Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INFORMATICA

Ciclo XXVIII

Settore Concorsuale di afferenza: 09/H1

Settore Scientifico disciplinare: ING-INF/05

CREATING AND RECOGNIZING 3D OBJECTS

Presentata da: **Dott. Alioscia Petrelli**

Coordinatore Dottorato:	Relatore:
Chiar.mo Prof. Ing. **Paolo Ciaccia**	***Chiar.mo Prof. Ing.*** **Luigi Di Stefano**

Esame finale anno 2016

“*It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.*”

Sir Arthur Conan Doyle

Abstract

This thesis aims at investigating on 3D Computer Vision, a research topic which is gathering even increasing attention thanks to the more and more widespread availability of affordable 3D visual sensor, such as, in particular consumer grade RGB-D cameras. The contribution of this dissertation is twofold. First, the work addresses how to compactly represent the content of images acquired with RGB-D cameras. Second, the thesis focuses on 3D Reconstruction, key issue to efficiently populate the databases of 3D models deployed in object/category recognition scenarios.

As 3D Registration plays a fundamental role in 3D Reconstruction, the former part of the thesis proposes a pipeline for coarse registration of point clouds that is entirely based on the computation of 3D Local Reference Frames (LRF). Keys to the method are: (a) the observation that any two corresponding points endowed with a LRF provide a hypothesis on the rigid motion between two views, (b) the intuition that feature points can be matched based solely on cues directly derived from the computation of the LRF, (c) a feature detection approach relying on a saliency criterion which captures the ability the establish a LRF repeatably. Unlike related work in literature, we also propose a comprehensive experimental evaluation based on diverse kinds of data (such as those acquired by laser scanners, RGB-D and stereo cameras) as well as on quantitative comparison with respect to three other methods. We also address the issue of fairly and realistically set the many parameters that characterize coarse registration pipelines. The experimental evaluation vouches that our method can handle effectively data acquired by different sensors and is remarkably fast.

Driven by the ever-lower costs and growing distribution of 3D sensing devices, we expect broad-scale integration of depth sensing into mobile devices to be forthcoming. Accordingly, the thesis investigates on merging appearance and shape information for Mobile Visual Search and focuses on encoding RGB-D images in compact binary codes. An extensive experimental analysis on three state-of-the-art datasets, addressing both category and instance recognition scenarios, has led to the development of an RGB-D search engine architecture that can attain high recognition rates with peculiarly moderate bandwidth requirements. More in depth, the analysis seems to suggest that maintaining the processing of the color and depth information independent may turn out the most effective strategy. Moreover, Deep Features provided by Convolutional Neural Networks better encode appearance information, whereas shape is more effectively represented through Kernel Descriptors globally encoded by Fisher Kernel. Our experiments also include a comparison with the CDVS (Compact Descriptors for Visual Search) pipeline, candidate to become part of the MPEG-7 standard. Finally, our evaluation show that appearance and depth are synergistically beneficial in category recognition tasks, whereas shape features provide limited contribution in discriminating among different instances of objects. Further analysis suggests that learning to weigh the relative contribution of depth and appearance in a specific task is key to devise an engine able to work seamlessly both in category and instance recognition scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Before the advent of Microsoft Kinect, in 2010, most 3D acquisition systems had been purposely designed to provide accurate and noiseless acquisitions, often to the detriment of cheapness and ease of use. Since then, Kinect has established itself not only as an entertainment tool but also as a means to obtain real-time acquisitions at low cost. Consequently, in the last years, the computer vision community has given more and more attention to 3D sensing as a way to improve the capability of algorithms to perceive and understand the surrounding world. The effort in this direction has already led to the development of novel, sometimes groundbreaking, approaches to address hard vision problems like human pose recognition [85], SLAM [30], object recognition [12], object detection and semantic segmentation [39]. Moreover, inspired by the philosophy of Kinect, other manufacturers have proposed their depth sensors as Xtion by Asus, Senz3D by Creative and Microsoft itseft has recently launched on the market the Kinect 2.0 based on time-of-flight technology (see Fig. 1.1).

FIGURE 1.1: Examples of low-cost depth sensing devices. From left to right: Kinect, Xtion, Senz3D and Kinect V2.

All the cited devices project the observed 3D environment onto the 2D plane of the sensor so as to produce a map of depths that are usually referred to as 2.5D acquisitions. In other words, 3D data returned by the device is limited to the portion of the scene that is visible from the point of view of the sensor at the time of the acquisition, as shown in Fig. 1.2.

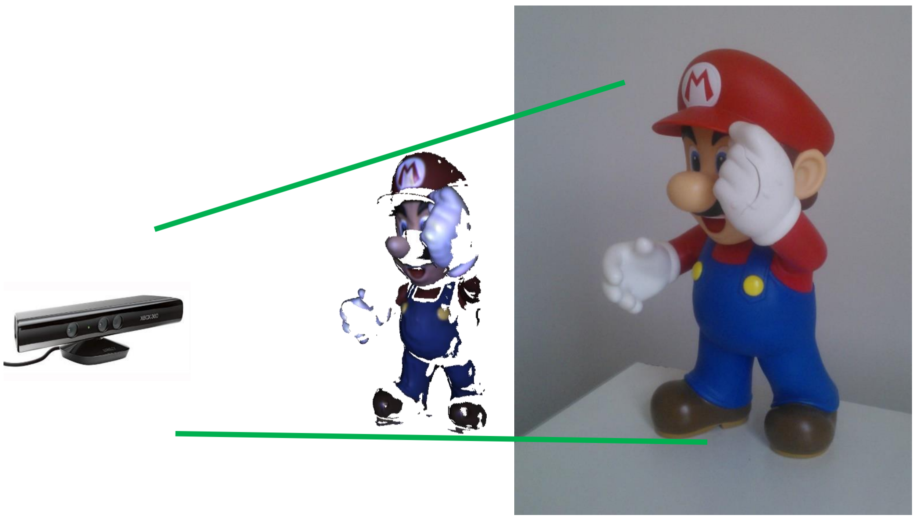


FIGURE 1.2: Example of partial view acquired with a depth sensor.

Such restrain strongly affects two fundamental tasks of computer vision: *3D Reconstruction* and *3D Object Recognition*.

The former concerns the creation of a tridimensional representation of a physical object, a fundamental task for a broad range of application areas such as e.g. digital preservation of cultural heritage, robotics and video games. Reconstructed 3D models are also widely used in 3D Object Recognition, which addresses the problem of picking out an object in an acquired scene from a set of objects stored in a database. 3D models allow for establishing the pose of the objects in the scene, which is essential in applications like robot grasping and augmented reality.

This thesis addresses both of these tasks. The process of creating a 3D model is not solved in a single comprehensive step, but involves a set of successive stages that will be described in detail in Chap. 2 and that start with the acquisition of the object from different vantage points so as to cover its entire surface. As each acquisition is represented with respect to a reference frame centered in the sensor, multiple acquisitions have to be aligned in a unique and common reference frame. Such process takes the name of *3D Registration* and is typically solved by aligning, in the first place, the partial views in pairs. This thesis focuses on this step, here referred to as *Pairwise registration*, which involves the matching of the overlapping surface between the two partial views. Usually, this problem is faced by extracting a set of characteristic points from the two views, that are eventually matched on the basis of a description of the neighbourhood (*support*) of the feature points. To achieve invariance of the descriptions to the pose of the object, the support is encoded w.r.t. a Local Reference Frame (LRF) defined on the feature point and anchored to the surface. A trait common to the descriptors proposed in literature is the representation in the form of sequences of floating-point numbers that are usually lengthy. For example, *Spin Image* descriptors [51] are of the order of 100 numbers, whereas *3D Shape Contexts* [35] produces feature vectors of about 2000 elements. Such descriptions, together with the large amount of feature points typically extracted in the detection stage, slow down the computation and burden on the subsequent matching stage, which, as a consequence, resorts to high-dimensionality indexing schemes. Chapter 3 addresses this issue and describes, as solution, a pairwise registration pipeline that avoids the computation of onerous descriptors and grounds the matching only on local reference frames. Chapter 4 reports the extensive experimental evaluation we performed against three state-of-the-art methods on a plethora of datasets acquired with different sensors, and shows that the proposed algorithm turns out faster than the other algorithms based on canonical descriptors. Surprisingly, it also proves to be more effective in aligning partial views that share a limited surface area.

Trends as relevant as the already mentioned wide diffusion of depth sensors, their increasing miniaturization, the advances in 3D computer

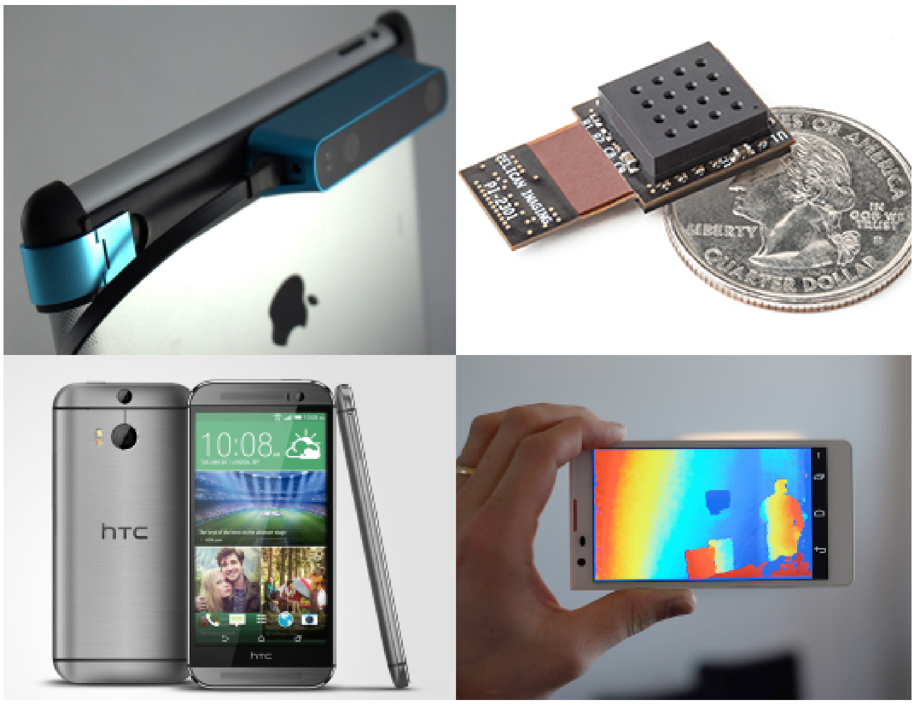


FIGURE 1.3: Examples of 3D sensing solutions for mobile devices. Top-left: the *Structure* sensor mounted on an iPad; top-right: the *PiCam* by *Pelican Imaging*; bottom-left: the HTC One (M8) smartphone; bottom-right: a prototype Tango device by Google.

vision and the growing development of software tools dedicated to mobile platforms seem to build momentum for integration of depth sensing into mobile devices at a large scale. Some existing solutions to endow mobile devices with depth sensing are shown in Fig. 1.3. The *Structure Sensor*[1] by *Occipital* is a structured light depth camera that can be clipped onto a tablet. Though originally designed for iOS devices, it can work with Android and Windows as well. In [100], *Pelican imaging*[2] proposed a camera array that captures light fields and synthesizes a range image. The camera is small enough to be embedded into next generation smartphones. In 2014 *HTC* released the *HTC One (M8)* smartphone, which combines the main RGB camera with a 2-megapixel

[1] http://structure.io

[2] http://www.pelicanimaging.com

depth sensor, and delivered the *Dual Lens SDK* to stimulate development of 3D applications on Android. *Project Tango*[3] by Google has just started shipping to professionals and researchers a prototype tablet equipped with 3D sensing capabilities and up-to-date APIs.

Motivated by the foreseeable broad availability of depth sensing on mobile devices and the achievements enabled by RGB-D imagery across a variety of computer vision tasks, in collaboration with STMicroelectronics that also funded this thesis, we investigated on joint deployment of color and depth in the realm of Mobile Visual Search.

Accordingly, the latter part of the thesis regards RGB-D Object recognition and addresses how to better merge appearance and shape information in order to improve the recognition capabilities of the system in the task of recognizing the specific instance of an object as well as in determining the category to which it does belong. Lai et al. in [58] provide the first experimental analysis on the benefits of exploiting depth information in object recognition. The other relevant works in this field ground on approaches based on machine learning and propose learned features for both RGB and depth channels: Bo et al. in [11, 10] proposed *Kernel Descriptors*, learned features that encode different patch attributes of the image like intensity and depth gradients, color and object size. Then, Bo et al. in [12] and Yu et al. in [103] introduced, respectively, the *Hierarchical matching pursuit* and the *Hierarchical Sparse Shape Descriptor*, unsupervised learning schemes that exploit sparse coding to learn hierarchical feature representations of RGB-D data. The recent successes of *Convolutional neural networks* (CNN) and, in wider terms, *Deep Learning*, prompted the research on RGB-D recognition in such direction and significant works have been published. [9] proposed the *Convolutional k-means descriptor*, whereas Socher et al. [88] combined convolutional (for low-level representation) and recursive neural networks (for higher order features) for classifying RGB-D images. Lately, Gupta et al. in [39] propose a framework for object

[3] www.google.com/atap/projecttango

segmentation and recognition that deploys the "AlexNet" proposed in [56] for encoding both the appearance and shape information.

Essential feature of an RGB-D Object Retrieval architecture is the ability to recognize the object regardless the acquisition viewpoint. Such invariance to pose can be obtained by modeling the objects comprising the database as a set of partial views acquired by different vantage points, which are, subsequently, matched against the query image in order to find the most similar one. As the recognition capability of the system tends to improve as the amount of acquisitions increases, typically the number of partial views acquired for each object reaches the hundreds. It is therefore clear that building the database turns out infeasible as the cardinality of objects becomes conspicuous. A solution that is usually adopted consists in reconstructing the complete 3D model of the object. Then, a large amount of partial views are generated synthetically by means of a virtual camera simulating the sensor employed to acquire the query images. That highlights the strong link between 3D Object Recognition and 3D Reconstruction and points out the importance of a 3D Registration stage able to operate in conditions of limited surface overlap between the partial views, which would enable to rely on fewer acquisitions, thereby to obtain a correct reconstruction cutting down time and effort spent for populating the database.

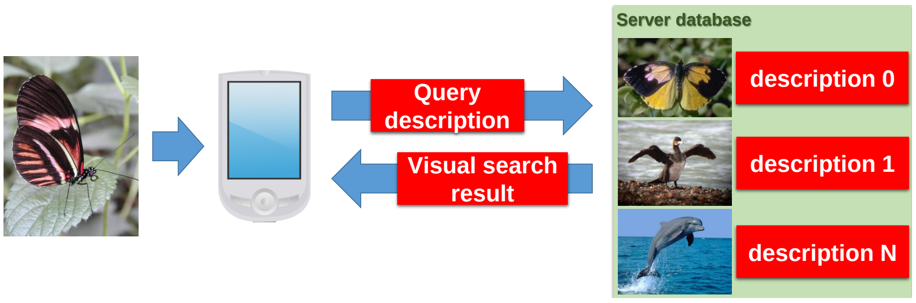


FIGURE 1.4: Overview of a Mobile Visual Search application.

A Mobile Visual Search engine allows the user to easily gather information about the objects seen in the camera field of view. Purposely, she/he would just snap a picture to have the mobile device computing a representation of the image which is sent to a remote server and matched into a database to recognize image content and report back relevant information (see Fig. 1.4). Technological advances and diffusion of mobile devices equipped with high-resolution cameras have made fertile the research on Mobile Visual Search [36, 23, 48, 21] and fostered the advent of both applications and development tools, such as Amazon Flow, CamFind, Google Goggles, SHOT & SHOP, Shot & Find, SnapPlay, Whatzit, Calliope, Visual Search for Shoes, Visual Search for Groceries.

As a Mobile Visual Search client transmits information to the server wirelessly, the search engine is subjected to strict bandwidth requirements so to reduce latency, cut down device power consumption and enhance user experience. Purposely, several approaches to either conceive compact image descriptors or compress existing ones have been proposed in literature [18, 20, 22, 52, 98, 37]. Besides bandwidth requirements, research on compact and binary description is also driven by the demand for handling databases that nowadays may comprise millions of images. Indeed, the ability to encode image content with as few bits as possible has become key to properly deal with storage issues and allow for efficient image matching.

Hence, our investigation addresses both how to merge the contributions provided by the color and depth channels as well as how to encode them in a compact binary code. Accordingly, this thesis propose, discuss and evaluate what we believe to be the first Visual Search engine specifically conceived for next-generation mobile devices equipped with RGB-D cameras.

Whereas Chapter 5 describes the Mobile Visual Search architecture we devised, Chapter 6 reports the results of an experimental investigation we performed aimed at identifying its best configuration. Our findings reveal that shape features are more effective in the case of category recognition. On the contrary, appearance is the primary trait that identifies a specific object. To take an example, the spherical shape is the very feature that characterizes a ball, whereas a particular type of ball (soccer ball, basket ball, etc...) is recognized on the basis of its textures and colors. Our findings also suggest that densely computed kernel descriptors aggregated at the image level through Fisher Kernel constitutes the best approach for the description of shape information. Instead, deep features are more effective for encoding appearance. Furthermore, both the representations can be compressed in binary codes by Spherical Hashing without any loss in descriptiveness. Indeed, high recognition rates can be achieved with binary codes as compact as 512-1024 bits in both category and instance retrieval experiments. Moreover, keeping the processing flows of the color and depth channels separate to concatenate the final binary codes seems not to hinder performance while potentially allowing for a great deal of flexibility at the system and application level. In the same chapter, we experimentally compare our proposal on three state-of-the-art RGB-D datasets with the *Compact Descriptors for Visual Search* architecture [1], an appearance-based search engine which, similarly to our proposal, is specifically aimed at generating compact encodings and it is currently in the Draft International Standard stage as Part 13 of the MPEG-7 standard.

The thesis concludes with Chap. 7 which reports the result of the work carried out during a three-month internship at the Imaging division of STMicroelectronics in Grenoble under the supervision of Dr. Alain Issard. This period led to the porting of Fisher Kernel and Spherical Hashing on the STxP70 ASMP embedded multicore platform. The algorithms have been plugged in the Visual Search architecture so as to assess the reduction in recognition rate caused by the adopted fixed-point representation. Furthermore, the scalability of the implementations have been evaluated on varying of the number of cores made available by the simulator of the embedded platform. Finally, to validate how the architecture works on a mobile device, the engine has been ported on a *Samsung Galaxy Tab Pro 10.1* equipped with a *Structure Sensor* for the acquisition of the depth image. The chapter ends by describing the details about the implementation and the considerations regarding its employment.

Chapter 2

3D Reconstruction

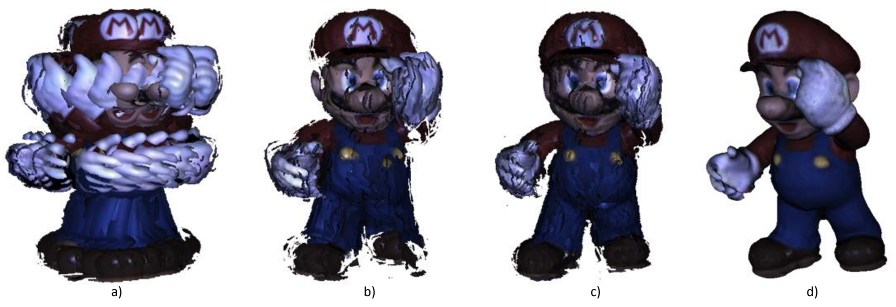


FIGURE 2.1: Result of the main stages of a 3D reconstruction pipeline. a) Set of acquired partial views, each represented with respect to the reference frame defined by the vantage point of the sensor. b) Coarse alignment of the input views. c) Refined global registration. d) Outcome of the 3D reconstruction process as a watertight manifold mesh.

3D Reconstruction is a fundamental task in 3D computer vision and plays an increasingly important role in reverse engineering and rapid prototyping, virtual reality, movie and game industries. Given a physical object, it concerns the automatic creation of an accurate and realistic 3D model. The majority of 3D sensors used to acquire shape information, such as laser scanners, stereo and time-of-flight cameras, structured light systems, share the inability to scan the entire object at once. Indeed, a single acquisition captures only the portion of the object seen from the viewpoint of the sensor as depicted in Fig. 1.2. Such acquisition, in this thesis referred to as *mesh* or *partial view*, represents the surface of the object as a lattice of points and triangles, and, typically, includes additional information such as surface normals computed on points (see Fig 2.2).

Hence, to obtain its complete model, the object has to be acquired from different vantage points up to covering its entire surface. Yet, upon completion of the acquisition session, the partial views are not aligned coherently but each is represented in its own reference frame as determined by the vantage point of the sensor (see Fig. 2.1a). Therefore, views have to be aligned with respect to a unique reference to obtain the final reconstruction of the object. This process, known as *3D Registration*, is most desirably accomplished without assumptions on the relative position and orientation of the views at hand and it is typically addressed by determining the rigid motions that align each pair of views. Such task, referred to as *Pairwise registration*, is ordinarily faced by a two-step procedure. The aim of the former (*Coarse pairwise registration*) is solely to provide a sufficiently correct alignment to the latter, which then attends to refine the registration until it converges (*Fine pairwise registration*). Eventually, the estimation of the rigid motions that align all the view pairs enables the arrangement of the views in a common global reference frame (*Coarse global registration*) (see Fig. 2.1b). Sensor noise, the discretization of the views in lattices and inaccuracies in the estimation of the rigid motions render the ongoing alignment not sufficiently precise to be correctly reconstructed. For this reason, a successive stage of *Fine global registration* refines the alignment so as to minimize the global misalignment error between corresponding points of the views. Once the partial views are correctly aligned (as shown in Fig. 2.1c), a final stage of *Mesh reconstruction* produces a manifold mesh (see Fig. 2.1d).

The purpose of this chapter is far from providing an exhaustive overview of the literature on this topic (that would be, by this time, too extensive to be addressed in a single manuscript), but it aims at explaining the steps that are commonly adopted to tackle this task. Such stages, summarized in Fig. 2.3, will be described in more details in the subsequent

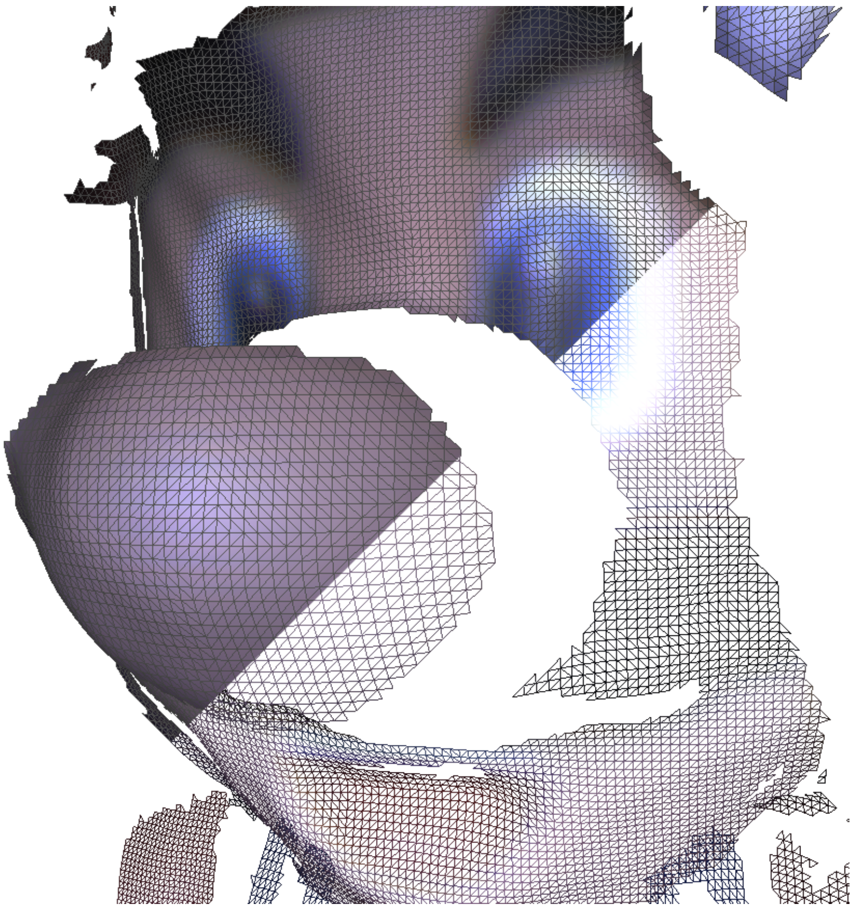


FIGURE 2.2: Example of 3D mesh composed by points and triangles.

sessions.

2.1 Coarse pairwise registration

The coarse registration methods published during the last two decades can be categorized into global and local approaches. Prominent methods within the former category rely on *Principal Component Analysis* [25], on *Extended Gaussian Images* [67] as introduced by Horn in [44], as well as on *Algebraic Surface Models* [93]. All these methods try to find the mapping between the two surfaces by computing descriptions that globally represent the views. Such criteria find it difficult to deal

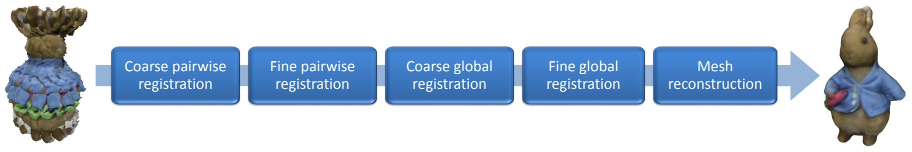

Coarse pairwise registration
Fine pairwise registration
Coarse global registration
Fine global registration
Mesh reconstruction

FIGURE 2.3: Overview of a typical 3D reconstruction processing flow.

with nuisances such as point density variation, noise and missing regions due to self-occlusions, thereby failing when the overlap between the two surfaces is limited. To overcome these issues, local methods have become established. Local methods have proved to be significantly resilient to nuisances such as those mentioned above and hence allow registering view pairs that share just a limited portion of surface. Typically, they follow a similar processing flow comprising the steps described below.

2.1.1 Feature extraction

At first, a subset of points (*feature points*) are extracted from the two partial views. Various criteria have been proposed, from simple uniform or random picking (with the only purpose to lighten the subsequent processing) to more sophisticated approaches (attempting to select the features that better characterize the surface). Among them, different works, inspired by [65], extend the *Difference of Gaussian* detector to 3D meshes. Examples are [61, 15, 19] and the MeshDoG detector proposed by [104]. In [3], instead, mesh points compete in a game theory framework until a subset of them stands out as the most dissimilar to the others. Mian et al. in [70] perform a mesh simplification and keep survived points as features. Furthermore, [106] proposes the *Intrinsic Shape Signatures* detector that discards points not possessing large variations in their neighborhood (here referred to as *supports*) along the three dimensions. This is carried out by computing the shatter matrix of the support points coordinates and checking if two of the eigenvalues of the matrix are too similar. A recent survey and evaluation of state-of-the-art 3D detectors can be found in [97].

2.1.2 Local description

Subsequently, each feature point is described so as to obtain a numerical representation of the neighbour surface of the point. On the one hand, such encoding should be as descriptive as possible, meaning that similar supports should provide similar representations. On the other hand, the description has to be robust to the nuisances already mentioned and invariant to rigid motions of the view. Such requirements have fostered the research community to propose a multitude of 3D descriptors, some of which have been evaluated in a recent survey proposed in [38].

The most widely-used method, proposed in 1997 by Johnson et al., is *Spin Image* [50] that projects the support point coordinates on a 2D image representation. In the same year, Chua and Jarvis proposed *Point Signature* [24] that encodes the distances of the surface points intersecting a support sphere from the tangent plane defined on the feature point. Tombari et al. introduce *Shot* [95], which provides a good trade-off between descriptiveness and robustness by encoding the feature point through orientation histograms of support normals. Other methods have been designed to focus on descriptive power, such as e.g. [91, 6, 70, 104], [19] which models each descriptor by a Hidden Markov Model, and [61, 69] that encode the magnitude of the Fourier Transform of the descriptor and discards phase component representing pose information. On the other side, works as [3, 81, 57] proposed loosely distinctive encodings, by relying, subsequently, on matching stages able to robustly handle a large amount of false positives and on geometric consistency methods.

2.1.3 Description matching

Descriptions are, therefore, used to establish correspondences between feature points of the two partial views. Such stage is typically performed by matching descriptions with respect to the euclidean distance in the description space. To speed up the nearest neighbour search, works like [95, 69, 61] deploy a Kd-tree index. Other works, grounding on descriptors not defined in euclidean space, resort to more sophisticated - and slower - matching schemes. For example, [3] exploits the same game-theory framework used for feature detection to discover correspondences through a competitive process among all possible correspondence pairs. [50], instead, computes the normalized linear correlation coefficient between *Spin Images* to match features, whereas, [57] sets multiple candidate matches for each feature, that eventually prune as an optimization problem through weighted bipartite matching.

2.1.4 Rigid motion estimation

Once a set of correspondences has been established, the rigid motion that aligns the two partial views is estimated. As a percentage of these correspondences are false matches, a robust estimator, such as e.g. RANSAC [33], is applied. Starting from the observation that three correct correspondences are sufficient to define a rototranslation, RANSAC iteratively picks out three correspondences at random and computes the rigid motion that best fits the correspondences in the least-squares sense. Such problem is called *Absolute orientation* and the closed-form solution can be obtained, for three or more points, by the method proposed in [43]. Then, the candidate rigid motion is applied to all the correspondences so as to count the number of them that aligns up to a user-defined threshold (*consensus set*). The process is repeated until a fixed number of iterations is reached or the largest consensus set so far obtained is sufficiently wide. As the loop stops, all the correspondences in the largest consensus set are used to estimate the final rigid motion.

Whereas several works deploy the standard RANSAC framework (e.g. [6, 57, 15, 69, 61]), others extend it or are inspired by its sample consensus based scheme. For example, [81] introduces the *Sample consensus initial alignment* (SAC-IA) that constrains the points of the three correspondences selected in each iteration to have pairwise distances greater than a threshold and check the goodness of the consensus set through the Huber penalty measure. The *4-Points Congruent Set* algorithm (4PCS) proposed in [2], instead, iteratively selects four planar points in a partial view and search for a fitting set of four points in the other.

2.2 Fine pairwise registration

Pairwise registration methods are able to align two partial views regardless their initial pose but, generally, provide rigid motion estimations that are not sufficiently accurate as input for the next stages of the reconstruction process. Grounding the estimation on a subset of the partial views points, sensor noise, and the discretization of the views, all together contribute to the coarseness of the resulting registration. Consequently, such initial coarse step is coupled with a subsequent refinement, that, complementarily, exploits all the points of the views but needs an initial guess to converge. Such task is solved effectively by the ubiquitous *Iterative closest points* algorithm (ICP) [102, 8].

In its early formulation, given two input point clouds (depicted, for the sake of clarity, as bidimensional curves in Fig. 2.4, left), each point of one cloud is matched with its closest point in the other one so as to estimate a rigid motion in a least square sense. As a single iteration is not usually sufficient to provide the minimum alignment error, the process is repeated up to convergence or a fixed number of iterations is reached (see Fig. 2.4, right).

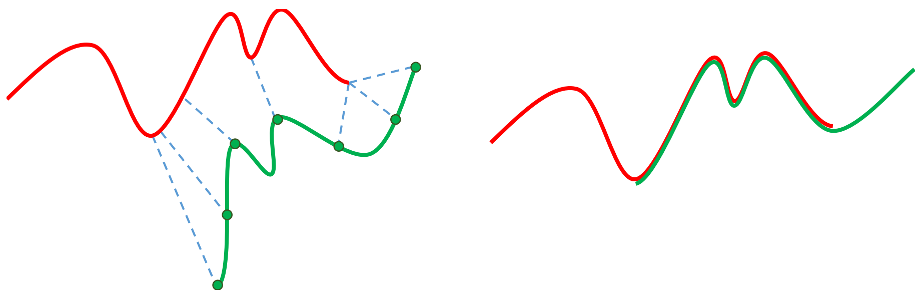


FIGURE 2.4: Outline of the ICP method. The red and green line represent, as 2D curves, the points clouds before and after the alignment.

Since its introduction in 1992, the research community has proposed several works addressing improvements and extensions to the original method. Most of them are summarized in [80, 63]. Among the most relevant ones, it is worth citing the deployment of a nearest neighbor search index to speed up the search for the closet point (the expensive step of the algorithm). Furthermore, the *point-to-point* distance is replaced by minimizing, instead, the sum of the squared distances between the points and the tangent plane at their corresponding points. At the cost of slower iterations, the algorithm generally converges in fewer steps. Other significant contributions regard the *Generalized ICP* [84], a probabilistic framework that exploits the maximum-likelihood estimation algorithm to minimize a *plane-to-plane* symmetrical distance, and the LM-ICP proposed in [34], that replaces the closed-form inner loop with the Levenberg-Marquardt algorithm.

2.3 Coarse global registration

Once rigid motions have been estimated for each pair of views, they are used to infer the rototranslations that align each view in a global reference frame. A common approach consists in defining a fully-connected weighted graph in which each node represents a different view, whereas the edges represent the estimated rigid motion between the views connected by the edge (see Fig. 2.5). To each edge is assigned, as weight,

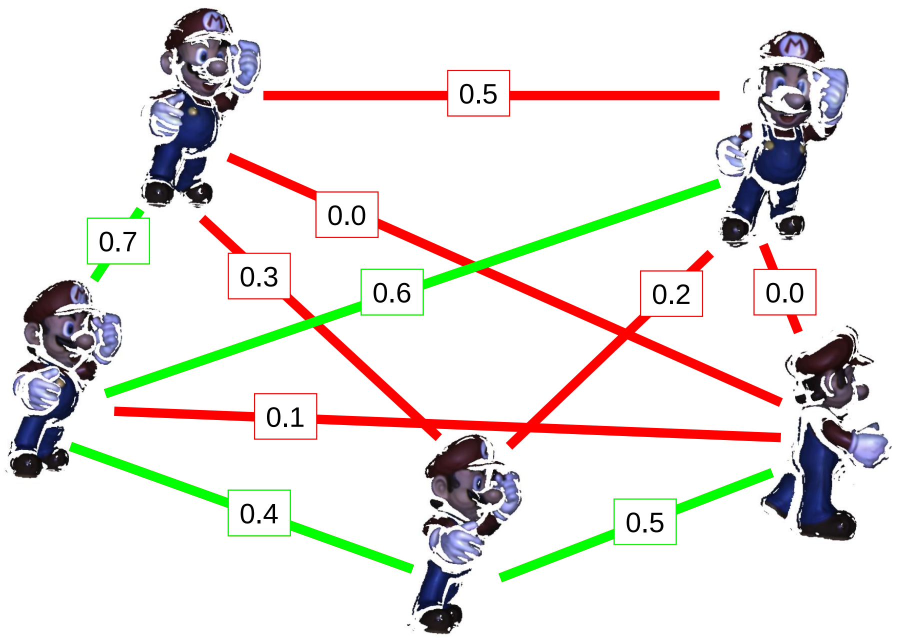


FIGURE 2.5: Fully-connected graph representing the rigid motions estimated between each pair of views. Edge weights are the percentages of overlapping area after the alignment, whereas green edges represent the maximum spanning tree extracted from the graph.

the percentage of overlapping area of the two views after the rigid motion is applied. An extended overlapping area suggests that two partial views share same portions of the object and the method succeeded in finding the correct rigid motion. To select the minimum subset of rigid motions that permits to align all the views and that, at the same time, maximize the probability to obtain a correct registration, the *Maximum spanning tree* is extracted from the graph (see again Fig. 2.5). Then, the identity transformation is applied to the view chosen as the root of the tree, whereas, for each other partial view, the concatenation of rigid motions that align it to the root view is computed and applied.

2.4 Fine global registration

Even though each pair of partial views are finely aligned through ICP, the global registration obtained as result of the previous stage does not usually come out sufficiently accurate to be correctly reconstructed (as shown in Fig. 2.1b). Such effect is due to the alignment errors that are negligible if considered individually, but that become conspicuous when they are accumulated as the rigid motions are concatenated. Such issue is typically solved by performing a further optimization that considers all the points at once and carrying out a joint minimization of all distances between corresponding points.

Former work that address such issue is [79], which proposes an optimization framework that spreads the alignment error evenly across view pairs. An implementation is provided by the author in the *Scanalyze* tool[1]. Nishino et al. in [73] apply conjugate gradient search to minimize the objective function and exploit M-estimators for robust outlier rejection. Fantoni et al. in [31] extend the idea of [34] to multi-view alignment. All these methods ground on the assumption that partial views are free from warpings and deformations and, therefore, rigid-body transformations are sufficient to a correct alignment. In [17], the authors argue that such deformations are not negligible as related to the unavoidable imperfection of sensor calibration. For this reason, they propose a framework that performs non-rigid warps to each partial view and jointly distributes error evenly across all meshes.

2.5 Mesh reconstruction

Once partial views are correctly aligned, the last step of the process aims at yielding a manifold and watertight mesh. A non-manifold mesh is a geometry that cannot exist in real world. Overlapping edges or

[1]`http://graphics.stanford.edu/software/scanalyze/`

triangles, disconnected vertices, or areas with no thickness are examples of causes of non-manifold geometries. A watertight mesh, instead, completely separates the internal volume of the object from the outside.

Typically, mesh reconstruction algorithms do not rely on the surfaces initially computed for each partial view, but discards all the triangles and computes the final surface on the basis of the points or, at most, their normals. The *Mesh zippering* algorithm, introduced in [99], is the only notable work that, instead, exploits the geometry of the initial meshes. Starting from two overlapping meshes, the algorithm performs tree steps to produce a single geometry. First, all the triangles of one mesh that overlap entirely the surface of other one are identified and discarded. Then, the triangles that still partially overlap the other geometry are clipped to perfectly adhere to the other geometry and finally the two meshes are merged. Such process is repeated for each partial view until all the views are integrated in the final mesh.

All the other relevant methods proposed over the years can be divided into two main categories: computational geometry based and implicit functions based.

2.5.1 Computational geometry based methods

Starting from a set of points sampled from a surface, the algorithms belonging to this category usually provide a resulting mesh composed by the initial points. For this reason they are more suited for low-level noise data scanned with high-precision sensors. On the other hand, they often provide theoretical guarantees for the quality of the resulting surface.

The *ball-pivoting* algorithm [7] is the first remarkable work falling under this category. Starting from a seed triangle, it is a region-growing method that searches for adjacent triangles by revolving a ball around each edge of the current triangle until the ball touches a new point. The edge extremes and the touched point define a triangle that is added to the mesh in a process that is repeated until all the points are scanned. The method assumes a uniform point density and a distance between adjacent points smaller than the user-defined radius of the ball.

Other approaches ground on the computation of the *3D Delaunay triangulation*, which partitions the volume enclosing the object in a set of tetrahedra whose vertices are the input points augmented with the eight volume corners. By observing that the sought surface is composed by the triangles shared by pair of inside/outside tetrahedra, the proposed methods differ only on the approach deployed for identifying tetrahedra located inside or outside the object. The *Powercrust* method, introduced in [5], exploits the *Voronoi diagram* - dual to Delaunay triangulation - that divides the volume so as to get, for each input point, a cell comprising all the points closer to that input point to any other. Under the assumption of a sufficiently dense sampling, the Voronoi tessellation is composed by long and skinny cells perpendicular to the surface. The algorithm selects, for each sampled point, the two most distant points in the corresponding Voronoi cell, called *poles*, so that a pole is inside and the other outside the object. Then, the method label the pole closest to the bounding box of the volume as outer and propagates the labels up to the inner poles. Finally, the Voronoi/Delaunay duality enables the proper labeling of the tetrahedra. Kolluri et al. in [55] propose the *Eigencrust* that, instead, builds a graph of the Voronoi vertices and applies a spectral graph partitioning to segment the tetrahedra.

2.5.2 Implicit functions based methods

This category of algorithms solves the problem through two main steps. The former computes an implicit function defined over a voxel partitioning of the volume enclosing the object. It estimates the signed geometric distance of the voxels to the unknown surface. As the distance is signed, most of the algorithms rely on point normals as additional information to ease this step. The latter extracts a zero-crossing isosurface as a polygonal mesh. This step is usually solved by means of the *Marching cube* algorithm [64] (or one of its variants), which takes the eight corners of each voxel and determines the triangles needed to represent the part of the isosurface that passes through the voxel. The extracted triangles are, then, merged in a single surface. Methods based on implicit functions are generally more robust to sensor noise and misalignments of the partial views as they perform an interpolation of the input points. On the other hand, the computation of the implicit function turns out the most critical part of the procedure and different proposal have been introduced to tackle this step.

Hoppe et al. in [42] compute the tangent plane of each input point and estimate the implicit function as point-plane signed distances. The work introduced in [27] grounds on the assumption that the input point cloud comes from a set of acquired partial views. A signed distance function is computed for each view by exploiting normal orientations and, then, they are merged together on the basis of the relative motion of the sensor. Finally, in 2006, Kazhdan et al. proposed the *Poisson reconstruction* method [53] that can be considered the de-facto standard algorithm for mesh reconstruction. Instead of estimating the implicit function, the method computes the *3D indicator function* defined as 1 at points inside the object and 0 at points outside. The key idea stems from the observation that the gradient of the indicator function is a vector field that is different from zero only at points near the surface, where it is equal to the surface normal. Thus, the computation of the indicator function can be obtained by inverting the gradient operator. As the divergence of the gradient (laplacian) of the indicator function equals the divergence of the normal vector field, the solution of the problem reduces to computing the divergence of the normal vector field and to solving the Poisson equation.

Chapter 3

Pairwise registration by local orientation cues

As described in the previous chapter, 3D registration is generally handled by describing a set of feature points extracted from the partial views. The descriptions are subsequently matched so as to find a set of correspondences from which estimate the rigid motion. A fundamental trait that any feature descriptor deployed in 3D registration (and, more broadly, in surface matching) ought to possess is invariance to pose.

Indeed, not all the feature descriptors proposed in the last years are inherently pose-invariant. For example, both [14] and [35] compute multiple descriptions, one for each angular subdivision of the support. This requires the matching stage to perform circular shifts in order to evaluate the similarity between two descriptors. In other proposals, though, the description method is itself endowed with invariance to pose. *Spin Images* [51], *FPFH* [81] and *Normal/Integral Hash* [3] are histograms wherein the contributions of the points within the support are related to the normal at the feature point only. [19] treats feature points by *Hidden Markov Models* that intrinsically own pose invariance, whereas [61] and [69] take the magnitude of the Fourier Transform of the descriptor, thereby gaining invariance to rotation which is encoded into the phase.

However, the most widespread approach ([57, 70, 6, 83, 89, 24, 91, 106, 95, 94, 54, 104]) to attain pose-invariance deploys a *Local Reference*

Frame (LRF) centered at the feature point and attached to the surface regardless of its orientation. Thereby, description can encode local shape traits with respect to a canonical reference associated with the feature point. The findings reported in [95, 78, 77] highlight clearly how the repeatability[1] of the computation of the LRF is key to robustness of the description and, accordingly, to the performance of the overall feature matching process.

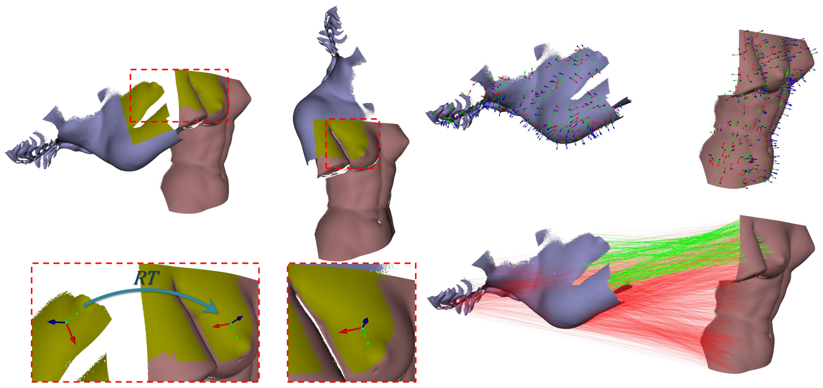


FIGURE 3.1: Alignment of two partial views by the method described in the chapter. The small overlap area between the two scans is highlighted in yellow. The zoom-in panels conveys the key idea of the approach: the only information deployed to establish correct correspondences deals with LRFs attached to feature points. Though, as illustrated by the panels, a single correct one would suffice, we deploy many correspondences to estimate the rigid motion robustly (Right, with green/red lines representing correct/wrong correspondences).

Although effective and fast algorithms pertaining computation of local reference frames have been devised [95, 78, 83, 77], in the field of registration, LRFs have so far only been considered instrumental to feature description. Conversely, this chapter proposes a different and novel registration paradigm, which stems from the observations that LRFs can indeed provide basic shape cues and that two corresponding points equipped with their LRFs allows the rigid motion that aligns two views to be computed (as illustrated in the zoom-in panel of Fig. 3.1). More precisely, we rely on the method proposed in [77] to compute

[1]An LRF algorithm is said to be repeatable when it provides the same canonical orientation across different views of a surface patch. Specific figures of merit to quantify this property have been proposed in literature [95, 78, 77].

highly repeatable LRFs at feature points and show how such computation provides the core of a coarse registration pipeline which does not require a costly feature description stage. Thanks to the minimal feature description, which also implies a light downstream correspondence process, the ensuing pipeline turns out remarkably fast without any loss of registration efficacy.

It is worth pointing out that the idea of using the LRF attached to points to estimate the rototranslation to align two views can be found also in [68], [32] and [70]. However, such papers rely on a single correspondence, whilst we deploy an *Hough Voting* scheme [96] to robustly account simultaneously for many correspondences (Fig. 3.1, Right). More importantly, the pipeline in [70] follows the standard paradigm whereby LRFs are primarily deployed to establish a canonical reference for the purpose of feature description. On the other hand, the work in [32] turns out infeasible unless views consist of quite a small number of points, as it consists in a RANSAC-based approach bound to operate with just a small fraction of outliers and it mandates running as many nearest neighbour searches as the number of points to determine the consensus set.

Although our registration pipeline can be feed with any kind of 3D feature points, as a second contribution of this chapter we propose a novel detector specifically conceived to provide features suited to our method. In particular, we argue that the underling saliency cue should capture the *orientability* of features, i.e. the ability to compute the LRF repeatably despite feature localization being possibly inaccurate. Accordingly, we develop an efficient algorithm which conveniently deploys the observed relationship between *orientability* and *flatness* to quickly extract features particularly suitable for our pipeline and uniformly distributed throughout the views, the latter being a beneficial property for coarse registration.

3.1 Pipeline Description

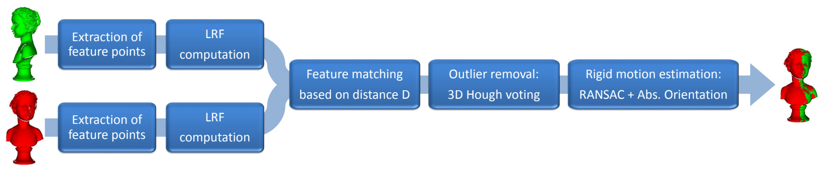

Extraction of feature points
LRF computation
Extraction of feature points
LRF computation
Feature matching based on distance D
Outlier removal: 3D Hough voting
Rigid motion estimation: RANSAC + Abs. Orientation

FIGURE 3.2: Outline of the proposed pipeline: LRF computation is key to match feature points based on an elementary shape cue (D) as well as to prune most outliers by Hough Voting. The final pose estimation is accomplished by RANSAC.

Given two partial views of an object acquired from different vantage points, V_I and V_J, the aim of a pairwise registration pipeline is to find the rigid motion that aligns the views so that the shared surface portions do overlap. Our method, outlined in Fig. 3.2, starts by extracting two sets of feature points, F_I and F_J, from the two views. This can be achieved either by random selection of a given number of points or by the feature detection algorithm described in Section 3.2.

3.1.1 LRF estimation

The second step of our pipeline differs significantly from mainstream literature approaches. As explained, we dismiss the time-consuming description stage carried out at this level of the pipeline and instead keep only the local reference frame computation ordinarily devoted to endowing description with invariance to pose. Purposely, we deploy the method introduced in [77], which requires the normals[2] associated to points and the computation of the *mesh resolution* (hereinafter mr, computed as either the average length of the edges of the meshes or, should the dataset consist of point clouds, the average distance between neighbouring points).

[2]The normals are computed by the *vtkPolyDataNormals* function of the VTK library (`http://www.vtk.org`), which computes the normal of each triangle and, then, computes the normal of each point by averaging the normals of the triangles connected to the point.

Given a feature point p, the algorithm starts by robustly estimating the z axis (colored in blue in Fig. 3.3) as the normal to the plane, π_z, that best fits the points within a small spherical support of radius R_z centered at p (depicted in blue in Fig. 3.3). The sign of the z axis is disambiguated so as to orient it coherently to the average normal computed over support points. It is worth pointing out that the algorithm is not critically affected by the stability of the computed normals as they are used - after having been averaged - only to disambiguate the sign of the z axis. A different support is considered for estimation of the x axis, namely the intersection between the surface and the spherical shell centered at the feature point and defined by the radii pair $[0.85 \times R_x, R_x]$ (depicted in red points in Fig. 3.3). Considering such points, the signed distance from plane π_z is evaluated and the point, p_D, exhibiting the largest distance, D, is selected. The x axis (represented in red in Fig. 3.3), is attained by projecting onto π_z the vector from the feature point to p_D. The y unit vector (shown in green in Fig. 3.3) is then given by the cross-product $z \times x$.

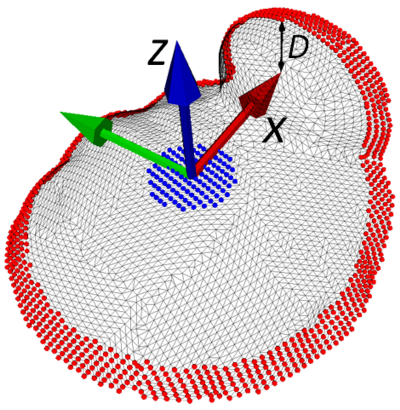


FIGURE 3.3: An example helping to describe the computation of the local reference frames.

The method in [77] possesses traits that render it particularly suited to the task of registration. First of all, the repeatability of the LRFs tends to increase significantly with the support size R_x. The main nuisance that would hinder repeatability along with increasing such a size turning out clutter, which, however, is not present in the registration task. This vouches that the method is robust to missing surface portions within the neighbourhood of a point, as it would happen at features located close to the boundaries of a view. Indeed, the repeatability of a LRF only depends on whether point p_D ends up or not in a missing region, such "highest" points tending to better withstand changes of the vantage point as they are less likely to be occluded by other surface patches. This is a quite favorable property in registration applications, as, when the views in a pair are acquired from angularly distant viewpoints, and thus are hard to align, most of the limited overlap is found at boundary regions. Another benefit of the approach dwells on its robustness to point density variations, both uniform, as induced by changes of the acquisition distance, and non-uniform, as determined by out-of-plane rotations of the sensor. Finally, the algorithm comes out fast even when applied to wide supports due to the small fraction of points actually involved in the computation.

3.1.2 LRF matching

Any pair of corresponding feature points equipped with correctly established LRFs defines the sought rigid motion. Stemming from this observation, the next stages of our pipeline aim at sifting out from the set of all the $F_I \times F_J$ candidate pairs a sizable subset of correspondences to estimate the rototranslation to align the views, i.e. to bring them to the same reference frame. We found a good cue to be the distance, D, inherently associated with each feature point upon computation of the LRF. Indeed, as the LRF algorithm establishes a canonical reference, D turns out to be a basic measurement related to the local shape of the surface around the feature point. Accordingly, assuming LRFs to have been correctly established, corresponding features should exhibit similar D values. This property can be exploited to discard candidate correspondences between feature points showing significantly different D values. To this purpose, the difference $D_{ij} = |D_i - D_j|$ is computed and normalized with respect to $D_{max} = \max_{i,j}(D_{ij})$ for each candidate pair. If the normalized difference, $\widetilde{D}_{ij} = \frac{D_{ij}}{D_{max}}$, is above a threshold, T_D, the pair is discarded. This is vouched by Fig. 3.4, in which we consider two very different datasets and show as a blue curve the *pdf* of $\widetilde{D}_{ij}$, $p\left(\widetilde{D}_{ij}|P\right)$, for good correspondences, P. Despite the diversity in the data, both curves look very similar and clearly show that when $\widetilde{D}_{ij}$ exceeds a certain threshold (such as e.g. 0.2) a feature correspondence is unlikely to be correct.

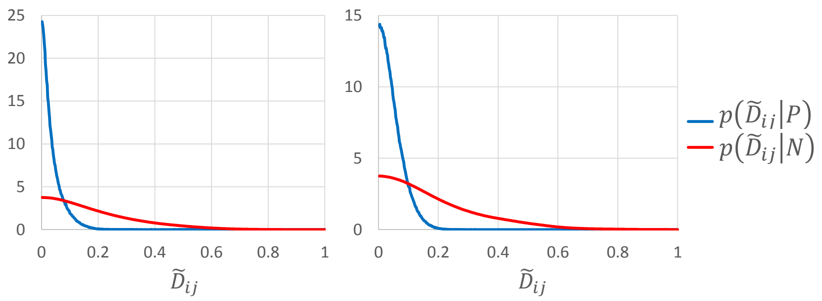


FIGURE 3.4: Probability density functions of normalized differences, $\widetilde{D}_{ij}$, for feature correspondences dealing with the high-quality *Neptune* dataset (left) and low-quality *DuckKinect* (right) dataset (see sec. 4.1). Blue and red curves concern correct and wrong correspondences respectively.

The proposed matching process is really fast, as it boils down to sorting both the features in F_I and F_J with respect to D and simultaneously scanning the sorted lists to collect those pairs that do not satisfy the previous pruning condition. In standard pipelines, instead, the matching stage is typically expensive as it involves computing distances in a high-dimensional description space so as to retain pairs of features appearing close one to another in that space. Furthermore, as shown in Fig. 3.5, if either of the two supports around corresponding features gets spoiled due to missing surface regions, the associated descriptors will be corrupted alike, so that, typically, the features would result far away one to another in the description space. This does not happen in our matching scheme due to the LRF algorithm being resilient to missing regions: as long as the LRFs have been correctly established, the

D values related to two corresponding features turn out similar. The proposed matching approach turns out also very robust to noise.

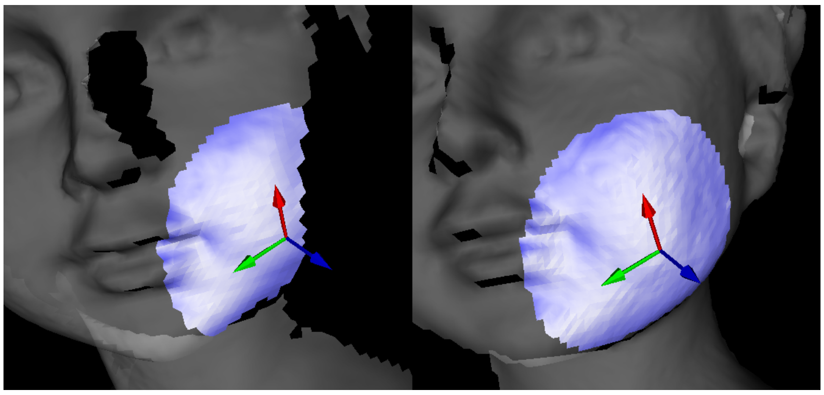


FIGURE 3.5: An example of the robustness of the LRF algorithm. Even though one of the two corresponding supports shows a wide missing region, the associated local reference frames are correctly aligned.

Indeed, the D value of the “highest” point p_D is typically large and, as such, it is unlikely that noise at p_D would decrease D as much as to promote another point far away from p_D to become the “highest” point. More likely, noise may promote another point in the vicinity of p_D, which, however, implies a tolerable error. Fig. 3.6 shows the result of an experiment carried out on the *Bunny* dataset in order to analyze the behavior of the matcher while different levels of Gaussian noise are injected into the data. For each view, 2000 points have been randomly extracted and matched setting $T_D = 0.01$ (see Table 4.1) so to evaluate the mean number of correct (TP) and wrong (FP) correspondences established across all 45 view pairs. Moreover, the chart reports the number of correctly aligned LRFs (TP_LRF) according to the figure of merit $\bar{A}$ introduced in [77]. Hence, Fig. 3.6 highlights clearly the robustness to noise of the matching process as the curves remain very stable as the noise level increases.

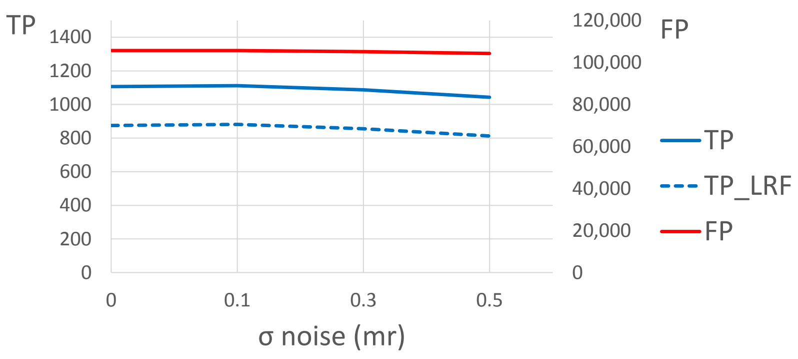


FIGURE 3.6: Feature matching on the *Bunny* dataset with increasing noise. The blue and red solid curves report, respectively, the average number of correct (left vertical axis) and wrong (right vertical axis) correspondences. The blue dashed curve highlights the average number of correct correspondences yielding aligned LRFs.

3.1.3 Hough voting

On the other hand, our basic shape cue D has a rather poor discriminative power compared to a high-dimensional descriptor, so that measuring close (or even identical) values cannot be treated as a sufficient condition to declare two features as corresponding. This is shown, again, in Fig. 3.4, where the red curves represent $p\left(\widetilde{D}_{ij}|N\right)$, i.e. the *pdf* of $\widetilde{D}_{ij}$ concerning wrong correspondences (N). Though choosing a suitable threshold, T_D, somehow in the range $[0, 0.2]$, assures the preservation of the majority of inliers, it still brings in a large percentage of false correspondences. Therefore, we further prune outliers by enforcing geometric consistency constraints according to the *Hough Voting* method proposed in [96], which, again, relies on the availability of repeatable LRFs attached to features. First, for each feature F_j in V_J, the centroid C_J of V_J is expressed with respect to LRF_j, i.e. the canonical reference attached to F_j, in order to obtain the set of $\mathcal{LRF}_J^j(C_J)$, where the notation $\mathcal{LRF}_J^j(.)$ expresses the change of basis from the global reference frame of V_J to LRF_j. Then, for each pair of correspondences (F_i, F_j), the change of basis from LRF_i to the global reference of V_I, $\mathcal{LRF}_i^I(.)$ is computed and applied to the transformed centroid of

V_J: $\mathcal{LRF}_i^I\left(\mathcal{LRF}_J^j\left(C_J\right)\right)$. In other words, for each pair of correspondences, a candidate rototranslation $\mathcal{RT}_j^i\left(.\right)$ is estimated as the transformation that aligns LRF_j to LRF_i: $\mathcal{RT}_j^i\left(.\right) = \mathcal{LRF}_i^I\left(\mathcal{LRF}_J^j\left(.\right)\right)$. This is used to move the centroid C_J of V_J to the new position $\mathcal{RT}_j^i\left(C_J\right)$. Correct correspondences (F_i, F_j) will vote coherently for the position of C_J in V_I, i.e. the estimated motions will tend to localize $\mathcal{RT}_j^i\left(C_J\right)$ in a unique position within the 3D volume associated with V_I, which is referred to as Hough Space in [96].

To implement the Hough Space, we compute the centroid and standard deviations of the x, y, z coordinates of the points in V_I. The origin of the Hough Space is given by the centroid and each dimension is taken as large as 4 times the corresponding standard deviation, so as to consider about 95$\%$ of the points of V_I and neglect possible outliers far away from the centroid. Moreover, the size of each dimension is further enlarged by a factor f_{hough}, to allow rotated centroids to fall outside the tight bounding volume around V_I. The thus defined volume is evenly quantized into cubic bins of side S_{bin}. Each of the $\mathcal{RT}_j^i(C_J)$ votes for the bin hit by the rototranslated centroid. Then, bin scores are computed by accumulating the votes falling into the $3\times3\times3$ neighbourhood centered at each bin. Eventually, the bin showing the highest score is selected in order to sift out the correspondences to be used to estimate the rigid motion to align the views.

The matching process based on the distance D and the Hough voting stage contribute synergistically to effective filtering of wrong correspondences. For example, considering again the experiment of Fig. 3.6, when exacting 2000 random features from each partial view of *Bunny*, the matching stage sifts out, on average, 2.66$\%$ of all the $F_I \times F_J$ candidate pairs, i.e. 106768 correspondences, whereas the further pruning performed by the Hough voting delivers about 0.54$\%$ of the pairs provided by the matching process, so to forward to the final stage of our pipeline only 561 correspondences on average.

3.1.4 Rigid motion estimation

Nonetheless, such pairs are not guaranteed to be inlier correspondences only. Indeed, for the sake of memory efficiency, the method in [96] relies on a 3-dimensional (translation only) rather than 6-dimensional (translation plus rotation) Hough Space, thereby allowing, in principle, different rigid motion hypotheses to vote for the same bin. Likewise, quantization effects may determine different hypotheses to collapse into a single bin. Therefore, given the correspondences associated with both the highest bin and its neighboring bins, we carry out the final rigid motion estimation stage robustly by applying the standard Absolute Orientation algorithm proposed in [43] within the RANSAC framework [33].

Finally, it is worth highlighting that the idea of matching 3D points based on LRFs has been proposed also in [71]. However, their work is not related to pairwise registration but instead aims at establishing correspondences between the points belonging a single mesh for the sake of symmetry detection. Accordingly, they deploy the curvature tensor defined in [4] as LRF and the ratio between the two principal curvatures as matching cue. The experimental evaluation reported in the next chapter demonstrates that, differently from our approach, the curvature-based LRF and matching cue adopted in [71] are not effective for the task of pairwise registration.

3.2 Feature Extraction

The registration pipeline described so far is agnostic with respect to the kind of features extracted in the first stage. As such, one may rely on random extraction of feature points or use any of the several 3D detectors proposed in literature (see [97] for a recent survey and evaluation of prominent proposals).

Unfortunately, the evaluation in [97] highlights that the computational efficiency of all considered proposals is by far unsatisfactory, so there is no algorithm that may be plugged into our pipeline without exceedingly slowing down the computation. Moreover, existing detectors rely on maximizing a specific saliency criterion which inherently privileges certain shape structures so that, generally, features tend to cluster in some areas rather than scatter uniformly across a view. However, for the purpose of accurate estimation of the rigid motion between two views, it is highly beneficial to rely on features as uniformly distributed as possible across the views. Accordingly, we conjecture that a suitable feature detector for coarse registration should better provide a good trade-off between saliency and uniformity rather than maximize saliency.

Based on the above considerations, investigation on suitable features to be fed into our pipeline seems to call for the design of a novel 3D detector that would provide rapidly salient and uniformly distributed points. First of all, this requires reasoning about the saliency criterion. Unlike mainstream work on the subject, our coarse registration pipeline is rooted only on the ability to compute LRFs repeatably despite nuisances. Accordingly, a suitable saliency criterion would capture this ability: "good" features for our pipeline are points where the LRF can be computed repeatably. We dub *orientability* such a peculiar saliency criterion[3].

In the LRF algorithm proposed in [77], the repeatability of the x axis is significantly dependent on the stability of the z axis, as the latter provides the reference plane to compute the signed distances that then would define the former. Thus, as the stability of the z axis is clearly highest at points located in flat surface areas, it seems that with the method in [77] there is an inherent relationship between *flatness* and *orientability*. To validate this intuition, we performed a qualitative experimental study aimed at comparing the *flatness* and *orientability* of

[3]Our notion of *orientability* differs from the use of this term to denote the property of consistently disambiguating the sign of the normal at every point of a surface.

surface points. Given a surface point p together with its normal n_p, the *flatness* at p is higher as the normals at the points within a neighbourhood of p are more closely aligned to n_p. Therefore, we define the *flatness* at p as the mean cosine between n_p and the normals at the points within a sphere of radius R_f centered at p. As the nuisance inherently associated with feature extraction is imprecise localization of feature points, the resilience to be captured by a proper *orientability* notion should address this type of nuisance. Thus, given a point p, the corresponding points p_j in the other views of a dataset are determined by applying *ground truth* rigid motions between the views (ground truth information is available for all the considered datasets). According to the above mentioned notion, p would exhibit high *orientability* whenever the LRF computed at p is correctly aligned to those computed at points p_j despite localization of the latter turning out imprecise. Hence, to capture this property, we compute the LRF at all the points, $p_{k,j}$ falling within a neighbourhood centered at each of the p_j, so as to then establish whether the LRF computed at p is correctly aligned or not to that computed at each $p_{k,j}$ (i.e. aligned or not to that computed at corresponding though imprecisely localized features). To establish whether any two such LRFs are correctly aligned or not we rely on the repeatability criterion proposed in [77] and, accordingly, calculate the *orientability* index at p as the percentage of correctly aligned LRFs.

Fig. 3.7 allows a visual comparison of the *flatness* and *orientability* maps of a partial view of the *Bunny* object of the Stanford Repository [26]. As a first consideration, the left map shows that our measure of flatness captures the curvature of the shape properly. Besides, as for the orientability map, it is worth highlighting the high orientability of the majority of points as a further proof of the effectiveness of the method adopted to compute LRFs. But more importantly, the comparison shows clearly that there exists a relationship between flatness and orientability. In particular, many red points in the left map turn out red also in the right one: the repeatability of the LRFs is usually poor at surface areas featuring a pronounced curvature so that the computation of the z axis

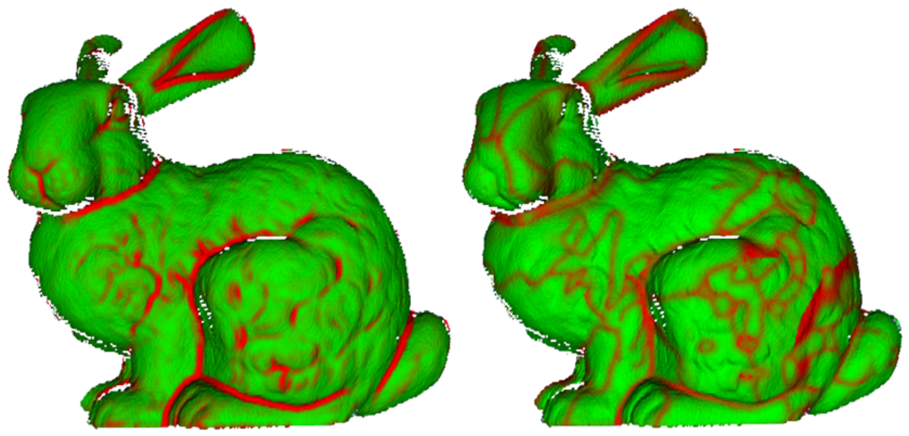


FIGURE 3.7: Flatness Vs Orientability. On the left the flatness map of a view, on the right the corresponding orientability map. The green colour represent flat and highly-orientable points respectively, red points feature high curvature in the flatness map and poor LRF repeatability in the orientability map.

turns out to be unstable (e.g. the ears of the *Bunny*).

As in real registration settings the rigid motions between views are obviously not available (we indeed seek to estimate them!), the defined orientability index cannot be measured directly in practice. Hence, the idea stemming from our analysis is to try to extract flat points instead, as with respect to our pipeline they are more likely to be salient (i.e. orientable) than high-curvature ones. As such, whereas most detectors present in literature are based on extraction of points exhibiting high curvature, we take, somehow paradoxically, just the opposite path. This approach provides at least two additional benefits, though. Firstly, computation of flatness is fast as it involves only inner products within a small supporting neighbourhood (as shown in Table 4.1 a small R_f suffices). Secondly, flat surface areas are less prone than high-curvature ones to self-occlusions caused by out-of-plane rotations of the sensor.

The devised feature detector is described with the aid of Fig. 3.8. The algorithm repeatedly extracts a random point p_{seed} (shown in light green in the zoom-in panel on the left side of the Figure). Then, the flatness index is computed at all the points within a spherical support of radius R_{search} centered at p_{seed} (highlighted in yellow in the panel), so as to

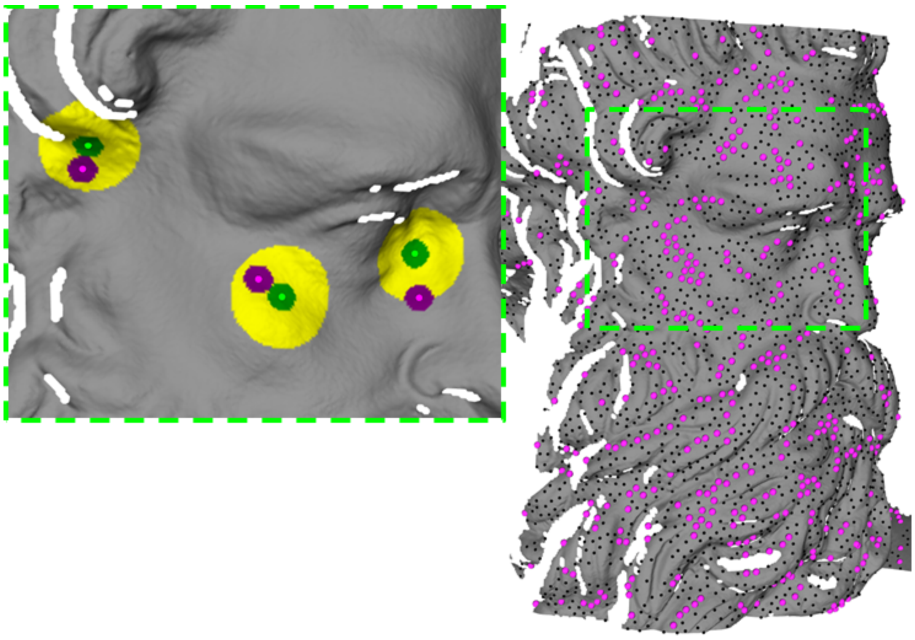
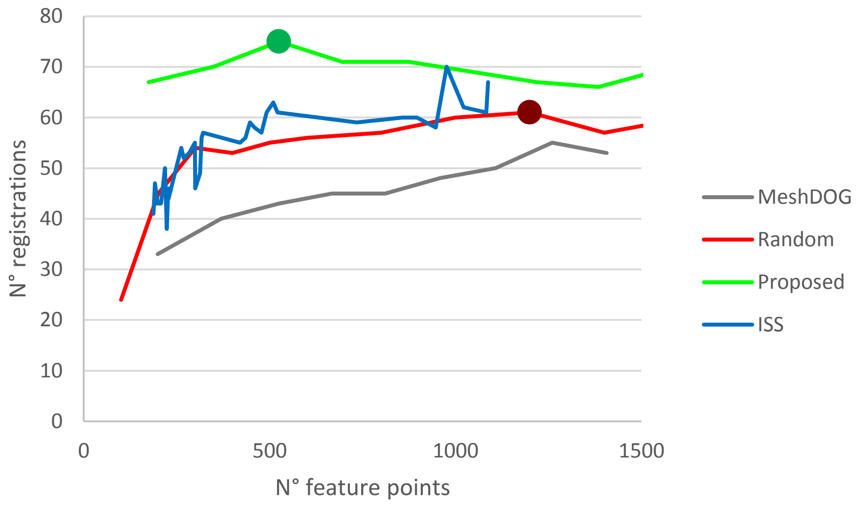


FIGURE 3.8: Exemplar feature extraction by the proposed algorithm. The zoom-in panel on the left shows the detection process at a glance: for each randomly chosen seed point (in green), the flattest point (in fuchsia) in its neighbourhood is extracted. The image on the right highlights the two-step extraction process: the points selected by the first step are shown as black smaller dots, while the final features provided by the second step as larger ones.

pick-up as feature the point, p_{max}, turning out maximally flat (coloured in fuchsia in the panel). To avoid further detections in the proximity of already selected features, the points at a distance lesser than $R_{discard}$ from p_{max} (in purple in the panel) are pruned from the set of candidate feature points, and, alike, those around p_{seed} according to $R_{discard}$ pruned from the set of candidate random seeds (in green in the panel).

The method continues to iterate until the percentage of discarded points, either as potential feature, p_{max}, or random seed, p_{seed}, gets higher than a threshold, T_{area}, which is tightly correlated to the fraction of the view that one wishes to explore during the feature extraction process. As for the requirement to trade-off between saliency and uniformity, random extraction of seeds ensures feature points to be scattered throughout the partial view, while the subsequent flatness maximization weighs in favor of saliency. Regarding efficiency, the critical step of the algorithm consists in gathering the points inside the support of radius R_{search}, which needs to be large enough in size and thus may slow down the adopted kd-tree search. To overcome this issue, we devised the two-step approach illustrated in the right image of Fig. 3.8. In the first step, the method is applied to the whole view with a small radius, R^1_{search}, to search for maximally flat points around random seeds. Due to R^1_{search} being small, the first step efficiently subsamples the view so as to provide a subset of candidate flat points. The second step consists of running the process with a larger radius, R^2_{search}, on the subsampled view made out of the candidate features provided by the first step only. Accordingly, much fewer instances of the slower search do take place and efficiency is not penalized. The termination threshold for the first step, T^1_{area}, is set high enough to explore a large fraction of the input view, whereas the threshold for the second step, T^2_{area}, represents the parameter that actually controls the amount of extracted features points.

We performed comprehensive experiments to validate the improvement brought in by the proposed feature extraction algorithm. In particular, we compared the performance delivered by our pipeline with features provided by the proposed detector, random extraction (our initial choice), Intrinsic Shape Signatures (*ISS*) [106] and *MeshDOG* [104]. According to the evaluation in [97], *ISS* and *MeshDOG* may be regarded as prominent fixed-scale and adaptive-scale detectors, respectively. Indeed, in their respective categories, both turn out to be the fastest and rank quite high in terms of repeatability. We found that the pipeline deploying the proposed detector delivers the best performance consistently across datasets: it can align a higher number of view pairs, and, in the event of similar registration rates, it comes out, on the whole, the fastest. This is perhaps surprising, as one might guess that random detection would deliver the highest efficiency. However, though our detection scheme is obviously slower than random extraction, it provides



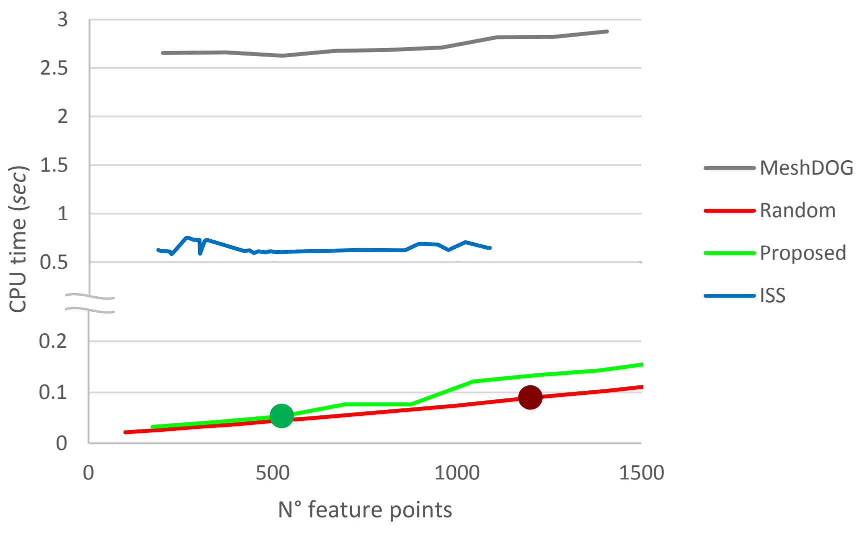


FIGURE 3.9: Comparison on the *Buste* dataset between proposed detector, random extraction, *ISS* and *MeshDOG*. In both charts the figure of merit is plotted as a function of the mean number of extracted feature points. The top chart reports the number of view pairs aligned correctly by our pipeline, whereas the bottom one shows the mean computation time required to align two views. To better compare our proposal to the random detector, the working points providing the highest number of correct alignments are highlighted by dots in both charts. Chapter 4 explains how the figures of merit plotted in the two charts are calculated.

better features, that is, according to our saliency notion, inherently endowed with more repeatable LRFs, so that a smaller amount of features needs to be forwarded to the next stages of the pipeline, which makes the overall computation faster.

The above described behavior is well pictured in Fig. 3.9, which shows the results regarding the registration of all the view pairs composing the *Buste* dataset (see sec. 4.1). The charts report the number of correctly aligned view pairs (top) and the mean CPU time to align two views (bottom) as a function of the average number of feature points detected in the partial views, which can be controlled by the user through a parameter in each of the four considered detectors. The top chart highlights how, regardless of the chosen number of extracted features, the proposed detector significantly improves the effectiveness of our pipeline with respect to the other detectors. It is worth highlighting here that, although any feature detection algorithm is inherently more repeatable than the random detector, the kind of saliency deployed by *ISS* and *MeshDOG* does not seem particularly suited to our pipeline. Indeed, the improvement provided by ISS over random extraction is modest on average and also not consistent across working points, whereas using the keypoints extracted by *MeshDOG* leads to lower registration rates than randomly extracted features. This may be explained by observing that, as also illustrated in Fig. 2 and Fig. 5 of [97], *ISS* and *MeshDOG* tend to fire on high-curvature structures, like those depicted in red in Fig. 3.7, than on flattish areas, as instead would be required by the saliency notion deployed in our pipeline. As for the bottom chart of Fig. 3.9, at first sight the computational efficiency would seem somehow comparable between our detector and random extraction, with *ISS* and *MeshDOG* definitely turning out to be slower[4]. However, a deeper analysis reveals that our detector can improve the efficiency of the pipeline. Indeed, as

[4]Due to software compatibility issues we could not run *MeshDOG* on the machine used to measure the computation times of the other detectors. Thus, the timings for *MeshDOG* reported in the bottom chart of Fig. 3.9 represent our best estimation of the actual efficiency of the algorithm.

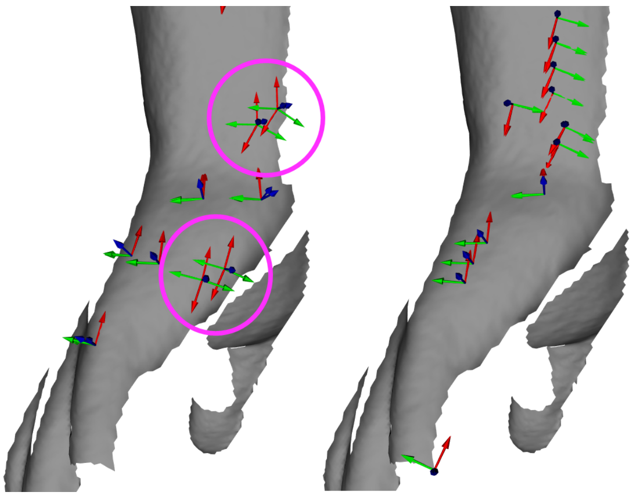


FIGURE 3.10: LRFs computed at corresponding features found on a portion of *Neptune*'s hand. The left image deals with randomly extracted features, the right image with features detected by our algorithm. Both images show the LRFs computed in V_I together with those computed in V_J and transformed into V_I according to the ground truth rigid motion. Misaligned LRFs are highlighted by the pink circles.

highlighted by the two dots in the top chart, choosing the feature quantity that yields the highest number of correctly aligned pairs for both the random detector and our detector, we end up requiring 1200 features (so to align 61 pairs) and 525 features (so to align 75 pairs) respectively. As pointed out by the dots in the bottom chart, at these two working points the proposed detector does render the pipeline faster than random extraction of features.

Finally, in Fig. 3.10 we show a qualitative experiment aimed at comparing the repeatability of the LRFs computed at random features and feature points detected by our algorithm. We consider two views, V_I and V_J, of the *Neptune* dataset (see sec. 4.1), extract feature points by both methods and detect correspondences based on the *ground truth* alignment transformation. Then, for the two methods, we compute the LRFs for each pair of corresponding features and, deploying again the ground truth rigid motion, draw both the LRFs in one view (i.e. V_I): the two LRFs drawn at each point will look more aligned as they have been computed more repeatably in the two views. Accordingly, in the right image (our detector) all LRF pairs look correctly aligned, whilst notable misalignments can be perceived in the pairs depicted in the left image (random detector).

Chapter 4

Pairwise registration: Experimental Evaluation

Even though the literature on pairwise registration is mature and provides a great number of remarkable works, surprisingly, they lack adequate experimental evaluation. Indeed, between all those cited in Chap. 2, most papers present experiments on data acquired with a sole type of sensor, while just a few of them consider more than one modality. Even more questionably, no paper attempts a comparison to other proposals. Usually, only the assessment between different variants of the proposed algorithm is presented and, sometimes, no quantitative results are provided. The only exception concerns [70], which considers three well-known datasets and compares the proposed method to a pipeline based on *Spin Images*. Furthermore, though a significant number of surveys have been published (see e.g. [92]), only one experimental evaluation has been issued ([82]). Unfortunately, it considers proposals that nowadays would be regarded as baseline methods. Such condition of things might be due to the lack of an acknowledged benchmark, let alone standard methodologies, on which to ground the evaluation process. This issue is worsened by the need to obtain and make proper use of the author's original implementation for the sake of fairness in the evaluation, as well as by that of running lengthy processes to tune, on diverse kinds of data, the many parameters which typically characterize coarse registration algorithms. As a result, we found it exceedingly difficult, if not impossible, to shed light on which coarse registration pipelines are most effective and under which conditions.

This chapter, therefore, describes an extensive experimental evaluation we carried out on a large ensemble of datasets acquired with different sensors and shows how our method easily adapts to the different modalities. Moreover, we compare quantitatively our proposal to a pipeline relying on the popular *Spin Image* descriptor as well as to two more recent methods, the comparison neatly proving that our algorithm is capable to align many views that can not be handled by the three other considered methods. Regarding the *Spin Image* pipeline (SI), proposed in [50], we have used the implementation available in the *Mesh Toolbox*. Feature matching relies on the normalized linear correlation coefficient between *Spin Images*, with correspondences then filtered on the basis of the similarity measure and partitioned into groups geometrically consistent with respect to their *Spin Map* distances. Each such group defines a rigid transformation which is further verified by ICP so to finally select the rigid motion that better aligns the two views. Concerning more recent methods, the former is the *4-Points Congruent Set* algorithm (4PCS) by Aiger et al. [2]. It randomly extracts a quadruple of coplanar points from view V_I and searches for those quadruples in V_J that can be transformed into the quadruple in V_I through a rigid motion. For each candidate, the whole view is transformed and the number of points in V_J that are close enough to a point in V_I is counted. The process is repeatedly run with different random quadruples of V_I, so that the rigid transformation racking up the largest set of points is eventually selected. The latter proposal is a recent local approach introduced in [15] (BSL12) which works on range images only. After scale-invariant detection, feature points are encoded in a cyclic description which imposes the matching process to shift and match the descriptor as many times as the number of angular subdivisions. Then, correspondences are subject to a geometric consistency check based on rigidity constrains, and a final RANSAC step ends up with the estimated rigid motion.

Hence, we compare the performance of our method, referred to here as LRF, to those yielded by SI [50], 4PCS [2] and BSL12 [14, 15]. The original implementations of the three algorithms have been kindly provided by the authors who also helped us a great deal through personal communications to tune the parameters of their methods properly. However, as far as SI is concerned, instead of describing all the points in one view and a subset in the other, as done in the original code, we have introduced a slight modification in order to execute random extraction of keypoints in both views. This has been necessary as otherwise the pipeline would have resulted exceedingly slow, making it impossible to complete most of the experiments. Indeed, even with the introduced modification, it turned out infeasible to follow out the experiments on the datasets comprising the largest number of views, i.e. *OilPump*, *Venus* and *Shell*.

4.1 Datasets

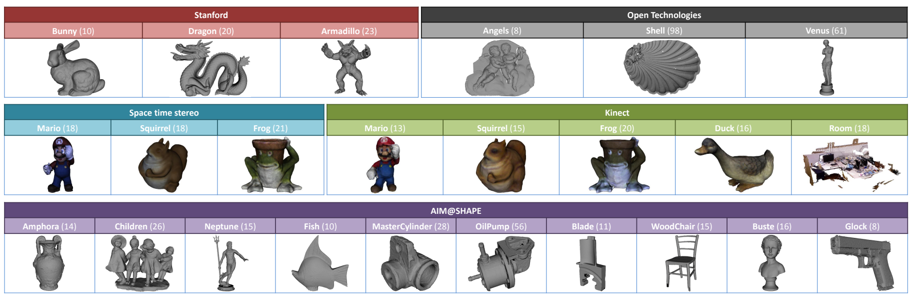


FIGURE 4.1: Thumbnails of the 24 datasets considered in the experimental evaluation. For each dataset the number of views, M, is reported between brackets.

An essential trait of a registration pipeline is to work successfully with different sensing modalities and under various real working conditions and nuisances. For this reason, as mentioned above, we evaluate the considered pipelines on the extensive collection of datasets detailed in Figure 4.1. Each dataset consists of a number of partial views for which normals are computed and the ground truth rigid motion between each view pair is available. Three of the datasets are taken from the very popular Stanford Repository [26], namely *Bunny*, *Armadillo* and *Dragon*, all acquired with a Cyberware 3030 MS laser scanner. Many other datasets come from the AIM@SHAPE Repository[1], and have been acquired with different scanners such as the very accurate Minolta VI910 (*Amphora*, *Children*, *Neptune*, *Fish*, *MasterCylinder*, *OilPump*, *Blade*, *WoodChair*), the Roland LPX-250 (*Buste*), and the low quality Minolta VI700 (*Glock*). Then, to extend the evaluation to different sensing modalities, we acquired four datasets in our laboratory by means of a Kinect device (*MarioKinect*, *DuckKinect*, *FrogKinect* and *SquirrelKinect*) and three datasets by a Spacetime Stereo set-up [28, 105] (*MarioStereo*, *SquirrelStereo*, *FrogStereo*). All our datasets were acquired by rotating the objects in the scene whereas the sensors were kept still. Then, we performed a background subtraction so as to hold only the partial views of the object. After that, we performed a coarse registration and, finally, we carried out a fine global registration by applying the *Scanalyze* tool. As a coarse registration algorithm should provide a rototranslation sufficiently correct to let the subsequent fine registration converge, such ground truth is sufficiently precise for validating the performance of the compared methods. We plan to render the datasets acquired in our Lab and the code of our method available for research purposes through the project web page. Moreover, we considered the *fr1/room* sequence from the *RGB-D SLAM Benchmark*[90], which deals with the reconstruction of an indoor scene acquired by a Kinect. In particular, we sampled the sequence every 10 frames so as to obtain a dataset comprising 18 partial views, referred to as *Room* in Figure 4.1. Finally, we added to the ensemble the three datasets scanned by an high-precision *Open Technologies*® system used for the experimental evaluation in [14, 15], i.e. *Angels*, *Venus* and *Shell*. The considered datasets feature different noise levels and point densities. *Kinect* datasets, and *Glock* alike, show the worst quality, with very

[1]http://shapes.aim-at-shape.net

noisy scans especially along the line-of-sight direction; *Open Technologies*® datasets, on the contrary, consist of very detailed and clean scans; *Spacetime Stereo* allows scans to be obtained with good precision and resolution. Moreover, every dataset comprises a different number of views and different degrees of overlap between them. *OilPump*, *Venus* and *Shell* respectively consist of, 56, 61 and 98 scans. On the other hand, *Glock*, *Angels*, *Bunny* and *Fish* include from 8 to 10 views. Even the physical objects they represents are quite different. *Open Technologies*® datasets deal with large objects, which have been scanned in the context of cultural heritage applications. Conversely, Stanford objects are extremely small. Furthermore, together with complex, rich of features objects, such as *Armadillo*, *Dragon* and *Mario*, we considered a set of mechanical parts (*MasterCylinder*, *OilPump*, *Blade*) as well as the *WoodChair* and *Room* datasets, so as to also address simple shapes and large flat surfaces that, as such, often turn out to be hard to reconstruct due to the scarcity of features. As range images are available only for the Stanford, *Open Technologies*®, Kinect and *MarioStereo* datasets, we have tested BSL12 only on this subset of datasets.

4.2 Methodology

The goal of a coarse registration algorithm is, in practice, to provide an alignment sufficiently correct to then permit a successful fine registration by ICP. Furthermore, even if the task is not subject to real-time constraints, execution time becomes relevant, especially when dealing with datasets comprising large sets of partial views. Finally, for those view pairs that have been coarsely registered, the accuracy of the alignment can be taken as an additional index of the quality of the algorithm. Given a dataset made out of M views, we consider all the $N = \frac{M(M-1)}{2}$ possible view pairs $\{V_I, V_J\}$, and, for each of them, attempt to estimate the rigid motion $\mathcal{RT}(V_J)$ that aligns V_J to V_I by means of the coarse registration algorithm under evaluation. Then, *Generalized ICP* [84] is

LRF

R_f	$R_{discard}$	R^1_{search}	R^2_{search}	T^1_{search}	T^2_{search}
$5 \cdot mr$	$2 \cdot mr$	$2 \cdot mr$	$20 \cdot mr$	0.9	0.9

R_z	R_x	T_D
$5 \cdot mr$	$[10, \quad 250] \cdot mr$	0.01

f_{hough}	S_{bin}	T_{RANSAC}	N_{RANSAC}	P_{RANSAC}
1.4	$2 \cdot mr$	$8 \cdot mr$	1000	0.99

SI

N_{points}	S_{bin}	N_{bin}	T_{angle}
$[1000, \quad 4000]$	$1, \quad 2, \quad 3 \ \cdot mr$	20	$^{\pi}/_{2}$

λ	$T_{saliency}$	T_{corr}	T_{vote}	T_{gc}	T_{ICP}
3	3	0.5	0.5	0.1	$4.0 \cdot mr$

4PCS

δ	f	T	N_{points}	D_{norm}
$[0.1, \quad 0.5]$	0.7	1.0	300, 500, 700	30.0

BSL12

σ	$N_{octaves}$	N_{maps}	$f_{sampling}$
$[0.5, \quad 4.0]$	3	2	2

M	L	Q	U
3	12, 24, 36	150	150

Generalized ICP

N	ε	R
10	0.1	$8 \cdot mr$

TABLE 4.1: Parameters for the four methods and Generalized ICP.

applied to the pair $\{V_I, \mathcal{RT}(V_J)\}$, and the resulting view $\mathcal{ICP}(V_J)$ is compared to $\mathcal{GT}(V_J)$, the latter being the view obtained by transforming V_J according to the known ground truth rigid motion which aligns V_J to V_I. In particular, if the Root Mean Square Error (RMSE) between $\mathcal{ICP}(V_J)$ and $\mathcal{GT}(V_J)$ is lower than $5 \times mr$, V_I and V_J are judged as

LRF

R_f	$R_{discard}$	R^1_{search}	T^2_{search}
$8 \cdot mr$	$8 \cdot mr$	$8 \cdot mr$	0.5

R_x	S_{bin}	T_{RANSAC}
$150 \cdot mr$	$6 \cdot mr$	$7 \cdot mr$

SI

N_{points}	S_{bin}	T_{ICP}
5000	$4 \cdot mr$	$7 \cdot mr$

4PCS

δ	f	N_{points}
0.2	0.5	700

BSL12

σ	L
4.0	36

TABLE 4.2: Different values for *Open Technologies*® datasets.

correctly registered by the algorithm under evaluation, otherwise a registration failure is recorded. In the event of successful registration, the RMSE between $\mathcal{RT}(V_J)$ and $\mathcal{GT}(V_J)$ is also recorded for the purpose of estimating the accuracy of the algorithm. Therefore, for each dataset and algorithm we collect the following three measurements. *$N°$ Registrations*, i.e. the number of correctly aligned view pairs; *CPU time*, i.e. the average execution time to compute the rigid motion to align a view pair (regardless the outcome being either success or failure); *RMSE*, which represent the average accuracy (i.e. RMSE) across all correctly registered view pairs. It is important to point out that *$N°$ Registrations* is the key performance index that captures the ability of the algorithm to handle view pairs featuring different degrees of overlap. The higher is the registration rate, the more effective in aligning views sharing a limited surface area is the algorithm.

4.3 Parameters

As outlined in the introduction of this chapter, coarse registration algorithms include many parameters which are hard to set properly. Although default settings are typically suggested by the authors, more often than not these guidelines come from insights gained by running the algorithm on one specific kind of data. Unfortunately, it turns out that such "standard" values are unlikely to make the method equally effective on diverse data. Thus, running our evaluation using a fixed set of parameters for each method would be unfair and also useless for shedding light on the real limits and merits of the algorithms. On the other hand, to sift out the best from a method in real working conditions, i.e. new data and no ground truth, one should manually set many parameters on a trial-and-error basis, which is simply infeasible. Therefore, we believe that a method capable of working seamlessly on diverse kinds of data should allow the user to get the desired result by setting just one or two parameters by trial-and-error.

Based on these considerations we contacted the authors of the algorithms involved in the evaluation and, following their indications, defined proper default values for all the parameters but the two identified as those most critically affecting performance, which we then tuned optimally on each dataset by an exhaustive grid search so as to maximize the number of correct registrations, *$N°$ Registrations*. In particular, the two parameters selected by authors of BSL12 are the Gaussian kernel size, σ, to be varied in the range $\{0.5, 4.0\}$ with 0.25 as step, and the number, L, of angular subdivisions of the descriptor, with possible values 12, 24 and 36. As for 4PCS, their authors decided to span δ in the range $\{0.1, 0.5\}$ with increment 0.1 and three possible numbers of extracted points N_{points} (300, 500, 700). Finally, SI has been tested by spanning the number of extracted keypoints, N_{points}, in the range $\{1000, 4000\}$ with step 500 and trying three values (1, 2, 3) for the size of the Spin Image bins. Likewise, we chose suitable defaults for all the parameters of our method but one, i.e. R_x, which was left free to vary in the range $\{10, 250\}$ with step 10 so to maximize N° *Registrations* on each dataset. R_x depends on the mesh resolution as well as on the extent of the flat areas of the objects, and it turns out to be the only parameter of our method that varies significantly across diverse data. Instead, the tuning we performed on the other methods has shown that at least two parameters affect their performance critically. Table 4.1 summarizes the parameter values adopted for each method, together with those related to *Generalized ICP*.

Open Technologies® datasets are indeed very different from all the other datasets considered in our evaluation, and in particular are characterized by a far higher point density and size. Moreover, BSL12 has been specifically designed to deal with this kind of data and therefore the authors already tuned their parameters optimally for *Open Technologies*® datasets. On the other hand, we found that the default settings chosen for LRF, 4PCS and SI are less appropriate on these datasets. Therefore, for the purpose of comparative evaluation, we found it fairer to also allow the other three methods to determine their default settings for these so diverse data. The authors of 4PCS kindly determined the parameter values of their method, and we did so for LRF and SI. The parameter values that turned out different from Table 4.1 are listed in Table 4.2.

4.4 Results

The results of the evaluation are summarized in Table 4.3, with the adopted color code (the darker the better for all the three indexes) helping to catch, at a glance, the relative performance of the algorithms. Accordingly, our proposal can align a larger number of view pairs with almost all the datasets (21 out of 24). Moreover, in terms of accuracy, our algorithm is only equaled by SI, despite the higher registration rate implying considering more challenging pairs in the computation of the

	N° Registrations				RMSE (mr)				CPU time (sec)			
	LRF	SI	4PCS	BSL12	LRF	SI	4PCS	BSL12	LRF	SI	4PCS	BSL12
Bunny (31/45)	27	20	16	14	1.09	0.74	3.77	2.37	0.74	338.53	1177.54	0.17
Dragon (108/190)	98	65	70	36	1.50	0.80	3.40	3.04	0.39	235.69	780.49	0.11
Armadillo (141/253)	118	72	95	39	0.94	0.95	3.23	2.25	0.20	72.50	453.07	0.12
Angels (18/28)	15	16	5	19	2.09	2.72	20.26	3.64	5.11	1074.31	386.86	0.58
Shell (822/4753)	620	--	104	646	1.71	--	22.32	3.24	1.21	--	258.07	0.48
Venus (256/1830)	207	148	51	160	2.25	2.91	15.94	4.54	0.68	414.2	393.91	0.25
MarioStereo (61/153)	58	44	57	38	4.10	4.03	6.44	9.66	0.65	136.28	19.94	0.10
SquirrelStereo (68/153)	44	34	33		2.77	3.43	4.74		0.54	112.68	358.96	
FrogStereo (83/210)	70	37	63		3.19	3.12	5.73		2.11	252.65	221.57	
MarioKinect (39/78)	30	26	32	9	2.32	1.56	3.33	8.49	0.08	115.04	951.67	0.01
SquirrelKinect (62/105)	34	33	33	9	2.24	1.33	2.04	5.41	0.03	88.32	1312.83	0.05
FrogKinect (96/190)	72	47	54	29	2.14	1.63	3.48	7.92	0.23	159.06	392.15	0.10
DuckKinect (61/120)	43	39	39	15	2.28	1.98	3.99	9.02	0.21	110.29	25.49	0.10
Room (55/153)	40	24	25	32	5.42	7.04	7.16	13.21	7.02	70.02	118.41	0.34
Amphora (63/91)	39	22	15		0.91	0.96	4.52		0.97	157.68	77.92	
Children (152/325)	123	96	81		0.82	1.07	6.25		1.82	65.13	64.69	
Neptune (34/105)	23	16	9		1.15	0.84	13.07		2.63	201.77	61.74	
Fish (22/45)	16	12	9		1.40	2.38	8.75		1.58	61.53	2.29	
MasterCylinder (245/378)	142	90	87		0.99	0.78	3.31		0.15	168.79	110.46	
OilPump (941/1540)	719	--	300		1.49	--	5.38		1.35	--	78.28	
Blade (29/55)	21	10	14		2.01	1.70	7.29		1.64	34.02	18.48	
WoodChair (58/105)	27	13	20		2.63	2.57	5.62		0.75	316.88	44.74	
Buste (85/120)	69	43	49		1.90	2.81	9.04		0.18	314.37	46.79	
Glock (16/28)	13	6	11		2.19	5.14	6.84		1.51	127.74	13.43	

TABLE 4.3: *N° Registrations*, *RMSE* and *CPU time* of the four methods on the 24 datasets considered in the evaluation. Darker colors denote, respectively, a larger number of view pairs correctly registered, more accurate alignments and faster computations. For each dataset, the number of view pairs that share at least 10% of their surface as well as the number of tested view pairs, N, are reported between brackets.

RMSE index. As for computational efficiency, BSL12 proves to be the fastest method although our pipeline fairly competes, with 4PCS and SI on the other hand turning out to be notably slower.

A more in-depth analysis of the results highlights that 4PCS overtakes our proposal solely on *MarioKinect* and for a couple of view pairs. Also, 4PCS obtains results comparable to ours on *SquirrelKinect*, *MarioStereo* and *Glock*, and gets good performances, in general, on *Kinect* and *Spacetime Stereo* datasets. However, 4PCS seems less suited to high resolution datasets such as those by *Open Technologies*®. Nonetheless, it is important to recall that, for each dataset, our evaluation provides the result that maximizes the registration rate, regardless of the associated computation time. By comparing the *CPU time* of LRF and 4PCS on *MarioKinect*, *SquirrelKinect*, *MarioStereo* and *Glock*, it is evident that 4PCS spends too much computational effort to obtain these high registration rates. Had the tuning process taken into account constraints on practically acceptable execution times, the registration rates of 4PCS would have turned out notably lower. It is worth observing that, between all datasets, *Neptune* is the one featuring the highest differences of point density between the views, as these include both close-ups on the head and wider scans around the body. The comparison between the performance of 4PCS and LRF on *Neptune* proves that significant benefits can be achieved by the latter, which is designed to handle point density variations robustly. As for BSL12, it turns out to be the best method on high resolution and clean data such as *Open Technologies*® datasets, i.e. the kind of data for which the method has been designed, tuned and tested. *Kinect* data lies exactly on the opposite side: low resolution and very noisy. Interestingly, Table 4.3 shows BSL12 to be much less effective on such diverse kind of data. It is also worth pointing out that on the *Venus* dataset LRF obtains a registration rate higher than BSL12. The peculiar cylindrical shape of the object causes acquisitions which involve mainly out-of-plane rotations of the sensor. On the contrary, the *Angels* object, a bas-relief, and *Shell*, acquired on one side only, permit a higher number of simpler in-plane rotations of the acquisition system. BSL12 relies on matching feature descriptors computed on 2D supports defined on range images: such kind of supports capture different portions of the physical space around a feature point due to out-of-plane rotations of the vantage point, which inevitably renders feature matching less effective. Differently, LRF relies on 3D supports, which are inherently invariant to any rigid motion of the sensor. Finally, SI provides registration rates similar to 4PCS on the low-precision *Kinect* datasets, and it seems to possess the ability to achieve good performance on the accurate *Open Technologies*® datasets, although this observation relies on the results pertaining the *Angels* and *Venus* datasets only. Generally speaking, even though SI does not turn out to be the best method on any dataset, it demonstrates a fair behavior across all the diverse sensing modalities considered in the evaluation.

The *RMSE* values reported in Table 4.3 vouch that the highest accuracies are obtained by our proposal and SI. As for the SI pipeline, the high accuracy is likely to be due to the verification stage that refines the final alignment through ICP. Instead, we ascribe the high accuracy of our proposal to the large quantity of uniformly distributed correspondences that survive the filtering stages of the pipeline and jointly participate in the final estimation of the rigid motion. This helps keeping the *RMSE* low across the entire surface acquired by a partial view.

Concerning computational efficiency, all the experiments were conducted on a PC equipped with a 3.50 GHz Intel i7 CPU and 16 GB RAM. The results show that the fastest method is BSL12, whereas, usually, 4PCS and SI spend conspicuously higher times to align views. However, BSL12 is optimized to work on multi-core architectures whilst our proposal exploits a single core so far. Moreover, a registration pipeline based on range images is inherently faster than one working on points clouds, due to the former deploying the lattice provided by the image to find the neighbours of a feature, the latter requiring a slower kd-tree search. Thus, the efficiency of our pipeline, which is comparable to that of BLS12, is due to the purposely devised feature extraction and inexpensive feature matching approaches.

As anticipated in Section 3.1, although in a different domain, the idea of matching 3D points based on a scalar cue derived from LRF computation can be found also in [71]. Thus, to highlight how the adopted repeatable LRF [77] and proposed matching cue are crucial ingredients in the design of our pairwise registration pipeline, we devised an experiment aimed at plugging into our method the LRF computation (curvature tensor) and matching cue (ratio of principal curvatures) proposed in [71], so to then comparatively ascertain performance (curvature tensors are obtained by means of the original implementation, kindly provided by the authors, of the algorithm used in [71] and proposed in [4]).

For the sake of fairness, instead of employing our detector of locally flat keypoints, we rely here on randomly extracted feature points and report the results for different numbers of extracted features. As for computation of the adopted LRF [77], for each dataset considered in the experiment we use the same support radius as in Table 4.3 and keep the matching threshold as in Table 4.1 ($T_D = 0.01$) independently of the number of extracted features. Instead, for the method proposed in [71], we specifically tuned the support radius and matching threshold for each number of extracted feature points so as to report here the best results. The N° *Registrations* performance indexes on three datasets are shown in Table 4.4. The results highlight clearly that the choice of the proper cues to compute and match LRFs is key to determining the registration rates provided by our pipeline. Indeed, we argue that whereas our approach allows the computation of the LRF on wide supports, the estimation of the principal curvatures forces the use of small supports that, as such, can hardly capture distinctive shape information around a keypoint. In addition, features related to pointwise curvatures are known to be susceptible to sensor noise. Finally, in the matching stage, the D_{max} feature results to have more discriminative power than the k_1/k_2 score based on curvatures.

	MarioKinect	
	Mitra	LRF
500	6	20
1,000	9	26
1,500	9	27
2,000	8	25
3,000	7	27

	Buste	
	Mitra	LRF
500	18	55
1,000	23	60
1,500	23	60
2,000	23	59
3,000	23	64

	Angels	
	Mitra	LRF
500	0	7
1,000	1	10
1,500	1	9
2,000	3	11
3,000	3	12

TABLE 4.4: Comparison between our proposed pipeline (LRF) and the pipeline (Mitra) that would be achieved by computing and matching local reference frames according to [71]. The Table reports the N° *Registrations* on three datasets (*MarioKinect*, *Buste*, *Angels*) for different quantities of randomly extracted features.

Once all the pairwise rigid motions are estimated, it is possible to align the views in a unique reference frame. Based on the pairwise registrations provided by our pipeline, we exploit the framework of [6] to get a global, though coarse, reconstruction by determining a spanning tree where the edges join the view pairs that maximize the overlap area. We apply this process to two *Open Technologies*® datasets and two *Kinect* datasets. The results are depicted in Fig. 4.2, Fig. 4.3 and Fig. 4.4. Due to both the acquisition quality of the *Open Technologies*® datasets and the accuracy of our pipeline, the views come out finely aligned, even though no ICP-based fine registration is run downstream. Conversely, for *DuckKinect* and *FrogKinect*, we then also run the *Scanalyze* tool to get a global refinement of the registration and the *Poisson Reconstruction* algorithm [53] to obtain the final 3D models. Although *Kinect* acquisitions are noisy and inaccurate, the results are worthy.

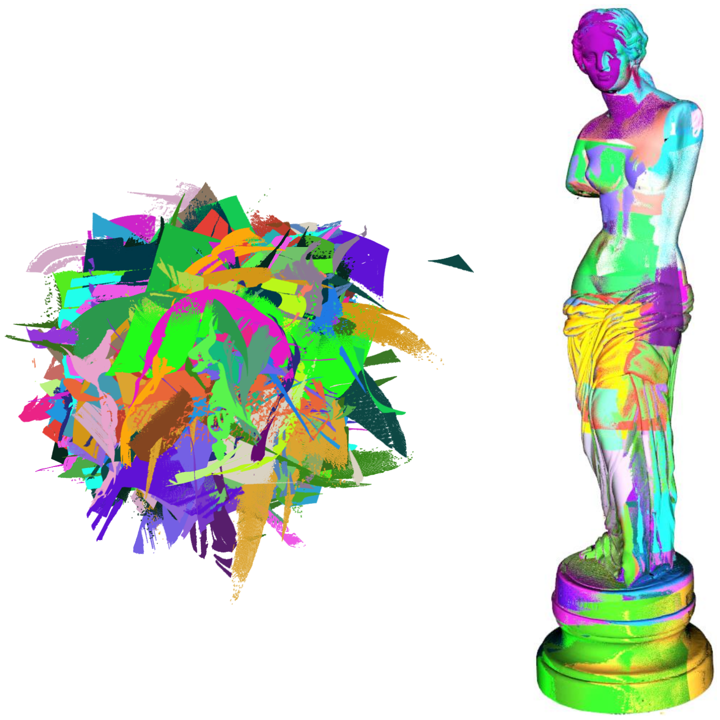


FIGURE 4.2: Registration of the *Venus de Milo* by alignment of 61 views and about 50 million points.

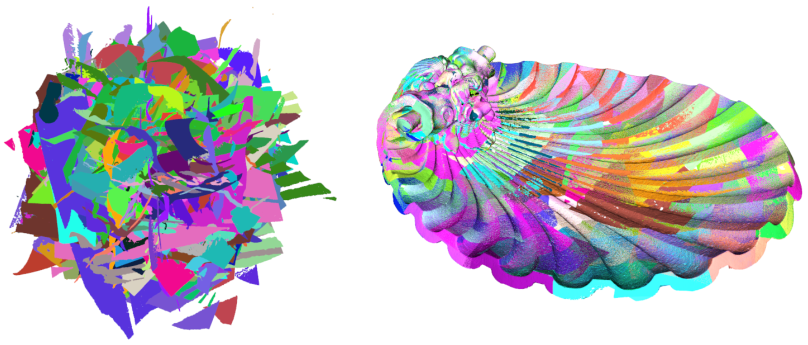


FIGURE 4.3: Registration of the *Shell* by alignment of 98 views and about 72 million points.

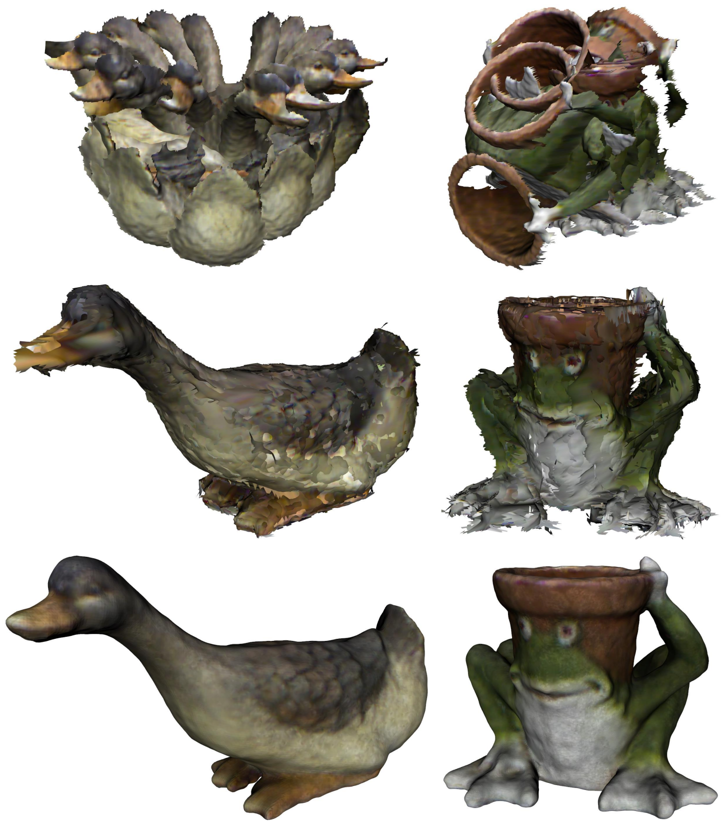


FIGURE 4.4: Reconstructions of *DuckKinect* and *FrogKinect*. Top: initial disarranged views. Center: coarse reconstructions provided by our pipeline. Bottom: final meshes attained by refining coarse reconstructions by *Scanalyze* and then running *Poisson Reconstruction*.

Chapter 5

RGB-D Mobile Visual Search

As discussed in the introduction, the foreseeable advent of depth sensing on mobile devices at a significant scale may pave the way to a new generation of mobile applications able to exploit 3D (i.e. shape) information. In particular, in this thesis, we are interested in investigating on whether and how Mobile Visual Search architectures may benefit of depth sensing capabilities. Accordingly, the next two chapters propose an investigation on how to encode both appearance and depth information to obtain compact binary codes that properly describe RGB-D images in Mobile Visual Search scenarios. More precisely, this chapter describes a suitable architecture and proposes different approaches for carrying out the stages involved in the processing flow. Next chapter reports a comprehensive experimental analysis aimed at establishing the best configuration of the pipeline and, in particular, at which level of the flow appearance and shape information should better be merged.

Although, to the best of our knowledge, this is the first work that proposes an RGB-D Visual Search engine amenable to mobile applications, research on object recognition over the past years has yielded a large body of work that leverages on RGB-D sensing [58, 12, 103, 9, 88, 72, 39]. Unfortunately, all these proposals rely on a computational flow unsuited to Mobile Visual Search. Indeed, they first encode the RGB and depth images into lengthy representations: e.g., in [88], Socher et al. stack a recursive neural network on a layer of CNN to build a feature vector of 32,000 elements, whereas in [12] the resulting descriptor is as large as 188,300. Then, this rich description feeds a classifier, such as a SVM or a Random Forest, that, depending of the task, recognizes either the object category or instance associated with image content. As already pointed out, though, in Mobile Visual Search scenarios compactness of description is at least as key as distinctiveness. Furthermore, a classifier is confined to recognition of learned classes only and would require a typically expensive training process to be able to work with new ones. Conversely, a Visual Search engine should feature far higher flexibility so to enable easy and fast updating of the database of images seamlessly handled by the application. Thus, a similarity search approach dealing with matching the query into a database of candidates images is more suited to Mobile Visual Search scenarios than a trained classifier, as also vouched by the reference architecture established within the *Compact Descriptors for Visual Search* (CDVS) [1] proposal, which is likely to become part of the MPEG-7 standard.

Accordingly, our architecture grounds on an such approach, as depicted in Fig. 5.1. Given an RGB and depth image pair, the pipeline extracts a set of local descriptors for representing both appearance and shape information. The subsequent stage aggregates the local descriptors into a global encoding of the whole image which is then compressed into a binary description through a similarity-preserving hashing stage. Then, the binary code is sent to the server where it is matched against a database of descriptions in order to find the most similar image. Besides such established paradigm dealing with aggregation of local features into a global representation, we also considered an approach based on deep neural networks that avoids the computation of hand-crafted features in favor of a learned global representation of the image (see Fig. 5.2).

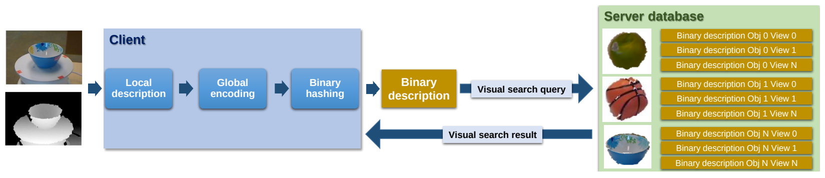

Client
Local description
Global encoding
Binary hashing
Binary description
Visual search query
Visual search result
Server database
Binary description Obj 0 View 0
Binary description Obj 0 View 1
Binary description Obj 0 View N
Binary description Obj 1 View 0
Binary description Obj 1 View 1
Binary description Obj 1 View N
Binary description Obj N View 0
Binary description Obj N View 1
Binary description Obj N View N

FIGURE 5.1: Outline of the proposed RGB-D Visual Search engine architecture.

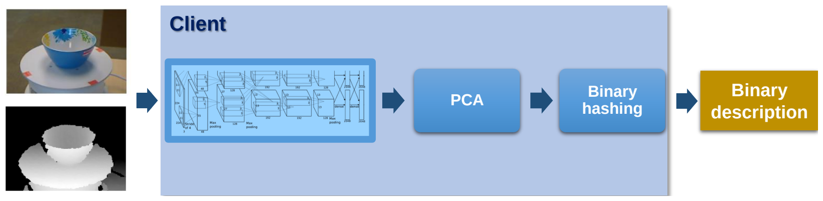


FIGURE 5.2: Processing flow of the client deploying deep neural networks.

5.1 Local description

5.1.1 SIFT

As a baseline local description approach we use SIFT[1] [65], which detects keypoints through DoG and produces descriptions of length $D = 128$. We apply SIFT on intensity images without any preprocessing, whereas depth images are rescaled in the range [1, 255], reserving the 0 value to denote invalid depths. To isolate depths belonging to the searched object, we modeled the distribution of depths of database images by a gaussian, then we linearly rescaled depths falling within $2 \times \sigma$ from the gaussian mean and saturated all the others.

5.1.2 Dense SIFT

To investigate on whether uniform sampling of features may turn out more beneficial than keypoint detection to Visual Search applications, we compute SIFT[1] descriptors on 16×16 patches sampled across a regular grid.

[1]SIFT and Dense SIFT features are computed by the OpenCV implementation.

5.1.3 Kernel Descriptors

Given the excellent results reported on a variety of RGB-D recognition tasks, we have considered the *RGB-D Kernel Descriptors* introduced in [11, 10]. Kernel descriptors are a generalization of descriptors based on orientation histograms, such as SIFT and HOG, which may suffer from quantization errors due to binning. Kernel descriptors overcome this issue by defining the similarity between two patches through kernel functions, referred to as *Match Kernels*, that average out across the continuous similarities between pairs of pixel attributes within the two patches. Local description is performed on patches sampled across a regular grid, with each patch represented by a 200-dimensional feature vector. The authors propose 8 types of kernel descriptors by defining match kernels for different patch attributes such as intensity and depth gradient, local binary patterns and object size. In our experiments we used the C++ implementation made available online by the authors, which permits to apply 4 types of Kernel Descriptors[2]. In particular, appearance information is described by kernels dealing with *Intensity Gradients* and *Color*, while shape information is captured by kernels based on *Depth Gradients* and *Spin Images*.

5.2 Global encoding

We compared the following three state-of-the-art methods for the global encoding of the local descriptors extracted from the image.

5.2.1 VLAD

The method, introduced in [47], learns at training time a set of N_C visual words via k-means clustering in the space of the local descriptors

[2] `http://www.cs.washington.edu/robotics/projects/kdes/`

extracted from the training database. Let $X = \{x_t, t = 1...T\}$ be a set of local descriptors extracted from a test image at encoding time. For each local description x_t, of length D, the nearest visual word c_i is found and the vector $x_t - c_i$ is computed and associated to c_i. Therefore, for each visual word c_i, the associated vectors $x_t - c_i$ are summed to form the vector e_i. Finally, all the e_i are juxtaposed to form the global $D \times N_C$ dimensional representation of the image.

5.2.2 Fisher kernel

In [46], Jaakkola and Haussler introduced *Fisher kernels* with the aim to combine the power of discriminative classifiers with the ability of generative models to handle representations comprising a variable number of measurement samples. The encoding vector is the gradient of the samples log-likelihood with respect to the parameters λ of the generative model u_λ:

$$G_\lambda^X = \frac{1}{T} \nabla_\lambda \log u_\lambda(X) \tag{5.1}$$

and, intuitively, it can be seen as the contribution of the parameters to the generation of the samples.

Perronnin et al. in [75] applied Fisher kernels to image classification by modeling visual vocabularies with *Gaussian mixture models* (GMM)[3]:

$$u_\lambda(x_d) = \sum_{i=1}^{N_G} \alpha_i u_i(x_d) \tag{5.2}$$

denoting $\lambda = \{\alpha_i, \mu_i, \Sigma_i, i = 1...N_G\}$, where α_i, μ_i and σ_i are, respectively, the mixture weight, mean vector and covariance matrix (assumed diagonal) of gaussian u_i.

[3]We use the Fisher Kernel and VLAD implementations available in the VLFeat libray (`http://www.vlfeat.org/`).

By assuming local descriptors x_d as generated independently by u_λ, the encoding vector can be computed as:

$$G_\lambda^X = \frac{1}{D} \sum_{t=1}^{T} \nabla_\lambda \log u_\lambda(x_t) \tag{5.3}$$

Let $\gamma_t(i)$ be the soft assignment of descriptor x_t to Gaussian i:

$$\gamma_t(i) = \frac{\alpha_i g_i(x_t)}{\sum\limits_{j=1}^{N_G} \alpha_j g_j(x_t)} \tag{5.4}$$

the D-dimensional gradients $G_{\mu,i}^X$ and $G_{\sigma,i}^X$ with respect to the mean μ_i and standard deviation σ_i can be derived as:

$$G_{\mu,i}^X = \frac{1}{T\sqrt{\alpha_i}} \sum_{t=1}^{T} \gamma_t(i) \left(\frac{x_i - \mu_i}{\sigma_i} \right) \tag{5.5}$$

$$G_{\sigma,i}^X = \frac{1}{T\sqrt{2\alpha_i}} \sum_{t=1}^{T} \gamma_t(i) \left[\frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1 \right] \tag{5.6}$$

The final encoding is the concatenation of the $G_{\mu,i}^X$ and $G_{\sigma,i}^X$ vectors for $i = 1...N_G$ and, hence, has length $2 \times D \times N_G$.

5.2.3 Efficient match kernels

Similarly to kernel descriptors exploiting match kernels to overcome the potential loss of descriptiveness due to binning in orientation histogram descriptors, *Efficient Match Kernels* (EMK)[4] [13] generalize the bag-of-words aggregation scheme to counteract binning errors. This method is specifically designed to aggregate local kernel descriptors into image-level representations. Accordingly, the implementation made available

[4]http://research.cs.washington.edu/istc/lfb/software/EMK.htm

by the authors allows to encode kernel descriptors only and does not permit training on different kinds of local representation. Unlike VLAD and Fisher kernel, EMK takes into account spatial information by performing the encoding using a spatial pyramid, as proposed by Lazebnik et al. in [59]. The image is subdivided in 1×1, 2×2 and 4×4 subregions on three level of abstraction and each of them is separately encoded[5]. The final description consists in the concatenation of all the encodings. For local descriptions based on appearance information, the single encoding is 500-dimensional, hence the image is represented with a $(1 + 4 + 16) \times 500 = 10500$ lengthy descriptor, whereas by encoding of shape information the description reaches a length of 14000 as the single encoding is 1000-dimensional.

5.2.4 Deep Features

In [39], Gupta et al. address the problem of globally encoding an RGB-D image through a Convolutional Neural Network (CNN) architecture. Purposely, they exploit the so called "AlexNet" proposed in [56], that processes a 256×256 RGB image and can produce a 4096-dimensional feature vector as output of the last hidden layer. Besides describing the RGB image, the authors of [39] deploy the HHA representation to map the depth image into three channels: *H*orizontal disparity, *H*eight above ground and *A*ngle between local surface normal and inferred gravity direction. Accordingly, AlexNet is also fed with the HHA representation as if it were an RGB image. The authors ground this approach on the hypothesis that RGB and depth images share common structures due to, for example, disparity edges corresponding to object boundaries in RGB images. Moreover, the authors perform fine tuning of AlexNet based on HHA data. In our pipeline, the 4096-dimensional RGB and HHA features do not directly feed the hashing stage, but they are first

[5]In the case of shape-based kernel descriptors, the third level of the pyramid is divided in 3×3 subregions.

reduced in dimensionality through *Principal component analysis* (PCA) (see Fig. 5.2).

5.3 Binary hashing

As the global descriptions obtained so far would typically require excessive bandwidth should they be sent to the server directly, the addressed Mobile Visual Search scenario calls for further compression. Thus, we perform a similarity-preserving hashing stage aimed at producing the final compact and binary description sent to the server. Among the several hashing methods proposed in the last years, we considered the baseline approach referred to as *Locality Sensitive Hashing* (LSH) [45] and the state-of-the-art *Spherical Hashing* (SH) method [40], which has been reported to turn out peculiarly effective on large datasets.

5.3.1 Locality Sensitive Hashing

Let N_b be the number of bits comprising the binary description, *Locality sensitive hashing* (LSH) defines the hashing functions simply by creating, at training time, a set of N_b random hyperplanes in the description space. Then, to perform the hashing of a new descriptor, each bit of the binary code is labeled as 1 if the description is situated in the positive half-space of the associated hyperplane, 0 otherwise.

5.3.2 Spherical Hashing

Spherical Hashing (SH) represents the data with a set of N_b hyperspheres and choose the value of the $i - th$ bit depending on whether the description is inside or outside the $i - th$ hypersphere. To determine the centers and radii of the hyperspheres, and iterative optimization process is performed so to achieve balanced partitioning of descriptions for each hash function as well as independence between any two hashing functions. Furthermore, the authors of SH propose a new distance in the Hamming space better suited to their coding scheme, namely the *Spherical Hamming Distance*, that normalizes the standard Hamming distance by the number of corresponding bits equal to 1 between the two strings. Corresponding bits set to 1 denotes that the two descriptions are inside the same hypersphere and therefore an higher likelihood that the two points are close in the feature space.

5.4 Matching

As far as the server side of our engine is concerned, in the current experimental embodiment, binary descriptions are computed for each image representing the objects populating the database and a similarity index is built by means of the *multi-probe LSH* scheme (mpLSH) proposed in [66]. Given a query image, the binary code received from the client is matched against the database applying the weighted k-NN search introduced in [29].

5.5 Fusion of appearance and shape

Although several works in literature show how the fusion of RGB and depth channels can improve the recognition ability of the proposed frameworks, few of them discuss at which level of the pipeline the fusion ought better occur. In [103], the authors suggest two different versions of their pipeline, the former fusing RGB and depth at the local description level, the latter at an higher level. Interested in investigating on this matter, we considered three different fusion approaches (as outlined in Fig. 5.3) and performed a comparison. The first approach, referred to as *Local Fusion*, computes the local description of both appearance and shape for each patch extracted from the image and then juxtaposes them.

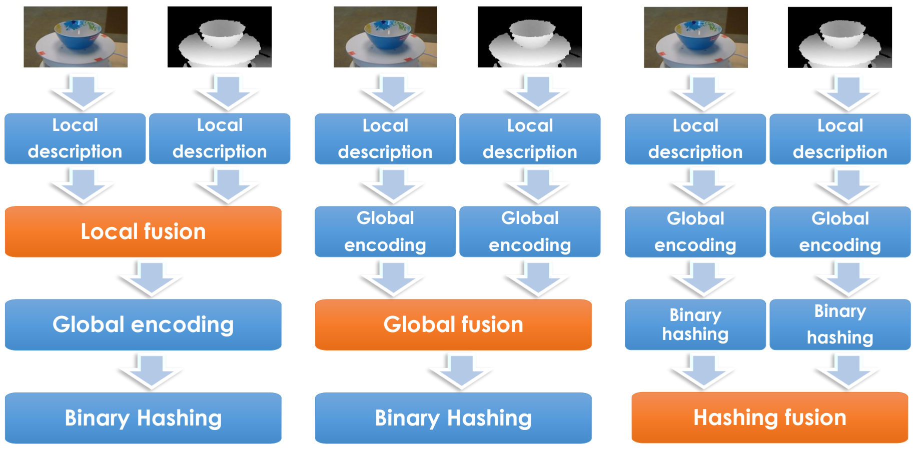


FIGURE 5.3: The three strategies for fusion of the appearance and shape information associated with RGB-D images considered in our experimental investigation.

VLAD and Fisher Kernel are therefore trained on the concatenation of the appearance and shape descriptions of image patches[6]. With *Global Fusion*, global descriptors are computed for appearance and shape separately and then concatenated before being delivered to the hashing stage. Finally, with *Hashing Fusion*, binary codes for RGB and depth images are computed independently and eventually juxtaposed just before the matching stage.

[6]As explained in the previous section, the available EMK implementation does not permit training on new data. Accordingly, EMK cannot be applied with *Local* fusion.

Chapter 6

RGB-D Mobile Visual Search: Experimental Evaluation

This chapter discusses the experimental evaluation of the Visual Search architecture introduced in the previous chapter. After an overview of the datasets considered in the evaluation, the chapter summarizes the key findings resulting from an extensive experimental investigation we performed to identify the best configuration of the architecture as well as to tune properly the main key parameters. Then, the engine resulting from such analysis has been compared with the *Compact Descriptors for Visual Search* (CDVS), a state-of-the-art RGB-based architecture designed for Mobile Visual Search.

6.1 Datasets

The experiments reported in this chapter have been performed on 3 state-of-the-art datasets of household objects: the *RGB-D Object Dataset*, *CIN 2D+3D* and *BigBIRD*. The former two datasets share a two-level category/instance structure that allows us to evaluate our framework on both category and instance recognition tasks, whereas BigBIRD consists of object instances not partitioned into categories.

6.1.1 RGB-D Object Dataset

The *RGB-D Object Dataset* [58] is nowadays the de-facto standard for evaluating and comparing visual recognition systems relying on RGB-D sensing. For each of the 300 household objects composing the dataset, a set of acquisitions from different vantage points has been collected and segmented from the background so as to gather a total of 41,877 RGB-D images (see Fig. 6.1, top). Each object belongs to one of 51 categories based on the WordNet hierarchy. This two-level organization allow researchers to evaluate their proposals both on category and instance recognition scenarios (see Fig. 6.1, bottom). As for instance recognition, we chose the *Alternating Contiguous Frames* methodology and, as suggested in [58], average results over 10 randomly defined trials.

6.1.2 CIN 2D+3D

The *CIN 2D+3D* dataset [16] shares with the RGB-D Object Dataset a similar two-level category/instance structure. Indeed, it consists of 18 categories, which in turn include about 10 instances each (see Fig. 6.2). The objects, placed on a turntable, have been acquired from 36 vantage points by rotating the turntable by $10°$ upon each acquisition. In [16], the authors propose a procedure aimed at evaluating simultaneously the ability to recognize both instances and categories. Conversely, similarly to the RGB-D Object Dataset methodology, we prefer to separately test the performance for the two tasks of category and instance recognition. Thus, for category recognition, we select a tenth of the instances for each category as test set and train the pipeline on the remaining ones. We repeat the procedure 10 times by choosing each time different instances for the test set and averaging the recognition rates thus obtained. Likewise, we perform 10 trials for instance recognition. Each trial splits a different tenth of the views of each instance and uses it as test set whereas the remaining acquisitions are used for training. As suggested

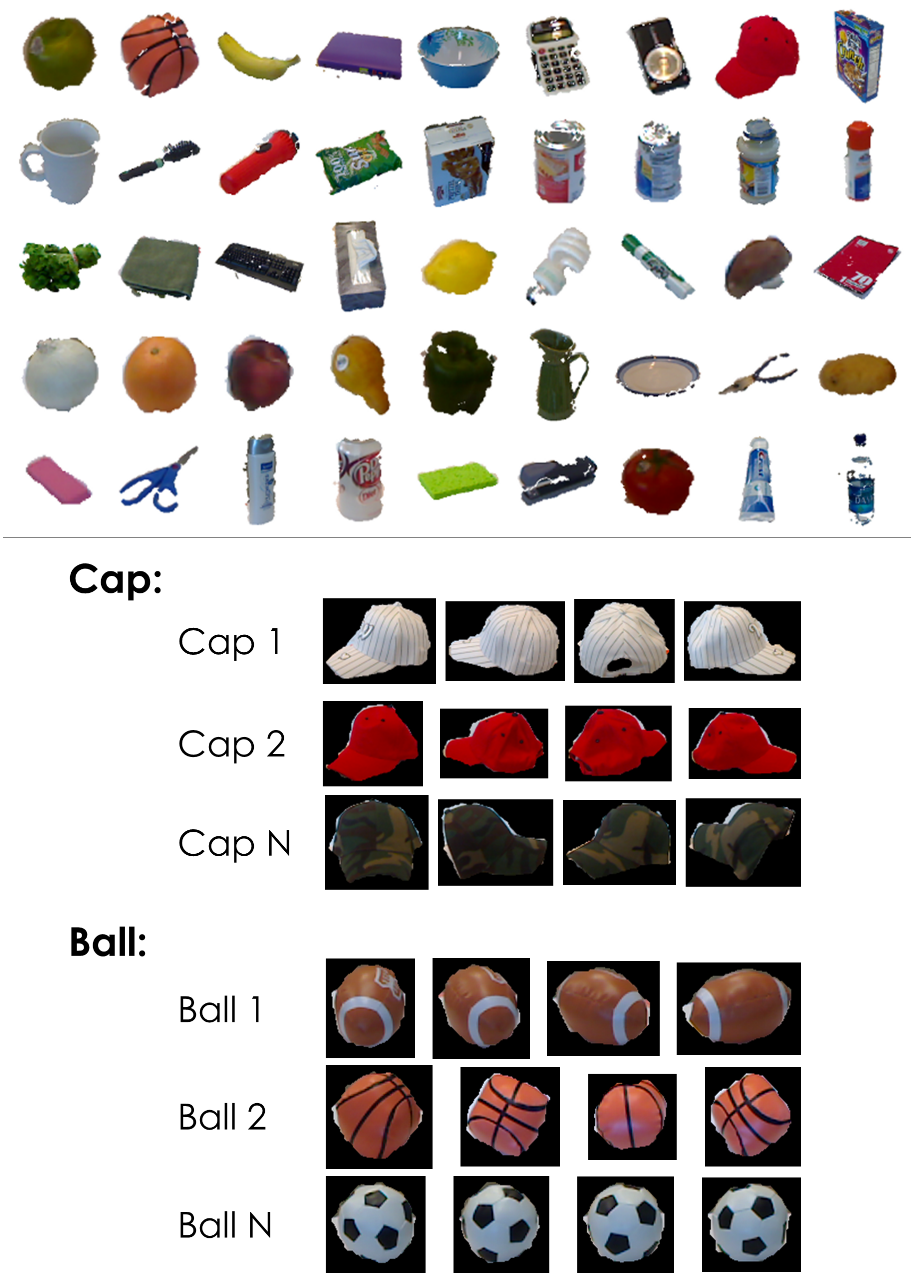


FIGURE 6.1: RGB-D Object dataset. Top: examples of objects composing the dataset. Bottom: category/instance organization of the dataset. Each category comprises a set of different instances which have been acquired from different vantage points.

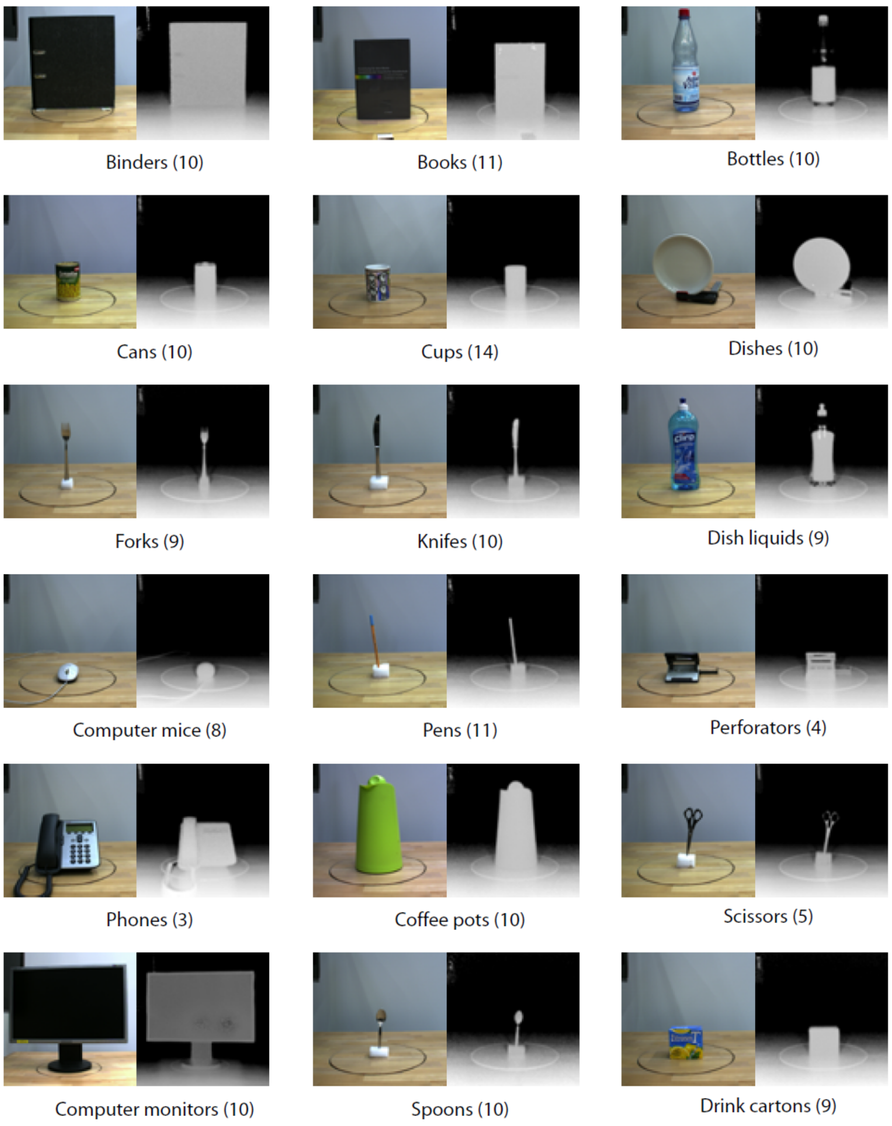


FIGURE 6.2: Objects composing the CIN 2D+3D dataset. The number of instances for each category is reported in brackets.

by the authors, we discard the "Perforator" and "Phone" categories from the evaluation as they do not include a sufficient number of instances. Instead, we do not aggregate "Fork", "Spoon" and "Knife" into a "Silverware" super-category.

6.1.3 BigBIRD

The *BigBIRD* dataset [87] comprises 125 object instances not partitioned into categories (see Fig. 6.3, top). The authors staged a setup, shown on the bottom of Fig. 6.3, that allows acquisition of an object from 5 different vantage points via 5 PrimeSense Carmine sensors uniformly disposed along an arc, so that the first sensor acquires the front of the object whereas the last sees it from above. Moreover, the object is placed on a turntable and acquired every 3°, making a total of 600 views for each object. The depth maps have been taken at a resolution of 640×480 pixels, whereas RGB images feature a resolution of 1280×1024 pixels. Mask images that allow segmenting out object pixels in the RGB images are also provided within the dataset. Objects are mainly supermarket products and include quite challenging instances as most of them are boxes recognizable only by their packages, which sometimes are very similar (e.g. as in the case of "chewy dipps chocolate chip" and "chewy dipps peanut butter" shown in Fig. 6.4) or distinguishable just by colour. Other products are bottles with very similar shapes which may be told apart from above based on the can only. Unfortunately, sensors are too distant from objects to properly simulate a typical Visual Search scenario wherein the object to be recognized would typically cover most of the query image. Therefore, by exploiting the available RGB masks, we cropped each RGB and depth image of the dataset so to get the object closed-up. First, by means of calibration information, we projected each valid point of the depth map onto the mask image so to obtain the mask associated with the depth map. Then, we computed the centroid and standard deviations σ_x, σ_y of foreground pixels. Therefore, we defined a bounding box centered at the centroid

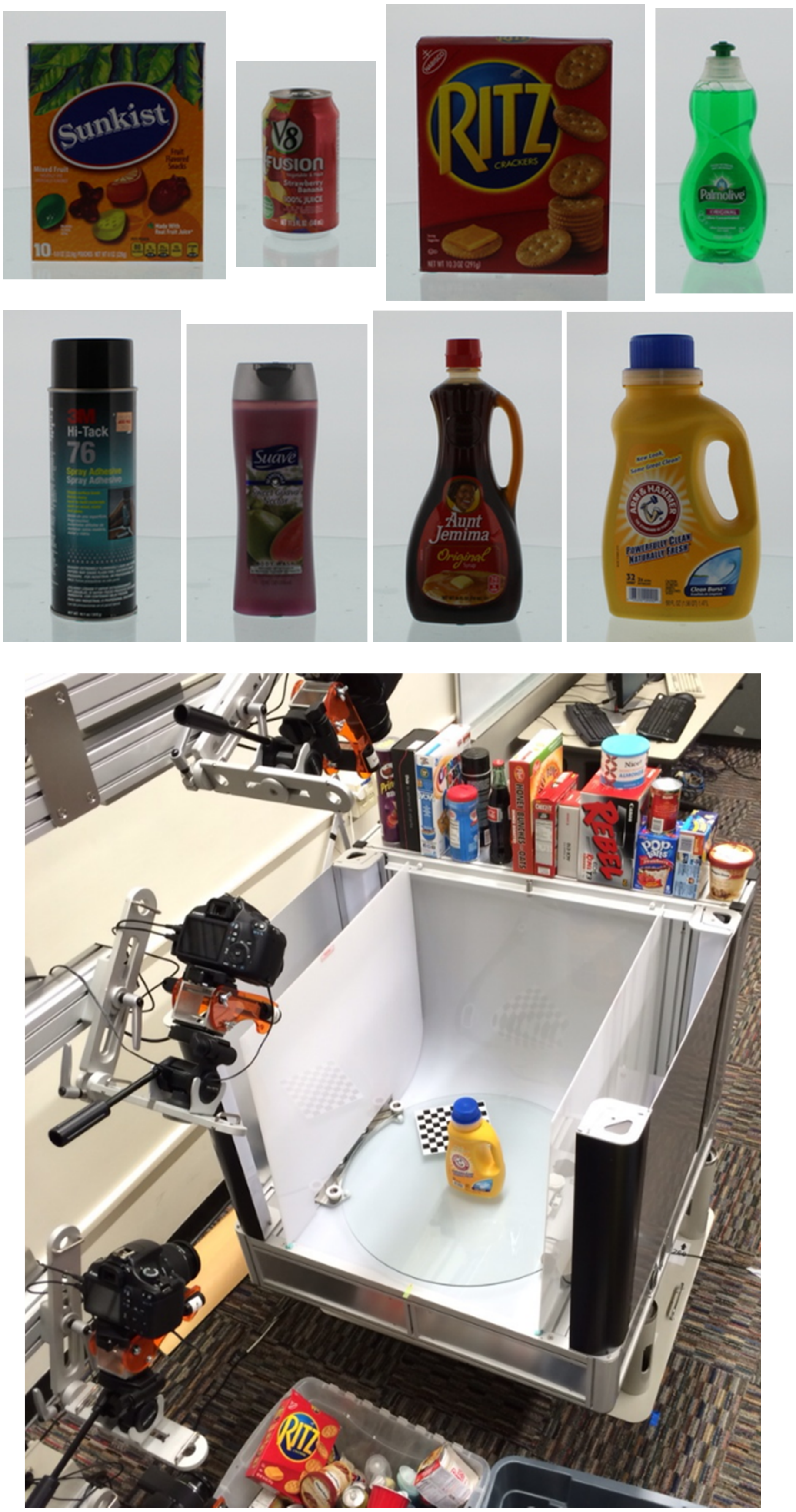


FIGURE 6.3: BigBIRD dataset. Top: examples of objects included in the dataset. Bottom: setup staged to acquire the dataset.

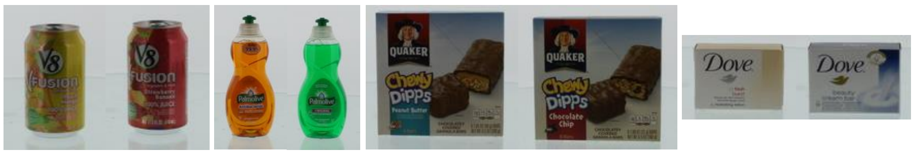


FIGURE 6.4: Examples of BigBIRD objects distinguishable by colour and texture only.

and wide $4 \times \sigma_x$ and $4 \times \sigma_y$ along x and y dimension, so to consider about 95% of the pixels of the object. The cropping bounding box is obtained by enlarging both dimension by a factor of 1.5. The process is repeated to perform the cropping of the RGB image which is finally rescaled to get the same dimensions as the cropped depth map. As reliable segmentation masks are not provided for 11 objects (the majority of them being transparent bottles), we discarded them from the data used in our experiments. The authors do not suggest a methodology to evaluate object recognition algorithms using the BigBIRD dataset; thus, for each of 10 trials, we randomly select 100 acquisitions and split them so as to perform testing on a tenth of them and training based on the others.

6.2 Experimental investigation

This section describes the experimental analysis we performed in order to determine the best configuration of each stage of the pipeline. As an exhaustive exploration of the parameter space would turn out infeasible, we performed a stage-by-stage tuning starting from the matching and backing up to the local description: once the proper configuration for a stage has been determined, the tuned parameters are kept fixed and adopted in the analysis of the upstream stage. For the tuning of the stages following the local description, we report only the investigation performed on the *RGB-D Object Dataset*, as probing experiments on the other datasets turn out consistent in terms of obtained results. Moreover, we adopted the kernel descriptors as specifically designed to encode both appearance and depth. Also, they proved to achieve competitive results on different recognition tasks. In this phase of the investigation we are not concerned with peak performance of the search engine but, rather, in comparing its diverse possible configurations. Therefore, and for the sake of time alike, between all kernel descriptors, we considered here only the *gradient match kernel* applied on image intensities, so to represent appearance information, and on depths, so to capture shape as well. Once the best configuration of such stages had been determined, we assessed the performance of the other local descriptors and compare them with deep features.

For each experiment run, we use the training set to perform k-means and GMM estimation, as required by VLAD and Fisher Kernel respectively, as well as to train LSH and SH. After that, we described each image of the training set with the trained pipeline and built the index used by the matching stage. Finally, we described all the test images and calculated the rate of them correctly recognized in the training set. This procedure is repeated for each of the 10 trials splitting differently the training and test sets. Eventually, the attained recognition rates are averaged. For each configuration, we run the pipeline while varying the length of the final binary code from 16 to 1024 bits and plot the attained mean recognition rates as a function of the code length.

6.2.1 Matching

The first experiments addressed the matching stage. The analysis, performed on different configurations of the client pipeline, revealed no loss in recognition rate and about a 10$\times$ speedup in applying the approximated mpLSH indexing scheme in place of an exhaustive search. In addition, we experimented with the *Spherical Hamming Distance* [40] as an alternative to the standard *Hamming Distance*, but did not perceive any improvement in the results. Moreover we tuned the weighted k-NN search so to set k=*9*.

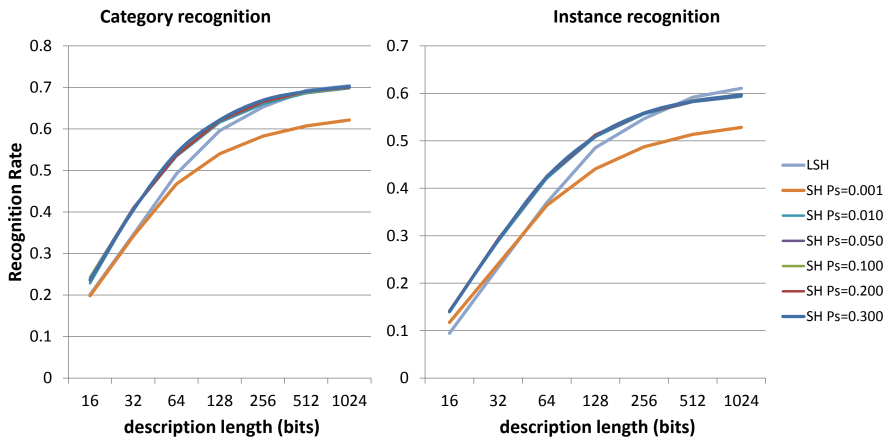


FIGURE 6.5: Comparison between Locality sensitive hashing (LSH) and Spherical hashing (SH).

6.2.2 Binary hashing

Then, to delineate about the *Binary Hashing* stage, we carried out experiments dealing with the different fusion strategies and global encoders. All the tests coherently indicated the trend that we report in Fig. 6.5 for the specific case of global encoding by EMK and *Hashing Fusion* strategy, with recognition rates plotted as a function of the length of the binary codes, N_b. In the iterative process used to construct the Spherical Hashing functions only a percentage P_S of all database descriptors are considered, but the authors do not provide hints or guidelines regarding the choice of this parameter [40]. Hence, as shown in Fig. 6.5, we experimented with different P_S values. The plots show that, for both the category and instance recognition tasks, $P_S = 0.01$ is sufficient to guarantee adequate training of SH and that SH turns out more effective than LSH with very compact codes while the two hashing approaches may be considered equivalent with longer codes. Thus, in all the subsequent experiments discussed in this paper we make use of SH with $P_S = 0.01$. Nonetheless, it is worth pointing out that our findings seem not to confirm the results reported in [40], wherein SH, especially when exploiting the *Spherical Hamming Distance*, is reported to outperform

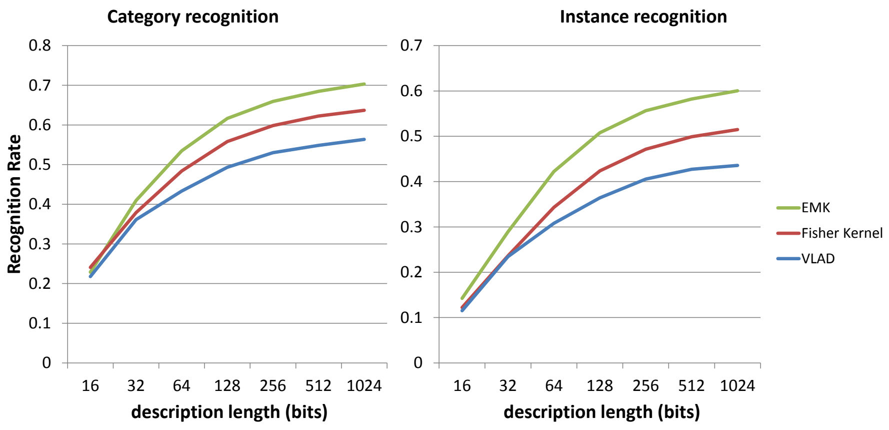


FIGURE 6.6: EMK vs. Fisher Kernel vs. VLAD.

LSH neatly. This inconsistency may likely be ascribed to the different experimental settings: besides the diverse evaluation protocols and adopted figure of merits, in [40], the experiments addressed very large datasets, such as *Tiny Images*, comprising millions of images whereas the standard *RGB-D Object Dataset* consists of thousands of RGB-D images.

6.2.3 Global encoding

As next step, we investigated on the three methods considered for the *Global Encoding* stage of our proposed architecture. Purposely, we performed the tuning of the number of visual words, N_C, learned by k-means in VLAD training and the number of gaussians, N_G, of the GMM used by Fisher Kernel. Accordingly, the results shown in Fig. 6.6 have been obtained with $N_C = 1$, $N_G = 2$. The charts highlight that EMK overtakes the other methods, followed by Fisher Kernel. Nonetheless, it is worth pointing out that EMK incorporates spatial information by aggregating local features in pyramidal subregions and juxtaposing the encodings of each subregion, whereas Fisher Kernel, and VLAD alike, disregards the information concerning the position of local features. The non-invariant global representation produced by EMK may

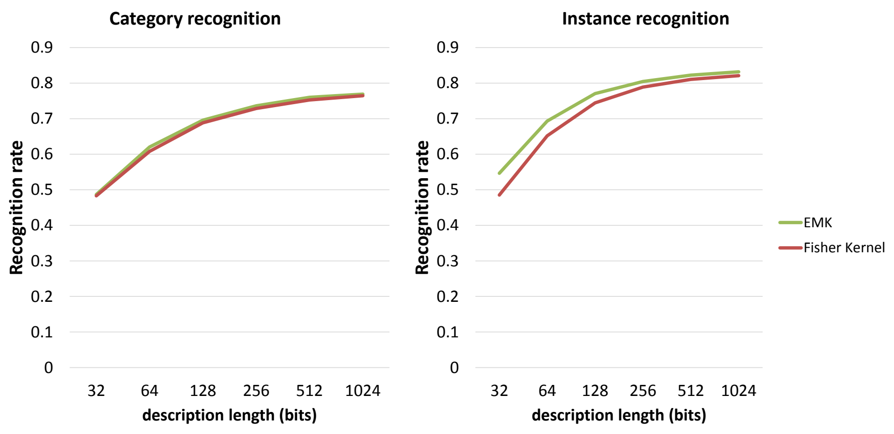
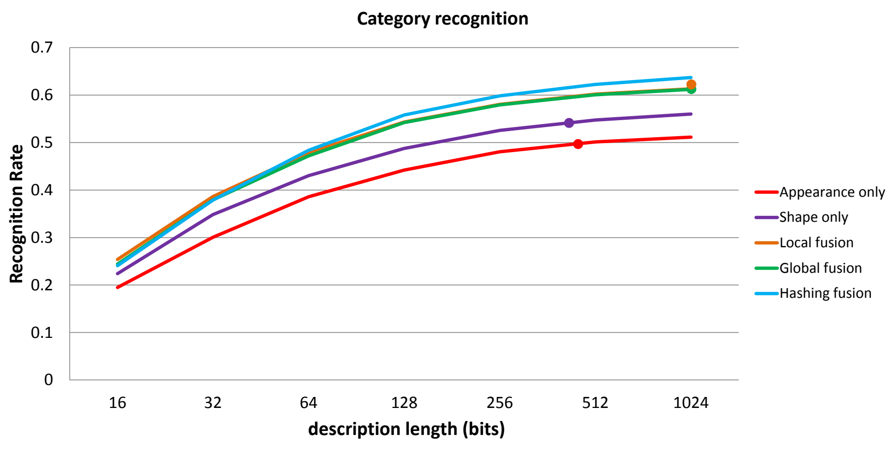


FIGURE 6.7: EMK vs. Fisher Kernel on encoding all four types of Kernel Descriptors.

thus turn out more effective as long as the images are acquired in a controlled setup, as it is indeed the case of the *RGB-D Object Dataset* where objects had been placed upright on a turntable and always acquired from a fixed distance to the sensor. On the other hand, we expect Fisher Kernel to yield superior results than EMK in scenarios involving in-plane rotations and distance variations of the RGB-D sensor while capturing the objects. To better assess which one of the two encoders is more suited for our architecture, we repeated the experiment but considering also the other two types of Kernel Descriptor so as to aggregate a richer description. The comparison is reported in Fig. 6.7. The curves, coming out now approximately overlapped, point out that the differences between EMK and Fisher kernel tend to disappear as the descriptive power of the pipeline increases. In the light of this result, Fisher Kernel emerges as the proper global encoder for the pipeline.

6.2.4 Contribution of appearance and shape

Having identified the optimal configuration and parameter tuning for each stage, we investigated on how the shape and appearance information separately contributes to the recognition capability of our engine as well as on how their synergy improves the overall performance, especially as the fusion strategy changes. The curves in Fig. 6.8 report the recognition rates obtained by exploiting RGB information only, depth information only as well as by fusing the two kinds of information according to the three considered strategies. We run the experiments for both instance and category recognition using Fisher Kernel as global encoder. The charts allow us to appraise separately the impact, for the two different tasks, of the RGB and depth features. In case of instance recognition, the appearance information provides the stronger contribution to the recognition ability whereas shape proves to have a more limited descriptive power. Conversely, in the category recognition task, shape features are more effective in discriminating the categories comprising the dataset. These results are in line with the findings reported in [10] and vouch as a specific instance of an object is better characterized by textures and colors whereas shape is the primary trait that allows to determine the category to which an object does belong. The figure also suggests that synergistic deployment of appearance and shape can improve the recognition rate in both tasks and by means of all the three considered fusion strategies. As for a comparison between these, the *Hashing Fusion* approach appears the best choice. That is somehow coherent with the results reported in [103] and seems to suggest that maintaining the processing of the two information flows as disjoint as possible along a recognition pipeline may turn out the most effective strategy. Furthermore, from a more practical viewpoint, *Hashing Fusion* would allow the server to match the two descriptions separately. This is particularly worthwhile in case in the addressed application scenario only one of the two descriptions may be available in either the query image or some database objects. Furthermore, having the chance to match them separately might turn out a favorable trait if under some circumstance acquisition of either appearance or shape may not be considered reliable enough. This may happen, for example, when acquiring a transparent object, so that shape features are likely faulty or, as regards appearance cues, had the RGB image been captured under very



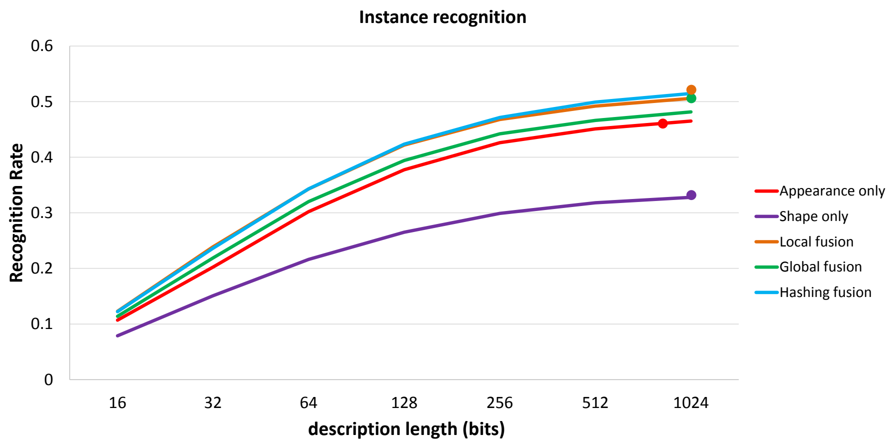


FIGURE 6.8: Performance when deploying the RGB channels (Appearance only), the depth channel (Shape only) or fusing the two kinds of information according to the *Local*, *Global* and *Hashing* methods. The performance attainable without binary compression are shown by circular dots.

low lighting conditions.

6.2.5 Impact of binary compression

We continue the analysis by studying the impact of binary compression on Visual Search performance. Purposely, we ran again the experiments reported in Fig. 6.8 by removing the final hashing stage and matching the descriptions provided by the global encoding based on the Euclidean

Distance. Thus, for each curve in Fig. 6.8 (but the two relative to Hashing Fusion) we also show by a circular dot the recognition rate obtained without performing binary compression. The important finding here is that there is no significant performance decay if an adequate description length is deployed within the hashing process, such adequate lengths being rather small. Indeed, the comparison between the recognition rates obtained by matching 1024 bits codes and uncompressed global encodings highlights a slight drop in performance only in case of *Global* and *Local Fusion* for the instance recognition task, whereas, under certain conditions, binary compression may even improve the recognition rate, as vouched by the curves dealing with category recognition for the *Appearance only* and *Shape only* pipelines.

6.2.6 Image features

This section analyzes the different image features considered in the previous chapter. Unlike the experiments reported so far, we carry out the comparison on all the three available datasets so as to better uncover the relative merits and limits of the methods. For local approaches, we deployed the configuration of the pipeline resulting from the tuning reported in previous sections: global encoding performed through Fisher kernel and SH for the computation of the binary codes that are juxtaposed according to the hashing fusion strategy. In the case of SIFT features, we only performed the tuning of the N_G components of the mixture, and left all the other parameters unaltered. According to our experiments, the best results can be obtained with a number of components as small as $N_G = 3$ if the DoG detector is applied, N_G = 1 for the dense description. In the case of kernel descriptors, both for appearance and shape description, we compute the two available kernel descriptors, and, eventually, perform the hashing separately followed by the juxtaposition of the resulting binary codes. For deep features, we deploy the Hashing fusion strategy and keep the same tuning both for spherical hashing and the matching stage. Moreover, our experiments indicate that the best results are achieved by feeding the hashing stage with the 100 principal components of the 4096-dimensional vectors computed by both the RGB and HHA networks. To compare the different types of features, we execute the pipeline by considering either the appearance information extracted from the RGB image only or the shape information extracted from the depth image only or fusing the two kinds of information by concatenating their binary codes.

The results of the experimental analysis are reported in Fig 6.9. Firstly, the charts reveal that encoding based on SIFT keypoints (the green curves in the figure) is not effective within our Visual Search architecture as it provides the lowest recognition rates in all but the experiment dealing with appearance-only description on BigBIRD. Better results are scored by methods leveraging on densely computed local descriptors. Indeed, if SIFT is applied to patches extracted across a regular grid, the recognition rate raises substantially (red plots), especially in category recognition experiments (first 2 rows of the figure). Overall, the best performance are provided by representations based on Kernel Descriptors and Deep Features. Accordingly, in the remainder of the discussion we will mostly focus on these two approaches.

We start by commenting the behavior of representations based on appearance information only (first column of Fig 6.9) and address the impact of the two types of Kernel Descriptors first. The charts report the recognition rates yielded by Kernel Descriptors based on either intensity gradients or color as orange and cyan curves respectively, whereas purple curves deal with the performance attained assigning half of the binary code to the former and half to the latter. In category recognition experiments, both Kernel Descriptors contribute significantly to the recognition ability of the pipeline, so that their synergistic deployment ends up in improving the recognition rate, as perceivable more clearly in the case of the RGB-D Object dataset. On the other hand, in case of instance recognition experiments, color seems the main cue that allows for telling apart objects in the considered datasets. This is particularly

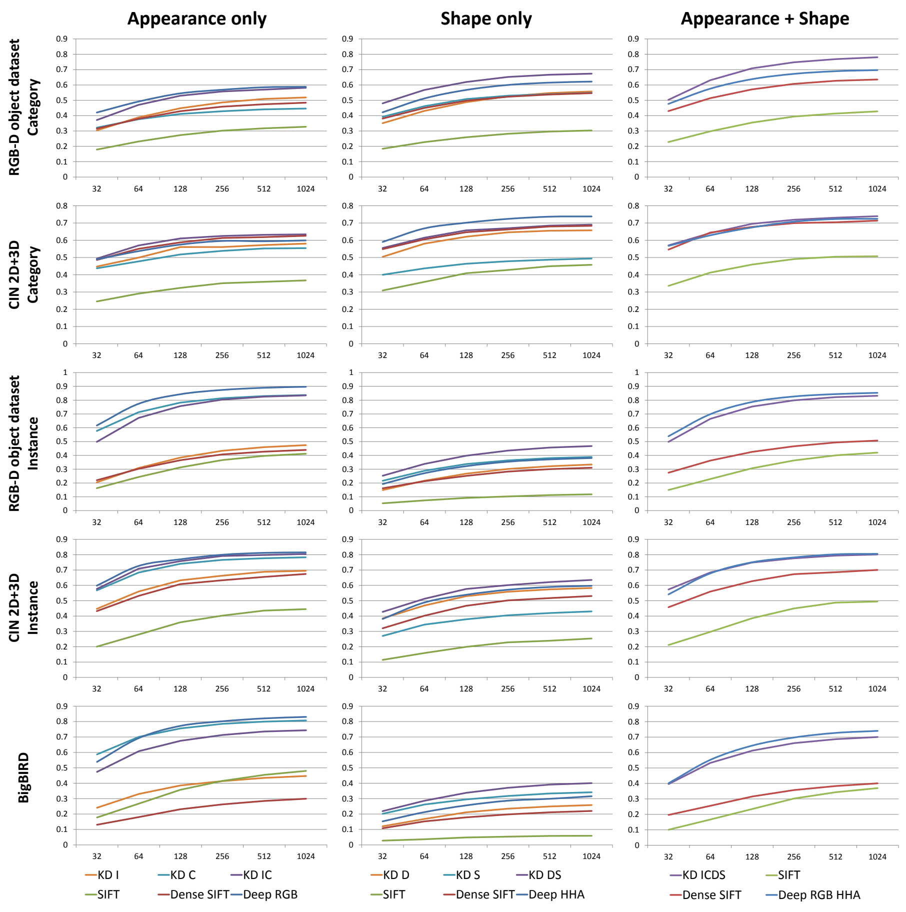


FIGURE 6.9: The charts are organized as a table, the rows dealing with the different datasets and recognition tasks (first 2 rows: category recognition, last 3 rows: instance recognition) and the 3 columns reporting, respectively, the results obtained with appearance-based descriptions only, shape-based descriptions only and fusion of appearance and shape. Each chart reports the recognition rate as a function of the length in bits of the binary code. The different curves are identified by the legend underneath columns. Kernel Descriptors (KD) based on Intensity gradients, Color, Depth gradients and Spin Images are labeled as I, C, D and S respectively.

noticeable in the BigBIRD dataset, as deploying half of the binary code to represent intensity gradients turns out even detrimental with respect to spending all bits to encode color. This can be ascribed to the nature of the dataset that, as already pointed out, consists mainly of boxes and bottles distinguishable by color features only (a few examples are shown in Fig. 6.4). The comparison between Kernel Descriptors and Deep Features (the blue plots in the charts) highlights how, with the exception of category recognition on the CIN 2D+3D dataset, the latter approach provides quite consistently higher recognition rates.

As for the experiments addressing representation of shape information only (second column of Fig 6.9), it is unclear which Kernel Descriptor allows for encoding more effectively the depth channel between that relying on Depth Gradients and on Spin Image, which represented by the orange and cyan curves respectively. Nonetheless, it is clear that fusing the two contributions by splitting the code bits evenly (purple curve) does increase the recognition rates insomuch as to outperform Deep Features in 4 out of the 5 experiments. This vouches as the two types of kernel descriptors are complementary and thus the recognition ability of the pipeline can benefit significantly of their synergistic deployment.

Looking now at the first two columns, it seems quite evident how shape is more relevant than appearance in category recognition experiments, the opposite being the case of instance recognition, as appearance turns out definitely the primary cue to tell apart the different objects comprising the considered datasets.

The third column of charts in Fig 6.9 reports the recognition rates attained by exploiting jointly the appearance and shape information provided by RGB-D images. In the task of category recognition (first 2 rows), Kernel Descriptors (purple curve) provide the best performance whereas Deep Features (blue curve) turn out more effective in distinguishing object instances (last 3 rows). This can be explained by observing that Kernel Descriptors seem more effective to encode shape information that, in turn, is more relevant to the task of category recognition,

	Appearance	Shape	Best
RGB-D Object Dataset - Category	Deep RGB	KD DS	KD ICDS
CIN 2D+3D - Category	KD IC	Deep HHA	KD ICDS
RGB-D Object Dataset - Instance	Deep RGB	KD DS	Deep RGB
CIN 2D+3D - Instance	Deep RGB	KD DS	Deep RGB
BigBIRD	Deep RGB	KD DS	Deep RGB

TABLE 6.1: Summary of the results reported in Fig 6.9. For each dataset and both types of experiment, the first two columns highlight the method providing the best recognition rate in case either only appearance or only shape information is deployed for image representation. Then, the last column highlights the approach yielding the highest possible recognition rate assuming that both kinds of information are available.

whereas Deep Features better capture the appearance information that is key to effective instance recognition. In Table 6.1 we summarize the results shown in Fig 6.9 by highlighting the approaches providing the best performance when deploying either appearance or shape information only (first 2 columns). Furthermore, the last column of the table reports the configuration yielding the highest recognition rate when both kinds of information are available. In the case of category recognition, exploiting both appearance and shape information is beneficial as the best configuration involves the combined use of all Kernel Descriptors. Conversely, for the task of instance recognition, our evaluation suggests to simply discard the shape contribution for the available code bits would be best spent to encode the RGB image only by Deep Features.

Puzzled by the above finding, we devised an additional type of instance recognition experiment, whereby the bits of the binary codes are no longer split evenly between appearance and shape but, instead, according to a varying ratio. We run the experiments setting the description length to 1024 bits (i.e. the lengthiest considered in Fig 6.9) while deploying Deep Features to encode the RGB image and Kernel Descriptors (*Depth gradient* and *Spin Image-based*) to encode the depth image, i.e. the best approaches to represent appearance and shape respectively. In Fig 6.10 we report the obtained recognition rates: as expected, peak

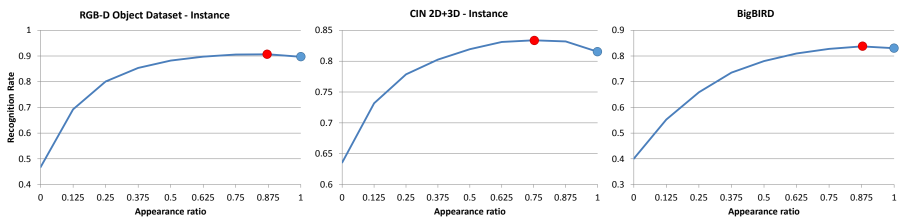


FIGURE 6.10: Instance recognition experiments with a varying relative contribution of appearance (Deep Features) and shape (Kernel Descriptors). The horizontal axis indicates the ratio of bits of the binary code deployed to encode appearance. Accordingly, the performance of the best methods in Table 6.1 are denoted by blue dots (all bits encode appearance by Deep Features). The best recognition rates attainable by splitting code bits unevenly between appearance and shape are highlighted by red dots.

performance are reached with a high ratio of code bits deployed to represent appearance. Interestingly, though, the best performance are never achieved by allocating the totality of the binary code to appearance information, but, rather, by splitting properly code bits between appearance and shape. In particular, with CIN 2D+3D the best recognition rate is reached by allocating $1/4$ of the binary code to shape, while the optimal ratio is $1/8$ for both the RGB-D Object dataset as well as BigBIRD. Indeed, a shape-to-appearance ratio of about $1/8$ would provide better performance than disregarding shape with all the considered datasets. Hence, proper deployment of the depth channel associated with RGB-D images may contribute to improve instance recognition performance even in scenarios where texture and color provide the primary cues to tell objects apart.

6.3 Performance evaluation

Having identified a suitable configuration of the proposed Visual Search architecture by an experimental analysis concerned with the diverse options for the main processing stages, we are now interested in assessing its peak performance as well as in evaluating ours with respect to other comparable proposals.

As for the first objective, the investigation points out that the architecture provides the highest recognition rates when RGB channels are encoded through deep features, whereas shape information is represented by *depth gradient kernels* and *spin kernel descriptors*. Also, according to the *Hashing Fusion* strategy, the descriptions are computed separately and, then, juxtaposed.

As regards the second objective, as already pointed out, unfortunately we are not aware of any Visual Search engine devised to address the case of RGB-D data. For this reason, we are confined to a comparison with search engines dealing with RGB images. Among the available options, we choose the *Compact Descriptors for Visual Search* (CDVS) [1] because, similarly to our proposal, it is specifically aimed at generating compact encodings and it is currently in the Draft International Standard stage as Part 13 of the MPEG-7 standard.

6.3.1 Compact Descriptors for Visual Search

The CDVS pipeline extracts a set of keypoints using a Laplacian of Gaussian (LoG) detector and compute descriptors through SIFT. Then, Fisher kernel is applied to obtain a global encoding of the image that is binarized by labeling each component as 1 if positive and 0 otherwise. To enrich description, in addition to the compressed global encoding, CDVS sends to the server a compressed version of the local SIFT descriptors as well as an histogram-based encoding of the feature coordinates that are subjected to a geometric consistency check by means of the *DISTAT* algorithm [60]. CDVS defines six possible lengths for the final code sent to the server: 512, 1K, 2K, 4K, 8K and 16K bytes.

6.3.2 Results

The comparison between CDVS and our pipeline is reported in Fig. 6.11 for the three datasets.

RGB-D object dataset

Considering first the RGB-D object dataset, the curves clearly show as our proposal is more effective both in the task of category and instance recognition. Moreover, our pipeline proves to be extremely cheap in terms of bandwidth requirements, as its highest recognition rates are reached by transmitting 1024 bits. Conversely, to get its best performance, CDVS requires, at least, binary codes of 1024 bytes in the case of instance recognition and 4K bytes if applied to the task of category recognition. A deeper analysis reveals that CDVS is more effective in the case of instance recognition. Indeed, the difference between the recognition rates of our proposal and CDVS, in the task of instance recognition, is about 20 percentage points, whereas in the case of category recognition the difference grows to nearly 40%.

CIN 2D+3D

The behaviour of the compared pipelines are corroborated by the charts concerning the CIN 2D+3D dataset. Again, our proposal achieves higher recognition rates. In the case of instance recognition, CDVS obtains similar results and confirms that it has been essentially devised to work in instance recognition scenarios. As a matter of fact, the retrieval stage takes advantage of the geometric consistency check on keypoint coordinates and it is enhanced by the matching of the local descriptors which are sent together with the global encoding. This helps the recognition of specific instances of an object but limits the ability to identify its category. Anyway, it is worth noting that our proposal reaches the highest

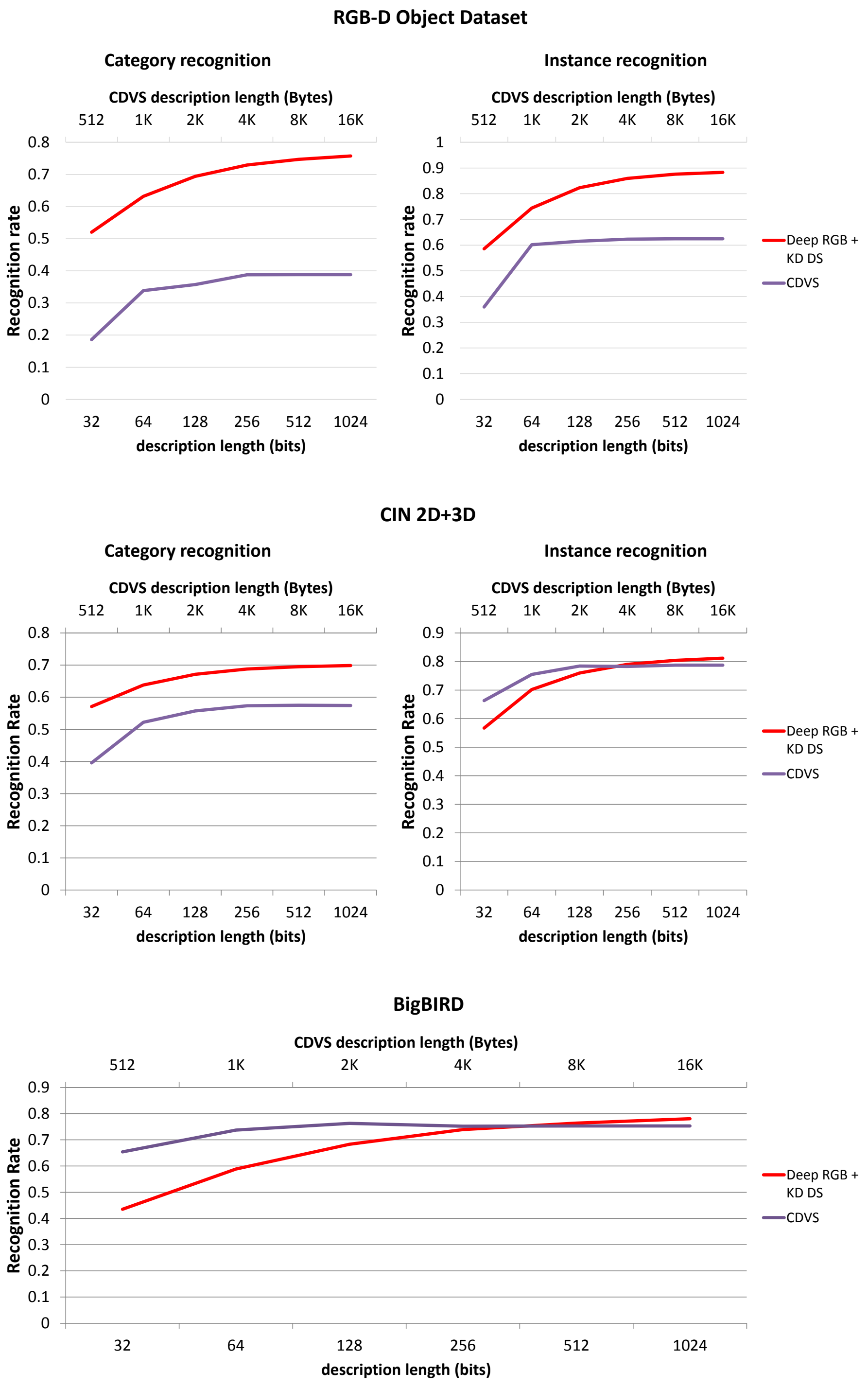


FIGURE 6.11: Comparison between our Visual Search engine (deploying deep features for the encoding of appearance and Kernel descriptors for representing shape information) and CDVS. The top and bottom horizontal axes report, respectively, the description lengths in bytes for CDVS and the description lengths in bits for our proposal.

recognition rate just by exploiting 256 bit descriptions whereas CDVS requires at least 2048 bytes to obtain the best performance.

BigBIRD

The observations made on results regarding instance recognition of CIN 2D+3D dataset are also applicable to the case of BigBIRD dataset. Indeed, CDVS and our proposal share comparable recognition rates, but the former obtains the best performance by exploiting 2048 bytes as description length, whereas the latter attains the same accuracy but encoding a $1/16$-long binary code.

Chapter 7

Implementation of the Visual Search engine on embedded/mobile platforms

This chapter addresses primarily the work carried out during a three-months internship at ST Microelectronics Imaging Division in Grenoble under the supervision of Dr. Alain Issard. The internship was aimed at implementing the architecture described in Chap. 5 on the STxP70 Application-Specific Multiprocessor (ASMP) embedded multi-core platform.

Focus of this thesis is on studying how to aggregate appearance and shape in a compact description. Moreover, the comparison reported in Sec. 6.2.6 shows how the architecture is agnostic with respect to the deployed local descriptors, as it can be fed with different local features. For these reasons, the work performed during the internship addresses the implementation of the Fisher kernel and Spherical hashing algorithms (described, respectively, in Sec. 5.2.2 and Sec. 5.3.2), that encode the image description in a global binary code.

As STxP70 processors do not have hardware support for floating-point operations, both algorithms have been converted to fixed-point and plugged into the Visual Search architecture so as to assess the reduction in recognition rate caused by the adopted fixed-point representation. Then, the algorithms have been executed on the *Gepop* simulator, in order to evaluate the performance and scalability of the implementations while varying of the number of ASMP cores.

The concluding part of this chapter, instead, describes the implementation of the whole architecture on a *Samsung Galaxy Tab Pro 10.1* equipped with a *Structure Sensor* for the acquisition of the depth image. Tests performed by acquiring objects in real-world settings helped to shed light on the limits of the pipeline and future directions of the work.

7.1 Implementation on the STxP70 ASMP platform

7.1.1 STxP70 ASMP architecture

The STxP70 ASMP is a configurable SMP architecture devised by STMicroelectronics with up to 16 STxP70 cores – 32-bit dual-issue RISC CPUs. Fig. 7.1 depicts the architectural template of the STxP70 ASMP. It relies on a one-cycle access shared L1 data memory, organized in several banks with interleaved addressing, all of which can be accessed simultaneously. In case of bank access conflict, processors block until access is granted by a round-robin arbiter that ensures fair access for all processors. Additionally, each processor has 16KB of program cache (I$), but no data cache. The physical prototype for the STxP70 ASMP platform is based on an *Field Programmable Gate Array* (FPGA) implementation whose configuration is limited to 8 STxP70 cores and 512 KB of shared data memory, organized into 32 memory banks.

STxP70 ASMP programming model

The standard parallel programming model supported bySTxP70 ASMP is the OpenMP 2.5 programming model[1]. In the OpenMP programming model, the application code is annotated with a series of pragmas that are interpreted by a compiler to produce parallel code. An OpenMP runtime is linked to the application and is responsible for thread creation, dispatching, parallel scheduling, synchronization and thread termination. Upon compiling code with OpenMP pragmas, the compiler automatically applies the necessary transformations and adds calls to the OpenMP runtime. The user can also explicitly call OpenMP runtime functions within the application code. If parallel code generation is disabled in the compiler options, the OpenMP pragmas in the user code are ignored producing a sequential executable. This is especially useful for debugging purposes, to evaluate the actual parallelization benefits over the sequential version of the application, as well as to ensure portability between sequential and parallel systems.

Gepop simulator

Fisher kernel and Spherical hashing implementations have been tested on a Gepop simulator, a cycle-approximate simulator for the STxP70 ASMP platform. It is built over STxP70 ISS simulators and integrates hardware device models for other components such as DMAs, memories and interconnects. Experimental measurements show that Gepop reports errors in the order of 10% in estimating the execution time on a physical device.

[1] `www.openmp.org`

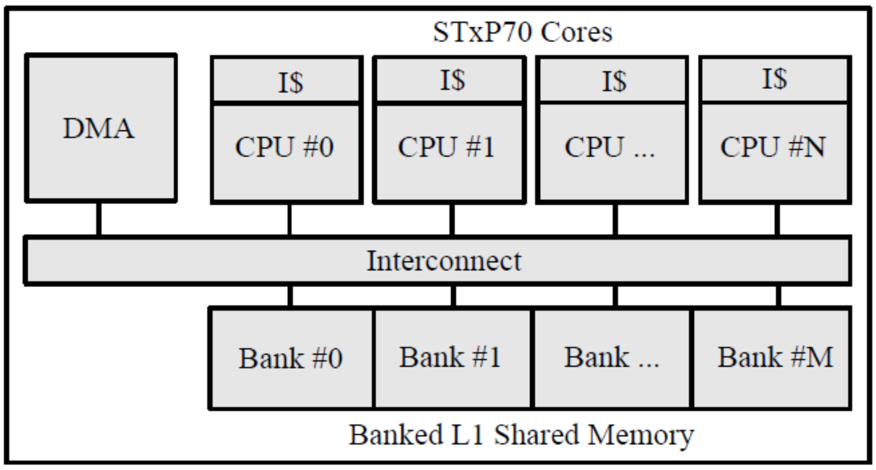


FIGURE 7.1: Architectural framework of the STxP70 ASMP.

7.1.2 Implementation of Fisher Kernel

Given a set of local descriptors $X = \{x_t, t = 1...T\}$, to perform the encoding, at first, the method computes the posteriors $\gamma_i(x_t)$ through 5.4. As the generative model is a mixture of gaussians, the $g_i(x_t)$ are defined as:

$$g_i(x_t) = \frac{w_i}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left(-\frac{1}{2}(x_t - \mu_i)^\mathsf{T} \Sigma_i^{-1} (x_t - \mu_i)\right) \tag{7.1}$$

To avoid numerical instability due to the high dimensionality D of the local descriptor space, 7.1 is expressed as:

$$g_i(x_t) = \exp\left(\ln w_i - \frac{D}{2}\ln\pi - \frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(x_t - \mu_i)^\mathsf{T} \Sigma_i^{-1} (x_t - \mu_i)\right) \tag{7.2}$$

The Fisher Kernel algorithm used in the experiments reported in Chap. 6 includes two improvements proposed in [76]: the former involves the L2 normalization of the final encoding vector aimed at rendering the representation more robust to scale variations. The latter stems from the observation that, as the number of gaussians increases, the vectors become sparser. As a matter of fact, the soft assignment of the descriptors x_t to the gaussians tends to become an hard assignment - only few among the corresponding $\gamma_t(i)$ will turn out not equal to zero - as the number of gaussians increases. Accordingly, most of the $G^X_{\mu,i}$ and $G^X_{\sigma,i}$ (see eq. 5.5 and eq. 5.6) will come out null. A possible solution is to "unsparsify" the encoding by applying the following power normalization:

$$f(z) = sign(z)|z|^{\alpha} \tag{7.3}$$

with $\alpha = 0.5$.

As the work focused mainly on the computational efficiency of the ported algorithm, we performed again the tests described in Sec. 6.2.6 but disabling the two improvements so as to assess their contribution in terms of recognition rate. The comparison has been performed on the three datasets introduced in Chap. 6 by applying only the Kernel descriptor based on intensity gradients. The light blue curves shown in Fig. 7.2 represent the recognition rate obtained if the two improvements are enabled, whereas the red curves let to evaluate the loss in recognition rate when the improvements are lacking. When a binary code of 1024 bits is deployed, only 2-3$\%$ percentage points are lost on average. Unfortunately, the peculiar traits of the datasets used for the comparison do not permit to assess thoroughly the contribution of these two improvements. Indeed, as the datasets have been acquired in controlled setups, they do not include scale variations and, therefore, it is difficult to appreciate the contribution caused by the L2 normalization. Moreover, as the experimental investigation reported in Chap. 7.2 proved that a mixture of only two gaussians is sufficient to properly represent the data, the benefit of the Power normalization is hardly estimable.

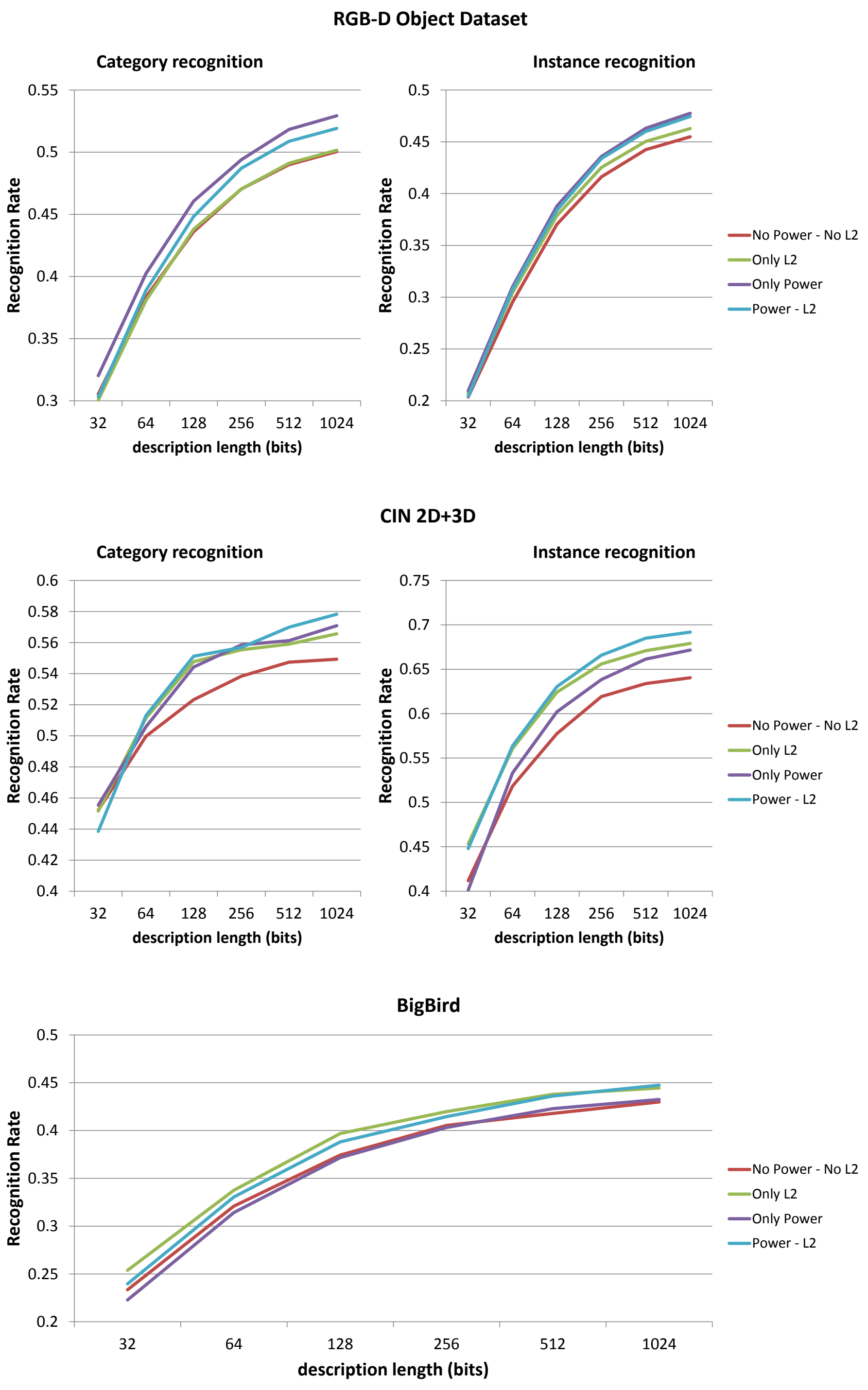


FIGURE 7.2: Contribution in terms of recognition rate of the L2 and Power normalization proposed in [76] on the RGB-D Object, CIN2D+3D and BigBIRD datasets.

Fixed-point conversion

For the fixed-point conversion of the algorithm, a 32 bit-based representation with 16 fractional bits ($Q15.16$) has been adopted. Unfortunately, both the computation of 7.2 and the L2 normalization involve the summation of addends, along the D dimensions of descriptor space, that usually produces overflows of a 32 bit accumulator due to the effect of the “curse of dimensionality”. Such issue has been solved by summing the addends in a 64 bit accumulator and finally converting back the result of 7.2 and L2 normalization to 32 bit representation.

Logarithmic terms in 7.2 are precomputed as they are independent from X, whereas fixed-point computation of exponential is performed by expanding it to a power series:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ... \tag{7.4}$$

The Visual Search architecture has been equipped with the fixed-point version of the algorithm and run on all the three datasets. The results reveal that there is no loss in recognition rate with respect to the floating-point version. Such behaviour could be due to the adopted 64-bit representation that preserves the floating-point precision.

Scalability evaluation on ASMP platform

To evaluate the multi-core scalability of the algorithm on the ASMP platform simulator, the fixed-point version has been parallelized by means of the OpenMP library. As the mixture representing the data is composed by only two gaussians, the processing has not been parallelized by separating the encoding with respect to the mixture components. Instead, the computation of posteriors $\gamma_i(x_t)$ and gradients $G^X_{\mu,i}$ and $G^X_{\sigma,i}$ are performed by splitting up, among the available threads, iterations along the local descriptors X.

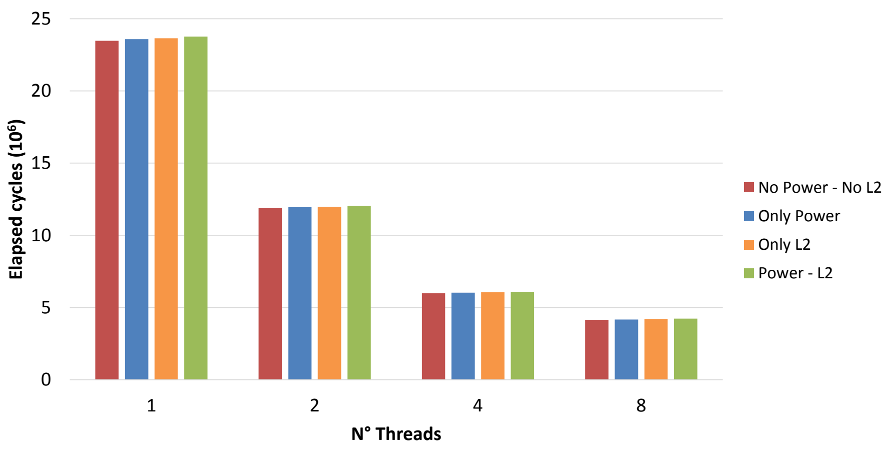


FIGURE 7.3: Execution times, expressed in elapsed processor clock cycles, spent for the computation of a Fisher vector as the number of available ASMP cores increases. The chart shows the computation times in the case the L2 and Power normalization are disabled (No power – No L2), in the case only one is enabled (Only L2, Only Power) and when both are applied (Power – L2).

Fig. 7.3 shows the processor clock cycles[2] spent for the computation of a single Fisher vector as the number of the ASMP cores increases from 1 to 8. The chart proves a good scalability of the algorithm as it is able to exploit the available cores. Furthermore, the chart permits to assess the computational load due to the L2 and Power normalization, which is, indeed, negligible with respect to the execution time spent for the computation of the posteriors $\gamma_i(x_t)$ requiring the computation of the exponential.

7.1.3 Implementation of Spherical Hashing

At hashing time, SH computes the euclidean distance between each hypersphere center and the descriptor produced by the global encoding stage. Then, it evaluates if each distance is greater or smaller than

[2]The clock frequencies for the ASMP could theoretically be as high as 1GHz according to some synthesis results in 28nm, but it is usually used at 500MHz as target frequency. Therefore, the expected execution times can be easily derived from this frequency and the number of elapsed clock cycles.

the hyperphere radius. As such euclidean distances are measured in an high-dimensional space (e.g. in the case of Kernel descriptors encoded by Fisher Kernel, global representations have length $2 \times D \times N_G = 800$), distance computations are affected by the curse of dimensionality, in a similar way to what happens in the case of Fisher kernel. To mitigate such effect, that is worsened by the square operations required by the euclidean distance, we changed the Spherical hashing formulation so as to consider *Manhattan distances* (L1), instead. Such modification affects the training process only on the counting on the number of sample points inside an hyperphere and the consequent update of its radius.

Fig. 7.4 shows the recognition rates yielded by a pipeline equipped with the original formulation of Spherical hashing (L2) and with the one using the Manhattan distance (L1). Here and in the rest of this chapter, we report only the results regarding the category recognition of the RGB-D dataset, as the outcomes in the case of the other datasets show similar trends. The comparison reveals that adopting the formulation based on L1 distance causes an almost constant loss of about three percentage points at varying of the bit-rate. However, such result could be due to the Fisher kernel encoding that has been specifically designed to produce descriptions in the euclidean space and that could be, therefore, beneficial to the original formulation of Spherical Hashing.

Fixed-point conversion

Both the L1 and L2 based formulations of SH have been converted to a fixed-point representation. To investigate on which representation length could be better suited for the fixed-point conversion of Fisher vectors and hypersheres parameters, we performed a comparison between 16, 32 and 64 bit representations. Differently from the static approach adopted for Fisher kernel that always assigns 16 bits to the fractional part, in the case of SH, the number of integer bits is determined on the basis of the data to encode: at training time, the maximum distance

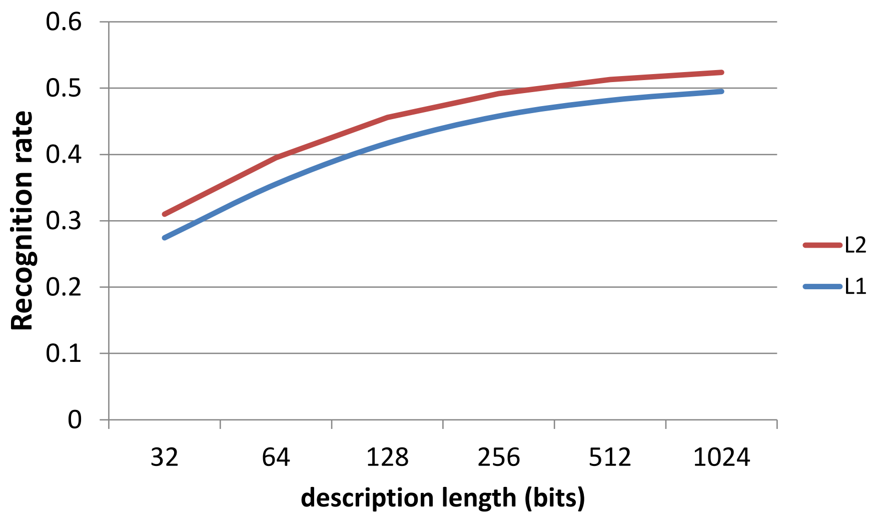


FIGURE 7.4: Recognition rates obtained by applying Spherical hashing based on Euclidean distance (L2) and the modified formulation grounding on Manhattan distance (L1).

between a sample descriptor and an hypersphere center is computed. The number of integer bits is set to the minimum number for which the integer part can represent the maximum distance. Fig.7.5 reports the comparison in recognition rates between the floating-point version and the three fixed-point conversions of the algorithm both in the case of L1 and L2 formulations. Both the charts show that the precision of the 16-bit conversion is not sufficient to represent the data (specially in the case of Euclidean distance). Conversely, 64-bit representation does not induce any loss in recognition rate. 32-bit representation, instead, manifests a different behaviour with respect to the L1 and L2 formulations: it perfectly represents data in the case of the formulation based on Manhattan distance, whereas causes a decay in recognition rate of about two percentage points when data is represented in euclidean space. As 32-bit representation is the most suitable for the ASMP architecture, Fig.7.6 directly compares the L1 and L2 floating-points versions of the algorithm with their corresponding 32-bit fixed point conversions. The chart shows that there is not significant difference between the two fixed-point versions if an adequate bit-rate is deployed.

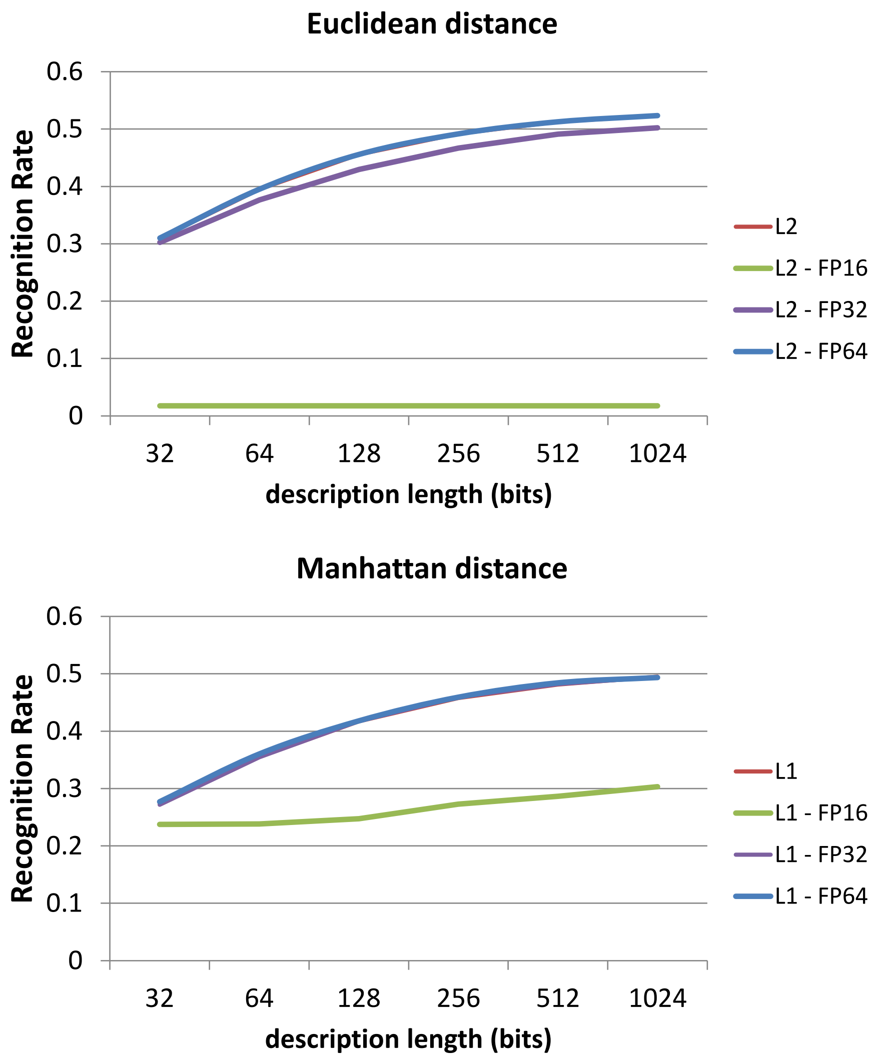


FIGURE 7.5: Comparison between the floating-point version of the Spherical hashing algorithm and the 16, 32 and 64 bit fixed-point porting. Results are reported both in the case of the original formulation (L2) and the modified version deploying the Manhattan distance (L1).

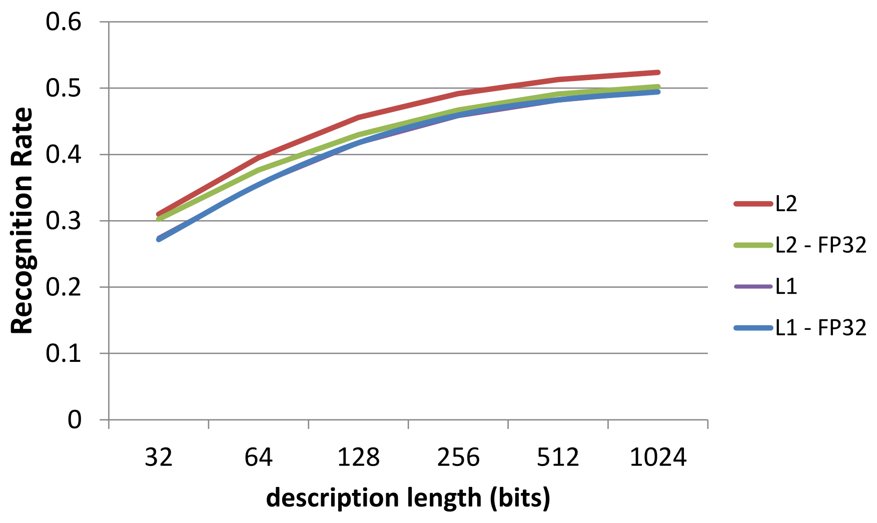


FIGURE 7.6: Comparison between the 32-bit fixed point formulations - based respectively on Euclidean (L1) and Manhattan (L2) distance - of the Spherical hashing algorithm. The corresponding floating-points versions are also reported.

Scalability evaluation on ASMP platform

The 32-bit fixed-point formulations of the algorithm have been implemented to the ASMP architecture. To assess the benefits introduced by the fixed-point conversion, we also performed the porting of the floating-point formulations. Fig. 7.7 reports the execution times required to compute a 1024-bit code as the number of available cores increases. A first look reveals an almost 10$\times$ speedup brought in by the fixed-point representation. The figure also reports a close-up of the fixed-point times to better appreciate their relative differences. The comparison between the formulation based on Euclidean distance and the version grounding on Manhattan distance reports a constant 10% time reduction along the increasing number of available cores. Furthermore, both the formulations prove to perfectly scale with respect to the number of cores.

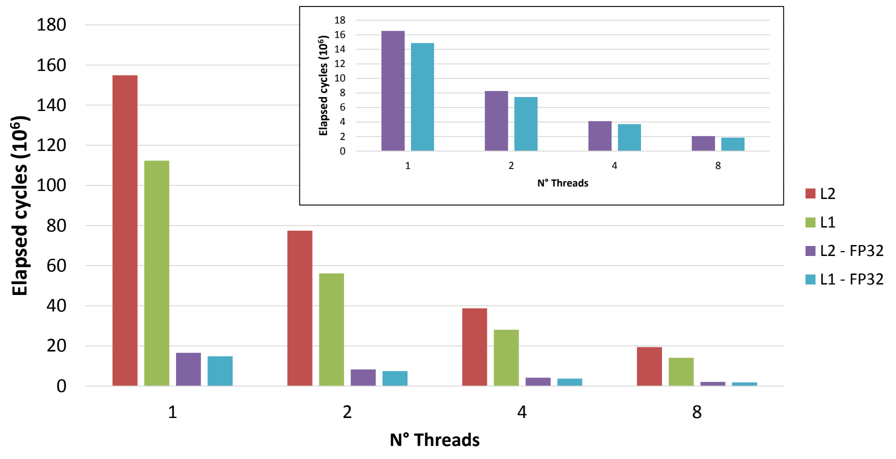


FIGURE 7.7: Computation times (in elapsed processor clock cycles) required to compute a 1024-bit code as varying of the the number of available ASMP cores. The chart shows the executions times regarding the 32-bit fixed-point versions of the L1 and L2 distance based formulations. Moreover, the corresponding floating-point versions are reported. The figure also shows a close-up of the fixed-point times to better perceive their differences.

7.2 Implementation on an Android tablet

The architecture has been deployed to devise an Android app that recognizes the category of an acquired object. The porting has been carried out on a *Samsung Galaxy Tab Pro 10.1* equipped with a *Structure Sensor* for the acquisition of the depth image (see Fig. 7.8). The pipeline describes the acquired RGB-D image with the four types of Kernel Descriptors. Fisher Kernel and Spherical Hashing have been trained on the entire *RGB-D Object Dataset* which is also deployed as database. The framework is composed by two main modules. The former, implemented in Java, includes the user interface (two screenshots are shown in Fig. 7.9) and the acquisition stage. The latter, implemented in C++, comprises the Visual Search engine. When a user acquires an object, the Java module displays the RGB and depth image on tablet screen (shown on the left of Fig. 7.9). Then, through the JNI protocol, the images are sent to the Visual Search engine. The application requires that the object to be recognized is placed on a flat support (such as e.g. a table or

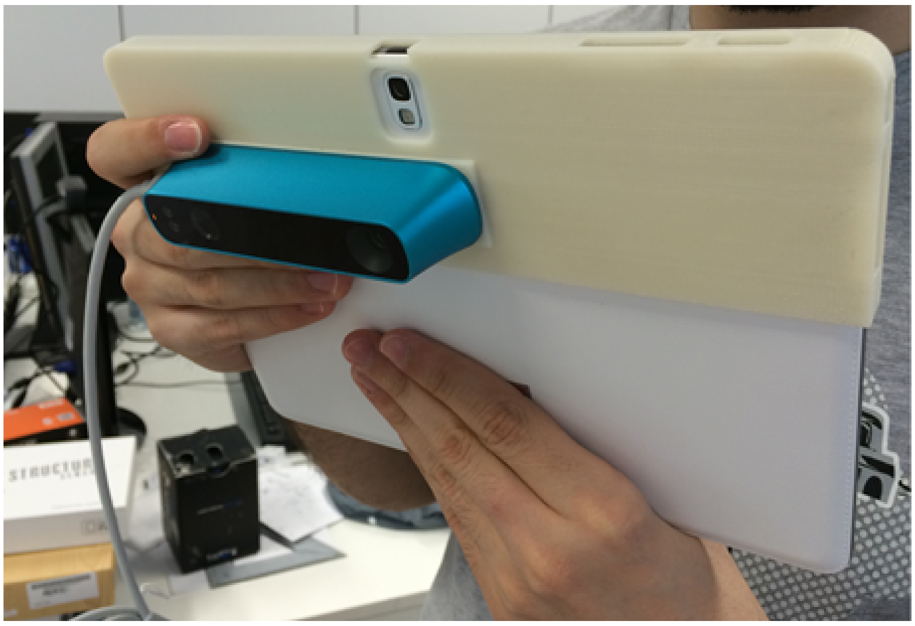
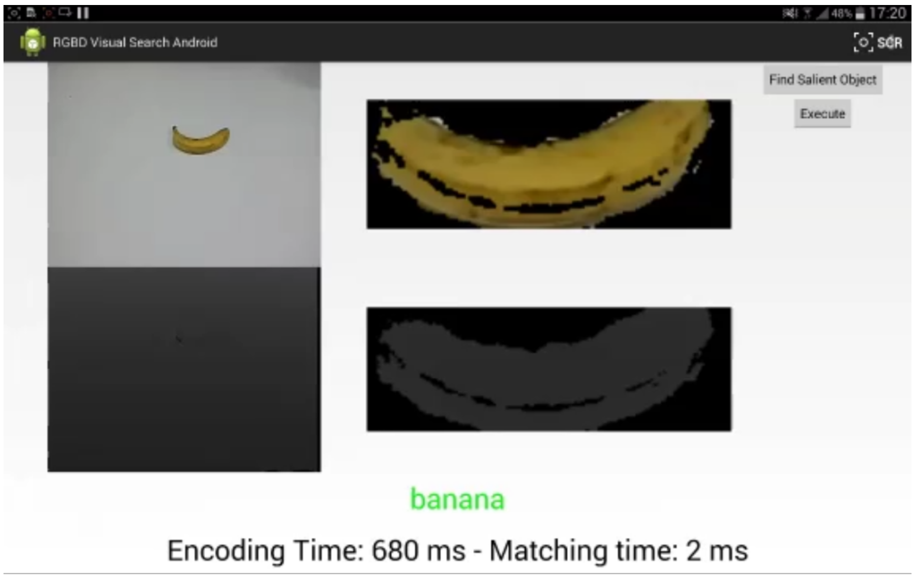


FIGURE 7.8: Structure Sensor clipped onto the Samsung tablet for the acquisition of the depth image.

a floor) so as to easily segment the background from the object through the plane detection algorithm proposed in [101]. More precisely, after the pixels belonging to the plane has been identified, the coordinates of the remaining pixels are modeled as a two-dimensional gaussian so as to define two regions of interest (relative to RGB and depth images) having the center in the mean of the gaussian and as large as 4 times the corresponding standard deviation (the two isolated regions of interest are displayed on the right part of the screen as shown in Fig. 7.9). Then, the engine is applied to the clipped images and the estimated category is displayed on the bottom on the screen (see again 7.9), together with the time required to produce the binary code and the time spent to match it against the database. Tests performed on the different images prove that, on average, the pipeline spends 1870 ms for producing the binary code and 5 ms to perform the matching.


RGBD Visual Search Android
Find Salient Object
Execute
banana
Encoding Time: 680 ms - Matching time: 2 ms

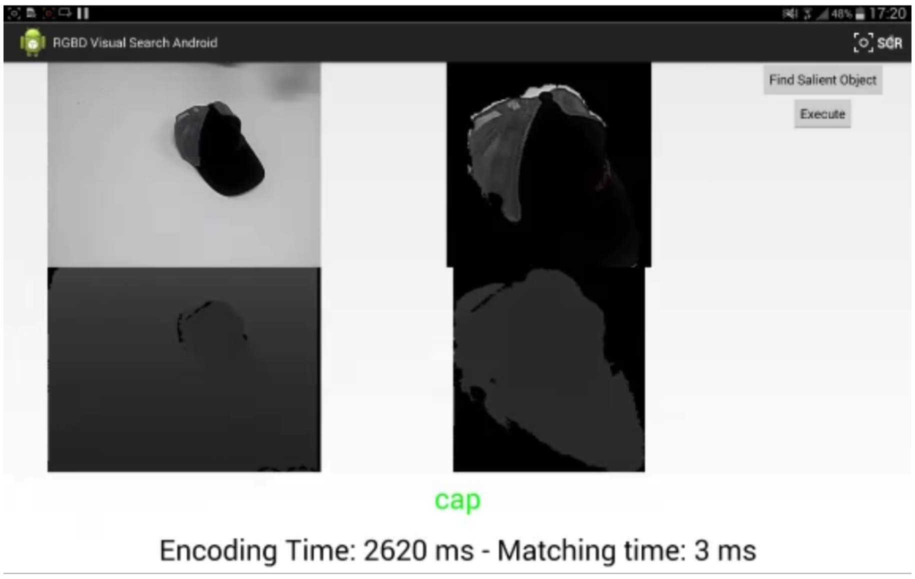

RGBD Visual Search Android
Find Salient Object
Execute
cap
Encoding Time: 2620 ms - Matching time: 3 ms

FIGURE 7.9: Two screenshots of the app user interface. The acquired RGB and depth images are shown on the left of the tablet screen. On the right, the app displays the portions of the images containing the object to recognize as the result of the plane detection procedure that segments the background. The bottom part of the screen displays the category of the object (in green) and execution times required to perform the recognition: the computational time spent for producing the binary code and the time required to perform the matching.

Chapter 8

Concluding remarks

The effectiveness of a 3D Object Recognition system strongly depends on the quality of 3D models representing the objects to recognize. 3D Registration, playing a fundamental role in 3D Reconstruction, addresses the problem of aligning in a unique reference frame two separately acquired views, relying only on information extracted from the object surface shared by the two views. Thus, the former part of the thesis introduces a 3D registration algorithm and points out the lack in literature of an adequate experimental comparison between the different methods proposed in the last twenty years. Purposely, we compare our approach with other three methods on a large corpus of datasets acquired with different sensing devices.

The evaluation neatly shows that, whereas 4PCS [2] gets better results with lower-resolution data, BSL12 [14, 15] is suited for high-precision datasets and SI [50] provides fairly stable performance, our approach attains considerable registration rates on any kind of dataset, regardless of the type of sensor used for acquisition. Furthermore, it runs in times comparable to those of BSL12, which exploits parallelism and works on range images.

As for its limits, even though our proposal can handle successfully noisy data, it is not robust to the presence of outliers in the input data. In the *4-Points Congruent Set* paper, instead, the authors show that 4PCS easily deals with a broad percentage of outliers. Such weakness in our pipeline is due to the definition of the local reference frame that does not account for the presence of spurious points in the support. Another issue is the large number of parameters of our pipeline. However, only a bunch of them actually affects performance significantly. Indeed, the proposed evaluation suggests that most parameters may be left at their default values (Table 4.1), and only the support radius, R_x, adjusted by trial-and-error to optimize performance on unseen data.

Nonetheless, in order to both make its usage even easier and to improve performance, the pipeline may be equipped with an initial stage aimed at estimating some parameters automatically. For example, it may be possible to try to quickly estimate on-line the value of the support radius, R_x, based on a set of random probes, e.g. so as to optimize the trade-off between the information content associated with the basic shape cue deployed to match features, D, and computational efficiency.

In the reconstruction stage, to build the spanning tree including the best pairwise alignments, it is necessary to check for all the combinations of view pairs so as to find those showing high overlaps. This process can be costly, especially in case of datasets, like *Venus*, *Shell* and *OilPump*, comprising a large number of views. Even though the *Hough voting* stage is itself rather inexpensive, such a large number of runs may slow down reconstruction time notably. Therefore, it would be beneficial to be able to abort the registration before the Hough stage in case a view pair is unlikely to belong to the spanning tree. This may be done efficiently on the basis of the distribution of the scores resulting from the fast feature matching stage. More precisely, as a small $\widetilde{D}_{ij}$ is more likely to come from a good correspondence while a larger one will come from a wrong correspondence (see Fig. 3.4), we may analyze the distribution of $\widetilde{D}_{ij}$ in the current view pair, $p\left(\widetilde{D}_{ij}\right)$, so as to guess, e.g. based on the probability of $\widetilde{D}_{ij}$ to be small enough ($p\left(\widetilde{D}_{ij} < T\right)$), whether the pair provides enough good correspondences and, as such, is likely to belong to the spanning tree.

As already mentioned, the main bottleneck of the pipeline resides in the search for neighboring points, especially in the extraction of the spherical support used for the computation of the local reference frame. Indeed, a standard kd-tree extracts all the points inside the sphere, which mandates a further filtering operation to then select only those in the shell used to determine the tangential axis. To speed-up the search, a dedicated indexing scheme may be devised to allow for a radius search that directly extracts only the useful points in the shell of the sphere.

Mobile devices with synchronized depth and color sensing may be foreseen to become more and more widespread in the forthcoming future. Based on these premises, the latter part of the thesis proposes the first investigation towards a plausible architecture aimed at Visual Search through RGB-D images and compact binary representations.

The findings emerging from experimental analysis suggest that a pipeline deploying densely computed Kernel Descriptors aggregated at the image level through Fisher Kernel followed by Spherical Hashing constitutes an effective reference architecture. Similar performance can be obtained by compressing Deep Features computed via Convolutional Neural Networks. In particular, Deep Features seem the best choice to represent appearance, whereas shape information is better captured by Kernel Descriptors.

Indeed, recognition rates between 70-80% can be achieved with binary codes as compact as 512-1024 bits in both category and instance retrieval experiments. Moreover, keeping the processing flows of the color and depth channels separate to concatenate the final binary codes seems not to hinder performance while potentially allowing for a great deal of flexibility at the system and application level.

In category recognition scenarios, both RGB and depth information contribute notably to ascertain the class to which a query object does belong. Instead, in instance recognition tasks, our experiments highlight how appearance features, like texture and colour, are key to tell apart the specific object instances stored into the database, whereas depth furnishes a limited, though still informative, contribution. Indeed, an approach based on simply juxtaposing the two representations does not take into account the different discriminative power that the two cues may convey in diverse scenarios. Hence, devising suitable strategies to learn and deploy the relative prominence of appearance and depth in diverse settings is among the key research issues to be addressed in order to leverage on depth sensing in forthcoming Mobile Visual Search scenarios. We are currently investigating on a learning-to-rank approach aimed at discovering the contribution brought in by the different descriptions on the basis of the Hamming distances between query and database binary codes.

As already mentioned, the available RGB-D datasets are not sufficiently realistic to take into account many of the nuisances occurring in real-world settings. In particular, the three datasets used in our evaluation deal with controlled setups wherein the illumination has been kept constant and the objects are always at the same distance from the sensor. In addition, they do not comprise clutter as the objects have been segmented from the background. This state of affairs calls for some substantial effort to create new datasets aimed at evaluating RGB-D Visual Search engines. As a first contribution along this direction, we have developed a standardized software interface that allows the aggregation of existing datasets[1] so as to treat them collectively and seamlessly as the server side database of an RGB-D Visual Search pipeline. We will then provide query RGB-D images dealing with objects belonging to the categories in the database and taken under uncontrolled settings and clutter, so as to enable more realistic category recognition experiments.

In such more realistic setup – in which objects comprising the database are typically segmented from the background, whereas query images may be affected by any type of clutter – the ability to properly describe the portion of the image containing the object of interest and discard

[1]Besides the datasets considered in this thesis, it is worth highlighting the MV-RED dataset, recently introduced in [62].

the surrounding background appears essential to effective recognition. To give an example, the app described at the end of chapter 7 performs an initial background detection. Unfortunately, it relies on an unrealistic assumption of background flatness. Accordingly, we are working on a more general framework able to weigh more the local features describing the object than those representing the surrounding background based on automatic estimation of a per-pixel visual saliency score.

During the very last part of the Ph.D., the learning-to-rank approach as well as the saliency-based encoding have been investigated and, even if at a very early stage, they proved to be promising. For this reason, the next two sections describe their current development and report preliminary experimental results.

8.1 Learning to rank color and depth

In a information retrieval system, in which different strategies independently produce different rankings, learning-to-rank approaches are used to learn a ranking model that fuses the individual ranking into a joint ranking in order to improve the recognition capabilities of the system. In our architecture, each processing flow describes the query image with a different descriptor (Kernel Descriptors, Deep Features, etc...) and produces a binary code that is matched in the Hamming space. Therefore, a processing flow can be seen as a strategy that yields a ranking of the database images based on the Hamming distance. Learning-to-rank approaches perform a supervised learning aimed at discovering which strategies produce better rankings in a particular task and, hence, learn how to properly weigh them in the final ranking. Such behaviour fulfills, in principle, our requirement to treat differently the contributions of the color and depth descriptions in category and instance recognition tasks. Accordingly, we exploit the *Ranking SVM* method proposed in [41, 49] that formulates the learning-to-rank problem as a classification problem.

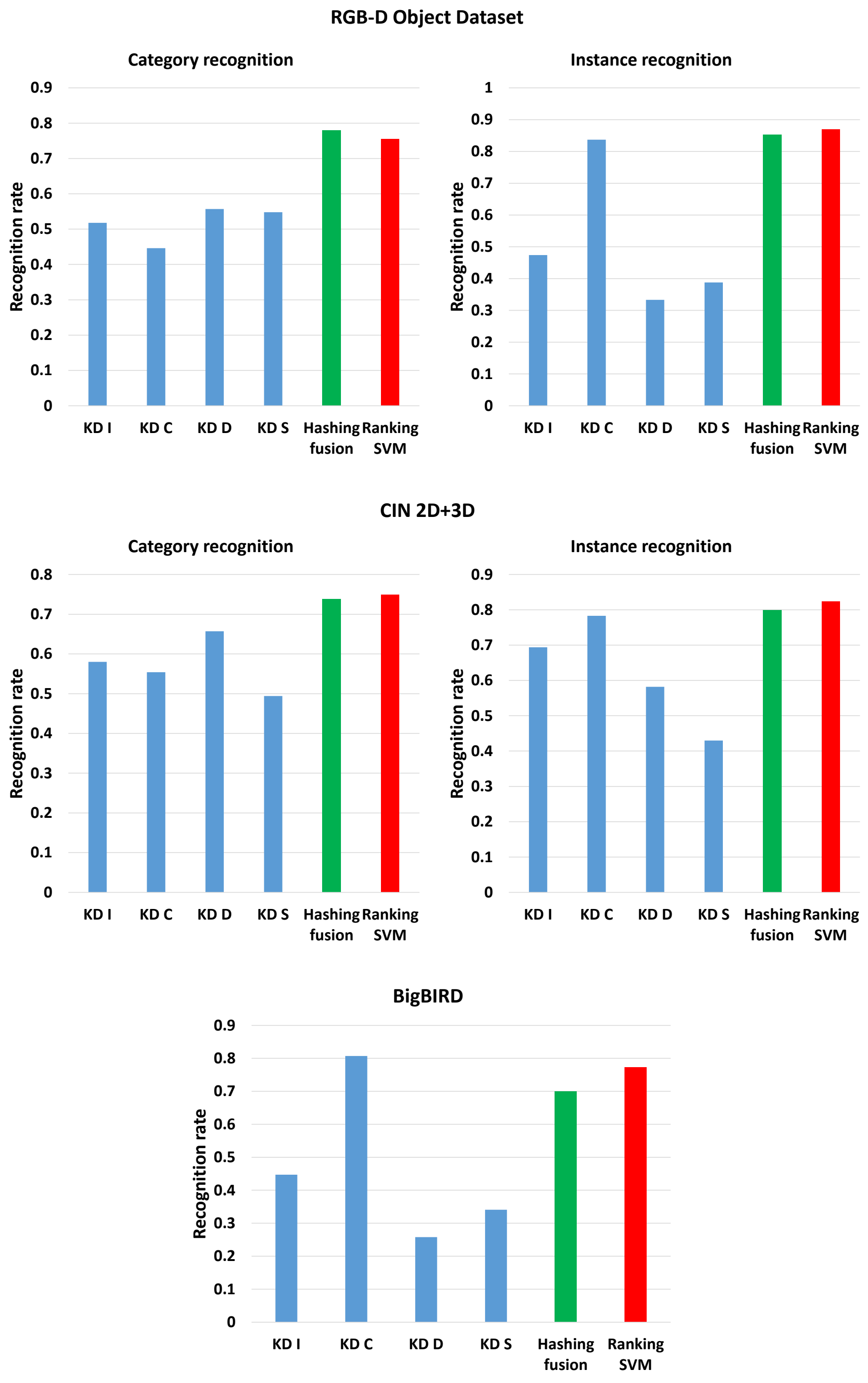


FIGURE 8.1: Preliminary results attained by exploiting the learning-to-rank approach (Ranking SVM) on the three datasets: RGB-D Object Dataset, CIN2D+3D and BigBIRD. Each chart reports the comparison with the recognition rates obtained by the pipelines deploying separately the four Kernel descriptors (KD I, KD C, KD D, KD S) and concatenating them through the Hashing fusion strategy (Hashing fusion). Overall, each pipeline transmits 1024 bits to the server.

Purposely, we applied our pipeline exploiting the four Kernel descriptors to obtain four-dimensional feature vectors containing the Hamming distances between the binary codes computed on the query and database images. Fig. 8.1 reports the comparison between the learning-to-rank approach and the Hashing fusion strategy which simply juxtaposes the binary codes produced by the separate encoding of the four Kernel descriptors. Moreover, the charts report the recognition rates obtained by deploying the four Kernel descriptors separately. As already discussed in Chap. 7, simple concatenation of the binary descriptions leads to an improvement in terms of recognition rates in the task of category recognition but turns out ineffective if applied in the recognition of the specific instance of an object and even detrimental in the case of BigBIRD dataset where loses about 10 percentage points with respect to the deployment of the Kernel descriptor based on color features only. Instead, the learning-to-rank approach better combines the contribution brought in by the different types of descriptions. As a matter of fact, in the case of BigBIRD dataset, the recognition rate raises to 77% and comes close to pair the performance of the description based only on color. This result proves that the method learned to weigh more the description based on color as also corroborated by the coefficients of the hyperplane estimated by the SVM training which show a prominent maximum corresponding to the coefficient related to the color feature. Such behaviour seems confirmed by the experiments on the other datasets in which the learning-to-rank approach slightly outperforms the Hashing fusion strategy in three out of four cases.

8.2 Saliency-based encoding

In real-world settings, the object acquired through the query image may be immersed in any kind of background. Therefore, relying on an estimation of the visual saliency of the diverse areas of the image holds the

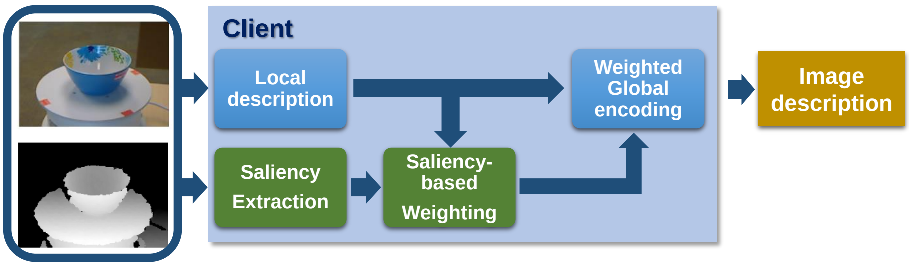


FIGURE 8.2: Outline of the pipeline deploying saliency to stronger weigh local features belonging to the object.

potential for isolating and recognizing the object. So as to validate such idea, we devised the pipeline depicted in Fig. 8.2.

Given a query image, the pipeline extracts a set of local descriptors. Furthermore, a saliency map (see Fig. 8.3 for an example) is extracted through the method proposed in [74], which synergically exploits both RGB and depth information. Then, a weight is assigned to each local feature grounding on the saliences of the pixels of the feature patch. So as to consider these saliency-based weights in the global representation of the image, we extended the Fisher Kernel formulation by introducing the feature weights in the computation of 5.5 and 5.6:

$$G^X_{\mu,i} = \frac{1}{\Omega\sqrt{\alpha_i}} \sum_{t=1}^{T} \omega_t \gamma_t(i) \left(\frac{x_i - \mu_i}{\sigma_i}\right) \tag{8.1}$$

$$G^X_{\sigma,i} = \frac{1}{\Omega\sqrt{2\alpha_i}} \sum_{t=1}^{T} \omega_t \gamma_t(i) \left[\frac{(x_i - \mu_i)^2}{\sigma_i^2} - 1\right] \tag{8.2}$$

where ω_t represents the weight associated to descriptor x_t and $\Omega = \sum_{t=1}^{T} \omega_t$.

As already discussed, a dataset suited for the evaluation of our approach does not exist yet. Accordingly, we have workarounded such issue by modifying the BigBIRD dataset so to vary the backgrounds of the images comprising the dataset. Precisely, for each image, we replaced

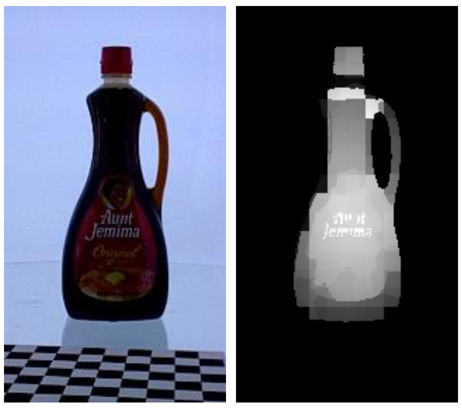


FIGURE 8.3: Example of saliency map. Brighter pixels represent more salient portions of the image.

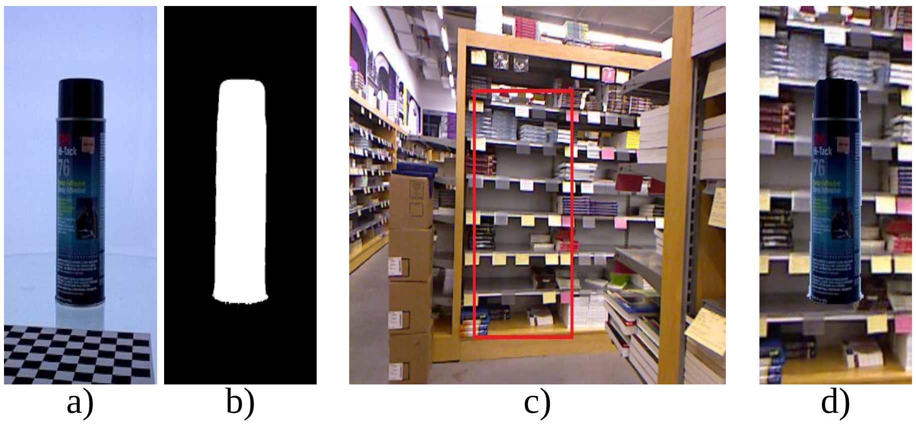


FIGURE 8.4: Procedure performed for modifying the BigBIRD dataset: for each image (a), the mask (b) included in the dataset is used to segment and discard the background. Then, a random image of the *NYU Depth V1* dataset is cropped at a random position so as to extract a region of the same size of the image (c). Finally, the foreground of the original image is merged with the extracted region (d).

the background by exploiting the mask provided by the authors of the dataset (see Fig. 8.4 b). The new background is extracted by cropping a random patch – of the same size of the image – from a random image of the *NYU Depth V1* introduced in [86] (see again Fig. 8.4 c).

We performed a comparison on this dataset between the pipeline deploying the standard formulation of Fisher Kernel and this new pipeline weighting on the basis of the saliency. The results, showing an improvement of about 20% in terms of recognition rate with respect to the standard pipeline, prove how the deployment of saliency may turn out highly beneficial in real-world scenarios.

Bibliography

[1] ISO/IEC JTC 1/SC 29/WG 11, Information technology - Multimedia content description interface - Part 13: Compact descriptors for visual search. 2014.

[2] D. Aiger, N. J. Mitra, and D. Cohen-Or. “4-points Congruent Sets for Robust Surface Registration”. In: *ACM Transactions on Graphics* 27.3 (2008), 85:1–85:10.

[3] Andrea Albarelli, Emanuale Rodolà, and Andrea Torsello. “Loosely Distinctive Features for Robust Surface Alignment”. In: *European Conference On Computer Vision (ECCV)*. 2010.

[4] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. “Anisotropic Polygonal Remeshing”. In: *ACM SIGGRAPH Papers*. San Diego, California, 2003, pp. 485–493.

[5] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. “The Power Crust”. In: *6th ACM Symposium on Solid Modeling*. 2001, pp. 249–260.

[6] Prabin Bariya, John Novatnack, Gabriel Schwartz, and Ko Nishino. “3D Geometric Scale Variability in Range Images: Features and Descriptors”. In: *International Journal of Computer Vision* 99.2 (2012), pp. 232–255.

[7] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. “The ball-pivoting algorithm for surface reconstruction”. In: *IEEE Transactions on Visualization and Computer Graphics* 5.4 (1999), pp. 349–359.

[8] Paul J. Besl and Neil D. McKay. “A method for registration of 3-D shapes”. In: *Transactions on Pattern Analysis and Machine Intelligence* (1992).

[9] Manuel Blum, Jan Wulfing, and Martin Riedmiller. “A learned feature descriptor for object recognition in RGB-D data”. In: *International Conference on Robotics and Automation (IROS)*. Ieee, 2012, pp. 1298–1303.

[10] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Depth kernel descriptors for object recognition”. In: *Intelligent Robots and Systems (IROS)*. 2011.

[11] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Kernel descriptors for visual recognition”. In: *Advances in Neural Information Processing Systems 23* (2010), pp. 1–9.

[12] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “Unsupervised feature learning for rgb-d based object recognition”. In: *International Symposium on Experimental Robotics* (2012), pp. 1–15.

[13] Liefeng Bo and Cristian Sminchisescu. “Efficient match kernel between sets of features for visual recognition”. In: *Advances in Neural Information Processing Systems* (2009), pp. 1–9.

[14] Francesco Bonarrigo, Alberto Signoroni, and Riccardo Leonardi. “A robust pipeline for rapid feature-based pre-alignment of dense range scans”. In: *International Conference on Computer Vision (ICCV)*. 2011, pp. 2260–2267.

[15] Francesco Bonarrigo, Alberto Signoroni, and Riccardo Leonardi. “Multi-view alignment with database of features for an improved usage of high-end 3D scanners”. In: *Journal on Advances in Signal Processing (EURASIP)* 2012.1 (2012), p. 148.

[16] B Browatzki and Jan Fischer. “Going into depth: Evaluating 2D and 3D cues for object classification on a new, large-scale object dataset”. In: *International Conference on Computer Vision Workshops (ICCV Workshops)* (2011).

[17] Benedict J. Brown and Szymon Rusinkiewicz. “Global non-rigid alignment of 3-D scans”. In: *ACM SIGGRAPH* (2007), p. 21.

[18] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. “Brief: Binary robust independent elementary features”. In: *European Conference on Computer Vision (ECCV)* (2010).

[19] U Castellani, M Cristani, S. Fantoni, and V. Murino. “Sparse points matching by combining 3D mesh saliency with statistical descriptors”. In: *Computer Graphics Forum* (2008).

[20] Vijay Chandrasekhar, Mina Makar, Gabriel Takacs, David Chen, Sam S. Tsai, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, and Bernd Girod. “Survey of SIFT compression schemes”. In: *International Conference on Pattern Recognition* (2010).

[21] Vijay Chandrasekhar, Gabriel Takacs, David M. Chen, Sam S. Tsai, Mina Makar, and Bernd Girod. “Feature Matching Performance of Compact Descriptors for Visual Search”. In: *Data Compression Conference* (2014).

[22] Vijay Chandrasekhar, Gabriel Takacs, David M. Chen, Sam S. Tsai, Yuriy Reznik, Radek Grzeszczuk, and Bernd Girod. “Compressed Histogram of Gradients: A Low-Bitrate Descriptor”. In: *International Journal of Computer Vision* (2011).

[23] David M. Chen, Sam S. Tsai, Vijay Chandrasekhar, Gabriel Takacs, Jatinder Singh, and Bernd Girod. “Tree Histogram Coding for Mobile Image Matching”. In: *Data Compression Conference* (2009), pp. 143–152.

[24] C. S. Chua and R. Jarvis. “Point Signatures: A New Representation for 3D Object Recognition”. In: *International Journal of Computer Vision* 25.1 (1997), pp. 63–85.

[25] DH Chung, ID Yun, and SU Lee. “Registration of multiple-range views using the reverse-calibration technique”. In: *Pattern Recognition* 31.4 (1998), pp. 457–464.

[26] B. Curless and M. Levoy. “A volumetric method for building complex models from range images”. In: *SIGGRAPH*. 1996, pp. 303–312.

[27] Brian Curless and Marc Levoy. “A volumetric method for building complex models from range images”. In: *SIGGRAPH*. 1996, pp. 303–312.

[28] James Davis, Diego Nehab, Ravi Ramamoorthi, and Szymon Rusinkiewicz. “Spacetime Stereo: A Unifying Framework for Depth from Triangulation”. In: *Transactions on Pattern Analysis and Machine Intelligence.* 27.2 (2005), pp. 296–302.

[29] Sahibsingh a. Dudani. “The Distance-Weighted k-Nearest-Neighbor Rule”. In: *Transactions on Systems, Man, and Cybernetics* (1976), pp. 325–327.

[30] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. “3D Mapping with an RGB-D Camera”. In: *Transactions on robotics* (2012), pp. 1–11.

[31] Simone Fantoni, Umberto Castellani, and Andrea Fusiello. “Accurate and automatic alignment of range surfaces”. In: *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*. 2012.

[32] Jacques Feldmar and Nicholas Ayache. “Rigid, affine and locally affine registration of free-form surfaces”. In: *International Journal of Computer Vision* (1996).

[33] MA Fischler and RC Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* (1981).

[34] Andrew W Fitzgibbon. “Robust registration of 2D and 3D point sets”. In: *Computer Vision and Image Understanding* 2.13-14 (2003), pp. 1145–1153.

[35] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. “Recognizing Objects in Range Data Using Regional Point Descriptors”. In: *European Conference On Computer Vision (ECCV)*. Vol. 3. 2004, pp. 224–237.

[36] Bernd Girod, Vijay Chandrasekhar, David M. Chen, Ngai-Man Cheung, Radek Grzeszczuk, Yuriy Reznik, Gabriel Takacs, Sam S. Tsai, and Ramakrishna Vedantham. “Mobile visual search”. In: *Signal Processing Magazine, IEEE* July (2011), pp. 61–76.

[37] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. “Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013), pp. 2916–2929.

[38] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. “A Comprehensive Performance Evaluation of 3D Local Feature Descriptors”. In: *International Journal of Computer Vision* (2015), pp. 1–24.

[39] Saurabh Gupta, Ross Girshick, Pablo Arbel, and Jitendra Malik. “Learning Rich Features from RGB-D Images for Object Detection and Segmentation”. In: *European Conference on Computer Vision (ECCV)* (2014), pp. 1–16.

[40] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. “Spherical hashing”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 2957–2964.

[41] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. “Large Margin Rank Boundaries for Ordinal Regression”. In: *Advances in Large Margin Classifiers*. MIT Press, 2000. Chap. 7, pp. 115–132.

[42] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. “Surface reconstruction from unorganized points”. In: *SIGGRAPH*. 1992, pp. 71–78.

[43] Berthold K. P. Horn. “Closed-form solution of absolute orientation using unit quaternions”. In: *Journal of the Optical Society of America A* 4.4 (1987), pp. 629–642.

[44] Berthold K. P. Horn. “Extended gaussian images”. In: *Proceedings of the IEEE* 12 (1984).

[45] Piotr Indyk and Rajeev Motwani. “Approximate nearest neighbors: towards removing the curse of dimensionality”. In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (1998), pp. 604–613.

[46] T Jaakkola and D Haussler. “Exploiting generative models in discriminative classifiers”. In: *Advances in Neural Information Processing Systems* (1999).

[47] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. “Aggregating local descriptors into a compact image representation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2010).

[48] Rongrong Ji, Ling-Yu Duan, Jie Chen, Hongxun Yao, Junsong Yuan, Yong Rui, and Wen Gao. “Location Discriminative Vocabulary Coding for Mobile Landmark Search”. In: *International Journal of Computer Vision* (2011), pp. 290–314.

[49] Thorsten Joachims. “Optimizing search engines using clickthrough data”. In: *International conference on Knowledge discovery and data mining*. 2002, pp. 133–142.

[50] Andrew Edie Johnson and Martial Hebert. “Surface registration by matching oriented points”. In: *International Conference on 3D Digital Imaging and Modeling* (1997).

[51] Andrew Edie Johnson and Martial Hebert. “Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes”. In: *Pattern Analysis and Machine Intelligence* 21 (1999), pp. 433–449.

[52] Matthew Johnson. “Generalized Descriptor Compression for Storage and Matching”. In: *British Machine Vision Conference* (2010), pp. 23.1–23.11.

[53] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. “Poisson Surface Reconstruction”. In: *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Cagliari, Sardinia, Italy: Eurographics Association, 2006, pp. 61–70.

[54] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. “Hough transform and 3D SURF for robust three dimensional classification”. In: *European Conference On Computer Vision (ECCV)*. 2010, pp. 589–602.

[55] R Kolluri, J R Shewchuk, and J F O’Brien. “Spectral Surface Reconstruction from Noisy Point Clouds”. In: *SIGGRAPH*. Vol. 71. 2004, pp. 11–21.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances In Neural Information Processing Systems* (2012), pp. 1–9.

[57] NA Kulkarni and S Kumar. “Vote based correspondence for 3D point-set registration”. In: *Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP)* (2012).

[58] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. “A large-scale hierarchical multi-view rgb-d object dataset”. In: *International Conference on Robotics and Automation* (2011), pp. 1817–1824.

[59] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2006).

[60] Skjalg Lepsoy, Gianluca Francini, Giovanni Cordara, Pedro Porto, and Buarque de Gusmiio. “Statistical modelling of outliers for fast visual search”. In: *International Conference on Multimedia and Expo* (2011).

[61] Xinju Li and Igor Guskov. “Multi-scale Features for Approximate Alignment of Point-based Surfaces”. In: *Eurographics Symposium on Geometry Processing* (2005).

[62] Anan Liu, Zhongyang Wang, Weizhi Nie, and Yuting Su. “Graph-based characteristic view set extraction and matching for 3D model retrieval”. In: *Information Sciences* 320 (2015), pp. 429–442.

[63] Wang Liying and Song Weidong. “A review of range image registration methods with accuracy evaluation”. In: *Urban Remote Sensing Joint Event* (2009), pp. 1–8.

[64] William E. Lorensen and Harvey E. Cline. “Marching cubes: A high resolution 3D surface construction algorithm”. In: *SIGGRAPH*. Vol. 21. 4. 1987, pp. 163–169.

[65] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal on Computer Vision* 60.2 (2004), pp. 91–110.

[66] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. “Multi-probe LSH: efficient indexing for high-dimensional similarity search”. In: *International Conference on Very Large Data bases* (2007).

[67] A. Makadia, A.I. Patterson, and K. Daniilidis. “Fully Automatic Registration of 3D Point Clouds”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2006.

[68] M. R G Márquez and Shin Ting Wu. “An automatic crude registration of two partially overlapping range images”. In: *Brazilian Symposium on Computer Graphics and Image Processing* (2008), pp. 245–252.

[69] Takeshi Masuda. “Log-polar height maps for multiple range image registration”. In: *Computer Vision and Image Understanding* 113.11 (2009), pp. 1158–1169.

[70] A. Mian, M. Bennamoun, and R. Owens. “A Novel Representation and Feature Matching Algorithm for Automatic Pairwise Registration of Range Images”. In: *International Journal of Computer Vision* 66.1 (2006), pp. 19–40.

[71] NJ Mitra, LJ Guibas, and Mark Pauly. “Partial and approximate symmetry detection for 3D geometry”. In: *ACM Transactions on Graphics* (2006).

[72] Toru Nakashika, Takafumi Hori, Tetsuya Takiguchi, and Yasuo Ariki. “3D-Object Recognition Based on LLC Using Depth

Spatial Pyramid". In: *International Conference on Pattern Recognition* (2014).

[73] Ko Nishino and Katsushi Ikeuchi. "Robust simultaneous registration of multiple range images". In: *Asian Conference on Computer Vision (ACCV)*. January. 2002, pp. 71–88.

[74] Houwen Peng, Bing Li, Weihua Xiong, Weiming Hu, and Rongrong Ji. "RGBD Salient Object Detection : A Benchmark and Algorithms". In: *European Conference on Computer Vision*. 1. 2014, pp. 92–109.

[75] Florent Perronnin and Christopher Dance. "Fisher kernels on visual vocabularies for image categorization". In: *Conference on Computer Vision and Pattern Recognition (CVPR)* (2007).

[76] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. "Improving the fisher kernel for large-scale image classification". In: *European Conference on Computer Vision (ECCV)*. 2010, pp. 143–156.

[77] Alioscia Petrelli and Luigi Di Stefano. "A Repeatable and Efficient Canonical Reference for Surface Matching." In: *Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*. 2012, pp. 403–410.

[78] Alioscia Petrelli and Luigi Di Stefano. "On the repeatability of the local reference frame for partial shape matching". In: *International Conference on Computer Vision (ICCV)*. 2011, pp. 2244–2251.

[79] Kari Pulli. "Multiview registration for large data sets". In: *3Dim* (1999), pp. 160–168.

[80] S Rusinkiewicz and M Levoy. "Efficient variants of the ICP algorithm". In: *International Conference on 3D Digital Imaging and Modeling* (2001).

[81] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast Point Feature Histograms (FPFH) for 3D registration”. In: *International Conference on Robotics and Automation* (2009), pp. 3212–3217.

[82] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. “A review of recent range image registration methods with accuracy evaluation”. In: *Image and Vision Computing* 25 (2007), pp. 578–596.

[83] Thiago R. dos Santos, A. Franz, H.-P Meinzer, and L. Maier-Hein. “Robust multi-modal surface matching for intra-operative registration”. In: *25th International Symposium on Computer-Based Medical Systems* 0 (2011), pp. 1–6.

[84] A. Segal, D. Haehnel, and S. Thrun. “Generalized-ICP”. In: *Robotics: Science and Systems*. 2009.

[85] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. “Real-time human pose recognition in parts from single depth images”. In: *Computer Vision and Pattern Recognition* (2011), pp. 1297–1304.

[86] Nathan Silberman and Rob Fergus. “Indoor scene segmentation using a structured light sensor”. In: *International Conference on Computer Vision Workshops (ICCV Workshops)* (2011), pp. 601–608.

[87] Arjun Singh, James Sha, Karthik S. Narayan, Tudor Achim, and Pieter Abbeel. “BigBIRD: A large-scale 3D database of object instances”. In: *International Conference on Robotics and Automation* (2014), pp. 509–516.

[88] Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. “Convolutional-Recursive Deep

Learning for 3D Object Classification”. In: *Advances in Neural Information Processing Systems* (2012), pp. 1–9.

[89] F. Stein and G. Medioni. “Structural Indexing: Efficient 3-D Object Recognition”. In: *Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 125–145.

[90] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *International Conference on Intelligent Robot Systems*. 2012.

[91] Y. Sun and M. A. Abidi. “Surface Matching by 3D Point’s Fingerprint”. In: *International Conference on Computer Vision (ICCV)* 2 (2001), pp. 263–269.

[92] G. K. L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, F. C. Langbein, Yonghuai Liu, D. Marshall, R. R. Martin, Xian-Fang Sun, and P. L. Rosin. “Registration of 3D point clouds and meshes: a survey from rigid to nonrigid.” In: *Transactions on Visualization and Computer Graphics* 19.7 (2013), pp. 1199–217.

[93] JP Tarel, H Civi, and DB Cooper. “Pose estimation of free-form 3D objects without point matching using algebraic surface models”. In: *Workshop Model Based 3D Image Analysis* (1998).

[94] F. Tombari, S. Salti, and L. Di Stefano. “Unique shape context for 3d data description”. In: *ACM Workshop on 3D Object Retrieval*. 2010, pp. 57–62.

[95] F. Tombari, S. Salti, and L. Di Stefano. “Unique Signatures of Histograms for Local Surface Description”. In: *European Conference On Computer Vision (ECCV)*. 2010, pp. 356–369.

[96] Federico Tombari and Luigi Di Stefano. “Object Recognition in 3D Scenes with Occlusions and Clutter by Hough Voting”.

In: *Proceedings of the 2010 Fourth Pacific-Rim Symposium on Image and Video Technology*. 2010, pp. 349–355.

[97] Federico Tombari, Samuele Salti, and Luigi DiStefano. “Performance Evaluation of 3D Keypoint Detectors”. In: *International Journal of Computer Vision* 102.1–3 (2013), pp. 198–220.

[98] Tomasz Trzcinski, CM Christoudias, Pascal Fua, and Vincent Lepetit. “Boosting Binary Keypoint Descriptors”. In: *Computer Vision and Pattern Recognition* (2013).

[99] Greg Turk and Marc Levoy. “Zippered polygon meshes from range images”. In: *SIGGRAPH* 94pp (1994), pp. 311–318.

[100] Kartik Venkataraman, Dan Lelescu, Jacques Duparr, Andrew McMahon, Gabriel Molina, Priyam Chatterjee, and Robert Mullis. “PiCam: an ultra-thin high performance monolithic camera array”. In: *Siggraph Asia* (2013).

[101] Junhao Xiao, Jianhua Zhang, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre. “Fast plane detection for SLAM from noisy range images in both structured and unstructured environments”. In: *IEEE International Conference on Mechatronics and Automation (ICMA)*. 2011, pp. 1768–1773.

[102] Chen Y. and G Medioni. “Object modelling by registration of multiple range images”. In: *Image and Vision Computing* (1992), pp. 2724–2729.

[103] Kuan-Ting Yu, Shih-Huan Tseng, and Li-Chen Fu. “Learning hierarchical representation with sparsity for RGB-D object recognition”. In: *International Conference on Intelligent Robots and Systems* (2012), pp. 3011–3016.

[104] A. Zaharescu, E. Boyer, and R. Horaud. “Keypoints and Local Descriptors of Scalar Functions on 2D Manifolds”. In: *International Journal of Computer Vision* (2012).

[105] Li Zhang, Brian Curless, and Steven M. Seitz. “Spacetime Stereo: Shape Recovery for Dynamic Scenes”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. Madison, WI, 2003.

[106] Yu Zhong. “Intrinsic Shape Signatures: A Shape Descriptor for 3D Object Recognition”. In: *International Conference on Computer Vision Workshops (3DRR)*. 2009.

Author's publications related to PhD topic

[107] Alioscia Petrelli and Luigi Di Stefano. "Pairwise Registration by Local Orientation Cues". In: *Computer Graphics Forum* 00.0 (2015), pp. 1–15.

[108] Alioscia Petrelli, Danilo Pau, and Luigi Di Stefano. "Analysis of Compact Features for RGB-D Visual Search". In: *International Conference on Image Analysis and Processing (ICIAP)*. Vol. 9280. 2015, pp. 14–24.

[109] Alioscia Petrelli, Danilo Pau, Emanuele Plebani, and Luigi Di Stefano. "RGB-D Visual Search with Compact Binary Codes". In: *International Conference on 3D Vision (3DV)*. 2015.

Author’s publications during the PhD course related to other topics

[110] Samuele Salti, Alioscia Petrelli, Federico Tombari, Nicola Fioraio, and Luigi Di Stefano. “A traffic sign detection pipeline based on interest region extraction”. In: *International Joint Conference on Neural Networks (IJCNN)*. 2013, pp. 1–7.

[111] Samuele Salti, Alioscia Petrelli, Federico Tombari, Nicola Fioraio, and Luigi Di Stefano. “Traffic sign detection via interest region extraction”. In: *Pattern Recognition* 48.4 (2015), pp. 1035–1045.

[112] Federico Tombari, Nicola Fioraio, Tommaso Cavallari, Samuele Salti, Alioscia Petrelli, and Luigi Di Stefano. “Automatic detection of pole-like structures in 3D urban environments”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2014, pp. 4922–4929.