SAPIENZA
UNIVERSITÀ DI ROMA

Evaluating perception in industrial scenarios: tools, metrics and benchmarks

Master thesis

Daniele Evangelista

evangelista.1665872@studenti.uniroma1.it

RoCoCo - Cognitive robot teams laboratory
Department of Computer, Control and Management Engineering Antonio Ruberti
Sapienza University of Rome

Supervisor:
Eng. Dr. Alberto Pretto

June 24, 2018

Abstract

Having rigorous test bed as benchmark for novel technologies, theories and methods is one of the key factor for improving more and more the research in every field. This concept is even more prominent in highly structured and rigorous settings such as the industrial one. This work is focused in particular on those scenarios and tries to propose rigorous benchmark procedures and tools for testing widely spread sensors, such as *Microsoft Kinect 2* or *Intel Realsense SR300* involving standard tools of common use in research in industrial settings, e.g. *RoboCup@Work* objects. As like as common commercial sensors, over the years, industry has seen a very fast growth of custom and complex sensors, such as structured light sensors, active and passive stereo perceptive systems and many others. For those last mentioned sensor categories, to the knowledge of the author, only one publicly available dataset existed up to the 2017, for this reason this work also introduces a novel dataset, the *RAW Dataset*, built with multiple sensors with different features, namely *passive* and *active stereo cameras* and *structured light sensors*. The dataset provides both rgb and depth images from 15 different scenes of objects in multiple arrangement, there are scenes with no clutter at all and other completely randomly built. All the scenes are provided with the ground truth 6D position of the objects. The ground truth will be used for evaluating some state-of-the art techniques for object detection and localization, as like as 3D scene reconstruction algorithms. This investigation will focus on properties that are important for practical applications in robotic industry, such as comparing the full 3D rigid transformation of the objects, evaluating the goodness of objects' bounding box detection, and some other relevant and specific 3D reconstruction approaches. The sensor setup of this dataset also introduces some novelty, a new sensor for passive and active stereo vision has been developed and tested, the *FlexSight Sensor*, built in fulfillment, behalf and supported by the European Community's project ECHORD++.

Nomenclature

Acronyms and Abbreviations

API	Application Programming Interface
GUI	Graphical User Interface
CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
OpenCV	Open source Computer Vision (library), `www.opencv.org`
Git	Git revision control, `www.git-scm.com`
RBP	Random Bin Picking
PPT	Pick and Place Task
IoU	Intersection over Union
AD	Average Distance
VSD	Visible Surface Distance
d.o.f.	Degrees of freedom

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision is one of the most prominent and prolific area of the computer science research, in particular perception and image processing are gaining more and more interests among the community. This work focuses the attention on machine and robotic perception in industrial settings. In those scenarios, having accurate algorithms and procedures is extremely important, since results may influence the overall productive chain. In this sense, many industrial realities, as like as big *ICT (Information Communication Technology)* companies, are investing a lot in improving production systems, most of the time introducing new technologies and tools within the community, but also promoting events and challenges all over the world for enforcing researchers in pushing always out of state-of-the-art limits. That is the case of international competitions like *Amazon Picking Challenge*[1] and the *RoboCup@Work*[2], which is also the one in which we are actively involved as participating team and part of the Organizing Committee (OC) (the author of this thesis is part of the Robocup@Work OC from 2016). Given this experience we tried to extrapolate needs and requirements of the recent research in industrial settings and then propose our solution to the problem of testing and rigorously assert novel ideas, algorithms, software and tools in general.

1.1 Motivations and Contributions

Having rigorous test bed as benchmark for novel technologies, theories and methods is one of the key factor for improving more and more the research in every field. This concept is even more prominent in highly structured and rigorous settings such as the industrial one. We focused in particular on those scenarios and tried to propose rigorous benchmark procedures and tools for testing widely spread sensors, such as *Microsoft Kinect 2* or *Intel Realsense SR300* involving standard tools of common use in research in industrial settings, e.g. *RoboCup@Work* objects. As like as common commercial sensors, over the years, industry has seen a very fast growth of custom and complex sensors, such as structured light sensors, active and passive stereo perceptive systems and many others. For those last mentioned sensor categories, actually no publicly available datasets existed up to the 2017 when MVTecc released its ITODD dataset [2], for this reason with this work we also introduce a novel dataset, the *RAW Dataset*, built with multiple sensors with different features, namely *passive* and *active stereo cameras* and *structured*

[1] https://www.amazonrobotics.com/
[2] http://www.robocupatwork.org/

light sensors. The dataset provides both rgb and depth images from 15 different scenes of objects in multiple arrangement, there are scenes with no clutter at all and other completely randomly built. All the scenes are provided with the ground truth 6D position of the objects. The ground truth will be used for evaluating some state-of-the art techniques for object detection and localization, as like as 3D scene reconstruction algorithms. This investigation will focus on properties that are important for practical applications in robotic industry, such as comparing the full 3D rigid transformation of the objects, evaluating the goodness of objects' bounding box detection, and some other relevant and specific 3D reconstruction approaches. The sensor setup of this dataset also introduces some novelty, namely we tested and developed a new sensor for passive and active stereo vision, the *FlexSight Sensor*, built in fulfillment, behalf and supported by the European Community's project ECHORD++[3].

1.2 Related Works

Many recent and past works need to be mentioned if compared to our work, especially when talking about benchmarking tools like datasets or similar test beds. Several datasets for 3D object detection were introduced in the past, in particular, the work of Fireman [3] gives a detailed and complete review of what is the past and present research line about that. Particularly interest on 3D object pose datasets has been given in the work of Hodan et al. [4], where a comprehensive list of dataset and their tools and metrics is shown in detail.

From the introduction into the market of commercial sensors such as the aforementioned Kinect 2 or Realsense SR300, many datasets have been acquired and made publicly available, but the most of them are of less interest for industrial application. On the contrary, the work from Hodan et al. [5], introduced the so called *T-LESS Dataset*, a large and very interesting dataset of texture-less objects oriented to the industrial settings. This dataset will be further presented in Sec. 3.2.1. On the same line of the T-LESS Dataset, also the work from MVTec [2] has recently introduced interesting tools for evaluating algorithms and techniques of object detection and 3D pose estimation in complex and concrete industrial settings, involving again texture-less objects with high relevance for manufacture and industrial scenarios.

Our proposed RAW Dataset has similar focus as the MVTec ITODD and T-LESS datasets, similar in evaluation and sensor setups, while introducing different approaches in the acquisition phase since our sensor setup is not fixed as the other two and is much more inspired to real industrial applications where cameras, sensors in general, are mounted directly at the end-effector of a robotic manipulator, so they move while it moves.

[3] http://echord.eu/

1.3 Structure of the Thesis

In the following chapters an overview of the state-of-the-art about perception in industry will be given (Chapter 2), focusing the attention on sensors technology (Sec. 2.2), 3D Scene Reconstruction and Object Detection techniques (Sec. 2.3 and 2.4), and state-of-the-art software libraries commonly used in industrial settings (Sec. 2.5). In Chapter 3 an overview of commonly used metrics (Sec. 3.1) and benchmark datasets (Sec. 3.2) are going to be presented. After this detailed view of techniques, theories and tools about perception in industry, in Chapter 4 the RAW Dataset is presented in detail, with a large overview of its details and composition (Sec. 4.1 and Sec. 4.2), the acquisition setup and procedure (Sec. 4.3 and 4.4) and finally how ground truth has been computed (Sec. 4.5) giving also a large view of the developed tools (Sec. 4.5.1 and 4.5.2). In Chapter 5 some experiments on the aforementioned RAW Dataset are going to be presented and brief discussion will be derived (Sec. 5.3). Finally, in Chapter 6 final conclusions will be given, and on going and future works will be announced. This document contains also an appendix (Appendix A) where some more details about the FlexSight project (A.1) are given.

Chapter 2

Perception in Robotics: Applications, Sensors and Algorithms

In this chapter we will discuss about robot perception in industrial settings, focusing the attention on the most common applications and problems, such as Random Bin Picking (RBP) and Pick&Place (PPT), the exploited sensors, and the algorithms and techniques used to solve the presented problems. We will also introduce some state of the art commercial libraries that nowadays are commonly used in several industrial applications.

2.1 Robotics in Industry

Over the years, industry has become one of the most important scenario where robots and automated systems have gained a widespread diffusion. This is basically thanks to the fact that robots can easily and quickly perform repetitive and complex tasks, while providing a constant and very high accuracy. Into the factories, robots are involved in every ring of the productive chain, from heavy loads handling to precise and accurate placing, from iron and metal soldering to small part assembly. The higher accuracy and speed that robots can reach, with so high levels of repeatability and precision, brought robots, manipulators in particular, at the top level of the industrial requirements.

The importance of robots in the industrial world is supported also by hi-tech giants, such as Amazon, Google and others, that in the last years demonstrated high interest in investing and developing their technologies in order to improve their performance and business. One interesting example is represented by the Amazon Picking Challenge competition proposed the first time at the International Conference on Intelligent Robots and Systems (IROS) in 2015 by the Amazon Robotics division[1]. This competition is thought to enhance and improve the research in the robotic manipulation, focusing the attention in grasping from random and highly cluttered environments, such as warehouse shelves or bins full of irregular and disorganized objects. In Figure 2.1, some examples of typical tasks that teams involved in the Amazon Picking Challenge should compete.

[1] https://www.amazonrobotics.com

Figure 2.1: **Amazon Picking Challenge example scenario.** In the images, the team is trying to detect and grasp objects from the red bin on the bottom, and place them on the shelf.

2.1.1 Random Bin Picking (RBP)

Monotonous tasks such as unloading a bin one part at a time into a machine, bulk parts sorting, and order fulfillment are labor-intensive, moreover they can even be dangerous if the parts or operations are heavy, sharp and they change position every time one object is moved away. For years, bin picking robots have been tackling these tedious jobs, but there are still so many applications to be realized.

While more capable than ever, robotic bin picking still has its limitations. It is all matter of accuracy. While robots are applauded for their repeatability, random bin picking requires accuracy in the face of chaos. The robot has to locate a part in free space, in an unstructured environment where the parts keep shifting positions and orientations every time a part is removed from the bin. That requires a delicate balance between robotic dexterity, machine vision, software, computing power to crunch all the data in real time, and a grasping solution to extract the parts from the bin. All those highly technical challenges give to RBP higher complexity and difficulty.

First of all, a brief introduction to all the kinds of Bin Picking must be given. There are three main types of bin picking: structured, semi-structured, and random bin picking. Each presents an increasing level of application complexity, cost and cycle time:

- **Structured**. Parts are positioned or stacked in the bin in an organized, predictable pattern, so they can be easily imaged and picked.
- **Semi-Structured**. Parts are positioned in the bin with some organization and predictability to help aid imaging and picking.
- **Random**. Parts are in totally random positions in a bin, including different orientations, overlapping, and even entangled, further complicating the imaging and picking functions.

From now on we will concentrate on Random Bin Picking, that is the main area in which we tested our work, moreover, the RAW dataset that will be presented later in this thesis and all the developed tools, have been thought for such robotic task in particular.

Figure 2.2: **Random Bin Picking Example.** Robotic random bin picking and part loading system uses 3D vision guided robots with magnetic grippers to locate and pick parts for a heat treating operation.

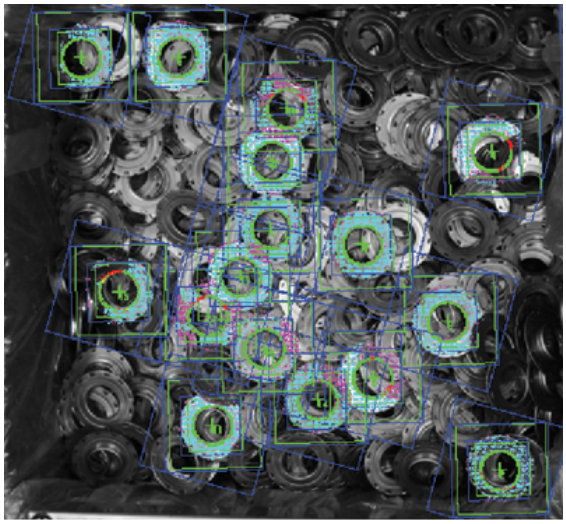


Figure 2.3: **Machine vision algorithm execution example in RBP task.** Here, a 3D area sensor is performing localization and recognition of random parts in a bin. Thanks to this technology, the robot should be capable of planning its next pick in the clutter.

An example of a robotic system performing random bin picking operation is depicted in Figure 2.2. In this kind of scenarios, the role of vision is crucial. Vision systems must be capable of recognize, detect and localize the parts with extremely high accuracy. Typical scenarios are like the one in Figure 2.3, it's clearly visible how the environment is cluttered, all the parts are randomly spread inside the bin, and the task of recognizing and localizing the next part to be picked, it's extremely difficult.

2.1.2 Pick&Place Tasks (PPT)

More generic, but with the same importance as the previous one, Pick&Place (PPT) task is another interesting robotic application in which vision and perception play an important role. Differently from the aforementioned Bin Picking applications, in Pick&Place scenarios, camera setup is not always fixed, they are mounted on the robotic arm, and the arm itself is not said to be fixed on the ground.

A clear example of Pick&Place scenario is the RoboCup@Work Competition, where the robotic arm that performs pick and drop operations with the objects, is actually mounted on a mobile platform that operates in a more large environment. Another example of robotic Pick&Place operation are the robot assisted assembly [6], in which the robot assists the human in performing operation in industrial scenarios guided by natural language interaction.

2.2 Sensors Technologies

In this section, we provide some details about the sensor technologies and a selection of relevant and state-of-the-art sensors typically employed in the pick&place and bin-picking applications (Sec. 2.1). The presented use cases represent only a small portion of the 3D sensor available in the market: anyway, almost all other sensors exploit one of the technologies described below.
Most of state-of-the-art industrial systems currently works with 3D sensors, sometimes coupled with a color or a gray-level camera, since they enable to obtain in a direct way the 3D position of the objects of interest. While almost all RGB cameras are currently based on standard and mature technologies (i.e., they are based on CCD or more frequently CMOS sensors), depth information can be obtained using several types of technology, each one with its strengths and weaknesses.

2.2.1 Passive VS Active Sensor

Traditionally depth information has been acquired using passive stereo cameras, i.e. cameras with two or more separate image sensors. Depth of 3D points are recovered by means of a correspondence problem: matched points projections are triangulated between pairs of sensors. Just like monocular cameras, stereo cameras belong to the class of passive sensors, i.e. sensors that do not modify the surrounding environment in order to acquire data. Unfortunately these systems fail in absence of salient visual

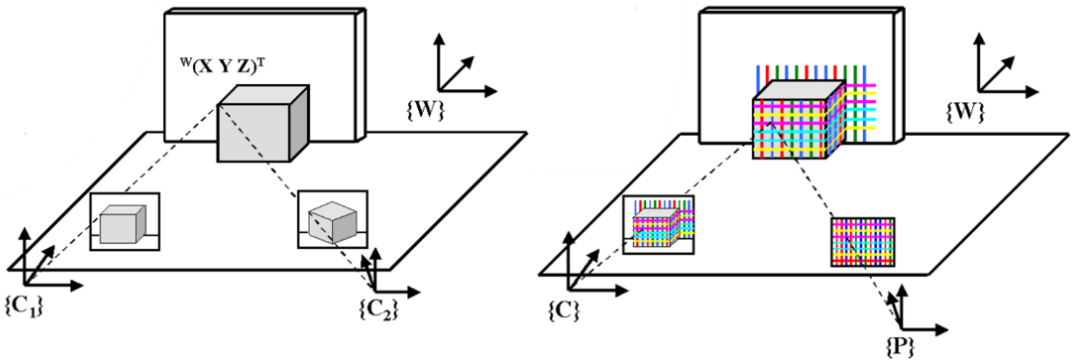


Figure 2.4: **Passive Stereo sensor VS Active Stereo sensor**. Passive stereo camera (left) VS. active depth sensor (right) with colored pattern projector (image taken from `http://eia.udg.es/~qsalvi/Tutorial_Coded_Light_Projection_Techniques_archivos/frame.html`).

features in the surfaces of the framed scene: for this reason, only very few modern systems are currently based on this type of technology.

On the other side, active sensor uses light emitters that project a specific pattern or a light with a specific wavelength *(Time-of-Flight sensors, ToF)*: all these sensors modify in some way the surrounding environment (i.e., they illuminate the scene). The correspondence problem in this case is solved, e.g., by searching the known pattern in the camera image (pattern decoding, used in *Structured Light sensors, SL*) or performing a traditional stereo matching algorithm using visual features synthetically created by the light projector (*Active Stereo sensors, AS*). In the first case (SL) just one camera is required, but the camera-projector couple should be carefully calibrated. In the second case two calibrated cameras are required, while an accurate calibration between the cameras and the projector is not required: the projector is just required to illuminate the framed scene.

An qualitative comparison of Active Stereo sensor and Passive Stereo sensor is given in Fig. 2.4.

2.2.2 Time-of-Flight Sensors

ToF sensors works by emitting a typically IR light carrier with known wavelength, then measuring the phase shift of the reflected light on the receiver side (Fig.2 right). 2D LIDAR (Laser Imaging Detection and Ranging) range scanners (Fig.2 left) use one panning emitter-receiver couple, while ToF cameras (Fig.2 right) use several emitters and a matrix of receivers that provide a depth map of the framed scene.

- **Pros**: ToF cameras are compact sensors that can provide full depth maps in real time, well suited also for moving devices.
- **Cons**: The depth map resolution is generally low, while the accuracy is not always

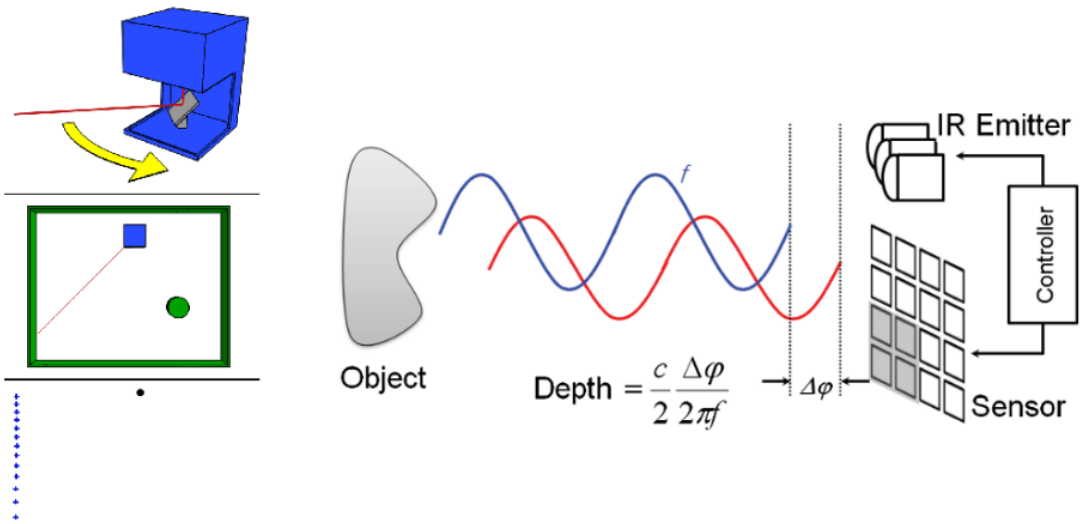


Figure 2.5: **Time-of-light sensor technology**. LIDAR range scanner (left, images taken form `https://it.wikipedia.org/wiki/Lidar`) and time-of-flight array principle (right image taken from [1]).

adequate, in particular close to surfaces' edges and corners. Generally they are not very robust when dealing with reflective materials or black surfaces.

A novel sensor in the ToF area is the *Basler ToF Camera*, depicted in Fig. 2.6 This new sensor provides cutting edge features in the field of the ToF cameras: it has a relatively high resolution (640 X 480) with high frame rate (20 fps) and a large working range. Unfortunately, like other ToF sensors, the accuracy (+/- 1cm) is often not adequate for many industrial applications.

2.2.3 Structured Light Sensors with Moving Head

Structured light sensors are based on light triangulation: they typically use a laser projector that thanks to a specific lens produces a known "sparse" pattern (i.e., typically a line). A conventional camera mounted at a known distance from the laser (baseline) frames the scene in order to obtain the distance from the illuminated 3D points. The most common structured light sensors use a laser line projector: such type of sensors can measure a profile for each image (see Fig. 2.7). It is possible to obtain a full 3D point cloud of the working area by integrating each measurements while moving the camera-laser couple over the whole area.

- **Pros**: These type of sensors generally provide high quality 3D reconstructions while being quite robust when dealing with reflective materials.
- **Cons**: They usually require expensive devices to move the scanning head in order to be able to scan the entire area; the cycle time is increased by the scanning time

Figure 2.6: **The Basler ToF Camera**.

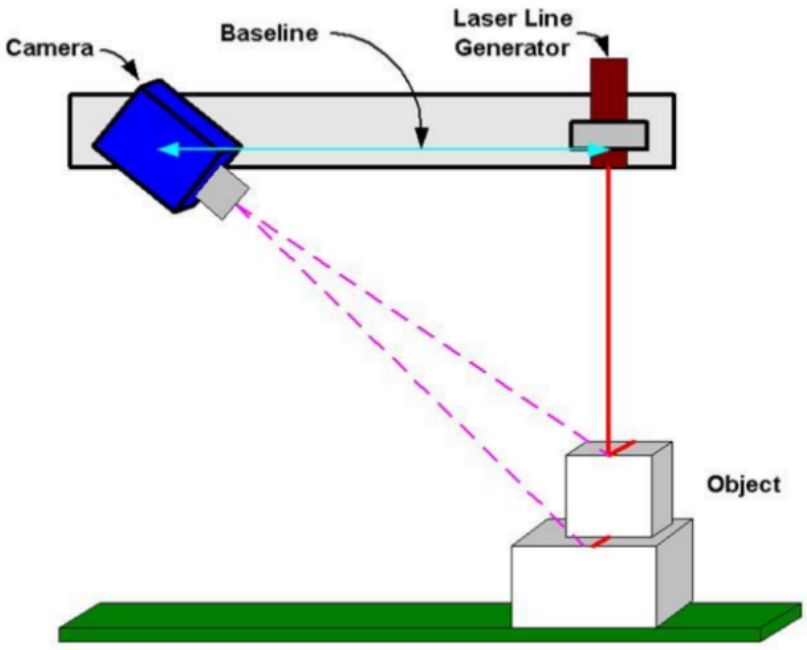


Figure 2.7: **Structured light camera basic principle**. This image is taken from `http://blog.teledynedalsa.com/`.

Figure 2.8: **Some Structured Light sensors from Sick**[2]. On the left the Sick ICV-3D, on the right the Sick Scanning Ruler S1200.

(typically from than 1 to 3 seconds).

An example of this kind of sensor is the *Sick IVC-3D* depicted in Fig. 2.8 (left). Sick [2] is one of the best known companies in the field of LIDARS and industrial structured light sensors. The Sick IVC-3D is a family of scanning heads able to triangulate a laser line in real-time, using an integrated laser projector. It requires to be mounted on a mobile system since the laser beam is fixed: the sensor should be moved over a 2D plane. The sensor streams data over an ethernet interface, and it is factory calibrated.

2.2.4 Structured Light Sensors with moving pattern

A variant of the previous sensor can be obtained motorizing the laser line projector: in this case it is not necessary to move the scanning device, but this new solution usually brings to higher costs in production and sell.

- **Pros**: These type of sensors generally provide high quality 3D reconstructions while being quite robust when dealing with reflective materials.
- **Cons**: The cycle time is increased by the scanning time (typically from 1 to 3 seconds).

An example of this sensor is the *Sick Scanning Ruler S1200* reported in Fig. 2.8 (right). It is a high-performance, factory calibrated, depth sensor which integrates a motorized laser projector. The field of view allows to detect objects inside an Euro pallet or an USA pallets, with a relatively high resolution (756×512) and a high depth accuracy, typically from 1 mm to 4 mm, depending on the distance from the working area.

[2] https://www.sick.com

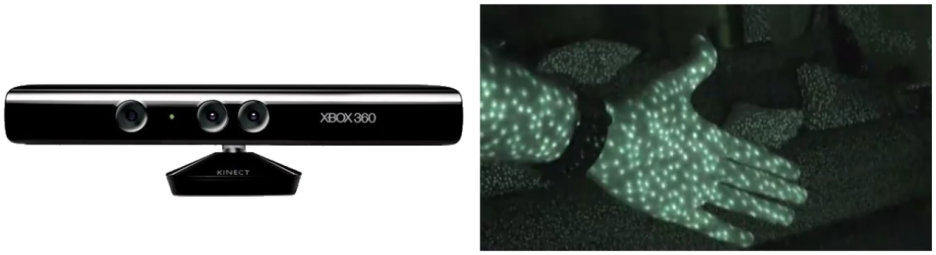


Figure 2.9: **Structured Light Sensors with Dense Pattern example**. The Microsoft Kinect RGB-D camera (left) and the dot matrix pattern projected by its near-infrared illuminator (right, image extracted from the video `https://www.youtube.com/watch?v=nvvQJxgykcU&feature`).

2.2.5 Structured Light Sensors with Dense Pattern

In order to avoid using mechanical devices to move the scanning head or the projector, some devices employ a dense pattern generator that illuminate the scene with a divergent dense pattern, commonly a $N \times M$ pseudo random dot matrix pattern (see Fig. 2.9). The pattern is known and each pattern patch (a $P \times P$ sub-matrix, $P << N, M$) represents a unique binary matrix. Detecting the pattern patches from the camera and knowing the baseline between the camera and the pattern projector, it is possible to estimate the depths by means of triangulation. The classical example of such sensors is represented by the Microsoft Kinect RGB-D camera (Fig. 7 left): it uses a near-infrared projector and it provides a dense 320×240 depth map.

- **Pros**: Generally compact sensors that can provide full depth maps in real time well suited also for moving devices.
- **Cons**: Hard to calibrate, not very robust against sunlight and other external light sources when using an eye safe projector.

An example of industrial SL with Dense Pattern laser sensor is the *Pick-It 3D Camera* (See Fig. 2.10). Pick-IT[3] introduced a complete bin-picking system for small parts that includes a custom-built depth sensor that is, under a technological point of view, very similar to the Microsoft Kinect 1 RGB-D camera. Pick-It produces two versions of its sensor: one short-range (400 - 800 mm) and one long-range (600 - 1300 mm). None of these sensors can deal with container larger than 600×800 mm.

2.2.6 Active Stereo with Changing Pattern

As the conventional stereo pairs, active stereo sensors create depth maps by triangulating corresponding points projections. In AS, the projector illuminates the scene in order to simplify the points matching. A simple but effective way to match points between the two cameras is to project a sequence of light patterns so that every pixel is

[3] www.pickit3d.com

Figure 2.10: **Pick-it 3D camera**. The Pick-it 3D camera (left) installed on top of a small box (right).

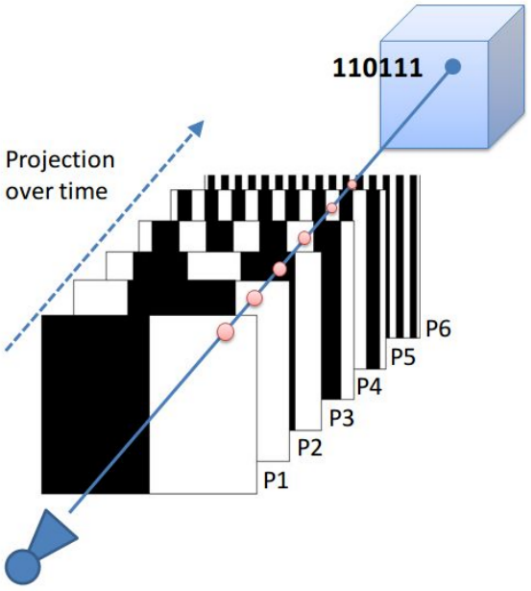


Figure 2.11: **Pick-it 3D cameraTime-Multiplexing coding**.

Figure 2.12: **EnShape Inspect sensors family**.

encoded with a *code-word* identified by the pattern sequence. This technique is called time-multiplexing coding, and the most common structure of the patterns is a sequence of stripes that decrease their thickness over time (single axis encoding, Fig. 2.11). These sensor are typically based on *DLP (Digital Light Processing)* projectors.

- **Pros**: For specific configurations and by using high-end projectors, these type of sensors can provide the most accurate 3D reconstructions, for this reason they are usually used for 3D printing applications.
- **Cons**: The cycle time is increased by the projection time (typically from 1 to 2 seconds). Due to the projector type (DLP), they are not robust against sunlight and other external light sources: usually these sensors require a protected environment with no external light sources.

As an example of this kind of technology, the sensor that needs to be mentioned here is the *EnShape Inspect* sensor (See Fig. 2.12) from EnShape[4]. It is actually a Active Stereo sensor with changing *pseudo-random* laser pattern. Dense pseudo random pattern projectors can be used also to perform active stereo scene reconstruction: since each pattern patch is unique, it is relatively easy to match their image projections between the two cameras. This sensor is also capable of providing 3D reconstruction in real-time, so it is well suited for moving devices, e.g. robotic manipulators, but it has only one limitation, it is extremely sensible to illumination differences since it uses eye safe laser projector so ambient light can easily influence the pseudo-random projected pattern.

2.3 3D Scene Reconstruction

The ability to estimate the full 3D structure of the environment is often an essential requirement in an industrial scenario, since many object detection and localization techniques are based on 3D approaches, so efficient methods for reconstructing the scene are needed. As introduced in the previous section, most of the more promising depth sensors, both passive or active, are based on "match and triangulate" techniques: in this section we introduce a selection of effective 3D scene reconstruction algorithms that

[4] http://www.enshape.de/

exploit this techniques. We will focus on 3D stereo reconstruction approaches, since they are by far the most popular and since they can be tested using our experimental sensor, the FlexSight Sensor (FSS).

2.3.1 Stereo Matching

Stereo matching is used to correlate points from one digital image of a stereo pair with the corresponding points in the second image of the pair. However, finding the best algorithms and parameters, is usually difficult, since different aspects must be considered: accuracy, completeness, occlusions, computational efforts, etc. The approach of using stereo vision to acquire object's 3D geometric information is on the basis of visual *disparity*. The concept of disparity in computer vision basically refers to the difference in coordinates of similar features within two stereo images. This concept is strictly related with what actually happens in our visual apparatus. Our binocular sight is also called parallax, given the horizontal displacement of our eyes. This disposition of the point of views generates the so called binocular disparity and as like as the definition we gave before, it is nothing more than the difference in image location of an object seen by the left and right eyes. From this image difference, it can be easily derived a relationship with the 3D world, namely we are able to reconstruct the depth of a given observed point starting from those displacements. The drawing in Fig. 2.13 shows an explanation of the relationship existing between image displacement, namely pixels' disparity, and the depth. In the drawing, light from the point A is transmitted through the entry points of a pinhole cameras at B and D, onto image screens at E and H. The two camera lens are respectively at distance $BD = BC + CD$. That means that there are two couple of similar triangles, namely the triangle couples (ACB, BFE) and (ACD, DGH). So the displacement d between the two points E and H on the two images is given by:

$$\begin{aligned} d &= EF + GH \\ &= BF(\frac{EF}{BF} + \frac{GH}{BF}) \\ &= BF(\frac{EF}{BF} + \frac{GH}{DG}) \\ &= BF(\frac{BC + CD}{AC}) \\ &= BF(\frac{BD}{AC}) \\ &= \frac{k}{z} \end{aligned} \tag{2.1}$$

where $k = BD * BF$ and $z = AC$ is the distance from the camera plane to the object, namely the *reconstructed depth* of the pixel. So assuming the cameras are level, and image planes are flat on the same plane (images must be rectified), the displacement in the horizontal axis between the same pixel in the two images is given by:

$$d = \frac{k}{z} \tag{2.2}$$

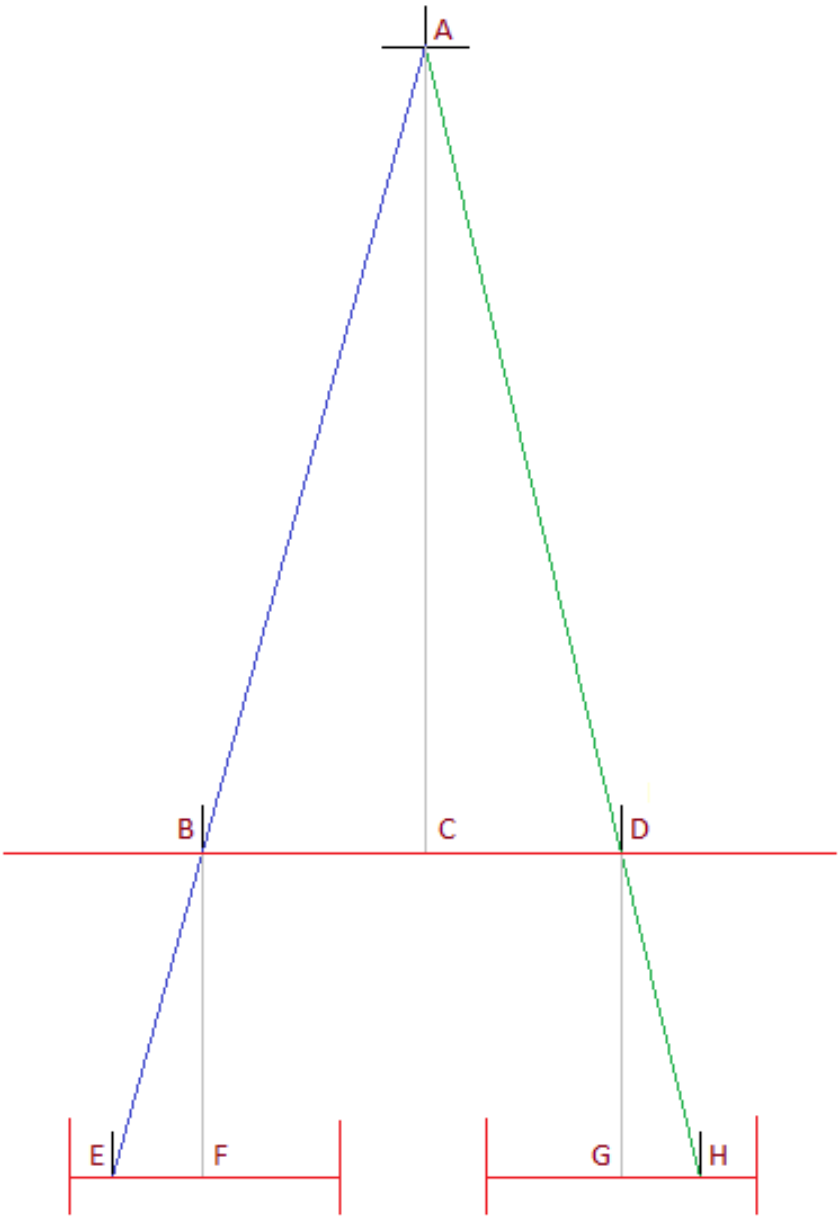


Figure 2.13: **Image disparity and Depth**. The drawing describes the relationship of image displacement to depth with stereoscopic images, assuming flat co-planar images. Image taken from `https://en.wikipedia.org/wiki/Computer_stereo_vision`.

where k is the distance between the two cameras times the distance from the lens to the image. Moreover, the depth component in the two images are z_1 and z_2, given by:

$$\begin{aligned} z_1(x,y) &= \min\{v : v = z_2(x, y - \frac{k}{z_2(x,y)})\} \\ z_2(x,y) &= \min\{v : v = z_1(x, y - \frac{k}{z_1(x,y)})\} \end{aligned} \tag{2.3}$$

There are many techniques that exploit this brief and intuitive geometrical concept, in particular we will now address two popular techniques, namely *Semi-Global Matching* from [7] and *PatchMatch Stereo* from [8]. Those techniques aims to find the same disparity ideally introduced before by exploiting image features and pixel color information. They have been also intensively used as benchmark tools for our RAW dataset and our experimental sensor setup, that is why we are now going to approach to them more in detail.

Semi-Global Matching (SGM)

The Semi-Global matching is, actually, one of the best matching strategies used both in photogrammetry and computer vision, offering good results with low run-time. It is commonly used in many real-world applications and implemented inside the firmware of several stereo based depth-cameras. The Semi-Global Matching method [7] performs a pixel-wise matching allowing to shape efficiently object boundaries and fine details.

The algorithm works with a pair of images, and their camera parameters, either intrinsic and extrinsic parameters must be known. That is because it assumes to know the epipolar constraint between the two images, the two cameras actually, so corresponding points are assumed to lie on the same horizontal image line, e.g. images have been rectified using such epipolar constraints. This algorithm realizes the minimization of a global smoothness constraint, combining matching costs along independent one-dimensional paths trough the image. The novel idea in SGM is that it actually computes the pixel matching cost through several paths in the image, not only on the horizontal epipolar line direction like it happened with previous implementations, e.g. scanline algorithm. All those paths and pixel-disparity pairs, aggregate in the computation of a cost function that SGM tries to minimize.

More in detail, the cost function $L'_r(p,d)$ of the pixel p and disparity d, along the path r is defined as follow:

$$\begin{aligned} L'_r(p,d) = & C(p,d) + \min(L_r(p-r,d), \\ & L_r(p-r,d-1) + P_1, \\ & L_r(p-r,d+1) + P_2, \\ & \min_i L_r(p-r,i) + P_2) - \min_k L_r(p-r,k) \end{aligned} \tag{2.4}$$

In Eq. 2.4 the first term $C(p,d)$ is the similarity cost (i.e. a value that penalizes,

using appropriate metrics, solutions where different radiometric values are encountered in the neighbor area of the corresponding points), the second minimization term is for evaluating the regularity of the disparity field, in fact there are two penalization terms P_1 and P_2 that basically control small and large change in the disparity with respect to the previous point along the matching path r. In the end there is a regularization term that allows to reduce the final value subtracting the minimum path cost of the previous pixel from the amount, if no regularization term is used, the cost tends to gradually increase during cost aggregation along the path.

Minimizing such cost function for a 2D image space is a NP-complete problem. SGM minimizes the cost by means of Dynamic Programming, with the novel idea of computing the optimization combining several individual path, symmetrically from all directions through the image. Summing the path costs in all directions and searching the disparity with the minimal cost for each image pixel p, produces the final disparity map.

PatchMatch Stereo

PatchMatch [9] is a iterative algorithm for finding the best match between image patches across two different images A and B. The core of the algorithm is very simple: initially, for a patch centered at (x, y) in image A, a random matching patch is assigned (called nearest neighbor) ad a offset $f(x, y)$ or v in image B. With $D(v)$ we then denote the error distance between the two patches at (x, y) in A and at $((x, y) + v)$ in B. In every iteration, the algorithm refines the nearest neighbor for every patch of image A basically performing two steps:

1. **Propagation Step**: Assuming that the two neighboring patches offsets are likely to be the same, the nearest neighbor for a patch at (x, y) is improved by propagating the known offsets of neighbors of (x, y) only if the patch distance $D(v)$ is improved. Moreover, this propagation is performed using different offsets w.r.t. odd and even iteration. Refer to the original paper for more details.

2. **Random Search**: If v_0 is the new offset after the propagation step, further refinement is done by constructing a window around that pixel and searching through a sequence of offsets $\{u_i\}$ at exponentially decreasing distance from v_0. Also here, the patch distance errors $D(u_i)$ are computed and the offset is updated only if the error decreases for some of the new offsets.

This is the very original idea of PatchMatch, and it can be easily applied to any problem that involves matching of part of an image, such as stereo matching. Here in particular, the search space can be drastically reduced, because supposing the images are aligned, the algorithm only has to look for similar patches, offsets technically speaking, by looking on the epipolar lines that are horizontal and parallel, so the search space reduces to lines instead of the full image space.

2.4 Object Detection

All the robotics application introduced in Sec. 2.1 requires to detect and localize one or more known objects in the working area: we will introduce here some machine vision algorithms and techniques for object detection and localization. In particular it will be shown a comparison between standard state of the art techniques based on shape-based approaches, and deep learning networks oriented to object detection and recognition.

2.4.1 The object detection and localization

In computer vision, object detection and localization are two similar tasks. Basically, given an image I and a set of objects represented by some identifiers $O = 1, 2, \cdots, n$, the classes of the objects in general, the task is to estimate each object in the image. For estimation here we can assume many different elements:

- **Bounding-box**: namely a sub-region of the image I that contains the entire object instance;
- **6D object pose**: the 3D position of the object w.r.t. the camera frame. For 3D position it is intended the 3D rotation matrix R and the translation t.

However, in literature, an explicit definition of the two distinct problems of 6D Localization and 6D Detection has been given [4]:

> *Given a training set $T = \{T_1, T_2, \cdots, T_n\}$ for a set of rigid objects represented by identifiers $O = \{1, 2, \cdots, n\}$ an estimator should estimate a sequence $E_I = [(o_1, \hat{P}_1, s_1), (o_2, \hat{P}_2, s_2), \cdots, (o_n, \hat{P}_n, s_n)]$, where $\hat{P}_i$ is an estimated 6D pose of an instance object $o_i \in O$ with confidence $s_i \in (0, 1]$.*

we define:

- **6D Localization Problem**: *in input at the problem a multi-set $L_I = \{o_1, 0_2, \cdots, o_n\}$ is given. So the output is $|E_I| = |L_I|$. The Estimator knows how many objects are in the image I.*
- **6D Detection Problem**: *No further information are given in input to the problem. The size of the output $|E_I|$ only depends on the estimator capabilities.*

2.4.2 Template Based Approaches

Template based approaches, more in general in literature called *Template Matching*, are techniques in digital image processing for finding small parts of an image which match a template image, or a template model of an object and so on. It can be used in manufacturing as a part of quality control, a way to navigate a mobile robot, or as a way

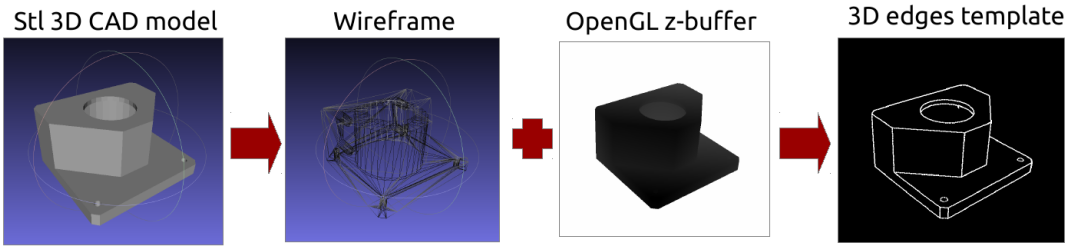


Figure 2.14: **D²CO input data example**. The raster models used in D²CO are computed from the 3D CAD models of the objects.

to detect edges in images. As already anticipated, as template we can intend different elements, they can simply be parts of an image, or they can be more complex elements such as the 3D CAD model of an object that is going to be found in the image.

In the first case, where we look for similar part of the image, the problem can be further exploited by simplifying the search space introducing the concept of *features*. Features are nothing more than a simple, and most of the time effective, way to reduce the dimension of the search space by describing the problem in lower dimension. Examples of features are edges in an image, corners, or more complex elements such as *SURF*, *SIFT* or *ORB* descriptors [10]. The last mentioned are very commonly used in computer vision, especially in image processing for autonomous localization and mapping (SLAM), place recognition, image classification and so on.

An example of template based approach is the D²CO algorithm from [11], where a 3D CAD model of an object is used as a template for object detection, localization and pose refinement within an image. This is the same approach that we used in our experiments for computing the ground truth for our aforementioned RAW Dataset, that is why in the following section it will be further explained.

D²CO Algorithm

As already anticipated, standard approaches are based on object shape and appearance. This means that a CAD model of the object is needed, since the algorithm mainly works on edges extraction and some sort of template matching between the projected model and the edges extracted from the image.

An example of standard algorithm that is totally based on shape and edges is the one presented in [11]. This is the very same approach used also in RoboCup@Work by the SPQR@Work team for performing the final registration step in the object detection pipeline.

The algorithm is called D²CO, namely Direct Directional Chamfer Optimization. D²CO refines the object position employing a non-linear optimization procedure, where the cost being minimized is extracted directly from a 3D image tensor composed by a sequence of distance maps. No ICP-like iterative re-association steps are required: the data association is implicitly optimized while inferring the object pose. This approach

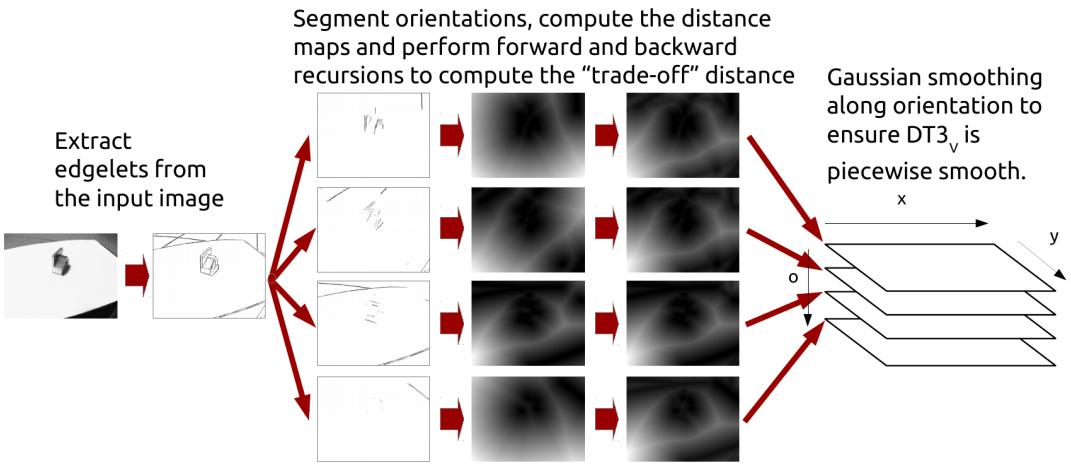


Figure 2.15: **Directional Chamfer Distance**. DCD tensor computation pipeline.

is able to handle textureless and partially occluded objects and does not require any off-line object learning step: just a 3D CAD model is needed (see Figure 2.14).

The Directional Chamfer Distance (DCD) tensor encodes the minimum distance of an image point to an edge point in a joint direction/location space. Unlike the Chamfer distance, the DCD takes into account also the edges directions, see Figure 2.15 for further explanations.

The algorithm extract a set of object candidates by pre-computing the (projected) raster templates along with their image orientations for a large number of possible 3D locations; for each point, it looks up the DCD tensor, computing the template average distance. The templates are then sorted for increasing distances. D^2CO finally refines the object position employing a non-linear optimization procedure that minimizes a tensor-based cost function, look at Figure 2.16 for a registration example. Then, the Levenberg Marquardt algorithm with a Huber loss function is used in order to reduces the influence of outliers. During the optimization, both the (projected) points position and the (projected) points orientation are constantly updated.

2.4.3 Deep Learning Neural Networks for Object Detection

Despite standard techniques for object detection like the one introduced in the last section, here we are going to explore some techniques that in the recent years have become the state of the art in most of the common computer vision tasks, such as object detection. Those approaches are based on *CNNs (Convolutional Neural Networks)* and are commonly associated with the Deep Learning literature. In Machine Learning in fact, CNN is a class of deep, feed-forward artificial neural network.

The idea of this kind of networks comes from biological processes:

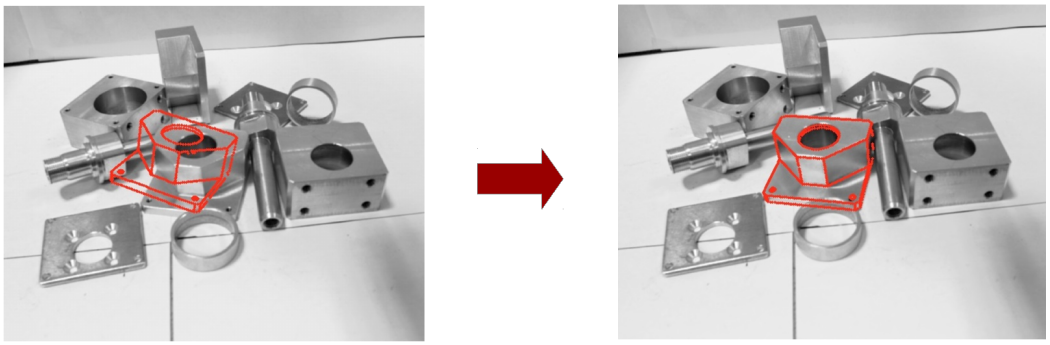


Figure 2.16: **D^{2}CO Registration Example**. An example of object registration using the D^{2}CO algorithm.

- The connectivity pattern between neurons is inspired by the organization of the animal visual cortex;
- Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field;
- The receptive fields of different neurons partially overlap such that they cover the entire visual field.

Following those intuitions, also in Computer Vision, more in detail in image processing, CNNs have been applied analyze and process visual data such as our brain does. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage and is something that comes directly from the input data used during the training phase of the network, that is why this kind of approaches are also called *data driven*, because they mostly relies on huge amounts of data.

More in detail a CNN consists of an input and an output layer, as well as multiple hidden layers. Those hidden layers typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers. In particular:

- **Convolutional Layer**: Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each neuron here processes data only for its receptive field. Those kind of input partitioning allows CNNs to tolerate translation of the input image (e.g. translation, rotation, perspective distortion, etc.)
- **Pooling Layer**: Those kind of layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the

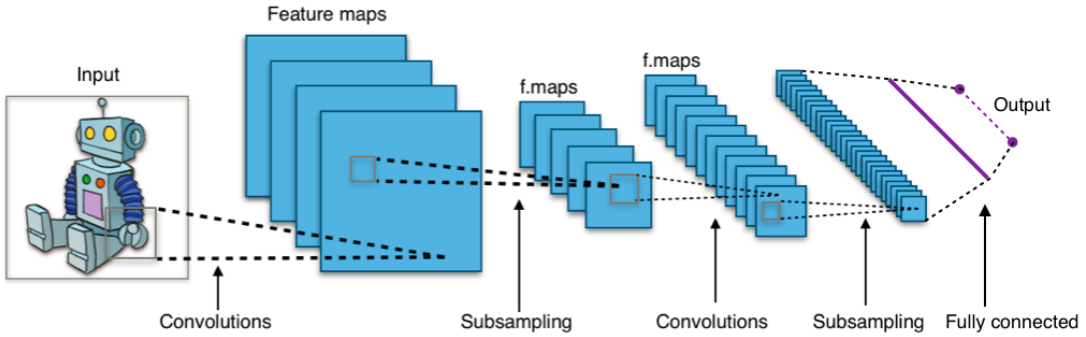


Figure 2.17: **Typical CNN architecture**. Typical CNNs architectures are composed by single or many convolutional layers, each followed by some pooling layer in order to reduce the dimension of the data for the next convolutional layer. As our scope is to evaluate object detection and classification results, commonly used architectures uses fully connected layer at the end of the network for performing classification.

maximum value from each of a cluster of neurons at the prior layer. Pooling layers are commonly used after convolutional ones in order to reduce the dimension of the data, also called data dimensionality reduction.

- **Fully Connected Layer**: Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP).

An example of a typical CNN architecture is given in Figure 2.17. One real architecture that we also tested with our RAW Dataset is called *YOLO (You Only Look Once)* from [12], and its second improvement from [13] which realized state of the art results on standard detection tasks like *PASCAL VOC* and *COCO*.

YOLO (You Only Look Once) Deep Neural Network Architecture

The revolutionary YOLO architecture is a very deep neural network composed by 24 convolutional layers followed by 2 fully connected layers. A detailed view of this architecture is given in Figure 2.18.

This has been considered a very novel and interesting idea since it realizes detection as a regression problem, in particular it divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. A very useful example of this concept is given in Figure 2.19, where it is explained how the process goes on. Basically, instead of directly predicting the bounding box of the object, they perform regression on the offsets between some pre-computed bounding boxes used as test and the real one. In this way, the process is much more easy and fast.

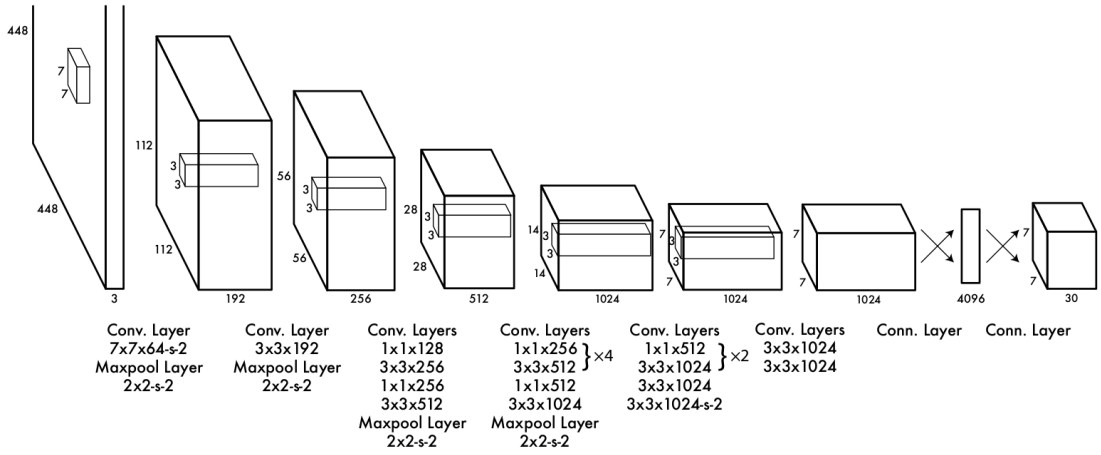


Figure 2.18: **YOLO Architecture**. This detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers.

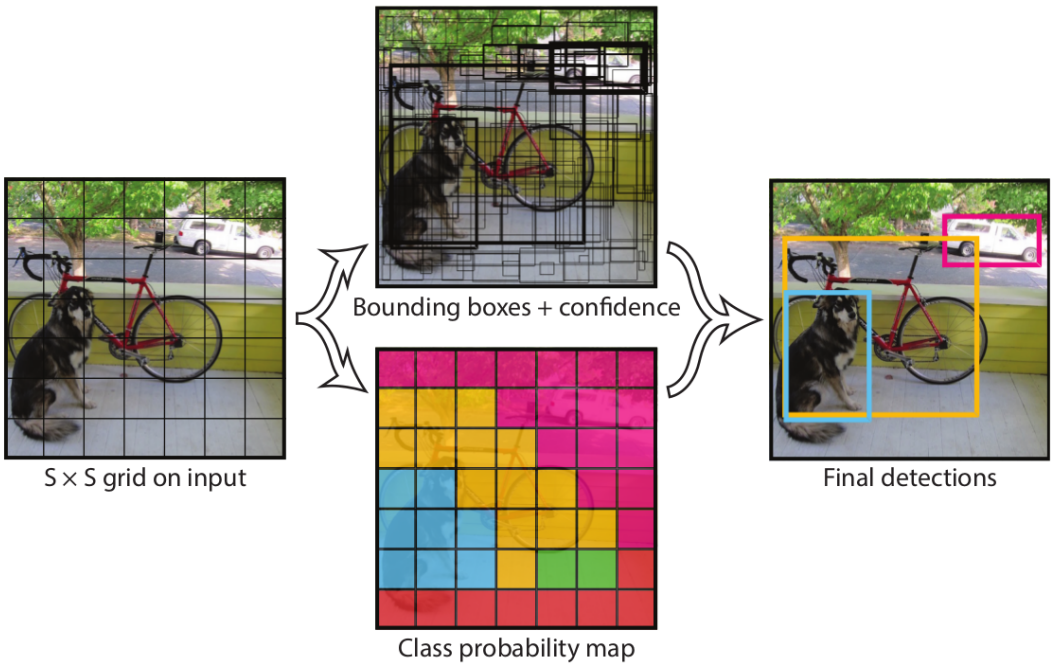


Figure 2.19: **YOLO Image Analysis**. The image is sampled with a specific window, then each cell is then converted into class probability map and fused with the bounding boxes and their confidence in order to obtain the final decisions.

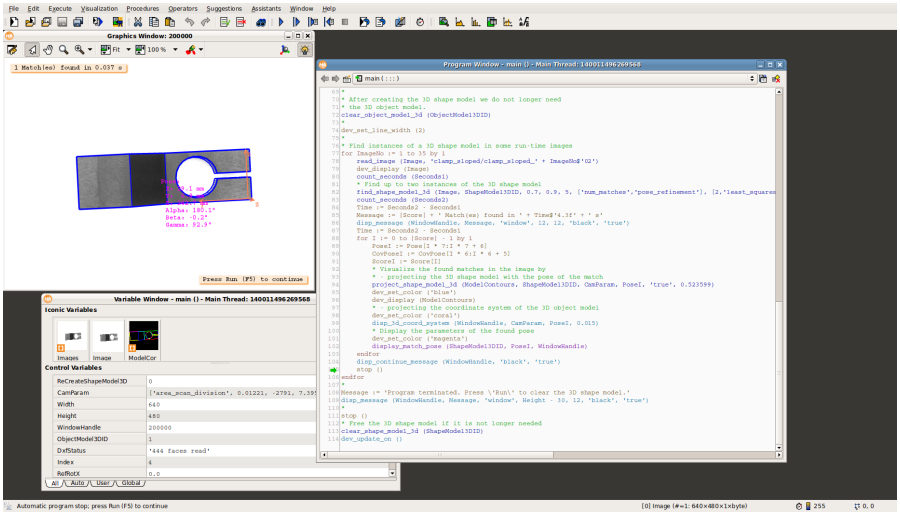


Figure 2.20: **Hdevelop GUI example.** An example of using the Hdevelop software from the Halcon Libraries. In particular here we are performing an object detection and localization task.

2.5 State of the art Software Libraries in Industry

Machine Vision is one of the most active area in industrial settings. Over the past years, many software companies and Open Source communities have dedicated lot of effort in developing robust and effective techniques and algorithms in order to assist industrial realities, such as companies and start ups, in performing computer vision assisted tasks, e.g. random bin picking, Pick&Place tasks and so on.

In the following subsection a list of tools and libraries will be introduced, focusing mainly on the MVTec's Halcon Libraries, which are the one that we used in the experiment phase of this work.

2.5.1 Halcon Libraries

Halcon[5], from MVTec, is a set of commercial software developed and sold explicitly for industrial settings. Over the past 5 years it has become the state of the art in machine vision for industrial tasks. It serves all industries with an extensive library of more than 1600 operators for blob analysis, morphology, matching, measuring, identification, and 3D vision, to name just a few.

The full library can be accessed from common programming languages like C, C++, C#, Visual Basic .NET, and Delphi. In particular, our tests have been developed using the C++ APIs. In the following chapters we will test this standard Machine Vision approaches over the RAW and T-Less datasets, and compare them with completely different approaches such as Deep Learning CNNs for object localization and recognition.

The Halcon Library has also an interactive and friendly GUI, provided in order to

[5]http://www.mvtec.com/products/halcon/

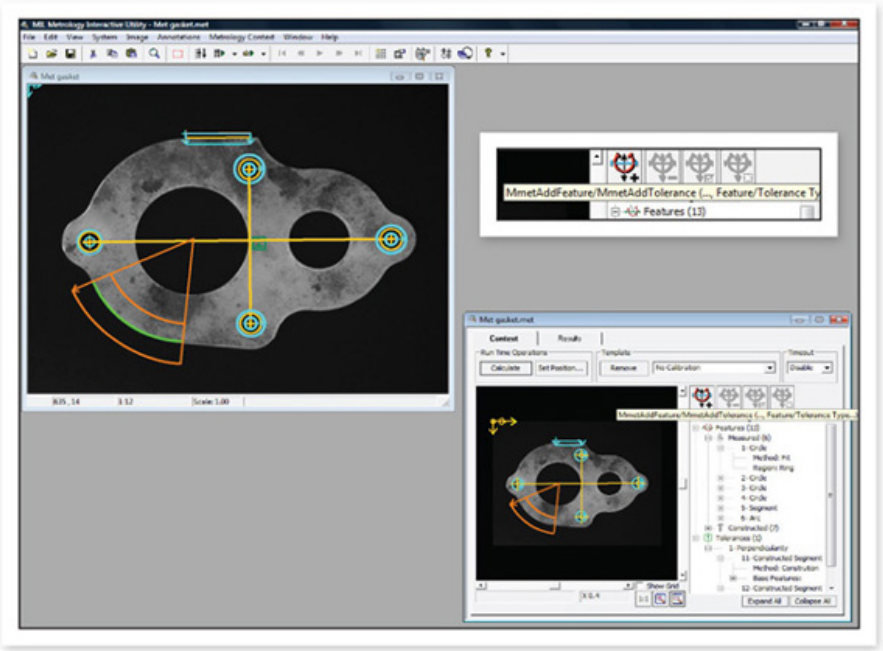


Figure 2.21: **MIL GUI example.** The graphical user interface of the Matrox Imaging Library.

facilitate the interfacing with the low level software APIs. The aforementioned software tool is called HDevelop, and an example of its usage and graphical interface is depicted in Figure 2.20.

As anticipated, this software library is under commercial license, and our distribution has been sold to La Sapienza University of Rome that can use it for research and other non-commercial purposes.

2.5.2 Matrox Imaging Library (MIL)

Another important software library that needs to be mentioned is the Matrox Imaging Library[6] (MIL). MIL is a complete collection of software tools for developing machine vision in lots of different scenarios, it is not restricted to the industrial one such as for the previously mentioned Halcon Libraries, but it covers also medical images applications and many others.

MIL includes also a graphic user interface for fast developing and prototyping of solutions. An example of this GUI is depicted in Figure 2.21.

This library is not part of the tests and examples performed during this work.

[6]https://www.matrox.com/imaging/en/products/software/mil/

Chapter 3

Benchmarks and Metrics in Industry

In this chapter all the tools and the benchmarks for object localization and detection will be exposed. In particular, first the metrics and the techniques used to evaluate pose estimations will be presented, then some publicly available datasets will be presented.

The problems that are going to be tackled are related to the 6D object pose estimation task. A 6D object pose is mathematically the 3D position of the object in the space plus 3 more terms that describe its rotation:

$$\hat{P} = (x, y, z, \alpha_x, \alpha_y, \alpha_z)^T \tag{3.1}$$

The Eq. 3.1 refers to the 6D pose estimate of an object, the first three terms x, y, z describe the position in the camera reference frame, while the last three terms $\alpha_x, \alpha_y, \alpha_z$ represent the object's rotation angles along the x, y, z camera axis respectively. This formulation can be easily manipulated in terms of rotation matrices and translation vectors:

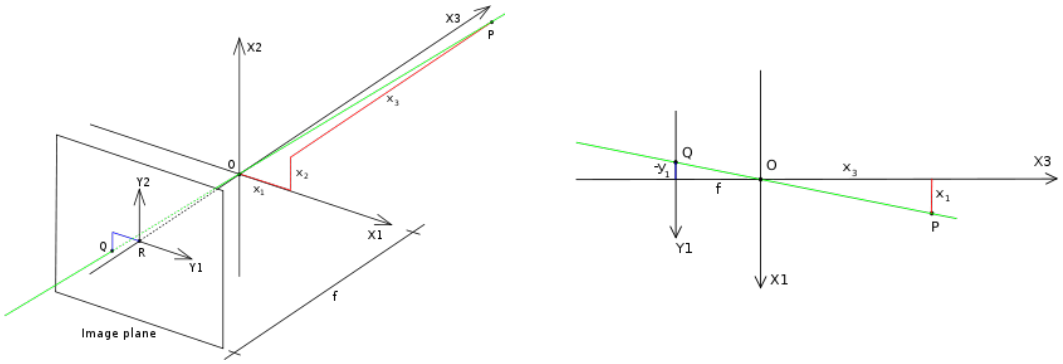
$$\hat{P} = (R, t) \tag{3.2}$$

The Eq. 3.2 is a manipulated version of 3.1 where the rotation matrix R encodes the three angles of rotation $\alpha_x, \alpha_y, \alpha_z$ and the translation vector t encodes the x, y, z position of the object in the reference frame of the camera.

Each object pose estimate can also be projected back to the camera image plane. This projection between the 3D space and the 2D space of the image plane is achieved by applying the projection obtained with a given camera matrix model. The camera model that we consider here is the so called Pinhole Camera Model, and it's geometry is depicted in Figure 3.1.

The mapping from the coordinates of a 3D point P to the 2D image coordinates of the point's projection onto the image plane, according to the pinhole camera model is given by:

$$x_i = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.3}$$



(A) 3D view (B) 2D view

Figure 3.1: **Pinhole Camera Geometry.** In (A) the geometry of the pinhole camera in 3D view, in (B) its 2D representation.

In Eq. 3.3 x_1, x_2, x_3 represent the 3D position in space of a generic point, f is the focal length of the camera, and u, v are the image coordinates obtained after the projection.

The formulation just explained is for an ideal pinhole camera located in the origin and with focal length equal for the x and y axes of the image. Typically, the situation is different, and the more generic formulation is like the following:

$$x_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.4}$$

In Eq. 3.4 the camera is formulated using its internal parameter f_x, f_y, c_x, c_y that represent respectively the two focal lengths on the x, y image axis and the camera optical center $(c_x, c_y)^T$.

With Eq. 3.3 and 3.4 we can project each point of the 3D object model onto the image plane and compare the estimated pose with the ground truth one following one of the metrics that are going to be explained in the sections below.

3.1 Metrics

The problem of evaluating how good is a pose estimate w.r.t. the ground truth is a challenging and still very open topic in computer vision community. Taking inspiration from [4] we will first focus on bounding box estimates comparison and then will pass to the more complex and challenging problem of compare two object 6D pose estimate.

3.1.1 2D Intersection over Union (IoU)

The most common way to measure accuracy of object detection in 2D domain is to calculate the Intersection over Union score [14]:

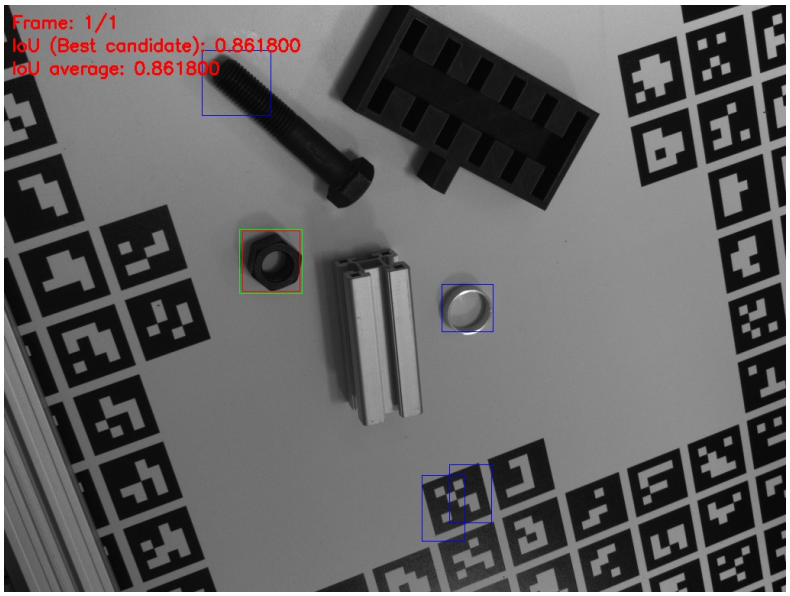


Figure 3.2: **IoU score estimation example.** An example of estimating the 2D bounding box of an object, in particular, the small nut, called M20 in the RAW dataset, with the Halcon Libraries. In red the ground truth bounding box, in green the Halcon best candidates, in blue the last two best candidates.

$$s_{IOU}(\hat{B}, \bar{B}) = \frac{area(\hat{B} \cap \bar{B})}{area(\hat{B} \cup \bar{B})} \tag{3.5}$$

In Eq. 3.5 $\hat{B}$ and $\bar{B}$ are the estimated and ground truth 2D region respectively. Depending on the task, $\hat{B}$ and $\bar{B}$ can be rectangular regions (given by bounding boxes) or segmentation masks. For evaluation of 6D object pose estimates, the 2D regions can be obtained by projection of the object model M in the estimated pose $\hat{P}$ and the ground truth pose $\bar{P}$. Such pose error function is ambiguity-invariant, but since it operates in the projective space, it provides only a weak information about fitness of the object surface alignment. That basically means that every projection of the model that realizes in the same bounding box will have the very same score, even if the object is completely flipped among the two 6D poses. It is said that the IoU score is not ambiguity-invariant, we will see in the further section what ambiguity is.

That is the reason why we only consider IoU for 2D measurement comparison, and so, only for object detection tasks, where just the location of the object within the image plane is considered to be interesting, not its exact position and orientation in 3D space. Examples of bounding boxes comparison, so IoU estimation is given in Figure 3.2. Usually a object detection is considered correct, when the IoU score of 2D bounding boxes of an object in the estimated and the ground truth pose is above a threshold (e.g. 0.5).

3.1.2 5cm, 5deg Criteria

After having introduced the metrics used for comparing object detection estimates, we now pass to tackle the problem of estimating the "goodness" of a 6D pose estimate. In the introduction of chapter 3 we formulated the pose of an object in 3D space in terms of rotation matrix and translation vector, as stated in Eq. 3.2. From this simple formulation, intuitively, a simple error function can be built, in particular we can consider the error of the estimated pose $\hat{P} = (\hat{R}, \hat{t})$ w.r.t. the ground truth pose $\bar{P} = (\bar{R}, \bar{t})$ as composed by two independent terms, the translational error (e_{TE}) and the rotational error e_{RE} respectively. These errors can be used as a metric for the computation of the goodness of a 6D pose estimate.

More in detail, the aforementioned e_{TE} and e_{RE} can be mathematically expressed as:

$$e_{TE}(\hat{t}, \bar{t}) = \|\hat{t} - \bar{t}\|_2 \tag{3.6}$$

$$e_{RE}(\hat{R}, \bar{R}) = arccos(\frac{Tr(\hat{R}\bar{R}^{-1}) - 1}{2}) \tag{3.7}$$

In particular, Eq. 3.6 is simply the squared distance from the 2 points represented by their respective translation vectors $\hat{t}$ and $\bar{t}$, and Eq. 3.7 represents the rotational error in the axis-angle representation of the rotation matrices $\hat{R}$ and $\bar{R}$. More over, in e_{RE}, the function $Tr()$ is the trace of the rotation matrix.

After the previous formulation, we can now introduce a simple and intuitive criteria for evaluating the goodness of a 6D pose estimate. In [15], the so called *5cm, 5deg* criteria have been introduced. It simply states that it is considered a good pose estimate if and only if $e_{TE} \leq 5cm$ and $e_{RE} \leq 5deg$. Of course, more general formulation of the same criteria can be used, by imposing variable thresholds t_{TE} and t_{RE} in the place of the constants $5cm$ and $5deg$.

This criteria was initially used for evaluating camera pose estimations, and it perfectly fits the case, but in terms of computing the goodness of a 6D object pose estimate, it is not adaptive to object model projections, in the sense that 2 different poses that refer to 2 different model projection onto the image plane can assume the very same e_{TE} and e_{RE}, so those errors are still not ambiguity-invariant, as like as the IoU score.

3.1.3 Pose Ambiguity Invariant Metrics

Before introducing the idea of Average Distance for model points let's introduce the concept of *Indistinguishable Poses* and *Invariance to Pose Ambiguity*. As anticipated in the last two sections, IoU score and rotational and translational errors are not invariant to pose ambiguities, so let's see what this concern before introducing how we can overcome this issue.

Indistinguishable Poses

If we take as example a generic object model, call it M, we can define a class of *indistinguishable* poses for this model with the following:

$$[P]_{M,I,\epsilon} = P' : d(v_I[PM], v_I[P'M]) \leq \epsilon \tag{3.8}$$

where $v_I[M] \subseteq M$ is that part of the object model M that is actually visible in the image I, that is not self-occluded or occluded by some other object, d is a distance between the surfaces, and ϵ is a threshold that basically controls the level of detail that we want to achieve in comparing two different views of the same model in the image.

Invariance to Pose Ambiguity

Pose Ambiguity is a necessary property for a 6D object pose estimator. Given an image I, a model M and an estimated pose $\hat{P}$, the error $e(\hat{P}, \bar{P}; M, I)$ is required to be invariant under pose ambiguity iff:

$$\forall \hat{P}' \in [\hat{P}]_{M,I,\epsilon}, \forall \bar{P}' \in [\bar{P}]_{M,I,\epsilon} : e(\hat{P}', \bar{P}') \approx e(\hat{P}, \bar{P}) \tag{3.9}$$

where the approximation is given because of the ϵ tolerance term. A pose error function e that satisfy such property is called *ambiguity-invariant*.

This property is extremely important because an 6D object pose estimator relies its estimates only on the single image, there is no tracking information, and any other spatial relation with the model pose in the image that can distinguish two different pose estimates, so pose ambiguity cannot be removed in any way, that is why we need to model it somehow.

Average Distance (AD)

When the sets of visible points for the model M in the image I is available, one possible solution to estimate how good is the detected pose $\hat{P}$ is to evaluate the average of the maximum of distances between corresponding points of model M for each pose pair $(\hat{P}, \bar{P}) \in Q = [\hat{P}]_{M,I,\epsilon} \times [\bar{P}]_{M,I,\epsilon}$, and finally take the minimum of those average distances:

$$d_h((\hat{R}, \hat{t}), (\bar{R}, \bar{t}); M) = \frac{1}{|M|} \sum\nolimits_{x_1 \in M} \min_{x_2 \in M} \|(\hat{R}x_1 + \hat{t}) - (\bar{R}x_2 + \bar{t})\|_2 \tag{3.10}$$

This distance is very intuitive, but not so effective, and we will see whit an example in the following subsection why take the maximum instead of the average, is a more convenient way.

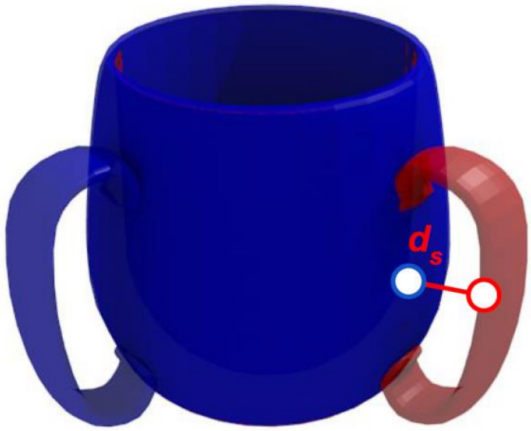


Figure 3.3: **Maximum vs Average Distance.** In this example, it is perfectly shown how taking the maximum instead of the average, is much more convenient. For the mug depicted in the picture, the blue pose (the ground truth), and the red one (the estimation) have very low average distance, while high maximum distance. This explain how maximum distance is more convenient for this case, it reflects better the misalignment between the two poses. The average distance d_h cannot be drawn here, because this distance does not relates with any geometrical points, it's just a score.

Maximum Distance (MD)

As anticipated in the last section, taking the maximum instead of the average, can be more effective, also compared with the task that we want to perform. Imagine to take a mug, like in the example in Figure 3.3, where the Maximum Distance d_s is calculated performing the maximum instead of the average:

$$d_s((\hat{R},\hat{t}),(\bar{R},\bar{t});M) = \max_{x_1 \in M} \min_{x_2 \in M} \|(\hat{R}x_1 + \hat{t}) - (\bar{R}x_2 + \bar{t})\|_2 \tag{3.11}$$

Whit this little modification to the formulation of the distances, the situation completely changes. In the example in Figure 3.3 the distance d_s reflects better the misalignment between the estimated pose and the ground truth. But, is this the right choice? Is this distance reflecting all the necessary when estimating the pose of the mug? In Figure 3.4 there is another possible distance that can reflect better the misalignment between ground truth and estimated pose. It is clearly evident that maybe the new distance d_p, is better that the maximum distance d_s. So we need to find a solution to this problem, and one possible way is to introduce the concept of *equivalence classes*, and then calculate the distances also considering them.

Considers all poses from the equivalence class $[(\hat{R},\hat{t})]$ of the ground truth pose (given by pre-defined symmetries of the object), we can define the new distance d_p with the following:

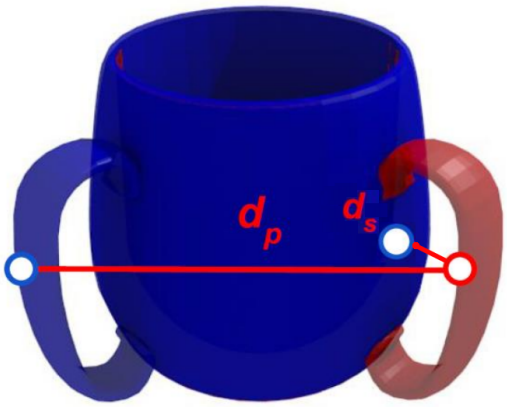


Figure 3.4: **Maximum Distance ambiguities.** In this example it is clearly evident that the distance d_p better reflects the misalignment w.r.t. the d_s one.

$$d_p((\hat{R},\hat{t}),(\bar{R},\bar{t});M) = \min_{(\hat{R},\hat{t})\in[(\hat{R},\hat{t})]} \max_{x\in M} \|(\hat{R}x+\hat{t})-(\bar{R}x+\bar{t})\|_2 \tag{3.12}$$

This new distance reflects better the misalignment between ground truth pose and estimated one, facilitating the evaluation of the pose estimate in tasks like robot grasping and others.

Visible Surface Distance (VSD)

Do we need actually to perform the evaluation of the distance d_p for all the points of the model M. It is actually a very computationally expensive task. That is why we now introduce the concept of *Visible Surface*. The key-point here is to evaluate the distance only on the visible part of the model within the image:

$$e_{VSD}(\hat{P},\bar{P};M,I,\delta,\tau) = avg_{p\in\hat{V}\cup\bar{V}} c(p,\hat{D},\bar{D},\tau) \tag{3.13}$$

In Eq. 3.13, $\hat{V}$ and $\bar{V}$ are the 2D masks of the visible surface of the projected models, with the ground truth pose $\hat{P}$ and the estimated pose $\bar{P}$ respectively. $\hat{D}$ and $\bar{D}$ are the *distance images* obtained by rendering the two models in the camera reference frame. δ is a tolerance threshold used for the computation of the visibility masks, and $c(p,\hat{D},\bar{D},\tau) \in [0,1]$ is the matching cost at pixel p:

$$c(p,\hat{D},\bar{D},\tau) = \begin{cases} \frac{d}{\tau} & if p \in \hat{V}\cap\bar{V} \wedge d > \tau \\ 1 & otherwise, \end{cases} \tag{3.14}$$

More in detail, in Eq. 3.14, the distance image d is given by:

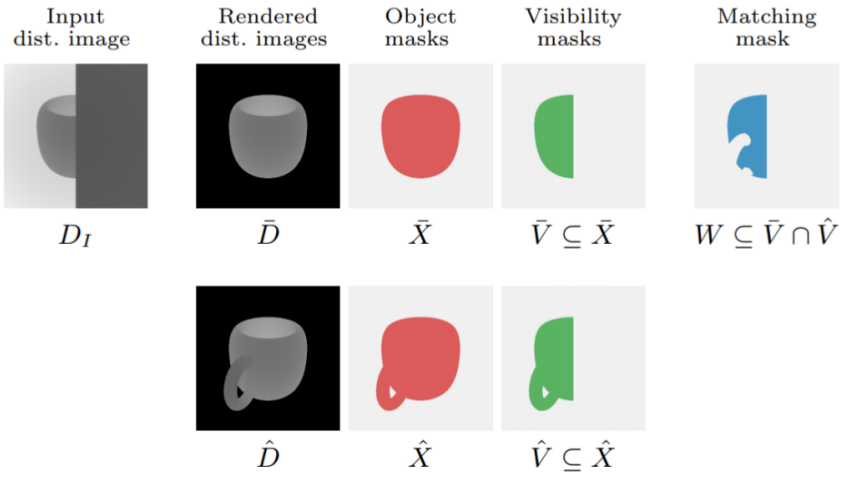


Figure 3.5: **Visibility Masks Example.** An example of how to compute the visibility mask for the mug in the image.

$$d = |\hat{D}(p) - \bar{D}(p)| \tag{3.15}$$

and it can be easily derived from the two depth images of the ground truth and the estimated poses respectively

In order to better understand the concept of visible surface, and the role of the visibility masks $\hat{V}$ and $\bar{V}$ in the equations 3.13 and 3.14, let's take as example the mug depicted in Figure 3.5.

3.2 Industrially Oriented Datasets

After having introduced and formulated the metrics used for comparing object detection and pose estimation measurements, we can now introduce some benchmark tools and data used for industrial applications. Typical settings are those with *texture-less* objects, namely, objects that do not present any kind of texture. Given this property, object detectors and pose estimators can rely their estimations only on shape based approaches, without any sort of rbg texture based techniques.

Examples of texture-less datasets are the *T-LESS* Dataset [5] from the Center for Machine Perception of the Czech Technical University of Prague and the *MVTec ITODD Dataset* [2] from the same company that distributes the Halcon Libraries.

Figure 3.6: **T-LESS scenes examples.** A compact view of the scenes present in the T-LESS dataset.

3.2.1 T-LESS Dataset

The T-LESS Dataset [5], has the name suggests, is a publicly available dataset[1] composed by thirty different texture-less objects. All the objects in the dataset are from the industrial scenario and do not present any particular feature, moreover, most of them are also very similar in shape and color.

The dataset is composed by training and test images, the former are represented by RGB and RGB-D images of multiple views of the single object, while the latter are composed by 21 test scenes where the objects are collected together in more cluttered and unstructured forms. All the scenes, both training and test ones, have been acquired with a fixed sensor setup composed by the following RGB and RGB-D cameras:

- Primesense Carmine 1.09 (Short Range) that provides registered RGB-D images with the following features: RGB:1280 $\times$ 1024 pixels, Depth:640x480 pixels;
- Microsoft Kinect v2 that provides images with registered depth with 1920 $\times$ 1080 pixels of resolution;
- Canon IXUS 950 IS high resolution camera that provides RGB images of 3264$\times$2448 pixels of resolution.

All the three sensors have been mounted on a fixed structure and a rotating turntable with the objects on top has been built for positioning and rotating the training and test scenes. Examples of acquired scenes are in Figure 3.6.

[1] http://cmp.felk.cvut.cz/t-less/index.html

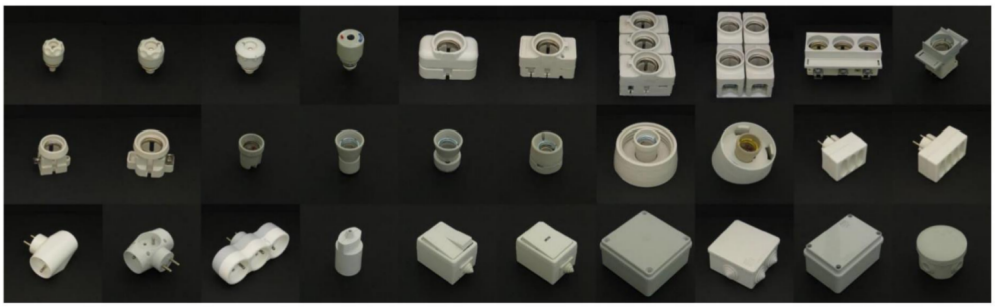


Figure 3.7: **T-LESS Objects Samples.** A compact view of the thirty object classes present in the T-LESS dataset.

In the T-LESS data packages also the CAD models of the objects are given to the public. In particular they released two different kind of CAD models:

- Manually created CAD models;
- Automatically reconstructed CAD models.

All the objects presented in the T-LESS dataset are also shown here in Figure 3.7. As like as for our RAW dataset, they organized their objects in multiple scenes, 21 as already mentioned, classified by order of complexity: *easy* and *hard* objects scenes. Each scene has been accurately annotated, and all the objects in it have been localized following the approach explained below:

1. A dense 3D model of the scene was reconstructed with the system from [16]. This was accomplished using all 504 RGB-D images of the scene along with the sensor poses estimated using the turntable markers;
2. The CAD models were then manually aligned to the scene model;
3. To increase accuracy, high-resolution images from Canon camera sensor were used to refine manually all the detected misalignment;
4. The final poses were distributed to all the test images with the aid of the known camera-to-turntable coordinate transformations.

3.2.2 MVTec ITODD

MVTec Industrial 3D Object Detection Dataset, MVTec ITODD [2] in code, is a public dataset for 3D object detection and pose estimation also related to industrial setups, like the previously mentioned T-LESS dataset. The dataset contains 28 object classes, disposed in 800 different scenes, all labeled with 3D rigid transformations with respect to the camera sensors.

The sensors used during the acquisition of the scenes are composed as follow:

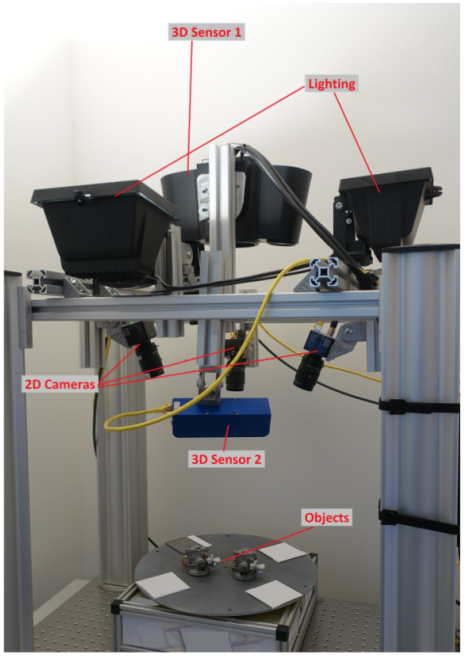


Figure 3.8: **MVTec ITODD Sensor Setup.** In the picture there is a compact view of the fixed sensor setup used for the acquisition of the MVTec ITODD dataset scenes.

- High-Quality 3D sensor: a multi-shot 3D stereo sensor that provided both a depth image and a grayscale image with the same viewpoint of the previous one. This sensor also provided high-quality 3D reconstructed cloud of the scene by using multiple random projected patterns and a space-time stereo approach for reconstructing the scene with an accuracy of around 100 μm;
- Low-Quality 3D sensor: very similar to the previous sensor but with a shorter baseline, a wider field of view and less shots per scene. Given those features, the 3D reconstructed scene is much more noisy than the previous one, with an accuracy of around 1-2 mm;
- 3 High.Resolution Cameras: they provided grayscale images, and the scenes where captured twice, one time with the random patterns and the last time without.

In Figure 3.8 there is a compact view of the fixed sensor setup used for the acquisition of the scenes in the dataset.

As like as for the T-LESS dataset and the RAW Dataset that we are going to present in the next chapter, the object in the MVTec ITODD dataset are texture-less objects, that is for remarking again the high orientation to the industrial settings of these kind of works. All the object classes used in the MVTec ITODD dataset are depicted in Figure 3.9.

The ground truth 3D pose of all the objects in the MVTec ITODD scenes have been annotated using a T-LESS similar approach. It consists in a semi-manual procedure

Figure 3.9: **MVTec ITODD Object Classes.** All the 28 object classes of texture-less object used during the acquisition of the MVTec ITODD dataset scenes.

based on 3D reconstruction and ICP alignment. Once the pose in the first view of the scene has been labeled, this pose has been propagated to all the other view of the scenes by means of the rotating turntable with markers on top.

Chapter 4

The RAW Dataset

In this chapter the RAW dataset is going to be presented. This dataset has been produced as a requirement of the FlexSight[1] project. FlexSight is a research project founded by the European Community's project ECHORD++[2]. This project involves three main partners: the Ro.Co.Co. Laboratory[3] of DIAG (Department of Computer, Control, and Management Engineering Antonio Ruberti at Sapienza University of Rome), which provides expertise in software design and development for robotic perception applications; IT+Robotics[4], which provides expertise in the industrial robotics domain; and Robox[5], a company specialized in the development of innovative hardware and software solutions for robotics and control systems.

More details about the FlexSight project can be found in the Appendix A.1 of this thesis.

4.1 RAW Dataset Features

As like as the previous datasets presented in sections 3.2.2 and 3.2.1 also the RAW Dataset has been built for industrial scenarios. The main scope of the RAW Dataset is to build a large and consistent test bed for texture-less object detection and localization algorithms. The name of the dataset is related to the RoboCup@Work competition, and it contains all the objects used in the robotic competition plus other objects from other past robotics events, such as the RoCKIn@Work[6] (Robot Competitions Kick Innovation in Cognitive Systems and Robotics) and ERL[7] (European Robotics League).

As already mentioned, all the objects present in the dataset are texture-less objects and are inspired to the industrial settings, e.g. parts of a motor, bearings, nuts, motor axis and others. A compact view of all the objects present in the dataset is given in Figure 4.1, and the complete list of objects with their specific name and class is reported in Table 4.1.

[1] http://www.flexsight.eu/
[2] http://echord.eu/
[3] http://www.dis.uniroma1.it/ labrococo/
[4] http://www.it-robotics.it/
[5] http://www.robox.it/en-US/
[6] http://rockinrobotchallenge.eu/
[7] https://www.eu-robotics.net/robotics_league/index.html

Name	Class	Description
F20_20_B	0	Black small aluminum profile
F20_20_G	1	Gray small aluminum profile
M20	2	Small nut
M20_100	3	Heavy large screw
M30	4	Big nut
R20	5	Plastic tube with shaped borders
S40_40_B	6	Black large aluminum profile
S40_40_G	7	Gray large aluminum profile
V20	8	Plastic tube without shaped borders
Bearing_box	9	Small motor box for bearings
Bearing_box_B	10	Small motor box for bearings (Type B)
Motor	11	Black plastic reconstruction of a motor
Axis	12	Aluminum motor axis
Distance_tube	13	Small aluminum ring
Bearing	14	Real bearing for motors
Cover_plate_NOHOLE	15	Thin aluminum plate with hole
Cover_plate_HOLE	16	Thin aluminum plate without hole
Cover_plate_BOX	17	Black plastic box for cover plates
Bearing_box_BOX	18	Black plastic box for bearing boxes

Table 4.1: **The RAW Dataset Objects.** In this table is reported the list of all the objects present in the RAW dataset, with their name, class and a brief description.

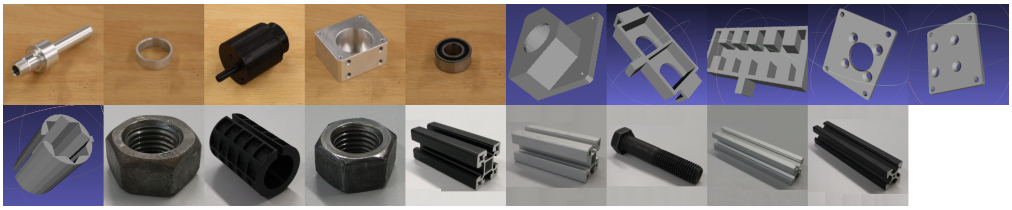


Figure 4.1: **RAW Dataset Objects.** A compact view of all the 19 classes of objects present in the RAW Dataset.

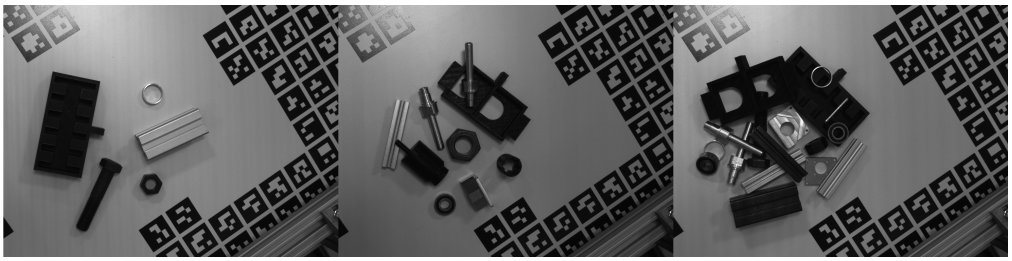


Figure 4.2: **RAW Dataset Scenes Comparison.** From left to right: easy scene, medium scene and hard scene. All the images refers to the camera left of the FlexSight sensor and are without the projected laser pattern.

The main features of this dataset are the following:

- **19 industry relevant objects**: no discriminative color, no texture, often similar in shape;
- **3 different sensors**: FlexSight sensor, Microsoft Kinect2, Intel RealSense SR-300;
- **Test images (7K from each sensor)**: originated from 15 test scenes. The scene complexity varies from simple scenes with several isolated objects to very challenging ones with multiple object instances and a high amount of clutter and occlusion. Images include: RGB and Depth for Kinect2 and RealSense SR-300; 2 grey-scale images (with and without projected laser pattern) for the FlexSight sensor;
- **19 object 3D CAD models**: all the objects have a specific 3D CAD model.

Each image of the dataset comes with ground truth estimate of all the objects present in the scene. The full 3D position in space, rotation plus translation with respect to the camera frame, has been annotated with a specific protocol that involved both manually intervention of the user by means of our own developed labeling tool and some other automatic procedures for propagating the pose in a given reference frame to all the other images of the dataset.

4.2 Details of the Scenes

As already anticipated, the RAW Dataset comes with 15 different scenes. In particular, those scenes are aggregated in three different categories: *easy*, *medium* and *hard*. This categorization depends on the level of complexity of the disposition of the objects in the scenes. An example of easy medium and hard scene comparison is given in Figure 4.2. Moreover, easy scenes does not contain any occlusion or overlapping of the objects, while medium and hard scenes do.

Each scene is composed as follow:

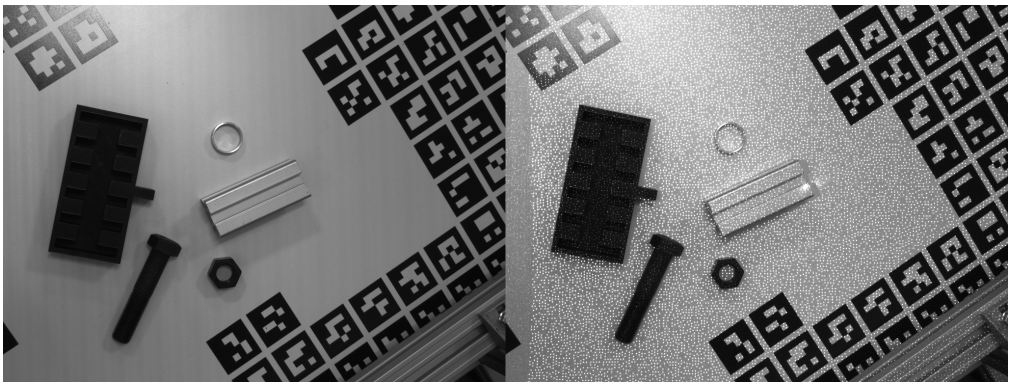


Figure 4.3: **FlexSight Sensor Acquisition (with and without laser pattern).** On the left the FlexSight Sensor image acquired without projected laser pattern, on the right with projected laser pattern.

- **FlexSight Sensor Data**: 465 images, both for camera left and camera right of the sensor. Each image has been acquired with and without the projected laser pattern (See Figure 4.3).
- **Microsoft Kinect 2**: 465 rgb-d images. We provide both rgb and depth image, together with the automatic generated registered point cloud.
- **Intel Realsense SR300**: 465 rgb-d images. For the Intel Realsense SR300 only rgb and depth images are provided.

4.3 Setup and Sensors

As already mentioned in the introduction if this chapter, the RAW Dataset has been acquired in fulfillment of the data acquisition requirement of the FlexSight project. For this scope, a completely new and custom robot-sensors setup has been built in collaboration with the two other partners of the project, IT+Robotics and Robox. This setup consists in a custom built cell with a robotic manipulator inside, which has the experimental sensor mounted at its end-effector. A 3D model view of the robotic cell is given in Figure 4.4.

In the following sections the robotic arm and all the sensors will be explained in detail.

4.3.1 The Robotic Arm

The robotic arm used for the acquisition of the RAW Dataset is a 6 d.o.f. robotic manipulator. It has been provided by Robox as contribution to the FlexSight project.The company also provided their own control system with which we interfaced via serial communication for positioning and sending commands to the robot during the acquisition procedure. A picture of the robotic manipulator is given in Figure 4.5.

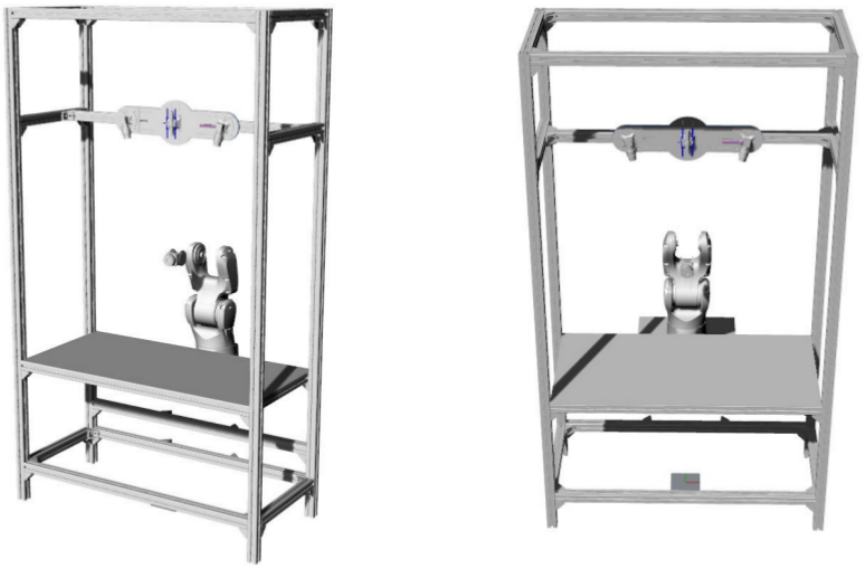


Figure 4.4: **FlexSight Robotic Cell 3D Model View.** A 3D reconstruction of the custom robotic cell developed and built for the RAW Dataset acquisition. The figure just reports the robotic cell with the first prototype of the FlexSight sensor mounted on the top and a fake robotic arm as placeholder for the real one.

4.3.2 The FlexSight Sensor

The main contribution of this thesis project was the acquisition of a novel dataset with a novel sensor, that provides both high-resolution images for active and passive stereo vision. This sensor, we named it FlexSight Sensor is composed by 2 main components:

- A stereo camera setup, composed by two high resolution grayscale cameras. Each camera provides high-resolution grayscale images, in particular: 2048×1536 pixels of resolution.
- A pseudo-random laser projector. This was involved to achieve active stereo capabilities.

A detailed view of the FlexSight sensor is given in Figure 4.6. The sensor has been mounted on a custom support at the end-effector of the robotic manipulator. Moreover it presents a *Arduino MEGA* controller used for triggering the two cameras and the laser projector in the same moment in order to achieve precise and accurate image acquisition. Both the cameras and the laser projector have been powered by 5V DC adapter which provides stabilized and regular DC current to each device.

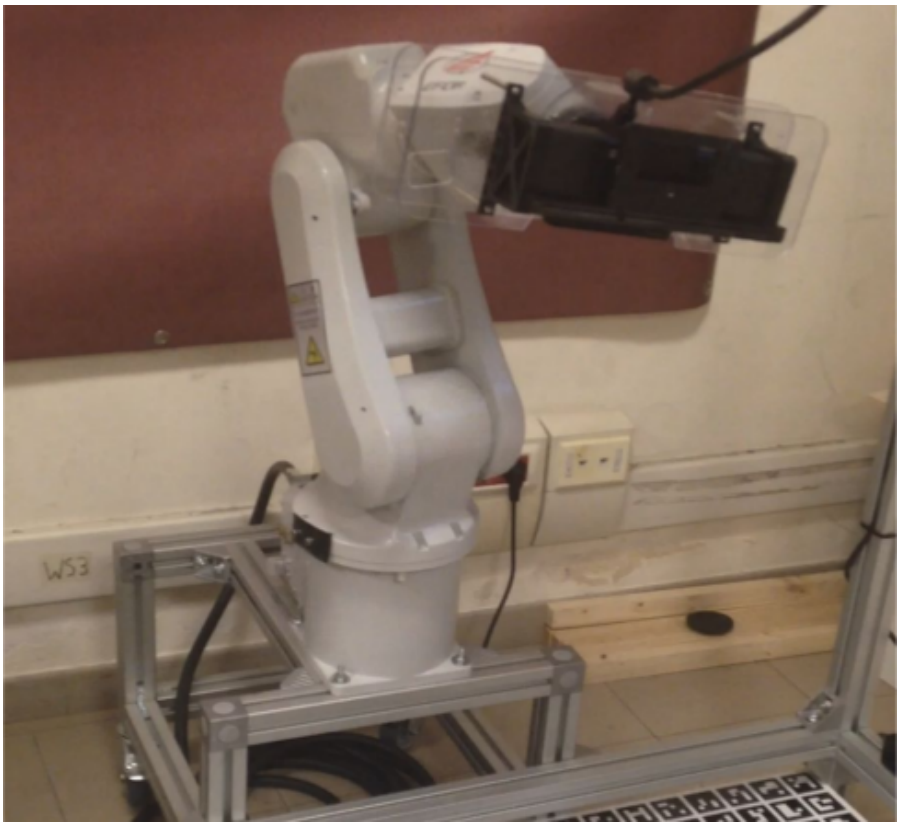


Figure 4.5: **FlexSight Robotic Manipulator**. In the picture, the 6 d.o.f. robotic manipulator used for the acquisition of the RAW Dataset scenes.

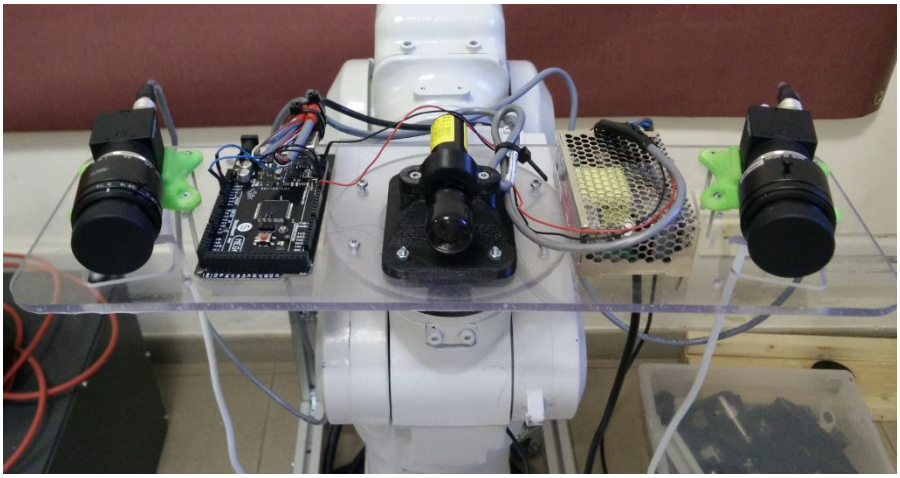


Figure 4.6: **FlexSight Sensor Detail**. From left to right: right high-resolution point gray camera, arduino controller, pseudo-random laser projector, AC adapter, left high-resolution point gray camera.

4.3.3 Microsoft Kinect 2 and Intel Realsense SR300

On top of the FlexSight sensor, two commercially available 3D sensors have been mounted, namely a Microsoft Kinect 2 RGB-D sensor and a Intel Realsense SR300. Both the commercial sensors provided already registered depth and rgb images at low and high resolution respectively.

- **RGB images**: 1920×1080 pixels for both the sensors.
- **Depth images**: 512×424 pixels for Microsoft Kinect 2 and 640×480 pixels for the Intel Realsense SR300.

Moreover, the two sensors have been chosen because they provide different features and can be of major interests for other commercial and common usages. In particular, the Intel Realsense SR300 is a so called *short range* sensor, in the sense that its depth estimation is accurately working in short range, namely $0.2m$ to $1.5m$. While the Microsoft Kinect 2 provides wider range, namely $0.8m$ to $4.5m$.

4.4 Acquisition Procedure

All the scenes of the RAW Dataset have been acquired following a rigid and specific protocol. In particular, we have defined a set of camera poses by sampling a set of different semi-sphere in the range of $(85cm, 95cm)$ from the surface where the objects are placed. From the set of all the extrapolated poses we then developed a automatic collision-check procedure that eliminated all the poses where the robot body and the experimental FlexSight Sensor collide with the any of the elements of the cell.

At the end of the first automatic pose check procedure, more than 500 poses have been selected, then we manually tested all those poses and manually double checked all obtaining at the end 465 different camera poses.

With the list of all the poses and the robot and sensor setup safely working we started the acquisition of the 15 scenes of the dataset. The acquisition of each single scene was divided in two separated steps:

- **FlexSight + Realsense Acquisition**: in this phase we only acquired images and depth images from the FlexSight sensor and the Intel Realsense SR300.
- **Kinect 2 Acquisition**: in this phase we only acquired images and depth images from Microsoft Kinect 2.

This separated sequence was decided in order to solve the laser interference among the two commercial sensor, the Microsoft Kinect 2 and the Intel Realsense SR300. Since those sensors are basically structured light based sensors, they work with internal laser patterns projectors, and they basically work in the same frequency ranges, that is why

Figure 4.7: **FlexSight Sensor Acquisition Sequence**. From left to right: the acquisitions with $800\mu s, 2400\mu s, 10000\mu s$ and $52000\mu s$ respectively.

we experienced some interference in the depth images of both and decided to completely turn off the other sensor while the first was performing the acquisition.

For the FlexSight sensor we also split the acquisition in two different phases, namely with and without the projected laser pattern. Moreover, for each frame acquisition we took 4 different images that differs only for exposure time of the two cameras, namely $800\mu s, 2400\mu s, 10000\mu s$ and $52000\mu s$. In this way we can provide to all the future users of the RAW dataset the possibility to work both with the individual acquisitions or performing some HDR image manipulation using all the 4 acquisitions. An example of acquisition sequence from the FlexSight sensor is given in Figure 4.7.

4.5 Ground Truth Estimation Protocol

Having a good estimate of the ground truth 3D position of each object in the scene is a central point for our RAW Dataset. We tried to achieve high level of accuracy in the ground truth estimation by establishing a rigid and fixed protocol. This protocol is composed by different phases that consist both in manually and automatic procedures. The steps can be schematized as follows:

1. **Manual estimation of the first initial guess**: In this step the user is asked to manually position the rendered object model to the first initial guess. This step is performed using our own developed labeling tool;

2. **Automatic 3D pose optimization**: In this step a accurate and fine optimization of the initial guess is performed. Once the user has selected its desired initial guess for the object model, an automatic optimization pipeline is performed using both single and multi-view information fusion. This step is performed using the aforementioned labeling tool and using the algorithm proposed in section 2.4.2, with an expressly tuned parameterization of its multi-view implementation. In particular, for the FlexSight sensor, both the left and right images are used for optimize the pose of the object, while, for the other two commercial sensors, different views of the same scene have been used to accomplish this step.

3. **Automatic object pose propagation**: Having the pose of the object in the first frame of the scene is the starting point for the propagation to all the other frames. All the objects of the scene have been specifically positioned within a plane where

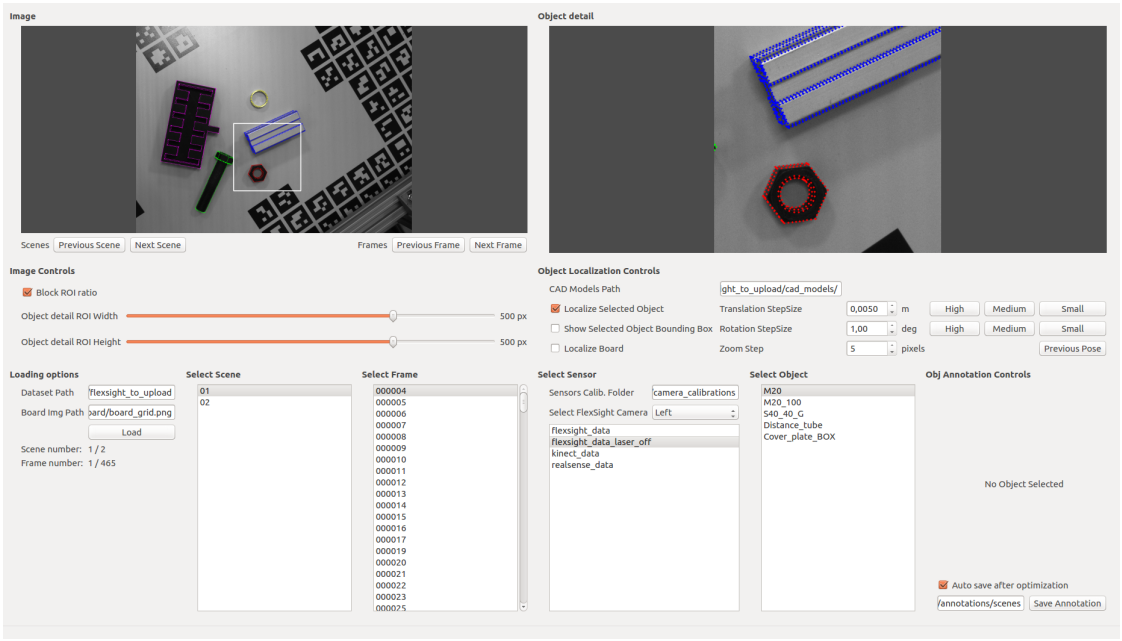


Figure 4.8: **Labeling tool software detail**. In the left image there is the representation of the selected scene, on the right one there is the detail of the area where the user is focused on. As the user labels the object pose, it is drawn on the two images.

a custom Aruco markers board has been printed. This board acts as our world reference frame, so once the board is detected in all the other frames, the new object position is directly derived with a simple transformation.

Further details on the labeling tool and the results of the pose propagation approaches will be given in the next sections.

4.5.1 Labeling tool

In order to reduce the labeling time, a dedicated tool has been developed. In Figure 4.8 there is a detailed view of the GUI that we developed. As it can be seen from the picture, many options are given to the user both to manually select the scene and the frame to annotate and to automatically refine the position of the desired initial guess.

Many useful tools have been implemented, in particular, there is the possibility to always highlight the position of each labeled object and of the world reference frame given by the automatic detection of the Aruco board on the plane. An example of board detection performed within the labeling tool GUI is given in Figure 4.9. Other useful feature is the possibility to show the bounding box of each labeled object, see Figure 4.10.

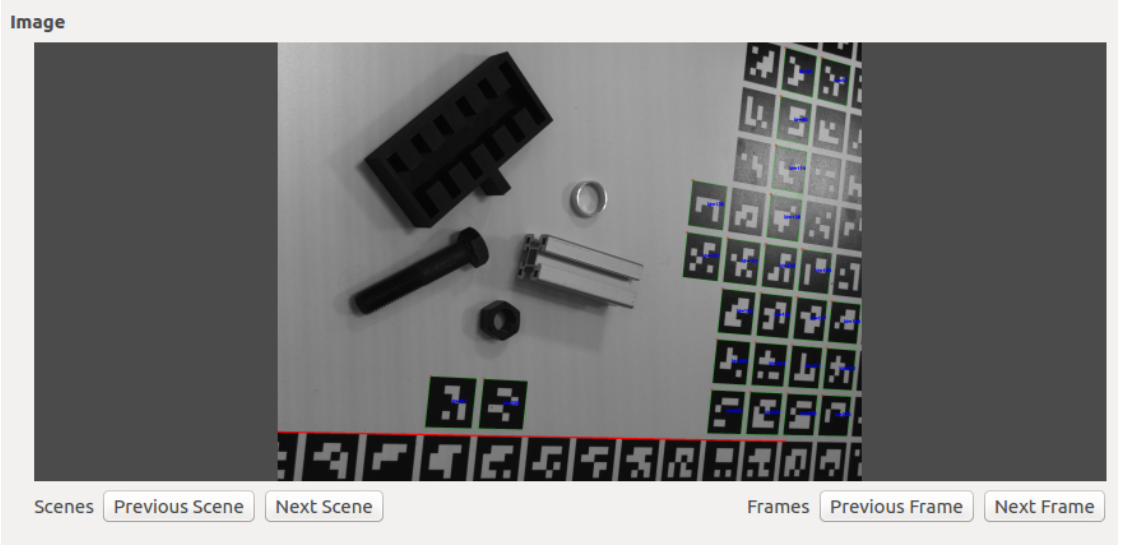


Figure 4.9: **Labeling tool Aruco board detection**. The tool automatically detects the Aruco board, and the user is able to draw it on the images at any time.

Figure 4.10: **Labeling tool Bounding Box feature**. The tool gives the possibility to draw the bounding boxes of all the labeled objects.

4.5.2 Pose Propagation Pipeline

The design of the Aruco board has been specifically thought in order to let the labeling procedure as much rapid as possible. Having a clearly visible structure in all the images of the dataset is very convenient in order to automatically replicate, namely propagate, the object poses from one image to the other.

As already anticipated in the introduction of this section, the last step of our ground truth estimation protocol is the object pose propagation. Here we can distinguish two different ways of propagation of the object pose:

- **Sensor to Sensor**: here the pose is propagated from one sensor, e.g. the FlexSight camera left, to another sensor, e.g. the FlexSight camera right.
- **Frame to Frame**: here the pose is propagated from one single image acquisition, e.g. the first image of the scene, to another image of the same scene.

The two propagation techniques basically differ in terms of the used approach: in Sensor to Sensor pose propagation we used the relative transform of the two sensors in order to propagate the pose of the object. The transformation has been computed accurately calibrating the relative position of each sensor that composes our experimental sensor setup by using our own developed calibration tool; in Frame to Frame pose propagation instead we only use the information derived by detecting the Aruco board and basically propagate the manually labeled object pose in the first image of the scene to all the other images of the same scene.

Given that the Aruco board was not always perfectly visible in all the images of the scenes, its detection cannot be taken as always reliable. For this reason we decided to give more importance to the first pose propagation approach and use the second one just as a comparison for the results.

Chapter 5

Experiments

In this chapter we will exploit in detail some experiments performed over the RAW Dataset that has been presented in Chapter 4. In particular we will first focus on some object detection experiments performed both with standard shape based approaches as described in Sec. 2.4.2 and deep learning based approaches as described in Sec. 2.4.3. Moreover, we will exploit the results of our implementation of the SGM algorithm described in Sec. 2.3.1 comparing its results with the 3D scenes coming directly from the two commercial sensors in our sensor setup, namely the *Microsoft Kinect 2* and *Intel Realsense SR300*.

5.1 Object Detection

As already introduced in Sec. 2.4, object detection is basically the task of computing the 2D bounding box of a given object within an image. Bounding box is essentially a portion of the image that contains the object. Such basic information is extremely important in robotic perception, especially in robotic vision, where the ability to detect objects in different situations is a crucial point. Our tests basically rely on testing the performances of two different kind of approaches, namely shape based and deep learning.

The so called shape based method is based on classical template matching techniques (see Sec. 2.4.2): in this thesis, we used a state-of-the-art, commercial implementation provided by the Halcon library: this implementation provide cutting edge results out-of-the-box, and it is highly customizable, so many problems can be tackled with specific and accurate approaches. As Deep Learning based approach, we selected a very recent and effective network presented in [13] and [12]: for further details, refer to Sec. 2.4.3.

We remark here that actually the Halcon library shape-based matching implementation is able to provide the *full* 6D pose of the searched object. Moreover, given the no availability of general deep learning solution to the localization problem, namely to retrieve the 6D pose of a given object, we performed tests only in terms of 2D bounding box object detection.

The tests have been performed on a subset of the RAW Dataset, since the availability of the ground truth for the entire dataset has not been achieved yet, moreover it is not part of the scope of this work. The subset of the RAW Dataset is anyway enough large to permit consistent and coherent statistical evaluations, and it is composed as follows:

- **2 complete scenes**: the tests have been performed using just the first two scenes of the dataset, and just with respect to the data obtained from the experimental FlexSight Sensor, as its performance evaluation is a consistent and important part of the FlexSight project (See Appendix A.1);
- **Almost 4 thousand of images**: the two scenes contains both camera left and camera right of the FlexSight Sensor, both with and without projected laser pattern;
- **All the images are labeled**: all the considered images have are accompanied with the relative ground truth 3D position of each object and the relative bounding box as well;
- **5 Different Object Classes**: the involved images only contain 5 object classes of the RAW Dataset, namely: *Distance_tube*, *M20*, *M20_100*, *Cover_plate_BOX*, *S40_40_G*;
- **Synthetic data generation**: for the deep learning approaches, data augmentation has been employed. In particular we will see in the following subsections how we trained the YOLO deep neural network described in Sec. 2.4.3 with 2 different version of the RAW Dataset subset, namely the first is the original one and the second is a synthetic version of it.

The results will be organized by class of the object and will be expressed in terms of IoU accuracy over the detected bounding box.

5.1.1 Experiment Setup

Shape Based Approach - Halcon object detection

In this experiment, we performed both object localization and detection since the core of the object detection pipeline of the Halcon libraries is essentially based on 3D object localization on 2D images, then given the 3D position of the object w.r.t. the camera frame, it is projected onto the 2D image and the bounding box of the object is extracted. Examples of this procedure is given in Fig. 3.2. Moreover, given the highly customization level that those library has, we investigated object detection performances by considering multiple settings:

- **Single object candidate detection**: Only the object with the highest score is considered;
- **Multiple object candidates detection**: We analyzed both the performances using in sequence: *5 candidates*, *4 candidates*, *3 candidates* and *2 candidates*.

Parameter	Value
batch size	64
height	416
momentum	0.9
decay	0.0005
learning rate	0.0001
max batches	15000
policy	steps
steps	100, 8000, 10500

Table 5.1: $YOLO_O$ **training parameters.** In the table are reported the most significant parameters that we tuned in order to train the $YOLO_O$ network. All the others are assumed as with their default value.

Deep Learning Approach - YOLO

Deep learning is become one of the prominent and more prolific area of research in the last decade, and latest techniques and developments have actually now reached the state-of-the-art in many tasks. Object detection is one of them. That is why we wished to test some of those techniques, e.g. YOLO deep neural network, and compare its performance w.r.t. the technique used in the last experiment. In this experiment we are going to show multiple results of YOLO object detection capabilities and explain how we achieved higher levels of IoU Avg. detection rate w.r.t. the Halcon libraries.

The experiment has been conducted using almost the very same settings of the previous experiment, with just few modifications in order to be compliant with this totally different approach. In particular, given the data driven nature of this approach we must subdivide the dataset in two different sets: *training set*, *evaluation set*. In particular we used 95% of the initial subset as training set, and the remaining 5% as validation set. The high disparity among the two set sizes is simply because such approaches need lot of data in order to be trained successfully, that is we gave almost all the subset to the training set. The validation set at the end is big enough to obtain reliable results (~ 100 images). From now on, we will refer to this version of the dataset as $YOLO_O$, as it is directly built from the original RAW Dataset, and just adapted to the YOLO format. The same acronym will be given to the related network as well.

The training of the $YOLO_O$ network has been conducted using the set of parameters given in Tab. 5.1. With the very same set of parameters we performed another train of the YOLO network. In particular we used a *synthetic* version of the RAW Dataset, from now on we will refer to it as $YOLO_S$, built by manipulating the same training images used for $YOLO_O$'s training. The basic idea was to change the background of all the images with random images downloaded from the publicly available set of images at the following site: `https://snippets.khromov.se/stock-photo-archive-zip-77-images/`. Some example of images realized with such technique, are given in Fig. 5.1.

Figure 5.1: **Images from the synthetic RAW Dataset**. In the figure, some examples of the images produced for training the $YOLO_S$ network.

5.1.2 Experiment Results

Shape Based Approach

As already anticipated, each result will be presented differentiating for each class of the considered objects. The Tab. 5.2 show the accuracy results in terms of Intersection over Union (IoU) for each of the involved object, look at the section referred to Shape Based Approach. In Tab. 5.3 is also reported the percentage of the *5cm,5deg* criteria satisfaction, this criteria has been introduced in Sec. 3.1.2. In particular we reported the scores in terms of $\frac{num\_correct\_locs}{num\_imgs}$, where $num\_correct\_locs$ is the total number of time the localized pose has satisfied the aforementioned criteria, while testing on num_imgs total images.

As expected, the performances slightly increase when including in the search procedure the highest number of candidates. That is essentially due to the fact that not always the first best candidate, namely the object with the highest score, is exactly the object that we are looking for. If we compare more than one candidate with the ground truth bounding box we find that for some classes of objects, like the *M20_100*, only when we consider 5 candidates we achieve to detect the actual object, and even in such cases, the one that has the highest score is not the right object (See. Fig. 5.2 for a better visualization of the described problem). Moreover, from the results in Tab. 5.3 it is clearly visible that some object classes, like the *Cover_plate_BOX* are basically never

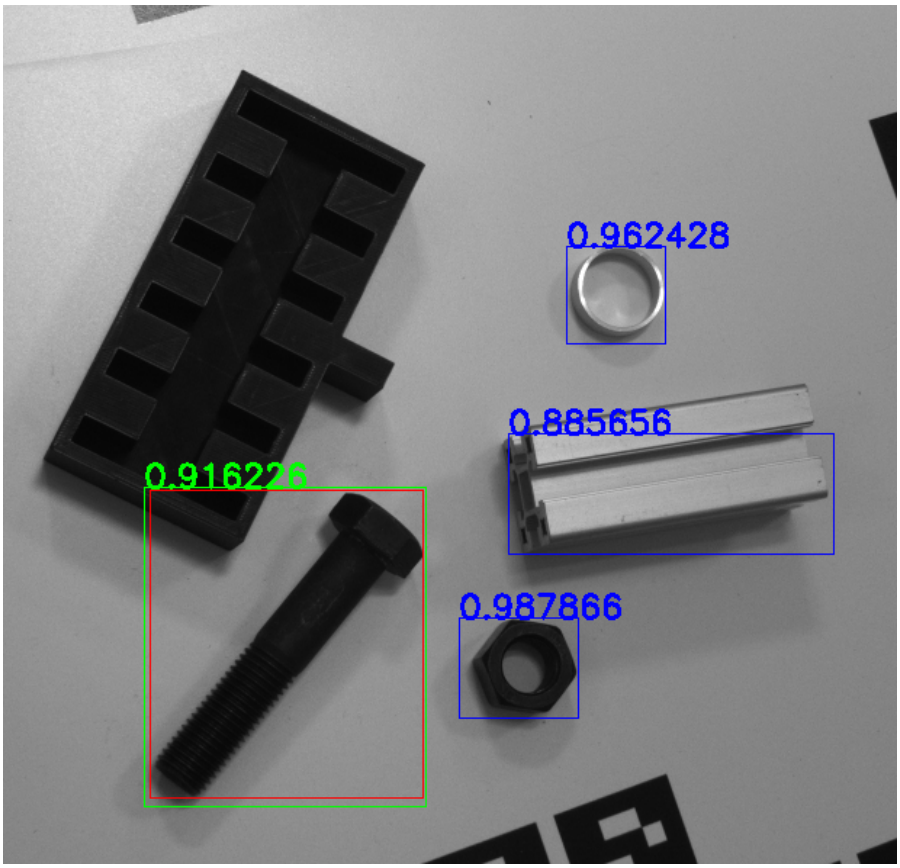


Figure 5.2: **Best Candidate Problems with Halcon Libraries**. The picture depicts an example of erroneous best candidate detection. The candidate that actually refers to the real M20_100 object is not the one with the highest score. In a single candidate approach the detection would have failed.

	Shape Based					Deep Learning	
Obj Class	*1Cand*	*2Cand*	*3Cand*	*4Cand*	*5Cand*	$YOLO_O$	$YOLO_S$
Distance_tube	0.862	0.862	0.862	0.862	0.862	0.897	0.825
M20	0.744	0.744	0.746	0.746	0.746	0.901	0.816
M20_100	0.000	0.000	0.475	0.475	0.797	0.944	0.890
Cover_plate_BOX	0.799	0.799	0.799	0.799	0.799	0.954	0.921
S40_40_G	0.809	0.809	0.809	0.809	0.809	0.939	0.829

Table 5.2: **IoU Average results**: the table reports the results in terms of IoU Average (Intersection over Union Average) of all the tested methods. Results are reported by object class organized in columns by Shape Based Approach and Deep Learning Approach. Shape Based Approach results are reported by number of candidates used for the detection.

	Shape Based				
Obj Class	*1Cand*	*2Cand*	*3Cand*	*4Cand*	*5Cand*
Distance_tube	1.0	1.0	1.0	1.0	1.0
M20	0.83	0.83	0.83	0.83	1.0
M20_100	0.0	0.0	0.83	0.83	0.83
Cover_plate_BOX	0.0	0.0	0.0	0.0	0.0
S40_40_G	0.5	0.5	0.5	0.5	0.67

Table 5.3: **5cm,5deg criteria evaluation**: The table reports the percentage of the *5cm,5deg* criteria satisfaction. In particular here we reported the scores in terms of *num_correct_locs*/*num_imgs*, where *num_correct_locs* is the total number of time the localized pose has satisfied the aforementioned criteria.

well localized, even if this goes in contrast with the results of IoU score in Tab. 5.2. This means that the pose does not satisfy the *5cm,5deg* criteria, but the bounding box is somehow acceptable, and this can only be explained by the fact that the shape based approach basically always detected the *Cover_plate_box* flipped, so the bounding box is correct while the pose is completely flippet w.r.t. the ground truth position.

Deep Learning Approach

In the **Deep Learning** section of Tab. 5.2 are reported some results of the test performed with the two networks, namely $YOLO_O$ and $YOLO_S$. The results have been arranged in the same way as the experiment shown before in order to give a comprehensive and consistent way for comparing the two approaches. It is clearly evident that the Deep Learning approaches overcome the shape based ones by more than $\sim 10\%$. The main reason why we performed also this other kind of training is that the first network, the $YOLO_O$, seemed to be overfitting the dataset used for training, and the reasons are essentially two:

- **Little size of the training set**: We used only a subset of the overall RAW

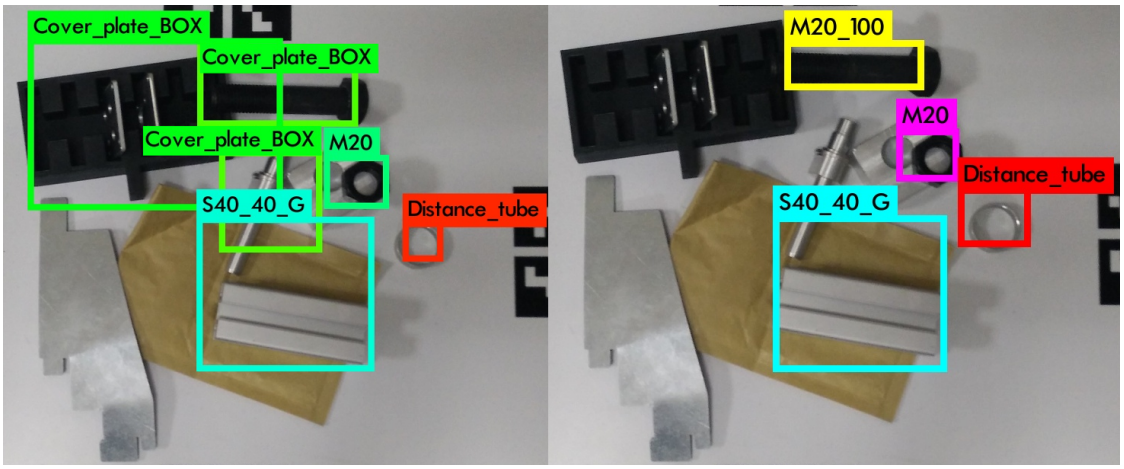


Figure 5.3: **Detection Results on totally different scene**. The scene depicted in the image is taken out of the RAW Dataset, is not part of it essentially. Was built for testing $YOLO_O$ and $YOLO_S$ performances out of the training and test set. On the left there is the result of $YOLO_O$'s detection, on the right the $YOLO_S$'s one. It is clearly evident that $YOLO_S$ is much more robust than the first one. So results show that we overcome a bit the problem of overfitting with the network trained with the standard RAW Dataset.

Dataset, and maybe they are too few images for successfully such a big network (more than 20 convolutional layers + fully connected layers at the end);

- **Too similar images**: images used for the training phase are extremely similar, even if taken from completely different views. Illumination conditions are almost always the same, background is always the same, and so on.

We basically claim that such configuration brought the $YOLO_O$ network to overfitting a bit the training set, and clearly evidence of this is given by the result of the detection performed on very different scenes, with objects in a much more cluttered environment. Look at Fig. 5.3 for better understanding the problem. Results show that we overcome a bit the problem of overfitting with the network trained with the standard RAW Dataset.

5.2 3D Reconstruction

As already anticipated, this section is going to cover the part of 3D scene reconstruction. This will mainly focus on the reconstruction capabilities of our experimental sensor, the *FlexSight Sensor*. The sensor is essentially a stereo camera system with active capabilities. This feature is given by the introduction of a laser projector that projects on the scene a pseudo-random pattern that helps 3D reconstruction techniques essentially by adding more structure to the scene. An explicit example of scene frame acquired with and without laser pattern is given in the previous chapter, refer to Fig. 4.3 for further details.

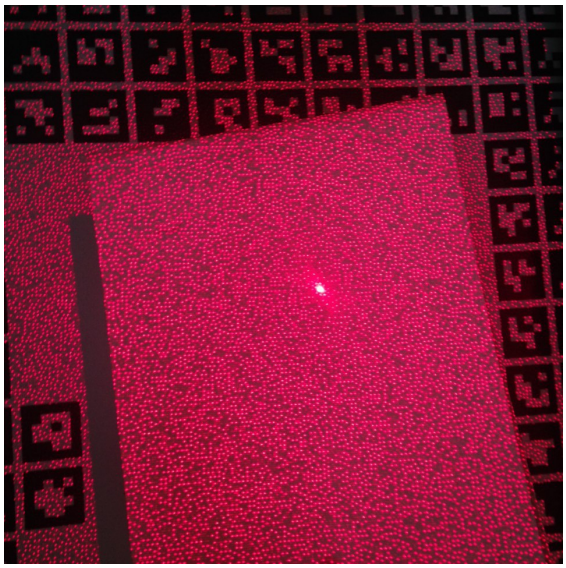


Figure 5.4: **Zero Point Problem**. This problem basically consists in a higher concentration of illumination in the zero point of the pattern, namely the center. The result is that not all the points of the pattern have the same illumination intensity, in fact, the central point is exactly the one that creates more *noise* in the pattern.

Although the introduction of a pseudo-random pattern helps the reconstruction since it introduces extremely structured information within the image, many laser projectors, like in our case, may suffer from the so called *zero point* problem. This problem basically consists in a higher concentration of illumination in the zero point of the pattern, namely the center. The result is that not all the points of the pattern have the same illumination intensity, in fact, the central point is exactly the one that creates more *noise* in the pattern, an example of this problem is reported in Fig. 5.4.

Anyway, by performing *HDR (High Dynamic Range)* processing, the problem can be somehow mitigated, in the sense that this technique basically tries to uniform the level of light intensity over the all image.

With our own implementation of the algorithms and techniques already described in Chapter 2, at Sec. 2.3, we were able to reconstruct the 3D scene using the active stereo camera setup of the FlexSight Sensor. In the rest of this section we will give a qualitative summary of the results. A more explicit and quantitative evaluation of reconstruction performances will be tackled as a future work, since this will also be the last step of the FlexSight project development.

5.3 Discussion

In the context of this work we find more interesting to focus the point of the discussion on the results obtained from the object detection experiments. The reason is that in those settings we can give a clear and supported hypothesis about the future of industrially

oriented approaches to object detection, namely we can arise an interesting question on top of all discussions: *Is Deep Learning ready for industry?*

Given the impressive results obtained by YOLO in our experiments, the answer to the previous question would be positive with no discussion. Despite that, a more careful analysis is required.

The dataset used to train YOLO includes objects instances that, even if placed in slightly different positions in the scene, look very similar to the ones included in the evaluation dataset, also thanks to the fact that both have been acquired with very similar light conditions. In other words, in this task a strong generalization of the network is not required since each class we are looking for change mainly due to a rigid transformation. Thus, high detection results are here not a surprise.

On the other side, YOLO provides poor results when tested on drastically different scenes: scenes with unknown objects, more clutter, and completely random disposition of the known objects, also many of them with hard occlusions (see Fig. 5.3 for an example). In such new scenes instead, performances of a classic shape based approach are essentially the same, because they do not rely on any data collection and do not need to be trained with it. One of the causes of such performance fall may be due to overfitting. That is also verified by the fact that during training, $YOLO_O$ reached very low levels of *loss* and a very high accuracy on the training set. Moreover, given the already mentioned similarity of the images of the RAW Dataset, this task, for a big network such as the YOLO one, could not be so hard to be learned: there are essentially too much parameters in the network to perfectly represent all the information contained in the training set. That is why we introduced the $YOLO_S$ setup, where such overfitting problems have been mitigated in part. The $YOLO_S$ provides better results than the first setup, and this let us to formulate a more reasonable answer to the previous question: *we didn't get clear and robust evidences to the readiness of Deep Learning for industry so far, but in some tasks it can be taken as a accurate and fast substitute for standard computer vision approaches.* This double side solution to the question is totally legitimated from the fact that industrial settings are extremely structured, organized and fixed, there is a low level of unpredictability, and mostly relevant industrial problems can be tackled with overfitting approaches. We can, and must, overfit the problem, because the scenarios requests explicit and strict solution to its problems. For example, the problem of detect and localize an object in industrial scenarios can be tackled with a mixed approach: Deep Learning can be used while performing object detection only, e.g. bounding box detection, but the last step of pose refinement and 3D object localization could be left to standard shape based approaches where even just an exhaustive search will bring to the scope when working on the restricted search space found by the Deep Learning object detection phase.

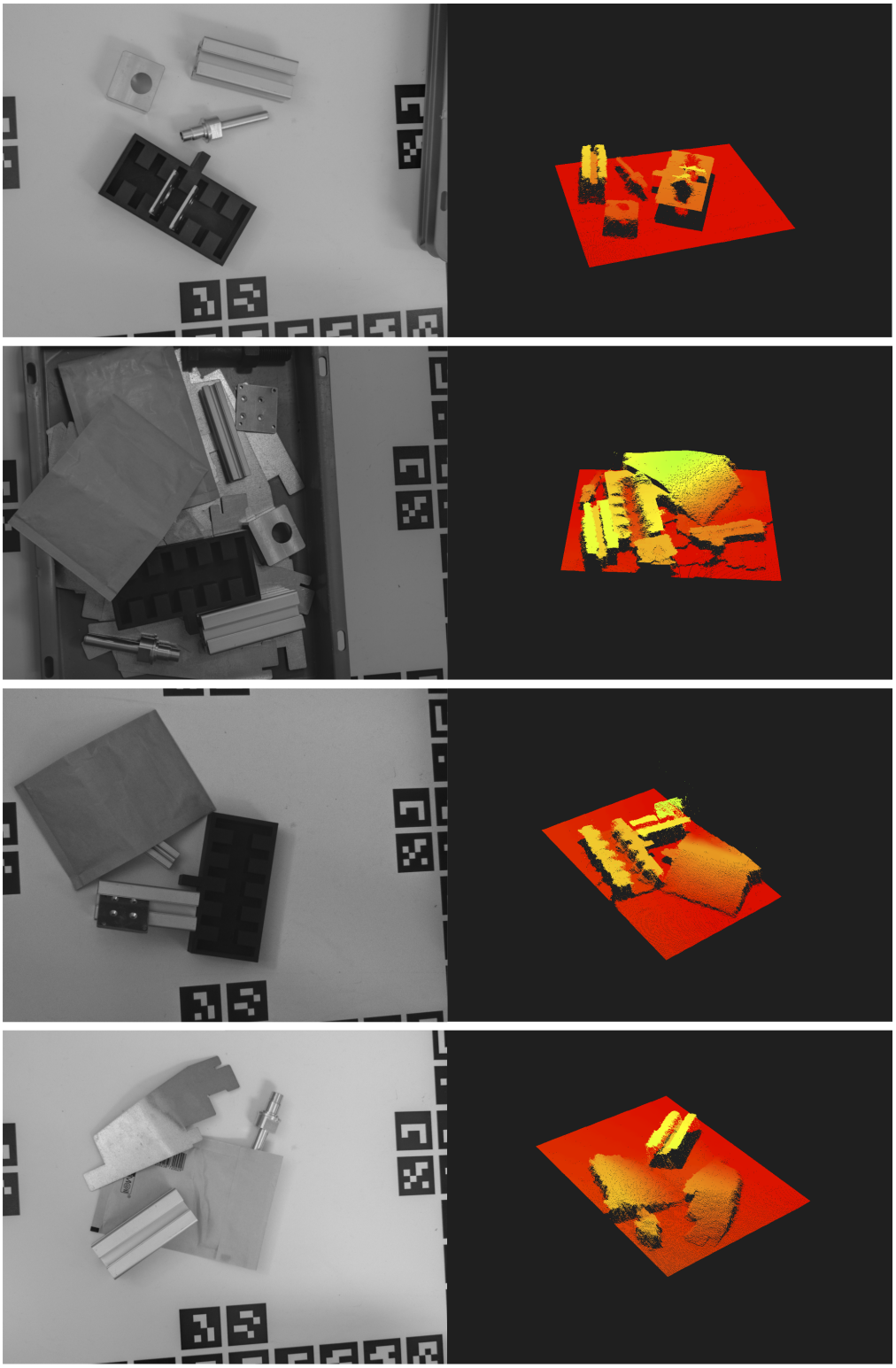


Figure 5.5: **Some stereo matching based 3D reconstruction examples**: (left column) input FlexSight sensor left images, (right column) output estimated point-clouds.

Chapter 6

Conclusions

In this work we have presented a large view about perception in industrial settings, especially focusing on robotic and machine vision. We analyzed state-of-the-art technologies, theories and their software and hardware implementations particularly looking at 3D object detection and localization and 3D scene reconstruction.

We proposed novel benchmarking tools, the RAW Dataset in particular. Up to the knowledge of the author, this dataset will be one of the few complete and rich test bed for industrially oriented studies, research activities and projects in general.

Nevertheless, this is still an on going work and many other plans are near to be accomplished before actually achieve the level of completeness needed in such strict and controlled settings. As part of the FlexSight project, the RAW Dataset will be used as first test bed solution for the largely introduced FlexSight Sensor, and given the nature of the project, the RAW Dataset will be augmented for supporting also test cases for other industrial applications, such as deformable object detection and localization.

This work is also the introduction to other already mentioned future works:

- Reach a complete coverage of the ground truth for all the scenes of the RAW Dataset;
- Deeply analyze the effectiveness of our implementation of the 3D scene reconstruction method using the metrics described in Sec. 3.1.3 of Chapter 3;
- Build and development of the final prototype of the FlexSight Sensor;
- Test the final FlexSight Sensor prototype on the RAW Dataset using all the tools developed in this work;
- Development and implementation of novel Deep Learning approaches to the problem of object localization, in terms of 6D pose detection, rotation and translation. A totally neural network approach still does not exist for this kind of problem.

Acknowledgements

There are many people that I would like to thank and, therefore, this Chapter is, inevitably, in Italian, and let me leave such a formal language from now on, those words are really coming out of my heart.

In primo luogo, vorrei dedicare questo lavoro alla mia famiglia, i miei genitori e mia sorella. Se nella vita sono diventato la persona che tutti oggi conoscono, devo questo a loro, a quell'impegno continuo nello starmi vicino, supportarmi e sostenermi in ogni scelta o difficoltá. Grazie per avermi dato tutte queste possibilitá.

Ringrazio *Linda*, la figura piú importante della mia vita, la mia ragazza, il mio piú importante pensiero. Avere vicino persone che danno la vita per te, che danno tutto per farti stare bene, é una *fortuna* impagabile, che io ho il privilegio di possedere. Non sei una semplice comparsa in questa mia vita, ne sei la protagonista e sono fiero e felice di poterti avere al mio fianco, ora e per sempre.

Ringrazio le persone che hanno contribuito a realizzare questo lavoro, in particolare *Alberto* e *Marco*, le due figure che stanno attivamente contribuendo ad arricchire il mio percorso accademico e soprattutto professionale. Fonti di ispirazione inesauribili e su cui so di poter sempre contare e con le quali spero in futuro di continuare a collaborare.

Ringrazio tutto lo staff del laboratorio *Ro.Co.Co.* dell'universitá *La Sapienza di Roma*, dai dottorandi ai professori che mi hanno seguito e formato in questi anni di studio.

Ringrazio i ragazzi del mio team, *SPQR@Work*, *Wilson, Luca, Francesco, Giulio* e tutti i ragazzi che frequentano quotidianamente il nostro laboratorio. Le giornate di studio, di lavoro e di gioco (perch per noi questa é prima di tutto una passione ed un gioco) non sarebbero le stesse.

Ringrazio i miei amici, piú o meno vicini, siete molti e vorrei nominarvi tutti, ma sapete che leggendo queste due righe mi sto riferendo a voi.

Le persone da ringraziare sarebbero infinite e non basterebbe un libro intero per descrivere le nostre avventure e i motivi per cui ho l'obbligo di ringraziarvi adesso, ma ahimé, lo spazio sta per finire. Sappiate solamente che potrete sempre contare su di me, come io so di poter fare con voi.

Grazie, e speriamo di ritrovarci di nuovo insieme per una nuova avventura!

Daniele

Bibliography

[1] Shim, H., Lee, S.: Performance evaluation of time-of-flight and structured light depth sensors in radiometric/geometric variations. Opt. Eng. 51(9), 094401 (2012)

[2] Drost, B., Ulrich, M., Bergmann, P., Hrtinger, P., Steger, C.: Introducing mvtec itodd a dataset for 3d object recognition in industry. IEEE International Conference on Computer Vision (ICCV) (2017)

[3] Firman, M.: RGBD datasets: Past, present and future. CoRR **abs/1604.00999** (2016)

[4] Hodan, T., Matas, J., Obdrzalek, S.: On evaluation of 6d object pose estimation. European Conference on Computer Vision Workshops (ECCVW) (2016)

[5] Hodan, T., Haluza, P., Obdrzalek, S., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. IEEE Winter Conference on Applications of Computer Vision (WACV) (2017)

[6] Evangelista, D., Villa, W.U., Imperoli, M., Vanzo, A., Iocchi, L., Nardi, D., Pretto, A.: Grounding natural language instructions in industrial robotics. Human-Robot Interaction in Collaborative Manufacturing Environments (HRI-CME), Workshop at IROS2017 (2017)

[7] Hirschmuller, H.: Accurate and efficient stereo processing by semi-global matching and mutual information. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)

[8] Bleyer, M., Rhemann, C., Rother, C.: Patchmatch stereo - stereo matching with slanted support windows. Proceedings of the British Machine Vision Conference (2011)

[9] Barnes, C., Goldman, D.B., Shechtman, E., Finkelstein, A.: The patchmatch randomized matching algorithm for image manipulation. Commun. ACM **54**(11) (November 2011) 103–110

[10] Karami, E., Prasad, S., Shehata, M.: Image matching using sift, surf, brief and orb: Performance comparison for distorted images. arXiv preprint arXiv:1710.02726 (2017)

[11] Imperoli, M., Pretto, A.: D^2CO: Fast and robust registration of 3d, textureless objects using the directional chamfer distance. Proc. of 10th International Conference on Computer Vision Systems (ICVS 2015) (2015)

[12] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[13] Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

[14] Everingham, M., Eslami, S., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision (2015)

[15] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in rgb-d images. Conference on Computer Vision and Pattern Recognition (CVPR) (2013)

[16] Steinbrucker, F., Sturm, J., Cremers, D.: Volumetric 3d mapping in real-time on a cpu. IEEE International Conference on Robotics and Automation (ICRA) (2014)

Appendix A

Appendix

A.1 FlexSight Project

One of the key challenges in most robotics applications, from robotaided manufacturing to service robotics applications, is the capability to automatically identify and locate various types of objects, in such a way that the robot can grasp and manipulate them accurately and reliably.

Depending on the application, objects can be regularly disposed in trays, but often they are randomly placed inside bins, or over tables, pallets or conveyor belts. A reliable perception systems is thus required to identify and precisely locate the objects, and to guide robots during the manipulation tasks. Due to the strong impact of these systems in terms of productivity, they have been widely studied in the last decades. Nowadays, many products have been proposed to solve this problem that, under a scientific point of view, it can be considered a very mature topic. Still, almost all the implemented systems assume to deal with rigid, non-deformable objects; moreover, they often assume that object instances of a given object class (e.g., a package) have all the same size and the same form factor.

FlexSight aims to remove these limiting assumptions, by proposing a perception system that is also able to recognize and localize several types of deformable objects that can be commonly found in many industrial and logistic applications, such as soft sacks, deformable and variable size packaging, food products, articles of clothing, flexible assembly parts, etc ... As *deformable objects* (See Fig. A.1) are intended either as an object that can change its shape or dimensions due to a stress or, with an abuse of notation, as each object that can be obtained from a basic template by applying a resize operator along one or more directions (e.g., a cubic parcel post can be conceptually *deformed* in many kind of parcel posts).

The main objectives of the FlexSight experiment are:

- Enable a robot to perceive a large and widespread class of rigid and deformable objects in an accurate and reliable way, with a particular emphasis on the computational speed of the whole system;
- Implement a prototype of a compact industrial sensor (the FlexSight Sensor, FSS for brevity), that integrates inside a robust and small chassis all the required sensors

Figure A.1: **Examples of deformable objects**. As deformable objects are intended either as objects that can change their shape or dimensions due to a stress or, with an abuse of notation, as each object that can be obtained from a basic template by applying a resize operator along one or more directions (e.g., a cubic parcel post can be conceptually deformed in many kind of parcel posts).

and a processing unit suitable to run the detection and localization algorithms. Our aim is to provide the robot integrators with a sensor that can be easily fitted into conventional robotics cells, thus enabling fully automated and unsupervised procedures in new areas, where usually the intervention of an operator for handling operations is required;

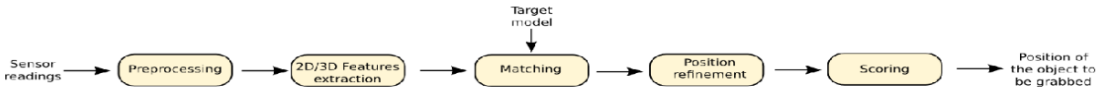
- Integrate the FlexSight Sensor inside a working system that will be tested in several industrial and logistic use cases.

A.1.1 Progress beyond the current state of the technology

A wide range of industrial products are packaged in soft bags, sacks or soft packagings. The handling and the palletization of such items during the production is addressed with well engineered and repeatable processes. This is not the case for the industrial end user of these products: they are often randomly placed inside bins or over pallets, so pick&place operations are performed with forklifts, hoists and/or cranes driven by an operator, leading to significant drawbacks, including health issues (toxic material), hygiene (e.g., in food industry) and efficiency of manual operation. On the other hand, the task of automatically sorting small packages with variable and deformable shapes (e.g., food packages, parcel posts, etc) inside unstructured environments still remains a challenging problem. A robotic system that can automatically identify, locate, grasp and manipulate all these types of deformable objects is therefore highly desirable.

Many industrial robotics pick&place systems have been studied and proposed in the last decades. Typically: (1) Either these systems assume to deal with rigid, non deformable objects, where an exact CAD model of the searched objects is provided, or (2) they assume to deal with slightly different instances of the same object class (as in the case of fruits), but objects are placed on a plane or on a conveyor belt and they are isolated, thus making the identification and manipulation relatively easy. Within the FlexSight experiment, we aim to overcome these assumptions. The challenge is first of all technological: even if there is a relevant body of scientific papers dealing with this topic, in most cases they are designed for different purposes (e.g., object categorization) or they are proof of concepts, that is they do not consider the challenges arising in bringing these technologies in highly reliable, industrial environment, that will be addressed below.

Most of the current pick&place systems employ a recognition and localization system based on vision, 3D depth sensors or RGB-D cameras. They typically employ a standard pipeline of processing blocks:



Sub-sampling, clustering and noise reduction are often used in the preprocessing blocks, while 2D and/or 3D descriptors are widely used in current systems. The matching block usually aims to associate the extracted features with features pre-computed from a rigid template (e.g., the CAD model) of the object, sample consensus methods are widely used in this case. The result is a set of object candidates, that is refined using iterative optimization strategy (e.g., Iterative Closest Point).

Many research works overcome the assumption of the rigid template model, by proposing several models and techniques to deal with deformable objects but they actually, devoted little effort to develop and test industrial systems able to accurately recognize and localize in 3D, highly deformable and variable objects. In the FlexSight experiment we aim to overcome most of these limitations, enabling an effective *from lab to market* technology transfer: we will investigate among the most promising 3D object modeling solutions, designing an automatic procedure that acquires a multiple-template deformation model (target multi-model) of the objects. The recognition procedure will extract a set of object hypotheses using model-based, visual and 3D structure cues, matched with the target multi-model. Position refinement will be obtained with an efficient optimization procedure that estimates both the position and the deformation parameters of each candidate. Leveraging on the perception outputs, the grasping and manipulation modules will be designed using state of the art hardware and software solutions, enabling reliable and efficient pick&place operations.

A.1.2 FlexSight Project Partners

Lab for Cooperative Cognitive Robots (LabRoCoCo, coordinating participant)

Sapienza Universit di Roma is one of the largest and oldest Universities of Italy and Europe. The Department of Computer, Control and Management Engineering Antonio Ruberti[1] (DIAG) includes about 65 faculty members. Internationally renowned research groups in computer science, system science and management science are active at DIAG. Every year DIAG publishes hundreds of papers in the foremost international journals and conference proceedings.

The research group involved in the project is within the Lab RoCoCo (Cooperative Cognitive Robots)[2] has a strong background in knowledge representation and reasoning,

[1] https://www.dis.uniroma1.it/
[2] http://labrococo.dis.uniroma1.it/

cognitive robotics, information fusion, mapping and localization, object recognition and tracking. multi-robot coordination, robot learning, stereo-vision, vision-based people detection. Previous EU-funded projects: SPEAKY for Robots (Echord)[3], RoCKIn[4], ROVINA[5], Flourish[6].

IT-Robotics Srl

Established in 2005, IT+Robotics[7] is a spin-off company of the University of Padova. The mission of IT+Robotics is to increase the flexibility of industrial processes by transforming the latest results of the academic research into industrial solutions. IT+Robotics is primarily involved in the fields of robotics and machine vision. IT+Robotics has great experience in random bin picking systems: its product Smart Pick 3D, presented in 2009, is currently used in more than 30 european industrial plants. The products to be collected can vary extensively, and therefore require different hardware and software technologies. By maintaining this approach, over the years IT+Robotics has developed different robot guidance solutions, each of which is designed for a specific product family. IT+Robotics has developed WorkcellSimulator (WCS), a software suite for robot off-line programming, featuring motion planning algorithms based on artificial intelligence methods and vendor independent robot code generation. Moreover, IT+Robotics, in collaboration with Robox, has developed cROS[8], a library written in ANSI C that provides a single thread implementation of the basic features required to implement a ROS (Robot Operating System) node. cROS has been released as open-source in 2015.

Robox S.p.a.

Established in 1975, Robox S.p.a.[9], has always been in the forefront of the industrial robotics field. Since the very beginning it has specialized in the design and manufacturing of micro-computerized controllers for robots. From 1987, with the spreading of the Flexible Automation, Robox has exported its know-how to non-robotic applications. Its motion controllers are now employed to control the movement of any machine.

Since 1990 Robox has been included among the Highly Specialized Research Laboratories authorized by the Italian Ministry of Education, University and Research.

Its motion controllers and related tools are applied in a variety of industrial robots, such as: spot welding, arc welding, assembly, plasma cutting, laser cutting, water jet cutting, adhesive coating, pick & place, palletizing/depalletizing, painting, etc.

[3] http://www.echord.info/wikis/website/speaky.html
[4] http://cordis.europa.eu/project/rcn/106855_en.html
[5] http://www.rovina-project.eu/
[6] http://flourish-project.eu/
[7] http://www.it-robotics.it/?lang=en
[8] http://www.it-robotics.it/products/real-time-systems/cros/?lang=en
[9] http://www.robox.it/it-IT/index.php