AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa magisterska

Rozpoznawanie obiektów na obrazach RGB-D pod kątem zastosowań w robotyce

Object recognition in RGB-D images for robotic applications.

Autor:	*Jakub Olesiński*
Kierunek studiów:	*Automatyka i Robotyka*
Opiekun pracy:	*dr inż. Paweł Rotter*

Kraków, 2016

AGH

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

FACULTY OF ELECTRICAL ENGINEERING, AUTOMATICS, COMPUTER SCIENCE AND BIOMEDICAL ENGINEERING

DEPARTMENT OF AUTOMATICS AND BIOENGINEERING

Master of Engineering Thesis

Object recognition in RGB-D images for robotic applications.

Author:	*Jakub Olesiński*
Degree programme:	*Automatics and Robotics*
Supervisor:	*dr inż. Paweł Rotter*

Kraków, 2016

Contents

1. Introduction

The aim of this work is to survey modern RGB-D image processing algorithms for model-based object recognition. Analysis of each method is focused on their usability in time and resource constrained robotic environment. Based on provided performance evaluation, selected methods are used to develop a complete, applicable in robotics, object recognition system.

The first chapter provides project background. A formal problem statement is included and followed by categorized solution proposals found in literature, outlining the scientific context of this work. The RGB-D imaging basics are subsequently introduced, together with software tools and test environment utilized throughout the project.

Second chapter introduces preprocessing techniques. Firstly, different data representation and conversion methods are discussed. Basic depth processing operations are introduced afterwards, including spatial transformations, neighbourhood selection and normal estimation. Different noise types encountered in RGB-D images and their corresponding filtering methods are further discussed.

Model fitting with sparse feature matching algorithms is presented in chapter three. Selected keypoint detectors and descriptors of both color and shape modalities are compared and an efficient matching technique is provided. Further, correspondence clustering and pose estimation methods are evaluated.

The following chapter is about

1.1. Problem statement

Given an input image and an object of interest, the task of object recognition is to provide answer to the following questions:

– Is the queried object *present* in the image?

– If it is, then what is the *pose* of the object?

If the object has the properties of a rigid body, its pose is given by the linear and angular position.

Object recognition is a process that comprises determination of both object instance presence and its pose in a given scene image.

Robotics: why object recognition Object recognition: why RGB-D

1.2. Related work

Survey of local methods [1].

1.3. Software tools

Throughout this work, most of the software was developed with an extensive use of the Point Cloud Library (PCL)[2], which is an open source C++ library for three dimensional image processing. Since 2011, it is developed by a large scientific community, maintained by the Open Perception foundation and delivered under BSD license. PCL provides implementations of a multitude of novel algorithms for 3D filtering, feature estimation, segmentation, registration and model fitting, together with tools for visualization and camera interfacing.

The Open Computer Vision (OpenCV)[3] library was utilized as another source of implementation for the analysed algorithms. OpenCV, which is a more mature project, developed since 1999 is likewise delivered under open source BSD license. While PCL is mainly focused on volumetric data analysis, OpenCV provides a wide array of functions for 2D intensity and color image processing.

1.4. Test environment

Dataset Willow

Recognition system was validated against sample dataset from XXX, which provides full object models together with real world scene scans and ground truth position annotations. To

Dataset own Xtion sensor

Timing perf hardware

Testing of implemented algorithms was done on two hardware platforms: an embedded computing board NVidia Jetson TK1 and personal laptop Lenovo Y50-70. Parameters of both computers are summarized in table [x].

	NVidia Jetson TK1	Lenovo Y50-70
CPU:	ARM Cortex-A15 2.32GHz x4	Intel Core i7-4720HQ 2.60GHz x4
GPU:	NVIDIA Kepler GK20a	NVidia GeForce GTX 960M
	with 192 SM3.2 CUDA cores	with 640 SM5 CUDA Maxwell cores
	(upto 326 GFLOPS)	(upto xx GFLOPS)
RAM:	2GB DDR3L 933MHz 64 bit	16GB DDR3L 1600MHz (4GB GPU 2.5Mhz 128bit)

Table 1.1: Test hardware specification

2. Pre-processing

This chapter describes basic RGB-D image processing tool set, that prepares an input image for extracting relevant information during object recognition.

2.1. Representation

Depth and colour information of the environment can be represented in computer memory by the means of various data structures, which differ in their complexity and applicability. In case of typical, low-cost RGB-D cameras, the raw data that is retrieved from the device firmware and operating system drivers come in a form of two images, one from a conventional colour camera and the other from the depth sensor, as depicted on Figure x.

Figure XXX - drawing of camera with 2 raycasts, 2 images

Data retrieved from a depth sensor have a form of one channel, two dimensional image. Each pixel contains measurement of the distance between depth camera reference frame origin and a corresponding point on the reflected surface. In case of Asus Xtion sensor, this data is stored as 32 bit floating point values. It is important to note, that depending on the acquisition method, not all of the surface point distances are measured. For example, if the surface is translucent, most sensors will fail to retrieve proper distance. Such situation is reflected in data by the use of *Not a Number (NaN)* pixel values.

Due to the displacement between color and depth sensor [reference], RGB-D cameras require a calibration step to align the measurements. For this purpose, open source packages such as [XXX] can be used. Finally, successful RGB-D data alignment yields a four channel digital image, which can be defined more formally as:

$$I : \{1, \ldots, M\} \times \{1, \ldots, N\} \to [0,1]^4, \qquad I(u,v) = \begin{bmatrix} I_R(u,v) \\ I_G(u,v) \\ I_B(u,v) \\ I_D(u,v) \end{bmatrix}, \tag{2.1}$$

where:

- $M \in \mathbb{N}, N \in \mathbb{N}$ - row and column image size,
- $I_R(u,v), I_G(u,v), I_B(u,v)$ - red, green and blue colour channel intensities,
- $I_D(u,v)$ - depth channel intensity

RGB-D image formulation 2.1 is mostly suitable for per channel, classic two-dimensional image processing, such as applying morphological operators or linear filters [XXXImProc]. It does, however, limit the three dimensional surface information to a single viewpoint, which is undesirable for some applications in robotics, especially for object modelling and environment mapping. A generalized data representation that overcomes this limitation is called a *point cloud*. It can be defined as a point set $C \subset \mathbb{R}^N$ of N-dimensional vectors. Depending on the usage, the points can represent different image modalities and features, such as point location in some reference frame, colour, surface normal vector or more advanced descriptors (see chap. 3). Direct conversion from a depth map to point cloud, requires projective transformation [XXX] of the depth channel to render Cartesian x, y, z point coordinates in camera reference frame. In this manner, multiple RGB-D image frames, taken from different viewpoints can be stored within a single point cloud, by registering their points to a common coordinate system.

There are other possible representations of depth data. In contrast to point clouds, *Patch volumes* [XX] and *Signed distance functions* are dense representations of the three-dimensional environment, where each point represents distance to the nearest surface. Patch volumes use BLABLABLA. OctoMaps use BLABLABLA. Those representations, however, are mainly used for environment mapping and SLAM applications. This work will further focus on point clouds, as they are the most covered representation for object recognition. ??

2.2. Neighbourhood

Similarly to classical image processing, a wide range of point cloud processing algorithms rely on the notion of point neighbourhood. Given a metric function d and a query point p_q in the point cloud P, two commonly used types of neighbourhoods can be defined. A *r-neighbourhood* is a set composed of all the points $p_i \in P$ that lie within a sphere of a radius r and a center in p_q, thus satisfying the condition

$$d(p_q, p_i) \leq r. \tag{2.2}$$

The other type is *k-neighbourhood*, a set of k nearest points in the sense of given metric. Proper determination of neighbourhood size parameters, either r or k correspondingly, is crucial in further analysis. Too small values do not provide enough information about the surface. Too large, on the other hand, will average the surface and skip small details. Complexity and efficiency of neighbourhood search depends on different internal data representation in a point cloud.

From computational point of view, a point cloud that is directly converted from RGB-D image can be internally stored in a two dimensional array of points, with elements corresponding to image pixels. Point clouds with such data arrangement are commonly referred to as *organized*. Image-like ordering is beneficial, because it reflects the spatial distribution of points directly into memory.

Also known as support. Organized neighbourhoods. KD-tree. Flann. *"scale" corresponds to the "support radius" (TombariA et al, 2013)*

2.3. Surface normals

Many of the algorithms used further during the analysis of a point cloud base on the notion of a surface normal, known from the 3D geometry. These vectors are also widely used for shading in 3D computer graphics and a variety of methods have already been developed to solve the surface normal estimation problem. One of the simplest uses the least-squares plane fitting to estimate the normal to a plane tangent to the surface, which approximates the desired vector [4]. More specifically, for a given point on the surface, it's k-neighbourhood centroid $\bar{p}$ is calculated as:

$$\bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \tag{2.3}$$

The tangent plane is thereafter defined by the centroid $\bar{p}$ and the sought normal vector $\vec{n}$. The latter is computed by minimizing the total distance from every k-neighbour p_i to the tangent plane, given by $\sum\limits_{i=1}^{k}(p_i - \bar{p}) \cdot \vec{n}$. The minimization problem can be solved by utilizing the covariance matrix $C \in \mathbb{R}^{3x3}$, given by:

$$C = \frac{1}{k-1} \sum_{i=1}^{k} (p_i - \bar{p}) \cdot (p_i - \bar{p})^{\intercal} \tag{2.4}$$

The covariance matrix C is symmetric, positive semi-definite and possess three real eigenvalues $\lambda_j \geq 0, i = 1, 2, 3$. The eigenvector corresponding to the smallest eigenvalue is an approximation of the desired normal vector $\vec{n}$, disregarding the sign. Furthermore, if the viewpoint v_p is known, the normal $\vec{n}$ has to be oriented towards v_p, which means that it has to satisfy the condition:

$$\vec{n} \cdot (v_p - p_i) > 0 \tag{2.5}$$

Processing time. Cuda implementation. Complete solution used [5].

Example PCD with normals:

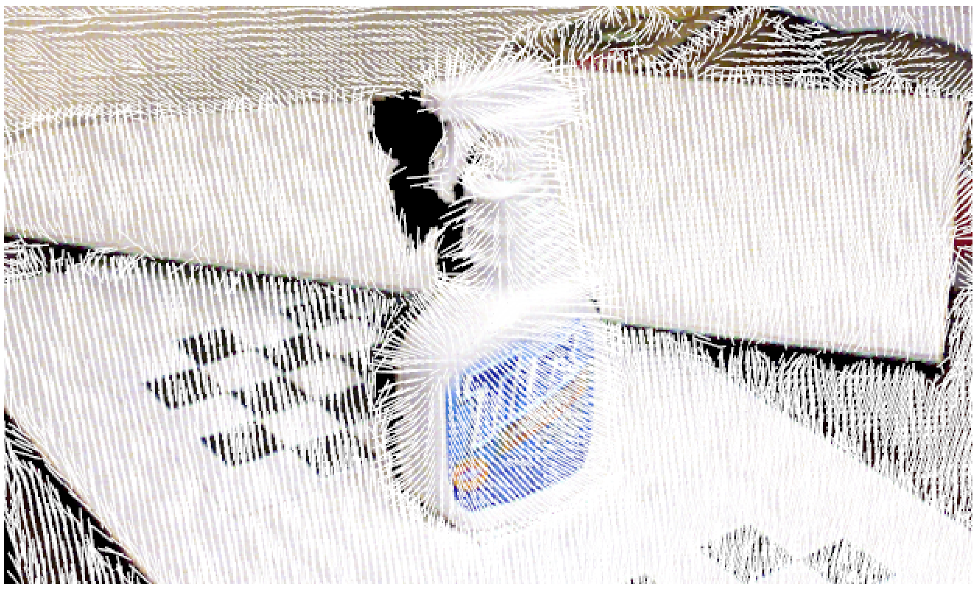
Figure 2.1: Normals

2.4. Noise filtering

Types of noise, frequency response, low pass filtering, nonlinear filtering

Boundary estimation

Occlusion and clutter from Guo

One of the most important stages of data preprocessing is the filtration. Probably the most basic point cloud filter is the pass-through filter, which reject all points outside a given range along a specified dimension. This procedure allows to focus the analysis process on the region of interest, i.e. the reachable workspace of a manipulator. Apart from limiting the cloud dimensions, the number of data points can be also reduced by downsampling. The voxel grid filter is typically used for this purpose. This filter creates a three dimensional regular grid over the input point cloud data and then, in each voxel, approximates all the present data points with their centroid. Such reduction is particularly useful when large point cloud datasets have to be processed online with limited computing resources. Finally, the filtration process has to cope with numerous measurement errors present in the raw data acquired from the 3D camera. Such measurement noise manifests itself in the form of sparse outliers, which corrupt the results of further processing, i.e. the surface normals estimation. The impact of those irregularities can by reduced by applying an outlier removal filter. The simplest form of such filter rejects all the data points which does not have enough neighbours within a specified radius. A more refined version is based on the neighbouring points distance distribution. Firstly, the mean distance from each point to all its neighbours is calculated. Next, based on the assumption that the resulting distribution is Gaussian, all the points whose mean distances lay outside of an interval defined by the global distances mean and their standard deviation are rejected from the dataset. The effects of statistical outlier removal are presented in the Figure **??**.

2.5. Segmentation

The point cloud segmentation or cluster extraction is the process of grouping together data points that meet some common condition of similarity. Ideally, each of such segmented clusters represents a different object in the scene. A popular segmentation method is based on the euclidean distance. In this approach, two points are considered to belong to the same cluster, if their relative distance is less than a selected threshold. This process is similar to the flood-fill algorithm, popular in the 2D image processing. There are also other varieties of segmentation, i.e. based on the smoothness constraint or colour similarity.

3. Sparse feature matching

Solutions to problem stated in section 1.1 are commonly divided [?] into feature-based and template-based. The former approach, also referred to as local matching, is focused on comparing point neighbourhoods between the scene and model datasets.

The solution in such methods is composed of several steps. Firstly, an input subset with elements of rich and distinguishable neighbourhood information is selected. Points that belong to this subset are referred to as *keypoints*. For each keypoint, its neighbourhood information is expressed in the form of a *descriptor* vector. By the means of selected metric, commonly the L_2 norm, descriptors are further compared between the scene and model datasets, to find *correspondences* with minimal distance. Such point pairs are then grouped to share similar geometric constrains and finally, the largest clusters are used to calculate affine transformation, which renders the initial problem solution.

The inherent locality of feature matching methods has direct implications on the solution performance. Such methods are by design robust to occlusions [?]. Each point is processed independently, which enables data parallelization to boost time performance. Conversely, key-point identification requires costly analysis of the whole input dataset, thus a balance between the recognition performance and time effectiveness is required [?] for real-time applications.

3.1. Shape description

There is a multitude of proposals for shape key-point detectors existing in literature. An overview and performance evaluation of the most popular methods can be found in [6] and [7]. From both evaluations, the *Intrinsic Shape Signatures* (ISS) [8] detecor is worth particular attention. [6] states that ISS, as a fixed-scale detector, copes well with full three-dimensional models and provides a proper balance between the repeatability rate and time efficiency. In [7], the ISS is evaluated with the best repeatability rate among detector implementations available in PCL. The ISS introduces a saliency measure, defined by the smallest eigenvalue of the neighbourhood scatter matrix. For a given point p and its neighbourhood N, the scatter matrix Σ is given by

$$\Sigma(p, N) = \frac{1}{|N|} \sum_{q \in N} (q - \mu_p)(q - \mu_p)^T, \ \mu_p = \frac{1}{|N|} \sum_{q \in N} q \tag{3.1}$$

By denoting the eigenvalues of $\Sigma(p, N)$ as $\lambda_1 > \lambda_2 > \lambda_3$, the ISS detector classifies p as a keypoint, if the following condition is satisfied

$$\frac{\lambda_2}{\lambda_1} < \epsilon_1 \ \wedge \ \frac{\lambda_3}{\lambda_2} < \epsilon_2 \ \wedge \ \lambda_3 > \epsilon_3. \tag{3.2}$$

Thresholds ϵ_1 and ϵ_2 are meant to provide sufficient difference between variations along each principal direction, which aids in estimation of a repeatable reference frames for further description stages. The third threshold ϵ_3 ensures that the variations are large enough to consider the point as interesting. A further improvement to the ISS detection criteria is to apply non-maximum suppression over the saliency measure computed at each point. With this edge thinning technique, a point is classified as a keypoint, if it has the largest l_3 over its neighbourhood. An example of keypoints detected with ISS is presented on figure 3.1. The object (cleaning spray in the center of the figure) has marked points in areas of rich shape information, like the bottleneck or the nozzle. It is also visible, that ISS classifies wrong points in two cases. Firstly, when a point lies on a visibility boundary (i.e. the edges of flat surfaces in the figure), thus it should be extended with boundary detection as described in xx. Secondly, when the sensor noise is high (distant wall flat surface in upper left edge of the figure). This issue can be handled by ignoring points above certain z-distance from the sensor or by adaptatively selecting neighbourhood radius for saliency measure.

The overall time complexity of ISS classification is at the order of $O(nk)$, where n is the image size and k is the max neighbourhood size. This translates into average execution times on target platforms, provided in table 3.1.

the ISS method with boundary point re- moval (ISS-BR) was used to detect the keypoints in the scene and the models. As reported in (Salti et al, 2012), ISS achieves the best performance compared to other keypoint detectors when used in conjunction with fea- ture descriptors.

	NVidia Jetson TK1	Lenovo Y50-70
PCL ISS	4s	4s
PCL ISSBD	4s	4s
CUDA ISS	4s	4s
CUDA ISSBD	4s	4s

Table 3.1: ISS detector time performance statistics.

It is important to note, that in practical applications the keypoint detection step is often replaced by volumetric down sampling. About voxel grid. Model is densely grained. This avoids

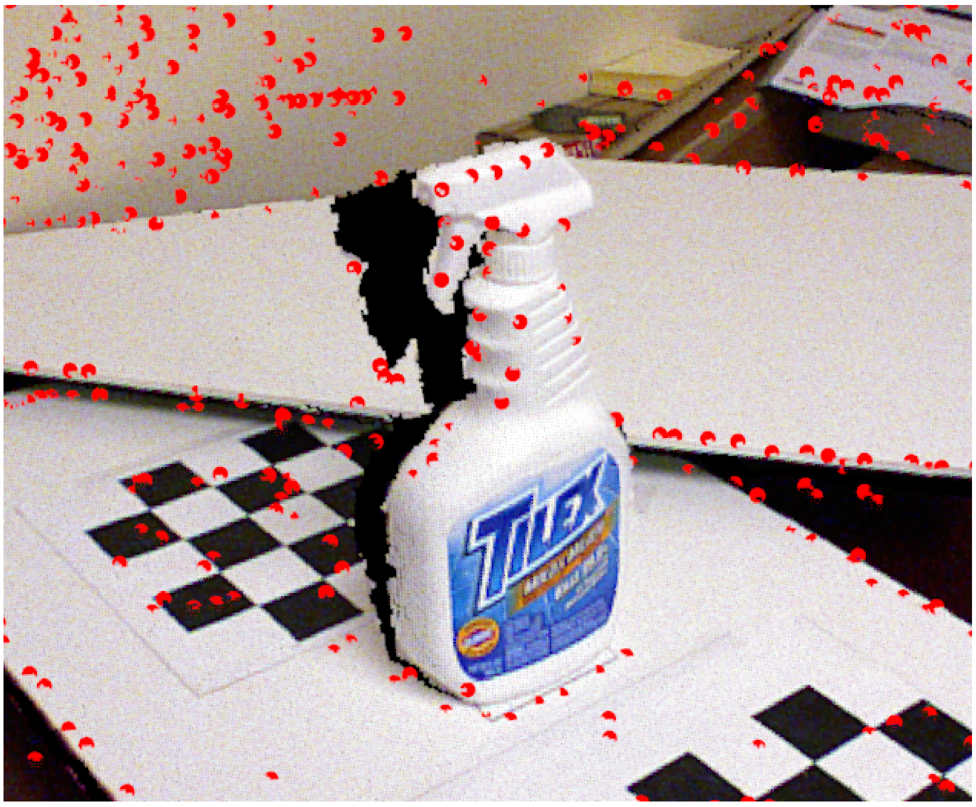

Figure 3.1: ISS keypoints (marked with red) computed with $\epsilon_1 = \epsilon_2 = 0.75$ and $\epsilon_3 = 0.0005$

computational costs of keypoint detection, but increases the complexity of further matching and clustering stages. How to choose this tradeoff.

Similarly to keypoint detection methods, the literature also abounds with different keypoint description proposals. A comprehensive comparison of the most common methods can be found in [9]. The *Fast Point Feature Histograms* (FPFH) [10, 11] and *Signature of Histograms of OrienTations* (SHOT) [12] descriptors are particularly interesting for the purposes of this work, given the real-time execution assumptions.

FPFH represents the relative orientation of normal vectors between the query point p and each of its neighbours $q \in N$. For each point pair (p, q), a new coordinate frame u, v, w originating at p is constructed as

$$u = n,\ v = n \times \frac{p-q}{\|p-q\|_2},\ w = u \times v, \tag{3.3}$$

where n is the normal vector at p and $d = \|p-q\|_2$ is the Euclidean distance. Using this reference frame, the difference between normals at p and q is expressed by the angular features

$$\alpha = v \cdot n,\ \phi = u \cdot \frac{(p-q)}{\|p-q\|_2},\ \theta = \arctan(w \cdot n, u \cdot n). \tag{3.4}$$

The angular features are further binned into a histogram H, typically dividing them into 5 subdivisions, which leads to $3^5 = 125$ bins. Such histograms are computed every point in the input point cloud. Finally, the FPFH descriptor is formed as

$$FPFH(p) = H(p) + \frac{1}{|N(p)|} \sum_{q \in N(p)} \frac{1}{\|p-q\|_2} H(q). \tag{3.5}$$

The FPFH descriptor computational complexity is $O(nk)$.

The SHOT proposal emphasises the importance of defining a unique and unambiguous local coordinate system, namely the *Local Reference Frame* (LRF), as a basis for a comparable descriptor construction. A modified version of support covariance matrix (as defined in eq. 3.1) is introduced, where each neighbour is weighted by its proximity to the query point:

$$M(p) = \frac{1}{\sum\limits_{q \in N(p)} (r - \|p-q\|_2)} \sum_{q \in N(p)} (r - \|p-q\|_2)(p-q)(p-q)^T. \tag{3.6}$$

The authors determine the LRF coordinate system axes, x, y and z with the corresponding eigenvectors of M, e_x, e_y and e_z in decreasing eigenvalue order. The sign of the axes is disambiguated by aligning the orientation of x and z axes with the majority of vectors they are representing. Concretely, the signed x axis is given by the formula

$$x(p) = \begin{cases} e_x, & \text{if } \sum\limits_{q \in N(p)} \text{sign}\left((p-q)^T \cdot x\right) \geq 0 \\ -e_x, & \text{otherwise} \end{cases}. \tag{3.7}$$

The z axis sign is determined analogically and y is a cross product of the former, $y = z \times x$. The authors have proven experimentally a high repeatability of such LRF, even in presence of noise and clutter. To form a descriptor, support is further divided with a spherical grid of 2 radial, 2 elevation and 8 azimuth divisions, as depicted on figure 3.2 (for better visibility, only half of azmiuth partitions are shown), resulting in 32 cells total. For each cell, the cosine of the angles θ_q between the normal at the query point n_p and each normal at cell points n_q are binned to form a histogram. Uniform binning of the cosine function provides robustness to LRF perturbations, as it leads to coarser angle binning for normals close to the z axis. Another problem, called *boundary effect*, arises due to the rough division of the support volume. Depending on the inaccuracies in LRF estimation, points that are close to the boundaries between neighbouring cells are not repeatably accounted to the same histogram. To avoid this problem, the authors propose to interpolate the neighbouring bins to spread the counts among them. After this step, the local histograms are being normalized by their L_2 norms, to account for varying point densities. Finally, the SHOT descriptor is composed by concatenating the local histograms, each with typically of 11 bins, resulting in 352 element descriptor.

Extension with color. [13]. Performance extension in binarized form [14].

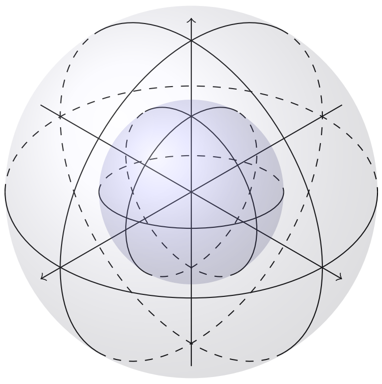

Figure 3.2: SHOT descriptor spherical grid

3.2. Texture description

ORB [15]

3.3. Clustering

Matching. brute force, geometric consistency, hough

3.4. Alignment

umeyama, ransac

4. Dense template matching

4.1. Ransac

4.2. Linemod

4.3. Neural networks

Convolutional neural networks

5. Post-processing

5.1. Pose refinement

icp

5.2. Hypothesis verification

ghv, ghv from v4r
multi pipeline

6. Applications

6.1. Scenario 1

something simple with recognition
simulation of something simple

6.2. Scenario 2

something simple with classification
simulation of something simple

6.3. Scenario 3

Robocup@Work
simulation of Robocup@Work?

6.4. Scenario 4

APC?
Synthetic dataset from APC gazebo. simulation of APC?

Summary

Bibliography

[1] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, “3d object recognition in cluttered scenes with local surface features: a survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, 2014.

[2] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[3] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.

[4] R. B. Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[5] O. K. Smith, “Eigenvalues of a symmetric 3×3 matrix,” *Communications of the ACM*, vol. 4, no. 4, p. 168, 1961.

[6] F. Tombari, S. Salti, and L. Di Stefano, “Performance evaluation of 3d keypoint detectors,” *International Journal of Computer Vision*, vol. 102, no. 1-3, pp. 198–220, 2013.

[7] S. Filipe and L. A. Alexandre, “A comparative evaluation of 3d keypoint detectors in a rgb-d object dataset,” in *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, vol. 1, pp. 476–483, IEEE, 2014.

[8] Y. Zhong, “Intrinsic shape signatures: A shape descriptor for 3d object recognition,” in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 689–696, IEEE, 2009.

[9] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, and N. M. Kwok, “A comprehensive performance evaluation of 3d local feature descriptors,” *International Journal of Computer Vision*, vol. 116, no. 1, pp. 66–89, 2016.

[10] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pp. 3212–3217, IEEE, 2009.

[11] R. B. Rusu, A. Holzbach, N. Blodow, and M. Beetz, “Fast geometric point labeling using conditional random fields,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7–12, IEEE, 2009.

[12] F. Tombari, S. Salti, and L. Di Stefano, “Unique signatures of histograms for local surface description,” in *European conference on computer vision*, pp. 356–369, Springer, 2010.

[13] F. Tombari, S. Salti, and L. Di Stefano, “A combined texture-shape descriptor for enhanced 3d feature matching,” in *2011 18th IEEE International Conference on Image Processing*, pp. 809–812, IEEE, 2011.

[14] S. M. Prakhya, B. Liu, and W. Lin, “B-shot: A binary feature descriptor for fast and efficient keypoint matching on 3d point clouds,” in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 1929–1934, IEEE, 2015.

[15] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *International Conference on Computer Vision*, (Barcelona), 11/2011 2011.