Robotic sole Deburring: from Burrs Identification to Path Planning from Human Demonstration

Tesi di Laurea Magistrale in
Mechanical Engineering - Ingegneria Meccanica

Author: **Luigi Cacciani**

Student ID: 995075
Advisor: Prof. Andrea Maria Zanchettin
Co-advisors: Prof. Paolo Rocco, Ing. Alessandra Tafuro
Academic Year: 2022-23

Abstract

The landscape of shoe manufacturing is a complex blend of craftsmanship, technology, and innovation: tracing its roots back to 7000 BCE, the industry has continuously evolved. In the contemporary era, the Industry 4.0 framework is reshaping production systems, emphasizing the transition from conventional CNC robots, which offer speed and efficiency but lack of adaptability, to intelligent manufacturing. Footwear manufacturing faces a critical operation known as sole deburring, the removal of unintended protrusions on the edges of the sole, called *burrs*. Sole deburring is traditionally a manual task, undertaken by skilled workers. This thesis focuses on developing a robust path-planning pipeline for autonomous robotic deburring of soles. Key problems addressed include the accurate detection and precise segmentation of soles with burrs in industrial settings, dealing with complexities such as varying lighting, diverse sole appearances, and potential occlusions. Employing Deep Learning approaches, namely *Detectron-2* and *Pix2Pix*, robust performance across diverse scenarios has been achieved Additionally, a novel, efficient and automated method for burrs identification in shoe soles has been developed leveraging image processing techniques. Identifying the optimal orientation for the cutting tool during sole deburring is a complex task traditionally based on human experience. The thesis introduces a novel solution by teaching the robot this orientation from videos of expert demonstrations. The pose of the tool in the videos is extracted with a pose estimation neural network, namely *EfficientPose*. The final challenge involves the robot's utilization of the outcomes from the preceding computations to autonomously formulate an efficient deburring path. Experimental results showcase the high precision of the developed steps, underlining the potential of these technologies in advancing the field.

Keywords: Robotic sole deburring, Object detection and segmentation, Burrs identification, Pose estimation, Learning from Demonstration, Path planning.

Abstract in lingua italiana

La produzione di calzature è una complessa combinazione di artigianato, tecnologia e innovazione: dalle radici del 7000 a.C., l'industria è in continua evoluzione. Nell'era contemporanea, la rivoluzione dell'Industria 4.0 sta ridefinendo i sistemi di produzione, incentivando il passaggio dalle tradizionali macchine *CNC*, che offrono velocità ed efficienza ma mancano di adattabilità, alla robotica intelligente. Il ciclo produttivo di una calzatura prevede un'operazione critica nota come *sbavatura* della suola, ossia la rimozione di materiale indesiderato ai bordi della suola chiamato "bava". La sbavatura è tradizionalmente un compito manuale, eseguito da lavoratori esperti e specializzati. Questa tesi si concentra sullo sviluppo di un processo di pianificazione della traiettoria per la sbavatura autonoma delle suole da parte di un robot. I maggiori problemi affrontati includono la precisa identificazione e segmentazione delle suole con bave in ambienti industriali, gestendo complessità come illuminazione variabile, aspetto diversificato delle suole e potenziali occlusioni. Utilizzando approcci di Deep Learning (DL), in particolare *Detectron-2* e *Pix2Pix*, è stato possibile ottenere un risultato robusto in diversi scenari. Inoltre, è stato sviluppato un nuovo metodo efficiente e automatizzato per l'identificazione delle bave nelle suole, sfruttando tecniche di elaborazione delle immagini. Individuare l'orientamento ottimale per lo strumento di taglio durante la sbavatura è un compito complesso, tradizionalmente basato sull'esperienza del lavoratore. Questa tesi introduce una soluzione innovativa, insegnando al robot l'orientamento da video di dimostrazioni di esperti. La posizione dello strumento nei video è estratta con una rete neurale di stima della posa, chiamata *EfficientPose*. Infine, è stato studiato l'utilizzo dei risultati delle fasi precedenti per la formulazione autonoma di una efficiente traiettoria di sbavatura. I risultati sperimentali evidenziano l'alta precisione di ogni fase sviluppata, sottolineando il potenziale di queste tecnologie nell'evoluzione del settore.

Parole chiave: Sbavatura robotica delle suole, Rilevamento e segmentazione di oggetti, Identificazione delle bave, Stima della posa, Apprendimento tramite dimostrazioni, Pianificazione della traiettoria.

Contents

1 | Introduction

1.1. Robotic deburring: general overview

Shoe manufacturing stands as a complex and dynamic industry, combining craftsmanship, technology, and innovation meeting diverse preferences and functional needs.

The historical trajectory of shoe production, dating back to 7000 BCE [6] with early sandals, showcases a continual evolution toward sophistication. The Middle Ages introduced the "Turnshoe method", refining techniques for enhanced comfort and durability. The 1500s witnessed the advent of the "hand welted" method, a labor-intensive yet durable approach still present today in its evolved form, "Goodyear welting". Industrialization in the mid-18th century marked a significant shift, with sewing machines making their debut, even though manual labor persisted. The 1910s brought stitchless shoes, optimizing the process with mass-manufactured and glued-on soles.

Traditionally, shoe manufacturing demands significant manual labor and skilled expertise, often in challenging work environments. However, the landscape is evolving with the rise of small-scale, flexible production, aligning with modern customer preferences. In the contemporary era, smart factories in Industry 4.0 are becoming indispensable for small-batch, multi-product flexible production systems [41]. The transition from digital to intelligent manufacturing is pivotal. Industrialized nations prioritize intelligent manufacturing technology as the advantages of traditional manufacturing diminish. In today's market, characterized by small batches and multiple varieties, production lines must dynamically reconfigure process paths and manufacturing units [23].

In the domain of footwear manufacturing, the process of sole deburring emerges as a critical operation for ensuring the quality and safety of shoes. A burr, as defined by [15], is an unintended byproduct during manufacturing, manifesting as small projections extending beyond the designed workpiece surface and having a relatively small volume. In the context of shoe sole production, burrs often arise from the injection molding process, specifically in the creation of outsoles (Figure 1.1). This process involves the interaction of two molds, each representing half of the sole's shape. Liquified rubber injected into this mold assembly solidifies. The confined space between the molds results in the rapid solidification of this material, giving rise to burrs along the sole's edges. Due to the direct integration of the sole into the final shoes, the removal of these burrs, known as deburring, becomes an imperative quality control measure.

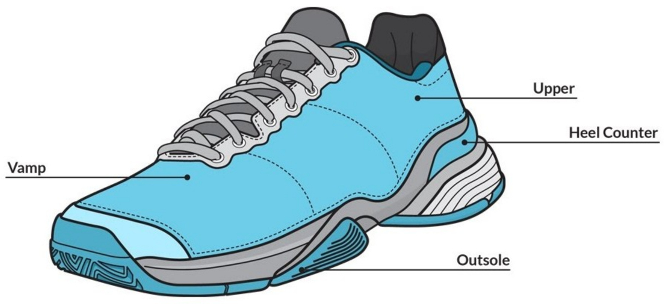


Figure 1.1: Shoe components

Traditionally, skilled human workers have undertaken the meticulous task of deburring, leveraging their expertise to visually analyze workpieces and identify regions requiring precise removal. However, the paradigm is shifting with the rapid evolution of automation and robotics. Conventional CNC robots, designed for repetitive tasks, demonstrate efficiency for high-volume production but struggle when confronted with the need for adaptability. The integration of more intelligent robotics represents a paradigm shift beyond conventional capabilities.

The journey through this innovative realm, however, is not without challenges [54]. The traditional method, relying on human expertise, is renowned for its adaptability since every workpiece is crafted. It struggles however when high production rates and low costs are needed. Conventional CNC robots offer speed and efficiency but lack the ability to adapt and change in response to different models or required customizations (Figure 1.2). The emergence of intelligent robotics aims to address these limitations, introducing a system able to recognize and react.

The transformative landscape of intelligent manufacturing, marked by the integration of new-generation artificial intelligence (AI) technology, has earned significant attention in recent years. The paradigms of smart manufacturing (SM) and intelligent manufacturing (IM) [79] encapsulate the integration of manufacturing and advanced information technologies. Notably, the evolution of IM, as described by Zhou et al. [39], spans three stages: digital manufacturing, digital-networked manufacturing, and **next-generation**

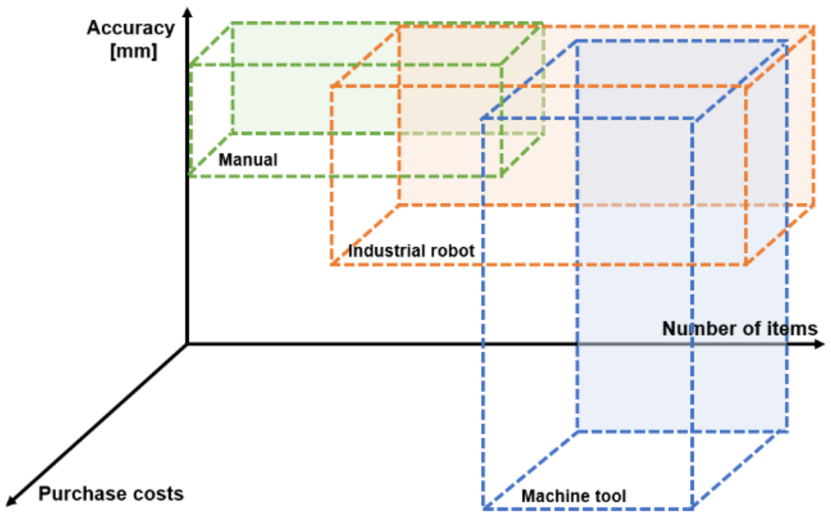


Figure 1.2: Comparison of deburring technologies [54]

intelligent manufacturing (NGIM). The current era, characterized by strategic breakthroughs in AI, is moving into the intelligent stage of informatization, with new-generation AI technology at its core.

Machine learning (ML) and Deep learning (DL) arise as the core of the transformation. Sundar Pichai, CEO of Google, has emphasized the pervasive impact of machine learning across diverse products and services (*"Machine learning is a core, transformative way by which we're rethinking everything we're doing"*). Machine learning, as a subfield of AI, stands as one of the most used approaches to drive recent industrial advantages and commercial applications. The advent of deep learning, a subfield of ML vaguely inspired by the human brain, has witnessed remarkable advancements and applications across various domains [66]: it has been successfully applied to diverse problems in areas such as natural language processing, cybersecurity, business, virtual assistants, visual recognition and robotics.

Robotics has seen notable advancements through the incorporation of machine learning and deep learning algorithms, which offer solutions for autonomy and decision-making [13].

This intersection of intelligent manufacturing, deep learning and robotics frames the backdrop for exploring intelligent robotic deburring in footwear manufacturing. This thesis deals with this transition, highlighting potential advantages while addressing associated challenges. Human expertise remains essential and the coupling with cutting-edge technologies emerges as a promising avenue for the progress of manufacturing.

1.2. Problem statement and thesis research aim

This thesis focuses on developing a robust path-planning pipeline for the autonomous robotic deburring of shoe soles. The primary aim is to create a system that can adapt and operate independently, irrespective of the sole's position or dimensions.

Key problems addressed in this research include:

1. **Detection and segmentation of soles with burrs:**
   - Accurate detection of the soles with burrs in the work plane
   - Precise segmentation of the detected sole
2. **Identification of the burrs:**
   - Identify and delineate regions on the sole that require deburring
3. **Cutting tool orientation:**
   - Determining the orientation of the cutting tool during the deburring process to ensure efficient material removal
4. **Path planning:**
   - Creating a comprehensive path planning pipeline that incorporates the results from previous calculations

Detecting and segmenting soles with burrs present unique challenges that demand both precision and robustness, particularly in the dynamic context of industrial settings. Achieving high precision is crucial, directly impacting the subsequent burrs identification step and the deburring path calculation. The complexities of industrial conditions add several challenges: factors such as varying lighting, diverse sole appearances, and potential occlusions may introduce unpredictability. The selected method must demonstrate robust performance, delivering consistent results across diverse industrial scenarios.

Accurate **burrs identification** is a problem of high interest in the industrial landscape, where the presence of burrs is an inevitable challenge. While these undesired protrusions necessitate removal, identifying and locating them can be time-consuming and costly. Manual deburring traditionally relies on the expertise of skilled workers who visually inspect the workpiece to discern regions to be removed. This thesis proposes a novel method for burr identification for shoe soles. Leveraging image processing techniques, the proposed approach offers an efficient and automated means of precisely identifying burrs.

Since the sole deburring is performed through a cutting tool of a peculiar shape, identifying the optimal **orientation for the cutting tool** during the operation is a complex task. Typically, skilled workers rely on their experience and visual judgment to determine the right pose during the manual deburring process. In this thesis, then, it has been decided to teach this orientation to the robot from the expert demonstration. The conventional approach involves physically guiding the robotic arm through various deburring sequences to record the poses. However, this method has limitations, as it restricts the demonstrator's freedom and agility in showcasing the optimal deburring technique. In this thesis, a novel solution is proposed to overcome these challenges: the optimal tool orientation is extracted from videos of experts performing deburring.

The final challenge tackled involves the robot's utilization of the outcomes from the preceding computations to autonomously formulate an efficient **deburring path,** ensuring precise spatial alignment and integrating the acquired orientation knowledge.

These aspects are individually examined and described in detail in the subsequent chapters.

1.3. Thesis structure

This thesis is organized into six chapters, each dedicated to a specific aspect of the robotic deburring study, along with a conclusive chapter summarizing key findings and suggesting potential future developments:

- **Chapter 2:** explores the current state-of-the-art works related to burrs identification methods and path planning for deburring operations;
- **Chapter 3:** provides a detailed overview of the proposed solution for sole detection and segmentation;
- **Chapter 4:** describes the custom-made method developed for burrs identification;
- **Chapter 5:** presents the proposed solution for accurately estimating the pose of the deburring tool during the demonstrated deburring operation;
- **Chapter 6:** explores the comprehensive path planning pipeline, detailing the integration of the obtained results and showcasing real-world experiments with a robotic arm;
- **Chapter 7:** conclusions and future developments

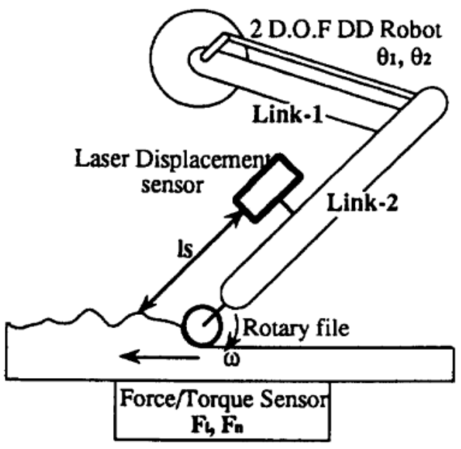
2 | State of the art

2.1. Burrs identification methods

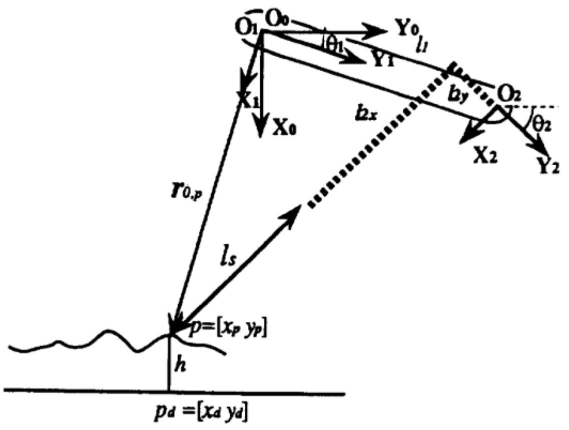
A burr is a geometric anomaly that manifests as a material protrusion on the surface of a workpiece during its manufacturing process, resulting in an extension beyond the initially specified dimensions. Notably, burrs exhibit a negligible volume relative to the overall workpiece but remain generally undesirable, although inevitable [14]. Consequently, it is necessary to employ precise and efficient detection methodologies for the identification of burrs.

Currently, a diverse collection of methodologies exists for the detection and measurement of burrs. The selection of the most suitable system is highly application-specific: workpiece's features, available instruments and required burr quantities (height, thickness, volume, hardness, etc.) result in an extensive variety of possible methodologies [14]. One possible categorization of techniques for the identification and quantification of burrs is suggested by [16]: burrs measurement methods can be classified into contact and non-contact methods. Multiple contactless systems are present in literature with lasers and cameras representing some of the key options. Due to the non-uniformity and considerable size variability of burrs, when touch sensors are employed for burr data measurement, the presence of nearby burrs can disturb the measurement due to the conical shape of the tracer point. Likewise, when utilizing laser sensors, issues related to interference and diffraction tend to manifest [45]. Vision systems, together with image processing techniques, offer a promising solution to address these aforementioned challenges.

Direct measurement of the burrs' size is performed in [68]. This measurement is accomplished by employing a laser displacement sensor (LDS) mounted on the deburring robot. The schematic configuration is illustrated in Figure 2.1a. A beam of light is directed towards the workpiece's surface with burrs, allowing for precise measurement of the distance $\boldsymbol{ls}$. By combining this distance with knowledge of the robot's configuration and the desired surface location, the burr's height $\boldsymbol{h}$ is determined through vector operations, as depicted in Figure 2.1b.



(a) Robot configuration, equipped with LDS.



(b) Schematic of the vectors.

Figure 2.1: Robotic setup with LDS [68].

Likewise, in [40] a laser sensor mounted on the robot's wrist is employed to perform scanning and measurements of workpiece surfaces that have welding beads. The collected surface data is then used to extract the center line and measurements of the bead. Building on the analogy between a welding bead and a burr, the approach described in [46] utilizes the previously outlined method for burr detection. The adaptation involves matching the measured shape with a modeled one using the least squares method to derive the parameters characterizing the burr.

The study proposed by Wulf [81] leverages the temperature variance between burrs and the steel slab during the cutting process, employing a high-temperature thermographic camera to detect and discern the burrs based on thermal contrast. Nonetheless, it's important to note that the applicability of this approach is constrained to the particular steel slab cutting process.

One of the prior examples of the employment of visual systems for burrs detection is represented by [45]. In this approach, a 2D image is captured and subsequently converted into a binary image, consisting solely of black and white pixels (with intensities of 1 and 0, respectively). This transformation is achieved through a process known as thresholding, wherein a threshold value is used to set pixels with intensities higher than the threshold as 1, while the rest are set to 0 [63]. The contour of the burrs is then extracted, and pertinent burr data is derived using a deburring model that correlates the derived values with the burr contours.

Thresholding for detection is exploited also in [51]: image processing techniques are harnessed using a charge-coupled device (CCD) camera, which is a sensor capable of con-

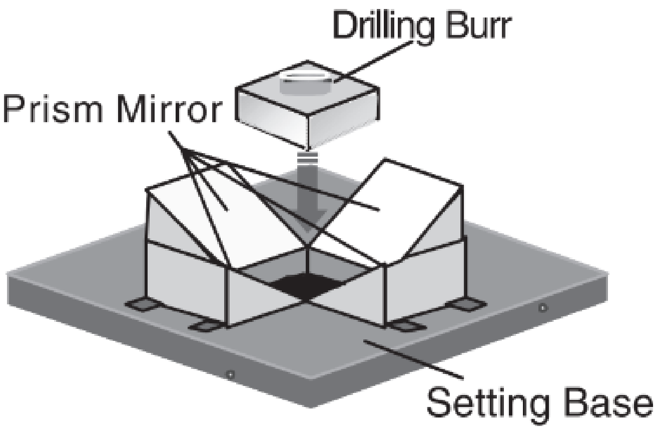
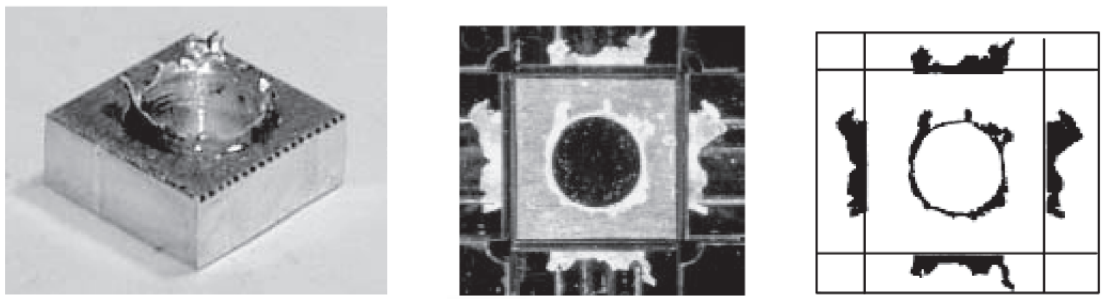


Figure 2.2: CCD camera configuration for burrs detection [51]

verting light into electrical charge, enabling the acquisition of high-quality digital images [7]. This CCD camera is positioned vertically above the surface of the burr specimen to capture images, together with four mirrors at 45°, as shown in Figure 2.2, which allow to capture the 3D shape of the burr. The images are rearranged in a 2D configuration and the thresholding procedure is then applied: pixels corresponding to the burrs are assigned a zero intensity (depicted as black), enabling the measurement process. An exemplifying pipeline of the process is depicted in Figure 2.3.

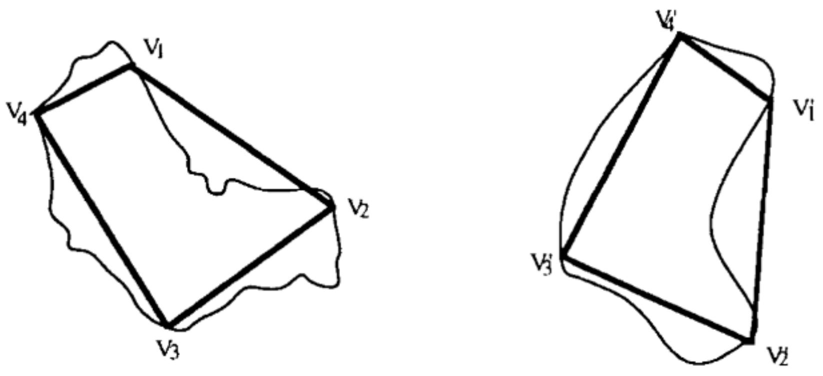


(a) Burrs to be measured (b) Captured image (c) Processed image

Figure 2.3: Threshold image Processing with CCD camera [51].

A comparable approach is suggested in [24], which utilizes a CCD camera for capturing an image of the workpiece. Subsequently, thresholding is applied to obtain the segmentation of the burrs in the image. To further refine the detection process, an edge detection algorithm, specifically the Canny operator [22], is implemented to extract the contours of the burrs.

The visual system introduced by Tsai [75] offers an illustration of contour matching as a detection method. This approach involves a comparison between the ideal geometric profile and the actual one to identify peripheral defects. The assumption in this study is that irregular burrs exhibit a pronounced variation in curvature, whereas a nominal profile appears smoother. To execute this, boundary points are chosen from the smooth segments present in the model containing burrs, and a polygon is generated by connecting these points (as shown in Figure 2.4). The same polygon is then sought in the nominal profile, and the two profiles are overlaid. Any disparities between the two images serve to identify and characterize the burrs.



(a) Object with burrs (b) Model object

Figure 2.4: Contour matching for burrs detection [75].

A recent frontier comprises the use of Neural networks for burrs detection. An example is provided in the paper by Rahul and Chiddarwar [48], where they address the problem of *domain shift*. This issue arises when the dataset is captured from one domain (with specific conditions of intensity or texture in the images) and the trained model may struggle to recognize burrs in images from a different domain. To overcome this challenge, the authors propose a novel dataset augmentation method. In the first step of their approach, the dataset is augmented by modifying intensities and textures while preserving the structures of the burr features. This is done to ensure that the network is not biased towards specific appearances. In the second step, the authors aim to discard potentially misleading associations between the object of interest and the background. They achieve this by employing a preventative pseudocorrelation augmentation (PPA) method, which separates the object from its surroundings. The pipeline of this augmentation method is illustrated in Figure 2.5.

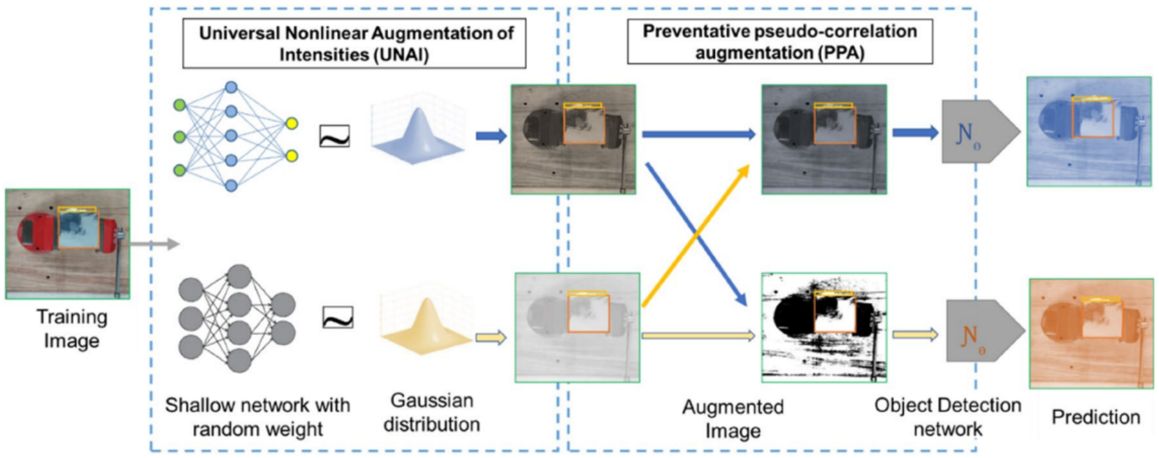

Universal Nonlinear Augmentation of Intensities (UNAI)
Preventative pseudo-correlation augmentation (PPA)
Training Image
Shallow network with random weight
Gaussian distribution
Augmented Image
Object Detection network
Prediction

Figure 2.5: Pipeline for image augmentation [48]

2.2. Path planning for deburring operations

The robotic deburring process comprises two steps: planning and motion execution [53]. In the planning phase, the predominant task is to obtain the robot path, even though considerations on the tool-path correction and machining parameter estimation are typically considered in this stage. Subsequently, the motion execution part translates these planned trajectories into physical deburring actions: mechanical deburring techniques are employed. To increase accuracy, the incorporation of real-time feedback control through force measurements can be included [55].

There exist different approaches in the literature to generate a path for robotic deburring. The three primary ones can be listed as follows [53]:

- **CAD/CAM-Based approach:** utilizing Computer-Aided Manufacturing (CAM) software in conjunction with the Computer-Aided Design (CAD) model of the workpiece, this approach selects the edges requiring deburring and the CAM software plans, generates, and simulates the tool path. The resultant is often a form of numerical control (NC) code, such as G-code, or robot-specific languages: in the latter case, the CAM software performs a compilation step [77], so the quality of the robot program is closely related to the post-processor within the CAM software.
- **Sensor/Vision-Based approach:** leveraging specialized sensors or vision systems to detect and interpret critical information about the workpiece's geometry, the edges are detected and subsequently the deburring path is defined.
- **Teaching through Human Demonstration:** entailing the physical guidance of an operator, the robot records the demonstrated path. The demonstration could be

either direct (like kinesthetic teaching [12], Figure 2.6a, the end-effector is moved by hand and joint positions are logged), indirect (the robot is directed through devices like a teach pendant or haptic devices [21]) or non-interactive (the robot is uninvolved during the demonstration, with methods such as vision systems analyzing demonstration images [52], as exemplified in Figure 2.6b).

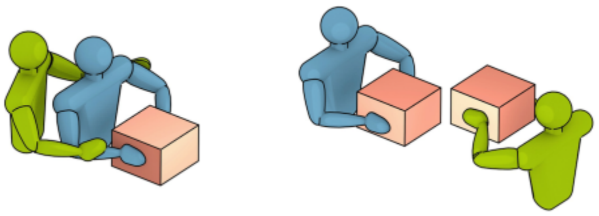

(a) Kinesthetic teaching (b) Observational learning

Figure 2.6: Learning from human demonstration methods [20]

In literature, numerous instances can be found of path planning methods for deburring operations, employing the previously mentioned approaches. In [44], a vision-assisted robot offline programming system has been built up: 2D vision cameras equipped with techniques like the Hough transformation are utilized to identify prominent features on the workpiece. These identified features then serve as the foundation for constructing the deburring path. As exemplified in [44] this system recognizes straight lines on the workpiece. An Off-Line Programming (OLP) system subsequently translates these recognized features into a deburring path, as visually exemplified in Figure 2.7. Similarly, in [60], a camera detects the workpiece's edges useful for path generation. It should be noted that both methods rely on straight-line recognition only and have been tested on geometrically simple shapes.

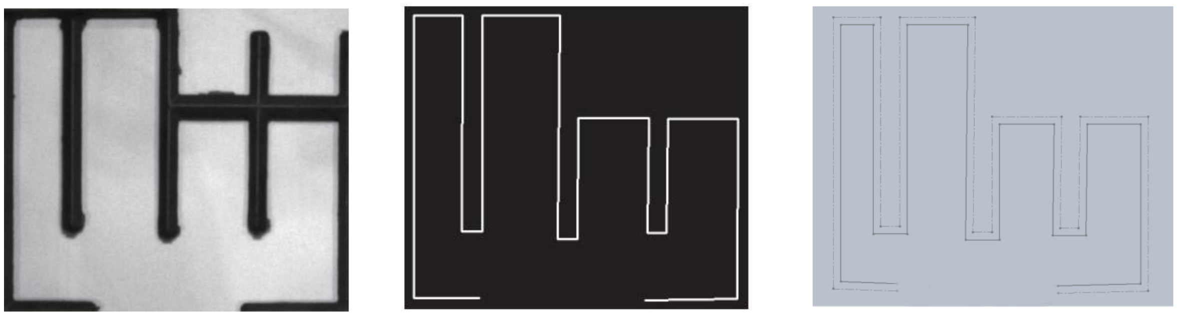

(a) 2D image of the workpiece (b) Hough straight lines (c) Result: the inside line is the path of the tool

Figure 2.7: Burrs identification pipeline with straight-line recognition [44]

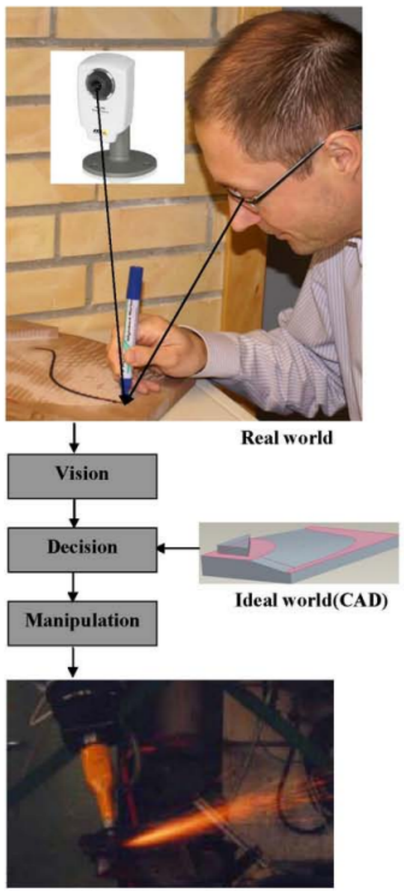


Figure 2.8: Visual representation of the pipeline of *"Vision-Based Robot Programming"* [70]

Innovative solutions integrate vision, CAD data, and human demonstration. The approach proposed in [70] leverages a worker's expertise to identify the burrs. The pipeline is represented in Figure 2.8: the deburring path is manually drawn with a pen on the workpiece and then digitalized through image processing techniques (i.e. Canny edge algorithm [22]) to find the coordinates in plane x-y. The z coordinate is obtained by intersecting the x-y path with the CAD model. Similarly, the method presented in [84] involves manual path drawing on the workpiece, followed by a robot, equipped with an eye-in-hand camera, tracking and recording the 2D path. The z coordinate is registered by the robot end effector through a tool, maintained continuously in contact with the surface.

In the study presented in [71], CAD/CAM and human demonstration approaches are joint: this process combines CAD/CAM for precise contour extraction from the CAD model and direct teaching with impedance control to manually select key contact points on the workpiece. Utilizing an iterative closest point (ICP) algorithm [85] to align the

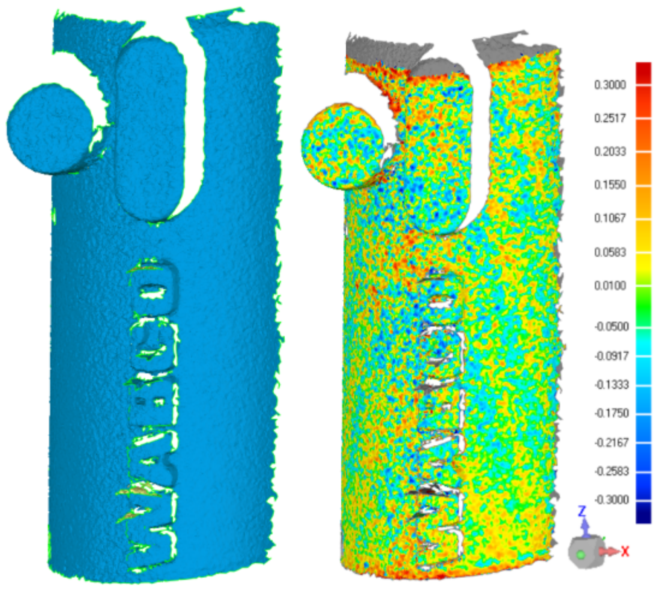


Figure 2.9: Surface measurement of the workpiece and deviation map between the registered target and source surface [43]

CAD model's tool path with the teaching points, a transformation matrix is obtained to account for positional and orientational errors in the workpiece and the tool path is corrected accordingly.

In [43] the tool path is taught off-line based on a reference workpiece, manually deburred. Subsequently, a 3D scan employing a laser-triangulation sensor is conducted online on the workpiece: relative rotation and translation with respect to the reference (Figure 2.9) are derived using the ICP algorithm and the tool path is adjusted.

Similarly, [78] proposes an offline teaching on a reference workpiece and adaptation through a single-point laser displacement sensor (SP-LS) mounted on the end effector and moved in a set of control points to obtain the 3D measurement.

An improvement is seen in [56], wherein a 3D scan of the workpiece is executed through a 3D camera and then alignment with the CAD model is achieved using convolutional neural networks, enhancing the robustness of burrs detection [50]. The resultant point cloud is further transformed into a triangular mesh using surface reconstruction algorithms. A visual representation of the pipeline is reported in Figure 2.10. The burrs are subsequently identified through the detection of intersections between the mesh and the plane situated between the two molds involved in the casting process.

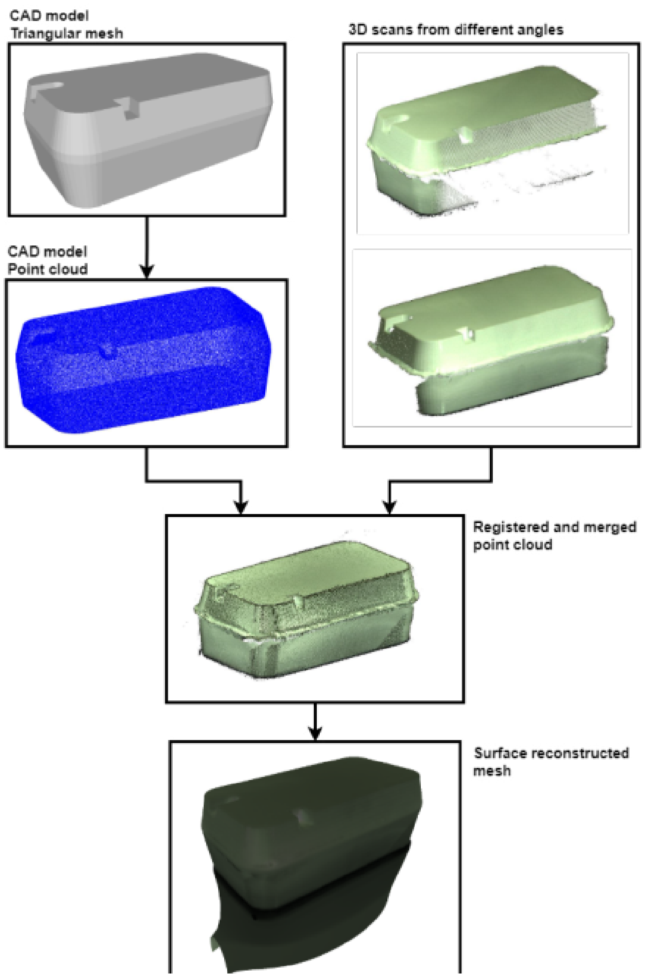


Figure 2.10: 3D model and point cloud alignment for burrs identification [56]

3 | Sole segmentation

The initial step in automating sole deburring using robotic systems involves precisely locating the section of the workpiece that needs material removal. In the review of existing literature, various methods for identifying burrs have been highlighted, each applying distinct technologies and strategies tailored to specific applications.

In this research, a two-stage approach has been adopted. Initially, deep learning (DL)-based object detection and segmentation techniques have been employed to identify the location of the sole and its geometrical features. Following this, image processing methods have been applied to identify the burrs region and define the deburring tool path.

This chapter is dedicated to the development of the sole detection and segmentation component within the chosen pipeline.

3.1. Object detection and segmentation

The proposed method for sole deburring relies on image acquisition and processing. Specifically, a *Intel RealSense D453i RGB camera* is mounted on the robot's end effector. When the workpiece with the sole requiring deburring is positioned beneath the camera, a 2D image is acquired. From this acquired image, a neural network is deployed to perform two tasks. Firstly, it identifies the precise location of the sole by determining its bounding box and class label, essentially creating a rectangular boundary around the sole's position in the image. Secondly, it carries out the segmentation process, delineating all the pixels within the image that pertain to the sole.

For these tasks, the chosen neural network architecture is Detectron-2 [80], which is well-suited to handle the precise object detection and segmentation required for effective deburring.

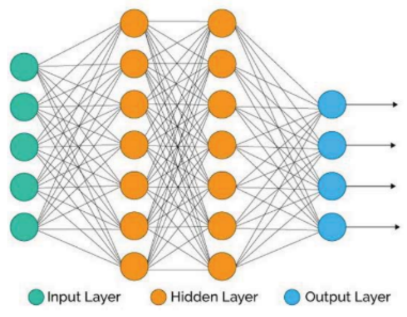
3.1.1. Detectron-2

Detectron2 [80] represents an advanced deep learning framework developed by the Facebook AI Research (FAIR) group. It is built upon Mask R-CNN (MRCNN) [35] and serves as a robust open-source platform, particularly tailored for computer vision research. Its core competencies are tasks such as precise object detection, semantic segmentation, and the identification of key-points within images.

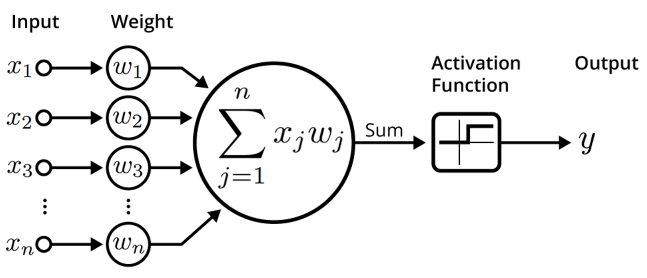
What distinguishes Detectron2 is its versatility and scalability, enabling the training of diverse models capable of achieving state-of-the-art results in object detection. This feature, coupled with the computational performance of its models, renders it the optimal choice for the shoe sole segmentation objective.

CNNs, RCNNs, fast-RCNNs, faster-RCNNS, Mask-RCNNs

An **Artificial Neural Network (ANN)** [28], also referred to as a Neural Network (NN), is a computational model inspired by the biological nervous system, particularly the brain, for data processing tasks. It consists of interconnected neurons organized into layers, as illustrated in Figure 3.1a. The input layer receives data, the output layer generates results, and any intermediate layers are referred to as hidden layers. Neurons within the network become active when the sum of their inputs surpasses a certain threshold, and their output is computed by applying an activation function to this weighted sum. A visual representation of how they work is depicted in Figure 3.1b. A neural network "learns" a task by determining the appropriate weights that establish the correct relationship between input and desired output.



(a) ANN architecture [28].



(b) Artificial neuron illustration.

Figure 3.1

Convolutional Neural Networks (CNNs) [57] are a specialized category of neural networks that share a structural foundation with ANNs but are tailored for processing and analyzing visual data, particularly 2D matrices like images. The input layer holds the pixel values of the input image. The distinctive architecture of CNNs revolves around convolutional layers, which are adept at learning features from input data.

Region-Based CNNs (RCNNs) [32] represent an evolutionary step in the realm of deep learning for object detection purposes. Girshick et al. introduced the novel concept of region proposals through *selective search* [76]. R-CNNs generate a set of potential object regions, called *regions of interest*, and, subsequently, a pre-trained CNN is applied individually to each proposal to extract features. Support Vector Machines (SVM) [27] are then exploited to find the bounding box and the class of every output feature. Its architecture is displayed in Figure 3.2. While R-CNNs significantly improve object detection accuracy, they are computationally expensive due to the need for external region proposal methods. Each proposal demands individual CNN computations, and there is no sharing of processing. Given that these regions frequently overlap, a significant amount of redundancy is carried out.

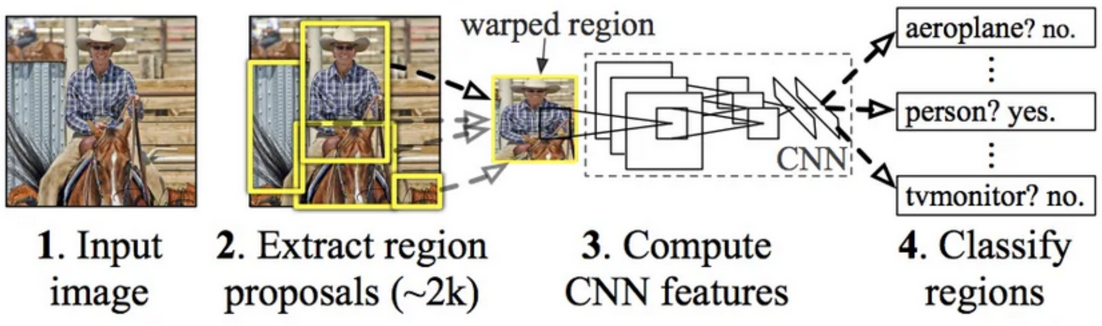


Figure 3.2: RCNN architecture [32]

Girshick extended the prior work on R-CNNs by introducing **Fast R-CNNs** [31]. In contrast to the approach of applying a distinct CNN to each proposal, Fast R-CNNs adopted a more efficient strategy. They employed a single CNN to extract features from the entire image, generating a convolutional feature map. Subsequently, a Region of Interest (RoI) pooling layer identifies region proposals from this feature map. From the RoI feature vector, the class and the bounding box of the proposed region are obtained. Its architectural configuration is depicted in Figure 3.3. Fast RCNN is faster than RCNN due to the fact that the convolution operation is performed just once on the entire image, rather than repeatedly on each region proposal. As a result, the principal bottleneck in Fast R-CNN shifts to the task of identifying the region's proposals (i.e. selective search algorithm).

Shaoqing Ren et al. [62] came up with an object detection algorithm that eliminates

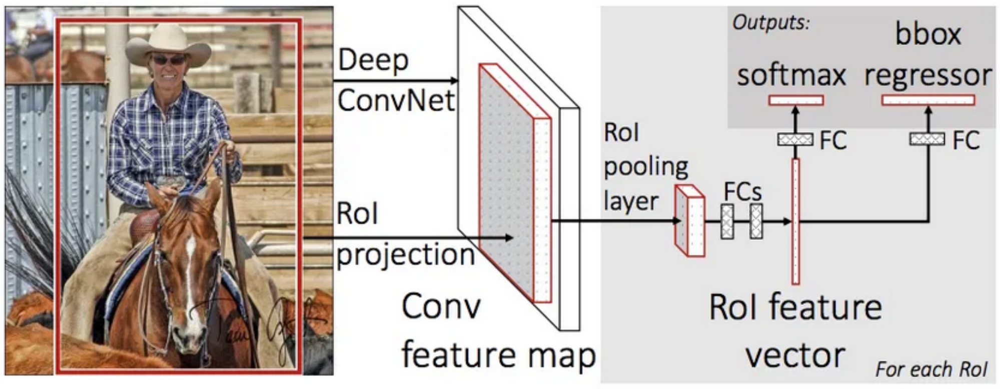


Figure 3.3: FastRCNN architecture [31]

the selective search algorithm, developing **Faster-RCNNs**. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which outputs a convolutional feature map. Instead of using a selective search algorithm on the feature map to identify the region proposals, a *region proposal network* is used to predict the region proposals. This network is trained together with the rest of the model. The predicted region proposals go through a RoI pooling layer used to classify the image within the proposed region and predict the bounding boxes. Its layout is presented in Figure 3.4.

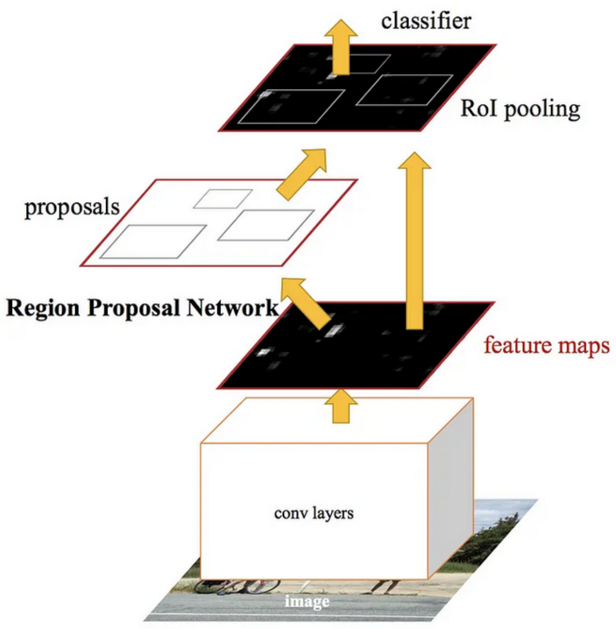


Figure 3.4: FasterRCNN architecture [62]

Mask R-CNNs [36] extend the capabilities of Faster R-CNNs by introducing a mask prediction branch, alongside the existing bounding box prediction branch, for image segmentation. Its structure is presented in Figure 3.5. This addition enables Mask R-CNNs not only to predict the class and bounding box for each region of interest but also the pixel-level position of the object through an additional fully convolutional network.

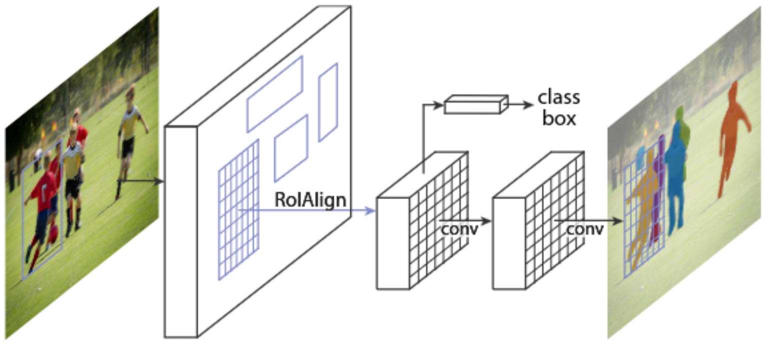


Figure 3.5: MaskRCNN architecture [36]

Detectron-2 architecture

Detectron-2 [80] is a state-of-the-art library for object detection and instance segmentation, based on MaskRCNN. The architecture, as schematically represented in Figure 3.6 consists of three main blocks: the Backbone Network, the Region Proposal Network (RPN) and the ROI heads.

1. **Backbone Network** allows the extraction of multi-scale feature maps from the input image
2. **Region Proposal Network (RPN)** identifies object regions from the multi-scale features
3. **Box and Mask Head** predicts the coordinates of the bounding boxes and object segmentations and the associated classes

The backbone stage integrates a variety of sub-networks, including ResNet, ResNeXt, Feature Pyramid Networks (FPN). Each of these possesses distinct attributes, making them well-suited for specific tasks. *ResNet* [34] deploys residual blocks with skip connections effectively mitigating the challenges associated with training deep neural networks. This innovation enables them to learn intricate features with a high degree of accuracy. *ResNeXt* [82], an extension of the ResNet framework, further enhances feature extraction capabilities by introducing the concept of cardinality, which involves incorporating multiple parallel internal paths or branches within a single block. This approach leads to improved network accuracy without the need to increase the network's depth. *FPN* [47]

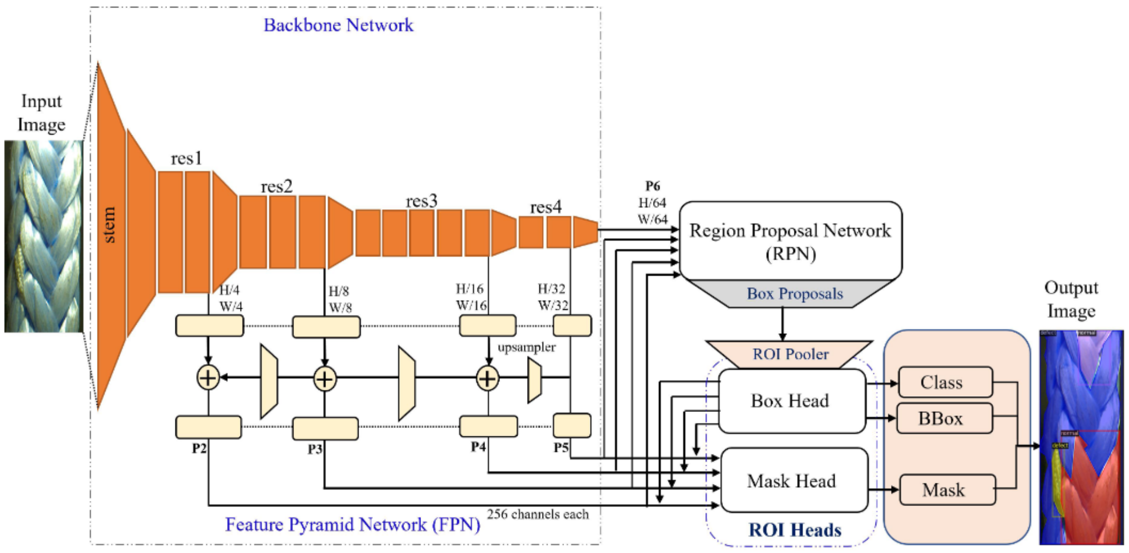


Figure 3.6: Detectron-2 architecture [61]

is designed to address the challenges of multi-scale object detection and feature learning. It achieves this by constructing a feature pyramid, making it particularly effective for tasks where objects may appear in various sizes.

3.2. Model training for sole segmentation

DL-based object detection and segmentation is a complex task that requires proper selection of the model to be used and adequate training data. In the next sections the details of the procedure followed to obtain the sole's detection and segmentation on Detectron-2 are presented.

3.2.1. Dataset acquisition

Several annotated datasets for object detection and segmentation are available in the literature [65] but none of them contains the information needed for this study's purpose. Therefore, a novel dataset has been created.

A set of 2D images of shoe soles, with and without burrs, has been captured using the *Intel RealSense D453i* RGB sensor. To enhance the model's capacity for generalization, variability has been introduced in the dataset. This includes capturing samples with variation in sole orientation and incorporating two distinct background settings. In addition to these variations, the dataset accounts for the practical scenario in which the sole is attached to a base using rounded clips during the deburring process. Consequently, images of the sole in this clamped configuration have been integrated into the dataset. In total, a dataset of 71 images has been obtained, with a resolution of 640x480. For each image in the dataset, manual annotations have been applied to include both bounding box and segmentation mask labels:

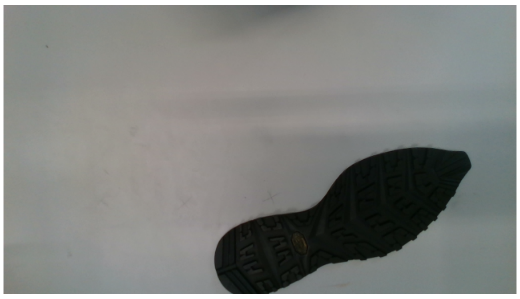
- **Bounding Box Label:** this label comprises five numerical values. The first value designates the class, which in this case is singular (1 as "sole"). The second and third values represent the coordinates of the box's center, while the fourth and fifth values denote the width and height of the bounding box.
- **Segmentation Label:** this label contains the pixel coordinates that collectively constitute the mask, delineating the region of the sole in the image.

Samples from the utilized dataset are visually depicted in Figure 3.7, while their corresponding annotations are presented in Figure 3.8.

The dataset has been partitioned into two subsets: training and validation. The training set comprises 56 images, accounting for approximately 80% of the dataset, and the remaining 15 images are allocated for testing.

(a) Sole on deburring base.

(b) Sole with burrs.


(c) Nominal sole.

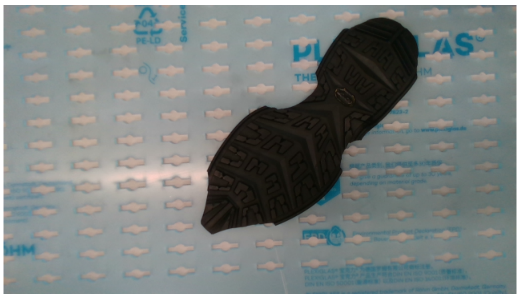

(d) Background change.

Figure 3.7: Dataset RGB for Detectron-2.

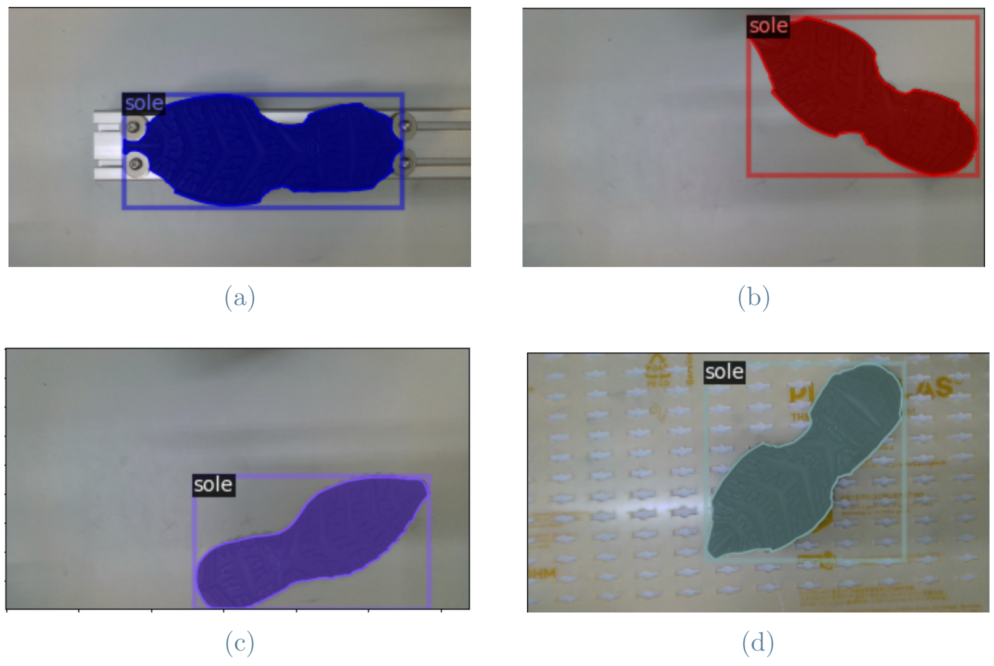


Figure 3.8: Dataset bounding box and segmentation annotations.

3.2.2. Training parameters and tested models

With the aim of identifying the best-performing model, an extensive evaluation has been conducted using three distinct models sourced from the Detectron-2 model zoo (a repository of state-of-the-art models for computer vision research [80]). The Residual Network (ResNet [34]) combined with Feature Pyramid Network (FPN [47]) backbone architecture is chosen for its optimal balance between speed and accuracy. These models are specifically:

- **R50-FPN**, built upon the ResNet-50 architecture
- **R101-FPN**, founded on the ResNet-101 architecture
- **X101-FPN**, constructed on the more recent ResNeXt [82] network

All the models have been trained in Google Colab for 5000 iterations. During training the Stochastic Gradient Descent (SGD) optimizer has been employed. The learning rate for the initial 1000 iterations has been set to 0.0005, followed by a reduction to 0.00025 for the subsequent iterations.

By testing and comparing the performance of these three distinct models, the aim is to determine which one exhibits the most favorable results in terms of sole segmentation.

3.2.3. Evaluation metric: Average Precision (AP)

To assess and rank the performance of various models accurately and objectively, an evaluation metric is essential. In this context, the standard COCO (Common Objects in Context) evaluation metric, available at [8], has been employed. Specifically, the **Average Precision (AP)** metric has been chosen, as it is a widely accepted and utilized metric for evaluating object detection tasks.

The calculation of the Average Precision (AP) metric involves several other key metrics, including IoU, confusion matrix (TP, FP, FN), precision and recall.

The **Intersection over Union (IoU)** is a numerical value that falls within the range of 0 to 1. It quantifies the degree of overlap between the ground truth and the prediction. When the two bounding boxes or segmentations perfectly align, the IoU is 1, indicating perfect overlap. Conversely, if the two bounding boxes or segmentations do not intersect at all, the IoU is 0, meaning no overlap. Mathematically, the IoU is computed as the ratio of the area of intersection between the two bounding boxes/segmentations to the area of their union, as depicted in Figure 3.9. This calculation provides an objective measure of how accurately the predicted bounding box/segmentation aligns with the true object location.

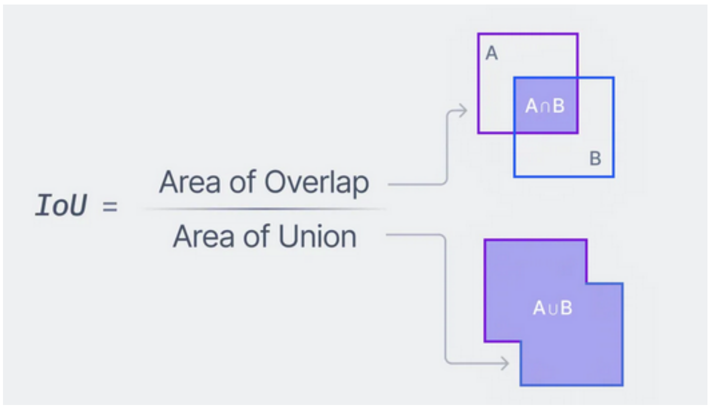


Figure 3.9: IoU definition

Given a threshold, a prediction is considered *correct* if the IoU between the ground truth and the prediction is higher than the threshold value. With this definition, more metrics can be defined as:

1. **True Positive (TP)** if the model correctly (true) predicted that a bounding box/segmentation exists in a certain position (positive class)
2. **False Positive (FP)** if the model incorrectly (false) predicted that a bounding box/segmentation exists in a certain position (positive class)

3. **False Negative (FN)** if the model incorrectly (false) didn't predict a bounding box/segmentation in a certain position (negative class)

The final parameters required for computing the Average Precision consist of Precision and Recall. Given the definitions provided by the equations 3.1 and 3.2, **Precision** signifies the proportion of correct predictions relative to the total number of predictions, whereas **Recall** expresses the proportion of correct predictions relative to the total number of ground truths, which is equivalent to the sum of true positives and false negatives.

$$Precision = \frac{Correct\ Predicitons}{Total\ Predictions} = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = \frac{Correct\ Predictions}{Total\ Ground\ Truth} = \frac{TP}{TP + FN} \tag{3.2}$$

A model is considered highly effective when it demonstrates both elevated precision and recall, meaning it accurately identifies objects while capturing all instances within the dataset. It's important to emphasize that precision and recall metrics are intricately linked to the values of TP, FP and FN, and these metrics vary with the selected Intersection over Union (IoU) threshold.

Ultimately, the **Average Precision (AP)** is a numerical value ranging from 0 to 1, and it's often expressed as a percentage, spanning from 0% to 100%. AP is defined as the average precision computed across all recall values at a specific Intersection over Union (IoU) threshold.

As reported in [65], to compute the Average Precision (AP) for a specific class within a dataset, it is necessary to first arrange the data entries in descending order based on their confidence scores. These confidence scores can be either the classifier output probabilities or Intersection over Union (IoU) scores from the object detection algorithm. The AP is subsequently determined with the formula:

$$AP = \sum_{k=1}^{n} P(k)\Delta r(k) \tag{3.3}$$

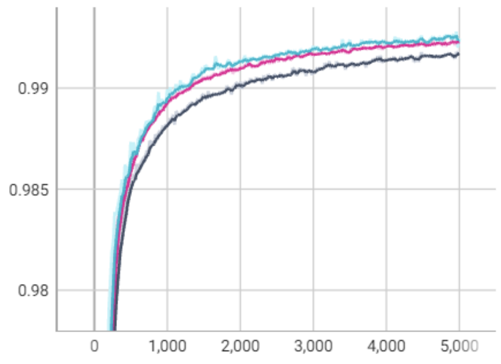
Where n is the number of ground truths, *P(k)* is the precision of the *k-th* prediction, and $\Delta r(k)$ is the variation of recall between steps k and *k+1*.

3.2.4. Results

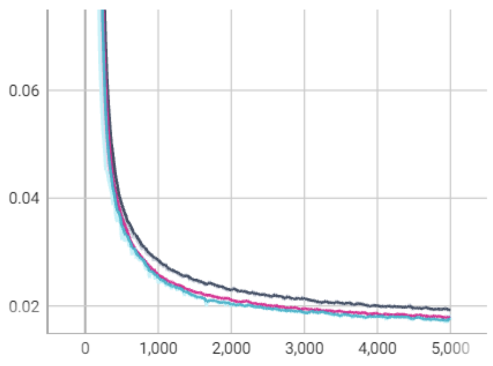
The most critical training curves for this study are depicted in Figure 3.10. These include the accuracy and loss of the masks, as well as the total model loss, encompassing classification, bounding box, and mask losses. It is evident from the curves that all the trained models have reached a plateau in both loss and accuracy, signifying the completion of the training process.

R-50-FPN	●
X101-FPN	●
R101-FPN	●

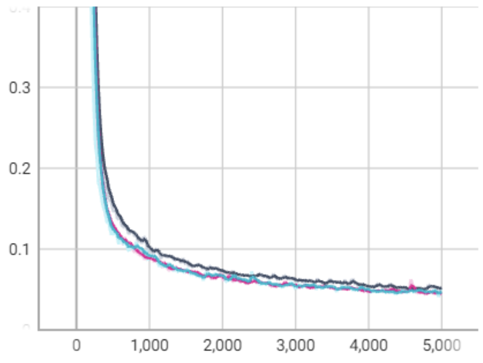
(a) Legend



(b) Mask accuracy



(c) Mask loss



(d) Total loss

Figure 3.10: Detectron-2 training curves

The Average Precision (AP) for the three models under consideration has been computed, offering a comprehensive assessment of their performance. It's worth noting that while IoU thresholds typically range from 0.5 to 0.9 in common scenarios, this study employs considerably higher thresholds of 0.95, 0.975, and 0.99. The reason behind these elevated thresholds lies in the specific characteristics of this task and the operating conditions. In this context, object detection is relatively straightforward due to the presence of only one sole in each provided image, with the black color of the sole providing a high-contrast distinction against the bright background. Consequently, false predictions, whether positive or negative, with lower thresholds are nearly non-existent, resulting in an AP of 100%. This behavior persists up to a threshold of 95%. However, to meet the precise segmentation requirements of this study, the threshold has been pushed even further to 0.99.

The resulting AP values are shown in Table 3.1.

	R50-FPN	**R101-FPN**	**X101-FPN**
$IoU = 0.95$	100%	100%	100%
$IoU = 0.975$	91.716%	100%	97.805%
$IoU = 0.99$	73.440%	71.617%	**77.494%**

Table 3.1: AP values for trained models varying the IoU threshold.

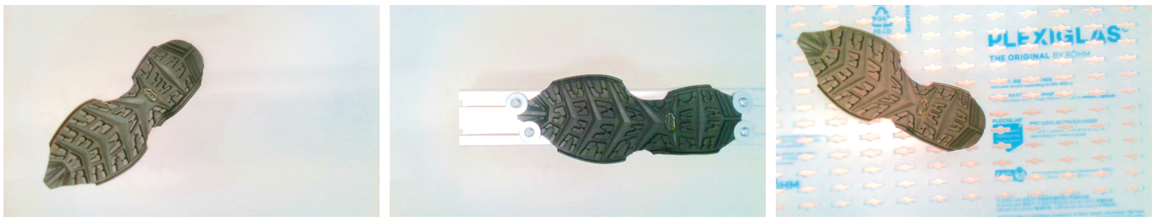
It can be concluded that while all the models exhibit excellent accuracy, the X101-FPN model has been selected for the application due to its exceptional performance under these stringent segmentation demands.

3.3. Deep Learning vs classic Computer Vision Image Processing Techniques for Sole Segmentation

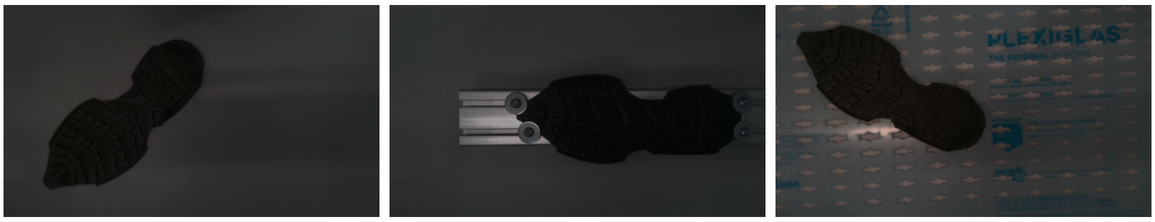
In the realm of image segmentation, traditional methods have been a well-established presence, commonly classified into three categories [30]: characteristic features thresholding, edge detection, and region extraction. Feature thresholding-based methods have been particularly prevalent. The objective of this section is to elucidate the motivation behind the choice of a deep learning approach in this study, as opposed to the more conventional thresholding methods, in the analyzed scenario of sole segmentation.

To support this choice quantitatively, a set of 15 images sourced from the Detectron-2 dataset has been utilized for testing. Images both with and without the deburring base and featuring two distinct backgrounds have been taken into consideration. The testing process includes evaluation under three distinct lighting conditions: standard, brightened, and dark. These datasets have been obtained by changing the exposure of the standard condition in post-processing. Some samples are reported in Figure 3.11

(a) Standard Lighting



(b) Brighter Lighting



(c) Darker Lighting

Figure 3.11: Datasets for segmentation methods performance comparison

The Intersection over Union (IoU) metric, described in Chapter 3.2.3, has been employed to quantify the segmentation methods' performance.

Image thresholding is a technique that involves partitioning an image into distinct regions based on intensity levels. Through the selection of a threshold value, pixels with intensity levels exceeding this threshold are designated as white (intensity = 1), while those below the threshold are classified as black (intensity = 0). This binary representation simplifies the process of detecting and isolating objects of interest within the image. Mathematically, thresholding can be defined as follows:

$$r_{i,j} = \begin{cases} 1 & p_{i,j} \geq \mathrm{T}. \\ 0 & p_{i,j} < \mathrm{T}. \end{cases} \tag{3.4}$$

where $r_{i,j}$ is the resulting intensity of the pixel at coordinates *(i,j)*, $p_{i,j}$ is the intensity of the pixel from the input image and T is the value of the threshold.

Various techniques exist for image thresholding. For the purpose of this study, three of them have been tested and compared to Detectron-2's results: simple global thresholding, Otsu's global thresholding and adaptive (local) thresholding.

3.3.1. Simple Global thresholding

Initially, a **simple global thresholding** approach has been evaluated. A single threshold value has been determined through a trial-and-error procedure to maximize the arithmetic mean of the IoU values across the dataset captured under standard environmental lighting conditions. Subsequently, Detectron-2 and the binarization method using this threshold have been applied to acquire masks for the dataset under standard lighting conditions, and the resulting mean IoU has been calculated and reported in Table 3.2.

Mean IoU	**Simple global thresholding**	**Detectron-2**
Standard lighting	**97.86%**	96.69%

Table 3.2: Performance comparison simple global thresholding and Detectron-2, standard lighting

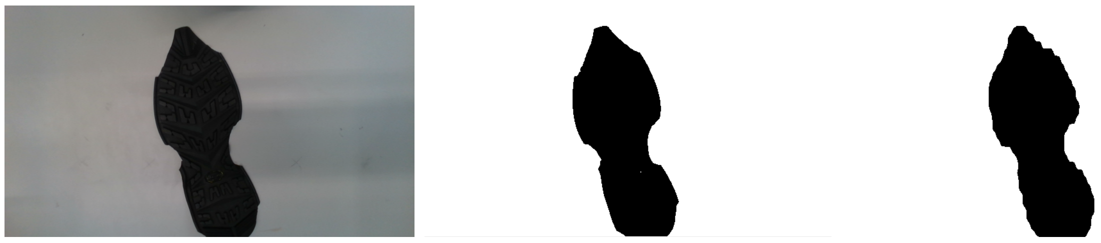
The results indicate that, in this scenario, despite both methods achieving a remarkably high level of performance, the thresholding approach actually surpasses the accuracy of segmentation achieved by Detectron-2.

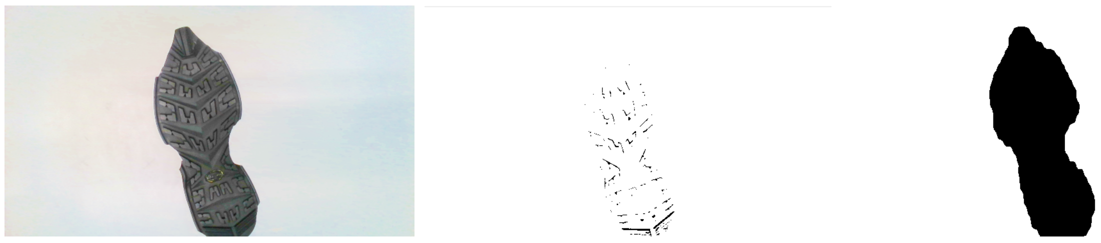
Subsequently, both segmentation methods have been assessed on datasets subjected to modified lighting conditions, specifically, a brighter and a darker lighting scenario.

Mean IoU	**Simple global thresholding**	**Detectron-2**
Brighter lighting	2.90%	**96.10%**
Darker lighting	30.35%	**96.65%**

Table 3.3: Performance comparison simple global thresholding and Detectron-2, varying lighting conditions

The results reported in Table 3.3 reveal that the accuracy of Detectron-2 remains consistent across varying environmental conditions, while the performance of the thresholding method experiences a significant decline. This disparity can be attributed to the fixed threshold value, which doesn't adapt to changes in pixel intensity. In brighter lighting conditions, the pixel intensities tend to increase. Consequently, some pixels originally belonging to the sole may now surpass the threshold and be incorrectly classified as part of the background. Conversely, in darker lighting conditions, the opposite can occur where pixels that are part of the background might be misclassified as part of the sole. This variability in lighting conditions can result in inaccurate segmentation outcomes. An example is reported in Figure 3.12.


(a) Standard Lighting: original image, simple global thresholding and Detectron-2 segmentation


(b) Brighter Lighting: original image, simple global thresholding and Detectron-2 segmentation

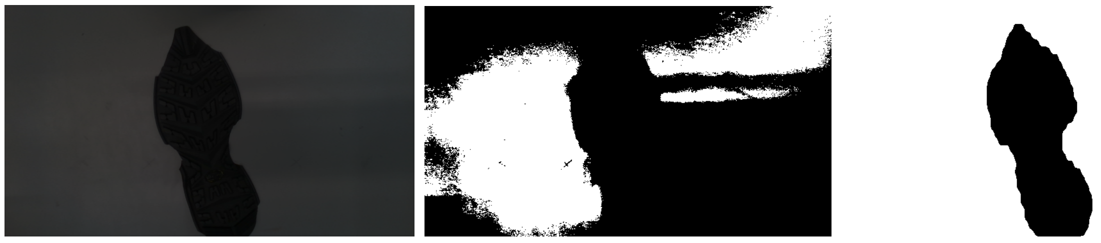

(c) Darker Lighting: original image, simple global thresholding and Detectron-2 segmentation

Figure 3.12: Simple thresholding and Detectron-2 segmentation in different lighting coniditions

Therefore, it's evident that simple global thresholding lacks robustness against changing lighting conditions and cannot be effectively employed in real-case industrial settings. Detectron-2, instead, shows stable performance in segmentation under all lighting conditions.

An effective approach to address this challenge is to adjust the threshold value dynamically for each new image acquired. Manual adjustment is clearly impractical for real-world applications, as the system could not autonomously adapt to varying lighting conditions. Therefore, the focus shifts to methods that can dynamically select the appropriate threshold when processing an image. In this context, two specific techniques, the Otsu method and adaptive thresholding, have been examined.

3.3.2. Otsu's method global thresholding

Otsu's method [58] is designed to automatically determine the optimal threshold value. It operates under the assumption that an image contains two classes of pixels: the background and the foreground. Consequently, the grayscale histogram of the image is expected to exhibit a bi-modal distribution (Figure 3.13), consisting of two distinct peaks. One peak represents the uniform background color, while the other corresponds to the color of the object to be detected. The optimal threshold value is computed by minimizing the variance between these two peaks.

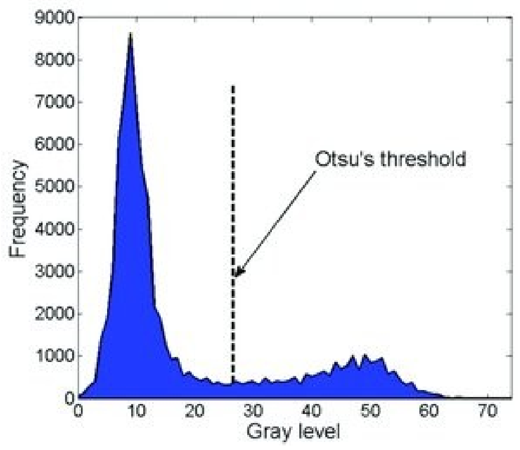


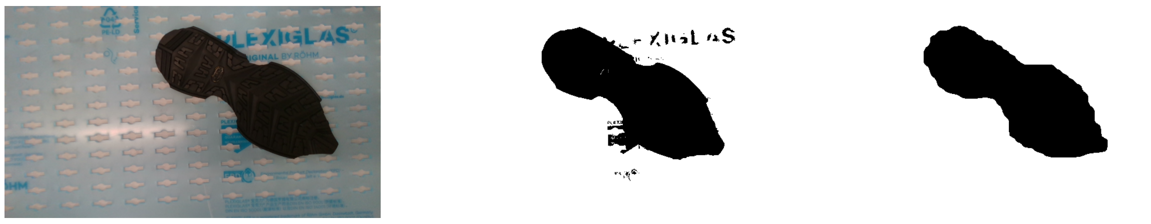
Figure 3.13: Bi-modal grayscale histogram and Otsu's threshold

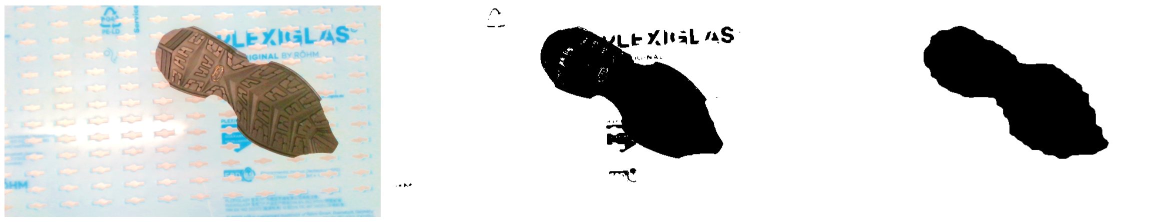
The Otsu's optimal threshold has been found and both the thresholding and Detectron-2 have been tested on the datasets.

Mean IoU	**Otsu's global thresholding**	**Detectron-2**
Standard lighting	94.82%	**96.69%**
Brighter lighting	94.94%	**96.10%**
Darker lighting	92.99%	**96.65%**

Table 3.4: Performance comparison Otsu global thresholding and Detectron-2, varying lighting conditions

The outcomes across the datasets, reported in Table 3.4 indicate that thresholding with the Otsu's threshold has improved its robustness in response to changes in lighting conditions. However, it now faces a different challenge: the assumption of a bi-modal distribution may not hold true. In images that feature objects on the background, like the deburring base or textured surfaces, certain portions of the background are erroneously categorized as part of the sole's mask. In contrast, Detectron-2 remains robust to such background alterations. The result is visible in Figure 3.14


(a) Standard Lighting: original image, Otsu's global thresholding and Detectron-2 segmentation


(b) Brighter Lighting: original image, Otsu's global thresholding and Detectron-2 segmentation

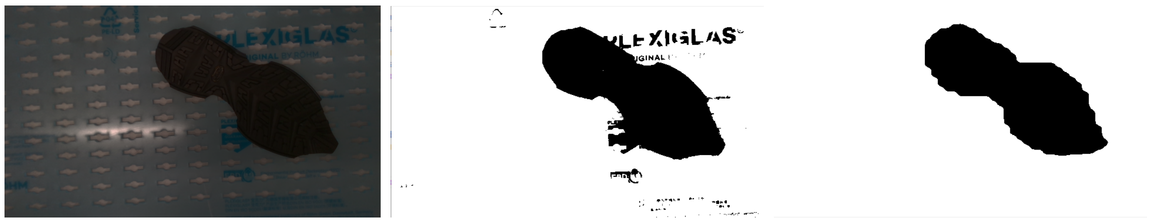

(c) Darker Lighting: original image, Otsu's global thresholding and Detectron-2 segmentation

Figure 3.14: Otsu thresholding and Detectron-2 segmentation in different lighting conidi-tions

3.3.3. Adaptive thresholding

The final enhancement method examined is **adaptive thresholding**. Unlike the two previous techniques, where a single threshold applies to all pixels in the input image, adaptive thresholding divides the image into multiple smaller regions and determines a local optimal threshold value based on the characteristics of each region.

The threshold in adaptive thresholding is typically computed using either the arithmetic mean or the Gaussian mean of pixel intensities within each region. For this study's testing, the arithmetic mean has been employed, where each pixel provides an equal contribution to the threshold value. Additionally, a constant value (which requires tuning) is subtracted from the mean to fine-tune the thresholding process.

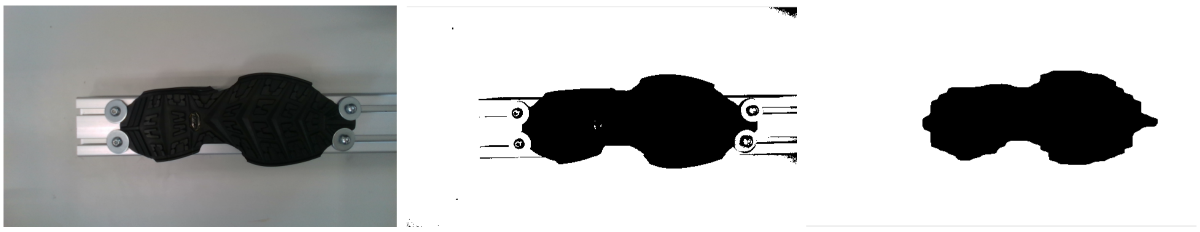
Therefore, the local threshold value T becomes:

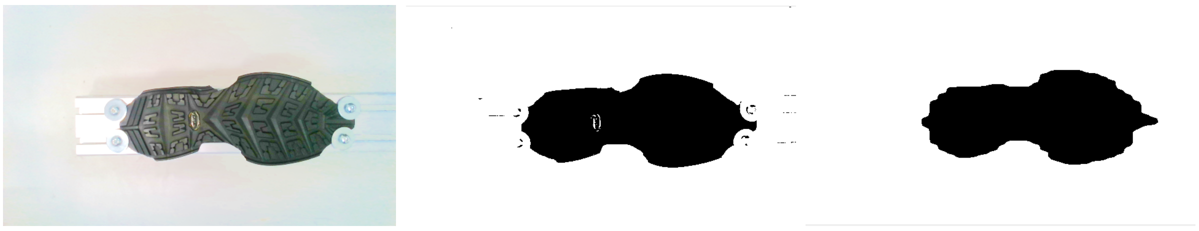
$$T = mean\{I_L\} - C \tag{3.5}$$

where I_L is the local sub-region L of the image and C is a constant.

The adaptive thresholding method has then been tested on the datasets.

The results indicate that while adaptive thresholding offers improved robustness to local lighting changes compared to simple thresholding or Otsu's method, it still cannot accurately separate the background from the sole segmentation, as reported in the example in Figure 3.15.


(a) Standard Lighting: original image, adaptive thresholding and Detectron-2 segmentation


(b) Brighter Lighting: original image, adaptive thresholding and Detectron-2 segmentation

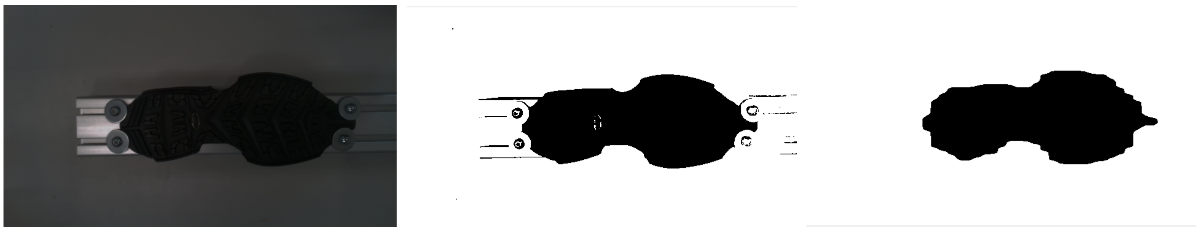

(c) Darker Lighting: original image, adaptive thresholding and Detectron-2 segmentation

Figure 3.15: Adaptive thresholding and Detectron-2 segmentation in different lighting coniditions

In summary, Detectron-2 has proven to be the optimal approach for this specific application, thanks to its robustness against variations in lighting conditions and background changes. Consequently, it has been selected for the sole segmentation purpose in this study.

An additional consideration is that Detectron-2, being a neural network, possesses classification capabilities. While the current model has been trained to recognize a general class ("sole"), Detectron-2 has the potential to be trained to distinguish various classes such as right and left shoes or different types of soles. Achieving similar results with traditional Computer Vision techniques would require additional and complex steps, especially in identifying and translating into code distinctive features for each class. Hence, to facilitate future developments, Detectron-2 remains the optimal choice for this study.

3.4. Reconstruction of occluded sole profiles

The segmentation achieved using Detectron-2 [80], described in the preceding chapter, provides exceptionally precise segmentation identification capabilities when presented with a 2D image of the sole (as reported by the AP values in Table 3.1). In principle, this level of segmentation precision would enable the burr identification process. However, an additional step has been incorporated into the deburring pipeline developed in this study. This addition is necessitated by the physical working conditions that the robotic system is bound to encounter, both in the experimental setup for this study and in real-world industrial scenarios.

During the deburring process, the sole is subjected to a certain level of force. Consequently, it is imperative to secure the sole in place above a fixed base and ensure it remains immobilized. To maintain the aesthetics of the sole, any solution involving perforation has been discarded, as this would compromise its final appearance. It's important to note that deburring is the final operation performed on the sole before it is glued to the rest of the shoe. An appropriate approach is to grasp the sole. However, any gripping system inevitably obstructs a portion of the sole's surface. These occlusions alter the segmentation as identified by the object detection algorithm, leading to inaccuracies in burr identification.

In the experiments conducted in this study, the sole was held in place using four circular rings positioned at its extremities, tightened with screws, as illustrated in Figure 3.16.

To achieve the complete and precise identification of burrs across the entire sole, the approach adopted involves the reconstruction of the complete profile. For this purpose, a

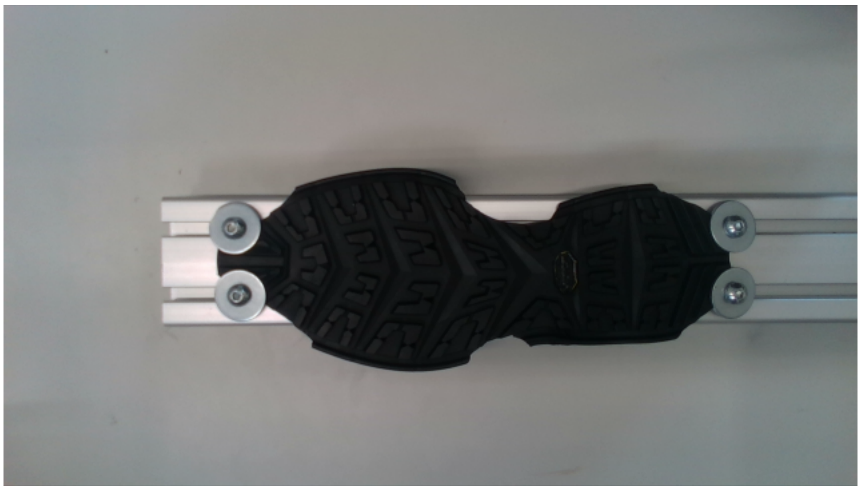


Figure 3.16: Sole clamping during experiments

deep learning approach has been selected. Specifically, a Conditional Generative Adversarial Network (cGAN) [38], namely Pix2Pix, has been employed for the task of image-to-image translation.

3.4.1. Image to image translation

Image-to-image translation, in the context of computer vision, refers to the process of transforming an input image from one representation or style to another. This transformation is learned from training data that includes pairs of input and output images. The objective is to establish a mapping or model that can convert an input image into an output image. This technique can be applied to various tasks, as exemplified in Figure 3.17, such as changing a summer landscape photo into a winter scene or converting a horse image into a realistic zebra photograph.

The predominant deep learning approach for this task involves Generative Adversarial Networks (GANs) [33].

Generative Adversarial Networks (GANs)

A Generative Adversarial Network (GAN) [33] is a deep learning framework designed for image-to-image translation tasks. It consists of two primary components: a generator (G) and a discriminator (D).

The **generator** model **(G)** has the capability to produce new and realistic synthetic samples that closely resemble data originating from an existing distribution of samples. In contrast, the **discriminator** model **(D)** is trained to distinguish between the fake

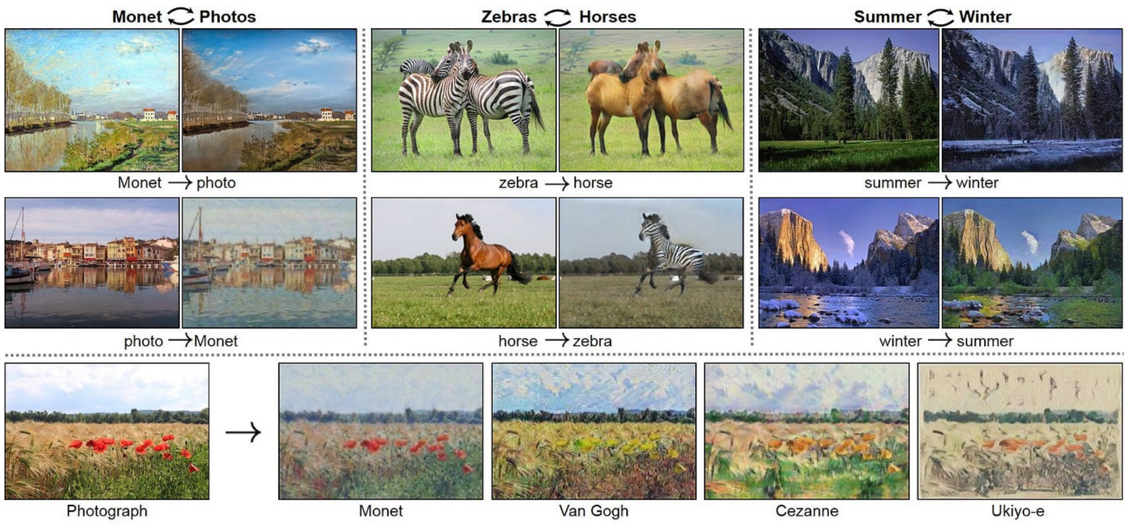


Figure 3.17: Examples of image-to-image translation

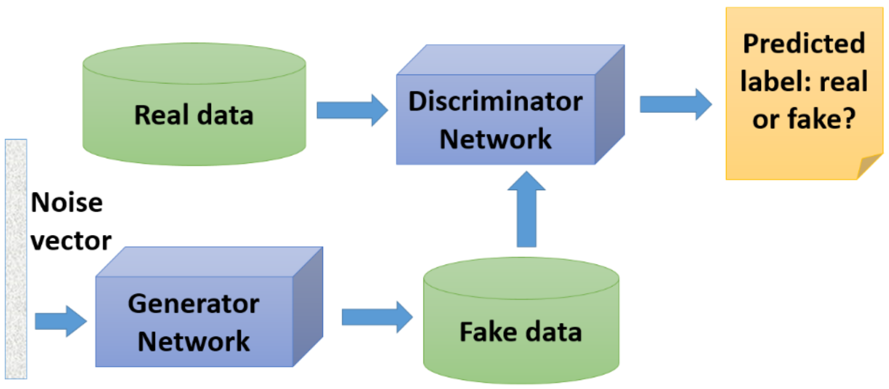


Figure 3.18: GANs architecture

data generated by G and real data. Therefore, the training objective for the GAN can be framed as a two-player min-max game, where the generator (G) aims to minimize the objective function, while the discriminator (D) strives to maximize it. This adversarial setup drives the generator to create increasingly convincing samples and the discriminator to become more expert at differentiating real from fake data.

The basic architecture of GANs is shown in Figure 3.18.

3.4.2. Pix2Pix Architecture

The proposed approach is built upon the **Pix2Pix** [38] architecture, which is a state-of-the-art framework for image-to-image translation developed by Isola et al. in 2016. Pix2Pix is based on conditional Generative Adversarial Networks (cGANs) [49].

Conditional GANs [49] represent a specific type of GAN architecture designed for generating images with specific attributes or characteristics, *conditional generation*. Unlike other GAN models, which produce random images within a given domain, cGANs are designed to generate targeted images of a particular type based on the provided input. Pix2Pix operates as a conditional GAN, where a target image is created from an input image. The generator model takes an input image and produces a translated version. The discriminator model, on the other hand, receives an input image along with a translated one and must determine whether the image is real or fake.

The Pix2Pix architecture is composed of two key components: a Generator G and a Discriminator D. The Generator G utilizes an encoder-decoder network or U-Net with skip connections, while the Discriminator follows a patch-GAN architecture.

U-Net Generator

Unlike a conventional GAN model, the U-Net generator takes an image as input, and randomness is introduced through the utilization of dropout layers. These dropout layers are incorporated during both the training phase and when making predictions.

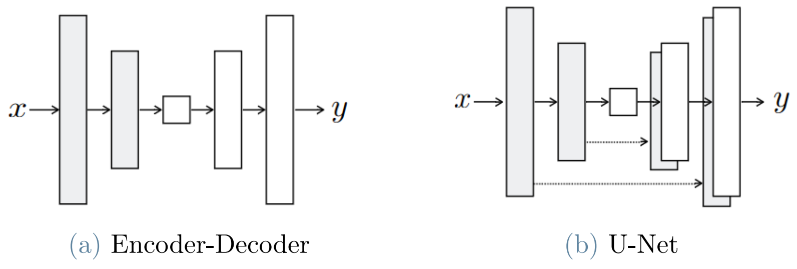


Figure 3.19: Encoder-Decoder and U-Net comparison [38].

The U-Net's architecture can be conceptually divided into an *encoder network* followed by a *decoder network*. In the typical encoder-decoder configuration (Figure 3.19a) the generator takes an image as input and applies a *downsampling* process across several layers until it reaches a *bottleneck layer*. Subsequently, the representation is *upsampled* across additional layers before producing the final image of the desired dimensions. The U-Net model (Figure 3.19b) shares similarities with this approach, particularly the downsampling to a bottleneck and subsequent upsampling to create the output image. However, what distinguishes the U-Net model is the incorporation of *links or skip-connections* between layers of equivalent size in the encoder and decoder. This linkage allows the bottleneck to be bypassed. For instance, the first layer of the encoder is paired with the same-sized feature maps as the last layer of the decoder, and these two layers are merged. This process is reiterated for each layer in the encoder, connecting it with the corresponding layer in the decoder, thus creating a U-shaped model, as exemplified in Figure 3.20.

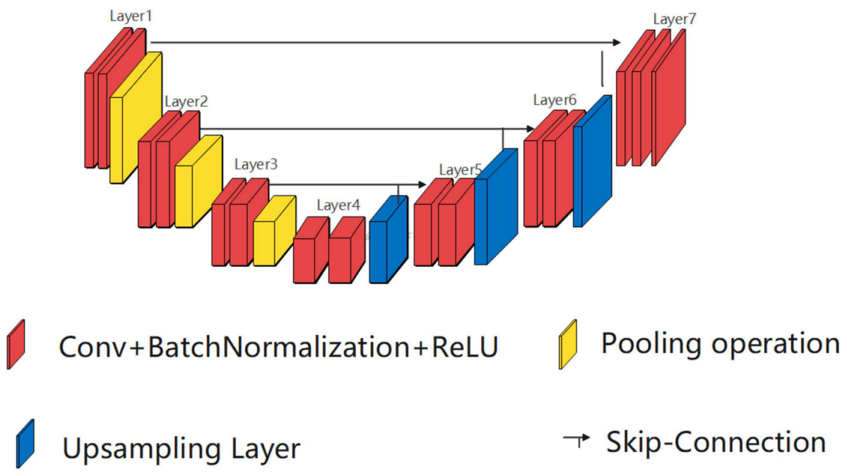


Figure 3.20: U-Net architecture

Patch-GAN Discriminator

The discriminator model in the Pix2Pix framework is responsible for evaluating images from both the source and target domains, determining the likelihood of whether the target domain image is a real or a generated counterpart.

Therefore, an essential requirement for training the discriminator model is the availability of an image dataset consisting of paired source and target images. These paired images serve as the basis for the model to learn the relationships between the two domains.

In contrast to the conventional GAN model, which employs a deep convolutional neural network to classify entire images, the Pix2Pix model utilizes a **PatchGAN**. The Patch-GAN is a deep convolutional neural network designed to classify patches of an input image as real or fake, rather than making an assessment about the entire image. This approach allows to capture fine-grained, high-frequency details, improves computational efficiency compared to evaluating whole images, and results in a model with fewer parameters. The output of the network is a feature map of real/fake predictions, which can be averaged to produce a single score.

3.4.3. Dataset generation

The primary objective of the Pix2Pix network is to reconstruct the profile of a sole with burrs where it is obstructed. In an ideal scenario, this network should be capable of reconstructing any kind of profile, given that the shape of the burrs is random. The dataset used for training the Pix2Pix model, then, includes several instances of segmented soles with burrs, enhancing the model's ability to generalize across different scenarios.

To train Pix2Pix, the dataset should be structured with pairs of images, where each pair consists of:

- **Ground truth**: This image represents the real segmentation of the sole without any occlusions. Essentially, it is what the model should ideally identify.
- **Model input:** This image depicts the segmentation of the sole with occlusions, serving as the input for the model, which needs to reconstruct the unobstructed profile.

Creating such a structured dataset with a significant volume of images manually can be time-consuming. To overcome this challenge, a Python script has been developed to automatically generate occlusions. Ten distinct profiles of soles with burrs have been used as ground truths to be reconstructed. The script operates by randomly rotating and translating the ground truth profile within the image domain, ensuring that the entire profile remains inside the image without being cut off. This step enhances the model's generalization capabilities, making it independent of the orientation and position of the sole within the image.

After generating the ground truths, the script proceeds to create occlusions. Specifically, ten white circles have been drawn to occlude the black segmentation, simulating the effect of ring and screw occlusions. The centers of these circles have been placed at random points along the contour of the sole, and the radius of each circle has been randomly set within a range of 10 to 30 pixels.

To build the dataset, starting with ten profiles, 25 ground truths have been created for each profile, and for each ground truth, 40 images with random occlusions have been generated. As a result, the final dataset comprises 10000 elements. Some examples are reported in Figure 3.21.

For training and evaluation purposes, the dataset has been divided into training and validation subsets, with 80% of the elements allocated for training and the remaining 20% reserved for validation.

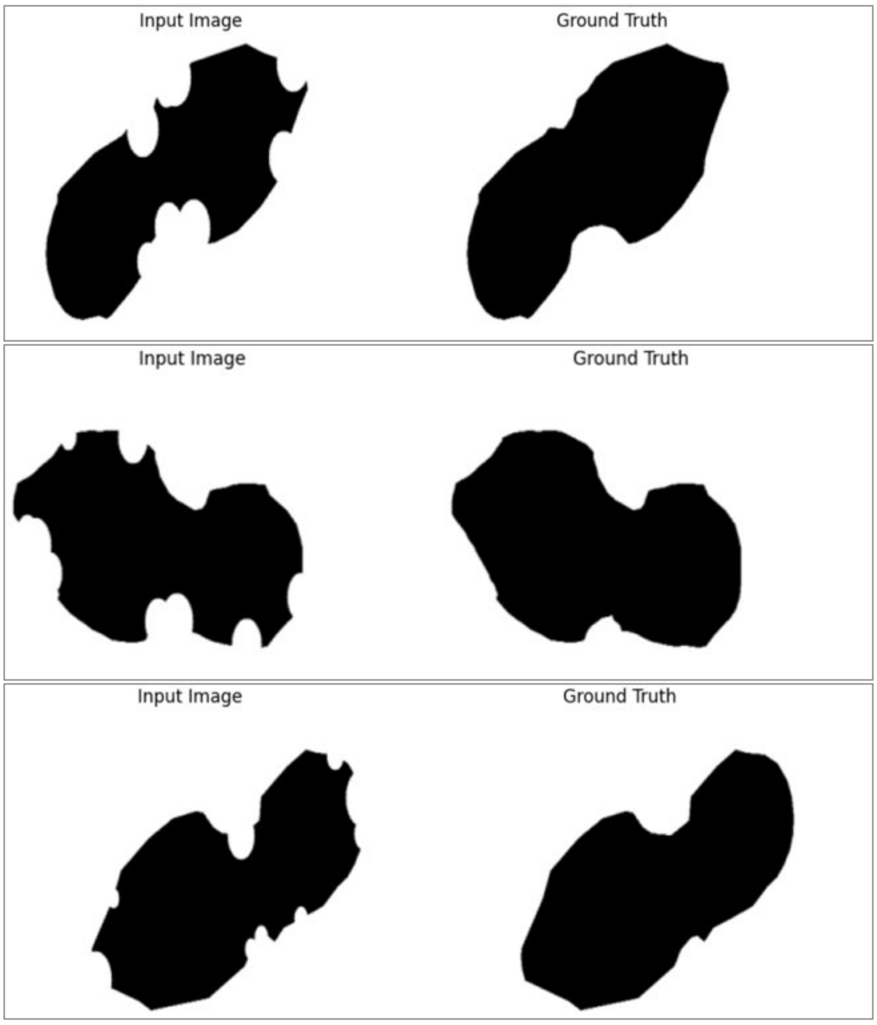


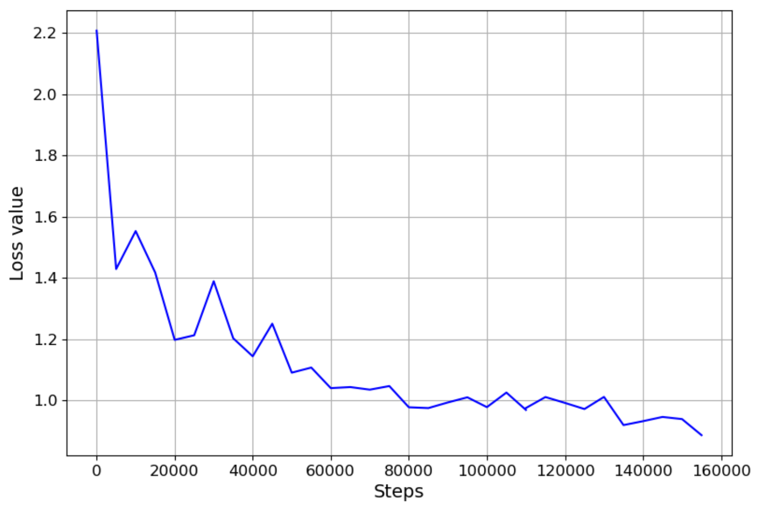
Figure 3.21: Dataset Pix2Pix

3.4.4. Implementation details

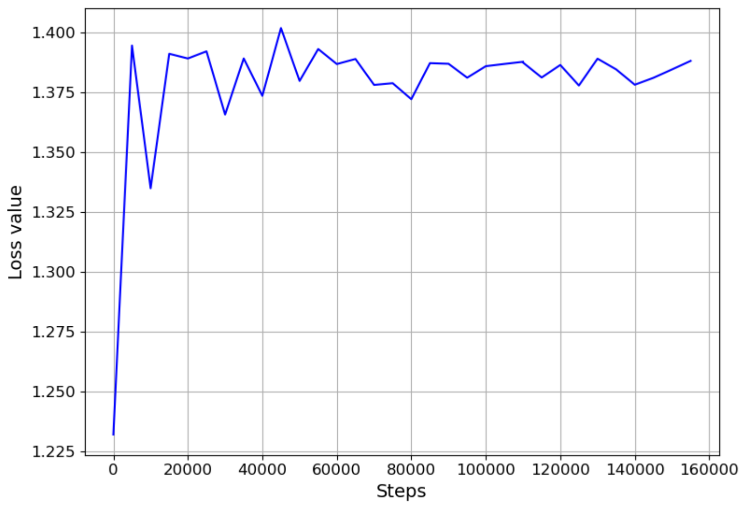
The model has been trained for 30000 steps using the Adam optimizer [42] for both the generator and the discriminator. The initial learning rate was set to 1×10^{-4} with the default parameters (exponential decay rates for the moment estimates, specifically $beta\_1 = 0.9$ and $beta\_2 = 0.999$).

3.4.5. Results

The training curves, which depict the losses of both the generator and discriminator models, are illustrated in Figure 3.22. Both curves exhibit a plateau behavior, suggesting that the training has reached its completion.



(a) Generator total loss



(b) Discriminator total loss

Figure 3.22: Generator and Discriminator training losses evolution

Some examples of the predicted reconstruction from the validation dataset are reported in Figure 3.23. It is visually evident that the model correctly reconstructs the occluded portions of the profile regardless of the burr's specific shapes.

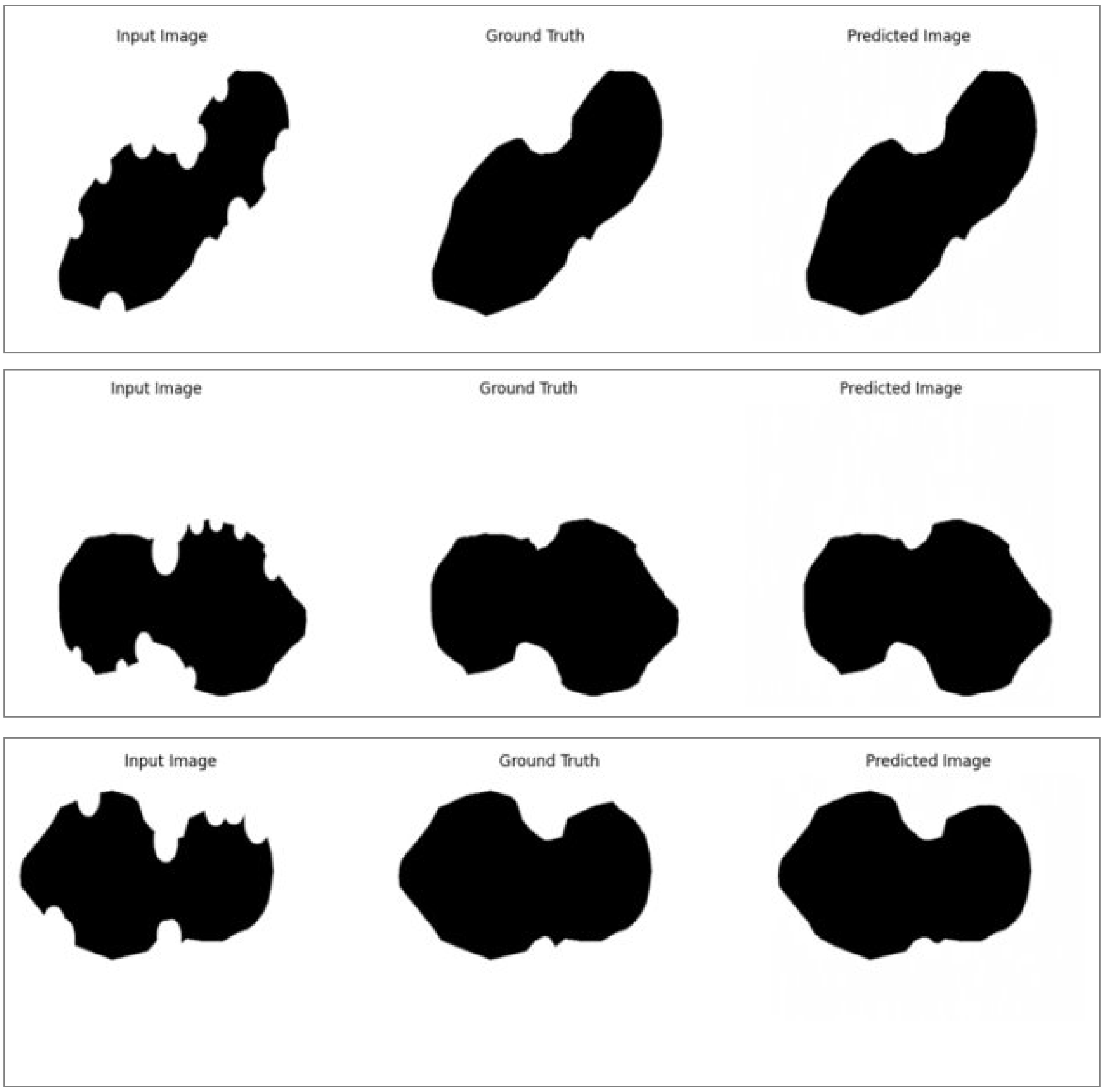


Figure 3.23: Pix2Pix result

The model has been subsequently assessed using the occluded segmentation produced by Detectron-2. The outcome is displayed in Figure 3.24, revealing the model's inability to accurately reconstruct the occluded sole's profile. This unexpected result has been attributed to the irregularities in the segmentation's profile from Detectron-2, which exhibits a segmented shape both in areas with real occlusions and those without. The model interprets these oscillations as small occlusions and attempts to address them, leading to a blurred incorrect reconstructed profile.

For future developments, then, it is essential to introduce an extra post-processing step for the segmentation generated by Detectron-2. Specifically, it should be capable of dis-

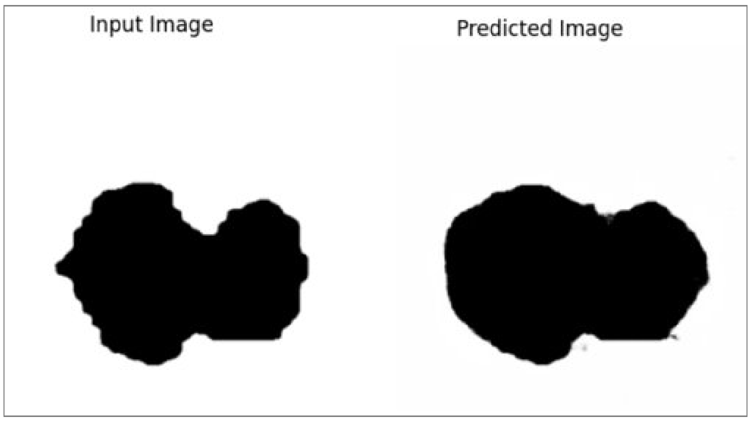


Figure 3.24: Pix2Pix reconstruction of the segmentation from Detectron-2

tinguishing sections with occlusions from those without. This separation will allow the smoothing of the profile in areas lacking occlusions, eliminating the oscillating behavior. The model, then, should be able to correctly reconstruct the occluded portions of the segmentation.

4 | Burrs identification method

Up to this point, the project pipeline has followed the sequence described in Chapter 3, which can be summarized as:

1. Detection of the sole and generation of its corresponding mask through Detectron-2 [80];
2. Reconstruction of the occluded portions of the mask through the use of Generative Adversarial Networks (GAN), specifically Pix2Pix.

As a result, a meticulous segmentation of the sole with burrs has been successfully achieved. Given that the ultimate objective is to eliminate these burrs, a novel method for burrs identification has been developed. Leveraging computer vision and image processing techniques, implemented in Python with the aid of the OpenCV library [18], the nominal profile of the sole can be overlaid onto the profile containing the burrs in their correct positions. This approach serves to precisely delineate the path that the deburring tool should follow and, consequently, find the regions that require removal.

A significant achievement lies in the development of a method that demonstrates scale, position, and orientation independence. This means that the sole with burrs can vary in size, be oriented in any direction, and be located anywhere within the acquired image, and the method remains effective.

In a concise overview of the method's outline, the following steps are employed:

1. Extraction of the nominal profile from an image featuring a sole without burrs;
2. Calculation of the oriented bounding box for this nominal profile (An oriented bounding box is a rectangle whose edges are not parallel to the basis vectors of the frame but are instead rotated to align with the object's orientation [67]);
3. Scaling, rotation, and translation of this bounding box to standardize its size and position at the origin of the plane. This establishes a "template" of the nominal profile;

Subsequently, when the segmented image of the sole with burrs is available, the method proceeds as follows:

1. Calculation of the oriented bounding box for the sole with burrs. The longest edge of this rectangle determines its size, the rotation angle relative to the horizontal axis specifies the rotation value, and the center of the bounding box indicates its position;
2. Scaling, rotation, and translation of the nominal template to match the dimensions, orientation and center with the bounding box of the sole with burrs.

The details of the aforementioned steps are presented in Chapters 4.1 and 4.2.

However, it's worth noting that this initial correspondence doesn't always yield the correct alignment of the nominal profile within the one containing burrs. Therefore, this initial alignment serves as a starting point for an optimization process aimed at maximizing the overlapping area between the nominal profile and the one with burrs. This optimization involves fine-tuning the scale, rotation and translation of the nominal profile. It is described in detail in Chapter 4.3.

It's noteworthy that this method demonstrates computational efficiency during the optimization search and can attain an overlap percentage (measured as the intersecting area over the nominal one) exceeding 99.5%.

4.1. Nominal sole profile template extraction

To enable burr recognition, the nominal profile of the sole must be known. In this methodology, the nominal profile is obtained through computer vision and image-processing techniques, utilizing a real sole that is free of burrs. This process is carried out using the Python OpenCV library [18].

The initial step involves capturing an image of the sole. The utilized image is represented in Figure 4.1. To optimize and facilitate the subsequent processing, the black sole is strategically positioned against a bright light gray background. This contrast enhances the ease of extracting a binary image.

Subsequently, OpenCV [18] is employed to conduct contour detection, which involves identifying the borders of the sole. In this context, a contour represents a collection of boundary pixels that share the same color and intensity. Following the approach suggested in [37], to prepare for this contour detection process a thresholding operation is initially applied to obtain a binary image (Figure 4.2a). In particular, the simple

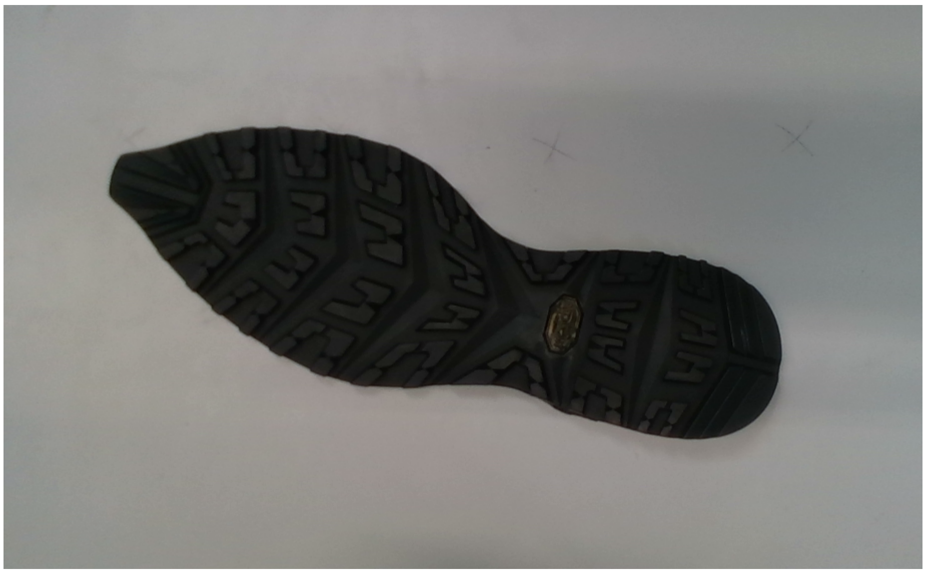
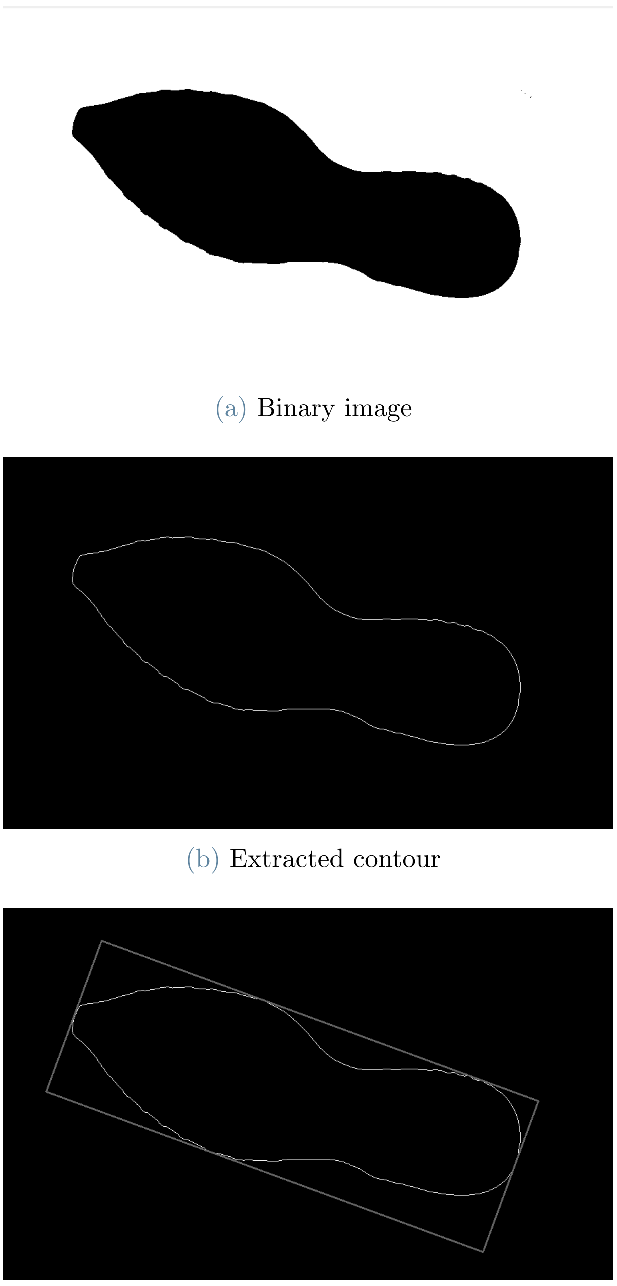


Figure 4.1: Image of the nominal sole used for profile extraction

global thresholding method has been employed, selecting the threshold that optimizes the binarization result.

After this thresholding step, the image is transformed into one composed solely of black and white pixels, effectively highlighting the object of interest, which, in this case, are the borders of the sole within the image. These object borders are turned black, resulting in uniform intensity, and thereby simplifying the contour detection algorithm. The *findContours()* [9] function is subsequently utilized to identify the external contour of the nominal sole (Figure 4.2b).

Subsequent to the contour identification, the oriented bounding box [67] encasing it is determined using the OpenCV function *minAreaRect()* [10] (Figure 4.2c). This bounding rectangle is rotated to minimize its area. As a result, the orientation within the plane of the profile, as well as its dimensions, are extracted


(a) Binary image

(b) Extracted contour

(c) Oriented Bounding Box

Figure 4.2: Steps for nominal contour extraction

Afterward, the nominal contour undergoes the following transformations:

1. Rotation of the initially oriented bounding box to align its longest side horizontally
2. Translation to reposition the center of the bounding box at the origin of the plane
3. Scaling to ensure the longest side of the bounding box has a unitary length

As a result of these operations, a **template** representing the nominal profile of the sole is generated. This template is now ready to be adjusted through translation, rotation, and scaling to fit precisely within the profile containing the burrs. These adjustments are made once the position, orientation, and dimensions of the profile with burrs are determined.

4.2. Contour matching algorithm initialization

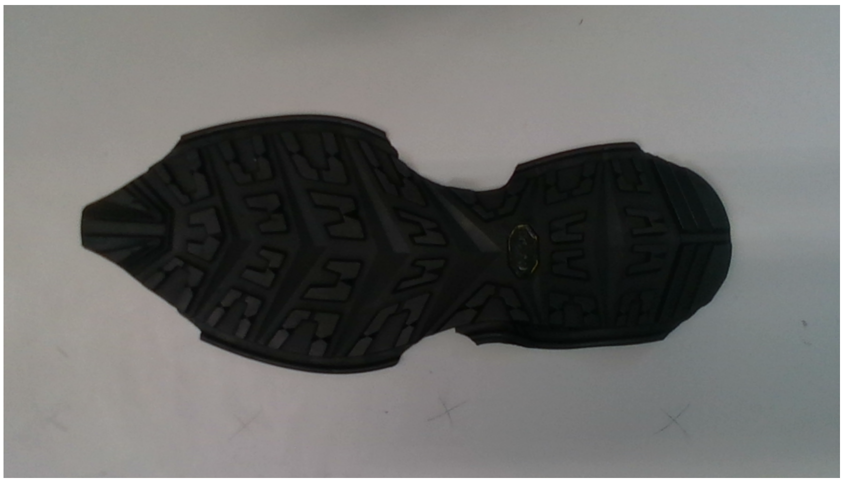
The segmentation of the sole with burrs has been obtained through Detectron-2 and Pix2Pix as described in Chapter 3. The objective of the burrs-detection method is to correctly overlay the nominal profile inside the one with burrs in order to recognize the regions of the sole to be deburred.

To achieve this, it is crucial to determine the position, orientation, and dimensions of the sole designated for deburring. In this study, the chosen approach hinges on the use of the oriented bounding box. To obtain this information, the OpenCV function *minAreaRect()* [10] has been employed. The oriented bounding box provides the necessary information for the transformation of the nominal template.

The nominal template is adjusted through the following steps:

1. Scaling is applied to the template so that the longest edges of both bounding boxes become equal;
2. The scaled template is subsequently translated to align the centers of the two bounding boxes;
3. Finally, the scaled and translated template undergoes a rotation process to align with the orientation of the bounding box containing the burrs.

Figure 4.3a serves as a sample to showcase the behavior of the method. Its segmentation is found through Detectron-2 [80] (as described in Chapter 3) and the previously outlined steps are carried out in order to determine the initial guess of overlap between the nominal profile (found in Chapter 4.1) and the profile containing burrs. The final result is shown in Figure 4.3b.


(a) Starting image with burrs

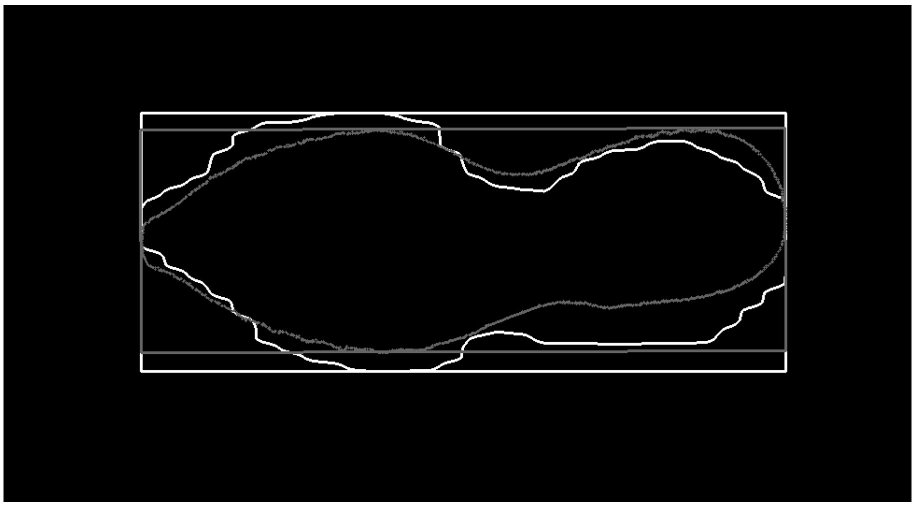

(b) Initial guess of overlap

Figure 4.3: Output of contour matching initialization stage

To assess the accuracy and effectiveness of the code, the **percentage of overlapping area** is introduced. This metric is defined as:

$$Percentage\ of\ Overlapping\ Area = \frac{Intersecting\ Area}{Nominal\ Area} \cdot 100\ (\leq 100\%) \tag{4.1}$$

so the ratio between the area of intersection between the nominal profile and the profile with burrs and the area of the nominal profile itself. The ideal position and orientation of the nominal profile within the profile containing burrs would yield a percentage of overlapping area of 100%.

It is evident that the positioning of the nominal profile within the profile containing burrs in Figure 4.3b is not entirely accurate. The percentage of overlapping area, in this case, stands at only 94.57%. This discrepancy can be attributed to the fact that the transformations applied to the nominal template are based on data obtained from the oriented bounding box of the profile with burrs. This bounding box is determined as the minimum-area rectangle around the contour and is significantly influenced by the shape and dimensions of the burrs. For instance, in scenarios where burrs are unevenly distributed around the profile, the center of the bounding box tends to shift towards the side with larger burrs. Moreover, the orientation of the rectangle can also exhibit variation. Consequently, the nominal profile with coincident centers and the same orientation may result in an incorrect overlap.

It is noteworthy, however, that the outcome depicted in Figure 4.3b represents a fortuitous scenario where burrs are absent from the top and bottom regions of the sole. Consequently, the bounding box's longest side inherently provides the accurate scaling and dimensions of the nominal sole, thereby augmenting the percentage of overlapping area. In cases where burrs extend to the top and bottom sections of the sole, any inaccuracies in the scaling of the nominal template would be corrected in the next steps.

To determine the precise positioning of the nominal profile, an optimization procedure is carried out. The initial location obtained through the bounding boxes' alignment serves as the starting point for this search.

4.3. Contour matching optimization

The objective is to **maximize the percentage of overlapping area**, aiming to find the accurate position of the nominal profile within the one containing burrs.

The method relies on image processing techniques, the metric needs to be calculated every time the configuration changes. The optimization in this study, then, relies on a **direct search** of the solution. To accomplish this, the nominal profile is systematically scaled, translated and rotated into various configurations, thus exploring the neighboring space. The specific transformation that optimizes this objective is then extracted.

The implemented procedure to achieve this goal is the following:

1. Initiate from the position determined by the bounding boxes' match, as described in the preceding section, and calculate the initial percentage of overlapping area;

2. Set the ranges and step sizes for scaling, rotation and translation **based on the initial percentage of overlap**. A lower overlap percentage corresponds to larger

movement ranges and steps, while a higher overlap percentage entails smaller ranges and steps;

3. Following each relocation and rescaling of the nominal profile, calculate the new percentage of overlap. If it doesn't exhibit improvement compared to the previous result, implement another transformation step on the nominal profile. However, if there is an improvement, **redefine the ranges** and step sizes, making them smaller than the previous settings. This discovered position then serves as the initial point for the subsequent search iteration;

4. Continue the procedure until the percentage of overlapping area exceeds **99.5%**. Once achieved, store the final scale, position and orientation and conclude the optimization process.

The refinement of the ranges and step sizes after each objective improvement significantly accelerates the convergence of the process compared to a straightforward global search method. This adaptive approach optimally balances the search space exploration. When the overlap is initially low, a wider range ensures finding a better position and a broader step size speeds up the search. As the overlap percentage increases, precision becomes more crucial, necessitating a finer refinement with a reduced range and step size. This fine-tuned approach allows for pixel-level adjustments and small-scale transformations in the vicinity of the previous step's location.

It is important to note a significant aspect of the algorithm. The computation of the Percentage of Overlapping Area (P.O.A.), as defined in Equation 4.1, involves the nominal area in the denominator. This design choice ensures a variable for maximization, and 100% indicates the entire nominal contour is correctly positioned within the sole with burrs. However, a potential issue may arise if the code generated a nominal scale considerably smaller than that of the sole with burrs. In such cases, the nominal profile might be entirely immersed in the one with burrs, yielding a P.O.A. of 100%, yet the result is incorrect due to the small scale.

To avoid this concern, when a scaling range and step are specified, the sole is initially set to the maximum scale within the range and subsequently downscaled from the maximum to the minimum scale. Given the continuous updating of the P.O.A. at each step, with adjustments to ranges and steps based on improvements, the algorithm consistently yields the optimal overlapping solution with the largest scale possible. This measure ensures that the algorithm avoids the potential mistake of erroneously considering the nominal profile too small relative to the sole with burrs, enhancing the accuracy and reliability of the results.

The iterative process concludes when the percentage of overlap surpasses the 99.5% threshold, guaranteeing an exceptionally high level of precision in burrs identification for the subsequent robotic deburring operation.

4.4. Results

Starting from the example presented in Figure 4.3b in the preceding chapter, the optimization procedure ultimately yields the final outcome illustrated in Figure 4.4. By applying this optimization method, the percentage of overlapping area improves **from 94.57% to 99.63%**.

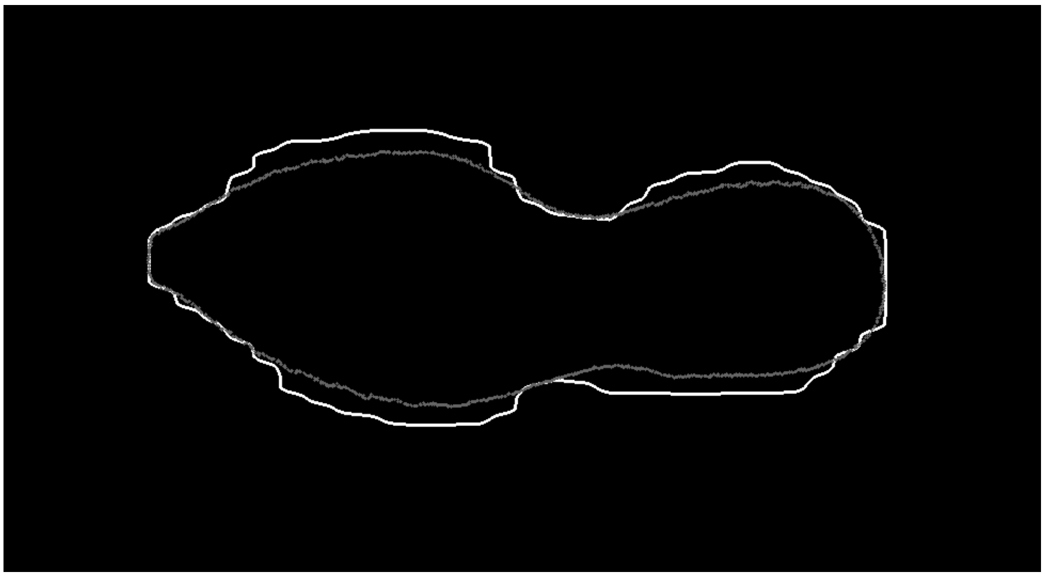


Figure 4.4: Optimized contour matching for burrs identification

From this profile configuration, deriving the segmentation of the burrs is straightforward. Essentially, the burrs correspond to the area located between the profile with burrs and the one without them. Therefore, the two contours have been filled with white masks using OpenCV. To obtain only the white regions of the burrs, the XOR (exclusive OR) logical operator has been applied. Since the white masks consist of pixels with an intensity equal to 1, the pixel-wise application of the XOR logical operator allows retaining only the pixels belonging to either the sole with burrs or the sole without, but not the areas in common. The result is shown in Figure 4.5.

It is evident how, due to small errors in the image processing process and from the segmentation of Detectron-2, some areas along the profile are wrongly identified as burrs. These must be filtered out. In order to retain only the thickest region a morphological

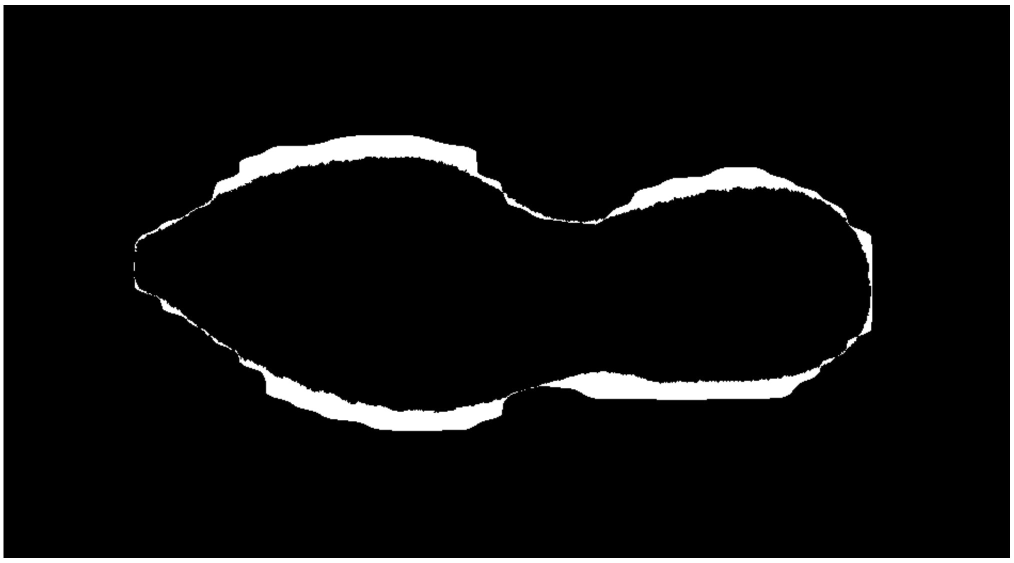


Figure 4.5: Identified segmentation of the burrs

operation [72] has been implemented. Using a 5-pixel square structuring element, an erosion operation [64] is applied. The structuring element is moved around the image and a white portion is preserved if and only if all the pixels within the structuring element have an intensity equal to 1 (white), otherwise it is entirely set to black (background). This process effectively eliminates small areas previously mistaken as part of the sole, ensuring that only the thickest physical burrs are retained.

The final result is depicted in Figure 4.6.

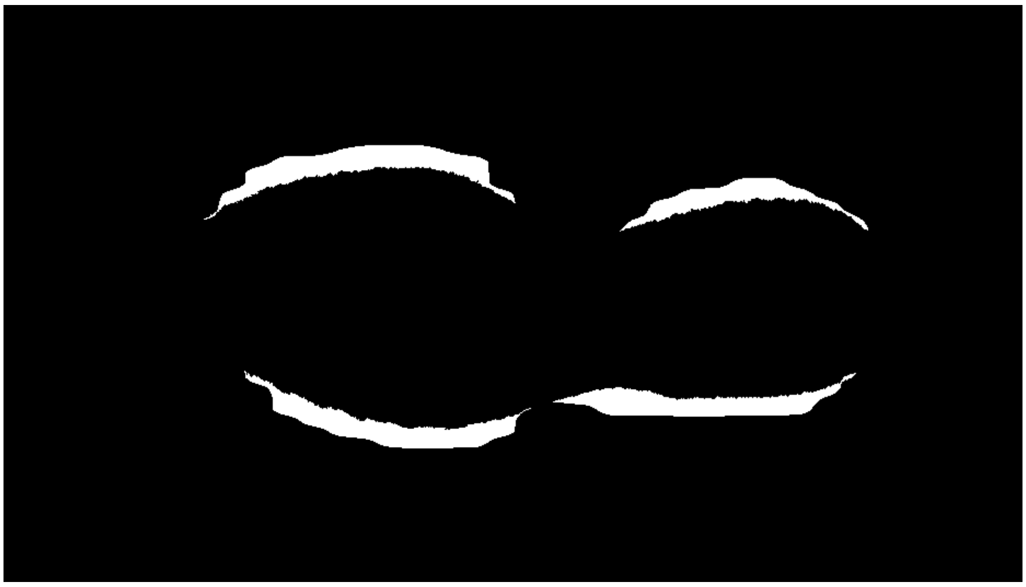


Figure 4.6: Identified segmentation of the burrs

5 | Tool pose estimation

Up to this point in the study, the path planning for the deburring process includes the positions that the deburring tool must follow, as defined in earlier chapters, specifically the collection of points composing the nominal profile correctly positioned inside the one with burrs (Chapter 4.4).

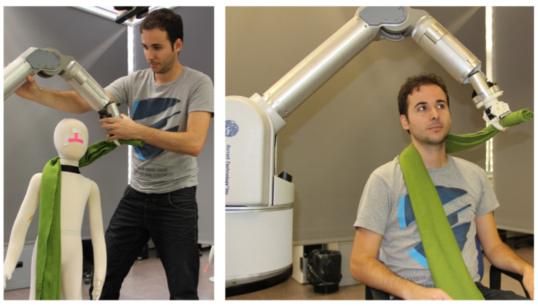
However, for effective deburring, the optimal orientation of the deburring tool during cutting is crucial. The orientation of the tool plays a fundamental role in the correct execution of the deburring process on the sole, as the shape of its blade is optimized for cutting in a specific pose, and the cutting blade is only present on one side of the tool's tip. The ideal orientation, therefore, depends on the shape and position of the burr, particularly the local curvature. The tool should be inclined in a specific way to achieve an optimal cut.

In this study, the proposed approach involves teaching the optimal orientation to the robot through human demonstration.

In the field of robotics and automation, **learning from demonstration (LfD)** [83] involves allowing a robot to acquire new skills by learning to imitate an expert demonstrator.

There exist several methods for acquiring demonstrations, broadly categorized into direct and indirect techniques [26]. In **direct** demonstration, the learning agent itself performs the task. Kinesthetic teaching (Figure 5.1a), for example, involves physically manipulating the robot to guide its movements. On the other hand, **indirect** demonstration involves an external agent demonstrating the task. In the case of indirect demonstration by observation (Figure 5.1b), for instance, a teacher performs the task, and the demonstrations are recorded through cameras.

For this study, where the goal is to determine the optimal orientation of a deburring tool, an expert in deburring performs the task. Given the need for high precision in deburring tasks, especially regarding the small dimension of the tool's cutting edge and the precise positioning, direct methods such as kinesthetic teaching may pose challenges. Therefore,


(a) Kinesthetic teaching [25]

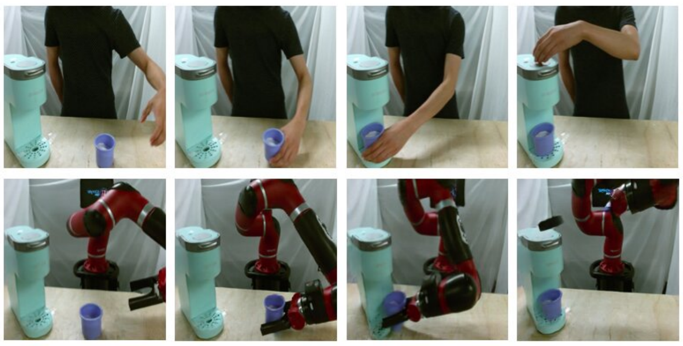

(b) Visual indirect teaching [69]

Figure 5.1

an indirect method has been chosen, where the demonstrator is free to perform deburring effectively without the constraints of physically moving the robot. Specifically, videos of experts performing deburring tasks are analyzed, and the pose of the tool is extracted from these videos. To determine the 6D pose of the tool from the videos, a deep learning approach for pose estimation from RGB images has been employed.

5.1. 6D pose estimation from RGB images

Pose estimation techniques provide the position and orientation of the detected object with respect to the camera frame.

Over the years, with the advent of Convolutional Neural Networks (CNNs), artificial intelligence and deep learning have significantly influenced the field of image processing, particularly in subfields such as object detection and 6D pose estimation. These deep learning methods for pose estimation can be categorized into three primary types: 2D-3D correspondence, direct estimation, and pose refinement.

- **2D-3D Correspondence:** In this approach, a neural network is trained to identify 2D key points in an image of the target object. Subsequently, a Perspective-n-Point

(PnP) [59] algorithm is employed to determine the 6D pose of the object.

- **Direct estimation:** Direct estimation methods leverage CNNs to predict the pose of the target object directly. These techniques extend from instance segmentation models like Mask R-CNN.

- **Pose refinement:** Pose refinement methods refine the initial pose estimation, improving its accuracy. This refinement step follows the pose predictions made by the previous two categories.

For the specific objective of this study, **EfficientPose** [19] has been selected as the network for deburring tool pose estimation. EfficientPose is a direct estimation method that extends the segmentation network EfficientDet. It stands out due to its state-of-the-art performance, simplicity, and relatively low computational cost. Moreover, it also includes a pose refinement step, making it well-suited for the study's purposes.

5.1.1. Efficient Pose

EfficientPose [19] is a cutting-edge, supervised learning technique specifically designed for the estimation of 6D object poses using RGB images. It is widely cited in the literature for its efficiency, scalability, and exceptional accuracy. It is a one-shot method that performs 2D bounding box detection and full 6D pose estimation. This 6D pose includes the calculation of 3D rotation (R) and 3D translation (t). Essentially, the 6D pose describes the rigid transformation from the object's coordinate system to that of the camera. It is built upon the state-of-the-art 2D object detection method EfficientDet [74], expanding its capabilities to include the prediction of object poses. Remarkably, it does so while preserving the advantageous aspects of the base network and without incurring significant additional computational costs. The extension primarily consists of the addition of two supplementary subnetworks, dedicated to forecasting object translations and rotations, similar to the subnetworks responsible for classifying objects and finding their 2D bounding boxes.

The two extra subnetworks can be described as:

- **Rotation Network:** This subnetwork is responsible for predicting a rotation vector $\mathbf{r} \in \mathbb{R}^3$. To enhance the accuracy of the initial rotation prediction, an iterative refinement module is introduced. This module takes the current rotation prediction, $\mathbf{r_{init}}$, as well as the output of the final convolution layer, before the initial regression layer. It calculates a $\Delta\mathbf{r}$ to iteratively refine the rotation prediction. This process is repeated $N_{iter} = 1 + [\phi/3]$ times, with each iteration updating the rotation as

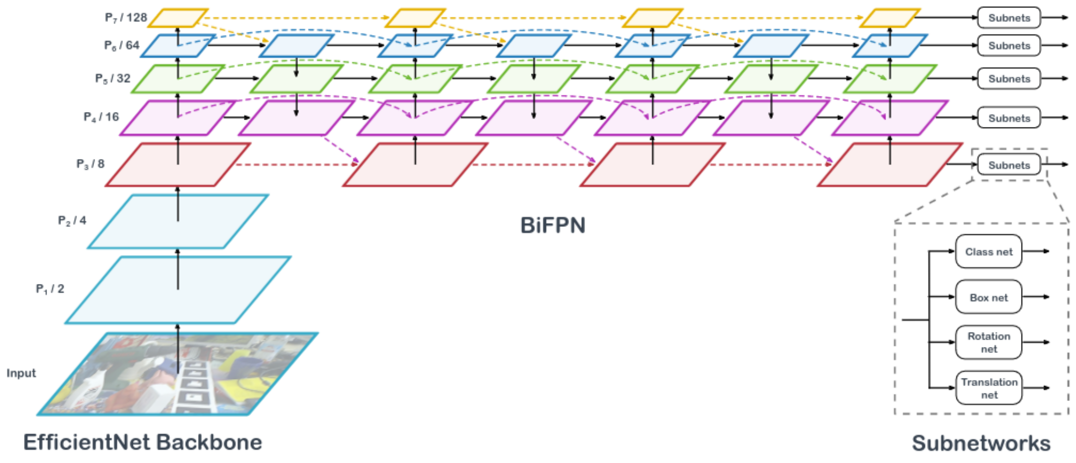


Figure 5.2: EfficientPose architecture

$\mathbf{r} = \mathbf{r_{init}} + \Delta\mathbf{r}$. The intermediate rotation $\mathbf{r}$ from each iteration is used as the new $\mathbf{r_{init}}$ for the next iteration.

- **Translation Network:** This subnetwork focuses on estimating the translation vector $\mathbf{t} \in \mathbb{R}^3$. The translation task is divided into two parts. First, it identifies the 2D center point of the object $c = [c_x, c_y]^T$ and it estimates the depth t_z. The translation vector $t = [t_x, t_y, t_z]^T$ is obtained through the camera's intrinsic parameters, using the formula

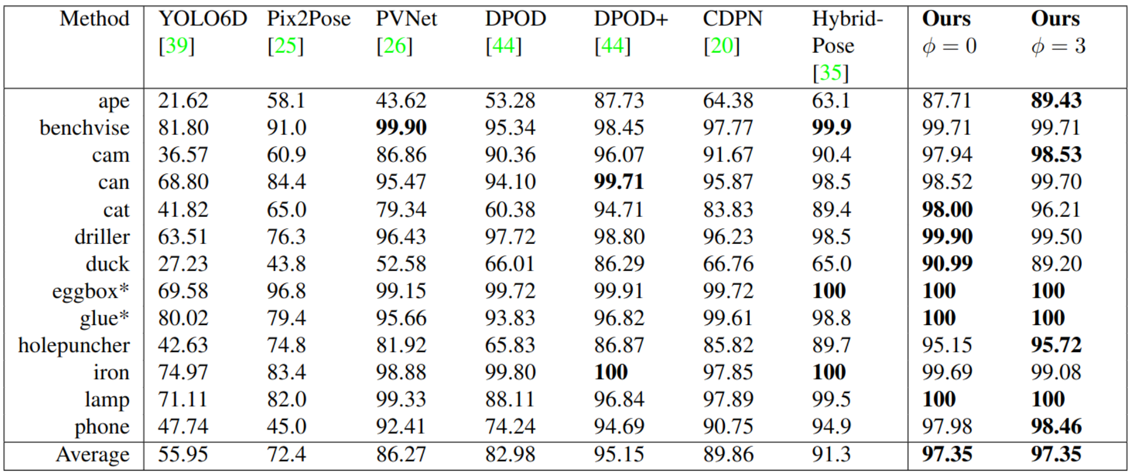
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/f_x & 0 & p_x/f_x \\ 0 & 1/f_y & p_y/f_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_x \\ cy \\ 1 \end{bmatrix} \tag{5.1}$$

where f_x, f_y are the focal lengths and (p_x, p_y) is the principal point.

A schematic representation of EfficientPose's architecture is shown in Figure 5.2.

The model is scalable thanks to the hyperparameter ϕ, which provides flexibility in adjusting the network's dimensions. This hyperparameter allows for the augmentation of the number of layers, which can enhance the model's performance in exchange for increased computational costs. Furthermore, it also influences the number of iterations used for pose refinement, granting control over the trade-off between computational resources and model accuracy.

EfficientPose introduces a significant improvement with a novel 6D augmentation technique. This technique leverages image rotation and scaling transformations to generate additional data for training the network. The key advantage of this approach is its capacity to reduce the extensive time required for data annotation, enabling the utilization of smaller initial datasets, and enhancing the network's ability to generalize from the available data. Figure 5.3, extracted from the official EfficientPose paper [19], provides a quantitative demonstration of the superior performance of this network in comparison to other widely used alternatives. The illustration showcases the ADD(-S) values, a prevalent metric for assessing pose estimation algorithms as detailed in Chapter 5.4. These values are presented across all objects within the Linemod dataset [17].



Method	YOLO6D [39]	Pix2Pose [25]	PVNet [26]	DPOD [44]	DPOD+ [44]	CDPN [20]	Hybrid-Pose [35]	**Ours** $\phi = 0$	**Ours** $\phi = 3$
ape	21.62	58.1	43.62	53.28	87.73	64.38	63.1	87.71	**89.43**
benchvise	81.80	91.0	**99.90**	95.34	98.45	97.77	**99.9**	99.71	99.71
cam	36.57	60.9	86.86	90.36	96.07	91.67	90.4	97.94	**98.53**
can	68.80	84.4	95.47	94.10	**99.71**	95.87	98.5	98.52	99.70
cat	41.82	65.0	79.34	60.38	94.71	83.83	89.4	**98.00**	96.21
driller	63.51	76.3	96.43	97.72	98.80	96.23	98.5	**99.90**	99.50
duck	27.23	43.8	52.58	66.01	86.29	66.76	65.0	**90.99**	89.20
eggbox*	69.58	96.8	99.15	99.72	99.91	99.72	**100**	**100**	**100**
glue*	80.02	79.4	95.66	93.83	96.82	99.61	98.8	**100**	**100**
holepuncher	42.63	74.8	81.92	65.83	86.87	85.82	89.7	95.15	**95.72**
iron	74.97	83.4	98.88	99.80	**100**	97.85	**100**	99.69	99.08
lamp	71.11	82.0	99.33	88.11	96.84	97.89	99.5	**100**	**100**
phone	47.74	45.0	92.41	74.24	94.69	90.75	94.9	97.98	**98.46**
Average	55.95	72.4	86.27	82.98	95.15	89.86	91.3	**97.35**	**97.35**

Figure 5.3: EfficientPose [19]: ADD(-S) Performance on Linemod Dataset [17] and Comparative Analysis with Other Pose Estimation Networks

5.2. Dataset generation

In the domain of supervised deep learning methods, dataset annotation plays a fundamental role. The quality of the annotations directly influences the quality of the results achieved: precise annotations are an essential prerequisite for obtaining accurate outcomes. Given that machine learning models learn the relationship between input and output data, the creation of a dataset comprising suitable training inputs and their corresponding ideal outputs (ground truths) is the only way to ensure the desired behavior is learned.

When considering pose estimation from RGB data, precise annotations of the full 6D pose of the target object, in a large volume of images, are required to create an effective dataset. However, the process of acquiring and annotating such data can be an exceedingly time-consuming and often costly task.

To address these challenges, this study has opted to use **synthetic data** to train the

EfficientPose network. This approach offers the advantage of generating an essentially limitless quantity of precisely labeled data, significantly reducing the time and effort needed for the dataset creation. Furthermore, it helps minimize the potential for errors that might arise from incorrect labeling. By leveraging a set of background images taken from the real working environment, the 3D model of the tool (the object of interest for pose estimation, Figure 5.4) has been placed in a predetermined position and orientation relative to the camera frame. This process results in the creation of a highly accurate ground truth annotations.

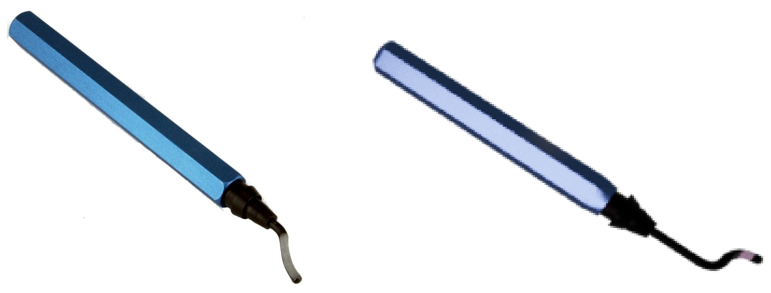

(a) (b)

Figure 5.4: Real manual deburring tool and 3D model used for dataset generation

To create the annotated synthetic dataset used for training and testing EfficientPose, **BlenderProc** [29], an open-source framework, has been utilized. BlenderProc enables the rendering of photorealistic images from generated 3D scenes. By specifying the positions of the 3D model of the deburring tool and the camera, it is possible to extract the relative 6D full pose, thereby generating a highly accurate ground truth dataset for pose estimation.

However, when it comes to deploying models trained on synthetic data into real-world environments the *reality gap* may be a significant challenge to overcome. This gap refers to the disparities and inconsistencies that often exist between the synthetic or simulated data used to train the model and the real-world data these models encounter in practical applications. Bridging the reality gap is a crucial objective, as it directly affects the effectiveness and reliability of the model when deployed in real-world scenarios.

To improve the model's capability to recognize the deburring tool and its orientation in real-world scenarios, where the tool is manually held by a human hand over the sole requiring deburring, the dataset's backgrounds are composed of images featuring soles and a combination of soles and hands, as exemplified in Figure 5.5. This approach enhances the model's robustness in identifying the tool among various elements in the image and helps narrow the reality gap by making more realistic images, thus improving accuracy.

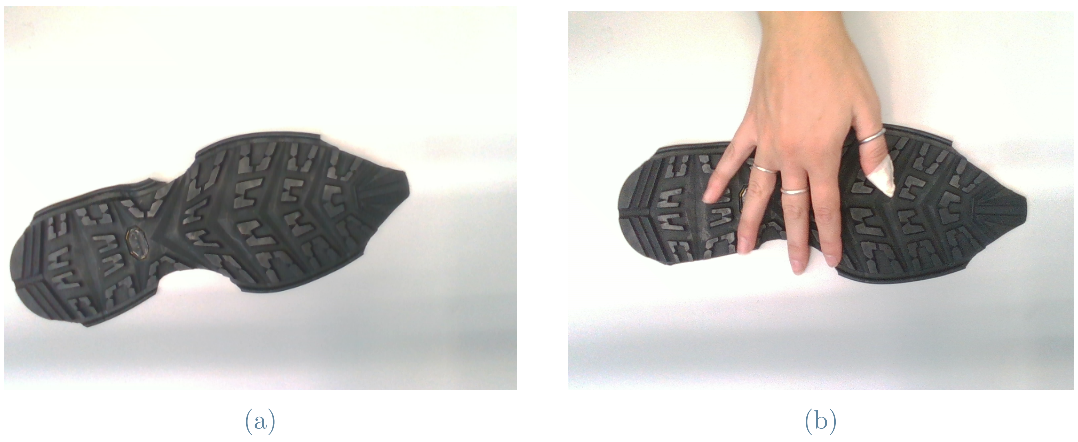

(a) (b)

Figure 5.5: Samples of EfficientPose's dataset background images

Furthermore, for more realism in synthetic images and enhanced accuracy in real-world scenarios, physical features were attributed to the deburring tool, including surface roughness and a metallic shading effect.

To ensure robust detection under varying lighting conditions, a point light source was used to generate the synthetic images. This light source is randomly moved in space for each image, providing the model with exposure to different shadowing and reflection conditions to improve its adaptability.

Considering the complexity of the pose estimation task, a decision has been made to fix the camera position within the scene. However, to introduce variability, the tool has been randomly positioned in every scene. Figure 5.6 shows two samples of random positioning. Spatial constraints have been imposed on the tool's position to ensure alignment with real-world scenarios where the model would be applied, particularly in videos of human demonstrations of deburring.

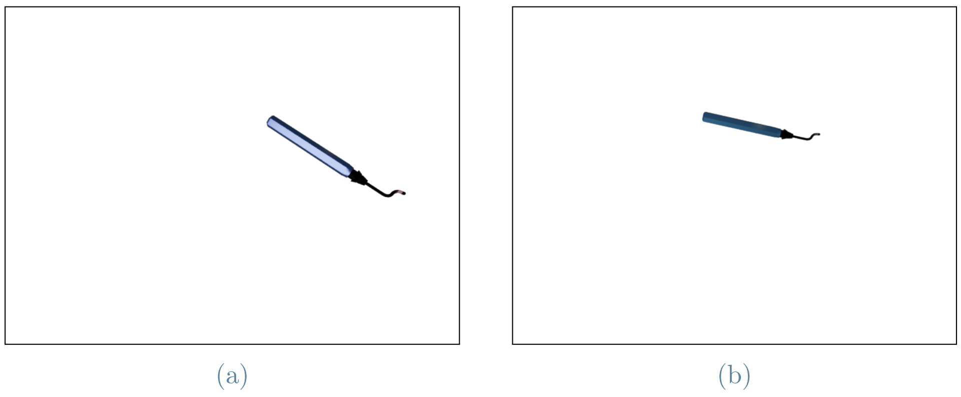

(a) (b)

Figure 5.6: Random positioning of 3D model deburring tool

A total of 10000 scenes have been created, with each scene generating a single image. Consequently, a dataset of 10000 annotated images has been obtained. This dataset has been then divided into two subsets: 80% for training (8000 elements) and 20% for validation (2000 elements). Some examples of elements of the dataset are reported in Figure 5.7.

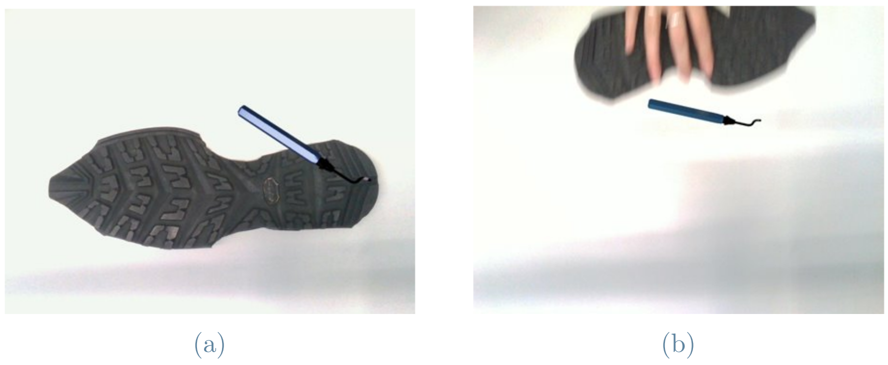

(a) (b)

Figure 5.7: EfficientPose's dataset samples

5.3. Training parameters

The model has been trained for 175 epochs and has been evaluated after every epoch. The learning rate has been dynamically changed during training using Keras' *ReduceLRonPlateau* [11] function: large rates quickly adjust the model but can lead to fluctuations, local minima and divergence; smaller rates avoid these issues but take an excessive amount of time to improve the model. The learning rate has been set as large at the beginning and then automatically decreased when the training stagnates. A reduction of 50% has been imposed after more than 25 epochs of stagnation. The initial and minimum learning rates have been set at $1e^{-4}$ and $1e^{-7}$ respectively. The scaling factor $\phi = 0$ has been used since the network must be able to predict the orientation of one single object only, so there's no need to oversize the network at the expense of the computational cost.

5.4. Evaluation Metric for Pose Estimation: ADD-S

The most commonly used metric for assessing the performance of a pose estimation network is the **ADD**, which is related to the Average Distance (AD). It quantifies the average error made by the model in predicting the pose of an object.

In this context, $\mathbf{R}$ and $\mathbf{t}$ represent the ground truth, while $\hat{\mathbf{R}}$ and $\hat{\mathbf{t}}$ represent the predictions generated by the network. Given a set of n points from the 3D model M of the object, the AD calculates the average of the pairwise distances between these points, transformed according to both the ground truth and the network's predictions (equation 5.4):

$$AD = \frac{1}{n} \sum_{x \in M} \|(\mathbf{Rx} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x} + \hat{\mathbf{t}})\| \tag{5.2}$$

A prediction is considered correct if the average distance is below a predefined threshold, typically set at 10% of the 3D model's largest dimension. The ADD metric measures the percentage of correct predictions.

In the case of the deburring tool, which features a handle with rotational symmetry, the matching between points can become ambiguous due to the lack of visual differences among multiple orientations. Consequently, the standard ADD metric may encounter significant challenges and the **ADD-S** metric is used instead. For symmetric objects the average distance is computed using the minimum distance between the predicted points and the ground truth (equation 5.4):

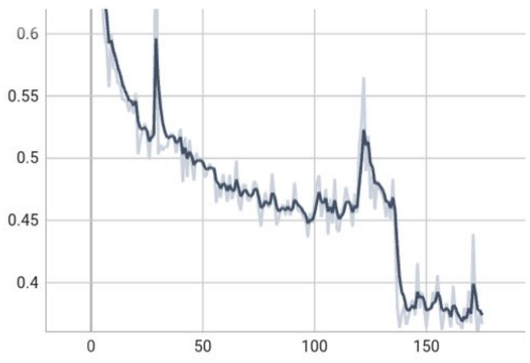
$$AD - S = \frac{1}{n} \sum_{x_1 \in M} \min_{x_2 \in M} \left\|(\mathbf{Rx_1} + \mathbf{t}) - (\hat{\mathbf{R}}\mathbf{x_2} + \hat{\mathbf{t}})\right\| \tag{5.3}$$

Analogously, the ADD-S represents the percentage of correct poses predicted.

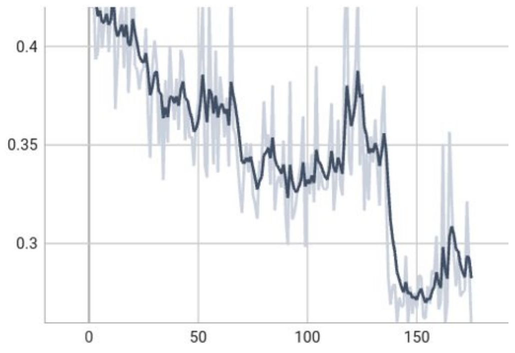
5.5. Results

The results of training EfficientPose with the dataset described in Chapter 5.2 and the training parameters outlined in Chapter 5.3 are presented.

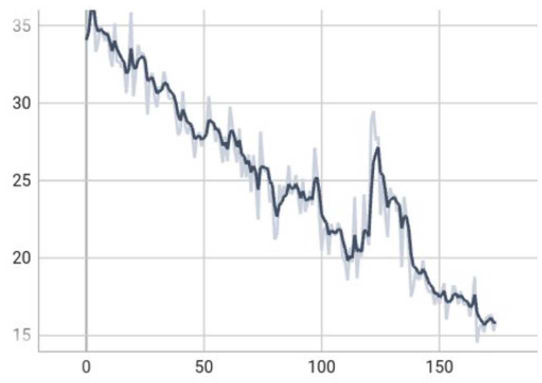
The evolution of the losses during training and validation and of the AD-S and ADD-S metrics during training is shown in Figure 5.8.



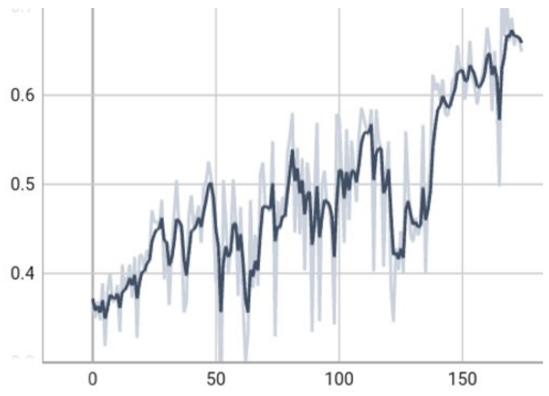
(a) Training loss



(b) Validation loss



(c) AD-S metric evolution during training



(d) ADD-S metric evolution during training

Figure 5.8: EfficientPose's training and validation curves

The model appears to stabilize at a plateau based on the loss curves, suggesting potential completion of the training. However, the evaluation metrics AD-S and ADD-S don't exhibit a stabilized behavior in the latter part, hinting at possible performance improvements with continued training. Notably, the validation loss tends to increase towards the end, indicating a potential initiation of overfitting. After experimentation, it has been observed that the loss consistently increased, and the evaluation metrics showed no improvement. Consequently, the decision has been made to conclude the training at 175 epochs.

At the end of 175 epochs, ADD-S achieves a final value of **64.85%**, with a peak during training of 68.7%. An illustrative example of a prediction from the validation dataset is presented in Figure 5.9, where the green 3D bounding box denotes the annotated ground truth, and the blue one represents the predicted pose.

Furthermore, the model has been tested on real images, showcasing its capability to generalize to real-world scenarios without significant performance degradation. As depicted

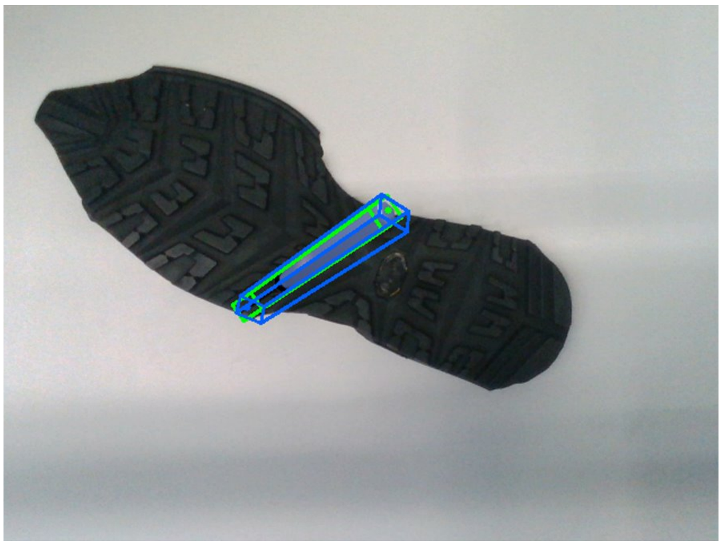
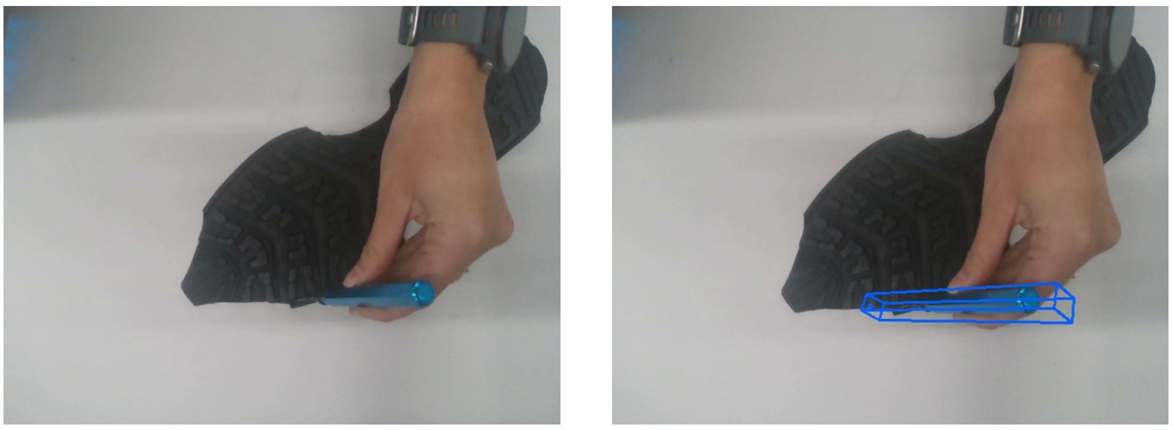


Figure 5.9: EfficientPose's prediction sample from validation dataset

in Figure 5.10, the model successfully identifies the real tool even when partially occluded by a hand and in the presence of the sole.



(a) Real image (b) EfficientPose's prediction

Figure 5.10: EfficientPose's prediction on real image

6 | Experimental results

In this chapter, all the steps outlined in the preceding chapters are brought together to accomplish the robotic deburring task. Each step in this study has proven effective in its required contribution. The ultimate objective is to integrate these diverse outputs cohesively, enabling to derive the final path, inclusive of both the position and orientation of the cutting tool. The complete pipeline has been tested both in a simulated environment and in a real-world scenario.

This chapter offers an in-depth exploration of the processes involved in translating the outputs of the previously described procedures into commands for the robot, enabling the effective execution of the deburring task. Specifically, Section 6.1 deals with the design of the deburring tool for robotic applications, while Section 6.2 provides a comprehensive description of the steps taken to derive the deburring path intended for the robot controller. The experiments have been conducted using a *Doosan A0509* robotic arm in the Mechatronics and Robotics lab for Innovation (*MERLIN* [4]) of Politecnico di Milano, and the results obtained are detailed in Section 6.2.3.

6.1. Robotic deburring tool design

The manual deburring process involves the use of a cutting tool with a design similar to the one depicted in Figure 6.1. The tool features a metal handle for manual grasping, and the cutting blade is specifically shaped to facilitate the deburring action.

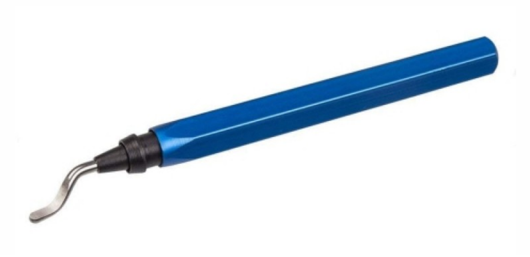


Figure 6.1: Manual deburring tool

The objective of this study is to automate this process using a robotic arm. However, conventional grasping of the tool with a gripper is deemed impractical due to the tool's small handle dimensions and shape, which prevent precise gripping and pose a risk of slippage during the task, compromising precision. Furthermore, grasping the tool conventionally would not provide accurate knowledge of the cutting edge's exact position.

For the purpose of the experimental validation of the trajectory planning pipeline, a specialized tool is essential. It allows for simulating the deburring process without the actual removal of burrs, in contrast to the real deburring operations, which would require an actuated tool, such as the one offered by *ATI automation* [1] (Figure 6.2).

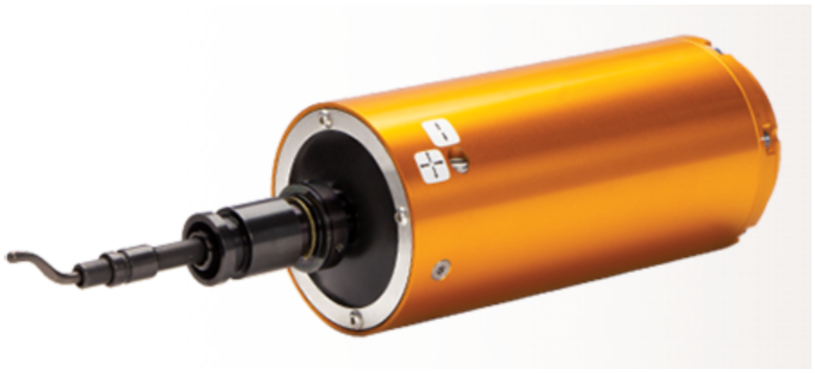


Figure 6.2: Robotic deburring tool by *ATI automation* [1]

Drawing inspiration from the ATI tool to fulfill this requirement, a custom-designed tool has been developed for attachment to the last link of the robot arm. The design comprises a base for robot attachment, a rigid body, and a specifically shaped blade. Acknowledging that a 3D-printed plastic tool might lack the necessary blade strength, the design incorporates a tool base for robot attachment, within which the manual deburring tool is securely attached. The final design is illustrated in Figure 6.3.

Specifically, the base features four $M6$ holes, positioned with centers along a circumference of 50 mm, aligning with the specifications of the *Doosan A0509* robot employed in this study. The base is 100 mm long and allows the allocation of the manual deburring tool up to the cutting tip.

6.2. Experimental setup

The experimental setup, illustrated in Figure 6.4, comprises several key elements, including the sole requiring deburring, the *Intel RealSense* RGB camera, positioned with a

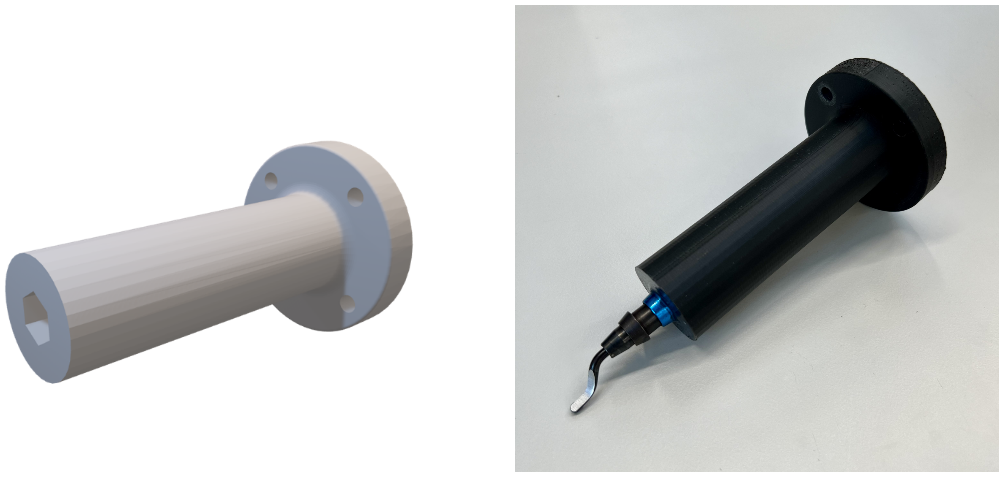

(a) 3D model of the designed tool (b) 3D printed tool

Figure 6.3: Designed tool for testing purposes

suitable support in correspondence of the final joint of the robotic arm, and the robotic arm itself, equipped with the designed mock deburring tool with the cutting blade at its tip.

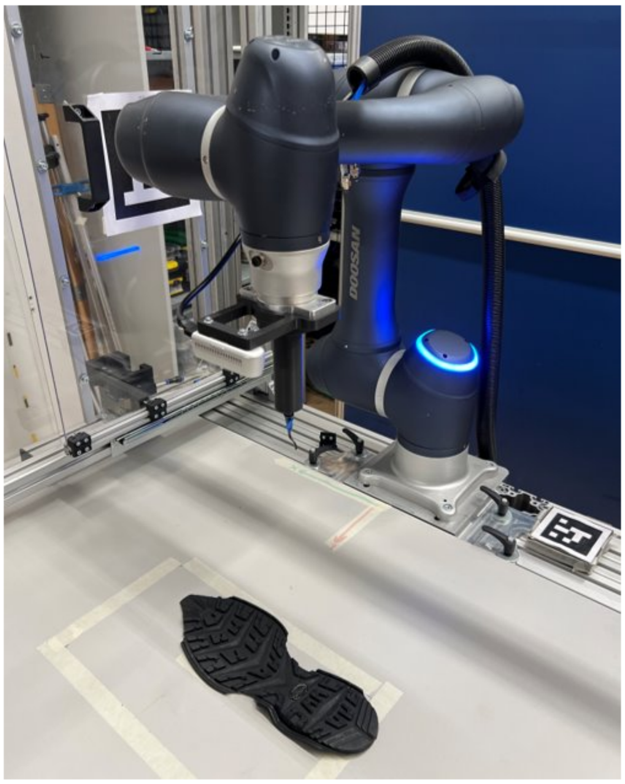


Figure 6.4: Experimental setup

6.2.1. Reference frames and transformations

Considering the multitude of elements involved in the deburring process and the computation of the deburring path, the results must be put together and transmitted to the robot controller, all with respect to the robot's reference frame. However, it is crucial to note that outcomes from the preceding steps are referenced in distinct frames.

The relevant reference frames are, therefore, detailed below.

- The acquired **image reference frame**: a spacial span of pixels, with resolution 640x480. The origin is set at the top left corner. The u axis goes from 0 to 640 from left to right, and the v axis goes from 0 to 480 from top to bottom, as shown in Figure 6.5.
- The **sole reference frame**: its origin O is set as the center of the oriented bounding box around the sole with burrs, x and y are parallel respectively to the short and long edges of the box. The z coordinate points upward and the y is directed towards the tip of the sole, as reported in Figure 6.5.

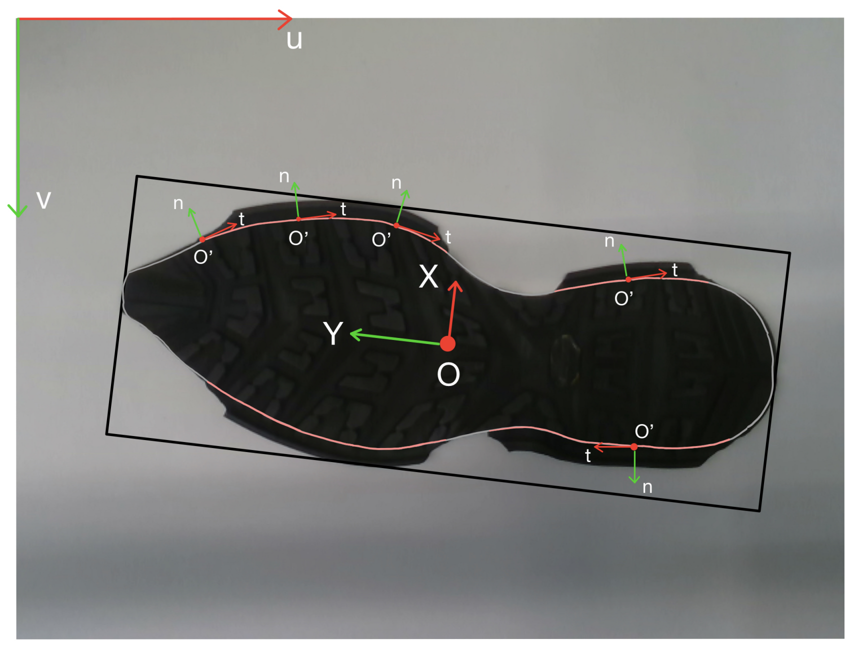


Figure 6.5: Pixel, sole and burrs reference frames

- The **burr's local reference frame**: at each point of the nominal profile where a burr is present, a local reference frame is defined, centered in O', composed by the z axis, the tangent and the normal to the profile in that point. The convention used includes having the z axis pointing upwards and the normal axis pointing outside of the sole. The tangent component's orientation is found accordingly. Some examples are reported in Figure 6.5.

- The *RealSense D435i* **camera reference frame**: centered in the RGB camera and *(x,y,z)* axis as reported in Figure 6.6.

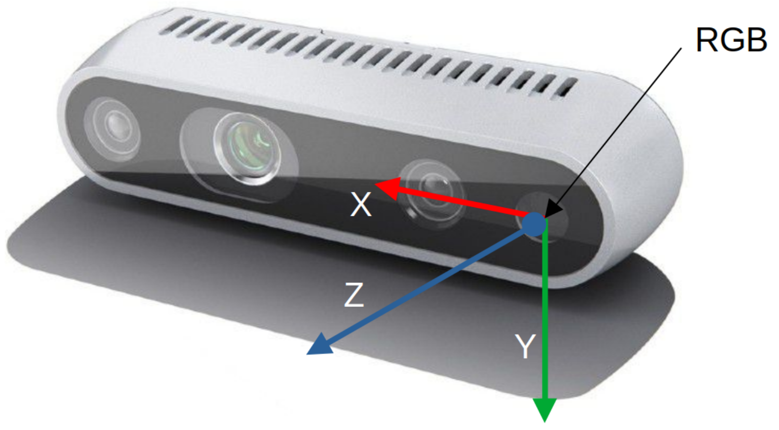


Figure 6.6: *RealSense D435i* RGB camera reference frame

- The **deburring tool base reference frame**: x and y coincident with the reference frame of the last joint of the robot, to which it is attached, while the z is directed towards the TCP (Figure 6.7)
- The **deburring tool tip reference frame** (Tool Center Point or TCP): the tangent is directed parallel to the cutting edge, the z axis points downward and the normal is found accordingly (Figure 6.7)

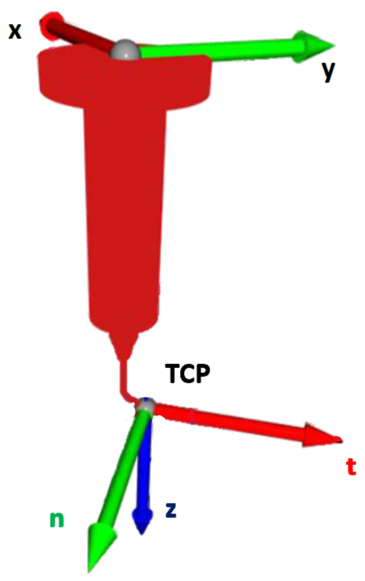


Figure 6.7: Deburring tool base reference frame and TCP reference frame

- The **robot base reference frame** (Figure 6.8)

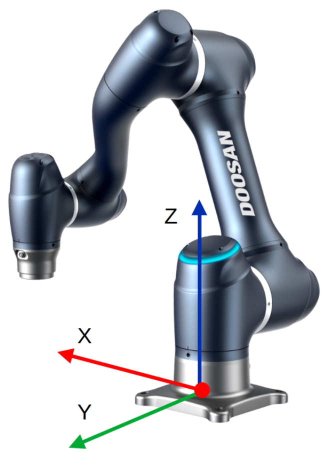


Figure 6.8: Robot base reference frame

The collection of the acting frames is visually depicted in Figure 6.9.

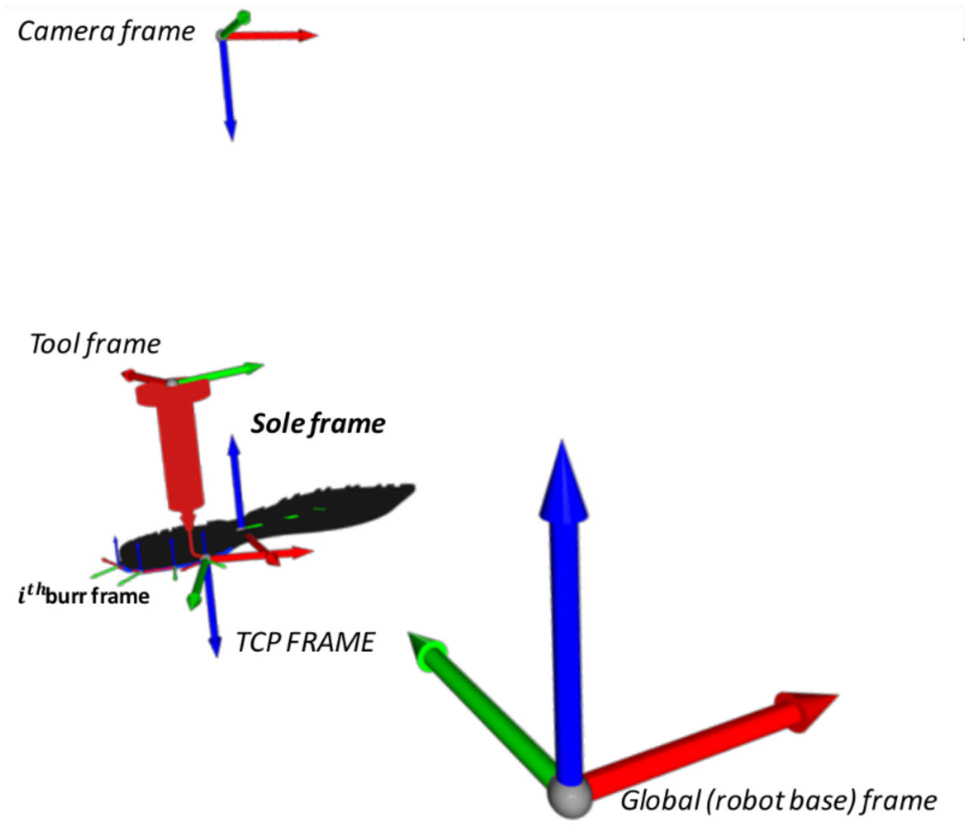


Figure 6.9: Reference frames for robotic deburring

In order to pass from one reference frame to another, transformation matrices are exploited. They are visually depicted in Figure 6.11 and described in Table 6.1. They are represented as **spacial transformation matrices**: 4x4 matrices which represent both the orientation and position of a coordinate system with respect to another. The upper-left 3x3 submatrix represents the rotation transformation while the first three rows of the last column represent the translation vector. The last row is always the vector $[0, 0, 0, 1]$. Examples of possible transformations are shown in Figure 6.10.

$$\begin{bmatrix} X & 0 & 0 & 0 \\ 0 & Y & 0 & 0 \\ 0 & 0 & Z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scale — Translation — No Change (Identity)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) & 0 \\ 0 & \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \cos(\varphi) & 0 & \sin(\varphi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} \cos(\varphi) & -\sin(\varphi) & 0 & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation along X — Rotation along Y (φ=Angle) — Rotation along Z

Figure 6.10: Simple examples of spacial transformation matrices

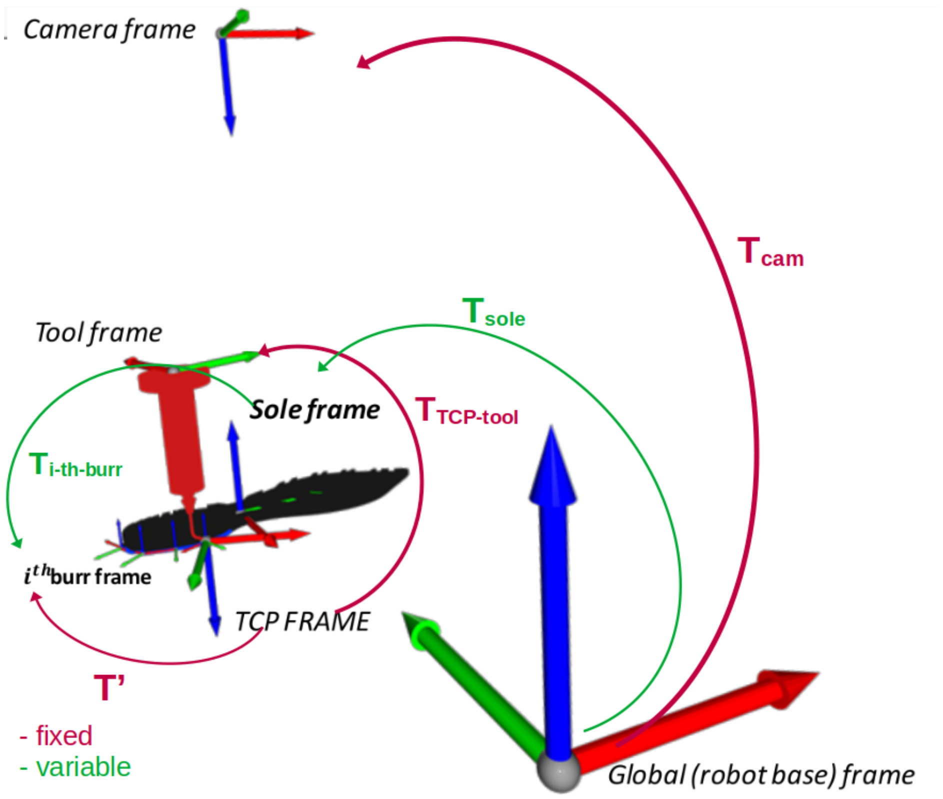


Figure 6.11: Visual representation of transformation matrices between reference systems.

Values	Description
$T_{cam} = \begin{bmatrix} 1 & 0 & 0 & -17 \\ 0 & -1 & 0 & 513 \\ 0 & 0 & -1 & 489 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Camera orientation and position in the global reference frame. Fixed.
$T' = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & t_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	TCP orientation and position in the burr local reference frame. Fixed. (**Learned from expert demonstration**)
$T_{i_{th}-burr} = \begin{bmatrix} t_x & n_x & 0 & O'_x \\ t_y & n_y & 0 & O'_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Burr point position and orientation in sole reference frame O. Variable (function of the specific burr point).
$T_{tcp-tool} = \begin{bmatrix} -0.776 & -0.629 & 0 & -12 \\ -0.629 & -0.776 & 0 & 0 \\ 0 & 0 & 1 & -166 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Tool base orientation and position in TCP ref. frame. Fixed.
$T_{sole} = \begin{bmatrix} rot\_sole\_x_x & rot\_sole\_y_x & 0 & pos\_sole_x \\ rot\_sole\_x_y & rot\_sole\_y_y & 0 & pos\_sole_y \\ 0 & 0 & 1 & depth\_sole \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Sole center position and orientation in global ref. frame. Variable (function of current position and orientation of the sole).

Table 6.1: Transformation matrices between reference systems.

6.2.2. Path planning pipeline

All the elements to build the deburring path are now in place. In this paragraph, the detailed outline of the path planning process is explained. Exploiting the results obtained in the previous chapters and transformations detailed in the previous section, the complete path of the tool is derived and communicated to the robot controller for path execution.

Image acquisition and burrs identification

The process initiates with a sole containing burrs, positioned anywhere within the robot's operational space.

The *RealSense* RGB camera captures an image of the sole (Figure 6.12a) when the robot is in the home position —a known configuration and easily accessible through a command. This serves as a consistent starting point for each deburring operation.

From the captured image, the segmentation of the sole is performed, as detailed in Chapter 3. Following segmentation, the burrs identification method, outlined in Chapter 4, is applied. The resulting identification of burrs is presented in Figure 6.12b.

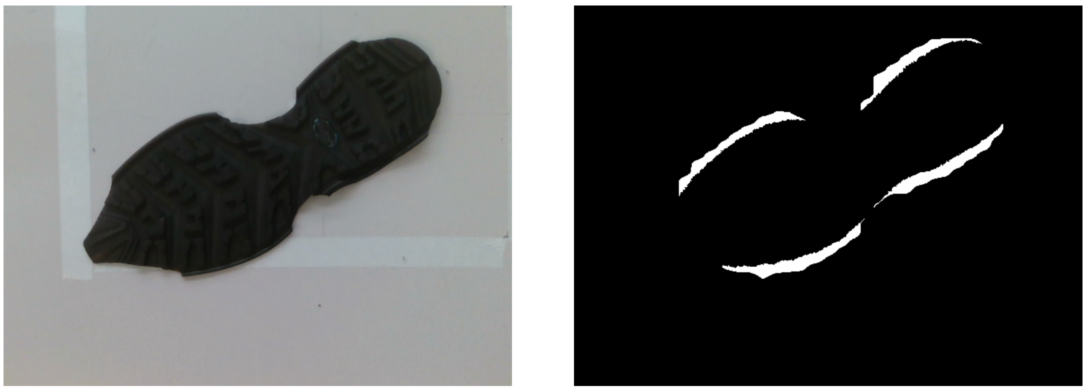

(a) RGB image of the sole with burrs (b) Identified burrs

Figure 6.12: Path planning pipeline: burrs identification

A unique identifier is assigned to each identified burr for tracking purposes. As an illustrative example, the top-left burr is chosen for describing the pipeline.

To determine the deburring path within the plane that the deburring tool should follow, the segmentation of the identified burrs is intersected with the nominal profile. This results in a collection of points constituting the internal part of the burrs' contour, as visually depicted in Figure 6.13.

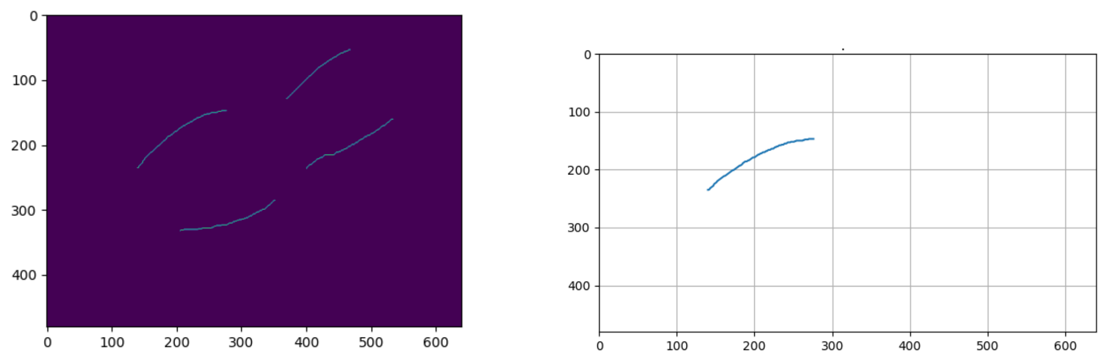

(a) Intersection nominal profile and burrs (b) Tool's trajectory

Figure 6.13: Deburring plane trajectories in image reference system

All the processes related to burrs identification are performed within the acquired image reference frame, denoted as the *(u,v)* frame in Figure 6.5. However, the points in the plane, where the tool should navigate to perform deburring, need to be transformed into the robot base frame. Notably, the vertical distance z from the camera to the sole is known and fixed, as each image is taken in the home position of the robot. The vertical coordinate to be provided to the robot controller, in the robot reference frame, is found using the transformation matrix T_{cam} (Table 6.1).

Two transformations are employed:

- **From image frame to camera frame**: utilizing intrinsic camera parameters (camera center (c_x, c_y) and focal lengths f_x, f_y) and the known depth distance z, the (x, y) coordinates are obtained from (u, v) coordinates through the following formulas:

$$x = \frac{(u - c_x)z}{f_x} \tag{6.1}$$

$$y = \frac{(v - c_y)z}{f_y} \tag{6.2}$$

- **From camera frame to robot frame**: employing the transformation matrix T_{cam} (Table 6.1)

The same transformation is also applied to the identified coordinates of the center of the sole O in the robot's reference frame.

Identification of Burr's Local Reference Frame

Once the trajectory vector (x, y, z) is determined in the robot frame, the local reference frame O' for each point along the identified tool trajectory is found.

Given the typically high number N of elements comprising the trajectory vector, resulting from numerical calculations, a parameter α is introduced to reduce this number. Specifically, a total of $n = N/\alpha$ points are considered. Starting from the initial one, every α-th point is chosen as a control point. These control points are then interpolated through a NURBS curve of degree 2, with unitary weights.

Next, **tangent** and **normal** unit vectors are identified. Numerical derivation is performed using finite differences with a step size of $h = 1 \times 10^{-4}$, utilizing the Python functions *get_ tangent()* and *get_ normal()* from the *nurbspy* library [2]. It is important to note that the normal vectors point outside the sole, the z-axis points upward, and, consequently, the tangents rotate clockwise around the sole profile.

The resulting local frames for the top-left burr are depicted in Figure 6.14, where the tangent vector is represented in red, the normal in blue, and the green dot denotes the calculated center of the sole.

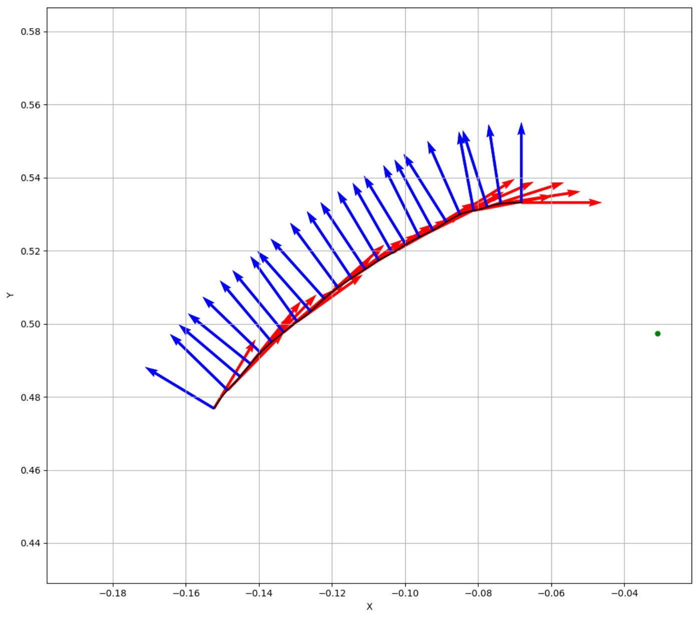


Figure 6.14: Burr's local reference frames

Tool orientation learned from human demonstration

The final element needed to complete the deburring path is the orientation of the tool, which is learned from a video of an expert human demonstrator. As discussed in Chapter 5, the optimal orientation has been acquired through the EfficientPose [19] network. The system aims to learn the transformation **T'** (Table 6.1), representing the orientation of the tool center point (TCP) in relation to the burr's local reference systems. The goal is to obtain a **constant matrix** applicable throughout the entire deburring operation.

For experimental purposes, **three videos** of deburring operations have been recorded, each showcasing the process with the sole in different positions and on different burrs. From each video, **116 frames** have been extracted. Each frame provides an image to be analyzed by the trained EfficientPose model, resulting in the 6D pose of the tool's base frame relative to the camera frame. Figure 6.15 illustrates predicted poses, represented as 3D bounding boxes, from examples extracted from the three videos.

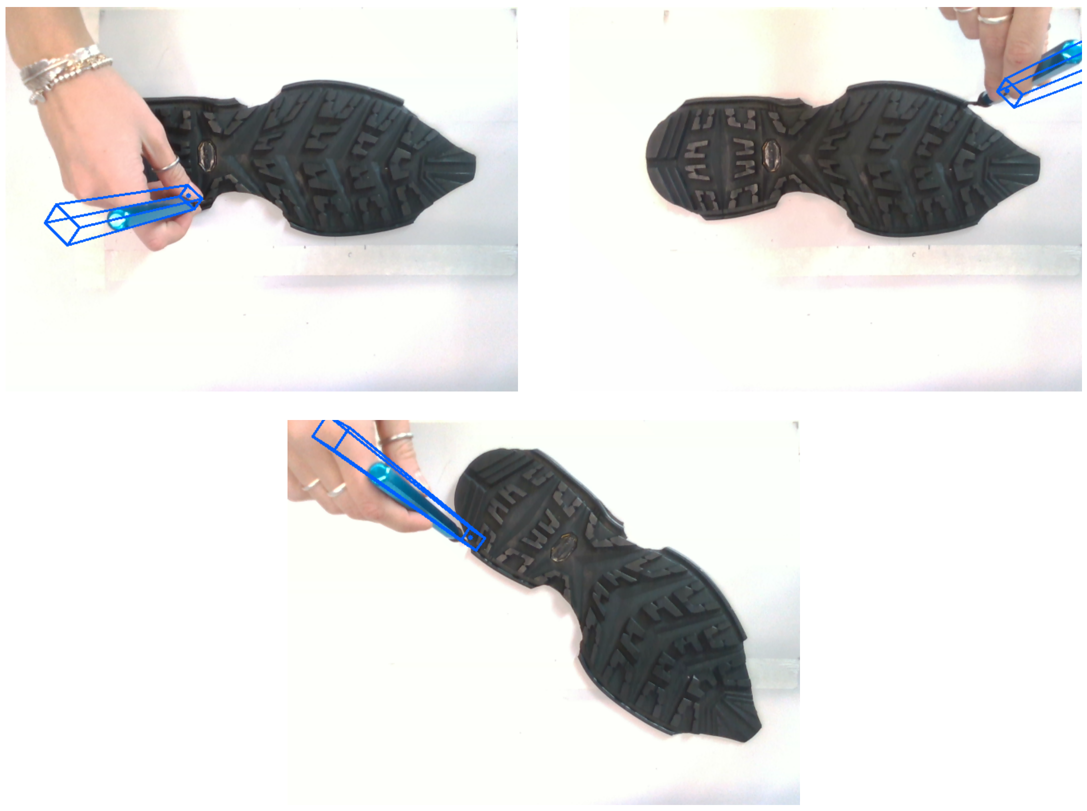


Figure 6.15: Tool pose predictions from expert's demonstration frames

Thanks to the known transformations reported in Table 6.1 the orientation of the TCP with respect to the burr's local reference system (T') has been found for every acquired frame, as schematically illustrated in Figure 6.16. In particular, the applied transformation is:

$$T' = (T^{G}_{i_{th}-burr})^{-1} \cdot (T_{cam} \cdot T_{TCP-camera\_frame}) \tag{6.3}$$

where $T_{TCP-camera\_frame}$ represents the output of EfficientPose (the orientation of the TCP with respect to the camera frame) while $T^{G}_{i_{th}-burr}$ is the orientation of the local burrs reference frame in the global robot reference system.

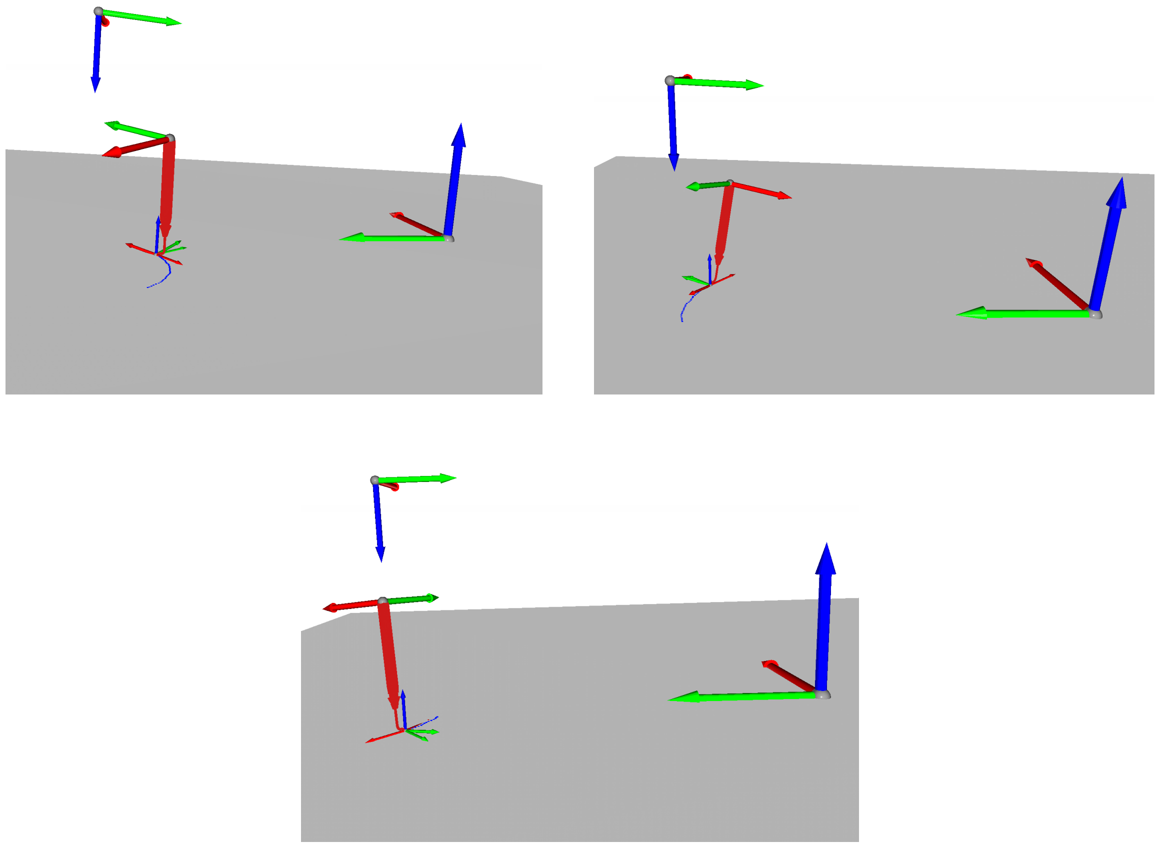


Figure 6.16: Extracted tool pose from video frames in camera and robot reference systems

To obtain the final orientation applied throughout all deburring paths, an **arithmetic mean of the poses** has been calculated. It's important to note that orientations have been transformed into quaternion representation to accurately perform the average. The resulting averaged orientation of the TCP frame in the burr's local reference frame, learned from the demonstrations, is given by the rotation matrix:

$$R_{final} = \begin{bmatrix} -0.986 & 0.021 & -0.167 \\ 0.025 & 0.999 & -0.022 \\ 0.166 & -0.026 & -0.986 \end{bmatrix} \tag{6.4}$$

This corresponds to Euler angles (XYZ form, in degrees) $[178.70\ -9.61\ -178.77]$.

With this, the path planning is complete, and the robot has a collection of points in space to follow, along with the local orientation to use at every point of the trajectory based on the calculated burr's local reference frame.

Imposed deburring direction

The final operation involves determining the direction of deburring that the robot should follow to perform the task consistently across all burrs. The chosen approach is to cut the burrs following a **clockwise** path. This is achieved through the following steps:

1. Utilize the OpenCV function *minAreaRect()* to determine the oriented bounding box of the sole with burrs, and subsequently, extract the center line along its longest edge.
2. Calculate the enclosing circle of each burr using the OpenCV function *minEnclosingCircle()*.
3. Classify each burr as either *"above"* or *"below"* the center line:
   (a) If the y coordinate of the circle's center is *less than* the value obtained by substituting the x coordinate of the circle's center into the equation of the center line, categorize the burr as *"above"* (Note: This process is performed in the image reference system, where the vertical axis increases from the top left corner to the bottom left corner).
   (b) If the y coordinate of the circle's center is *higher than* the value obtained by substituting the x coordinate of the circle's center into the equation of the center line, the burr is categorized as *"below"*.
4. For each burr, calculate the variances of the points along the x and y coordinates. If the variance along x is higher than along y, the burr evolves mainly along the horizontal direction. If the variance along y is higher, the burr evolves mainly along the vertical direction;

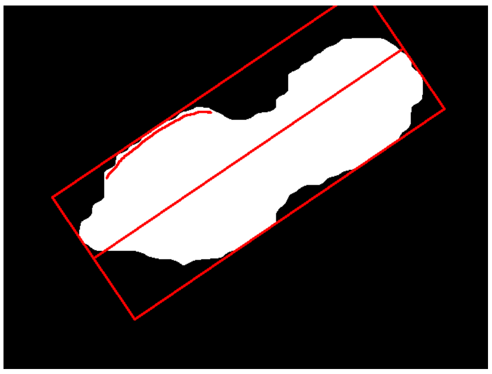
5. If a burr is *"above"*, perform the trajectory as follows:
   (a) If it evolves horizontally ($variance_x > variance_y$), the trajectory goes from the point with the lowest x to the one with the highest.
   (b) If it evolves vertically ($variance_y > variance_x$), the trajectory goes from the point with the lowest y to the one with the highest.
6. If a burr is *"below"*, perform the trajectory as follows:
   (a) If it evolves horizontally ($variance_x > variance_y$), the trajectory goes from the point with the highest x to the one with the lowest.
   (b) If it evolves vertically ($variance_y > variance_x$), the trajectory goes from the point with the highest $y$to the one with the lowest.

This ensures that the deburring direction is consistently clockwise for every considered burr.

For instance, for the top left burr considered as an example, depicted in red in Figure 6.17a, the burr is classified as *"above"* the center line. The calculated variances along x and y are

$$\begin{aligned} variance_x &= 1677.48\ mm^2 \\ variance_y &= 728.64\ mm^2 \end{aligned} \tag{6.5}$$

Since the x coordinate varies more ($variance_x > variance_y$), the deburring path is performed from the point with the lowest x value to the one with the highest, as visually represented in Figure 6.17b.


(a) Identified burr *"above"* the centerline

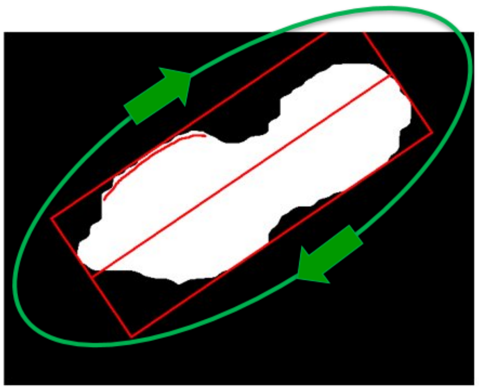

(b) Imposed clockwise deburring direction

Figure 6.17

6.2.3. Results

To validate the proposed approach, multiple real-world robotic deburring tests have been conducted in the Mechatronics and Robotics lab for Innovation (*MERLIN* [4]) of Politecnico di Milano University using the setup depicted in Figure 6.4.

To facilitate communication with the robot and execute the calculated trajectory, the Robot Operating System (ROS) [73] platform has been employed. ROS is a framework for writing robot software, and it provides tools and libraries for tasks such as hardware abstraction, device drivers, communication between processes, and package management. The *movesx()* function, integrated into the ROS framework for Doosan robots [5], has been used to execute the trajectory based on the calculated set of points. The robot moves along a spline curve path that connects the current position to the target position via the list of provided points of the task space given as input. This function enables smooth interpolation between waypoints, ensuring precise and controlled robotic movements during the deburring process.

To ensure the robustness and correctness of the planned trajectory, the behavior of the robot has been initially evaluated in a simulated environment using ROS and Gazebo [3] (Figure 6.18). Gazebo is a powerful robotics simulator that allows for testing and validating robotic algorithms in a virtual environment before deploying them to the physical robot. This simulated testing phase serves as a step to ensure the reliability and safety of the deburring operations before transitioning to real-world experiments.

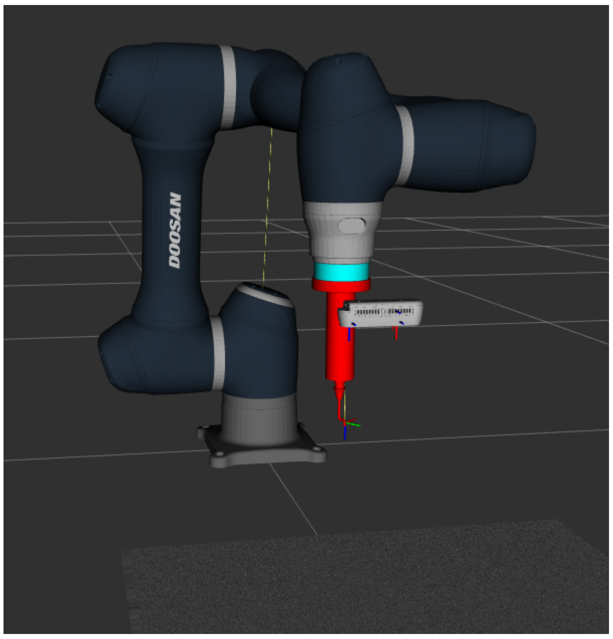


Figure 6.18: Simulation environment with Gazebo

The experimental validation aimed to quantitatively assess the accuracy of the computed deburring path by comparing it with the path demonstrated by an expert who **physically manipulated the robot** (Figure 6.19) to perform the deburring task. Two tests have been conducted, with the sole positioned randomly on the working plane. In each test, the trajectory for cutting four burrs has been demonstrated and recorded.

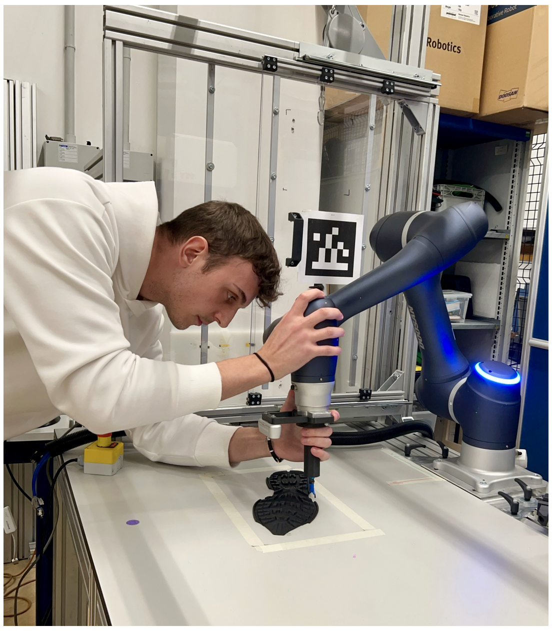


Figure 6.19: Direct demonstration for experimental validation

Due to the challenges associated with accurately positioning the robot during physical movement, only the in-plane trajectory (x, y) has been considered for comparison. Specifically, the average of the Euclidean distances between the calculated points from the path planning pipeline and the corresponding points measured from the expert demonstration has been computed as an evaluation metric.

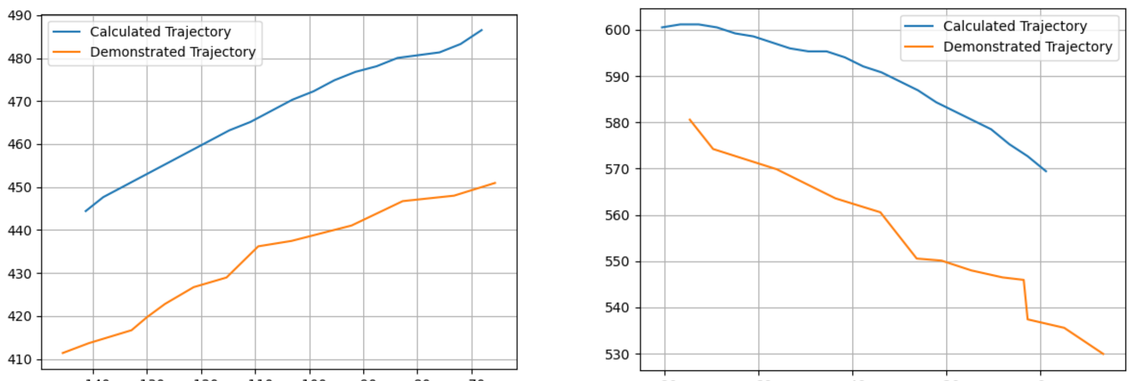
The results are reported in Tables 6.2 and 6.3 and examples of the compared trajectories are displayed in Figure 6.20.

	Average distance in mm
Burr 1	8.7
Burr 2	27.1
Burr 3	5.9
Burr 4	35.9
Mean distance	19.4

Table 6.2: First sole experimental validation

	Average distance in mm
Burr 1	5.2
Burr 2	7.1
Burr 3	11.5
Burr 4	24.7
Mean distance	12.13

Table 6.3: Second sole experimental validation



(a) (b)

Figure 6.20: Examples of calculated vs demonstrated trajectories

The average calculated distance is 15.8mm. This disparity is attributed to the accumulated errors from previous steps, as well as variations in the accuracy of the calculated matrices during reference system transformations. Additionally, the precise demonstration of the path by physically manipulating the robotic arm presented challenges in ensuring high precision. It is crucial to note that the primary goal of this study is to establish an initial deburring path. In practical industrial scenarios, a control algorithm is essential to compensate for these inherent errors and uncertainties. Despite the observed differences, the obtained result has been deemed highly acceptable within the context of the study's objectives.

7 | Conclusions and future developments

This chapter provides a comprehensive overview of the main accomplishments in the study and outlines potential directions for future development.

The thesis tackles the task of robotic deburring by introducing a complete path planning pipeline that integrates learning from demonstration, thus mitigating the need for difficult and specialized coding. The solutions proposed include: sole detection and segmentation, eventually reconstructing the occluded portions of the profile, burrs identification to identify the precise segmentation of the regions of material to be removed, and tool orientation learning from expert demonstrators' videos

The task of identifying and segmenting the sole has been effectively addressed through the implementation of Detectron-2 [80], an advanced open-source object detection network developed by Facebook AI Research. This network, based on Mask Region Convolutional Neural Network (Mask R-CNN) [36], has been trained on a custom-made dataset. The dataset comprises RGB images featuring soles, both with and without burrs, presented in various orientations against different backgrounds. Manual annotations have been applied to specify the class ("sole"), bounding box, and mask for training purposes. The model has demonstrated exceptional accuracy in recognizing soles and accurately outlining their masks under diverse conditions, showcasing robust performance regardless of lighting conditions, background variations, or orientation within the sole's plane.

In real-world scenarios, the gripping points during the grasping of the sole can lead to occlusions in the profile, affecting the precision of the burrs identification step. To address this challenge, a decision has been made to reconstruct the occluded parts of the identified profiles containing burrs. Image-to-image translation has been employed, specifically utilizing Pix2Pix [38], a conditional Generative Adversarial Network (cGAN). The goal is to reconstruct the profile with burrs, regardless of the shape of the burrs, in instances where occlusions are present. A custom dataset has then been, featuring ten distinct profiles of soles with burrs, each subjected to randomly generated occlusions. The trained model effectively reconstructs the occluded profiles presented in the validation dataset. However, challenges arise when posed in series to Detectron-2, due to irregularities in the segmentation obtained from the object detection network. The model struggles to reconstruct occluded portions from this segmentation, misidentifying undulated areas as occlusions and attempting to reconstruct them. Future works should focus on refining the obtained profile from Detectron-2, smoothing irregularities specifically in areas with occlusions, to enhance the accuracy of the reconstruction process.

The burrs identification process follows the achievement of the segmentation of the sole with burrs. A customized method has been developed, leveraging image processing techniques. Specifically, a template of the nominal profile has been extracted from an image of the sole without burrs. This template undergoes scaling, rotation, and translation to align the oriented bounding boxes of the sole with burrs and the nominal profile. Subsequently, an optimization procedure is executed to precisely determine the overlap of the nominal profile within the one with burrs. The optimization method employs a direct search approach, adaptively changing both the search space and steps to speed up convergence and achieve an accurate final result. The segmentation of the burrs is derived by pixel-wise subtraction of the mask of the nominal sole from the one with burrs. Following this, a smoothing operation is applied to eliminate small, non-physical identified burrs.

The optimal orientation for the deburring tool has been acquired through a learning-from-demonstration approach. Expert demonstrations in the form of videos have been obtained, and the tool's pose has been extracted using EfficientPose [19]. This network is a well-established literature model for 6D pose estimation from RGB images, chosen for its precision and ease of use. The training was conducted on a customized dataset, generating ground truths for pose estimation through synthetic data. A virtual scene has been crafted, placing the 3D model of the tool in a defined position relative to the camera. To enhance realism and generalization of the results to real-world scenarios, backgrounds featuring soles and hands have been incorporated and variability in the position of the tool and lighting have been carried out. The experiments have demonstrated the network's capability to identify the tool's pose even when held by a demonstrator in real-world videos, showcasing the successful transfer of knowledge from synthetic to real-world data.

Finally, the outlined processes result in the creation of an automated pipeline. This system begins with a sole with burrs positioned anywhere within the operational space of the robotic arm, and visible by the camera mounted on the robot's end effector. Subsequently, it autonomously computes the entire deburring path to be executed. In the practical execution of these experiments, a new robotic deburring tool has been designed. This tool includes a base designed for attachment to the robot that incorporates the manual deburring tool.

Future advancements comprise the incorporation of a force control algorithm to manage the actual execution of the path and deburring of the sole. This addition aims to refine the accuracy of the path derived by the pipeline, effectively addressing any errors and uncertainties inherent in the robotic deburring process. Furthermore, the application of Detectron-2 enables the classification of various types of soles, including distinguishing between left and right soles and different model identification. This flexibility allows for the creation and adjustments of the pipeline to accommodate these variations.

Bibliography

[1] Ati automation: Compliant deburring blade. URL https://www.ati-ia.com/products/deburr/deburring_cdb_main.aspx.

[2] Nurbspy python library. URL https://github.com/RoberAgro/nurbspy.

[3] Gazebo simulator. URL https://github.com/gazebosim.

[4] Merlin: mechatronics and robotics lab of politecnico di milano. URL http://merlin.deib.polimi.it/.

[5] Doosan robot: Ros programming manual. URL http://wiki.ros.org/doosan-robotics?action=AttachFile&do=get&target=Doosan_Robotics_ROS_Manual_ver0.971_20200218A%28EN.%29.pdf.

[6] How shoes are made – the history of shoemaking, 2020 - August. URL https://www.solescience.ca/history-of-shoemaking/.

[7] How a charge coupled devide (ccd) image sensor works, 2023 - October. URL https://www.teledyneimaging.com/media/1300/2020-01-22_e2v_how-a-charge-coupled-device-works_web.pdf.

[8] Coco detection metric, 2023 - October. URL https://cocodataset.org/#detection-eval.

[9] Opencv findcontours() function documentation, 2023 - October. URL https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a.

[10] Opencv minarearect() function documentation, 2023 - October. URL https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga3d476a3417130ae5154aea421ca7ead9.

[11] Keras reducelronplateau() function documentation, 2023 - October. URL https://keras.io/api/callbacks/reduce_lr_on_plateau/.

[12] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes

for kinesthetic teaching: A human-robot interaction perspective. In *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 391–398, 2012. doi: 10.1145/2157689.2157815.

[13] L. Alatabani, E. Sayed Ali Ahmed, and R. Saeed. *Machine Learning and Deep Learning Approaches for Robotics Applications*, pages 303–333. 05 2023. ISBN 978-3-031-28714-5. doi: 10.1007/978-3-031-28715-2_10.

[14] J. Aurich, D. Dornfeld, P. Arrazola, V. Franke, L. Leitz, and S. Min. Burrs—analysis, control and removal. *CIRP Annals*, 58(2):519–542, 2009. ISSN 0007-8506. doi: https://doi.org/10.1016/j.cirp.2009.09.004.

[15] J. Aurich, D. Dornfeld, P. Arrazola, V. Franke, L. Leitz, and S. Min. Burrs—analysis, control and removal. *CIRP Annals*, 58(2):519–542, 2009. ISSN 0007-8506. doi: https://doi.org/10.1016/j.cirp.2009.09.004.

[16] E. Bahce and B. Özdemiř. Burr measurement method based on burr surface area. *International Journal of Precision Engineering and Manufacturing-Green Technology*, 8, 06 2020. doi: 10.1007/s40684-020-00228-0.

[17] E. Brachmann. 6D Object Pose Estimation using 3D Object Coordinates [Data]. 2020. doi: 10.11588/data/V4MUMX.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] Y. Bukschat and M. Vetter. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach, 2020.

[20] S. Calinon. *Learning from Demonstration (Programming by Demonstration)*, pages 1–8. Springer Berlin Heidelberg, Berlin, Heidelberg, 2018. ISBN 978-3-642-41610-1. doi: 10.1007/978-3-642-41610-1_27-1.

[21] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar. Learning collaborative manipulation tasks by demonstration using a haptic interface. pages 1 – 6, 07 2009.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[23] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access*, PP:1–1, 12 2017. doi: 10.1109/ACCESS.2017.2783682.

[24] X. Chen, G. Shi, C. Xi, L. Zhong, X. Wei, and K. Zhang. Design of burr detection based on image processing. *Journal of Physics: Conference Series*, 1237(3):032075, jun 2019. doi: 10.1088/1742-6596/1237/3/032075.

[25] A. Colome, A. Planells, and C. Torras. A friction-model-based framework for reinforcement learning of robotic tasks in non-rigid environments. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015:5649–5654, 06 2015. doi: 10.1109/ICRA.2015.7139990.

[26] A. Correia and L. A. Alexandre. A survey of demonstration learning, 2023.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, sep 1995. ISSN 0885-6125. doi: 10.1023/A:1022627411411.

[28] R. Dastres and M. Soori. Artificial Neural Network Systems. *International Journal of Imaging and Robotics (IJIR)*, 21(2):13–25, Sept. 2021.

[29] M. Denninger, D. Winkelbauer, M. Sundermeyer, W. Boerdijk, M. Knauer, K. H. Strobl, M. Humt, and R. Triebel. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software*, 8(82):4901, 2023. doi: 10.21105/joss.04901. URL `https://doi.org/10.21105/joss.04901`.

[30] K. Fu and J. Mui. A survey on image segmentation. *Pattern Recognition*, 13(1):3–16, 1981. ISSN 0031-3203. doi: https://doi.org/10.1016/0031-3203(81)90028-5.

[31] R. Girshick. Fast r-cnn, 2015.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi: 10.1109/CVPR.2014.81.

[33] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

[34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *arXiv e-prints*, art. arXiv:1703.06870, Mar. 2017. doi: 10.48550/arXiv.1703.06870.

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2018.

[37] C. Huang, D. Chen, and X. Tang. Implementation of workpiece recognition and location based on opencv. In *2015 8th International Symposium on Computational*

Intelligence and Design (ISCID), volume 2, pages 228–232, 2015. doi: 10.1109/ISCID.2015.143.

[38] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. doi: 10.1109/CVPR.2017.632.

[39] Z. Ji, L. Peigen, Y. Zhou, B. Wang, Z. Jiyuan, and M. Liu. Toward new-generation intelligent manufacturing. *Engineering*, 4:11–20, 04 2018. doi: 10.1016/j.eng.2018.01.002.

[40] B.-H. Kang, J.-D. Kim, J.-Y. Yoo, J.-O. Park, K.-S. Lee, and H.-O. Shin. Robot intelligence and flexibility for smooth finishing of car body. *IFAC Proceedings Volumes*, 30 (14):303–311, 1997. ISSN 1474-6670. doi: https://doi.org/10.1016/S1474-6670(17)42739-X. IFAC Workshop on Intelligent Manufacturing Systems (IMS'97), Seoul, Korea, 21-23 July 1997.

[41] M.-G. Kim, J. Kim, S. Y. Chung, M. Jin, and M. J. Hwang. Robot-based automation for upper and sole manufacturing in shoe production. *Machines*, 10(4), 2022. ISSN 2075-1702. doi: 10.3390/machines10040255.

[42] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[43] H. Kosler, U. Pavlovčič, M. Jezeršek, and J. Mozina. Adaptive robotic deburring of die-cast parts with position and orientation measurements using a 3d laser-triangulation sensor. *Strojniški vestnik - Journal of Mechanical Engineering*, 62:207, 04 2016. doi: 10.5545/sv-jme.2015.3227.

[44] Z. Lai, R. Xiong, H. Wu, and Y. Guan. Integration of visual information and robot offline programming system for improving automatic deburring process. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1132–1137, 2018. doi: 10.1109/ROBIO.2018.8665148.

[45] K. C. Lee, H.-P. Huang, and S.-S. Lu. Burr detection by using vision image. *The International Journal of Advanced Manufacturing Technology*, 8:275–284, 1993. URL `https://api.semanticscholar.org/CorpusID:58944046`.

[46] Y. D. Lee, B. H. Kang, and J. O. Park. Robotic deburring strategy using burr shape recognition. 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)*, volume 3, pages 1513–1518 vol.3, 1999. doi: 10.1109/IROS.1999.811693.

[47] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi: 10.1109/CVPR.2017.106.

[48] R. M R and S. Chiddarwar. A causality-inspired data augmentation approach to cross-domain burr detection using randomly weighted shallow networks. *International Journal of Machine Learning and Cybernetics*, 14, 06 2023. doi: 10.1007/s13042-023-01891-w.

[49] M. Mirza and S. Osindero. Conditional generative adversarial nets. 11 2014.

[50] A. Mohammed, J. Kvam, I. F. Onstein, M. Bakken, and H. Schulerud. Automated 3d burr detection in cast manufacturing using sparse convolutional neural networks. *Journal of Intelligent Manufacturing*, 34(1):303–314, 2023.

[51] Y. Nakao and Y. Watanabe. Measurements and evaluations of drilling burr profile. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 220(4):513–523, 2006. doi: 10.1243/095440505X32625.

[52] A. Nguyen, D. Kanoulas, L. Muratore, D. Caldwell, and N. Tsagarakis. Translating videos to commands for robotic manipulation with deep recurrent neural networks. pages 1–9, 05 2018. doi: 10.1109/ICRA.2018.8460857.

[53] I. F. Onstein, O. Semeniuta, and M. Bjerkeng. Deburring using robot manipulators: A review. In *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, pages 1–7, 2020. doi: 10.1109/SIMS49386.2020.9121490.

[54] I. F. Onstein, O. Semeniuta, and M. Bjerkeng. Deburring using robot manipulators: A review. In *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, pages 1–7, 2020. doi: 10.1109/SIMS49386.2020.9121490.

[55] I. F. Onstein, C. Haskins, and O. Semeniuta. Cascading trade-off studies for robotic deburring systems. *Systems Engineering*, 25(5):475–488, 2022. doi: https://doi.org/10.1002/sys.21625.

[56] I. F. Onstein, M. Bjerkeng, and K. Martinsen. Automated tool trajectory generation for robotized deburring of cast parts based on 3d scans. *Procedia CIRP*, 118:507–512, 2023. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2023.06.087. 16th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

[57] K. O'Shea and R. Nash. An introduction to convolutional neural networks, 2015.

[58] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi: 10.1109/TSMC.1979.4310076.

[59] S. Pan and X. Wang. A survey on perspective-n-point problem. In *2021 40th Chinese Control Conference (CCC)*, pages 2396–2401, 2021. doi: 10.23919/CCC52363.2021.9549863.

[60] F. L. Princely and T. Selvaraj. Vision assisted robotic deburring of edge burrs in cast parts. *Procedia Engineering*, 97:1906–1914, 2014. doi: https://doi.org/10.1016/j.proeng.2014.12.344. "12th Global Congress on Manufacturing and Management" GCMM - 2014.

[61] A. Rani, D. Ortiz Arroyo, and P. Durdevic. Defect detection in synthetic fibre ropes using detectron2 framework. *I E E E Sensors Journal*, 2023. ISSN 1530-437X.

[62] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[63] P. Sahoo, S. Soltani, and A. Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41:233–260, 02 1988. doi: 10.1016/0734-189X(88)90022-9.

[64] K. A. M. Said and A. B. Jambek. Analysis of image processing using morphological erosion and dilation. *Journal of Physics: Conference Series*, 2071(1):012033, oct 2021. doi: 10.1088/1742-6596/2071/1/012033. URL `https://dx.doi.org/10.1088/1742-6596/2071/1/012033`.

[65] A. Salari, A. Djavadifar, X. Liu, and H. Najjaran. Object recognition datasets and challenges: A review. *Neurocomputing*, 495:129–152, 2022. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.01.022.

[66] I. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2, 08 2021. doi: 10.1007/s42979-021-00815-1.

[67] P. J. SCHNEIDER and D. H. EBERLY. Chapter 11 - intersection in 3d. In P. J. SCHNEIDER and D. H. EBERLY, editors, *Geometric Tools for Computer Graphics*, The Morgan Kaufmann Series in Computer Graphics, pages 481–662. Morgan Kaufmann, San Francisco, 2003. ISBN 978-1-55860-594-7. doi: https://doi.org/10.1016/B978-155860594-7/50014-X.

[68] K. Shimokura and S. Liu. Programming deburring robots based on human demon-

stration with direct burr size measurement. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 572–577 vol.1, 1994. doi: 10.1109/ROBOT.1994.351238.

[69] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos, 2020.

[70] B. Solvang, G. Sziebig, and P. Korondi. Vision based robot programming. pages 949 – 954, 05 2008. doi: 10.1109/ICNSC.2008.4525353.

[71] H.-C. Song and J.-B. Song. Precision robotic deburring based on force control for arbitrarily shaped workpiece using cad model matching. *International Journal of Precision Engineering and Manufacturing*, 14, 01 2012. doi: 10.1007/s12541-013-0013-2.

[72] R. Srisha and A. Khan. Morphological operations for image processing : Understanding and its applications. 12 2013.

[73] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. URL `https://www.ros.org`.

[74] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020. doi: 10.1109/CVPR42600.2020.01079.

[75] D.-M. TSAI‡ and W.-J. Lu. Detecting and locating burrs of industrial parts. *International Journal of Production Research*, 34(11):3187–3205, 1996. doi: 10.1080/00207549608905084.

[76] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013. URL `https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013`.

[77] A. Verl, A. Valente, S. Melkote, C. Brecher, E. Ozturk, and L. T. Tunc. Robots in machining. *CIRP Annals*, 68(2):799–822, 2019. ISSN 0007-8506. doi: https://doi.org/10.1016/j.cirp.2019.05.009.

[78] E. Villagrossi, C. Cenati, N. Pedrocchi, M. Beschi, and L. Molinari Tosatti. Flexible robot-based cast iron deburring cell for small batch production using single-point laser sensor. *The International Journal of Advanced Manufacturing Technology*, 92: 1–14, 09 2017. doi: 10.1007/s00170-017-0232-2.

[79] B. Wang, F. Tao, X. Fang, C. Liu, Y. Liu, and T. Freiheit. Smart manufacturing and

intelligent manufacturing: A comparative review. *Engineering*, 7(6):738–757, 2021. ISSN 2095-8099. doi: https://doi.org/10.1016/j.eng.2020.07.017.

[80] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[81] C. Wulf and M. Hayk. Automatic residual burr detection on steel slabs by a thermographic system. *Stahl und Eisen*, 30:36–37, 04 2007.

[82] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks, 2017.

[83] Z. Xie, Q. Zhang, Z. Jiang, and H. Liu. Robot learning from demonstration for path planning: A review. *Science China Technological Sciences*, 63, 07 2020. doi: 10.1007/s11431-020-1648-4.

[84] H. Zhang, H. Chen, N. Xi, G. Zhang, and J. He. On-line path generation for robotic deburring of cast aluminum wheels. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2400–2405, 2006. doi: 10.1109/IROS.2006.281679.

[85] Z. Zhang. *Iterative Closest Point (ICP)*, pages 433–434. Springer US, Boston, MA, 2014. ISBN 978-0-387-31439-6. doi: 10.1007/978-0-387-31439-6_179.

List of Figures

List of Tables