Università
degli Studi
della Campania
Luigi Vanvitelli

Scuola Politecnica e
delle Scienze di Base

Dipartimento di
Ingegneria

Corso di Laurea Magistrale in

Ingegneria Informatica

Tesi di Laurea in Robotica e Automazione

A multimodal perception system for detection of human operators in robotic work cells

Candidato
Dario Perrone
Matr. A18/227

Relatore
Prof. Ciro Natale

Correlatore
Dott. Gaetano Lettera

A. A. 2017/2018

Tesi

Dario Perrone

Marzo 2019

Contents

List of Figures

List of Tables

Abstract

Il monitoraggio dell'area di lavoro è una componente fondamentale delle moderne celle di lavoro industriali o in scenari di robotica di servizio in cui gli operatori umani condividono lo spazio di lavoro con i robot. Equipaggiare la cella con un adeguato sistema di percezione consente di implementare algoritmi di visione sensoriale per rilevare in modo affidabile la presenza umana. La scelta della tecnologia impiegata deve garantire la sicurezza degli operatori, quindi fornire degli input corretti all'architettura che monitora la velocità del robot, per poter applicare le normative in vigore sulla collaborazione uomo-robot. La presente tesi propone l'impiego di un nuovo sistema di percezione multimodale per il tracciamento umano , costituito da due sensori: una camera di profondità per identificare la forma ed il volume del corpo umano, ed una termocamera che supporta la prima informazione con la misura della temperatura. La strategia presentata si basa su una tecnica di Machine Learning. Dopo aver fuso le immagini di profondità con le immagini termiche correttamente sovrapposte, una rete neurale convolutiva è stata addestrata per predire la presenza di operatori umani nella scena osservata. La strategia consente inoltre di localizzare gli operatori e, quindi, utilizzare le stime ottenute per implementare scenari di collaborazione di tipo SSM (Speed and Separation Monitoring). Un'ampia sezione sarà dedicata ai risultati sperimentali per valutare l'applicabilità e le performance degli algoritmi presentati.

Abstract

Workspace monitoring is a critical hw/sw component of modern industrial work cells or in service robotics scenarios, where human operators share their workspace with robots. Sensory vision algorithms for reliable detection of human presence have been realized by equipping the robotic cell with a proper vision system. The choice of the adopted technology must guarantee the operators safety. Therefore, it provides the correct information to modulate the robot speed according to the current regulations, when a dangerous situation occurs. The work proposes a new multimodal perception system for a robust human detection, consisting of two sensors: a depth camera, which identifies the shape and the volume of the human body and allows to compute an accurate separation distance between the operator and the manipulator, and a thermal camera which improves the recognition. The developed strategy is based on a machine learning approach that processes merged images containing both depth and temperature data. A convolutional neural network predicts online the presence of the human operator into the observed scene. Therefore, the strategy localizes the human operator to implement a typical SSM (Speed and Separation Monitoring) collaborative scenario. To evaluate the applicability and the performance of the proposed algorithms, experimental results will be finally reported.

Chapter 1

Introduction

The thesis proposes a sensor fusion strategy which combines depth and thermal images to robustly detect human operators, Nowadays, industrial workcells or professional service robotics applications requires human-robot cooperation. Realize human safety is the current research challenge. The main idea is based on a sensor fusion approach: the depth image, which recognizes the human body shape, and the thermal image, which recognizes the common human body temperature about 37°C, have been merged to obtain a single image of greater information content. The two images has been combined together through a novel calibration method. The output images have been used to train a classifier that detects human presence in the scene and a detector that locates the human in the scene through the identification of bounding boxes. The detection algorithm is then interfaced a point-cloud based segmentation pipeline which computes the separation distance between the identified human operator and the moving robot. The whole architecture has been developed to implement a common SSM collaborative scenario, compliant with the actual ISOs standards.

1.1 Context

The safety standards for industrial robotic operations are laid out by the International Organization for Standardization (ISO) 10218-1 [1], 10218-2 [2] and by the upcoming ISO Proposed Draft Technical Specification (TS) 15066 [3]. Four types of collaborative scenarios are identified, which are addressed in *post-collision* and *pre-collision* scenarios [4]. Industrial safety requirements do not permit to have the use of *post-collision* systems because the physical impact between the robot and the human operator occurs before the complete stop of the machinery. Otherwise, a *pre-collision* scheme makes use of appropriate exteroceptive sensors to detect hu-

mans and prevent collisions. A *Speed and Separation Monitoring* (SSM) scenario requires that the robot speed should be monitored according to the robot separation distance from the human operator.

1.2 State of the art

Distance monitoring can be solved through motion capture systems, range sensors or artificial vision systems [5]. Nevertheless, localizing human operators robustly is not an easy task. It is often necessary to fuse several sensors with different properties.
Originally developed for military applications, thermal cameras provide relevant data to perform breast cancer diagnostic [6], infrastructure and electrical systems monitoring, gas or liquid detection [7], inspection and control tasks in industrial applications [8]. Thermal cameras are ideal for finding objects of a certain temperature: human detection and tracking (HDT) well fits for this case, as the body temperature is about 37° C. Unfortunately, thermal cameras do not support depth information, which are necessary to correctly compute the separation distance between the operator and the robot and apply the current regulations.
The main difficulty of fusing spatial and thermal images is that a correspondence between corresponding pixels needs to be found. Similar sensor fusion approaches for indoor human detection combine RGB data with depth information [9], using the Histogram of Oriented Gradients (HOG) proposed in [10] together with depth feature that describes the self-similarity of an image. Different strategies are based on Convolutional Neural Networks (CNN), widely used for object recognition [11] and human detection [12]. A CNN-based RGB-D human detector exploiting the depth information to develop a region of interest selection method (ROI) is proposed in [13]. However, the fusion of thermal and spatial information has gained attention in the last few years, especially in fields where the spatial data are used as the main source of information [14], but nowadays there are no standardized methods to robustly combine them.

1.3 Contributions

This thesis tackles the Human-Robot Collaboration (HRC) problem by introducing a novel approach to robustly detect human operators in collaborative work cells through a multimodal perception system aimed at minimizing false positives to avoid unnecessary robot stops. The architecture allows computing the separation distance

between the robot and the operator and follows the line of the current regulations ensuring the operators safety. The applicability of the approach in manufacturing industry has been obtained not by modifying the robot predefined path but by using the relative position of the human operator and the robot to define a safety metric to scale down the robot trajectory only when indispensable, thus trying to maximize the production time, i.e., in presence of humans.

The project was developed as an integration to the European project Labor [15]. The general objective of the project is to increase the level of automation of the current assembly process of fuselage parts such as panels and frames of a regional aircraft, by means of lean and flexible automated solutions in replacement of manual assembly or complex ad-hoc machine constructions and high-payload robots used in conjunction with external metrology systems. The adoption of such a solution will also allow human operators to share the area with robots during the manufacturing process, being this one of the key aspect of Industry 4.0.

Figure 1.1: Labor Logo

1.4 Outline

After a general overview of the Software and Hardware described in Chap. 2, the next chapters have been structured in order to deal with different problems and aspects independently. Chap. 3 deals with the problem of Sensor Fusion, introducing the cameras calibration method and describing the real pixel-by-pixel fusion algorithm. Since Chap. 5 and Chap. 6 deal with a standard Machine Learning problem concerning image classification and the object detection through the use of Convolutional Neural Networks(CNN), there is a complete overview of CNNs in Chap. 4 that explains the possible operations and the structure layers and shows the basic architecture of modern CNNs. The merged images resulting from sensor fusion algorithm

described in Chap. 3 is used for the Classification and Detection algorithms. Finally, the developed algorithms are used and interfaced within an available segmentation pipeline which computes the separation distance between the human operator and the robot and modulates the robot speed according to the actual regulations in the Chap. 7.

Chapter 2

Software and Hardware description

Industrial robots are finding large range of applications where they often have to perform actions in collaborative scenarios. A good architecture to percept the shared workspace is composed by a multimodal perception system which allows to implement robust algorithms to detect and localize human operators.

2.1 Hardware description

The following section is an overview of the hardware devices proposed to solve the human detection task. The operating principles and the technical characteristics of the chosen sensors are described in detail.

2.1.1 Depth Sensors

The ability to measure the distance, known as "range" or "depth", for a given point in a scene is the essential characteristic that distinguishes a depth camera from a conventional 2D video camera. There are various types of technologies such as the common Laser Range Finders that can only provide a single depth point and can use Time-of-flight or triangulation techniques. Another depth sensor variant is the scanning Lidar used in some automobiles where a few beam are scanned across the scene and a depth map is built up over time.

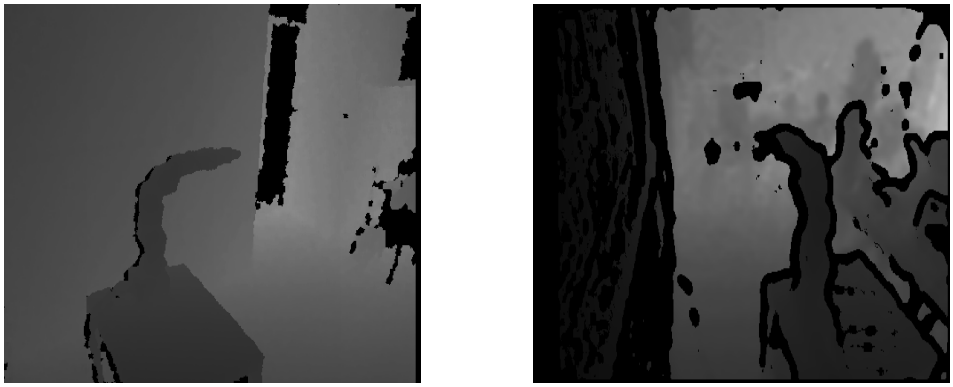

Figure 2.1: Kinect depth image on the left and Realsense depth image on the right.

Working principles

The *Time of flight* (ToF) technology is based on measuring the time that light emitted by an illumination unit requires to travel to an object and back to the sensor array: the scene is actively illuminated using a pulse of an emitted periodic radiation, e.g. laser, infra-red light or near infrared light (NIR). Assuming sensor and lighting placed in the same position, due to the distance between the sensor plane and the object, a time shift is caused in the optical signal which is equivalent to a phase shift in the periodic signal. This shift is detected in each sensor pixel by a so-called mixing process.

The *Structured light* approach is an active stereo vision technique that provides the projection of known patterns on objects in the scene with the aim of calculating the depth and surface information of the objects. Although many other variants of structured light projection are possible, patterns of parallel stripes are widely used: the way that these deform when striking surfaces allows for an exact retrieval of the 3D coordinates of any details on the object's surface.

The *triangulation* can be seen as a process of determining a point in 3D space given its projections onto two images coming from two different cameras which frame the same scene from two different views. This process is the basis of stereovision and 3D reconstruction and it is based on the human visual perception system. Given the two images and knowing the spatial relation between the camers, the corrensponding 3D point can be calculated by equipolar geometric consideration (figure 2.2).

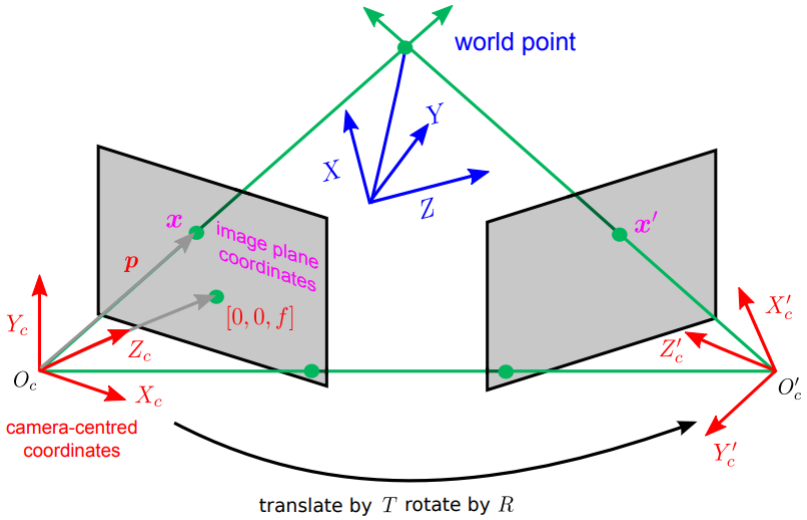

Figure 2.2: Stereo Vision Model

Depth Cameras

The adopted depth cameras are:

- A **Microsoft Kinect v1**, which is a single set-top unit that combines an RGB visible spectrum camera and an infrared (IR) spectrum 3D camera.
- An **Intel Realsense D435** which is a USB-powered depth camera and consists of a pair of depth sensors, RGB sensor, and infrared projector.

Figure 2.3: Microsoft Kinect v1 on the left and Intel Realsense D435 on the right

The Intel camera was used to complete the Kinect Camera's FoV as dark corners were initially present with the use of Kinect alone. The Kinect v1 camera combines structured light and stereo (Structure + Stereo) by utilizing an IR projection and two IR image sensors to provide more accurate depth information: IR emitter project a light pattern and an IR sensors detect the deformations in the projected pattern for

resolving the depth. The technique of analyzing a known pattern is called *structured light* and it is combined with the classic computer vision techniques of *depth from stereo*: the camera analyzes the shift of the speckle pattern by projecting from one location and observing from another. The depth map can be combined with a color camera to produce point clouds.

2.1.2 Thermographic Camera

A thermographic camera or thermal imaging camera is a device that distinguishes hot objects using infrared radiation. Unlike ordinary cameras operating between 400-700 nanometers of visible light range, the thermal camera operates in wavelengths as long as 14 μm.

Figure 2.4: Optris PI 450

All objects that have surface temperatures above absolute zero emit electromagnetic radiation and a special camera can detect this radiation in a way similar to the way an ordinary camera detects visible light. The emitted radiation is characterised by two features: wavelength and intensity. These two parameters of the emitted radiation are related to the temperature of the body surface through the following formulas:

$$Q_{tot} = \sigma \cdot T^4 \tag{2.1}$$

$$\lambda_{max} = 2.9/T \tag{2.2}$$

where $\sigma = 5.67 \times 10^{-8} W \cdot m^{-2} \cdot K^{-4}$ the Stefan-Boltzmann constant, λ_{max} is the peak wavelength of emission, T is the surface temperature and Q is the total radiation emitted by a body.

Images from infrared cameras tend to be monochrome or sometimes they are displayed in pseudo-color, i.e. the changes in color are used rather than changes in intensity to display changes in the signal. This technique, called slicing, helps the

human to appreciate the fine differences in intensity.

Figure 2.5: Thermal Image

The thermographic camera used in this vision system is the Optris PI 450 which has a frame rate of 80 Hz and an optical resolution of 382 x 288 pixels. The detectable temperature is within a range of -20 °C up to 900 °C while the spectral range is between 7.5 to 13 μm. The small and compact shape makes it suitable for different purposes and to be mounted in a vision system.

2.2 Software description

The section contains an overview of the software used for project development: the interface between the cameras and the sensor fusion algorithm was developed using the Robot Operating System (ROS), while the neural networks for the classifier and the detector were developed through the Keras and Yolo frameworks, respectively, then interfaced via ROS.

2.2.1 ROS

ROS [16] is an open-source robot operating system which includes a set of software libraries and tools that help building robot applications that work across a wide variety of robotic platforms. The operating system side provides standard operating system services such as hardware abstraction, low-level device control, message-passing between processes and package management. The ROS runtime "graph" is a peer-to-peer network of processes: it consist of numerous small computer programs which connect to each other and continuously exchange messages.

ROS is tools-based since there are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc. Another ROS property is the multi-linguages infact the software modules can be written in any language for which a client library has been written: currently client libraries exist for C++, Python, etc.

Figure 2.6: ROS logo

ROS is based on concepts as nodes, topics, messages and services. The nodes are single-purposed executable programs e.g. sensor drivers, actuator drivers, mapper and planner that are individually compiled, executed, and managed. Nodes are written using a ROS client library (roscpp, rospy), they can can publish or subscribe to a Topic and also provide or use a Service.
Nodes communicate with each other by publishing messages to topics so the topics represent a stream of messages with a defined type.

2.2.2 Keras

Keras [17] is high-level neural networks API written in Python widely used for easy prototyping. It is a deep learning library that supports both convolutional networks and recurrent networks, as well as combinations of the two. Moreover it allows the CNN to run on both CPU and GPU, according to the desired performance. Keras offers two ways to organize the layers of the CNN depending on the chosen network model. The simplest model type is the sequential one corresponding to a linear stack of layers composed by many convolutional, activation and pooling layers. The model needs to know the expected shape of the image input, i.e. the number of channels and the input images resolutions. For the case study, there are two channels, the first involved for the depth image and the second one involved for the thermal image, and the input image size is about 388x288, that is the resolution of the thermal camera.The input shape can be specified through the Dense 2D layer. To train the selected model, the learning process has been set to optimize the output classification accuracy. More in detail, the optimization method, the loss function to be optimized and the metric to use for the classification problem, have to be selected

to compile the learning process. The initial fit function starts the training process, by acquiring both the training images data set and the corresponding labels. The training process ends when the batch size, the number of epochs or the validation data has been reached. Appendix B contains details of the developed Keras software.

Figure 2.7: Keras logo

2.2.3 YOLO framework

Current detection systems propose image classifiers to perform objects recognition and localization. More recent approaches like R-CNN [18] combine region proposals with CNNs: for each image it extracts around 2000 bottom-up region proposals, computes features for a proposal using a convolutional neural network and then classifies each region using class-specific linear Support Vector Machine (SVM). Then a post-processing is needed to refine the bounding boxes and delete duplicate detections. These kind of pipelines are very complex, thus they are too slow and hard to optimize because each individual component must be trained separately.

You only look once (YOLO) [19] is a new approach for real-time object detection system and it is extremely fast and accurate. This framework deals with object detection as a single regression problem and involves the application of a single neural network which divides the image into regions and predicts bounding boxes and probabilities for each region. YOLO also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image. This makes it extremely fast, more than 1000x faster than R-CNN. YOLO focuses on an extreme speed/accuracy trade-off: as shown in Fig. 2.8, it runs significantly faster than other detection methods with comparable performance and this is the reason why this framework has been chosen for the case study.. After the first version, two other versions have been released: YOLOv2 [20] and YOLOv3 [21] which improve the accuracy while making it faster.

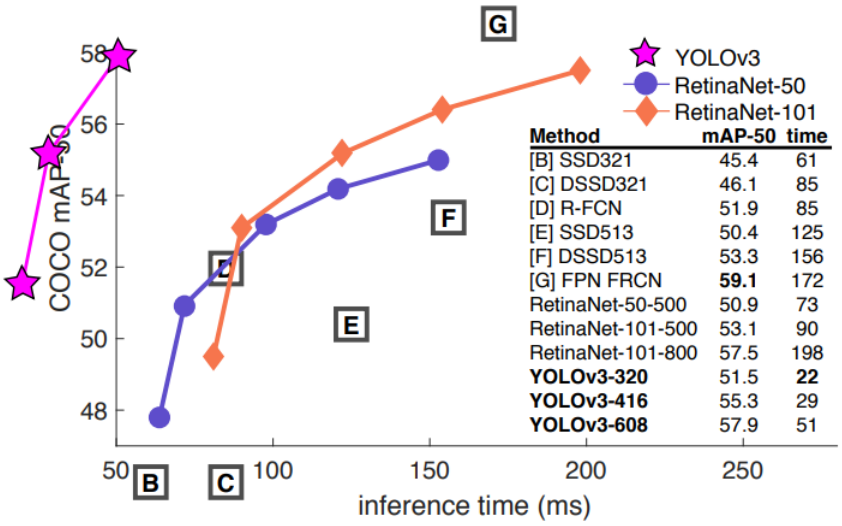

Figure 2.8: Detection methods performance graph

Chapter 3

Sensor Fusion

Sensor fusion combines sensory data to obtain output data of grater information content [22]. Generally, original sensor data suffer from uncertainty, imprecision, limited spatial and temporal coverage, thus a sensor fusion technique could partially solve these limitations. Image fusion techniques combine multi-modality sensor image of the same observed scene to provide an enhanced single view of a scene with extended information content [23]. Image fusion is divided at the hierarchical level ,e.g. by processing the features extracted from the input images, or at the low level, e.g. by processing original sensor images. The case study adopted the second methodology: the original depth data and the original thermal data have been merged to obtain a single, fused image. The output image can be a gray scale image or a chromatic image, according to the specific application target. The following ections explain the methods adopted to the sensors calibration, the technique developed to map the depth image pixels on the corresponding thermal image pixels and, finally, the strategy proposed to perform image fusion.

3.1 Vision System Calibration

In many experimental applications of robotic research, the vision system is equipped by a thermal camera as thermal information source. Generally, different approaches combine it with depth information, that is acquired by different sources, e.g. a depth camera, a laser range finder, a visual camera. Whatever is the source of acquisition, the resulting multi-sensor system has to be extrinsically calibrated to select the relation between the sensors. This step can be performed by using a calibration target (e.g., a perforated grid), [24], [25]. However, some proposed solutions explain the calibration approach with the only purpose of solving it, without a suitable contextualization or not considering the applicability at all, basing on fairly conservative

assumptions.

3.1.1 Pinhole Camera Model

The geometric model of pinhole camera consists of:

- a reference system (X,Y,Z) centered in O called the optical center and with the Z-axis coinciding with the optical axis
- a second plane, called image plane with a reference system (u,v) centered in the main point and with u and v axes oriented respectively as X and Y. The image plane is at focal length from the origin O of the camera reference system.

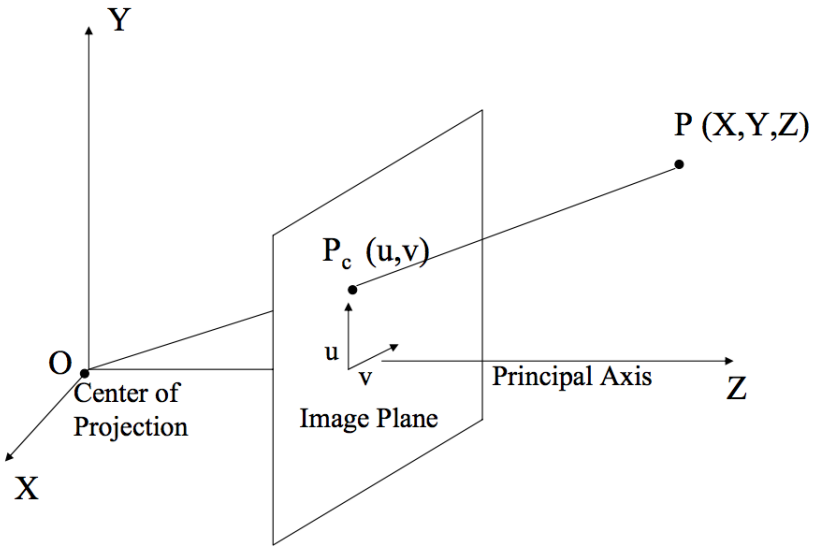

Figure 3.1: Pinhole Model

Using similar triangles as

$$\frac{f}{Z} = \frac{-u}{X} = \frac{-v}{Y} \tag{3.1}$$

which gives us

$$u = \frac{-f}{Z}X \tag{3.2}$$

$$v = \frac{-f}{Z}Y \tag{3.3}$$

Given a vector $P = [x, y, z]^T$, we use $\tilde{P} = [x, y, z, 1]^T$ to denote the homogeneous coordinates of P and $P_C = [u, v, 1]^T$ to denote the homogeneous coordinates $P_C =$

$[u, v]^T$.
Using homogeneous coordinates and matrix notation, we can write the 3.2 as

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -f_x \\ -f_y \\ Z \end{bmatrix} = \begin{bmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

The 3x4 matrix P is called the *camera perspective projection matrix.*
In the real case we have to consider the shape and size of the pixels and the position of the image plane respect to the optical center. If the origin of the 2D image coordinates system does not coincide with the Z axis intersects the image plane, a translation defined by (t_u, t_v) must be performed. Furthermore, a reascaling of the u and v axes is defined through the m_u and m_v parameters corresponding to the inverse of the pixel size. Thus

$$u = m_u \frac{-fX}{Z} + m_u t_u \tag{3.5}$$

$$v = m_v \frac{-fY}{Z} + m_v t_v \tag{3.6}$$

This can be expressed in matrix form as

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -m_u f & 0 & m_u t_u & 0 \\ 0 & -m_v f & m_v t_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} a_x & 0 & u_0 & 0 \\ 0 & a_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{P} = K[I|0]$$

where K only depends on the intrinsic camera parameters like its focal length, principal axis and thus defines the *intrinsic parameters* of the camera.

$$K = \begin{bmatrix} a_x & 0 & u_0 \\ 0 & a_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

In general, the reference system of the camera (X, Y, Z) is in a different position from the world reference system. We need to introduce a rigid transformation that connects the two reference systems made up of a rotation R and a translation t.

The *extrinsic parameters* E represent the coordinate transformation between the reference frame and the sensor frame.

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad E = \{R \mid t\}$$

3.1.2 Intrinsic Calibration

The intrinsic calibration of the cameras was carried out by using an available ROS software [26] which generates a .yaml file containing the intrinsic parameters compatible with the ROS camera drivers. To use this tool you need to have a table with a certain pattern that is clearly distinguishable from the camera.
A chessboard pattern is needed (see Fig. 3.2).

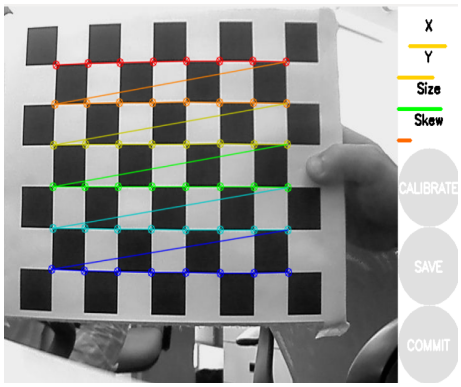

Figure 3.2: *camera_calibration* tool on chessboard pattern

Differently from the depth camera calibration, the thermal camera intrinsic calibration needs a perforated plastic grid with a circular pattern: grid has been properly heated to be clearly distinguishable from the thermal image, as shown in Fig. 3.3. The adopted software requires the pattern set-up (number of columns, number of the raws, the real size and the pattern type, e.g. square, circle). Once the pattern is recognized the operator performs linear and angular movements to improve the estimation of the calibration parameters. The structure of the output calibration file is shown in Appendix A. This type of file is supported by the ROS drivers for the Kinect camera and the Optris camera but is not supported for the Intel RealSense camera drivers for which a calibration has been performed through the tool of the parent company.

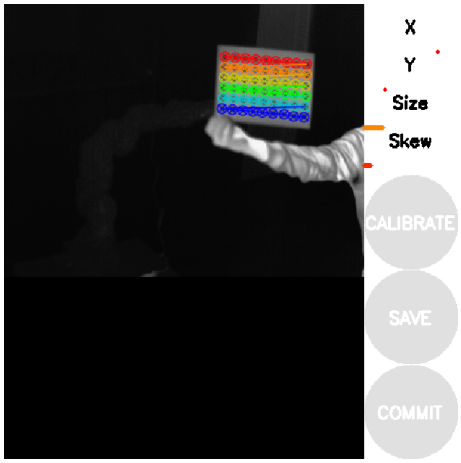

Figure 3.3: *camera_calibration* tool on grid pattern

3.1.3 Extrinsic Calibration Depth Camera

The purpose of the extrinsic calibration is to define the transformation matrix of the camera frame expressed in the world frame T^{world}_{camera}. The robot end effector has been equipped with a pointed tool appropriately modeled in CAD and printed through a 3D printer. To estimate the transformation matrix through the optimization of a cost function once the input data are provided, a MATLAB script has been written. The calibration algorithm provides the following steps:

- Move the robot to a certain $i-th$ configuration such that the pointed tool is visible from the depth camera.
- Collect the Cartesian points of the tip of the tool expressed in camera frame, $p_i^{camera}[x, y, z]$ indicated by the tip of the robot end effector within the P^{camera} matrix; the points coordinates are obtained by generating the Point Cloud of the scene for each robot configuration and reading the coordinates of the space portion pointed by the end effector assumed as a point.
- Collect the corresponding robot joint configurations and fill a $7xn$ Q matrix.
- The MATLAB script computes the Cartesian point of the tool tip through the forward kinematic of the acquired robot configurations. The obtained points are expressed in the robot base frame, p_i^{world} and organized in P^{world} matrix. Finally, the corresponding points matrices, P^{world} and P^{camera}, have been used to estimate the transformation matrix between the camera and the world frames through an optimization algorithm of a cost function.

This process for extrinsic calibration, described in more detail in following Algorithm 1, has been adopted for both the Kinect camera and the Realsense camera.

Algorithm 1 T_{camera}^{world} Estimation

input:
$p_i^{camera} \leftarrow$ i-th point of the space in camera frame
$q_i \leftarrow$ i-th robot configuration in the joints space
$P^{camera} \leftarrow p_i$ points collection
$Q \leftarrow q_i$ points collection
$T_{init} = T_{camera}^{world}$ initial estimation
procedure TRANSFORMATION MATRIX BETWEEN CAMERA AND WORLD FRAME T_{camera}^{world}
    for $i = 1 : size(Q)$ **do**
        K = Forward kinematics (q_i)
        $P_i^{world} = K.transpose$
    end for
    Cf = Cost function(T_{init}, P)
    T_{camera}^{world}= FminOptimiser(Cf, T_{init}, P, K)
end procedure

3.1.4 Extrinsic Calibration Thermal camera

Thermal imaging has become a helpful and widely used tool for several control tasks in industrial applications. The thermal image is often not sufficient to satisfy large and complex inspection tasks and therefore can be used as a complementary information, such as spatial information. Nowadays, thermal imaging provides relevant data to perform specific robotic applications that require thermal information. When combining two or more sources of acquisition, the resulting multi-sensor system has to be extrinsically calibrated to find the relative pose between the adopted sensors. This step can be performed by using a calibration target. In [24] and [25] a thermal camera and a depth camera are calibrated by using a perforated grid placed in front of the camera frames, close enough to the lenses and heated to be distinguishable from both sensor images. However, the proposed solutions propose the calibration approach with the only purpose of solving it, without a suitable contextualization or not considering the applicability at all in a specific scenario, basing on fairly conservative assumptions.
The experimental setup of this work is shown Fig. 3.4 and consists of two cameras rigidly attached together through two cross zip ties. They have been arranged in a way that their optical axes are aligned. The adopted cameras have different field of views (FOVs) and this implies that some depth pixels (*Microsoft Kinect v1*, Focal length: 6.1 mm, FOV: 57°x45°, image size: 640x480) are outside the thermal image

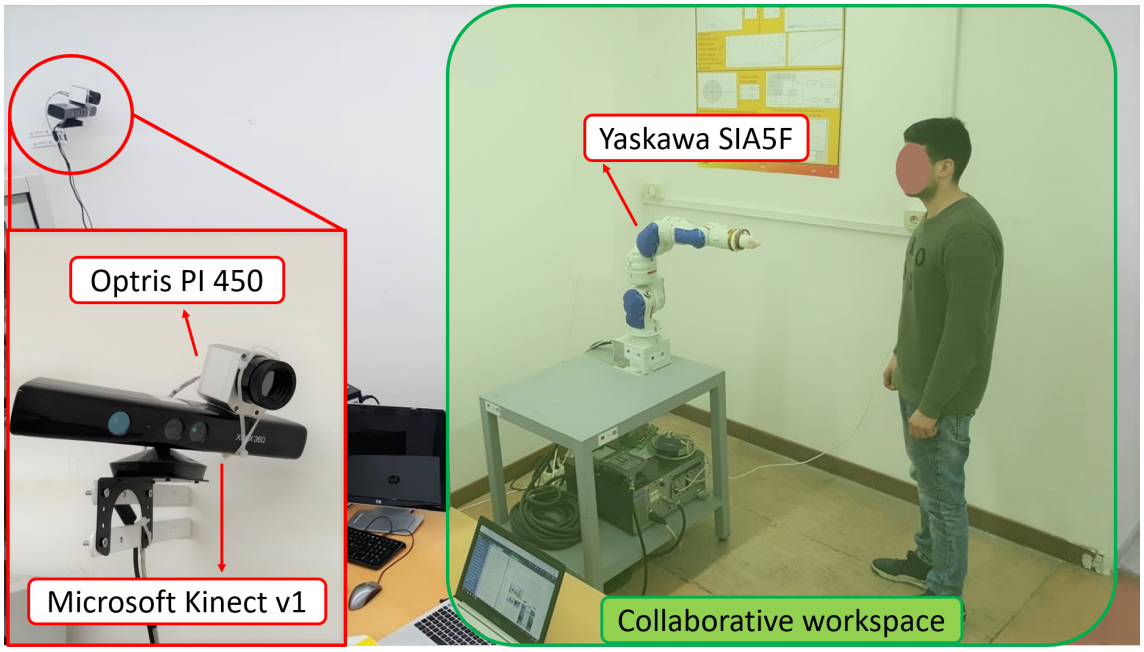

Figure 3.4: Perception system.

(*Optris PI 450*, Focal length: 15 mm, FOV: 38°x29°, Spectrum: 7.5 to 13 µm, image size: 382x288) and they are not used in the merging step (Section 3.2.1). The goal of the extrinsic calibration is to obtain an accurate identification of the camera poses, which guarantee the minimum accuracy error when the two camera views are merged. The extrinsic calibration of the thermal camera with respect to the depth camera has been solved by using 3 spheres attached to a flat cardboard support. To obtain an estimation of the transformation matrix $\boldsymbol{T}_t^d$, between the depth camera frame Σ_d and the thermal camera frame Σ_t, the spheres have been moved inside the collaborative workspace by placing the support in 10 configurations at distances from the camera in the range where the human operator is expected to act during the collaborative task. At every acquisition, the calibration target has been suitably heated to be detectable from both cameras. The coordinates $\boldsymbol{p}_k^d = \begin{bmatrix} x_k^d & y_k^d & z_k^d \end{bmatrix}^T$ of the kth center of the target sphere have been directly calculated from the depth image, while the corresponding thermal point coordinates have been calculated from the thermal image, assuming the distance from the lens equal to the depth value, i.e., $z_k^t = z_k^d$ and

$$x_k^t = \frac{(a_k - c_{x_t}) z_k^t}{f_{x_t}} \tag{3.7}$$

$$y_k^t = \frac{(b_k - c_{y_t}) z_k^t}{f_{y_t}}, \tag{3.8}$$

where a_k and b_k are the pixel coordinates of the sphere center in the thermal image, c_{x_t}, c_{y_t} are the pixel coordinates of the thermal image center and f_{x_t}, f_{y_t} are the focal lengths expressed in pixel-related units. Finally, the transformation matrix $\boldsymbol{T}_t^d$ has been estimated by minimizing a cost function that combines the corresponding data with a similar optimization Algorithm 1 of Sec. 3.1.3.

3.2 Depth-Thermal Image Fusion

The first requirement of a sensor fusion approach is to preserve all valid and useful information from the sources to be combined, while not introducing distortions. For the purpose of this work, the depth image and the corresponding thermal image have been merged to provide an enhanced single view of a scene with extended information content, through the mapping matrix of Sec. 3.2.1.

3.2.1 Mapping Process

The extrinsic calibrations explained in Sec. 3.1.3 allows to make correctly the mapping step, that means to find matches between the depth image and the thermal image. Since the adopted cameras have different FOVs and resolutions, the resulting map size must correspond to the smallest one. According to the experimental set-up, the mapping step builds a 382x288 matrix because of the smallest dimensions of the thermal image with respect to the depth image, as shown in Fig. 3.5.
The mapping step has been solved through a pixel-by-pixel procedure: the pixel of the depth map, of indices $[m, n]$, contains the depth value, $z^d_{m,n}$, which is acquired to compute the corresponding Cartesian point coordinates $P^d_{m,n} = [x^d_{m,n}, y^d_{m,n}, z^d_{m,n}]$ by using the following formulas:

$$x^d_{m,n} = \frac{(m - c_x) \cdot z^d_{m,n}}{f_{sx}} \tag{3.9}$$

$$y^d_{m,n} = \frac{(n - c_y) \cdot z^d_{m,n}}{f_{sy}} \tag{3.10}$$

where c_x and c_y are the pixel coordinates of the center (on the focal axis) in x and y directions of the image plane, f is the focal length of the camera and sx and sy are the dimensions of a pixel (in meters).

The Cartesian point is then expressed with reference to the thermal camera frame through the relation:

$$\tilde{P}^t_{m,n} = T^t_d \cdot \tilde{P}^d_{m,n} = [x^t_{m,n}, y^t_{m,n}, z^t_{m,n}, 1] \tag{3.11}$$

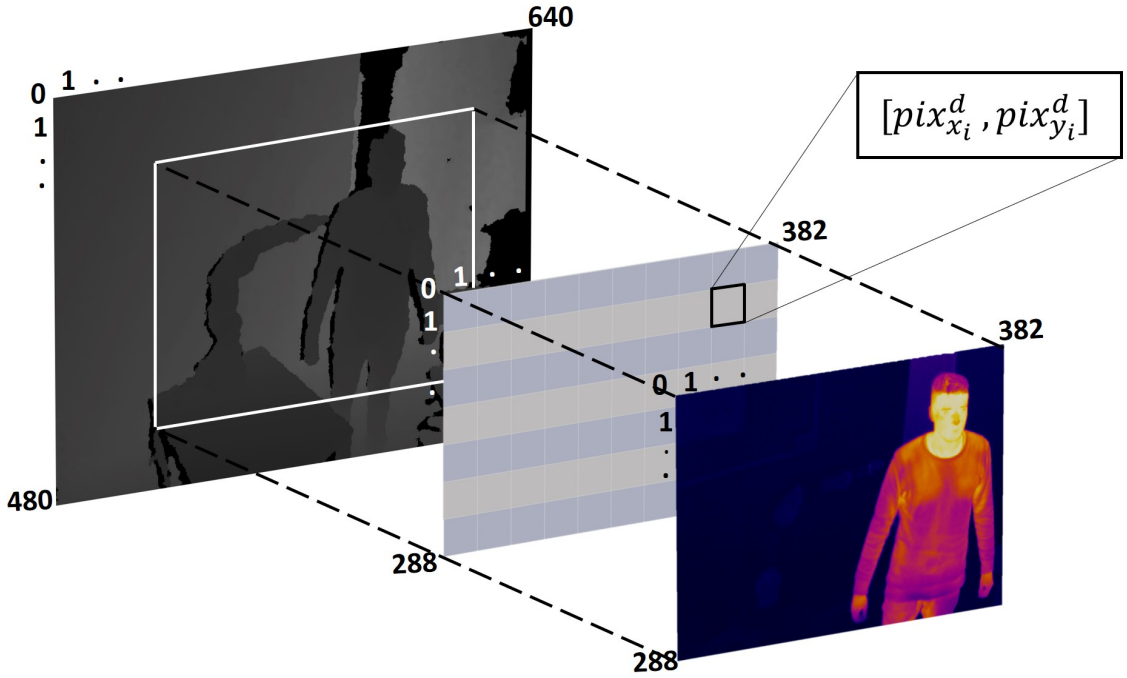

Figure 3.5: Depth and thermal mapping.

from which $P^t_{m,n}$ is immediately obtained from the first three components of $\tilde{P}^t_{m,n}$. Using the intrinsic parameters of the thermal camera, the corresponding pixel indices of the point $P^t_{m,n}$ into the thermal map , $[a, b]$, are finally computed: if they are contained in the FOV of the thermal image, the corresponding depth pixel indices $[m, n]$ are written into the mapping matrix at the indices $[a, b]$; on the contrary, they are discarded because they are outside the mapping image size. In other words, at the end of the assessments, the resulting map contains the two sensor information combined together for each corresponding pixel.

3.2.2 RGB Mapping Approach

Multi-modality sensor images can be combined through different image fusion techniques which work at different merging levels: pixel-by-pixel, combining signals, using relevant features or at symbol levels. This section provides an image fusion algorithm which falls into pixel level but represents a novel approach with respect to the most widely used pixel-level image fusion algorithms [27].

The proposed *RGB Mapping Approach* (RGB-MA consists of defining the intensities of empty RGB channels. This is also because, from the best of our knowledge, convolutional neural networks work better with RGB images. RGB-MA strength

is that it assigns the same priority to the input sources. The result is no longer a grayscale, as a depth image alone could be, or a withed average image which assigns different priorities to the sources, but it is an RGB image where the depth data have been mapped on the green channel and the temperature values have been mapped on the red channel (see Fig. 3.6).

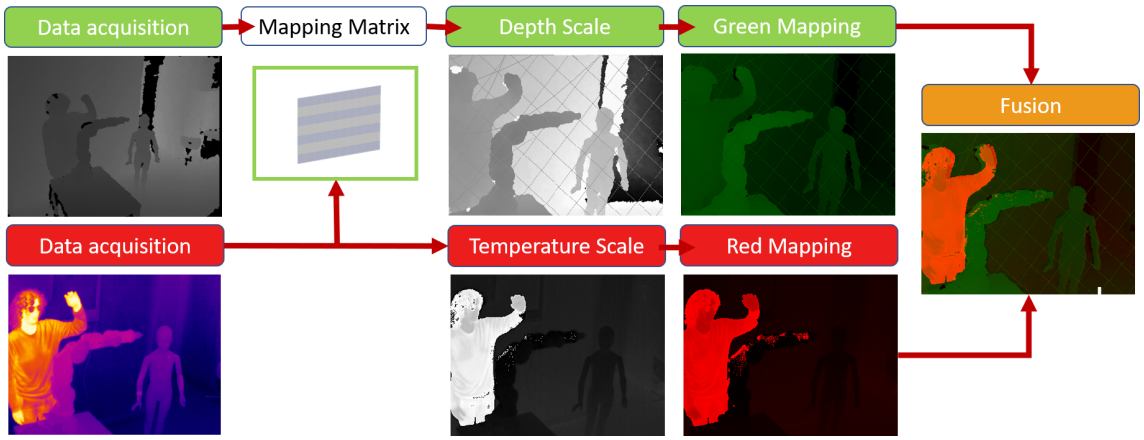

Figure 3.6: Depth and thermal mapping pipeline.

Specifically, the original depth sensor value, s^d, and the corresponding temperature sensor value, s^t, have been normalized into the interval $[0, 1]$. To do this, a *minimum* and a *maximum* variability ranges of the sources values have been defined: they do not actually correspond to the ranges of the sensors technique specifications, but they have been chosen according to the values detectable into the experimental workspace. More in detail, the detectable depth values are included between 0.30 m and 4.0 m, while the detectable temperature values are within the range $[0 - 50]$ °C, which are suitable for any type of human detection tasks.

The color information inserted into the specific channel of the *(i,j)-th* pixel of the output image must be mapped in 8 bits. The R value is computed acquiring $s^t_{i,j}$ from the thermal image and applying the Eq. 3.12; the G value is computed acquiring $s^d_{m,n}$ from the depth image, where m and n are contained into the *(i,j)-th* value of the mapping matrix (Sec. 3.2.1), and applying the Eq. 3.13; the B value of the resulting image is always zero.

$$R_{i,j} = round\left(\frac{x_t - min_t}{max_t - min_t}\right) \cdot 255 \tag{3.12}$$

$$G_{i,j} = round\left(\frac{x_d - min_d}{max_d - min_d}\right) \cdot 255 \tag{3.13}$$

$$B_{i,j} = 0 \tag{3.14}$$

The result is shown in Fig. 3.7. Note that the proposed image fusion technique leaves another channel that could be used for a further input source.

Figure 3.7: Depth and Themral sensor fusion images

Chapter 4

Convolutional Neural Network

Convolutional Neural Networks (ConvNets or CNNs) are a special type of neural networks widely used for object classification and recognition. The effectiveness of objects recognition in image recognition is one of the main reasons of their diffusion in computer vision and in applications like self-driving cars, robotics and drones.
In general, the use of traditional Fully Connected networks (FC) for image processing is not recommended for several reasons: FC layers do not exploit local structure and for for large size images they have huge number of parameters. For example, processing a 1000x1000x3 image with a FC network with 1000 layers, such layer has $\sim$ 3 billion weights and the problem becomes impractical.
Before CNN became popular, people used to extract features from images and then feed them into some classification algorithm like SVM. CNNs networks automatically extract these features from the images through convolutive operations and then use a prediction layer at the end.
This concept was presented by Yann le cun in 1998 for digit classification. His work is based on a single convolution layer. CNNs were then largely adopted by Alexnet in 2012 who used multiple convolutional layers to achieve state of the art on imagenet.

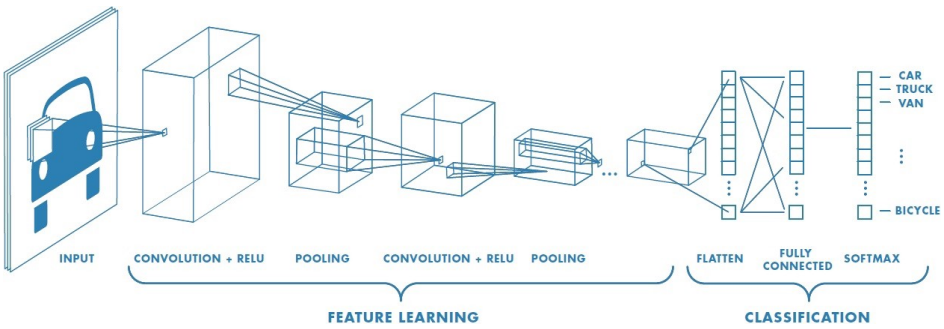

Figure 4.1: General model of a CNN.

4.1 Convolution

Convolution is a mathematical operation on two functions to produce a third function that expresses how the shape of one is modified by the other. Intuitively, the convolution of two functions represents the overlapping amount between the two functions. The convolution of two functions f and w is defined as the integral of the product of the two functions after one is reversed and shifted. As such, it is a particular kind of integral transform:

$$f * w = \int_{-\infty}^{\infty} f(\tau)w(t-\tau)d\tau \tag{4.1}$$

In image processing, a kernel is a small matrix used for blurring, sharpening, embossing, edge detection.
The function f is the input, w the kernel of the convolution. The general expression of a convolution is:

$$g(x,y) = (w * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)f(x-s, y-t) \tag{4.2}$$

The operation of convolution between matrices, although it may seem similar, is not a multiplication operation between matrices. The figure 4.2 shows a part of the convolution operation between the matrix I representing the image and the K matrix representing the kernel thus defined:

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad K = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

4.2 CNN layers

A ConvNet is a sequence of layers, Each layer processes its input data through a differentiable function, to activate the next layer and pass it its processed data. There are three types of ConvNet layer: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. Deciding the number of layers to use and the filter sizes or other network parameters is not a trivial problem and there is not a a standard tuning that is commonly used by researchers: the choice of the model of the network

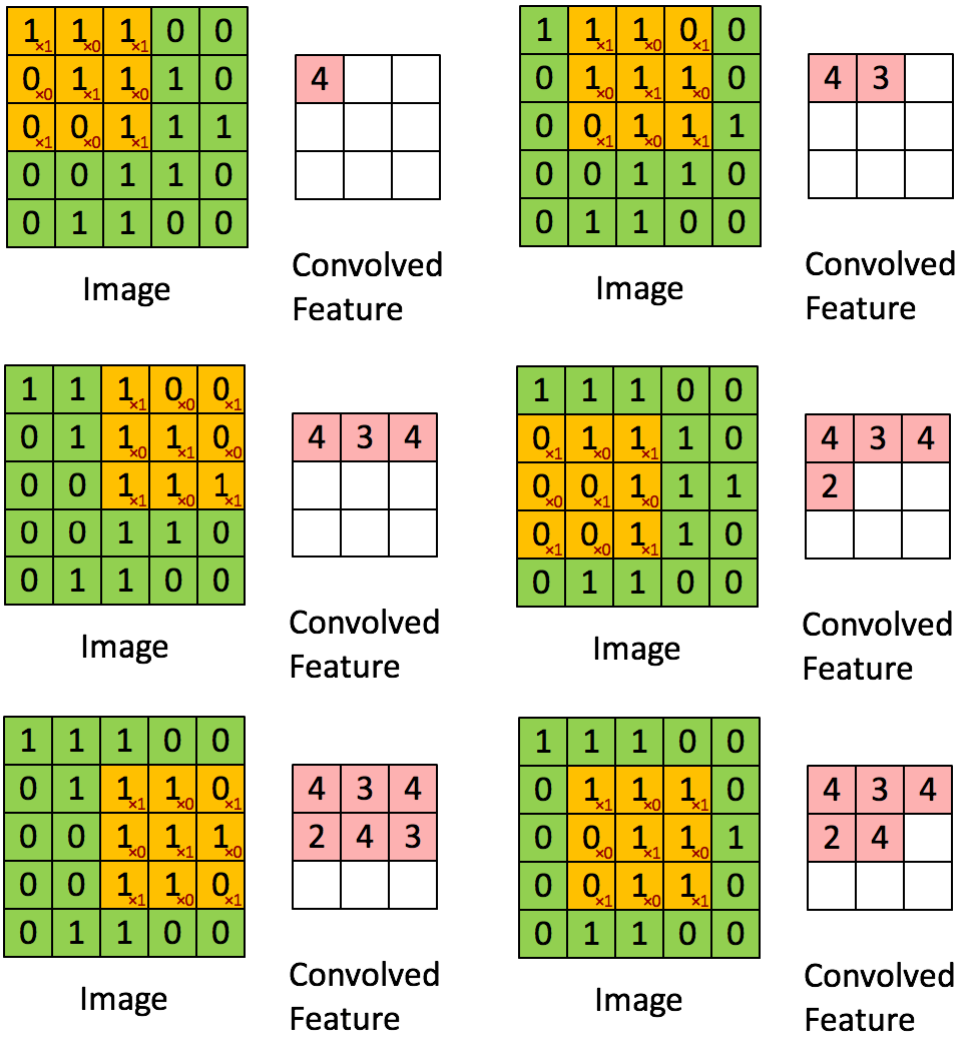

Figure 4.2: Matrix Convolution steps

depends on the type of input data. Image size, image complexity and the type of image processing task require a customized tuning of the network parameters. Generally, the final set-up depends on the experimental results.

4.2.1 Convolutional Layer

The parameters of the Convolution Layer consist of a series of learning filters generally with a small height and width but with a depth equal to the depth of the input volume. A typical filter can have dimensions 5x5x3 where 3 corresponds to the depth ie the size of the three RGB channels of an image. The filter slides and convolves across the width and height of the input volume: this operation will produce a 2-dimensional feature map that gives the responses of that filter at every spatial position. For each convolutional layer there will be a series of different filters each of those will produce a separate feature map; these activation maps will be stack along the depth dimension to produce the output volume.

A specific parameter of convolution layer is the *stride*: it specifies how many pixels we move at a time when we slide the filter on the image. For example when the stride is 2 then the filters jump 2 pixels at a time as we slide them around and this will produce smaller output volumes spatially.

Another important hyperparameter is the *padding*: sometimes it will be convenient to pad the input volume with zeros around the border (Fig. 4.3). The feature of zero padding is that it will allow us to control the spatial size of the output volumes infact as we keep applying convolution layers, the size of the volume will decrease faster than we would like. In the early layers of our network, we want to preserve as much information about the original input volume so that we can extract those low level features.
Generally, downstream of the convolution layer there is an activation function layer for rectification which takes the feature map generated by the convolutional layer and creates the activation map as its output. The rectified linear units (ReLUs) are a special implementation that combines non-linearity and rectification layers in convolutional neural networks and it is defined as:

$$f(x) = max(0, x) \tag{4.3}$$

It has been demonstrated to enable better training of deeper networks compared to the widely-used activation functions as the logistic sigmoid.

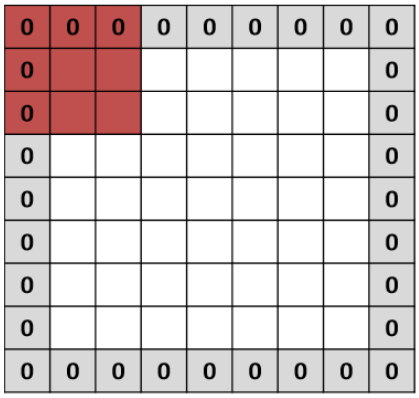

Figure 4.3: Padding example

4.2.2 Pooling

Convolutional networks may include local or global pooling layers which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. The main function of this layer is to reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer downsamples the volume spatially using the MAX operation, independently in each depth slice of the input volume as shown in figure (). In addition to max pooling, the pooling units can also perform other functions, such as average pooling which uses the average value from each of a cluster of neurons at the prior layer.

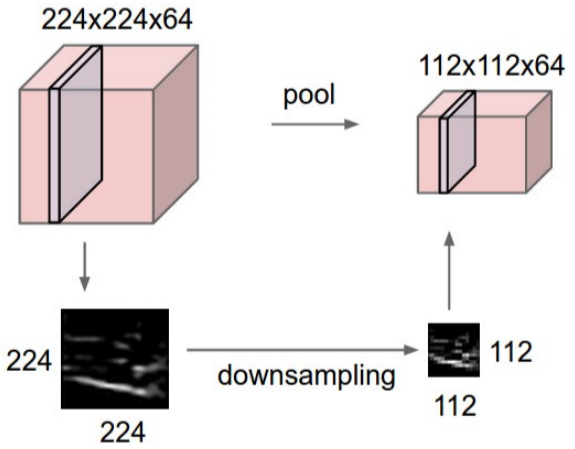

Figure 4.4: Pooling example

4.2.3 Fully-Connected Layer

The Fully Connected (FC) Layer is generally the last layer of a CNN. The output of the last Pooling Layer will be the input to the FC layer. This layer is called

Fully Connected because every node in the first layer is connected to every node in the second layer. The output is the result of a classification based on the features extracted by the previuos layer. FC layer is a traditional Neural Network containing a softmax activation function which returns a value ranging from 0-1 for each of the classification labels the model is trying to predict. Generally we can found one or two FC layers.

4.3 CNN architecture

The CNN classic architectures follow general design guidelines which machine learning practitioners can adapt to solve various computer vision tasks as image classification, object detection, image segmentation, and many other more advanced tasks. The first CNN architecture model was developed by Yann Lecun in 1998 [28] to identify handwritten digits. The current CNN architectures are inspired by the fist one. The basic idea of this model is to operate multiple convolutions and pooling and then transmitting the final signal via a fully-connected layer but due to the lack of adequate computing power, it wasn't possibleto extend this architecture to more complex applications.

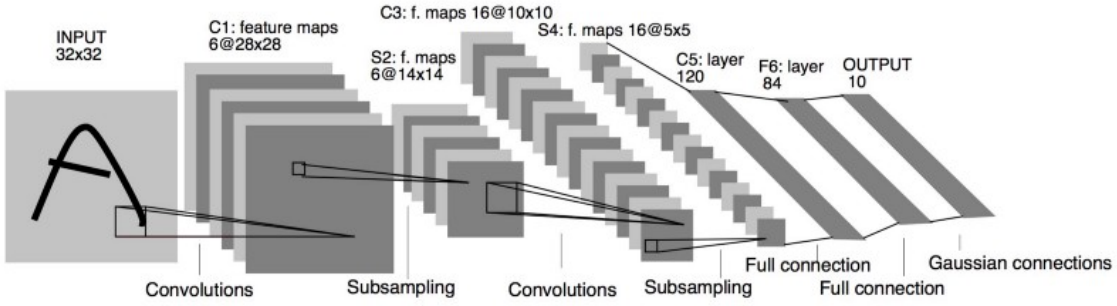

Figure 4.5: Lecun CNN

In 2012 Alex Krizhevsky presented AlexNet [29], an CNN architecture similar to LeNet-5 but considerably larger. The introduced innovation is the Rectified Linear Unit (ReLU) nonlinearity which permit a faster training. Furthermore, AlexNet implements a dropout step that consists in setting to zero a predefined percentage of layers parameters in order to limit the overfitting.
GoogleLeNet innovated the classic network architecture replacing the strategy of alternating convolutional and pooling layers with stacked inception modules.

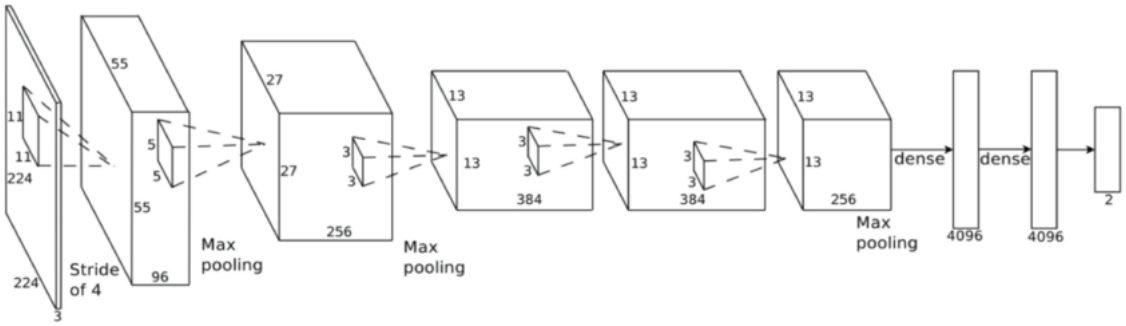

Figure 4.6: AlexNet CNN

Adding more layers eventually had a negative effect on the final performance, this phenomenon is known as *degradation problem*.The authors of the Deep Residual Network ResNet [30] propose a remedy to this degradation problem by introducing residual blocks in which intermediate layers of a block learn a residual function with reference to the block input.
Starting from these networks, other models have been developed; they can be more or less articulated but are based on the principles introduced by the classical network just seen.

Chapter 5

Classification system

The traditional classification problem aims assigning a specific category to a new instance on the basis of a training set and a previuos training process. In the field of machine learning, classification is a supervised learning problem. More in detail, each instance of the training dataset is correctly labeled with the corresponding category. The algorithm that implements the classification problem is called Classifier which map the input data into an output category. In this Chapter a novel method for the image classification problem is decribed. After the definition of the suitable categories, the instances contained into the input image are correctly labelled by the classifier. This is not an easy problem to solve. The main difference between object

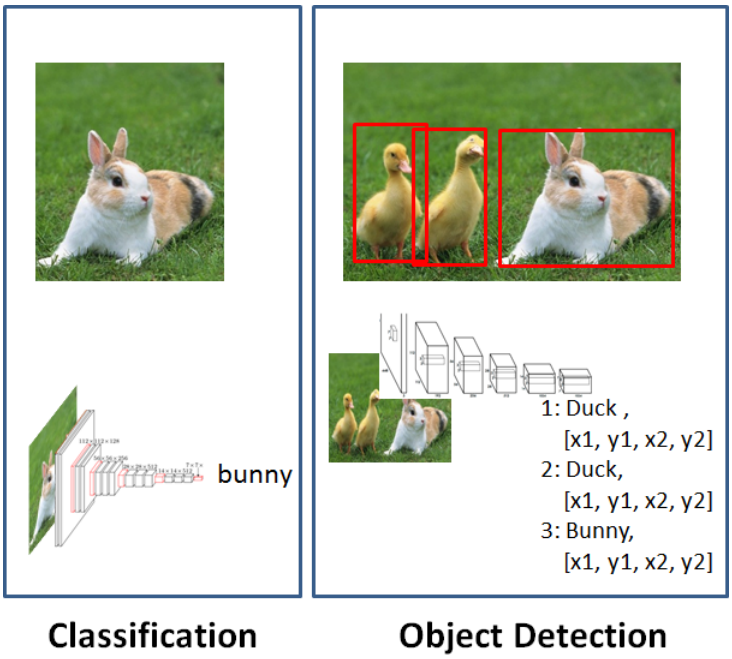

Figure 5.1: Classification example on the left and Detection example on the right

detection algorithms and classification algorithms is that the purpose of the object

detection problem is to draw a bounding box around the identified object. On the contrary the main objective of a common classification problem is to classifiy the entire image in one category. This difference is shown clearly in image (Fig. 5.1).

5.1 Problem Description

The purpose of the developed classifier is to to identify human presence into the observed scene. This means that twoclasses have been defined as the classifier output:

- *Human presence*, when there is at least one human operator into the observed image frame
- *Human absence*, when there are no human operators into the observed image frame

Unlike the classical image RGB classification problem, the input image for the developed classifer is obtained through the sensor fusion approach described in Sec. 3.2.1. Starting from the mapping matrix, it is possible to provide two 382x288 matrices (FoV of the thermal camera) which contain the temperature value (expressed in °C) and the depth value (expressed in cm), respectively. The result is a 382x288x2 matrix consisting of two channels. These objects are the inputs of the developed classifier and they are used to train the model and to predict the human presence.

The network model was created using the Keras framework: the network consists of several convolution, pooling and activation layers (see Sec. 4.2) of the Relu and Sigmoid functions (see Appendix B). The compilation was made by setting the type of loss as *categorical_crossentropy* indicated for classification problems and the training was carried out by setting the GPU appropriately to reduce waiting times.

5.2 Training

There are different available data sets for training Keras classifiers. Most of them contain RGB images which are then used to train the classifier. Since the case study is based on two sensory information, the online available datasets were impossible to use. Therefore, a new dataset has been acquired to build the classifier training set. The original sensor data have been organized in matrices as described in Sec. 5.1. The data matrices are provided by a ROS node which acquires the original depth data and the original thermal data from the cameras and accumulates data into two different matrices. A rosbag file is made to record these matrices. The acquired data

correspond to both scenarios with and without human operators. Several human configurations have been acquired to obtain a sufficient variability of the training dataset. On the other hand, several objects have been introduced into the scene without human operators in order to also have much more data representing 'Human absence' cases. Once the training data are recorded, they are processed through a Matlab script which assigns the correct label to all the samples and transforms the 2-matrix objects into a format compatible with Keras. The result will consist of two arrays: either n the number of samples of the dataset, an array nx288x382x2 (samples) and an array of length n (target).

5.3 Testing and Results

The trained network has been tested in different scenarios, at realtime. For each input frame, the Keras classifier returns a value between 0 and 1 that represents the probability of the human presence into the scene. More in detail, the temperature image and the depth image are published at 30Hz into the ROS network. Therefore, a pyton node reads these images and converts them into two matrices compatible with the Keras format. Finally, the matrices are sent to the developed Keras classifier which predicts the presence/absence of human operators. A value *thr* is set such that the values returned by the prediction are interpreted as follows:

```
float values;
float thr;
bool human_detected = true;

if(values > thr)
      return not human_detected;
else
      return human_detected;
```

The newtwork has been tested on a large variety of scenarios, including confusing objects. The test results have been organized in the confusion matrix of Table 5. Note that, unlike the classic RGB network that can confuse a plastic mannequine objects with shape similar to humans, the proposed approach is able to correctly distinguish only human operators thanks to the added thermal data (Fig. 5.2 and Fig. 5.3).

Figure 5.2: Human not detected from the developed image classificator

Figure 5.3: Human detected from the developed image classificator

		Actual Class		
		Human Detected	Human Not Detected	Total
Predicted Class	Human Detected	71	13	84
	Human Not Detected	11	65	76
	Total	82	78	160

Table 5.1: Classification Confusion Matrix

Chapter 6

Detection system

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Typically only a small number of instances of the object are present in an image, but there is a very large number of possible locations and scales at which they can occur.

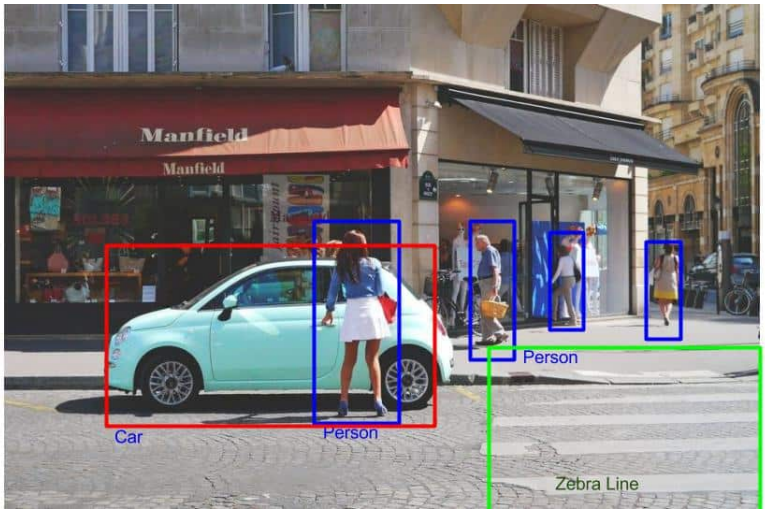

Figure 6.1: Object Detection example

The purpose of the detection is more ambitious than the classification: in this case the observed scene is not only classified as belonging to a certain category but the objective is to obtain the coordinates of the bounding boxes around each detected object must be predicted. The coordinates returned from the detector can be used for different goals and in different areas. Generally these systems can be used in the automotive field to allow the assisted driving: in this case the object to be recognized can be the pedestrian, road signs, probable obstacles, size of the road.

Nowadays there are many architectures and framework with a pre-trained CNN that allows to recognize some categories of objects and these networks are generally trained on the RGB images, so they can be used just on the traditional RGB cameras.
The new multi-sensor proposed approach aims at classifying and detecting the humans in the scene starting from a different type of images generated by thermal and depth sources. The performance of this CNN trained on a fusion of thermal and depth sources are then compared with the common pre-trained RGB models and with others CNNs trained only on the depth or on the temperature source.

6.1 Problem description

YOLO is the software which was adopted to develop a novel human detection system based on the multimodal sensory data. More in detail, YOLO is an open source neural network framework written in C and CUDA that supports CPU and GPU libraries and it is fully compatible with ROS environment. In this way it is possible to launch YOLO indicating the ROS topic to read (for example */camera/rgb/image*), the model used (yolov1, yolov2, tiny-yolov2 etc.) and the trained weights. In general, YOLO is trained by using RGB images and there are many pre-trained weights (available online) to test the detection on the topic of the RGB camera. Since we want to use the multimodal perception system and the sensor fused images, there are no existing YOLO pre-trained networks so the YOLOv3 CNN model has been re-trained to adapt the detection system to the Depth-Thermal (D-T) images. The following steps have been executed:

- definition of a *Human class*;
- exclusion of the YOLOv3 pre-trained classes from the prediction;
- building of the training data-set acquiring frames from D-T video stream;
- manual labelling of each frame;
- retrain of the YOLOv3 CNN weights.

6.2 Training

The training dataset is composed by a set of properly labeled images. Yolo requires that for each image (*image_name.jpg*) there is a corresponding text file (*image_name.txt*) which contains the information of the category of the detected objects

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6.2: Yolov3 CNN Model

(the label manually added) and the coordinates of the (manually drawn) bounding boxes. The images have been saved starting from the specific video stream and from a Python program that automatically saves the frames with a 2Hz frequency. Once the images have been saved, YOLO-Annotation-Tool (Fig. 6.3) has been used to label the images: the tool allows to draw the box around the object belonging to a specific class and automatically generates a file containing the information of the box and the class of the object. The tool generates text files with labeling information according to a format different from that required by Darknet for training; then another Python program is used to convert it to Yolo format.

Yolo-Annotation-Tool format:

```
<category number><bounding box left X><bounding box top Y>
<bounding box right X><bounding box bottom Y>
```

Darknet Yolo format:

```
<object-class> <x> <y> <width> <height>
```

Labeling is the longest time consuming operation of building a dataset infact it has been spent about one hour to label thousand images. Once the dataset has been built the *train.txt* and *test.txt* files containing the train and test images are listed

with the corresponding paths through a Python program and these files are finally sent to Yolo.
Before starting the training process, the following files must also be configured:

- *cfg/obj.data* This file specifies the number of classes to train, the path of the *train.txt* and *test.txt* files.
- *cfg/obj.names* This file lists the names of the categories for which we want to train. Every new category should be on a new line, its line number should match the category number in the .txt label files we created. In our case study the only defined category is HUMAN.
- *cfg/yolov3.cfg* The convolutional neural network model is defined in the *.cfg* file. Yolov3 (Appendix C) was used as a model whose batch size and subdivision parameters were modified. The batch size indicates the number of images used for each training step while the subdivision indicates the number according to which the batch will be divided to decrease GPU VRAM requirements. If you have a powerful GPU with loads of VRAM, this number could be increased.

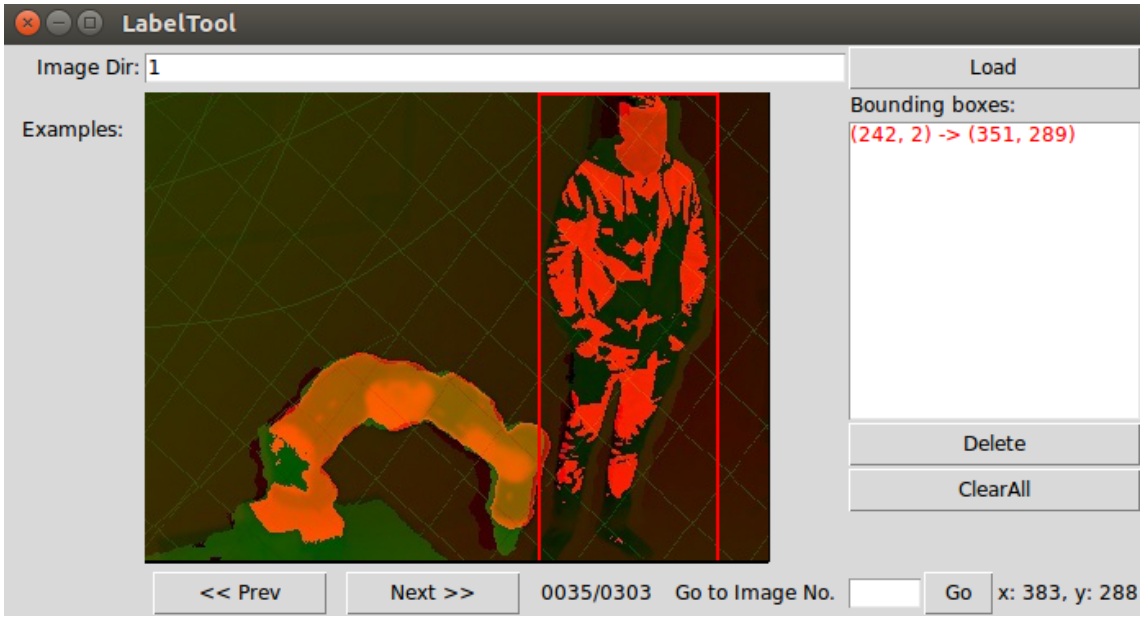

Figure 6.3: YOLO-Annotation-Tool in labeling operation

The training is performed on the Titan V GPU as the process is up to ten times faster than using CPU. The training goes on without ever stopping and saving the trained weights every hundred iterations in a backup file. During the training step, the framework publishes several parameters on the basis of which the user can decide whether or not to stop it. The published parameters are as follows:

- *Region Avg IOU* is the average of the IOU of every image in the current subdivision. IOU is a metric described in detail in Sec. 6.3. At 100% we have a perfect overlap of our bounding box and the target but we could probably have overfitted the data. A good value to stop the training is around 80%.
- *Avg Recall* is defined in code as recall/count, and thus a metric for how many positives Yolo detected out of the total amount of positives in this subdivision.
- *Count* is the amount of positives (objects to be detected) present in the current subdivision of images
- *Obj* is the average confidence score on those grids where there is an object
- *No Obj* is the average confidence score for the locations where there's no object.

6.3 Testing and Results

Metrics description

In general, each model is judged based on its performance on a data set, usually called the "validation / test" data set. This performance is measured using various statistics as accuracy, precision, recall. The use of one metric rather than another depends on the type of application in particular and in this case the Mean Average Precision (maP) has been adopted. Unlike classification problems for which simpler precision and recall metrics are indicated, for algorithms that solve object detection problems predicting the location of the object along with the classes, the Mean Average Precision is particularly used. The calculation of mAP is based on the calculation of various metrics such as precision, recall and IoU.
IoU (Intersection over union) is a metric to determine how accurately the model has detected a certain object. Computing the IoU consists in the division of the area of overlap between the bounding boxes by the area of union (see Fig. 6.4): we have "a match" when they share the same label and an IoU $>= 0.5$.

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.
The idea of mAP is conceptually similar in finding the area under the precision-recall graph replacing the precision value with the maximum precision for any recall.

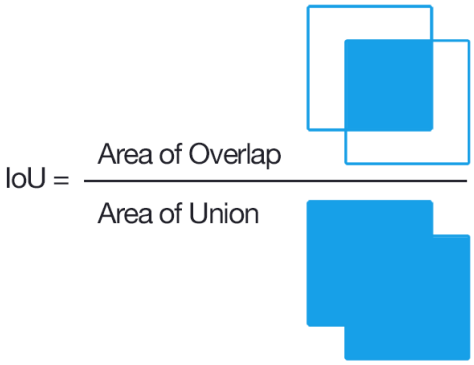

Figure 6.4: IoU calculation

Since the mAP does not take false positives and negatives into account, the percentage of the false detected was estimated as an additional metric.

Experimental Results

To test the performance of the *Sensor Fusion* approach, two others CNNs have been trained based on the depth and temperature information respectively. The networks have been trained and tested with the same input data set which differ only according to the source used.
Comparing the results of the Tab. 6.1 we can see that in the DT approach the mAP has a value comparable to that of the other approaches. However, the percentage of false positives is considerably lower than the others. In all cases the percentage of false negatives remains low.
The high percentage of false positives of single-source approaches is to be found in cases where hot objects (T based) or objects with shapes comparable to human ones (D based) can be exchanged as human (Fig. 6.5).
Note that even using pre-trained weights on the RGB camera, there are many false positives due to objects similar to human shapes and which make the classic RGB approach unusable in specific environments.
The CNN based on DT fusion can detect any human not often incurring false and negative positives and with good accuracy with a training dataset of only 1000 samples.
As shown in Fig. 6.6, in the case of the Sensor Fusion there is a good overlap between the predicted and the ground truth boxes and also both the hot robot and the presence of the dummy do not cause false positives.

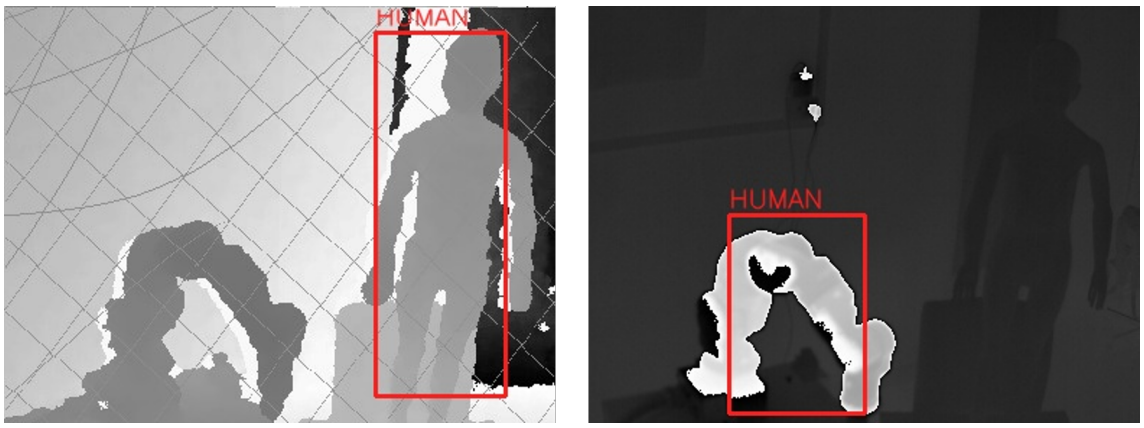

Figure 6.5: Two cases of human false positives: on the right a dummy detected in depth approach while on the left the hot moving robot detected in temperature approach.

Approach	**mAP %**	**False Positives %**	**False Negatives %**
Depth	65.09	26.87	17.53
Thermal	62.76	64.35	4.37
DT fusion	50.54	7.47	11.85

Table 6.1: CNNs Testing Results

Figure 6.6: Ground-truth and predicted boxes overlapping in Sensor Fusion based application

Chapter 7

Application

There are many application where human detection CNN can be applied as surveillance systems or to improve safety systems. In this case the application consists to robustly detect human operators in collaborative robot work cells through the multimodal perception system. Starting from the human detection, the system computes the separation distance between the robot and the operator and follows the line of the current regulations assuring the operators safety through the robot speed adjustment.

7.1 Segmentation

The basic assumption of the proposed segmentation algorithm (Fig. 7.1) is to process exclusively the information related to the dynamic objects present into the observed scene. This is because every point-cloud based strategy always represents a computationally heavy operation, then a *Background Segmentation* step has been initially developed to subtract the static environment. The cameras monitor the surroundings of the manipulator and the robot kinematic chain is fully visible. While the collaborative workspace is observed, the robot executes its task, thus becoming a dynamic entity. Therefore, the package *Real-time URDF Filter* [31] has been integrated at the beginning of the pipeline to distinguish the depth pixels belong to the robot model with respect to those belong to other dynamic entities and assign them a Not-a-Number (NaN) value. The background filtering has been developed through an efficient algorithm that performs the subtraction of a stored background, at pixel level: 50 frames of the static background are initially captured and the mean value of each pixel is stored in a memory area. At every acquisition, the current frame subtracts the static frame. The depth image is then converted to PCD and a uniform sampling filter can be applied to make the algorithm more reactive, by

selecting the clouds density.

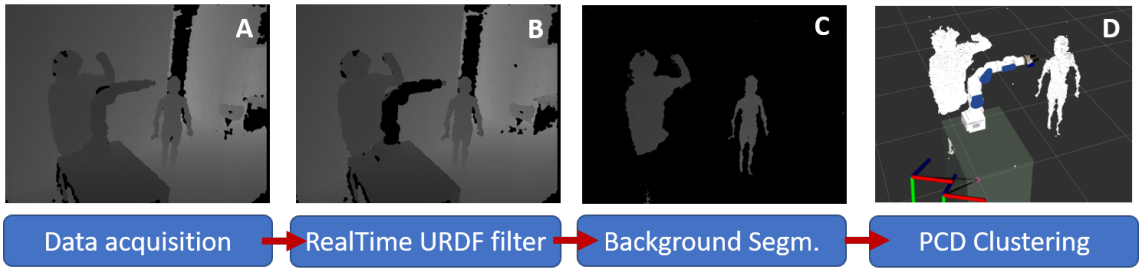

Figure 7.1: Segmentation Algorithm pipeline

Finally, the detection of dynamic entities is executed through a *PCD Clustering* step, which processes the point-cloud scene and provides some clusters as many as single dynamic areas are detected in the foreground. The *Euclidean cluster extraction* method is performed to distinguish all the clusters into the collaborative workspace. To compensate the sensors measurement noise that could sometimes provide false clusters, a first constraint is made by the definition of a minimum cardinality that the areas in the foreground should have, to be large enough to represent a human entity.

7.2 Human detection and cluster validation check

When a human is detected, the CNN returns the predicted bounding boxes that will be send to the segmentation pipeline to verify the human validation of each cluster generated in Sec. 7.1: each point of the cluster is transformed into depth pixel coordinates (by inverting Eq. 3.9-3.10). Since the bounding box is expressed in the thermal image plane, the selected pixel is converted into depth image coordinates through the mapping matrix (Sec. 3.2.1). If at least 50% of the cluster points belong to a bounding box, the cluster is labeled as human and passes the check. Figure 7.2 shows two clusters: the *red* human operator which is correctly detected by the CNN and the *yellow* plastic mannequin which is correctly not labeled as human.
Human Validation check is a fundamental step to compute the correct separation distance between human operators and the robot to apply the actual regulations of industrial robotic applications.

7.3 Separation distance computation

Therefore, the last one step of the segmentation pipeline (Fig. 7.2) identifies the nearest pair of points, one belong to the robot (P_R) and the other one belong to the

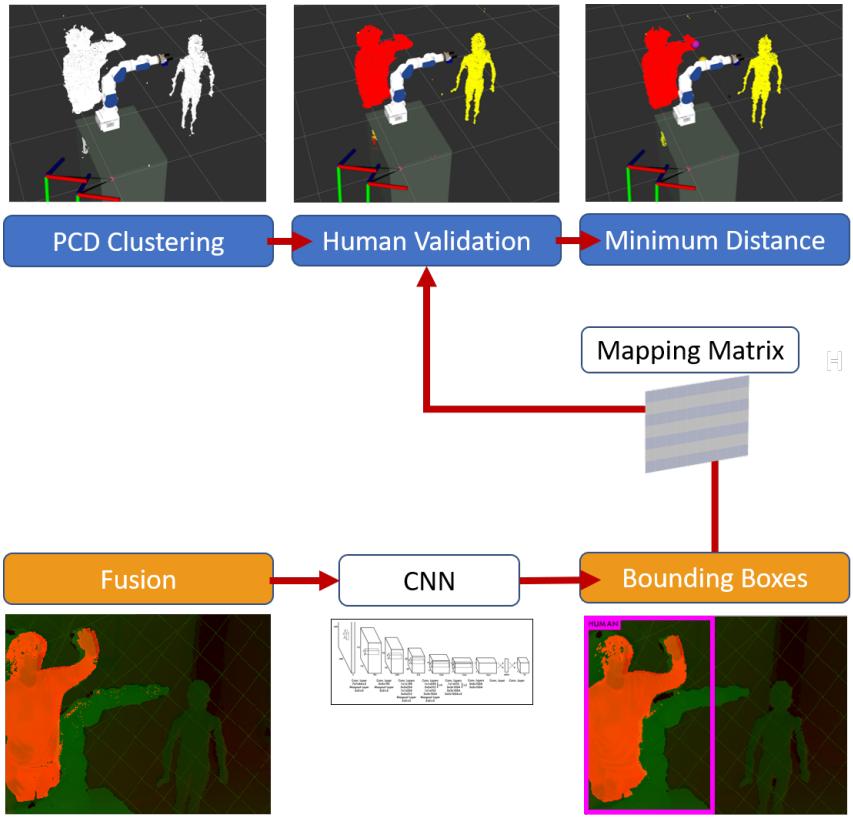

Figure 7.2: Human validation pipeline

operator (P_H), that minimizes the distance, i.e.,

$$P_H \in \mathcal{H}, P_R \in \mathcal{R} \mid d(P_H, P_R) \leq d(P'_H, P'_R) \quad \forall P'_H \in \mathcal{H}, P'_R \in \mathcal{R}, \tag{7.1}$$

where $d(\cdot, \cdot)$ is the Euclidean distance between two points, $\mathcal{H}$ and $\mathcal{R}$ represent the set of all points that belong to the operator and the robot, respectively. Moreover, alongside the HDT strategy, a robot modeling method has been implemented. The SoA assumptions factor only a singular representative coordinate of the robot (e.g., the end effector) or report its pose only in terms of either joint configurations or in terms of the Cartesian pose of the kinematic frames. Thus, they do not take into account the link volumes but only specific points. On the contrary, the proposed solution considers the entire robot volume. Primitive shapes, e.g., ellipses and spheres as in [32] and [33], have been used to model robot links.

The kinematic chain has been padded through dummy frames to protect the robot homogeneously, and creating a virtual 0.10 m diameter security sphere around each frame (compatible with the used robot). Made these assumptions, the pair of closest points can be immediately identified (Eq. 7.1): the algorithm calculates the distance

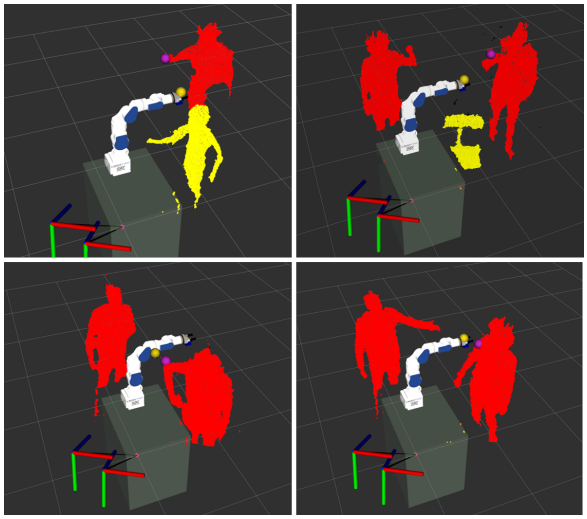

Figure 7.3: Identification of the minimum distance points between the whole robot and the closest human operator.

between all points of the verified clusters and the point origin of every robot frame. The robot point, P_R, will be on the virtual sphere around the identified frame. This step strongly justifies the choice of a point cloud based approach. In fact, it provides satisfactory accuracy and precision: it allows tracking humans also when they are not completely visible from the camera view, unlike common skeleton-based techniques; it is not necessary that human operators are in front of the camera view because the point cloud will recognize them anyway; the approach identifies more detailed body parts, e.g. a elbow, the head, an hand, the chin or the chest. Figure 7.3 shows the results: the developed CNN distinguishes human operators belong to the *Human class* (in red) from other clustered objects (in yellow), i.e. a plastic mannequin and a chair, which are not labeled as humans and they are not considered for the safety separation distance computation, even if they are possibly closer to the robot. Note that the closest human cluster is indirectly selected in multi-humans scenarios.

Chapter 8

Conclusion

This work is aimed at finding a robust solution to the human detection problem. The applicability of the novel approach is concerned to collaborative workspace, where the human operator safety is the main objective. The power of this approach lies in the sensor fusion and the thermal camera integration into the perception system. The extrinsic calibration algorithm between the thermal camera and depth camera has been developed to obtain a sufficient accuracy when overlapping the two images. This is an advanced sensor fusion strategy which performs two classic Machine Learning problems: Image Classification and Object Detection. The trained CNNs have returned excellent results despite the small number of samples in the training dataset. The most time consuming step has been the acquisition of the new dataset and the manual labeling of images to accurately train the network.
It has been shown that through this network it is possible to distinguish the cases of the presence of a mannequin or shapes similar to human body with respect to the real human one. This feature makes the network usable for different purposes such as video surveillance or how it has been analyzed in detail, allowing cooperation between human and robot in work cells. In particular, in addition to the application in industrial areas, if we think of a collaboration between human and robot in a clothing store where there are operators, mannequins, clothes and many forms that recall the human features, this algorithm could have robust results.

As possible future developments, the performance of the sensor fusion algorithm could be improved by increasing the frequency of publication of the merged image video stream. To increase the frequency of publication, the Sensor Fusion algorithm, then the Mapping algorithm could be rewritten in CUDA for some instructions and then run on the GPU significantly increasing the performance.
In addition, the networks trained on a thousand samples could significantly increase

the performance for human detection by increasing the number of samples of the training dataset and doing the network retraining.
Going in the specific application for human-robot collaboration (Sec. 7.2), possible improvements could be to increase the spatial coverage of the detection by installing a greater number of depth and thermal cameras with different angles and FoV. In addition, the Sensor Fusion could be extended by merging the use and fusion of other types of sensors into the vision system.
It would then be interesting for the application not only to carry out the human detection but also to make a prediction on the type of task or action of the human starting from his movements. Predicting a certain human behavior could favor the algorithms development that safeguard the security and that helps humans in work or daily activities.

Appendix A

Depth Intrisic Calibration .yaml file

```
image_width: 640
image_height: 488
camera_name: depth_A00362907668053A
camera_matrix:
  rows: 3
  cols: 3
  data: [587.0563067729362, 0, 320.6153422514201,
         0, 590.2429336923416, 250.4053454922714,
         0, 0, 1]
distortion_model: plumb_bob
distortion_coefficients:
  rows: 1
  cols: 5
  data: [-0.2248186612495552, 0.4608152675220439,
         -0.009977982350402648, 0.0125772926680182, 0]
rectification_matrix:
  rows: 3
  cols: 3
  data: [1, 0, 0, 0, 1, 0, 0, 0, 1]
projection_matrix:
  rows: 3
  cols: 4
  data: [588.2620239257812, 0, 326.7864095571531, 0,
         0, 592.9213256835938, 245.3447312767712, 0,
         0, 0, 1, 0]
```

Appendix B

Keras Classificator Model

```
model = Sequential()

model.add(Conv2D(32, (5, 5), strides=(1, 1),
                 input_shape=(288, 382, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.50))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.50))

# this converts our 3D feature maps to 1D feature vectors
model.add(Flatten())
model.add(Dense(2))
model.add(Dropout(0.5))
model.add(Activation('sigmoid'))
# COMPILE

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

```
with tf.device('/device:GPU:0'):
    model.fit(x_train, y_train,
              batch_size=BATCH_SIZE,
              epochs=EPOCHS,
              #verbose=1,
              #validation_data=(x_test, y_test)
              )


score = model.evaluate(x_test, y_test)
```

Appendix C

Yolo v3 Model

```
[net]
batch=64
subdivisions=8
width=382
height=288
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
max_batches = 20000
policy=steps
steps=-1,100,20000,30000
scales=.1,10,.1,.1

[convolutional]
batch_normalize=1
filters=16
size=3
stride=1
pad=1
activation=leaky
```

```
[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[maxpool]
```

```
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=1

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=30
activation=linear

[region]
anchors = 1.08,1.19,  3.42,4.41,  6.63,11.38,  9.42,5.11,  16.62,10.52
bias_match=1
classes=1
coords=4
num=5
softmax=1
jitter=.2
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .6
random=1
```

Bibliography

[1] Robots and robotic devices - Safety requirements for industrial robots. Part 1: Robots. Technical report, International Organization for Standardization, 2011.

[2] Robots and robotic devices - Safety requirements for industrial robots. Part 2: Robot system and integration. Technical report, International Organization for Standardization, 2011.

[3] Robots and robotic devices - collaborative robots. Technical report, International Organization for Standardization, 2016.

[4] Jochen Heinzmann and Alexander Zelinsky. Quantitative safety guarantees for physical human-robot interaction. *The International Journal of Robotics Research*, 22(7-8):479–504, jul 2003.

[5] F. Flacco, T. Kroger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *2012 IEEE International Conference on Robotics and Automation*. IEEE, may 2012.

[6] Nimmi Arora, Diana Martins, Danielle Ruggerio, Eleni Tousimis, Alexander J. Swistel, Michael P. Osborne, and Rache M. Simmons. Effectiveness of a noninvasive digital infrared thermal imaging system in the detection of breast cancer. *The American Journal of Surgery*, 196(4):523–526, oct 2008.

[7] Michael Vollmer and Klaus-Peter Möllmann. *Infrared Thermal Imaging*. Wiley-VCH Verlag GmbH & Co. KGaA, dec 2017.

[8] Herbert Kaplan. *Practical Applications of Infrared Thermal Sensing and Imaging Equipment.* SPIE Publications, 2007.

[9] Baopu Li, Haoyang Jin, Qi Zhang, Wei Xia, and Huiyun Li. Indoor human detection using RGB-d images. In *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, aug 2016.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages pp. 886–893 vol. 1. IEEE, 2005.

[11] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015.

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[13] Kaiyang Zhou and Adeline Paiement. Detecting humans in rgb-d data with cnns. In *15th IAPR International Conference on Machine Vision Applications (MVA), Nagoya University, Nagoya, Japan*. IEEE, 2017.

[14] Stephen Vidas, Peyman Moghadam, and Michael Bosse. 3d thermal mapping of building interiors using an RGB-d and thermal camera. In *2013 IEEE International Conference on Robotics and Automation*. IEEE, may 2013.

[15] Labor project, *https://www.labor-project.eu*.

[16] Anis Koubaa, editor. *Robot Operating System (ROS)*. Springer International Publishing, 2019.

[17] A. Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing, 2017.

[18] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2014.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[20] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017.

[21] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[22] Wilfried Elmenreich. An introduction to sensor fusion. 03 2019.

[23] Bo Yang, Zhongliang Jing, and Hai-tao Zhao. Review of pixel-level image fusion. *Journal of Shanghai Jiaotong University (Science)*, 15:6–12, 02 2010.

[24] Thomas Luhmann, Johannes Piechel, and Thorsten Roelfs. Geometric calibration of thermographic cameras. In *Thermal Infrared Remote Sensing*, pages 27–42. Springer Netherlands, 2013.

[25] J. Rangel and S. Soldan. 3d thermal imaging: Fusion of thermography and depth cameras. In *Proceedings of the 2014 International Conference on Quantitative InfraRed Thermography*. QIRT Council, 2014.

[26] James Bowman and Patrick Mihelich. Camera_calibration tool, *http://wiki.ros.org/camera_calibration*.

[27] Bo Yang, Zhong liang Jing, and Hai tao Zhao. Review of pixel-level image fusion. *Journal of Shanghai Jiaotong University (Science)*, 15(1):6–12, feb 2010.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[31] N. Blodow. Realtime urdf filter, 2012.

[32] Su Il Choi and Byung Kook Kim. Obstacle avoidance control for redundant manipulators using collidability measure. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*. IEEE, 1999.

[33] P. Bosscher and D. Hedman. Real-time collision avoidance algorithm for robotic manipulators. In *2009 IEEE International Conference on Technologies for Practical Robot Applications*, pages 113–121. IEEE, 2009.