AGH

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

Many thanks to all the people and beware global warming

Contents

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 tests, selected methods will be used to develop a complete, applicable in robotics,object recognition system.

The first chapter provides project background. RGB-D imaging basics are introduced and followed by software tools and testing environment used throughout the project. Subsequently, scientific context of the work is settled.

Second chapter introduces preprocessing techniques. Firstly, different data representations and conversion methods are discussed. Afterwards, basic operations are introduced, including spatial transformations, neighbourhood calculation and normal estimation. Efficient noise filtering, data segmentation and keypoint extraction methods finalize this chapter.

Model fitting algorithms are presented in chapter three. Performance of data alignment and correspondence matching methods is tested together with different descriptor approaches. Statistical methods of result verification are finally presented and used to test the complete object recognition system.

Chapter four generalizes models from previously developed approach into categories. A neural network approach is presented as a data classifier. Convolutional neural network architecture is proposed and its implementation, training and performance tests are provided.

The last chapter describes several application scenarios of the developed system. Here should be placed one sentence per scenario. Here should be placed one sentence per scenario. Here should be placed one sentence per scenario. Each tested against different noise conditions.

1. Background

1.1. Motivation

Read and merge few article intros

The field of RGB-D image processing was developing slowly, until the release of Microsoft's Kinect in 2010. This low-cost camera gained significant scientific interest because of the potential it brought into the areas of virtual reality.

1.2. Related work

1.3. Software tools

Throughout this work, all of the software was developed with an extensive use of the Point Cloud Library (PCL)[1], which is an open source C++ library for 2D/3D image and point cloud processing. 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. Since 2011, it is developed by a large scientific community and maintained by the Open Perception foundation. PCL is an 3D equivalent of the OpenCV, the largest computer vision software library [ref]. Methods developed under this project were also aided by OpenCV tools, in places where classic 2D image processing was sufficient. Most promising algorithms analysed in this work was parallelized with CUDA GPU computing framework to optimize processing time.

1.4. Test environment

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

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

Print stanford bunny ?

Washington V2 Database - how to read it into pcl

Synthetic dataset from APC gazebo.

YCB Benchmarks

2. Preprocessing

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 3.3). 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

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 array elements corresponding to image pixels. Point clouds with such data arrangement are commonly referred to as *organized*. Image-like ordering is beneficial in

2.3. Transformations

segmentation

2.4. Noise filtering

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

2.5. Segmentation

segmentation

2.6. Keypoints

segmentation

3. Recognition

keypoints

3.1. Alignment

alignment

3.2. Matching

matching

3.3. Descriptors

descriptors

3.4. Verification

verification

4. Classification

4.1. Neural network

convolutional neural network

4.2. Training

training

4.3. Results

descriptors

5. Applications

5.1. Scenario 1

something simple with recognition
simulation of something simple

5.2. Scenario 2

something simple with classification
simulation of something simple

5.3. Scenario 3

Robocup@Work
simulation of Robocup@Work?

5.4. Scenario 4

APC?
simulation of APC?

Summary

Bibliography

[1] 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.