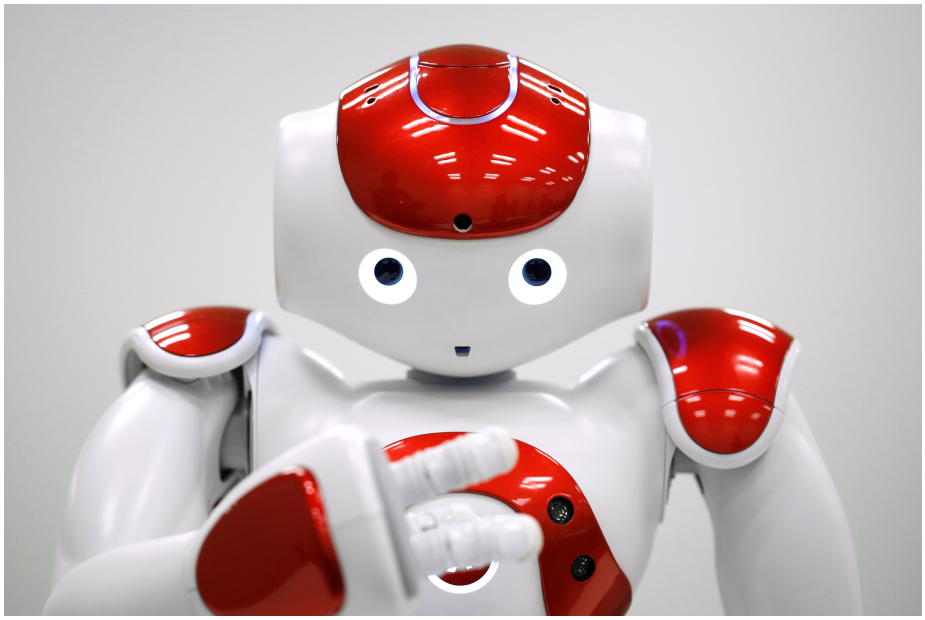


Robotic Project 3D localization and classification of trash objects for mobile cleaning robots

Mohammad Alkhatib And Vishnu Veilu Muthu
DarkROS
University of Burgundy

January 2, 2017

Contents

1 Introduction **4**

2 3D Computer Vision for Mobile Robotics techniques in ROS **5**

3 Cleaning Robot **6**

4 Mapping, Navigation and Detection **7**
4.1 Mapping and Navigation 7
4.2 Object Detection and classification 8

5 Packages **9**
5.1 Mapping and Navigation 9
5.1.1 Rtabmap Package 9
5.1.2 Frontier Exploration Package 9
5.2 Detection and Classification 10
5.2.1 Tabletop Package 10
5.2.2 AR tag tracker Package 10

6 Installation **11**
6.1 Simulation 11
6.2 Real Turtlebot 11
6.3 Rtabmap-ros (3D reconstruction of area) 11
6.4 Frontier-explorer(Autonomous navigation) 11
6.5 Object Recognition Packages 12
6.6 AR tag tracker Package 12
6.7 Speech Package 13

7 Implementation **14**
7.1 Mapping and Navigation 14
7.1.1 Simulation 14
7.1.2 Real Time 14
7.2 Detection And Classification 14
7.2.1 Adding objects to the Database 14
7.2.2 Training the Database 14
7.2.3 Test 3D object Detection And Classification 14
7.2.4 Test The Tag tracker 14
7.3 Running the Project 15

8 Results **16**

9 Project Management **17**

10 Conclusions and future work **18**

11 References **19**

List of Figures

1 Cleaning robot and Library assistant robot 4
2 Turtlebot 2 and ROS 5
3 Cleaning Robot 6
4 Mapping and Navigation 1 7
5 Mapping and Navigation 2 7
6 Objects to detect 8
7 Objects Detection 8
8 Tags Detection 8
9 Rtabmap with frontier explorer 16
10 3D Object detection and recognition 16
11 AR tag tracker 16
12 Project Management 17
13 Future work 18

1 Introduction

The robotics community has made impressive progress in recent years. Reliable and inexpensive robot hardware from land-based mobile robots, to quadrature helicopters, to humanoids is more widely available than ever before. Perhaps even more impressively, the community has also developed algorithms that help those robots run with increasing levels of autonomy.

In spite of (or, some might argue, because of) this rapid progress, robots do still present some significant challenges for software developers. This book introduces a software platform called Robot Operating System, or ROS,that is intended to ease some of these difficulties.

The aim of this Project is to introduce our robotic project which allowed us to understand the inner working of ROS and to gain an in depth knowledge of 3D computer vision for mobile robotics, robot navigation and control using ROS. Also to understand the power usage of ROS and its practical implementation on a turtlebot.

In this report, we will introduce a technique of 3D Computer Vision for Mobile Robotics implemented in ROS. Our idea is a cleaning robot, this project is focused on the believe that between 25 and 50 years every home will have a cleaning robot. In this project we used ROS and image processing techniques for mapping, objects recognition and visual servoing.

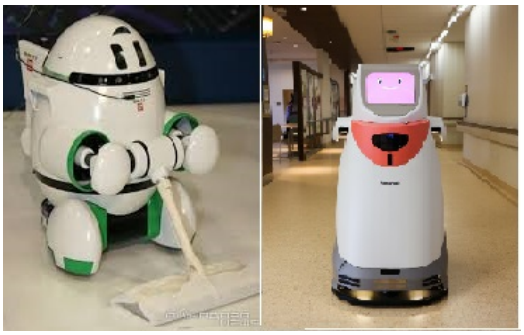


Figure 1: Cleaning robot and Library assistant robot

2 3D Computer Vision for Mobile Robotics techniques in ROS

Vision is a very important sense for humans. Robots also need to be aware of their environment for working in it. This is even more important for mobile robots that operate in unknown environments that are not prepared for robots. In this section, we focus on 3D Computer Vision for Mobile Robotics techniques implemented in ROS.

Components for detection and recognition of objects and their features enable robot to perceive and interact with the environments.

is SLAM using ROS with the help of the gmapping package which is a third party package by OpenSlam. This provides a depth based SLAM using Rao-Blackwellized particle filter.SLAM gives autonomous system for efficient creation of photo realistic 3D models of indoor and outdoor environments.

Most modern high definition 3D scans are done using RGB-D cameras. In recent years, with the introduction of the Microsoft Kinect, 3D point clouds became a hot topic of research.For 3D reconstruction, there are many techniques to do that, and we used this technique.

3D point clouds are also at the core of many robotics applications, ranging from Simultaneous Localization and Mapping (SLAM) and artificial intelligence to augmented reality.

For 3D recognition, we choose feature as they would be unique from on object to another which will allow us to recognize objects easily. In ROS there are several recognition methods:a textured object detection (TOD) pipeline using a mesh database.

In the following section we will propose our project in details, from mapping, navigation till objects recognition.

Figure 2: Turtlebot 2 and ROS

3 Cleaning Robot

The research and development of an autonomous mobile robot prototype able to vacuum cleaning a room or even an entire house is not a trivial challenge. In order to tackle such a task, so that it could be completed in six weeks (the duration of the course), some simplifications and assumptions will be made to the designers initial idea of an "ideal" autonomous mobile cleaner robot. Probably the decision that the most affects the robot complexity is the ability of mapping the environment and recognize objects.

With the aim of keeping our robot as simple as possible, while able to perform the initial goals, i.e. an autonomous mobile cleaner robot able to randomly navigate through a room or a house with the minimum human assistance, the following specifications were found, obstacle avoidance, floor detection,collision detection, and object recognition.

The main objective of the project is to separate the trash from a road side environment or an after party site. The process is designed that the robot is programmed to navigate and search the area for path and detecting object. Later, after detecting trash they are classified into three sections like paper cup, can or bottle.

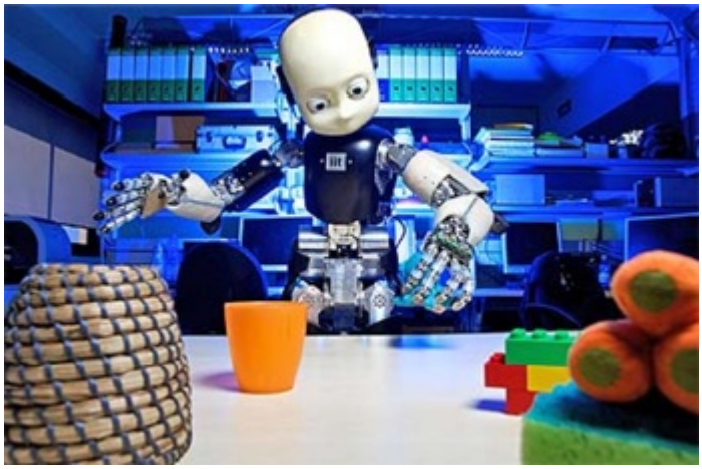


Figure 3: Cleaning Robot

4 Mapping, Navigation and Detection

4.1 Mapping and Navigation

The Navigation of the robot is designed to work with two environments.

The first part is designed with the known area mapping environment where a saved map for a particular boundary of area is used. The second is done with unknown area in the real world. This process is a collecting of mapping data with the surveillance of area with the increase of radius. The package (rtabmap ros) is used as a reference to design the process. And we used (Frontier explorer)package to explore the unseen area inside the boundary.

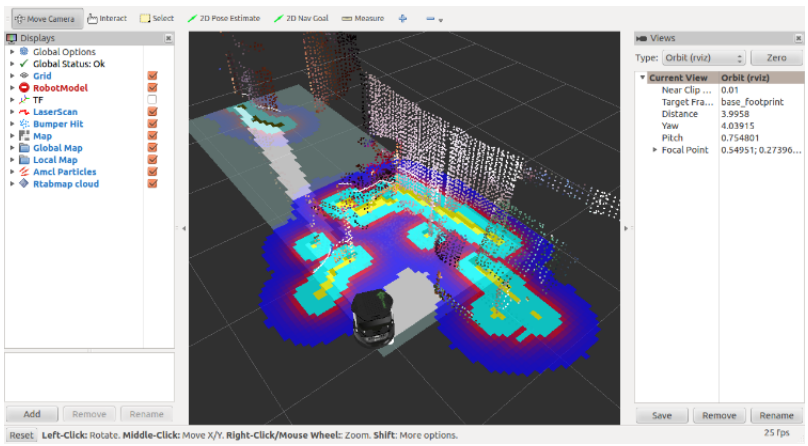


Figure 4: Mapping and Navigation 1

Figure 5: Mapping and Navigation 2

4.2 Object Detection and classification

The process is designed with scanning of the 3D data from the object, then building the mesh database. after detecting the object we classify it to cans, paper cup and bottles.

The package (tabletop object recognition) is used as a reference to design the process. And we used (AR tag tracker) package to detect and track tags.

Figure 6: Objects to detect

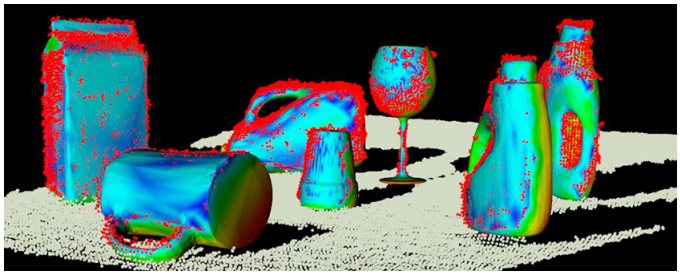


Figure 7: Objects Detection

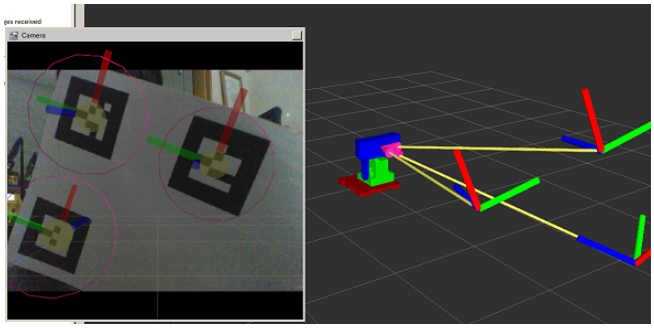


Figure 8: Tags Detection

5 Packages

In this section we will take in detail about the project used packages, and the advantages and disadvantages for each package.

5.1 Mapping and Navigation

The package (rtabmap ros) is used as a reference to design the process, and we used (Frontier explorer)package to explore the unseen area inside the boundary.

5.1.1 Rtabmap Package

RTAB-Map (Real-Time Appearance-Based Mapping) is a RGB-D Graph-Based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected. RTAB-Map can be used alone with a hand-held Kinect or stereo camera for 6DoF RGB-D mapping, or on a robot equipped with a laser rangefinder for 3DoF mapping.

5.1.2 Frontier Exploration Package

Frontier exploration is to perform autonomous planning movement, and exploration with simultaneous localization and mapping (SLAM), and used to explore the unseen area inside the boundary.

The explorer package utilizes frontier based exploration for multi-robot systems. Beside frontier detection, coordinated and uncoordinated exploration strategies are available to select goal points. Coordinated exploration enhances robot distribution and reduces redundancy in exploration reducing exploration time.

The explorer package comprises multiple functionalities to perform frontier based exploration. Frontier detection is utilized on local and global costmaps to further select navigation goals in the environment to proceed with exploration. The assignment of robots to goals is accomplished according to coordinated and uncoordinated exploration strategies being applied for the distributed multi-robot system.

5.2 Detection and Classification

The package (tabletop object recognition) is used as a reference to design and detect a mesh objects, we scan and detect the small objects. And we used (AR tag tracker) package to detect and track tags which we used due to the large size of the trash can.

5.2.1 Tabletop Package

There are two main components of object perception:

object segmentation: which part of the sensed data corresponds to the object we want to pick up?. And object recognition: which object is it?

Our manipulation pipeline always requires segmentation. If recognition is performed and is successful, this further informs the grasp point selection mechanism. If not, we can still grasp the unknown object based only on perceived data.

To perform these tasks we rely the following assumptions:
1- The objects are resting on a table, which is the dominant plane in the scene
2- The minimum distance between two objects exceeds a given threshold (3cm in our demo)
3- Object recognition can only handle 2 degrees of freedom: translation along the X and Y axes (with the Z axis assumed to be pointing "up" relative to the table). Therefore, in order to recognize an object, it must be rotational symmetric and it must have a known orientation, such as a glass or a bowl sitting "upright" on the table.

The sensor data that we use consists of a point cloud from the narrow stereo or Kinect cameras. We perform the following steps:
1- Segmentation : the table is detected by finding the dominant plane in the point cloud using RANSAC, and points above the table are considered to belong to graspable objects. We use a clustering algorithm to identify individual objects. We refer to the points corresponding to an individual object as clusters.
2- Recognition : for each cluster, a simple iterative fitting technique (a distant cousin of ICP) is used to see how well it corresponds to each mesh in our database of models. If a good fit is found, the database id of the model is returned along with the cluster.

The output from this components consists of the location of the table, the identified point clusters, and the corresponding database object id and fitting pose for those clusters found to be similar to database objects.

5.2.2 AR tag tracker Package

Due to the size of the trash can, we used tag detection package (AR tag tracker) .

AR tag tracker has three main functionalities:
1- Generating AR tags of varying size, resolution, and data/ID encoding
2- Identifying and tracking the pose of individual AR tags, optionally integrating kinect depth data (when a kinect is available) for better pose estimates.
3- Identifying and tracking the pose of "bundles" consisting of multiple tags. This allows for more stable pose estimates, robustness to occlusions, and tracking of multi-sided objects.

6 Installation

All about this project is in github repositories DarkROS.

To run this project, please follow the following steps.

Please install all the Packages and tools for the program to execute correctly.

6.1 Simulation

Used to simulate the packages of ROS and DarkROS using Gazebo,

Pre-install:
sudo apt-get install ros-indigo-gazebo-ros-pkgs ros-indigo-gazebo-ros-control
sudo apt-get install ros-indigo-turtlebot-simulator

Weblink:
`http://gazebosim.org/tutorials?tut=ros_installing`
`http://wiki.ros.org/turtlebot_simulator`

6.2 Real Turtlebot

Used to kick start all the sensor connection for the turtlebot robot.

Pre-install:
sudo apt-get install ros-indigo-turtlebot-bringup ros-indigo-turtlebot-navigation ros-indigo-rtabmap-ros

6.3 Rtabmap-ros (3D reconstruction of area)

Design the 3D environment of the arena using kinect.

Pre-install:
sudo apt-get install ros-indigo-rtabmap-ros

Weblink:
`http://wiki.ros.org/rtabmap_ros/Tutorials/MappingAndNavigationOnTurtlebot`

6.4 Frontier-explorer(Autonomous navigation)

Explore the unseen area inside the boundary.

Pre-install (Note:The modified code is placed inside the DarkROS):
`git clone https://github.com/paulbovbel/frontier_exploration.git`

Weblink:
`http://wiki.ros.org/frontier_exploration`

6.5 Object Recognition Packages

Used to recognize the 3D object in the environment.

Pre-install:
sudo apt-get install ros-indigo-ecto-*
sudo apt-get install ros-indigo-moveit-full
sudo apt-get install libosmesa6-dev
sudo apt-get install couchdb

Packages:
`git clone http://github.com/wg-perception/object_recognition_core`
git clone http://github.com/wg-perception/capture
git clone http://github.com/wg-perception/reconstruction
git clone http://github.com/wg-perception/linemod
`git clone http://github.com/wg-perception/ork_renderer`
git clone http://github.com/wg-perception/tabletop
git clone http://github.com/wg-perception/tod
`git clone http://github.com/wg-perception/transparent_objects`

Additional:
`git clone http://github.com/wg-perception/object_recognition_msgs`
`git clone http://github.com/wg-perception/object_recognition_ros`
`git clone http://github.com/wg-perception/object_recognition_ros_visualization`
`git clone https://github.com/wg-perception/ork_tutorials`

weblink:
`http://wg-perception.github.io/object_recognition_core/install.html#install`
`http://wg-perception.github.io/object_recognition_core/infrastructure/couch.html#object-recognition-core-db`
`https://wg-perception.github.io/ork_tutorials/tutorial01/tutorial.html`
`https://wg-perception.github.io/ork_tutorials/tutorial02/tutorial.html`
`http://wg-perception.github.io/capture/index.html#ork-capture`

6.6 AR tag tracker Package

Used to detect tags in the environment.
Pre-install (Note:The modified code is placed inside the DarkROS):
`Git clone https://github.com/sniekum/ar_track_alvar.git`

Weblink:
`http://wiki.ros.org/ar_track_alvar`

6.7 Speech Package

Pre-install:
sudo apt-get install ros-indigo-audio-common
sudo apt-get install libasound2
sudo apt-get install festvox-don

Weblink:
`http://wiki.ros.org/sound_play`

7 Implementation

All about this project is in github repositories DarkROS.

To run this project, please follow the following steps.

Please run these lunch files to run our project.

7.1 Mapping and Navigation

7.1.1 Simulation

Roslaunch `fake_turtle fake_turtle.launch` Use the command for the Gazebo to start up.

7.1.2 Real Time

```
Roslaunch real_turtle real_turtlebot.launch
Roslaunch rtabmap_ros demo_turtlebot_rviz.launch
```

7.2 Detection And Classification

7.2.1 Adding objects to the Database

```
rosrun object_recognition_core object_add.py -n "<name>" -d "<details>"
–commit
rosrun object_recognition_core mesh_add.py <add the ID code of DB>
<path to stl file> –commit
(or) rosrun object_recognition_capture upload -i <bag file> -n '<name>' -
d <detail> –commit
```

7.2.2 Training the Database

```
rosrun object_recognition_reconstruction mesh_object --all --visualize
–commit
rosrun object_recognition_core training -c `rospack find object_recognition_linemod`
/conf/training.ork
```

7.2.3 Test 3D object Detection And Classification

```
rosrun object_recognition_core detection -c  `rospack find object_recognition_ tabletop`
/conf/detection.object.ros.ork
```

7.2.4 Test The Tag tracker

```
Roslaunch ar_tracker_alvar pr2_indiv. launch
```

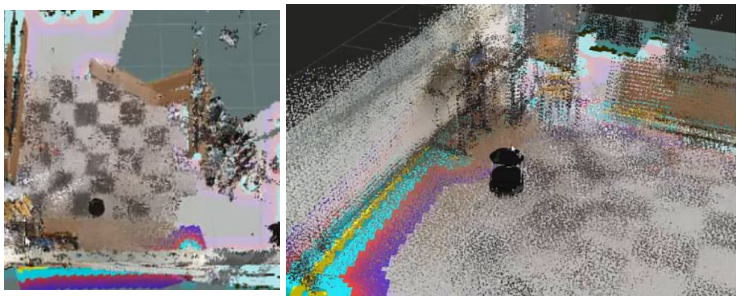
7.3 Running the Project

TurtleBot:
`Roslaunch real_turtle real_turtlebot.launch`
(or) `Roslaunch real_turtle real_turtlebot1.launch`
(or) `Roslaunch real_turtle real_turtlebot_final.launch`

Workstation:
`Roslaunch real_turtle real_turtlebot_station.launch`

8 Results

To see the full results with the implementation steps, please see the attached youtube link https://www.youtube.com/watch?v=RyFA3uSZs2g.



(a) Top view (b) Side View

Figure 9: Rtabmap with frontier explorer

Figure 10: 3D Object detection and recognition

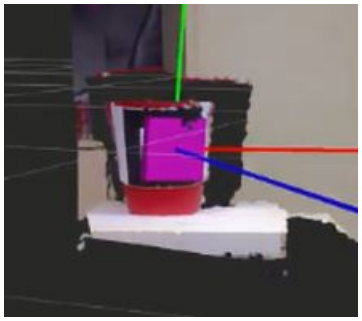


Figure 11: AR tag tracker

9 Project Management

Project management is a critical component of every successful automation project which make ease the project implementation.
To ensure the least amount of disruption when new requirements and process changes arise, we will meet regularly to review status and identify changes. Design and develop a flexible solution that can be quickly adapted to meet new requirements.

What we achieved:
1. Real-time project scheduling.
2. Analyze and expand project definition,specific requirements that constrain the scope and approach.
3. Review layout and equipment.
4. Component and system time-line management.
5. Github update, documentation, videos and codes.

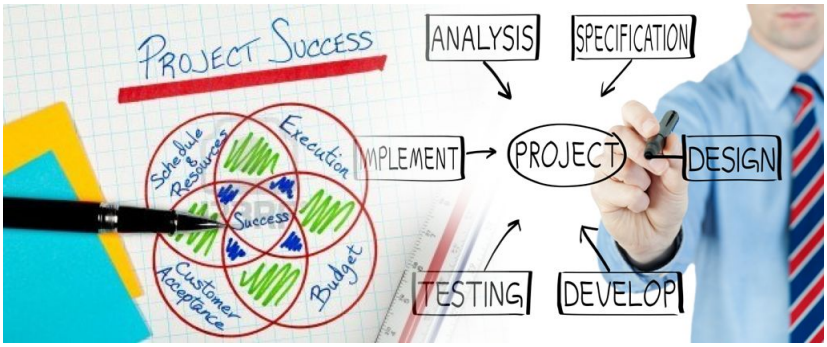


Figure 12: Project Management

10 Conclusions and future work

Overall we learned the strength of ROS especially navigation and vision. We also learned about the importance of using state machines to amalgamate all the task and goals of the robot to make it robust, easy to debug and scalable.

It is really interesting to discuss and analyze lot of new ideas and features as a new applications. The ideas took us to a new level of imagination. But within the limited time given and the limited tools we designed a cleaning robot using turtlebot in the laboratory. The project is designed to work in the real time environment process.

In this project, our robot tasks are detecting the trashes, recognize and classify them to categories. The process is designed that the robot is programmed to navigate and search the area for path and detecting object. In the meanwhile, detecting trashes and classified it into three sections like paper cup, can or bottle.

The future extension could be by robot control and picking objects using grasping hand. The control of the robot is switched between autonomous or manual remote control using a QR code and the remote control is much easy to navigate and start from the particular location. The picking or collection of the trash is implemented on the robot with the robotic arm.

Another future idea is probability mapping, which is to enhance and clean the search area efficiently. It is an advance option of determining the location where more of the trash is detected. By calculating the map, the next time the area can be searched as a first priority.

Another future idea is going for cleaning more complicated objects, such as dust using a brush and windows using a rage, which is hard to detect.

Thanks for giving us the opportunity to work on this project, we as a team enjoyed and appreciated your help and guidance.

Figure 13: Future work

11 References

1. BOSCH, Automatic 3D Reconstruction and Modeling. 2. A.Teichman, J.Levinson, Towards 3D Object Recognition via Classification of Arbitrary Object Tracks.
3. G.Bradski, 3D object categorization, detection, and viewpoint classification.
4. Philips company, Designing a Robotic Vacuum Cleaner.
5. Ramos-Garijo R, Prats M, Sanz PJ, Del Pobil AP, An autonomous assistant robot for book manipulation in a library.
6. GIZMODO, The Giant Robots That Serve the World's Largest Library Archives.
7. NEngelharda, F.Endresa, Real-time 3D visual SLAM with a hand-held RGB-D camera.
8. ROS By Example Volume 1 : https://github.com/pirobot/rbx1