uma nuvem de pontos. Em seguida, um descritor compacto é computado usando SVD para redução da dimensionalidade. Experimentos em quatro sequências de conjuntos de dados disponíveis publicamente foram realizados. Os resultados apresentam um aprimoramento na precisão em relação ao descritor M2DP, mantendo-se eficiente, e são competitivos em comparação à outro descritor que combina cor e forma.

**Palavras-chave:** Descritor de Nuvem de Pontos, Detecção de Fechamento de Loop, SLAM.

## LIST OF ABBREVIATIONS AND ACRONYMS

| | |
|---|---|
| c-M2DP | Color Multiview 2D Projection |
| CoSPAIR | Colored Histograms of Spatial Concentric Surflet-Pairs |
| CSHOT | Color Signature of Histograms of Orientations |
| EKF | Extended Kalman Filter |
| ICP | Iterative Closest Point |
| FN | False Negative |
| FoV | Field-of-View |
| FP | False Positive |
| KITTI | Karlsruhe Institute of Technology and Toyota Technological Institute |
| LIDAR | Light Detection And Ranging |
| M2DP | Multiview 2D Projection |
| OpenCV | Open Source Computer Vision Library |
| PCL | Point Cloud Library |
| PC | Principal Component |
| PCA | Principal Component Analysis |
| RGB | Red-Green-Blue Color Space |
| RGB-D | Red-Green-Blue Color Space and Depth |
| ROS | Robot Operating System |
| SHOT | Signature of Histograms of Orientations |
| SLAM | Simultaneous Localization And Mapping |
| SVD | Singular Value Decomposition |
| TP | True Positive |
| VFH | Viewpoint Feature Histogram |

## LIST OF FIGURES

## LIST OF TABLES

**CONTENTS**

# 1 INTRODUCTION

Autonomous robots became a common sight in many aspects of our society. Examples of their current developments and applications can be found in different environments, including stationary industrial arms in manufacturing (WHITE, 2017; MARR, 2019), mobile platforms automating logistics and retail warehouses (VINCENT, 2018; SIMON, 2019), automated cleaners and other domestic tasks (SCHUMACHER, 2018), search and rescue robots in emergency services (BROWN, 2018), and also self-driving vehicles (DAVIES, 2018). To accomplish these tasks autonomously in real-world environments, robots often rely on the ability to perform self-localization using a representation of their surroundings. Truly autonomous robots can either localize themselves in a map of the environment known beforehand, or build a map while simultaneously performing self-localization, also known as Simultaneous Localization and Mapping (SLAM) (SIEGWART; NOURBAKHSH, 2004).

Over the past decades, the SLAM problem has been widely investigated, with numerous approaches being proposed to improve different aspects of it, such as pose uncertainty, map building and measurements ambiguities (DURRANT-WHYTE; BAILEY, 2006). However, SLAM is still an open problem with varying research maturity depending on the type of robot being used, the operating environment and the given precision requirements (CADENA et al., 2016).

The recognition of previously visited places by a robot is an important task for SLAM. Also known as loop closure detection, this subject draws more attention as robots operate over long distances, accumulating errors during movement that increase pose uncertainty. If two observations correspond to the same place in the environment, a robot can recognize where it is and correct the drifting.

Based on the sensors equipped on the robot, different approaches can be employed to detect loop closures. Earlier works using range finders, namely sonars or LIDAR[1], usually employed comparisons between range measurements, among other methods, for this task (THRUN; BURGARD; FOX, 2005). Recent works using 3D LIDARs often perform a correspondence search, matching 3D feature descriptors extracted from point clouds generated using range measurements (BOSSE; ZLOT, 2013; RÖHLING; MACK; SCHULZ, 2015). Likewise, a common 2D camera based approach to detect loop closures is by matching feature descriptors extracted from images (CUMMINS; NEWMAN, 2008;

[1] Light Detection and Ranging

SÜNDERHAUF; PROTZEL, 2011), among several other methods such as computing the sum of absolute differences from image patches (MILFORD; WYETH, 2012).

Recently, He, Wang and Zang (2016) proposed the global point cloud descriptor Multiview 2D Projection (M2DP), and applied it to a LIDAR-based loop closure detection approach. It outperforms other state-of-the-art global point cloud descriptors by both accuracy and efficiency. The M2DP descriptor is built using signatures of geometric information, computed from multiple 2D projections of a point cloud. Its structure can be naturally extended to incorporate signatures of additional information that may be associated with the point cloud.

As shown in object recognition (LOGOGLU; KALKAN; TEMIZEL, 2016) and other surface matching (TOMBARI; SALTI; STEFANO, 2011; FENG; LIU; LIAO, 2015) works, combining color and shape information can enhance descriptiveness over shape only descriptors. Point clouds with associated color information are able to provide additional appearance details from the scene, which in turn can be exploited by point cloud descriptors. However, loop closure detection approaches that combine color with shape data are still insufficiently investigated.

Our focus in this work is to incorporate color information into a state-of-the-art point cloud descriptor, in order to improve the loop closure detection performed by an autonomous vehicle. Although 3D based approaches to loop closure detection have found significant results using only shape information (BOSSE; ZLOT, 2013; RÖHLING; MACK; SCHULZ, 2015; HE; WANG; ZHANG, 2016), adding color can improve its accuracy by making use of information commonly available from the vehicle cameras. Either by combining sensors, such as a 3D LIDAR and a single camera, or employing stereo[2] cameras, we were able to generate and use point clouds with associated color information for this goal.

## 1.1 Motivation

In this section we present our motivation for this work by beginning with a discussion about the requirements for autonomy of mobile robots. After that, we examine why these robots need to build a map and self-localize at the same time, and why the ability to recognize a previously visited place is so important for this task. Then, we discuss on how a mobile robot can have such ability, focusing on 3D vision-based approaches.

[2]Multiple camera configuration with partially overlapped field of views

### 1.1.1 How a mobile robot can achieve autonomy?

In mobile robotics, several problems need to be addressed in order to successfully achieve autonomy. According to Makarenko et al. (2002), localization, mapping and motion planning are the three fundamental problems that an autonomous mobile robot need solving, to be able to accurately perceive the surroundings and accomplish its objectives in a environment.

The **localization** problem comprises in estimating the correct robot's pose[3] through its sensors readings, assuming a priorly known map that represents the environment accurately. It can be either local, when the initial pose is known and subsequent estimates are done during movement, or global, when localization is done without an initial pose (STACHNISS, 2006).

On the other hand, the **mapping** problem assumes that the correct robot's pose is known to build an accurate representation of the environment. Sensors readings gathered by the robot are used, and the map is often incrementally enhanced by new readings over time (STACHNISS, 2006).

**Motion planning** consists in efficiently guiding a robot to reach a goal location. It assumes that an accurate map and a correct pose is known (SIEGWART; NOURBAKHSH, 2004).

Often, autonomous robots need to perform some of these tasks simultaneously, with their combination generating other problems as seen in Fig. 1.1. For instance, despite being able to be performed independently, localization and mapping can be so closely related that they cannot be decoupled. Over the past decades, building maps while simultaneously self-localizing became an important subject in order to solve robotic navigation without external sensors. **Simultaneous Localization and Mapping (SLAM)** is a widely investigated topic by the robotics community, but is still an open problem that draws much attention due to its application in different robots, sensors and environments.

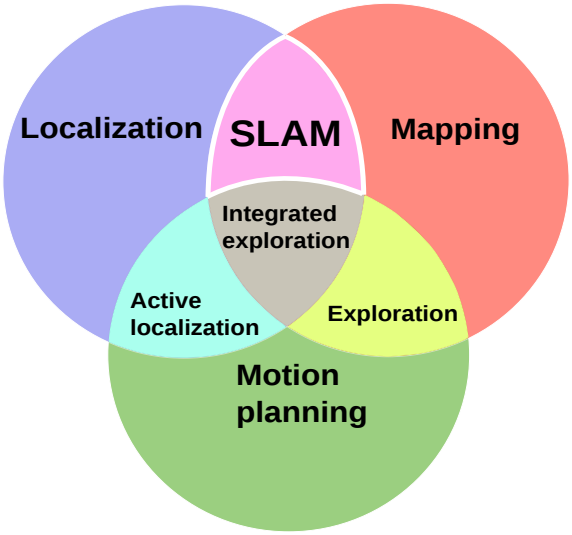
### 1.1.2 Why an autonomous mobile robot would need SLAM?

Building a map of the environment while self-localizing is considered a challenging problem in robotics (THRUN; BURGARD; FOX, 2005). Sensors measurements are used to construct a map of an environment partially or totally unknown, while the robot

[3] Position and orientation of the robot in the environment.

state, described by its pose, is simultaneously estimated in the map (DISSANAYAKE et al., 2001). SLAM is strongly dependent on an accurate map of the environment to determine the robot's pose, while at same time requiring that the pose, whose uncertainty increases as the robot moves, is precisely known to build a map (SIEGWART; NOURBAKHSH, 2004; THRUN; BURGARD; FOX, 2005).

Figure 1.1: Localization, Mapping, Motion control and the problems originated by their overlapping areas. Adapted from Makarenko et al. (2002).

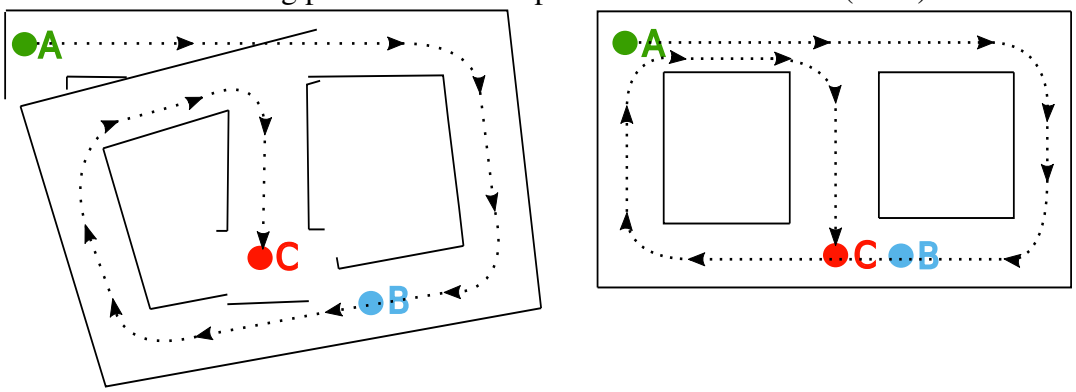



During movement, the robot new readings of the environment are associated with landmarks built in the map. Bailey and Durrant-Whyte (2006) remarked that early SLAM proposals employed naive **data association** methods, testing each measurement individually to cull unlikely associations. Such approaches are considered unreliable due to the lack of geometric relationship between landmarks (BAILEY; WHYTE, 2006), with later proposals (NEIRA; TARDOS, 2001) considering multiple associations and their geometric relations simultaneously with more robust batch validation gate methods. A reliable data association method is critical for SLAM algorithms, as any incorrect association can be catastrophic for map building and also can lead SLAM into a irrecoverable state (BAILEY; WHYTE, 2006).

Data association is also extremely important when a robot returns to a previously visited place, known as the **loop closure detection** problem (HO; NEWMAN, 2006). As previously mentioned, errors accumulated during robot movements increase the pose uncertainty and cause drifting, resulting in an inconsistent representation of the environment. When a loop closure is detected, the precise pose knowledge allows SLAM to update the map and reduce the drift. Also, detecting loop closures avoids the endless corridor issue where the robot assumes that it is always in a new place. Fig. 1.2-a shows that without

loop closure detection, places that in reality are near each other (e.g. B and C in Fig. 1.2-b) are considered distant, and revisits after long excursions (e.g. A in 1.2-b) are always considered as new places.

Figure 1.2: Comparison between SLAM and odometric maps, with trajectory dotted from starting place A to C. Adapted from Cadena et al. (2016).




(a) Map built using only odometry, without loop closure detection.

(b) Map built using SLAM, with loop closure detection.

### 1.1.3 How to recognize a revisited place?

Often considered a key aspect in SLAM algorithms, loop closure detection comprises in performing a search for correspondences between the sensor measurements and the landmarks previously observed during the trajectory (CADENA et al., 2016). However, detecting loop closures is a challenging task, as any false detection caused from issues such as perceptual aliasing can ruin the map building process, and SLAM consequently. Besides that, scalability, sensor noise and changes in the environment are also issues that can affect its performance.

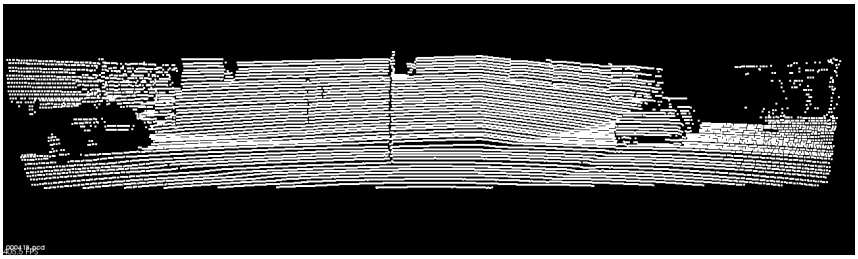
Loop closure detection methods often employ **appearance signature** techniques (ULRICH; NOURBAKHSH, 2000; NEWMAN; COLE; HO, 2006; SÜNDERHAUF; PROTZEL, 2011; HE; WANG; ZHANG, 2016), which are designed to exploit shape, color, texture and other available information from the sensor readings. These signatures are computed and can be used to find correspondences between places visited during the trajectory, being able to recognize two different visits to the same place through similarity metrics (BAILEY; WHYTE, 2006).

The focus of our work is on a 3D vision based loop closure detection approach that uses color information. While there is a significant amount of visual based methods

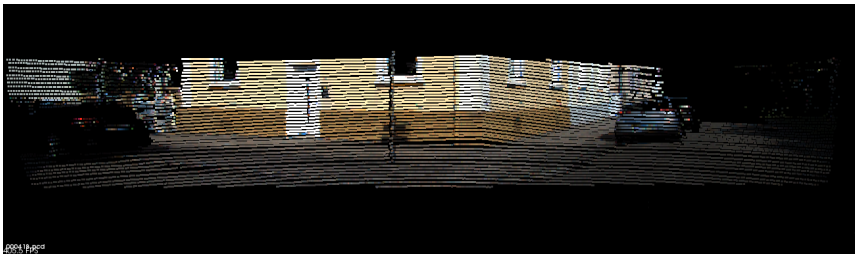
(CUMMINS; NEWMAN, 2008; SÜNDERHAUF; PROTZEL, 2011; MILFORD; WYETH, 2012; LOWRY et al., 2016; NASEER et al., 2015) exploiting 2D camera features to accomplish loop closure detection, few approaches (BOSSE; ZLOT, 2013; RÖHLING; MACK; SCHULZ, 2015; HE; WANG; ZHANG, 2016) employ 3D sensor readings such as LIDARs or stereo cameras. Spatial data captured from these sensors are often used to generate point clouds as seen in Fig. 1.3-b, enabling more descriptive scenes than using only 2D cameras. Besides, many of the current 3D sensors are also equipped with means to acquire color data from the scene, that can be used to generate colored point clouds, as seen in Fig. 1.3-c. Additionally, several works (TOMBARI; SALTI; STEFANO, 2011; FENG; LIU; LIAO, 2015; LOGOGLU; KALKAN; TEMIZEL, 2016) on object recognition and surface matching reports accuracy improvements when adding color information to 3D descriptors. Nevertheless, loop closure detection using descriptors with shape and color information still have not received much attention.

Figure 1.3: Comparison between different sensor readings. Color data provide more descriptive point clouds. Extracted frames from the KITTI dataset (GEIGER; LENZ; URTASUN, 2012).

(a) 2D image.



(b) Point cloud.



(c) Point cloud with color data.

Typical loop closure detection methods for point clouds employ a similarity measure that can be obtained through different matching approaches. Local methods (RUSU; BLODOW; BEETZ, 2009; TOMBARI; SALTI; STEFANO, 2010) can operate around detected keypoints from the point cloud, computing characteristics from each keypoint local neighborhood to build feature descriptors. However, keypoint repeatability, time efficiency and local descriptiveness are still struggling issues for them. Local methods can also directly operate on raw points, either exhaustively for every point when working with roughly aligned scenes (BESL; MCKAY, 1992; SEGAL; HAEHNEL; THRUN, 2009), or point samples when there are larger distances between the clouds (AIGER; MITRA; COHEN-OR, 2008). In comparison, global methods (RUSU et al., 2010; WOHLKINGER; VINCZE, 2011; HE; WANG; ZHANG, 2016) represent the entire cloud geometry into a single descriptor, reducing dimensionality and improving time efficiency. Nevertheless, invariance and relative transformation between point clouds can be challenging to achieve for global methods. Moreover, there are approaches that seek a compromise between local and global features by using hybrid descriptors, or doing offline vector quantization of local descriptors for a 3D bag-of-words (STEDER et al., 2011). Other point cloud matching alternatives can also use specific shapes (MORAL et al., 2013) or objects (MORENO et al., 2013), often requiring segmentation steps, offline classifier training, object recognition, and/or controlled environments to operate.

## 1.2 Objectives

In this work, we propose a global point cloud descriptor named Color M2DP (c-M2DP), which incorporates color information into a state-of-the-art point cloud descriptor, in order to improve loop closure detection performed by an autonomous vehicle equipped with 3D vision sensors and a color camera.

Our proposal is an extension of the state-of-the-art M2DP descriptor (HE; WANG; ZHANG, 2016), which is built using only geometric information from the point cloud, computed by projecting it into multiple 2D planes. Each plane represent a different viewpoint of the point cloud, from which a spatial density distribution is computed. These multiple distributions are in turn singular value decomposed (SVD), with the first left and right singular vectors being used as the final descriptor.

We noticed that the M2DP descriptor can be extended to incorporate color data in order to increase its descriptiveness. Along with the shape information, color distributions

are computed for each 2D projection that represents different viewpoints of the point cloud. These color distributions are concatenated to the M2DP shape signatures and used to build a signature matrix. Then we follow M2DP original steps and employ SVD to reduce the matrix dimensionality, concatenating the first left and right singular vectors and using it as our compact descriptor.

The main contributions of this work are:

- Color M2DP (c-M2DP), a global descriptor comprising of color and shape data for point cloud matching;
- An improved loop closure detection approach using the c-M2DP descriptor on point cloud sequences, generated either by employing camera-LIDAR sensor fusion or a stereo camera, from publicly available outdoor datasets.

## 1.3 Organization

This thesis is structured as follows. First, in Chapter 2, we discuss the theoretical background of this work, reviewing some of the main problems in robot state estimation, with approaches mentioned in the literature and other works that address them. In this chapter we also discuss about the point cloud descriptor M2DP, which is later extended in this dissertation, and present a brief overview of the related work, examining other point cloud descriptors and their respective applications for loop closure detection or other tasks. In Chapter 3, we present c-M2DP, a novel point cloud descriptor that uses color information, detailing the techniques employed while also describing our loop closure detection approach using it. In Chapter 4, we detail our experimenting platform, its settings, the evaluation methods employed, and our experimental results using the c-M2DP descriptor. Lastly, in Chapter 5 we discuss our conclusions about the current work and other possible future endeavors.

# 2 THEORETICAL FOUNDATION

In this chapter we detail concepts and techniques used throughout this dissertation, providing a theoretical background. First, in Section 2.1, we start on the concepts of state estimation in mobile robotics, namely localization and mapping that we discuss further on Sections 2.2 and 2.3, respectively. Then, in Section 2.4, we address the SLAM problem, where we discuss about its implementations in Section 2.4.1, data association in Section 2.4.2, and loop closure detection in Section 2.4.3. After that, in Section 2.5, we present the state-of-the-art Multiview 2D Projection (M2DP) descriptor, the basis in which our work extends from. Finally, we review existing 3D descriptors and loop closure methods that are related with our work in Section 2.6.

## 2.1 State Estimation in Mobile Robotics

In order to accomplish tasks in real-world environments, autonomous mobile robots must gather data of their surroundings through sensors, have the ability to build a representation of the environment and localize themselves in it, along with other objects and potential obstacles. In Table 2.1 we present four variables that are employed in the context of state estimation in mobile robotics that are used in the next sections, where we introduce the concepts of localization, mapping and SLAM.

Table 2.1: List of Symbols used for State Estimation in Mobile Robotics

| Symbol | Meaning |
|---|---|
| $\boldsymbol{x}_t$ | Robot's pose at the instant $t$. The trajectory of the robot is given by $\boldsymbol{x}_{0:t} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_t\}$. |
| $\boldsymbol{m}$ | Map of the environment given by $\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_n\}$, with $n$ landmarks, where $\boldsymbol{m}_i$ is the position of the $i$-th landmark in the map. |
| $\boldsymbol{z}_t$ | Vector of measurements made by the robot at instant $t$ given by $\boldsymbol{z}_t = \{\boldsymbol{z}_t^1, \boldsymbol{z}_t^2, \ldots, \boldsymbol{z}_t^n\}$, where $\boldsymbol{z}_t^i$ is the $i$-th observation at instant $t$. Measurements during the trajectory are given by $\boldsymbol{z}_{0:t} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_t\}$. |
| $\boldsymbol{u}_t$ | Control vector applied to move from pose $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$. The control commands history are given by $\boldsymbol{u}_{1:t} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_t\}$. |

## 2.2 Localization

The ability to known its own pose in the environment is fundamental for an autonomous robot. With knowledge of its own location in relation to surrounding obstacles, the robot is able to avoid them while performing a designed task, that can also require interacting with nearby objects or other entities of interest.

The **Localization** problem estimates the correct pose $\boldsymbol{x}_t$ of the robot at instant $t$, through inference methods using the measurement $\boldsymbol{z}_t$ (STACHNISS, 2006). It assumes that an accurate map $\boldsymbol{m}$ is known a priori, described in a global coordinate system representing the environment. As there is not a direct way to sense the pose, sensor measurements are integrated over time to be able estimate it. According to Thrun et al. (2005), localization can be seen as a transformation that obtains a correspondence between a map and the robot's local coordinate system.

Depending on the type of environment it operates, localization can be a difficult task. In **static environments** the robot's pose is the only variable that changes over time, while other environment elements remain fixed. In contrast, **dynamic environments** have objects and other elements (e.g. people) that vary their respective locations and configurations. These changes can be challenging, specially if they become permanent, generating divergences between measurements and the known map.

Thrun et al. (2005) divided localization in different problems with varying difficulty, depending if there is an initial knowledge of the pose or not. **Local localization**, or tracking, is a problem characterized by the initial robot's pose being known. It is accomplished by assuming a small pose error as the uncertainty is considered local, surrounding the correct pose. On the other hand, the **global localization** problem is characterized by the initial robot's pose being unknown. It is considered more difficult than the local problem, as the robot can be anywhere in the environment and pose uncertainty cannot be assumed using a local approach.

Also worth mentioning, the **kidnapped robot** is considered a harder variant of the global localization problem (THRUN; BURGARD; FOX, 2005), that can be seen as the robot's ability to recover from failures in localization. It occurs when a robot is unknowingly placed in a different location during its operation, resulting in a wrong pose belief that must be corrected.

## 2.3 Mapping

Autonomous mobile robots need to acquire and maintain a model of their surroundings. While navigating in an unknown environment, they must build a map representing the information gathered by their sensors, such as obstacles to avoid, free space and objects of interest.

The **Mapping** problem assumes a correct knowledge of the robot's pose $\boldsymbol{x}_t$, using the associated measurement $\boldsymbol{z}_t$ captured through its sensors at instant $t$, to build a map $\boldsymbol{m}$ representing the environment. In order to properly represent the $\boldsymbol{z}_{0:t}$ measurements, the adopted mapping technique must be suitable to the robot needs, while also being aware of the sensors noise and other perceptual limitations.

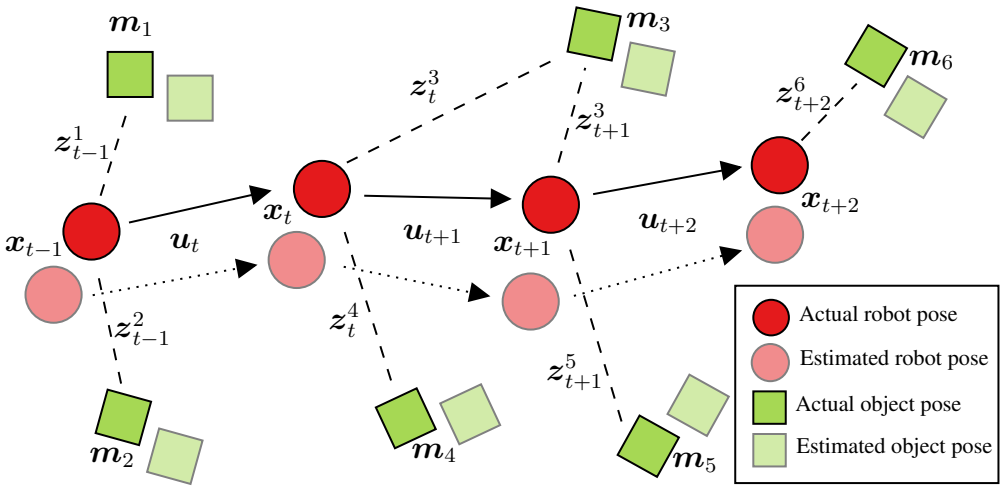
According to Thrun (1998), maps can follow metrical or topological models, or even combine these two approaches in order to better represent an environment. **Metric maps** use either 2D or volumetric 3D grids in a discrete manner to geometrically represent the environment. Each grid cell can store the desired representation properties, such as being an obstacle or free space. They can be easier to build and maintain, robust against viewpoint changes, but have a higher memory footprint. In contrast, **topological maps** use graphs to represent the environment, where nodes can store information, such as being a place already visited or a specific landmark, among other properties, and are connected by arcs if there is a direct path between them. They often allow more efficient planning at lower memory requirements, and can easily employ natural language approaches, but can be difficult to build, maintain, and also more sensitive to viewpoint changes.

## 2.4 SLAM

Simultaneous localization and mapping is considered one of the most challenging problems in robotics (THRUN; LEONARD, 2008; CADENA et al., 2016). In it, an autonomous robot is designed to navigate through a partially known or totally unknown environment $\boldsymbol{m}$, starting from a known pose $\boldsymbol{x}_0$. As the robot moves by applying the controls $\boldsymbol{u}_{1:t}$ (i.e. odometry), it can sense the environment through measurements $\boldsymbol{z}_{0:t}$, but the uncertainty of its pose $\boldsymbol{x}_t$ grows over time. As described by Thrun and Leonard (2008), the **SLAM** problem comprises in simultaneously estimating both the pose $\boldsymbol{x}_t$ and the map $\boldsymbol{m}$, by using the observed sensor measurements $\boldsymbol{z}_{0:t}$ and the controls $\boldsymbol{u}_{1:t}$ made by the robot during its trajectory.

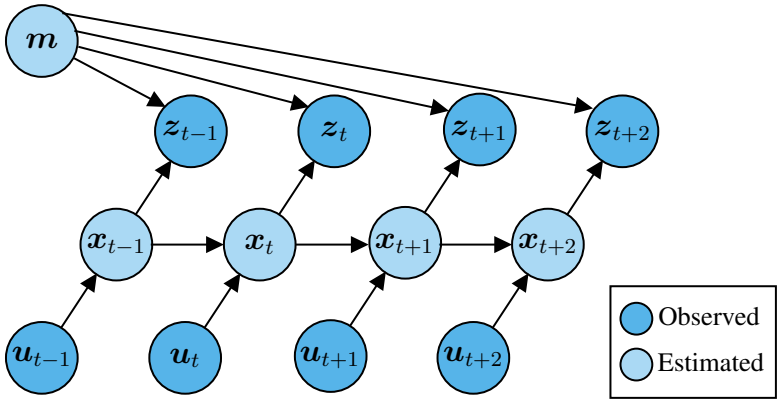
An example of the SLAM problem can be seen in Fig. 2.1. The robot observes landmarks $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ at the instant $t-1$, estimating the poses of both based on the measurements $\boldsymbol{z}_{t-1}^1$ and $\boldsymbol{z}_{t-1}^2$, respectively. Then, the control $\boldsymbol{u}_t$ is applied to move the robot from pose $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$. In this step, landmarks $\boldsymbol{m}_3$ and $\boldsymbol{m}_4$ observed at instant $t$ have their poses estimated through measurements $\boldsymbol{z}_t^3$ and $\boldsymbol{z}_t^4$, followed by robot movement from pose $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ using the control $\boldsymbol{u}_{t+1}$. The process is subsequently repeated, with the robot observing $\boldsymbol{m}_3$ again and $\boldsymbol{m}_5$ at instant $t+1$, until estimating the pose of $\boldsymbol{m}_6$, observed at instant $t+2$.

Figure 2.1: Example of the SLAM problem, with map $\boldsymbol{m}$ and robot's poses $\boldsymbol{x}_{t-1:t+2}$ being estimated through measurements $\boldsymbol{z}_{t-1:t+2}$ and controls $\boldsymbol{u}_{t:t+2}$. The actual robot's poses and map are unknown. Adapted from Durrant-Whyte and Bailey (2006).




Additionally, a graphical model of the SLAM problem is shown in Fig. 2.2, illustrating the causal relationship between the variables. By using the observed variables, which are the measurements $\boldsymbol{z}_{0:t}$ and controls $\boldsymbol{u}_{1:t}$ made by the robot, SLAM aims to estimate the poses $\boldsymbol{x}_{0:t}$ and the map $\boldsymbol{m}$.

Figure 2.2: Graphical model of the SLAM problem. Arcs indicate causal relationships between variables. Adapted from Thrun et al. (2005).



The SLAM problem difficulty comes from the tight coupling of building a map and self-localizing at same time. Siegwart and Nourbakhsh (2004) described it as a chicken-and-egg problem, where a correct pose knowledge is needed to accurately build a map, while an accurate map is required in order to perform self-localization and correctly estimate the robot's pose.

Although there is a wide range of SLAM implementations (DURRANT-WHYTE; BAILEY, 2006; CADENA et al., 2016), Thrun and Leonard (2008) identified three basic paradigms from which most of the others are derived, the extended Kalman filter (EKF) SLAM, the graph-based SLAM and the particle filter SLAM, that we summarize briefly:

- **EKF SLAM** is the earliest of them. It consists in using a single state vector of estimated poses, a set of environment landmarks, and an associated error covariance matrix for these estimates. An EKF update both the state vector and covariance matrix. New landmarks are added to the state vector, while the covariance matrix grows quadratically.
- **Graph-based SLAM** has every landmark and robot's pose as nodes in a graph. Arcs between consecutive poses indicate a control command applied for movement, and arcs between poses and landmarks indicate sensing at the same instant. As arcs in this graph are soft constraints, estimating the map and trajectory is done by relaxing them.
- **Particle filter SLAM** provides a representative sample from the posterior distribution through a set of particles, where each of them can be seen as a pose guess. Increasing the number of particles approximate the filter to the true posterior, but also scale exponentially with the state vector dimensions.

### 2.4.1 Implementations

In the literature, Thrun and Leonard (2008) distinguished SLAM by considering it through different dimensions. For instance, depending if the variables are estimated only at instant $t$, or for the whole trajectory, the SLAM problem is distinguished in two main forms. **Online SLAM** algorithms seeks to estimate a posterior probability over the current robot's pose $\boldsymbol{x}_t$, often being incremental. In contrast, **full SLAM** algorithms seeks to estimate a posterior probability over the entire trajectory $\boldsymbol{x}_{0:t}$, along with the map $\boldsymbol{m}$.

Additionally, in **passive SLAM** the robot is passively controlled by another agent,

while in **active SLAM** the robot uses exploration methods to pursue an accurate map. **Single** and **multi-robot SLAM** are very differentiable due to the interactions between multiple robots. Static or dynamic environments, and metric or topological mapping, mentioned in Section 2.2 and Section 2.3 respectively, are also distinguishing factors. In regards to the map density, **volumetric SLAM** approaches employ high dimensional maps that can allow photorealistic reconstructions depending on the resolution adopted, at the cost of being a heavy computational burden to build and maintain them. In contrast, **feature-based SLAM** build maps using only sparse features extracted from the environment, often prioritizing speed instead of precision, and can be limited to environments for which the features are designed (BOSSE; ZLOT, 2008).

Furthermore, sensors can have a strong influence in distinguishing SLAM implementations. For instance, in **visual SLAM**, cameras are the only exteroceptive[1] sensors employed. Often inexpensive and embeddable for robots, cameras can provide appearance information from the environment, while also enabling the use of techniques such as object recognition. However, they can be sensitive to lighting changes, occlusion, fast movements and featureless environments (FUENTES-PACHECO; RUIZ-ASCENCIO; RENDÓN-MANCHA, 2015). In the particular case of **monocular SLAM**, the lack of depth information from the camera results in issues during landmark initialization and scale ambiguity, differently from **stereo SLAM** or **RGB-D SLAM**, which are able to provide depth through triangulation or structured light, respectively (FUENTES-PACHECO; RUIZ-ASCENCIO; RENDÓN-MANCHA, 2015).

In contrast, **LIDAR-based SLAM** approaches employ laser range finders, which have the ability to provide precise range information from the environment, even for featureless surfaces and independently of lighting conditions (NÜCHTER et al., 2007). Among the earliest sensors used for SLAM, very robust 2D LIDAR-based algorithms were developed over the years (DURRANT-WHYTE; BAILEY, 2006). Recently, 3D LIDAR-based SLAM approaches have been attracting more attention, especially for autonomous vehicle applications (CADENA et al., 2016). Their precise range information can be used to build point clouds, which are particularly useful for applications that require accurate 3D representations. However, they are generally an expensive type of equipment, and processing such amounts of often sparse data can be a challenging task with high computational costs (DUBÉ et al., 2018).

Finally, having the ability to sense the identity of landmarks is another distinguish-

[1] Sensors that acquire information from the environment.

ing factor. In general, SLAM implementations are unable to directly identify landmarks and employ estimation methods in order to perform data association, which we discuss more about in Section 2.4.2. The degree of uncertainty allowed by the system is also considered, with some SLAM algorithms being able to handle only small amounts of error in a simple environment, while others must accumulate high uncertainties during excursions on more complex environments with large loops. Such uncertainties can be reduced by detecting these loop closures, being further discussed in Section 2.4.3

### 2.4.2 Data Association

In order to merge new data into the current map representation, SLAM algorithms require a reliable method for estimating correspondences between their sensor measurements and the observed landmarks. Known as the **data association** problem, it is considered a difficult task for SLAM, as incorrect associations between measurements and existing map landmarks often cannot be revised, resulting in wrong map estimations that can be irrecoverable (BAILEY; WHYTE, 2006).

Perceptual aliasing is one of the main issues for data association methods. Cadena et al. (2016) described it as challenging problem that occurs when different sensory inputs are computed and result in the same measurement. These ambiguous measurements can be erroneously associated with landmarks, resulting in false positive match result. Therefore, an incorrect association generates an inconsistent map, which in turn causes self-localization to fail. False negatives, on the other hand, occur when associations between measurements and landmarks are incorrectly rejected, which can reduce the accuracy, as fewer measurements are used for estimation (CADENA et al., 2016).

According to Bailey and Durrant-Whyte (2006), early data association methods verified correspondences between measurements and landmarks individually, being considered unreliable in most types of real environments and when pose uncertainty is high, as they did not examine the geometric relationships with other landmarks. In contrast, robust SLAM implementations employ batch validation gates (NEIRA; TARDOS, 2001; NEIRA; TARDOS; CASTELLANOS, 2003), which verify if the associations are mutually compatible by using the geometric relationship between landmarks. When sufficiently constrained, they can avoid or significantly diminish the effects of association errors and poor correspondences, at the cost being more computationally expensive (BAILEY; WHYTE, 2006; BOSSE; ZLOT, 2008).

Data association methods are strongly influenced by the map representation adopted by the SLAM approach (BOSSE; ZLOT, 2008). For instance, batch validation gates and appearance signatures (ULRICH; NOURBAKHSH, 2000; NEWMAN; COLE; HO, 2006) are usually employed in feature-based SLAM. Originated from image database indexing techniques, these appearance signatures are generally built with vision based approaches, such as computing histograms using shape, color, texture and other available information from the sensor readings. In these methods, correspondences between measurements and landmarks in the map are found by matching signatures using similarity metrics. Although they can provide additional discriminative information to assist a validation gate method, appearance signatures are generally faster, which makes them being used more often for long-term data association (BAILEY; WHYTE, 2006).

According to Cadena et al. (2016), the architectures of current SLAM algorithms are generally composed by front- and back-end modules. The front-end module abstracts data from the robot sensors to perform data association, which can be divided in short- and long-term approaches. The front-end provides information models to the back-end module, which uses it to perform localization, estimate the trajectory and build the map. Additionally, the back-end module can provide feedback to support loop closure detection and perform verification of it for the front-end.

Short-term data association methods are responsible for detecting and tracking landmarks in the environment, associating them with the corresponding measurements from consecutive sensor readings. They are designed for work in real-time during movement, but can rapidly accumulate errors when large motions occur. In contrast, long-term methods are designed to perform a correspondence search over the entire known map, which can be a costly task, but can reduce the amount of accumulated errors generated during the robot movement. As result, some implementations adopt delayed approaches, or even perform it offline (CADENA et al., 2016). Finally, long-term data association methods are also often employed in loop closure detection, a task that is critical for SLAM and is further discussed in Section 2.4.3.

### 2.4.3 Loop Closure Detection

After a long excursion, an autonomous robot that returns to a previously visited place must have the ability to recognize it as a revisit. Detecting the occurrence of loop closures is essential to build a correct representation of the environment during SLAM.

Otherwise, the robot would build a map without loops, in which the world is seen as an infinite corridor. However, as errors accumulate during long trajectories, they can increase pose uncertainty and drifting in the map up to a level that short-term data association methods are often unable to recognize a place being revisited.

**Loop closure detection** is an extremely important task for SLAM, being responsible for associating new measurements with older landmarks in the map. It is a long-term case of data association (CADENA et al., 2016), employing more globally applicable matching techniques instead of local ones (BOSSE; ZLOT, 2008). Detecting loop closures allows correcting the pose knowledge after long excursions, which in turn enables the map to be updated accordingly, achieving an accurate representation of the environment (CADENA et al., 2016).

According to Granström et al. (2011), loop closure detection can be seen as a place recognition problem applied for robotics, in which a correspondence search is done between the sensor measurements and landmarks previously observed during the trajectory in order to find loop closures. Cieslewski et al. (2016) expressed place recognition as a function

$$\boldsymbol{x} \in \boldsymbol{Q} \rightarrow f(\boldsymbol{x}) \in \boldsymbol{D}, \tag{2.1}$$

where a query set is denoted by $\boldsymbol{Q}$, and a database set by $\boldsymbol{D}$. Place recognition can be distinguished in either a localization done between places of two trajectories, one being database $\boldsymbol{D}$ and the other being query $\boldsymbol{Q}$, or a loop closure detection done within a single trajectory (CIESLEWSKI et al., 2016). In the latter case, both query and database sets are defined by a given instant $t$ and the pose $\boldsymbol{x}_t$ at that same instant,

$$\boldsymbol{Q}_t = \{\boldsymbol{x}_t\} \tag{2.2}$$

$$\boldsymbol{D}_t = \{\boldsymbol{x}_{t'} \mid t' < t - \Delta t\}, \tag{2.3}$$

where $\Delta t > 0$, and is a minimal difference used to avoid self queries.

According to Lowry et al. (2016), detecting loop closures is a challenging task that must address high levels of ambiguity, possible environment symmetries and sensor noise. Perceptual aliasing is a main issue, occurring when distinct places look similar enough that a false positive loop closure is detected. False loop closures are disastrous for the map building process, resulting in inconsistencies that can be irrecoverable for SLAM. Additionally, a place also may not be visited from the same viewpoint as before. Environment changes over time can also be very difficult to address, as places can be

drastically changed by dynamic elements, illumination, weather and seasons. Finally, as the map size increases, scalability can also be an issue due to the number of comparisons and dimensionality needed for an efficient correspondence search.

#### *2.4.3.1 Visual Loop Closure Detection*

Over the past decade, several visual loop closure detection methods were proposed (FUENTES-PACHECO; RUIZ-ASCENCIO; RENDÓN-MANCHA, 2015), operating in different map representations using image matching techniques. For instance, direct methods employ a dense analysis based on pixels intensities of the whole image, providing information even from image regions with small gradients (MILFORD; WYETH, 2012). They can be robust against motion blur, defocus and poor texture in images, but are generally more costly to compute. As an alternative, semi-dense methods employ the same analysis but only for regions with strong gradients, such as edges (ENGEL; SCHÖPS; CREMERS, 2014).

On the other hand, feature-based methods (ULRICH; NOURBAKHSH, 2000; MURILLO; KOSECKA, 2009; SÜNDERHAUF; PROTZEL, 2011) can compute gradients, intensities, or other image characteristics from features detected in the environment, such as corners, points or lines. They are generally optimized for speed rather than precision, are dependent on feature availability of the scene and often rely on detection and matching thresholds (CADENA et al., 2016). Moreover, inspired by text document analysis, bag-of-visual-words methods (LÓPEZ; TARDÓS, 2011) employ offline feature quantization in order to measure term frequencies in images, having shown reliable performance (CUMMINS; NEWMAN, 2008) in detecting loop closures.

Finally, visual loop closure detection methods can also use 3D shapes (MORAL et al., 2013) or objects (MORENO et al., 2013) detected in the environment. However, these methods require additional segmentation steps and controlled indoor environments to operate, and often employ recognition techniques based on offline trained classifiers.

#### *2.4.3.2 3D LIDAR-Based Loop Closure Detection*

Loop closure detection methods using 3D LIDARs are still insufficiently investigated, having not reached the same level of maturity as some visual methods (HE; WANG; ZHANG, 2016; DUBÉ et al., 2017). According to Nüchter et al. (2007), earlier 3D LIDAR-based approaches often generated point clouds from LIDAR readings and em-

ployed a costly dense analysis directly on the whole cloud, using registration techniques such as iterative closest point (ICP) (BESL; MCKAY, 1992). In contrast, recent 3D LIDAR-based approaches often employ feature-based methods (BOSSE; ZLOT, 2013; RÖHLING; MACK; SCHULZ, 2015), computing point cloud descriptors using surface normals, point distributions and other characteristics from features detected in the environment. Similar to visual methods, they can be optimized for speed rather than precision, are sensitive to featureless scenes and may rely on detection and matching thresholds.

Our focus on 3D feature-based approaches is due to their efficiency in computing characteristics detected from point clouds. Similar to image descriptors, global or local techniques can be employed to build point cloud descriptors. Global descriptors are computed by taking the entire point cloud geometry and characteristics into account, providing a single descriptor (RUSU et al., 2010; WOHLKINGER; VINCZE, 2011; HE; WANG; ZHANG, 2016). In comparison with local approaches, global descriptors have lower dimensionality and are often faster to compute, at the cost of being less precise, and unable to perform relative transformations between point clouds in general.

On the other hand, local descriptors are computed using the geometry and other characteristics inside a predefined support region around multiple keypoints (RUSU; BLODOW; BEETZ, 2009; TOMBARI; SALTI; STEFANO, 2010; TOMBARI; SALTI; STEFANO, 2011). They are regarded as being more precise than their global counterpart, and often can be used to compute relative transformations between point clouds more efficiently than using the costly direct methods (BESL; MCKAY, 1992). However, they require keypoint detection techniques for point clouds, which still struggle with issues such as keypoint repeatability, when detected regions overlap, local descriptiveness quality, and computing costs. As an alternative, some implementations employ point sampling techniques (BOSSE; ZLOT, 2013), or detect 3D shapes from point clouds, such as segments (DUBÉ et al., 2017), and then compute descriptors from them.

## 2.5 M2DP: Multiview 2D Projection

He, Wang and Zang (2016) proposed the global point cloud descriptor M2DP and applied it in a loop closure detection approach. Instead of performing an analysis in 3D space, which is often done by other descriptors using surface normals (TOMBARI; SALTI; STEFANO, 2010; RUSU et al., 2010) that can be costly to compute for large point clouds, M2DP computes point distributions from multiple 2D projections of the

point cloud. Using the scene spatial information, this approach aims to capture the point cloud intricate details, an ability that other similar descriptors (JOHNSON; HEBERT, 1999; WOHLKINGER; VINCZE, 2011) struggle with. The M2DP descriptor outperformed other state-of-the-art global descriptors by both accuracy and efficiency, in loop closure detection experiments done using various publicly available datasets, such as KITTI (GEIGER; LENZ; URTASUN, 2012).

Algorithm 1 presents how M2DP descriptor is computed, while input parameters and variables used are denoted in Table 2.2. Similar to other global descriptors (RUSU et al., 2010), in M2DP first steps, the centroid of point cloud $\boldsymbol{P}$ is computed, and $\boldsymbol{P}$ is demeaned, with the centroid being used the reference frame origin. Then, in order to achieve rotation invariance, PCA is performed on $\boldsymbol{P}$ points, with the first and second principal components being used to define both the $x$-axis and the $y$-axis respectively, aligning the point cloud.

After the first pre-processing steps, a 2D plane $X$ centered at origin with a normal vector $\boldsymbol{v}$ is defined, where $\boldsymbol{v}$ is characterized by the pair of parameters $[\theta, \phi]$, with $\theta$ being the azimuth angle, and $\phi$ the elevation angle. The point cloud $\boldsymbol{P}$ is projected into multiple 2D planes, where each plane is an unique $X$ generated using distinct $b$ and $q$ angles for the $[\theta, \phi]$ parameters, respectively.

**Algorithm 1** M2DP Descriptor

**Input:** $\boldsymbol{P}$, $b$, $q$, $l$, $h$, $\alpha$ and $\beta$, according to Table 2.2.
**Output:** A descriptor vector $\boldsymbol{d} \in \mathbb{R}^{\alpha+\beta}$.

$\overline{\boldsymbol{P}} = mean(\boldsymbol{P})$, obtaining the centroid of $\boldsymbol{P}$.
$\boldsymbol{P} \leftarrow \boldsymbol{P} - \overline{\boldsymbol{P}}$, demeaning $\boldsymbol{P}$.
Compute the PCA of $\boldsymbol{P}$.
Align the $x$-axis and $y$-axis of $\boldsymbol{P}$ with the 1st and 2nd PCs, respectively.
$Init(\boldsymbol{A}, 0)$, initializing the signature matrix $\boldsymbol{A} \in \mathbb{R}^{\alpha \times \beta}$ with zero values.
**for** $\theta = 0$ to $\pi$ **do**
    **for** $\phi = 0$ to $\frac{\pi}{2}$ **do**
        Build a 2D plane $X$ with normal vector $\boldsymbol{v} = [\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta]^T$.
        Project $\boldsymbol{P}$ onto $X$ to produce $\boldsymbol{P}^X$, where each point $\boldsymbol{p}_i^X = \boldsymbol{p}_i - \frac{\boldsymbol{p}_i^T \boldsymbol{v}}{||\boldsymbol{v}||_2^2}\boldsymbol{v}$.
        Generate the bins on $X$ using $l$ and $h$.
        Compute $\boldsymbol{s}^X$ by counting the points $\boldsymbol{p}_i^X$ inside each bin.
        Augment $\boldsymbol{A}$ by a row with $\boldsymbol{s}^X$.
        $\phi + \frac{\pi}{2q} \rightarrow \phi$.
    **end for**
    $\theta + \frac{\pi}{b} \rightarrow \theta$.
**end for**
$\boldsymbol{A} = USV^T$, run SVD on $\boldsymbol{A}$.
$\boldsymbol{d} = [U_1\ V_1]^T$, where $U_1$ and $V_1$ are the first columns of $U$ and $V$, respectively.

Table 2.2: List of Symbols used for the M2DP descriptor

| Symbol | Meaning |
|---|---|
| $\boldsymbol{P}$ | Point cloud representing a 3D environment, given by $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$, where $n$ is the number of points, and $\boldsymbol{p}_i \in \mathbb{R}^3$ with $\boldsymbol{p}_i$ being the position of the $i$-th point, given in Cartesian coordinates $[x, y, z]^T$. |
| $X$ | 2D Plane with normal vector $\boldsymbol{v}$ expressed as a function of $\theta$ and $\phi$, i.e., $\boldsymbol{v} = [\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta]^T$. |
| $\boldsymbol{P}^X$ | Point cloud $\boldsymbol{P}$ projected into plane $X$, where each point $p_i^X = p_i - \frac{p_i^T \boldsymbol{v}}{\|\|\boldsymbol{v}\|\|_2^2}\boldsymbol{v}$. |
| $\boldsymbol{s}^X$ | Shape signature vector for projection $\boldsymbol{P}^X$. |
| $b$ | Number of azimuth angles $\theta$. |
| $q$ | Number of elevation angles $\phi$. |
| $\alpha$ | Number of viewpoint angles in total, with $\alpha = b \times q$. |
| $l$ | Number of concentric circles. |
| $h$ | Number of shape divisions. |
| $\beta$ | Number of shape bins in total, with $\beta = l \times h$. |
| $\boldsymbol{A}$ | Signature matrix describing the input point cloud. |
| $\boldsymbol{d}$ | Final descriptor vector. |
| $r$ | Radius of a concentric circle. |

For each projection, the 2D plane is divided into $l$ concentric circles centered at the origin, generated with varying radii $[r, 2^2 r, \ldots, l^2 r]$, where $r$ is derived from the maximum radius ($l^2 r$), which is the distance between the farthest point of the cloud and the centroid. Each concentric circle is divided $h$ times, generating $l \times h$ bins. Similar to other descriptors (JOHNSON; HEBERT, 1999) that measure point distributions, the number of points inside each bin is counted. Then, a signature vector $\boldsymbol{s}^X$ that describes the current 2D projection is computed, and the signature matrix $\boldsymbol{A}$ is augmented by a row with $\boldsymbol{s}^X$. After the signatures were generated for every 2D projection of $\boldsymbol{P}$, the singular value decomposition (SVD) of the signature matrix $\boldsymbol{A}$ is computed, reducing its dimensionality and obtaining a compact signature as result. Finally, the first left and right singular vectors are concatenated and used as the M2DP descriptor.

Similar to other works (MILFORD; WYETH, 2012; BOSSE; ZLOT, 2013), M2DP performance in loop closure detection is evaluated by measuring its achieved accuracy during the correspondence search. Input point clouds are queried against a database of M2DP descriptors computed during the trajectory, calculating the $L2$ norm between the descriptors being matched. Loop closures are detected when the most similar of them are found under a predefined threshold, which is later used by the authors to generate precision-recall curves for accuracy evaluation.

## 2.6 Related Work

Loop closure detection methods for point clouds often identify previously visited places by employing a similarity measure between descriptors computed from each scene. For instance, the 3D Gestalt (BOSSE; ZLOT, 2013) is a local descriptor proposed for detecting loop closures in LIDAR-based point clouds. Keypoints are randomly selected from a downsampled point cloud. Bins are generated by radial and azimuthal splits of a cylinder support. Then, both mean and height variance from the points are computed within each bin. Additionally, the Neighbor-binary landmark density (NBLD) (CIESLEWSKI et al., 2016) is a local descriptor inspired by 3D Gestalt, that is also proposed for loop closure detection, but for sparse point clouds generated either by cameras or LIDAR sensors. Instead of heights, NBLD measures point density within bins, generated through radial, azimuthal and vertical splits of a cylinder support around each landmark.

Point cloud descriptors are also employed in other surface matching approaches, such as object recognition. For instance, the well-known local descriptor Signature of Histograms of Orientations (SHOT) (TOMBARI; SALTI; STEFANO, 2010) defines an spherical support grid around each keypoint, after establishing the local reference frame, and splitting the sphere using radial, azimuth and elevation divisions. Then, local histograms are built, accumulating the angle differences between surface normals within the local support and the feature point.

Global point cloud descriptors, such as the Viewpoint Feature Histogram (VFH) (RUSU et al., 2010) can also be applied for object recognition. For VFH, a histogram is built by computing angles differences between estimated normals and the centroid direction. Point locations are not considered, which can lead to ambiguities between surface normals estimated from different point clouds. Extending it, the Clustered Viewpoint Feature Histogram (CVFH) (ALDOMA et al., 2011) removes points with high curvatures that occur in edges or due to noise. Then, a region growing algorithm to identify stable regions from the depth data is applied, in order to compute their angular information and shape distribution. This extension is further improved in the Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) (ALDOMA et al., 2012), that employs the same components from CVFH, but computes histograms of distances between points in order to achieve more robustness.

Also worth mentioning, the Ensemble of Shapes (ESF) (WOHLKINGER; VINCZE,

2011) is a global descriptor designed for object recognition in dense point clouds captured through RGB-D sensors. Instead of surface normals, it uses a voxel grid to approximate surfaces from the point cloud, iterating over point samples of the grid to compute shape angles, distances and areas. The Global Fourier Histogram (GFH) (CHEN et al., 2014) is inspired by the original Spin Images (JOHNSON; HEBERT, 1999), creating a object-centered cylindrical coordinate system and establishing a reference frame. Points within bins, generated through vertical, azimuthal and radial splits, are counted to compute a histogram, which is then analyzed using a Fast Fourier Transform along the azimuth dimension.

Over the past years, the availability of 3D sensors that are able to capture color information have increased, resulting in the development of point cloud descriptors that combine both color and shape data, especially for object recognition applications. For instance, Tombari, Salti and Stefano (2011) proposed an extension of the SHOT descriptor that incorporates color along its original shape data. Named Color-SHOT (CSHOT), it inherits SHOT original parameters but defines an additional vector for its color bins. Then, CSHOT computes the sum of absolute differences between intensities of CIELab triplets, associated to each point within the local support.

Additionally, the Colored Histograms of Spatial Concentric Surflet-Pairs (CoSPAIR) (LOGOGLU; KALKAN; TEMIZEL, 2016) is a local descriptor proposed for object recognition, also built using shape and color. CoSPAIR splits its local support in multiple concentric spheres. Histograms of surface normals angular relations and CIELab channels are computed for each sphere. Recent descriptors specifically designed for RGB-D sensors also combine color and shape, such as the Local Ordinal Intensity and Normal Descriptor (LOIND) (FENG; LIU; LIAO, 2015), that computes angular relations between surface normals inside a support circle around the detected keypoints, while also computing intensities in image pixel distributions of 2D circle patches on the same locations.

In Table 2.3 we summarize the related work, indicating whether they operate globally or locally, compute color information, and if they use surface normals to compute shape data. As reported in object recognition works such as Tombari, Salti and Stefano (2011) and Logoglu, Kalkan and Temizel (2016), incorporating color into a point cloud descriptor can increase its descriptiveness, improving point cloud matching accuracy. However, detecting loop closures using point cloud descriptors that combine color and shape information, such as our proposal, is still an insufficiently investigated topic. Although the 3D Gestalt and NBLD descriptors were proposed for loop closure detection,

they operate locally and compute only shape information from the cloud. Our proposal share similarities with CSHOT and CoSPAIR, as both were also extensions that added color information to descriptors that only computed shape originally. In the following chapter we describe our proposed extension that incorporates color into the M2DP descriptor (HE; WANG; ZHANG, 2016), and then we apply it to a loop closure detection pipeline.

Table 2.3: Summary of the related work, indicating whether they operate globally or locally, and their use of normals and color information.

| **Descriptor** | **Reference** | **Context** | **Normals** | **Color** |
|---|---|---|---|---|
| VFH | Rusu et al. (2010) | Global | Yes | No |
| SHOT | Tombari, Salti and Stefano (2010) | Local | Yes | No |
| CSHOT | Tombari, Salti and Stefano (2011) | Local | Yes | Yes |
| ESF | Wohlkinger and Vincze (2011) | Global | No | No |
| CVFH | Aldoma et al. (2011) | Global | Yes | No |
| OUR-CVFH | Aldoma et al. (2012) | Global | Yes | No |
| 3D Gestalt | Bosse and Zlot (2013) | Local | No | No |
| GFH | Chen et al. (2014) | Global | No | No |
| LOIND | Feng, Liu and Liao (2015) | Local | Yes | Yes |
| CoSPAIR | Logoglu, Kalkan and Temizel (2016) | Local | Yes | Yes |
| NBLD | Cieslewski et al. (2016) | Local | No | No |
| M2DP | He, Wang and Zhang (2016) | Global | No | No |
| c-M2DP | Ours | Global | No | Yes |

# 3 A FAST POINT CLOUD DESCRIPTOR WITH COLOR INFORMATION AND ITS APPLICATION TO LOOP CLOSURE DETECTION

In this chapter we present our proposal, the c-M2DP descriptor, which is an extension that incorporates color information to the state-of-the-art M2DP descriptor. Also, we present how the c-M2DP descriptor was applied in a loop closure detection pipeline designed to evaluate its performance.

In Section 3.1, we introduce the c-M2DP descriptor with a overview, followed by the description of its structure in Section 3.1.1, detailing each of the steps employed in c-M2DP construction. At first, some of these steps remain unchanged in relation to M2DP, such as the reference frame in Section 3.1.2, and the shape signatures in Section 3.1.3. The addition of the color signatures is detailed in Section 3.1.4, followed by how both signatures are used to generate a compact descriptor in Section 3.1.5. Finally, a description about the c-M2DP application to loop closure detection is presented in Section 3.2, with details on the dataset sequences employed for this application in 3.3.

## 3.1 c-M2DP: Color Multiview 2D Projection Descriptor

Point cloud descriptors that incorporate both color and shape information are often proposed as an extension of implementations that only consider shape. They have shown increased descriptiveness in surface matching (TOMBARI; SALTI; STEFANO, 2011) and object recognition (LOGOGLU; KALKAN; TEMIZEL, 2016). However, their application for loop closure detection it is still insufficiently investigated, despite the potential improvements. For instance, we have noticed that the state-of-the-art M2DP descriptor (HE; WANG; ZHANG, 2016) presents a design that can be easily extended to also consider color information, without significantly compromising its conciseness and efficiency.

The M2DP descriptor provides a compact description for a LIDAR-based point cloud by projecting it into multiple 2D planes, and computing a shape signature from each of these projections. Our proposal takes advantage of the existing structure to incorporate the additional color signatures, which are also computed from each 2D projected point cloud with associated color information.

### 3.1.1 c-M2DP Algorithm

Algorithm 2 presents how the c-M2DP descriptor is computed, and was developed by modifying M2DP's Algorithm 1, extending it with the additional steps required to compute color data. It inherits the parameters and variables from the original M2DP descriptor, which are denoted in Table 2.2, while also requiring additional symbols that are presented in Table 3.1.

In a brief summary, Algorithm 2 projects an input point cloud $\boldsymbol{P}$ with associated color information onto multiple 2D planes, which are uniquely generated using a defined 2D plane $X$ and distinct $b$ azimuth angles and $q$ elevation angles. Then, we compute both the shape $\boldsymbol{s}^X$ and our proposed color $\boldsymbol{c}^X$ signatures from each projection $\boldsymbol{P}^X$, using them to build the signature matrix $\boldsymbol{A}$. Finally, we follow M2DP with a dimensionality reduction step, computing the SVD of $\boldsymbol{A}$ and obtaining a compact descriptor as result. The first left and right singular vectors of the signature matrix are used as the final c-M2DP descriptor.

**Algorithm 2** c-M2DP Descriptor

**Input:** $\boldsymbol{P}$, $b$, $q$, $l$, $h$, $w$, $g$, $\alpha$, $\beta$, $\gamma$ according to Tables 2.2 and 3.1.
**Output:** A descriptor vector $\boldsymbol{d} \in \mathbb{R}^{\alpha+\beta+\gamma}$.

1: $\overline{\boldsymbol{P}} = mean(\boldsymbol{P})$, obtaining the centroid of $\boldsymbol{P}$.
2: $\boldsymbol{P} \leftarrow \boldsymbol{P} - \overline{\boldsymbol{P}}$, demeaning $\boldsymbol{P}$.
3: Compute the PCA of $\boldsymbol{P}$.
4: Align the $x$-axis and $y$-axis of $\boldsymbol{P}$ with the 1st and 2nd PCs, respectively.
5: $Init(\boldsymbol{A}, 0)$, initializing the signature matrix $\boldsymbol{A} \in \mathbb{R}^{\alpha \times (\beta+\gamma)}$ with zero values.
6: **for** $\theta = 0$ to $\pi$ **do**
7:  **for** $\phi = 0$ to $\frac{\pi}{2}$ **do**
8:  Build a 2D plane $X$ with normal vector $\boldsymbol{v} = [\cos\theta\cos\phi, \cos\theta\sin\phi, \sin\theta]^T$.
9:  Project $\boldsymbol{P}$ onto $X$ to produce $\boldsymbol{P}^X$, where each point $\boldsymbol{p}_i^X = \boldsymbol{p}_i - \frac{\boldsymbol{p}_i^T \boldsymbol{v}}{||\boldsymbol{v}||_2^2}\boldsymbol{v}$.
10:  Generate the shape bins on $X$ using $l$ and $h$.
11:  Compute $\boldsymbol{s}^X$ by counting the points $\boldsymbol{p}_i^X$ inside each shape bin.
12:  Generate the color bins on each color channel using $l$, $w$ and $g$.
13:  Compute $\boldsymbol{c}^X$ by concatenating color histograms of each concentric circle.
14:  Augment $\boldsymbol{A}$ by a row with the concatenation of $\boldsymbol{s}^X$ with $\boldsymbol{c}^X$.
15:  $\phi + \frac{\pi}{2q} \rightarrow \phi$.
 **end for**
16:  $\theta + \frac{\pi}{b} \rightarrow \theta$.
**end for**
17: $\boldsymbol{A} = USV^T$, run SVD on $\boldsymbol{A}$.
18: $\boldsymbol{d} = [U_1\ V_1]^T$, where $U_1$ and $V_1$ are the first columns of $U$ and $V$, respectively.

Table 3.1: List of Additional Symbols used for the c-M2DP descriptor

| Symbol | Meaning |
|---|---|
| $\boldsymbol{c}^X$ | Color signature vector for projection $\boldsymbol{P}^X$. |
| $w$ | Number of color channels. |
| $g$ | Number of color space bins per color channel. |
| $\gamma$ | Number of color bins in total, with $\gamma = l \times w \times g$. |

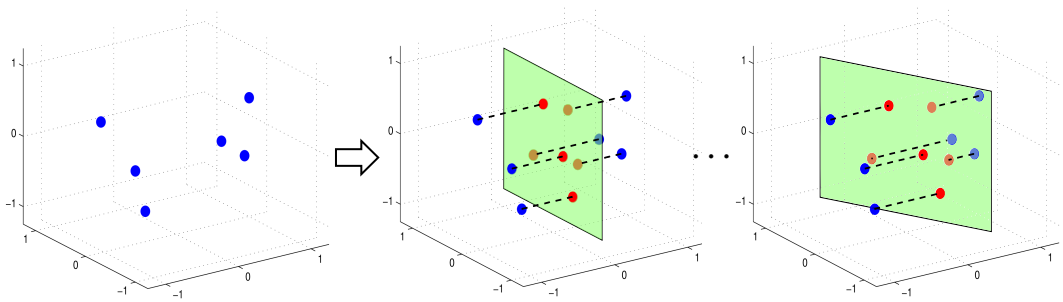
### 3.1.2 Reference Frame

The pre-processing steps are responsible for defining the c-M2DP reference frame origin and alignment. These steps remain the same as originally proposed for the M2DP descriptor, as they are essential to achieve rotation and shift invariance in 3D space, allowing coarse alignments between point clouds that are being matched.

At first, the centroid of an input point cloud $\boldsymbol{P}$ is computed in order to demean $\boldsymbol{P}$ (lines 1-2). Similar to other global descriptors (RUSU et al., 2010), the centroid is used as the descriptor reference frame origin. Then, PCA is performed on $\boldsymbol{P}$ points, defining both the first and second principal components respectively as the $x$-axis and $y$-axis of the descriptor reference frame (lines 3-4).

Once the reference frame is established, we define a 2D plane $X$ with the normal vector $\boldsymbol{v}$, with $\boldsymbol{v}$ being expressed as a function of its azimuth angle $\theta$ and elevation angle $\phi$ (line 8). Centered at the origin, $X$ represents a viewpoint defined by the pair of parameters $[\theta, \phi]$, which are used to generate multiple and unique 2D planes. Then, as seen in Fig. 3.1, we project $\boldsymbol{P}$ onto each of these generated planes (line 9), in order to compute shape and color signatures from each 2D projection.

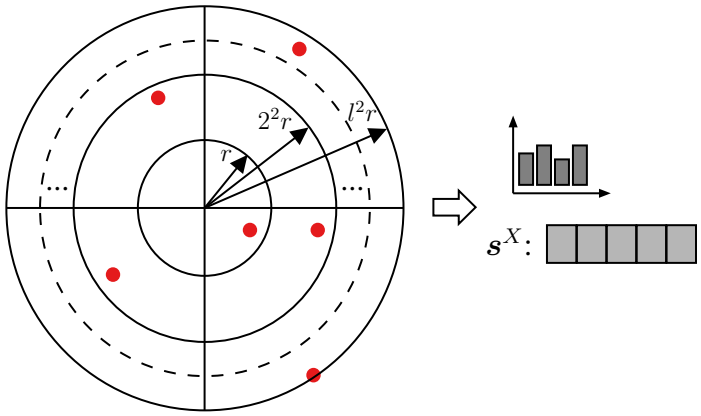
Figure 3.1: Projecting a point cloud $\boldsymbol{P}$ on multiple 2D planes.



### 3.1.3 Shape Signature

As the M2DP descriptor has the ability to provide accurate shape descriptions, that outperforms other state-of-the-art descriptors while maintaining a low computational cost, we chose to maintain the same process, computing shape signatures from each 2D projection of $\boldsymbol{P}^X$. Each 2D plane is split into $l$ concentric circles centered at the origin. They are generated with varying radii $[r, 2^2r, \ldots, l^2r]$, where $r$ is derived from the maximum radius ($l^2r$), which is the distance between the farthest point of the cloud and the centroid. Then, each concentric circle is divided in $h$ bins, indexed by the $x$-axis, generating a total of $l \times h$ shape bins (line 10), as shown in Fig. 3.2. Similar to other approaches (JOHNSON; HEBERT, 1999; CIESLEWSKI et al., 2016), the shape signature $\boldsymbol{s}^X$ is computed by counting the points that lie inside each shape bin (line 11).

Figure 3.2: A 2D plane with a point cloud $\boldsymbol{P}$ projected onto it. The plane is split into $l$ concentric circles, which in turn are divided in $h$ shape bins to compute the shape signature $\boldsymbol{s}^X$.



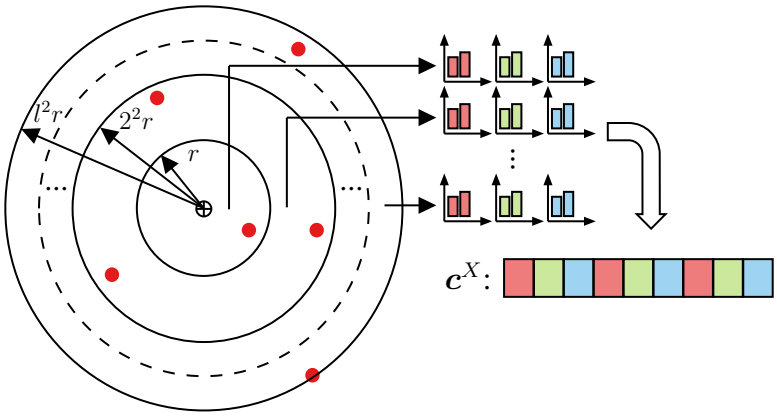

According to He, Wang and Zang (2016), point distributions can provide an accurate description of the point cloud geometric information, while being very efficient and less prone to noise in comparison with other descriptors (TOMBARI; SALTI; STEFANO, 2010; RUSU et al., 2010) that are based on surface normals. Shape matching approaches usually employ normals due to the rich information they can provide. However, estimating surface normals can be a costly process for large point clouds, often risking loss of information by requiring a downsampling step to reduce the number of points.

### 3.1.4 Color Signature

Our proposal aims to extend the M2DP descriptor by incorporating color information alongside the shape signatures. An intuitive approach is to build color signatures by taking advantage of M2DP existing structure, computing the color data associated with each point from the multiple 2D projections, in order to obtain color distributions through the point cloud.

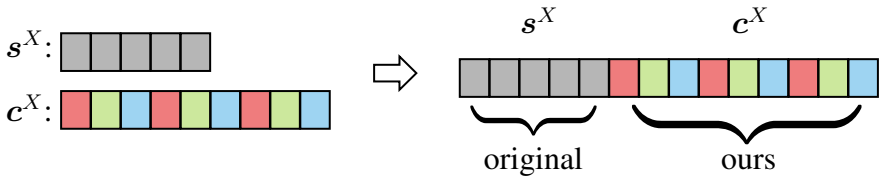
After splitting the 2D projected point cloud in multiple concentric circles and computing the shape signature $\boldsymbol{s}^X$, we can employ a similar process to build a color signature, by computing color histograms for each of the concentric circles, as illustrated in Fig. 3.3. A similar technique is employed by the CoSPAIR descriptor (LOGOGLU; KALKAN; TEMIZEL, 2016), which operates in a local 3D sphere support region split in multiple concentric spheres, and computes histograms of the color channels in RGB space for each concentric sphere for its color component.

Figure 3.3: A 2D plane with a point cloud $\boldsymbol{P}$ projected onto it. The plane is split into $l$ concentric circles. Color histograms are computed for each concentric circle and concatenated into color signature $\boldsymbol{c}^X$.




As implemented in Algorithm 2, once we have the 2D projection $\boldsymbol{P}^X$ split into $l$ concentric circles, we divide each channel of the color space in $g$ bins, generating a total of $l \times w \times g$ bins, where $w$ is the number of channels (line 12). Then, for every concentric circle, we compute histograms of the color channels and concatenate them into a single color signature vector $\boldsymbol{c}^X$ (line 13). After obtaining both shape and color signatures from the 2D projected point cloud, we normalize and concatenate them into a single signature vector, as shown in Fig. 3.4. Thus, we augment the signature matrix $\boldsymbol{A}$ by a row using the concatenated vector (line 14).

Figure 3.4: Concatenated shape and color signatures computed from projection $\boldsymbol{P}^X$.




### 3.1.5 Dimensionality Reduction

Once both shape and color signatures are generated, concatenated and included into the signature matrix $\boldsymbol{A}$ for every 2D projection, we begin the last steps to build the c-M2DP descriptor. At this point, despite the multiple viewpoints from the point cloud being represented in $\boldsymbol{A}$, it still is not a suitable descriptor. Its high memory footprint can be very costly for the storage requirements of an entire point cloud dataset. Also, due to its size, the matching process between descriptors would be slow during the correspondence search.

We maintain the same steps adopted by M2DP in order to achieve a compact descriptor. By computing the SVD of signature matrix $\boldsymbol{A}$, its dimensionality can be reduced (line 17). The first left and right resulting singular vectors of $\boldsymbol{A}$ are concatenated in the final step, producing the c-M2DP descriptor (line 18). As expected, in comparison with the M2DP descriptor, c-M2DP requires an increase on its size due to the addition of color signatures. However, as our proposal makes use of the existing structure to compute these color signatures, we avoid an unnecessary increase in computational costs while being able to represent more information from the point cloud.

## 3.2 Loop Closure Detection using c-M2DP

Besides the proposed point cloud descriptor, our proposal also involves applying c-M2DP to the loop closure detection problem in order to evaluate its performance. Similar to other visual and 3D LIDAR-based loop closure detection approaches (MILFORD; WYETH, 2012; BOSSE; ZLOT, 2013; CIESLEWSKI et al., 2016; HE; WANG; ZHANG, 2016), we developed a pipeline that, in summary, receives a query frame, computes the descriptor from it and uses the result to perform a correspondence search in a database.

Our first step comprise in loading the query set with the frames from a entire

sequence. Each frame is an input point cloud with associated color information, generated from the current sensors being employed for evaluation. As our goal is to detect loop closures within a single trajectory, the database set is composed by c-M2DP descriptors computed for each frame captured through the dataset sequence. Then, for every input point cloud, a c-M2DP descriptor is computed and used as a query for the correspondence search for loop closures is done within the descriptors database. In this process, loop closure detection comes down to finding the most similar descriptors between different point clouds.

We employed a simple brute-force approach for the search, with the matching between two descriptors being done using the $L2$ norm. A match is determined as a loop closure if the calculated $L2$ distance between two descriptors are under a confidence value threshold $v$, which is later used to generate our precision-recall curves for evaluation. Also, in order to avoid self-queries between a query and its match during the matching process, we set up an window to exclude adjacent frames from the query, considering a minimal difference between the current query and its neighbors.

Before we are able to run the loop closure detection process, we need to perform an offline pre-processing of the dataset sequences in order to generate the input point clouds. Although there are several public available datasets (PANDEY; MCBRIDE; EUSTICE, 2011; GEIGER; LENZ; URTASUN, 2012) recorded in distinct environments and employing various sensors, such as 3D LIDARs and stereo cameras, point clouds with associated color information are not usually provided ready for use by them. However, they can be generated through different techniques and tools, which we will further discuss in the following section.

### 3.3 Dataset Sequences

We chose to evaluate the c-M2DP descriptor performance in the KITTI odometry dataset 00, 05, 06 and 07 sequences (GEIGER; LENZ; URTASUN, 2012), as they were also used for evaluating the M2DP descriptor. Additionally, the KITTI dataset have color data available through its image frames, and provides an extensive documentation about its resources and also have very useful tools for various tasks, such as generating colored point clouds using its sensors.

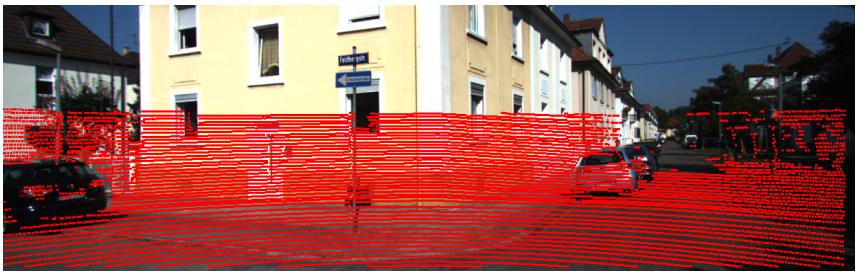
The KITTI dataset sequences were recorded using a variety of sensors simultaneously, including a Velodyne LIDAR with a field-of-view (FoV) of $360^{\circ}$ and a stereo Point

Grey Flea color camera system facing forward, both providing synchronized frames at 10 Hz, with the images being rectified. With these sensors, we were able to generate colored point clouds offline by either performing sensor fusion between the LIDAR and the left color camera, or through depth estimation using the stereo camera. We generated both for each sequence, as it enabled us to evaluate the c-M2DP performance on point clouds generated from different types of sensors. The performance of M2DP, and consequently c-M2DP, are still unknown on point clouds generated from stereo estimated depth, which are subject to distinct point densities and noise when compared with the ones generated from 3D LIDARs readings.

### 3.3.1 Camera-LIDAR Sensor Fusion

We used the kitti_lidar_camera[1] ROS package, and generated the point clouds using KITTI 3D LIDAR readings with associated color information captured by the left color camera for all sequences. Although the 3D LIDAR provides a 360° scan, we had to limit its readings due to the forward facing FoV of the color cameras, generating a subset point cloud in order to perform the sensor fusion. After that, we converted the coordinate system of the subset, projecting the 3D points into the 2D camera image, as shown in Fig. 3.5, by using the LIDAR-to-camera translation matrix provided by KITTI. Finally, using the color data of the 2D projection, we associated each color value with their respective 3D point, producing a color enriched point cloud. It is also worth mentioning that the subset point clouds generated without color were used in order to evaluate the M2DP descriptor in similar conditions with our proposal.

Figure 3.5: 3D LIDAR points projected on 2D image. Frame from the KITTI odometry dataset.



[1] https://github.com/LidarPerception/kitti_lidar_camera

### 3.3.2 Stereo Depth Estimation

Besides generating point clouds through camera-LIDAR sensor fusion, we also employed stereo depth estimation to generate point clouds with color information. The stereo camera images are provided in grayscale as input for the image_undistort[2] ROS package, that in turn uses the block matching technique from OpenCV[3] StereoBM algorithm to perform depth estimation. We maintain the default parameters for depth estimation of KITTI dataset sequences, as it produces results such as shown in Fig. 3.6. Finally, after having both the estimated depth and the color image, we are able to generate a colored point cloud using the scale and camera parameters provided by the KITTI dataset.

Figure 3.6: Depth estimated from stereo camera. Frame from the KITTI odometry dataset.

[2]https://github.com/ethz-asl/image_undistort
[3]https://opencv.org/

# 4 EXPERIMENTS AND DISCUSSION

In this chapter, we describe how the proposed c-M2DP descriptor was experimented and evaluated, discussing its performance results. In Section 4.1 we detail the development process and the tools employed for our experiments. In Section 4.1.1, we present the experiments settings, defining values for the descriptors and pipeline parameters. After that, in Section 4.1.2, we describe how the precision-recall curves were generated, and how the time efficiency was measured. Then, in Section 4.2 we perform an evaluation of the c-M2DP descriptor using RGB, HSV and CIELab color spaces in order to define the best color space to be used. In Section 4.3 we present the computing time costs associated with each descriptor, showing difference in results between the camera-LIDAR and stereo camera sequences. Additionally, in Section 4.4, we introduce our experiments comparing c-M2DP with both M2DP and CSHOT descriptors on the four KITTI dataset sequences. The precision-recall curves along with discussions about each descriptor accuracy are presented both in Section 4.4.1, for sequences generated using camera-LIDAR sensor fusion, and in Section 4.4.2, for sequences generated using stereo estimated depth.

## 4.1 Development and Experiments Details

Our platform for development and running experiments was a laptop equipped with an Intel i7 quad-core 2.00 GHz CPU and 8 GB RAM. Although the M2DP descriptor is provided by its authors in MATLAB[1] code, we chose to convert the code to C++ in order to integrate it in a loop closure detection pipeline, develop our c-M2DP extension, and compare both with another descriptor. For this conversion, we were able to use the Point Cloud Library[2] (PCL) 1.7 and the Eigen[3] Library 3.2, and accomplished a similar performance with equivalent results to the original MATLAB code. Also, our loop closure detection pipeline was developed with C++, using OpenCV[4] 2.4.9 BruteForce algorithm for descriptor matching.

With both the c-M2DP and M2DP descriptors implemented in the same environment, we were able to evaluate if there was an improvement on accuracy due to the addi-

[1] https://www.mathworks.com
[2] http://pointclouds.org
[3] http://eigen.tuxfamily.org/
[4] https://opencv.org/

tion of color information over the shape only descriptor. Also, we were able to measure average computing times of each descriptor in order evaluate their efficiency. Besides, we chose to compare c-M2DP results against the CSHOT descriptor (TOMBARI; SALTI; STEFANO, 2011), which also combines color and shape information computed from the point cloud.

Sharing some similarities with our proposal, the CSHOT descriptor was also proposed as an extension to incorporate color information into a shape-only descriptor. However, CSHOT was originally designed as a local method, being computed around multiple keypoints detected in the point cloud. As previously mentioned, there are several differences between global and local descriptors in regard to their capabilities, dimensionality and overall performance. A common approach (HE; WANG; ZHANG, 2016) for a comparison between them is to employ a global variant of the local method, if available. We modified the C++ implementation provided by the PCL in order to use its global variant for evaluation during our experiments. Instead of using the local neighborhood as support of a reference point, the whole point cloud is used as support of the centroid to compute the CSHOT descriptor.

### 4.1.1 Experiments Settings

Before we perform our experiments using the implemented descriptors in the loop closure detection pipeline, we must determine the values of their respective parameters. In the case of both M2DP and c-M2DP descriptors, these settings are used to build their respective structures, defining the type and amount of splits done in their support regions, which for global descriptors are the whole point cloud.

In their study, He, Wang and Zang (2016) defined the values of M2DP's parameters after performing a benchmark on one dataset sequence, evaluating both accuracy and time efficiency during this process. As shown on Table 4.1, we adopted the same parameter values from the original work for both M2DP and c-M2DP descriptors in our experiments. Also, we adopted the same definitions for the loop closure detection pipeline, by employing a $\pm 50$ frames window relative to the current query frame to avoid self-queries, and by considering a detected loop closure correct, if the two matched point clouds are less than $10m$ from each other in the ground truth trajectory provided by the KITTI dataset.

Additionally, the c-M2DP descriptor requires that the number of color bins for each color channel to be determined. We arbitrarily set it as $g = h$, being the same

number of shape bins per circle. Also, as only color spaces with 3 channels were used in our experiments (i.e. RGB, HSV and CIELab), we fixed $w = 3$. With these parameters and due to the dimensionality reduction, the c-M2DP descriptor vector ends up with $576$ in length, compared to M2DP's original vector length of $192$.

Table 4.1: M2DP and c-M2DP parameters

| **Parameter** | **M2DP** | **c-M2DP** |
|---|---|---|
| Azimuth angles ($b$) | $4$ | $4$ |
| Elevation angles ($q$) | $16$ | $16$ |
| Concentric circles ($l$) | $8$ | $8$ |
| Shape bins ($h$) | $16$ | $16$ |
| Color bins ($g$) | - | $16$ |

In regards to the CSHOT descriptor, we maintain the default parameters of its PCL implementation, which generates a vector with $1344$ in length. It is also worth mentioning that CSHOT and other descriptors (TOMBARI; SALTI; STEFANO, 2010; RUSU et al., 2010; LOGOGLU; KALKAN; TEMIZEL, 2016) that require surface normals, need a radius parameter to be able to estimate them. This parameter is usually tuned considering the point cloud density, in order to avoid reaching too many neighbor points and increasing the computational costs, or being insufficient to reach its neighbors. For the CSHOT descriptor, we adopted the same approach employed by He, Wang and Zang (2016) with the SHOT descriptor. Before running each dataset sequence, we define this radius parameter as being $5 \times \nu$, where $\nu$ is the point cloud resolution of the sequence's first frame, calculated by averaging the distance between each point and its nearest neighbor.

### 4.1.2 Evaluation Methods

As mentioned in Chapter 3, a match between two descriptors is considered a loop closure if the $L2$ distance between them is under a threshold $\upsilon$. Then, loop closures detected correctly are considered True Positives (TP), while incorrect ones are False Positives (FP), and the loop closures wrongly discarded are False Negatives (FN). Similar to other works (MILFORD; WYETH, 2012; BOSSE; ZLOT, 2013; GAWEL et al., 2016; HE; WANG; ZHANG, 2016; LOWRY et al., 2016), we evaluated each descriptor accu-

racy by using the relationship between their precision[5] and recall[6] metrics, generating the precision-recall curves. We also evaluated the descriptors efficiency, measuring the time consumed for computing them and during the matching process.

Similar to the precision-recall curves generated in SeqSLAM (MILFORD; WYETH, 2012), we generate ours by varying the $v$ threshold value between all the different $L2$ distances calculated during the matching process. Also, we present the descriptors recall rate at $100\%$ precision, which is an important comparison for loop closure detection methods. Any false positives (i.e. incorrect loop closures) can be disastrous, resulting in map building inconsistencies that can lead a SLAM system into a irrecoverable state.

## 4.2 Color Spaces

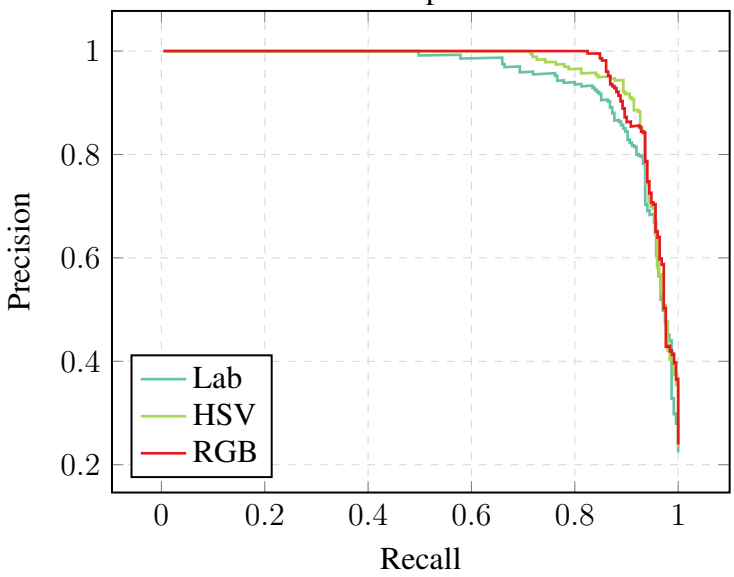
For the first experiment, our goal was to determine which color space the c-M2DP descriptor should use, by performing an evaluation of c-M2DP using the RGB, HSV and CIELab color spaces, and verifying which shows a higher accuracy. A similar approach was done by other descriptors that combined color and shape (TOMBARI; SALTI; STEFANO, 2011; LOGOGLU; KALKAN; TEMIZEL, 2016). For instance, the CSHOT descriptor chose CIELab after a comparison with RGB, while CoSPAIR evaluated the same three options as ours and also adopted CIELab for its histograms. For this experiment we ran the loop closure detection process using c-M2DP with three different color spaces. The KITTI 06 sequence was used, with point clouds generated by the fusion between LIDAR and camera. It is also worth mentioning that for the HSV and CIELab color spaces, a step for converting the point clouds from RGB was implemented.

In Fig. 4.1 we plotted the precision-recall curves from the results of this experiment. It shows that when using RGB, c-M2DP reaches a recall rate of $82.5\%$ at precision $100\%$, after which it begins to drop, reaching a recall rate of $89.2\%$ at $90\%$ precision and then dropping further. Other color spaces presented lower results, with HSV showing a $71.4\%$ recall with precision $100\%$, reaching $91.5\%$ at $90\%$ precision, and CIELab performing poorly with a recall of $49.8\%$ at precision $100\%$, reaching $86.8\%$ at $90\%$ precision. Therefore, due to RGB higher recall rate with precision $100\%$, we adopted it as the color space used for the c-M2DP descriptor.

[5]Precision is the proportion of correctly detected loop closures (TP) among the total of detected loop closures (TP+FP)

[6]Recall is the proportion of correctly detected loop closures (TP) among the actual loop closures in the sequence (TP+FN)

Figure 4.1: Precision-recall curves for KITTI 06 sequence comparing c-M2DP with different color spaces.




### 4.3 Time Efficiency

Maintaining a good balance between accuracy and efficiency is a challenging task for any point cloud descriptor approach. During our loop closure detection experiments using camera-LIDAR or stereo on KITTI sequences, we recorded both the time spent to compute each descriptor, and the time spent by each query correspondence search within the database. Our goal was to evaluate the time efficiency of each descriptor, investigating the costs associated with c-M2DP additional color information in comparison with M2DP. Also, we were able to compare their efficiency against the CSHOT descriptor, which can be expensive to compute as it depends on estimating surface normals, especially considering the high density and noise of the stereo-based point clouds.

In Table 4.2 we present the average times spent for each descriptor when using point clouds generated through camera-LIDAR sensor fusion. The average time to compute c-M2DP is only $23.2\%$ higher than M2DP. It is also worth noting that c-M2DP average computing time is $22.6\%$ faster than CSHOT.

Table 4.2: Average times in seconds to compute a descriptor and matching on point clouds generated using 3D LIDAR and color camera.

| **Descriptor** | **Computing** $(s)$ | **Matching** $(s)$ |
|---|---|---|
| M2DP | $\mathbf{0.0674} \pm 0.0041$ | $\mathbf{0.0043} \pm 0.0004$ |
| c-M2DP (Ours) | $\mathbf{0.0830} \pm 0.0052$ | $\mathbf{0.0051} \pm 0.0006$ |
| CSHOT | $0.1072 \pm 0.0168$ | $0.0059 \pm 0.0005$ |

In Table 4.3 we present the average times spent for each descriptor when using point clouds generated through stereo depth estimation. The increased point density of these point clouds caused an overall increase in the average times computing the descriptors. Similar to the previous experiment c-M2DP average time to compute is only $18.8\%$ higher than the M2DP descriptor. Also, it is important to highlight CSHOT's heavy computational burden. Estimating surface normals for a large number of points is a costly process, which often requires a downsampling of the point cloud, risking accuracy loss. This significantly affected CSHOT, with an average time of $\approx 1.7711s$ against only $\approx 0.4259s$ taken by c-M2DP, i.e. $315.9\%$ higher.

Table 4.3: Average times in seconds to compute a descriptor and matching on point clouds generated using stereo camera.

| **Descriptor** | **Computing** ($s$) | **Matching** ($s$) |
|---|---|---|
| M2DP | $\mathbf{0.3584} \pm 0.0816$ | $\mathbf{0.0044} \pm 0.0008$ |
| c-M2DP (Ours) | $\mathbf{0.4259} \pm 0.0956$ | $\mathbf{0.0054} \pm 0.0006$ |
| CSHOT | $1.7711 \pm 1.0159$ | $0.0061 \pm 0.0005$ |

## 4.4 Descriptors Precision-Recall Curves

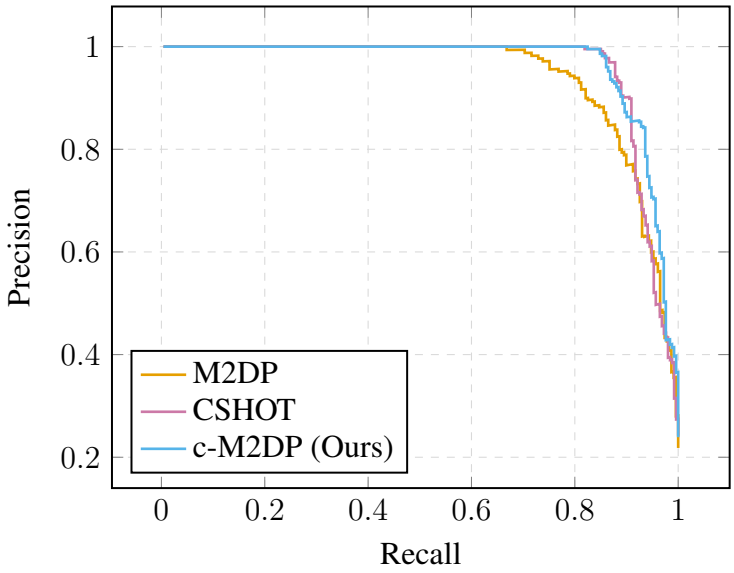
Our goal in the following experiments was the evaluation of c-M2DP descriptor accuracy when applied in a loop closure detection approach. With these results, we were able to investigate if the additional color information brought an improvement over the original M2DP descriptor. Also, they allowed the performance comparison between the c-M2DP and CSHOT descriptors. The KITTI odometry dataset sequences 00, 05, 06 and 07 were used for these experiments, allowing us to evaluate the descriptors behavior with different points clouds, generated either by the sensor fusion between 3D LIDAR and a color camera or estimated depth using a stereo camera.

### 4.4.1 Camera-LIDAR Sequences

At first, we ran the loop closure detection pipeline on point cloud sequences generated using camera-LIDAR sensor fusion. In Fig. 4.2 we plotted the precision-recall curves from KITTI 06 sequence, a simple big loop with 1101 frames. While c-M2DP presents a recall rate of $82.5\%$ at $100\%$ precision, reaching $89.2\%$ at $90\%$ precision, M2DP shows a

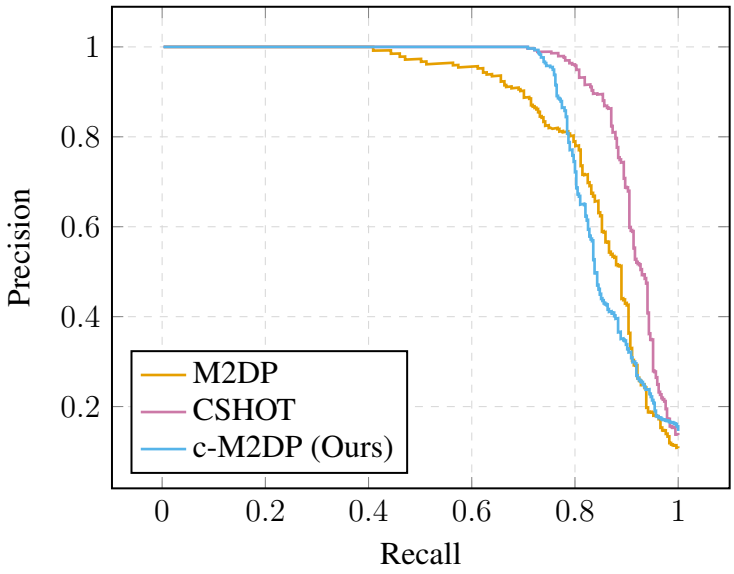
lower recall rate in comparison, of 66.8% at 100% precision and reaching 82.1% at 90% precision. Also with a lower recall, CSHOT shows 81.9% at 100% precision, reaching 90.6% at 90% precision.

Figure 4.2: Precision-recall curves on KITTI 06 camera-LIDAR sequence.




In Fig. 4.3 we plotted the precision-recall curves from KITTI 05 sequence, with 2761 frames and a few loops. At 100% precision c-M2DP presented a recall rate of 70.9%, reaching 76.5% at 90% precision. In comparison, M2DP performed poorly with a recall rate of 40.9% at 100% precision, and reaching 78.2% at 90% precision. Also, CSHOT shows a recall of 70.8% at 100% precision, reaching 83.5% at 90% precision.

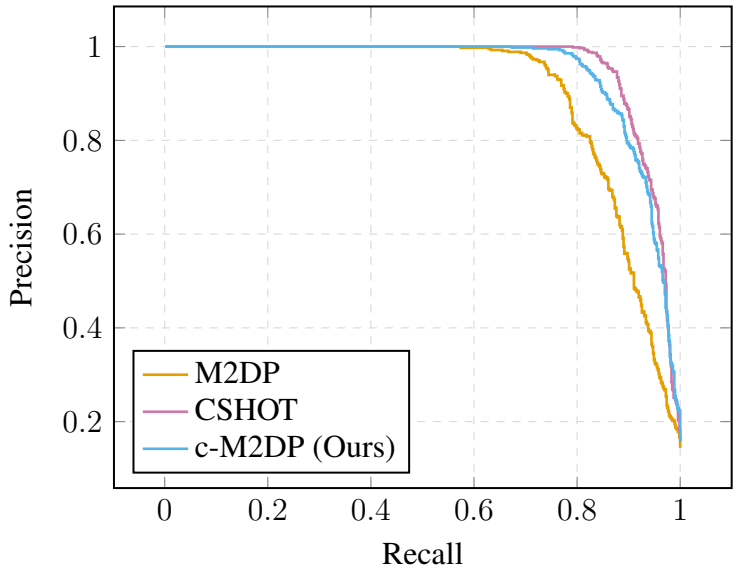
Figure 4.3: Precision-recall curves on KITTI 05 camera-LIDAR sequence.




In Fig. 4.4 we plotted the precision-recall curves from KITTI 00 sequence. It was the longest (4541 frames), and a challenging segment with dense vegetation. The

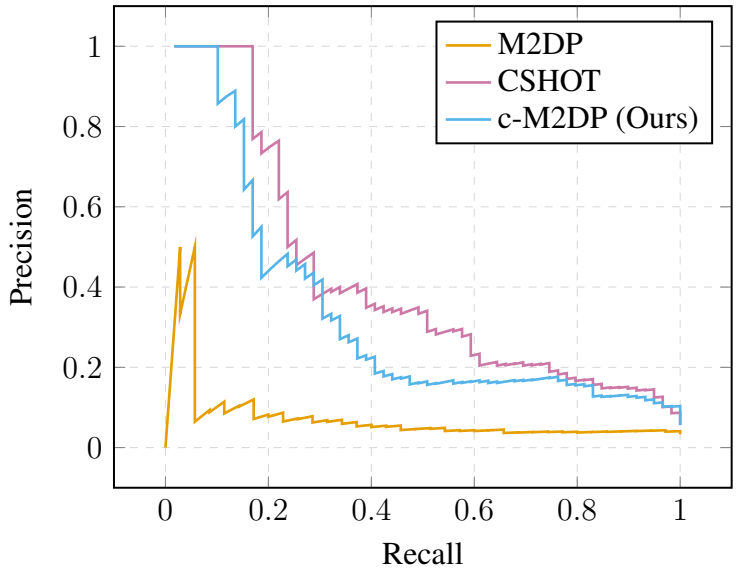
c-M2DP descriptor shows a recall rate of 67.3% at 100% precision, reaching a recall of 85.4% at 90% precision. As expected, M2DP shows a lower recall rate, with 57.4% at 100% precision, and reaching 78.2% at 90% precision. However, CSHOT presents a higher recall rate, reaching 79.2% at 100% precision, and up to 88.6% at 90% precision.

Figure 4.4: Precision-recall curves for KITTI 00 camera-LIDAR sequence.




In Fig. 4.5 we plotted the precision-recall curves from KITTI 07, a short (1101 frames) but extremely challenging sequence, with two different segments having very similar structures. Performing poorly, c-M2DP presents a recall rate of only 10.2% at 100% precision, after which it starts to drop dramatically. In comparison, M2DP was unable to provide any recall at 100% precision, while CSHOT shows the better recall rate of 17% at 100% precision, also dropping significantly after it.

Figure 4.5: Precision-recall curves on KITTI 07 camera-LIDAR sequence.



The recall rates of each descriptor at $100\%$ precision are summarized in Table 4.4. As expected, using a forward facing 3D LIDAR reduced the descriptors overall accuracy in comparison the results presented in the M2DP study, which uses a $360^{\circ}$ FoV for the same KITTI sequences. In general, c-M2DP results shows a significant improvement over the original M2DP. For instance, on sequence 05 c-M2DP presented a recall of $70.9\%$ with no false positives against only $40.9\%$ of M2DP. Also, on the challenging sequence 07, M2DP failed by always providing false positives. Competitive recall rates against the CSHOT descriptor are achieved by c-M2DP on sequence 06, with a higher recall of $82.5\%$ at $100\%$ precision. On sequence 05 both c-M2DP and CSHOT results are within a slight difference at $100\%$ precision, with their respective recall rates being at $70.9\%$ and $70.8\%$. However, on sequences 00 and 07 CSHOT presented better recall rates in comparison with the other descriptors.

Table 4.4: Recall at $100\%$ precision on KITTI camera-LIDAR sequences.

| **Sequence** | **M2DP** | **c-M2DP (Ours)** | **CSHOT** |
|---|---|---|---|
| KITTI06 | 0.668122 | **0.824701** | 0.818898 |
| KITTI05 | 0.408935 | **0.708861** | 0.708108 |
| KITTI00 | 0.574303 | 0.673295 | **0.791549** |
| KITTI07 | 0 | 0.101695 | **0.169492** |

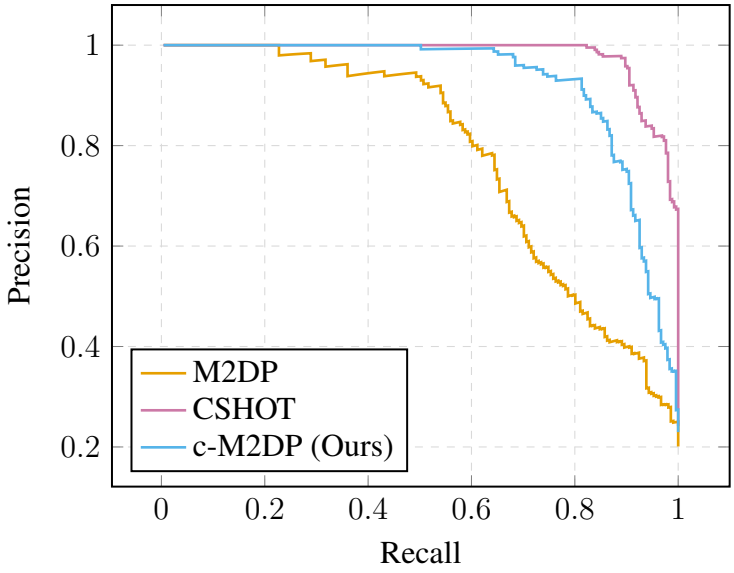
### 4.4.2 Stereo Camera Sequences

For the following experiments, we ran the pipeline using point clouds generated from stereo estimated depth. As mentioned in Chapter 3, as far as we known this was the first experiment with the M2DP descriptor using input from stereo cameras. In their study, He, Wang and Zang (2016) commented on evaluating M2DP with this and other types of sensors only as a possibility for a future work. We saw this as an opportunity to evaluate c-M2DP in comparison with its performance on 3D LIDAR. On the other hand, we expected CSHOT to show better accuracy than its previous 3D LIDAR results, as it was originally proposed for surface matching of dense point clouds generated by 3D scanners. The higher point density of the stereo-based point clouds allows surface normals to be more accurately estimated. However, due to the computational costs of this estimation process, CSHOT is significantly slower than the previous experiments, as seen in Section 4.3.

In Fig. 4.6 we plotted the precision-recall curves from the KITTI 06 sequence. In

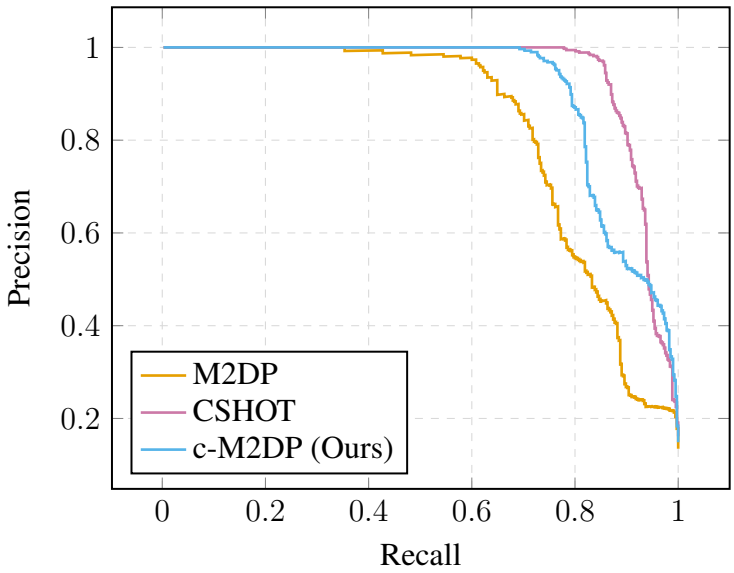
it, c-M2DP presents a recall rate of 50.2% at 100% precision, reaching a recall of 82.2% at 90% precision. As expected, M2DP shows a lower recall rate, with 22.8% at 100% precision, and reaches 54.5% with 90% precision. However, CSHOT presents the higher recall rate of 82.3% at 100% precision, reaching up to 91.7% when at 90% precision.

Figure 4.6: Precision-recall curves on KITTI 06 stereo camera sequence.




In Fig. 4.7 we plotted the precision-recall curves from the KITTI 05 sequence. At 100% precision c-M2DP shows a recall rate of 69.2%, reaching 79.2% before precision dropping below 90%. In comparison, M2DP performed poorly with a recall rate of 35.3% at 100% precision, reaching a recall of 64.9% at 90% precision. Again, CSHOT shows the higher recall of 77.9% at 100% precision, and reaches 87% at 90% precision.

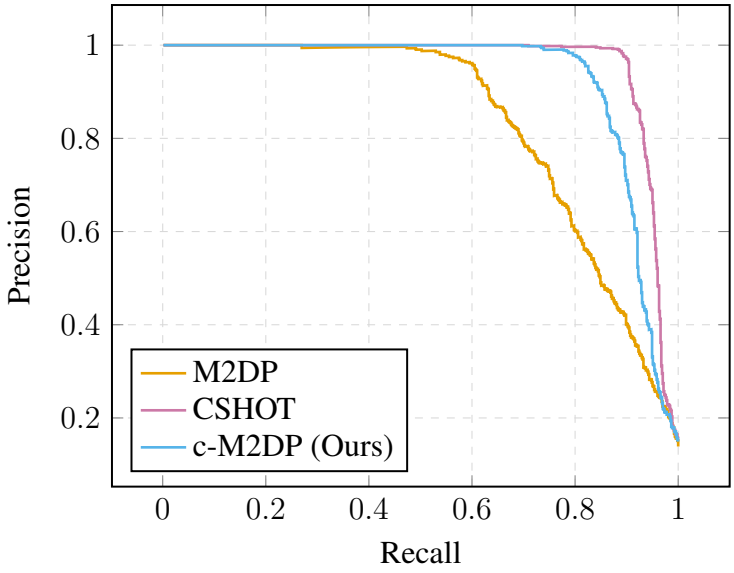
Figure 4.7: Precision-recall curves on KITTI 05 stereo camera sequence.




In Fig. 4.8 we plotted the precision-recall curves from the KITTI 00 sequence. It

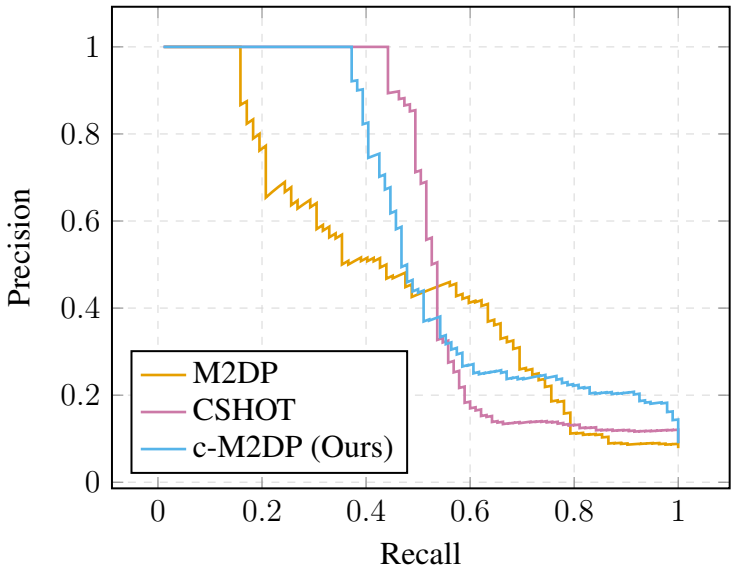
shows that c-M2DP presents a recall rate of 69.8% at 100% precision, reaching 85.3% at 90% precision. In comparison, M2DP shows a lower recall rate of 27% at 100% precision, keeping 50.2% at 99%, and reaching 63.2% at 90% precision. However, CSHOT shows a higher recall rate, of 70.9% at 100% precision and reaches 92.2% at 90% precision.

Figure 4.8: Precision-recall curves on KITTI 00 stereo camera sequence.




In Fig. 4.9 we plotted the precision-recall curves from the KITTI 07 sequence. While c-M2DP shows a recall of 37.2% at 100% precision, dropping significantly after it, M2DP presents a lower recall rate of 15.9% at 100% precision, and CSHOT shows a higher recall of 44.2% at 100% precision, both also dropping dramatically after it.

Figure 4.9: Precision-recall curves on KITTI 07 stereo camera sequence.




The recall rates of each descriptor at 100% precision are summarized in Table 4.5. For both M2DP and c-M2DP descriptors, the point clouds generated using estimated

depth from a stereo camera revealed to be more challenging than the ones generated from a 3D LIDAR. Still, c-M2DP additional color information proved to be a significant improvement over the original M2DP descriptor, as the shape signatures were more sensitive to these point clouds with increased density and noise. For instance, c-M2DP shows a recall of $69.8\%$ at $100\%$ precision on sequence 00 against only $27\%$ from M2DP. It is also worth mentioning that on sequence 07, the higher amount of information from these point clouds allowed better recall rates than the previous experiment on the same sequence using camera-LIDAR.

Although the CSHOT descriptor shows better results in the four sequences, with $82.3\%$ recall at $100\%$ precision in comparison to only $50.2\%$ of c-M2DP on sequence 06, our proposal achieved more competitive results against it on sequences 05, 07, and especially in sequence 00, where the difference between the recall rates at $100\%$ precision is slightly above $1\%$. These are significant results for our descriptor, considering that CSHOT accuracy comes at the cost of its time consumption increasing significantly, as previously discussed in Section 4.3, being several times slower to compute than c-M2DP due to the normal estimation on dense point clouds.

Table 4.5: Recall at $100\%$ precision on KITTI stereo camera sequences.

| **Sequence** | **M2DP** | **c-M2DP (Ours)** | **CSHOT** |
|---|---|---|---|
| KITTI06 | $0.227488$ | $0.502075$ | $\mathbf{0.822835}$ |
| KITTI05 | $0.353425$ | $0.692308$ | $\mathbf{0.778539}$ |
| KITTI00 | $0.269663$ | $0.697466$ | $\mathbf{0.709402}$ |
| KITTI07 | $0.158537$ | $0.372340$ | $\mathbf{0.442105}$ |

# 5 CONCLUSION

In this thesis we proposed the c-M2DP descriptor, an extension that incorporates color data into the global point cloud descriptor M2DP. Our proposal builds color signatures by computing and concatenating color histograms from multiple 2D projections of an input point cloud, while maintaining the efficient shape signatures from M2DP that are also computed from each 2D projection. These color and shape signatures are concatenated, and then used to compose a signature matrix that describes the point cloud. Finally, we employ M2DP's last step to reduce dimensionality of the signature matrix, resulting in a compact descriptor.

Additionally, we applied the c-M2DP descriptor to a loop closure detection pipeline and compared its performance with M2DP, in order to evaluate if the additional color information enabled more precise results and how it affected the descriptor time efficiency. Also, we compared c-M2DP performance with the CSHOT descriptor, that also combines color and shape data but requires estimating surface normals, which can be costly to compute. Our evaluation was done using point clouds generated from four KITTI dataset sequences. During the experiments, we measured loop closures detected correctly and incorrectly, along with the ones wrongly discarded, and used them to generate precision-recall curves for each descriptor. Besides, the times spent for the correspondence search and computing each descriptor were recorded to evaluate their efficiency.

Our first experiments were done with point clouds generated by performing sensor fusion between 3D LIDAR and color camera. After comparing the precision-recall curves generated using c-M2DP with three different color spaces, we selected the RGB color space for use due its better results. After that, we generated the precision-recall curves of each descriptor for the four KITTI sequences. An overall improvement on accuracy is presented by the c-M2DP descriptor in comparison to M2DP, while avoiding a significant increase in time consumption. Besides, c-M2DP results were competitive with the CSHOT descriptor, while also being faster to compute and having a significantly smaller vector length in comparison.

Our remaining experiments were done with point clouds generated from depth estimated using a stereo camera. It was, as far as we know, the first experiment with M2DP, and consequently c-M2DP, on the denser point clouds generated from this type of sensor. Although the CSHOT descriptor presented higher accuracy on these four sequences, it came at the cost of being several times slower than c-M2DP. The shape component of

CSHOT relies on estimating surface normals of the point cloud beforehand, requiring a properly tuned radius parameter to perform the estimation. Surface normals can be estimated more accurately when the point clouds are more dense, but computing them are a time consuming process due to the large amount of points. On the other hand, the c-M2DP descriptor achieved very competitive results, and is significantly faster to compute, as it uses the same low cost spatial density distributions of the original M2DP, along with the additional color histograms.

Despite the presented good results that support our proposal, there is still several aspects of the c-M2DP descriptor and its application to loop closure detection that can be further investigated. For instance, an intuitive next step is to perform an evaluation of c-M2DP while using colored point clouds generated with a FoV of $360°$, as available in the Ford Campus dataset (PANDEY; MCBRIDE; EUSTICE, 2011). Different outdoor environments, such as the Málaga urban dataset (BLANCO-CLARACO; MORENO-DUEÑAS; GONZÁLEZ-JIMÉNEZ, 2014), or indoor environments could also be employed and evaluated in a future work.

Finally, as the point clouds generated through stereo depth estimation revealed to be challenging for both M2DP and c-M2DP, further investigation could also be done on improving their shape signatures for more dense point clouds. Sampling techniques can be employed to reduce noise and high point densities (BOSSE; ZLOT, 2013; DUBÉ et al., 2017), although there is a potential loss of information. Additionally, future works could also investigate image pre-processing techniques used in visual methods (LOWRY et al., 2016), in order to provide more robust color information and potentially improve c-M2DP color signatures.

## REFERENCES

AIGER, D.; MITRA, N. J.; COHEN-OR, D. 4-points Congruent Sets for Robust Pairwise Surface Registration. In: **ACM SIGGRAPH 2008 Papers**. New York, NY, USA: ACM, 2008. (SIGGRAPH '08), p. 85:1–85:10. ISBN 978-1-4503-0112-1.

ALDOMA, A. et al. Our-cvfh – oriented, unique and repeatable clustered viewpoint feature histogram for object recognition and 6dof pose estimation. In: **Pattern Recognition**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 113–122.

ALDOMA, A. et al. Cad-model recognition and 6dof pose estimation using 3d cues. In: **2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)**. [S.l.: s.n.], 2011. p. 585–592.

BAILEY, T.; WHYTE, H. Durrant. Simultaneous localization and mapping (SLAM): part II. **IEEE Robotics Automation Magazine**, v. 13, n. 3, p. 108–117, Sep. 2006.

BESL, P. J.; MCKAY, N. D. A method for registration of 3-D shapes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 14, n. 2, p. 239–256, Feb 1992.

BLANCO-CLARACO, J.-L.; MORENO-DUEÑAS, F.-Á.; GONZÁLEZ-JIMÉNEZ, J. The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. **The International Journal of Robotics Research**, v. 33, n. 2, p. 207–214, 2014.

BOSSE, M.; ZLOT, R. Map Matching and Data Association for Large-Scale Two-dimensional Laser Scan-based SLAM. **The International Journal of Robotics Research**, v. 27, n. 6, p. 667–691, 2008.

BOSSE, M.; ZLOT, R. Place recognition using keypoint voting in large 3D lidar datasets. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2013. p. 2677–2684.

BROWN, E. **Cheetah III robot preps for a role as a first responder**. 2018. Available from Internet: <http://news.mit.edu/2018/cheetah-robot-preps-role-first-responder-sangbae-kim-0326>.

CADENA, C. et al. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. **IEEE Transactions on Robotics**, v. 32, n. 6, p. 1309–1332, Dec 2016.

CHEN, T. et al. Performance of global descriptors for velodyne-based urban object recognition. In: **2014 IEEE Intelligent Vehicles Symposium Proceedings**. [S.l.: s.n.], 2014. p. 667–673.

CIESLEWSKI, T. et al. Point cloud descriptors for place recognition using sparse visual information. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2016. p. 4830–4836.

CUMMINS, M.; NEWMAN, P. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. **The International Journal of Robotics Research**, v. 27, n. 6, p. 647–665, 2008.

DAVIES, A. **The WIRED Guide to Self-Driving Cars**. 2018. Available from Internet: <https://www.wired.com/story/guide-self-driving-cars/>.

DISSANAYAKE, M. W. M. G. et al. A solution to the simultaneous localization and map building (SLAM) problem. **IEEE Transactions on Robotics and Automation**, v. 17, n. 3, p. 229–241, Jun 2001.

DUBÉ, R. et al. SegMatch: Segment based place recognition in 3D point clouds. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2017. p. 5266–5272.

DUBÉ, R. et al. Incremental-Segment-Based Localization in 3-D Point Clouds. **IEEE Robotics and Automation Letters**, v. 3, n. 3, p. 1832–1839, July 2018.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part I. **IEEE Robotics Automation Magazine**, v. 13, n. 2, p. 99–110, June 2006.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. LSD-SLAM: Large-Scale Direct Monocular SLAM. In: FLEET, D. et al. (Ed.). **13th European Conference on Computer Vision (ECCV), Part II**. Zurich, Switzerland: Springer International Publishing, 2014. p. 834–849.

FENG, G.; LIU, Y.; LIAO, Y. LOIND: An illumination and scale invariant RGB-D descriptor. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2015.

FUENTES-PACHECO, J.; RUIZ-ASCENCIO, J.; RENDÓN-MANCHA, J. M. Visual simultaneous localization and mapping: a survey. **Artificial Intelligence Review**, v. 43, n. 1, p. 55–81, Jan 2015.

GAWEL, A. et al. Structure-based vision-laser matching. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2016. p. 182–188.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Piscataway, NJ, USA: IEEE, 2012.

HE, L.; WANG, X.; ZHANG, H. M2DP: A novel 3d point cloud descriptor and its application in loop closure detection. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2016. p. 231–237.

HO, K. L.; NEWMAN, P. Loop closure detection in SLAM by combining visual and spatial appearance". **Robotics and Autonomous Systems**, v. 54, n. 9, p. 740–749, 2006.

JOHNSON, A. E.; HEBERT, M. Using spin images for efficient object recognition in cluttered 3d scenes. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 21, n. 5, p. 433–449, May 1999.

LOGOGLU, K. Berker; KALKAN Sinan; TEMIZEL Alptekin. CoSPAIR: Colored Histograms of Spatial Concentric Surflet-Pairs for 3D object recognition". **Robotics and Autonomous Systems**, v. 75, p. 558–570, 2016.

LÓPEZ, D. Gálvez; TARDÓS, J. D. Real-time loop detection with bags of binary words. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2011. p. 51–58.

LOWRY, S. et al. Visual Place Recognition: A Survey. **IEEE Transactions on Robotics**, v. 32, n. 1, p. 1–19, Feb 2016.

MARR, B. **5 Major Robotics Trends To Watch For in 2019**. 2019. Available from Internet: <https://www.forbes.com/sites/bernardmarr/2019/03/08/5-major-robotics-trends-to-watch-for-in-2019/#278d489f5650>.

MILFORD, M. J.; WYETH, G. F. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2012. p. 1643–1649.

MORAL, E. Fernández et al. Fast place recognition with plane-based maps. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2013. p. 2719–2724.

MORENO, R. F. Salas et al. SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. Piscataway, NJ, USA: IEEE, 2013. p. 1352–1359.

MURILLO, A. C.; KOSECKA, J. Experiments in place recognition using gist panoramas. In: **IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)**. Piscataway, NJ, USA: IEEE, 2009. p. 2196–2203.

NASEER, T. et al. Robust visual SLAM across seasons. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2015. p. 2529–2535.

NEIRA, J.; TARDOS, J. D. Data association in stochastic mapping using the joint compatibility test. **IEEE Transactions on Robotics and Automation**, v. 17, n. 6, p. 890–897, Dec 2001.

NEIRA, J.; TARDOS, J. D.; CASTELLANOS, J. A. Linear time vehicle relocation in SLAM. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2003. v. 1, p. 427–433.

NEWMAN, P.; COLE, D.; HO, K. Outdoor SLAM using visual appearance and laser ranging. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2006. p. 1180–1187.

NÜCHTER, A. et al. 6D SLAM-3D mapping outdoor environments. **Journal of Field Robotics**, v. 24, n. 8, p. 699–722, 2007.

PANDEY, G.; MCBRIDE, J. R.; EUSTICE, R. M. Ford Campus vision and lidar data set. **The International Journal of Robotics Research**, v. 30, n. 13, p. 1543–1552, 2011.

RÖHLING, T.; MACK, J.; SCHULZ, D. A fast histogram-based similarity measure for detecting loop closures in 3-D LIDAR data. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2015. p. 736–741.

RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast Point Feature Histograms (FPFH) for 3D registration. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2009. p. 3212–3217.

RUSU, R. B. et al. Fast 3D recognition and pose using the Viewpoint Feature Histogram. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2010. p. 2155–2162.

SCHUMACHER, H. **Is this the end of household chores?** 2018. Available from Internet: <http://www.bbc.com/future/story/20180730-could-robots-do-our-household-chores-like-laundry>.

SEGAL, A.; HAEHNEL, D.; THRUN, S. Generalized-ICP. In: **Proceedings of Robotics: Science and Systems**. Cambridge, MA, USA: MIT Press, 2009.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company, 2004. ISBN 026219502X.

SIMON, M. **Inside the Amazon Warehouse Where Humans and Machines Become One**. 2019. Available from Internet: <https://www.wired.com/story/amazon-warehouse-robots>.

STACHNISS, C. **Exploration and mapping with mobile robots**. Thesis (PhD), 2006.

STEDER, B. et al. Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2011. p. 1249–1255.

SÜNDERHAUF, N.; PROTZEL, P. BRIEF-Gist - closing the loop by simple means. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Piscataway, NJ, USA: IEEE, 2011. p. 1234–1241.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)**. Cambridge, MA, USA: MIT Press, 2005. ISBN 0262201623.

THRUN, S.; LEONARD, J. J. **Springer Handbook of Robotics**. Heidelberg, Germany: Springer-Verlag, 2008.

TOMBARI, F.; SALTI, S.; STEFANO, L. D. Unique Signatures of Histograms for Local Surface Description. In: **11th European Conference on Computer Vision (ECCV), Part III**. Heidelberg, Germany: Springer-Verlag, 2010. (ECCV'10), p. 356–369.

TOMBARI, F.; SALTI, S.; STEFANO, L. D. A combined texture-shape descriptor for enhanced 3D feature matching. In: **18th IEEE International Conference on Image Processing (ICIP)**. Piscataway, NJ, USA: IEEE, 2011. p. 809–812.

ULRICH, I.; NOURBAKHSH, I. Appearance-based place recognition for topological localization. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Piscataway, NJ, USA: IEEE, 2000. v. 2, p. 1023–1029 vol.2.

VINCENT, J. **Welcome to the automated warehouse of the future**. 2018. Available from Internet: <https://www.theverge.com/2018/5/8/17331250/automated-warehouses-jobs-ocado-andover-amazon>.

WHITE, G. B. **How Many Robots Does It Take to Replace a Human Job?** 2017. Available from Internet: <https://www.theatlantic.com/business/archive/2017/03/work-automation/521364/>.

WOHLKINGER, W.; VINCZE, M. Ensemble of shape functions for 3D object classification. In: **IEEE International Conference on Robotics and Biomimetics**. Piscataway, NJ, USA: IEEE, 2011. p. 2987–2992.