Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Bachelor thesis

A computer vision based pick and place solution for 3D printers

at the Research Group Technical Aspects of Multimodal Systems, TAMS
Department of Informatics
MIN Faculty
University of Hamburg

Felix Kolwa

3kolwa@informatik.uni-hamburg.de
Studiengang Informatik
Matr.-Nr. 6026927
Fachsemester 12

Supervisors:
1. Dr. Mikko Lauri
2. Florens Wasserfall

Abstract

English
Pick and place machines are a standard in the industrial assembling process. Computer vision based analysis provides the backbone of automated assembly line tasks and has proven valuable in quality control or production [Ulrich et al., 2003]. 3D printers make affordable, small form factor manufacturing processes possible for individuals and small enterprises.The integration of electronic circuits into 3D printed objects has proven to be valuable for creating otherwise time consuming prototypes as well as wearable electronics, robotics or biomedical applications [Valentine et al., 2017] [Macdonald et al., 2014].Even though pick and place machines and 3D printers have been well established in the industry [Ulrich et al., 2003] and attempts at integrating electronics into 3D printed material have been well documented, there is still a lack of established hard- and software solutions to combine those two [Wasserfall, 2015]. Adding electronic components to the 3d printed objects require additional manual assembly [Wasserfall, 2015] which makes the production of completely enclosed electronics complicated and time consuming.This bachelor thesis is based on the pick and place solution that is part of the OctoPNP [OctoPnP, 2019] plugin for the OctoPrint [Octoprint, 2019] 3D printer software which uses computer vision aided guidance to pick and place electronic components as integrated part of the 3D printing process. The existing process will be analysed while also introducing a modified version which utilizes template matching in the form of the Generalized Hough Transform to find component position and orientation in raw image data. Both pipelines will be evaluated and analysed based on these results.

German

Pick and Place Maschinen bilden einen industriellen Standart in modernen produktions Prozessen, welche durch die automatisierte Analyse von Computer Vision Software unterstützt werden [Ulrich et al., 2003]. 3D Drucker bieten eine kostengünstige alternative zu industriellen Produktionsstätten für Privatpersonen und kleinen Unternehmen. Die Integration von elektronischen Schaltkreisen in 3D gedruckte Objekte hat sich sowohl für das schnelle und unkomplizierte Erstellen von Prototypen als auch für Anwendungen im Bereich der Robotik, wearable electronics oder Medizin als sinnvoll erwiesen [Valentine et al., 2017] [Macdonald et al., 2014]. Obwohl 3D Drucker und Pick and Place Maschinen weit verbreitet sind [Ulrich et al., 2003] und eine Reihe von gut dokumentierten Ansätzen von 3D druckbarer elektronischer Schaltkreise existieren, gibt es keinen etablierten Standart der beide Maschinen miteinander verbindet [Wasserfall, 2015]. Elektronische Bauteile in 3D gedruckte Objekte zu integrieren bedingt manuelle Schritte, welche den Prozess komplex und zeitaufwendig gestallten [Wasserfall, 2015]. Die vorliegende Bachelor Arbeit basiert auf dem OctoPrint [Octoprint, 2019] Plugin OctoPNP [OctoPnP, 2019], welches das Integrieren von elektronischen Bauteilen durch einen von Computer Vision gestützten Prozess realisiert, welcher direkt zur Laufzeit des 3D Druckvorganges ausgeführt wird. Der Pick and Place Prozess wird im Folgenden analysiert und eine modifizierte Lösung vorgestellt die Template Matching, in Form des Generalized Hough Transform, verwendet um Position und Rotation von Bauteilen in Bilddateien zu erkennen. Abschließend werden beide Ansätze getestet und miteinander verglichen.

Contents

1 Introduction

This chapter gives a quick overview of the structure and content of the bachelor thesis at hand. It also provides background information about the current state of 3D printable electronics with a focus on the pick and place assembly of integrated circuits in 3D printed objects.

1.1 Motivation

Pick and place machines describe robotic devices that are utilized to automatically take and set objects or working material without the need of human interaction on either material or machine [Li et al., 2017].

A special type of pick and place machine is the surface mount technology (SMT) component placement system which is capable of assembling surface mount device (SMD) components onto provided printed circuit boards with both precision and speed.
To ensure the right placement and object selection these machines rely on computer vision based software which processes live image data and extracts the necessary information [Li et al., 2017].

Through the advent of affordable CNC based 3D printers and the increased interest in small form factor manufacturing, open source projects like OpenPnP [OpenPnP, 2019] become more popular.

While Additive Manufacturing (AM) has been used to produce prototypes and validate design and assembly in the final steps of product creation, it has not been able to produce functioning models due to the limitations of the AM process [Macdonald et al., 2014]. To create prototypes that validate not only the form but also the functionality, additional steps had to be performed in which electronic circuits are assembled and installed by hand which is not only time consuming but also adds to the processes complexity through manual integration and debugging [Macdonald et al., 2014] [Wasserfall, 2015].

Through enhanced AM technologies such as stereolithography (SL) or fused deposition modeling (FDM) with the addition of embedded conductive material and automated placement of electronic components this gap is beginning to close and the production of fully functional 3D printed objects can be realized [Macdonald et al., 2014].

Even though most pick and place machines share similarities with popular 3 axis CNC based 3D printers, approaches to unify these two devices have been scarce. This is for once due to the added complexity and cost of adding extra hardware to the printer but also the lack of established soft- or hardware systems that incorporate all the necessary features.
One approach is the modified Kühling&Kühling Reprap 3D printer currently stationed at the 3D Printing lab at the Informatikum of the University of Hamburg.

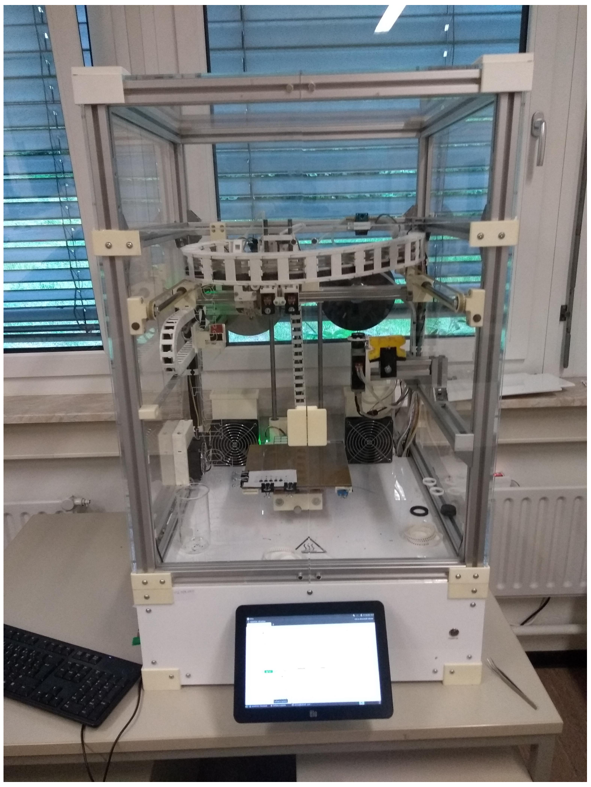


Figure 1.1: The modified Kühling&Kühling 3D printer this thesis is based on.

1.2 Methods and resources

Hardware

Hardware wise the thesis will be based on a Kühling&Kühling RepRap Industrial printer (see figure 1.1) which was modified and documented by Florens Wasserfall [Wasserfall, 2015] to support conductive printing as well as the integration of SMD components into the print through the before mentioned pick and place mechanism. The modified 3D printer uses FDM technology, sometimes also referred to as fused filament fabrication or filament freeform fabrication (FFF), to create objects from continues filament based on thermoplastic material. This material is usually provided on a coil and mechanically fed by stepper motors through a heated printing nozzle known as the hotend. During the printing process the machine lays down single horizontal layers of melted material at a time. After a single layer is completed the Z distance from the printbed is increased to print the next layer on top of the previous.

Print interface

The pick and place software will be based on OctoPnP [OctoPnP, 2019] a utility plugin for the 3D printer web interface OctoPrint [Octoprint, 2019]. All changes described in the thesis will affect the source of this plugin while staying compatible to the original OctoPrint software.

Slicer
The popular Slic3r software [Slic3r, 2019], which was modified and documented by Daniel Ahlers [Ahlers, 2015], will be used to convert 3D CAD files into machine readable GCode, place SMD components and route conductive traces into the 3D design files.

Computer vision
The backbone of the image processing will be handled by OpenCV a open source computer vision library. In addition to the detection of the component and the extraction of its physical shape and placement the suggested pipeline also includes an extra step "Shape detection" in which markings that indicate the components orientation are detected and analyzed.

Evaluation
To compare the changes that will be introduced to the current codebase and measure their rate of success a printable torture test will be developed that acts as a comparable template for the quality of the resulting prints.

1.3 Objective

The objective is to analyse and improve the pick and place solution for SMD components in 3D printers originally introduced in [Wasserfall, 2015]. This includes the analysis of its current state as well as other established practices for realising pick and place processes to enhance precision and reliability. The resulting software project is a working pick and place utility that can be embedded into the OctoPrint environment and is currently hosted at [Thesis Source, 2019].

The software is capable of extracting two dimensional position and rotation values of SMD components from search images and converts them into three dimensional cartesian coordinates that can be sent to the to the OctoPrint print server through G-Code hooks. These coordinates are used to align, grip and release SMD components using a vacuum pump attached to the printhead.

The strived for precision is to properly pick, align and place components as small as the 3216 (metric) package code, 100 ohm SMD resistor with dimensions of 3.2 mm length, 1.6 mm width and 0.5 mm height.

All modification to the original source code that is part of this thesis uses the same GNU Affero General Public License v3.0 [Open Source Initiative, 2019] that the original OctoPnP source is released under.

1.4 Contribution

The contribution of this bachelor thesis is the analysis of impact that extended computer vision techniques have on a the image processing process used for detecting SMD components in search images.

The new image processing workflow developed as part of this thesis is described in chapter 4. This solution uses a modified computer vision pipeline for extracting the position and orientation of electronic components with emphasis on implementing Generalized Hough Transform for template matching.

Template matching is used to reduce the amount of misplaced components by providing a more robust way of finding the components shape inside image data that contains otherwise distracting details in the background that are not part of the component and thus obfuscate the detected component shape. Generalized Hough Transform has proven valuable when dealing with slightly deformed shapes in search images which increases reliability of detected position and orientation. A detailed description of this process can be found in chapter 4.5.

1.5 Outline

The following chapter 3 will describe and analyse the original computer vision pipeline that was already running on the Kühling&Kühling 3D printer as part of the OctoPNP [OctoPnP, 2019] plugin. The main focus of this chapter is the analysis of the individual steps the computer vision pipeline takes to read raw image data and extract the component position and orientation. The hardware modifications of the 3D printer which enable printing integrated circuits and automatically place components are discussed here in more detail.

Chapter 4 introduces the modified computer vision pipeline that was developed as part of this thesis on basis of the pipeline described in chapter 3. All changes to the original pipeline are discussed and put into context. This chapter also explains the theory behind using Generalized Hough Transform for template matching and how it is used to retrieve component position and orientation from search images.

The next chapter 5 explains the testing procedure that was used to evaluate the performance of both pipelines. The test results are then used to analyse individual causes and solutions for reoccurring problems.

Finally, chapter 6 concludes the work described in the preceding chapters and provides further ideas for continuing work.

2 Related work

This chapter takes a closer look at existing work on 3D printable electronics, pick and place machines and computer vision solutions for image processing with a focus on SMT placement machines.

2.1 3D printing electronics

The ability to integrate electronic circuits into 3D printed objects has been subject of interest for researchers predating the advent of most customer grade additive manufacturing machines as shown by Kruth et al. in 1998 [Kruth et al., 1998]. To give functionality to the electronic parts, conductive routes have to be added to the 3D printed object. Direct writing (DW) of conductive material enables 3D printers to print these conductive routes into the object on a layer by layer basis in the same way that most 3D printers operate on thermoplastics [Espalin et al., 2014].

A common way of applying conductive material onto the printed object is by using an extruder system similar to the one used in FDM printers to lay down layers of molten filament. The conductive material has to be in a state of similar viscosity so the material is easily applicable but also not too fluid so it cures fast enough to create stable connections. Methods to dispense the conductive material include pump based dispensers that are activated during print time to lay down layers of material or by syringes (see figure 2.1) that are driven by stepper motors, the later of which is used in the 3D printer modified in [Wasserfall, 2015].

Another method for direct writing of conductive material is Aerosol Jet Printing (AJP). In this technique a conductive substrate is vaporized by pneumatic or ultrasonic atomizers which creates an aerosol that can be dispensed through a nozzle onto the 3D printed object in very thin layers as demonstrated in figure 2.2 [Wilkinson et al., 2019].

2.2 SMT placement machines

The use of SMD components in the assembly of printed circuit boards is a widely used practice which benefits the size of the final product just as well as storage space and manufacturing costs through the possibility of large scale productions and fast automated assembly which would not be feasible with manual labor.

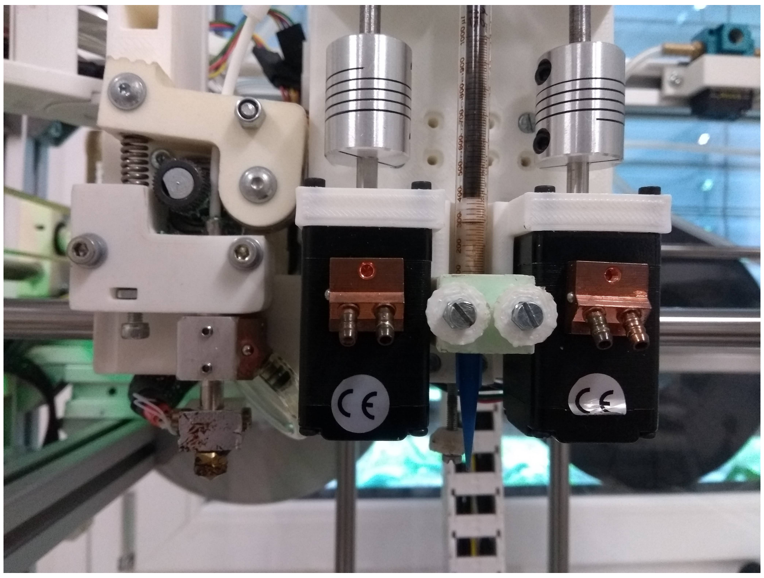


Figure 2.1: The syringe disposing conductive material into 3D printed objects as included into the Kühling&Kühling 3D printer by [Wasserfall, 2015].

While work on robotic general purpose pick and place machines is widely used and well documented, the development of SMT specialized pick and place machines, especially when integrated into the 3D printing process, is still in an evolving stage [Gokulnath et al., 2018] [Wasserfall, 2015]. Even though they operate in different application fields and vary greatly in size and form of their working material, the general workflow and capabilities share similarities.

The following is based on the work of Masri Ayob and Graham Kendall [Ayob and Kendall, 2008] as well as Moyer and Gupter [Moyer and Gupta, 1996] and gives an overview of different kinds of SMT placement machines. Industrial SMT placement machines come in a large variety of sizes and mechanical structures depending on their application purpose. Classifications are just as numerous with attempts to categorize based on their functionality like Selective Compliance Articulated Robot Arm (SCARA), High Speed Chip Shooters (HSCS) or Cartesian/Gantry based systems as defined by Moyer and Gupter. Chip shooters like the one in figure 2.4 (c) have either fixed pickup and place positions or eliminate the process of picking entirely by providing the placement machine with component reels that can directly be fed to the turret head. These machines are fast but also very limited in their capabilities and amount of different components they can assemble. While SCARA machines as seen in figure 2.4 (a), three jointed robotic arms, are best suited for tasks including high variety of uniquely shaped components within a small production volume, the cartesian machines in figure 2.4 (b) have a higher throughput and generally smaller build size. Cartesian pick and place machines, named after the use of cartesian coordinates for positional values, share great similarities with most modern 3D printers. They have three axis of linear control which are connected in right angles to each other. In contrast, a SCARA machine rotates around a center point which makes the inclusion into other CNC machines more complicated. The pick and place machine

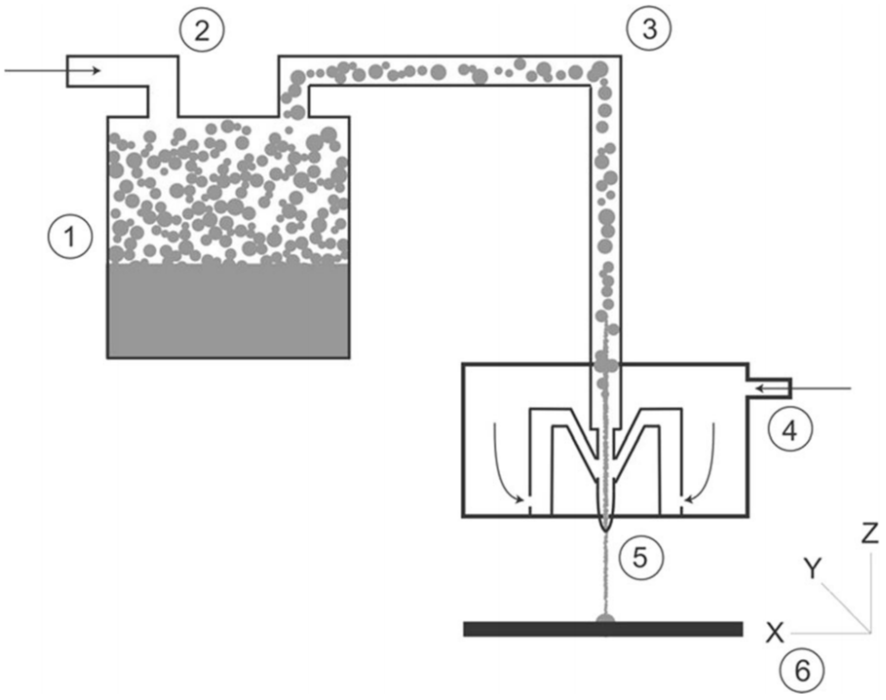


Figure 2.2: Schematic depiction of Aerosol Jet Printing as defined in [Wilkinson et al., 2019]. The aerosol is vaporized using a atomizer (1) and pumped by a transportation gas (2) through the transport system (3). The stream of aerosol can be controlled using an focusing valve at the nozzle (4). It is finally deposited onto the surface (5) through a dispension nozzle, directed by a computer-controlled translation (6).

discussed in this thesis is based on the cartesian approach.

2.3 Computer vision for pick and place machines

One of the main applications of computer vision algorithms in pick and place machinery is to detect and classify objects in provided image data. While the classification of objects in cluttered scenes is of high importance when dealing with robotic arms or other pick and place machines working in less controlled environments, it is of minor concern in the field of SMT pick and place machines. Due to the way electronic components are provided through sorted trays or feeder reels, the machine has exact information about type or shape of the component that is to be picked up.

The way object detection through computer vision is achieved are diverse. To give a better understanding of how these are implemented and to show a few common nominators among them the following text gives a few examples.

Feature extraction

Before any object can be recognized the raw image has to be preprocessed to only contain vital information. The following approach was described by Kumar et al. in [Kumar et al., 2014].

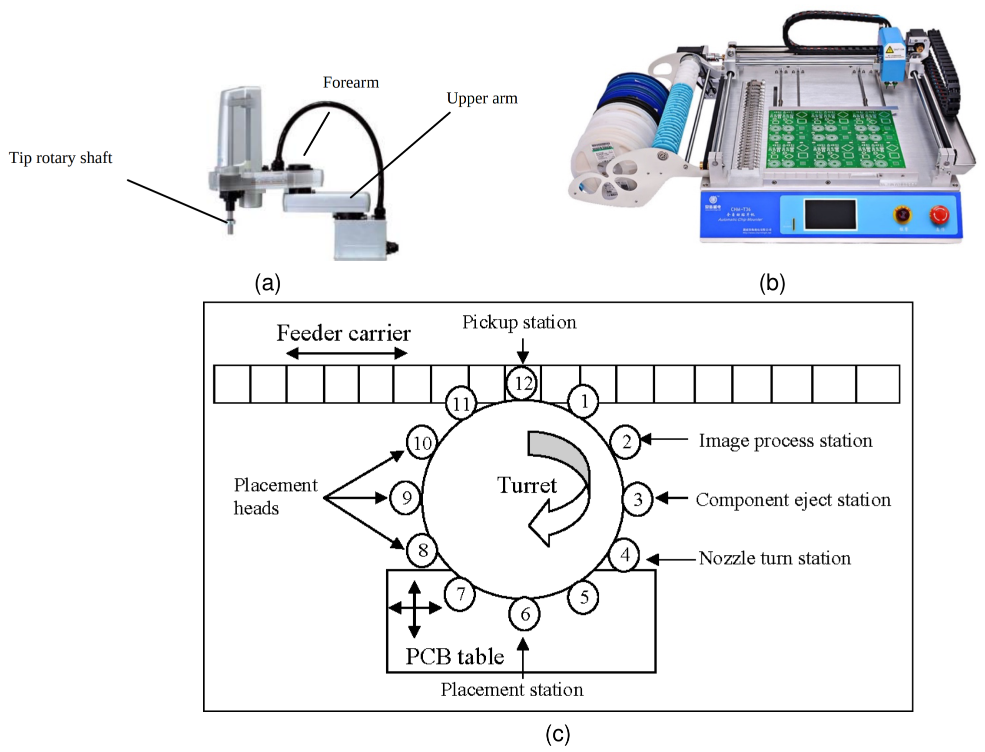


Figure 2.4: Three different forms of pick and place machines. Images show a Selective Compliance Articulated Robot Arm (a) [Li et al., 2015], a Cartesian/Gantry based system (b) [Gokulnath et al., 2018] and a schematic depiction of a High Speed Chip Shooter (c) [Ayob and Kendall, 2008].

• *Grayscale*
The image that is provided with a RGB color range is converted into a 2 dimensional grayscale image by removing hue and saturation information and only keeping the luminance.

• *Binarize*
Next the grayscale image is turned into a black and white binary image. The author implies the use of Otsu's method, a adaptive thresholding technique further described in 4.2.

• *Edge detection*
To ease further processing steps a edge detector algorithm is applied to read the contours from the image. The algorithm is based upon the Canny edge detector that finds edges by searching for local maxima of the gradient in the image which in a binary image are the transitions between the black and white areas in the image. The modified algorithm in further detail:

- *Smoothing* - For better results an additional step is performed before edge detection to clean the image from noise.
- *Canny* - Extracting the edges.

- *Non-Maximum Suppression* - Turning the blurred edges into sharp edges by deleting all but the local maxima in the gradient image.
- *Double Thresholding* - Optimizing the edge selection by preserving pixels above the high threshold and suppressing the ones weaker then the lower threshold value.
- *Edge tracking by Hysteresis* - Cleaning the edges by preserving strong edges with high threshold values as well as edges that connect to these strong edges.

While different feature extraction processes exist, most tend to follow a similar schema of grayscaling, binarization and edge detection.

Object detection
After extracting its features the image has to be analysed to detect the individual objects from the preprocessed raw image data. Kumar et al. [Kumar et al., 2014] proposes the use of Artificial Neural Networks (ANN) for classification.

A simpler approach, used in SMT pick and place machines that only handle image data containing single components, is the use of contour detection. In this process a contour detection algorithm is applied to the binary image to find separately contained shapes in the image. These can then be used to create rotated bounding boxes surrounding the shapes [Gokulnath et al., 2018]. To find the most likely candidate among the detected shapes, the footprint size is compared to the expected component size. The shape that has a footprint closest to the expected size is chosen. This approach is only valid as long as the raw image data only contains single components and is not cluttered with similar sized objects. The process is discussed in chapter 3 in further detail as part of the computer vision pipeline introduced in [Wasserfall, 2015].

Another concept used in a wide array of industrial applications is template matching. The following description is based on [Ulrich et al., 2003]. Due to the restrictions that apply in the production of SMD components there is little to no variation between individual units of the same type, package and manufacturer. Variables like dimensions or shape are known and can be provided to the template matching algorithm in a generalized way that can be applied to all units of the same component. Models, more commonly known as reference images, that are used to retrieve occurrences of the component can directly be generated from within the computer aided design (CAD) software that is used to design the electronic circuit. Ulrich et al. distinguish between gray value and feature based strategies, giving feature based approaches the advantage that its structure made of points, edges, polygons or regions are better suited to characterize objects then gray value information. Due to its robustness against occlusion, clutter and slight deformation, Ulrich et al. employ an optimized version of the Generalized Hough Transform, which in its form as described by Duda and Hart [Duda and Hart, 1972] is further analysed in chapter 4.5.

3 The existing Computer Vision Pipeline

As of the begin of this thesis the 3D printer hardware was already modified to support the mechanics for picking and placing of SMD components into 3D printed objects. The necessary hardware modifications include the addition of a vacuum gripper nozzle to the 3D print head as well as two cameras, one attached to the print head to take overhead photos of parts to be picked up, as well as a second camera attached to the print bed which can be used to take pictures from below to retrieve part offset and orientation.

This chapter is a short summary of the current pick an place workflow utilized by the aforementioned Kühling&Kühling Reprap 3D printer.

3.1 Picking

To pick a component the print head camera is first aligned with the tray containing the SMD component. The camera takes a picture that is cropped during capturing to only contain the approximate area of interest. The raw image is now saved on the print server for further processing. To simplify the extraction process the content of the tray is separated from its surrounding green box, as seen in figure 3.2, by cropping the image to the size of the inner tray. This way the image only contains the SMD component in front of a neutral background. To extract the component from the picture the image is turned into a gray scale image that is binarized afterwards. By reducing the image information to a binary state the contours can be easily extracted. These contours are now sorted and compared to the size of the expected component. When a appropriate candidate is found a bounding rectangle is created that surrounds the contour. This bounding rectangle is used to calculate the center of mass. Its position provides the offset of the component in relation to the printhead camera. The vacuum gripper that is fixed to the printhead is now aligned with the retrieved coordinates and the component is picked from the tray.

3.2 Placing

In the second part of the pick and place procedure the component that is now attached to the vacuum gripper is positioned above a print bed camera. Once again the image that is retrieved from the camera has to be preprocessed first. The vacuum nozzle holding the component in place provides consistent background color that is masked and separated from the picture

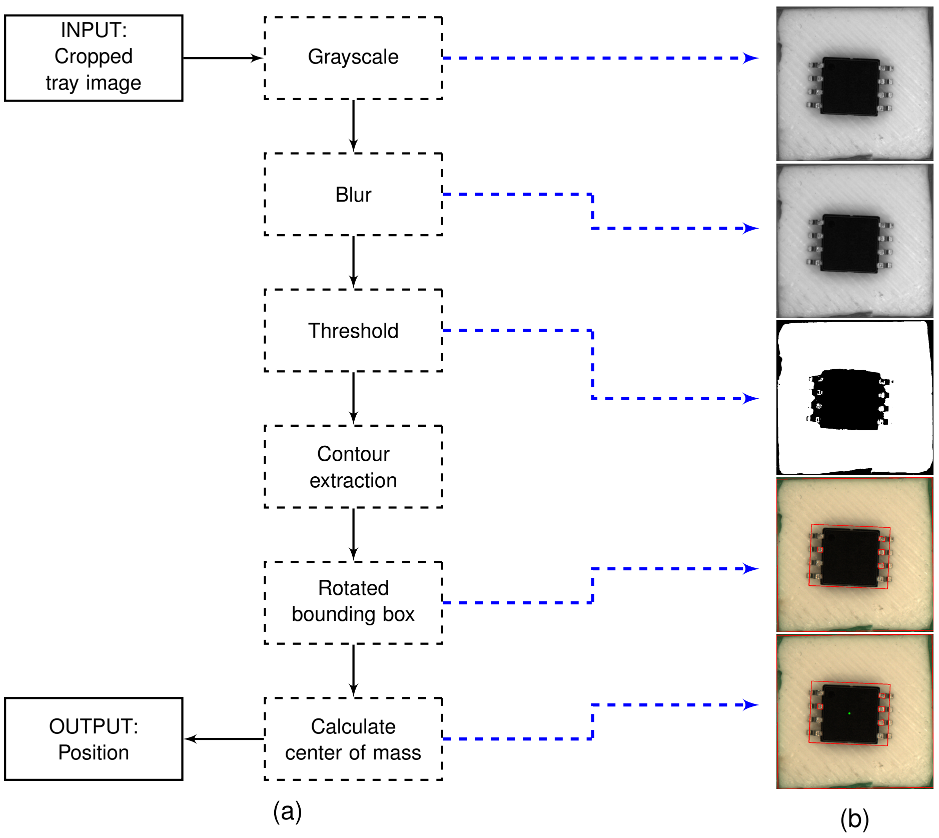


Figure 3.1: A schematic representation of the original cv pipeline for retrieving the components position from a provided tray image (a) as well as sample images depicting the individual steps of the cv pipeline (b).

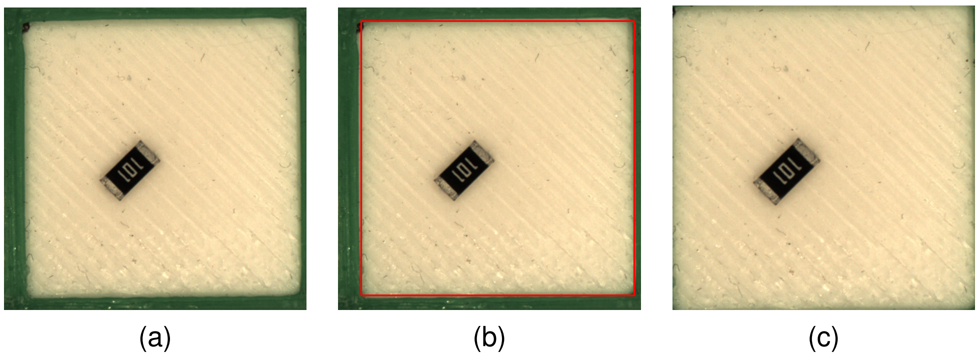


Figure 3.2: Sample sequence showing the process of cropping a tray input image (a) to the size of the inner tray (c) by extracting the shape of the surrounding green box (b).

through color filtering. Edges and contours are now detected and processed to calculate bounding rectangle and orientation. To achieve the desired component orientation the vacuum gripper is now rotating the component. Once the component is aligned correctly its center of mass is once again retrieved and used to calculate the components center offset to the center of the gripper nozzle. The part can now be placed on the printbed.

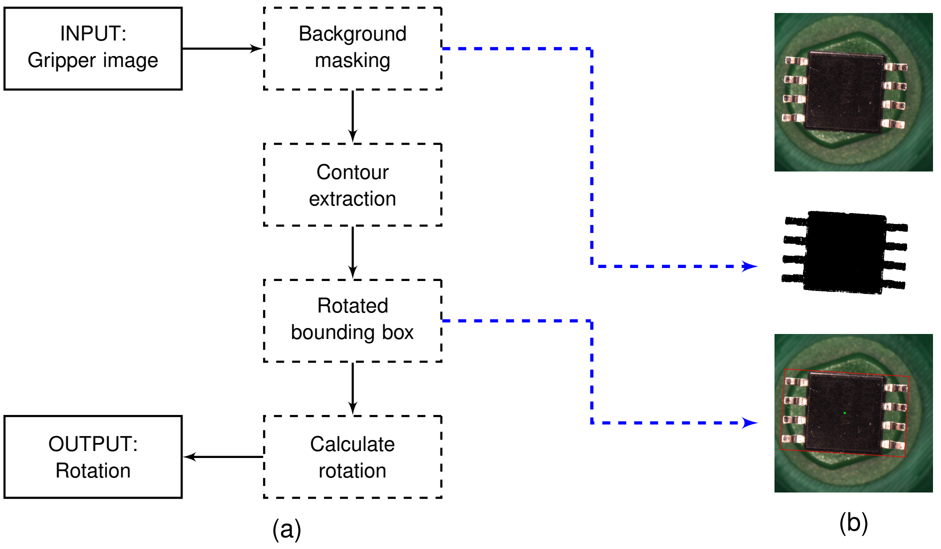


Figure 3.3: The schematic representation of the original cv pipeline for retrieving a components orientation and positional offset from the gripper nozzle (a) as well as sample images depicting the individual steps of the cv pipeline(b).

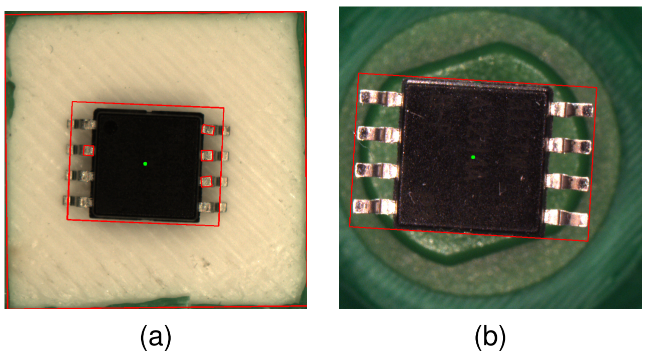


Figure 3.4: Sample images showing the successfully applied cv pipeline, retrieving component position and bounding box from a tray image (a) as well as a gripper image.

3.3 Sample execution

The following section provides sample output of two components, an ATTiny 85 and a white LED, to get a deeper understanding of how the pipeline works and what typical results look like.

ATTiny 85

The pictures in figure 3.4 show a successful run of the computer vision pipeline to detect the components position on the tray *(a)* as well as the detection of rotation and position offset on the vacuum gripper *(b)*. To visualize the results bounding boxes and detected position are drawn into the pictures. While the pipeline picked up rotation and position of the SMD component on the gripper image with rather high precision, the tray image shows a few of its weaknesses:

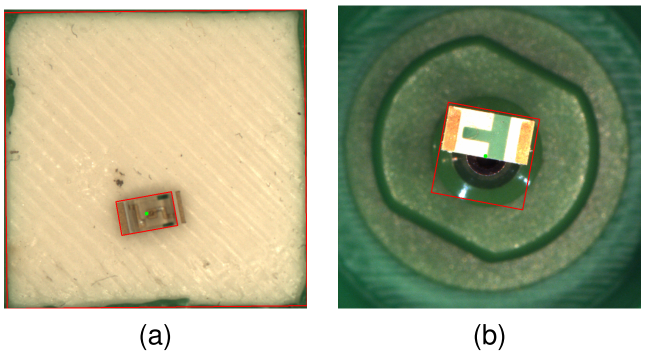


Figure 3.5: Sample images showing failures to detect a white LED on tray (a) and gripper images (b).

- *Shadows and background noise.*

  The retrieved bounding box includes parts of the background that are distorted by the shadow of the component on the white tray. These malformed bounding boxes lead to a miscalculation of the center of mass.

- *Parts blending in with the background.*

  When observed closely it shows that parts of the components metal legs are picked up as individual contours and therefore not as parts of the main object in the picture. This is also true for the ends of the components legs that are cropped from the rest of the object. These parts tend to blend in with the background and wont get picked up by the pipeline.

White LED

While the ATTiny 85 with its big, dark body is rather forgiving to these inaccuracies, other components like the white LED seen in figure 3.5 tend to intensify it. Parts of the component are so close to the background color that they are completely cropped from the object leading to big offsets of the center of mass *(a)*. The gripper image *(b)* shows how reflections and even the gripper nozzle itself are picked up from the pipeline as part of the object and are getting included into the bounding box. This shows more problems of the existing pipeline:

- *Vulnerability towards deformed shapes.*

  The pipeline has no knowledge of the supplied component other than its approximated size. Without further information the pipeline can not distinguish between shapes that are part of the component or malicious artefacts included in the picture.

- *Relying on manual calibration.*

  Values that are used throughout the image processing of gripper and tray images are calibrated by hand and are fixed for all components that are processed. This includes threshold values for creating binarized images but also the green color range that is cropped from the gripper images. Based on the diversity of possible components shapes, colors and sizes it would be rather hard to find universally applicable values for all of these components. Even further these values would have to be adapted to changing environmental conditions like lightning or even camera settings.

3.4 Summary

Images captured by the cameras included in the 3D printer have to be preprocessed, analyzed and interpreted into information which can be used by the pick and place machine. This includes shape, orientation and the center of mass of the SMD components. Extracting the exact information from optical image data can be error prone. Currently the image processing relies on manually set values for creating threshold images for edge detection or creating color masks that are used to remove the green background from the vacuum gripper nozzle. These values are fixed and depend on consistent images taken by the cameras. Exterior influences like lightning, focus or even the image content itself make it hard to find settings that can be universally applied to all images taken by the pick and place machine.

4 The modified Computer Vision Pipeline

To make the image processing less dependent on manual configuration and improve its reliability the modified image processing pipeline utilizes adaptive threshold techniques and auto configuration features that create color masks based on current camera settings and environmental influences before the 3D printing process starts.

Instead of just extracting a rectangular bounding box the image processing pipeline approximates the convex contour of the component to retrieve a more precise prediction of the actual center of mass of the component in the picture.

Furthermore a template matching method is implemented to improve on the retrieved orientation and offset of the component attached to the gripper nozzle. This is achieved by finding a supplied template image of the component, scaled to its predicted size, inside the gripper image. Due to the lack of a consistent production standard for component markings, templates still have to be created manually based on dimensions provided by the part manufacturer.

4.1 Automated color range selection

To crop the green background from the gripper image taken by the print bed camera a color range in **H**ue, **S**aturation and brightness **V**alue (HSV) color space has to be specified first. To simplify the process of creating the color range a new calibration feature has been added to the OctoPNP UI. Instead of modifying the color range by hand every time the camera settings change the user can now take a picture of the gripper nozzle from the UI which automatically creates a color range based on the image data. After the picture is taken it is converted into the HSV color range and blurred to smooth the resulting color values. Now the individual color values of each pixel are extracted and stored in a color matrix. Based on this matrix the mean and standard deviation for hue, saturation and the brightness can be calculated. These values can be used to create lower and upper boundaries for the background color range by either adding (upper bound) or subtracting (lower bound) the appropriate mean and deviation values. The resulting color range is then stored in the global OctoPrint configuration file so it won't have to be recreated each time the printing process starts. This process is visualized in figure 4.1

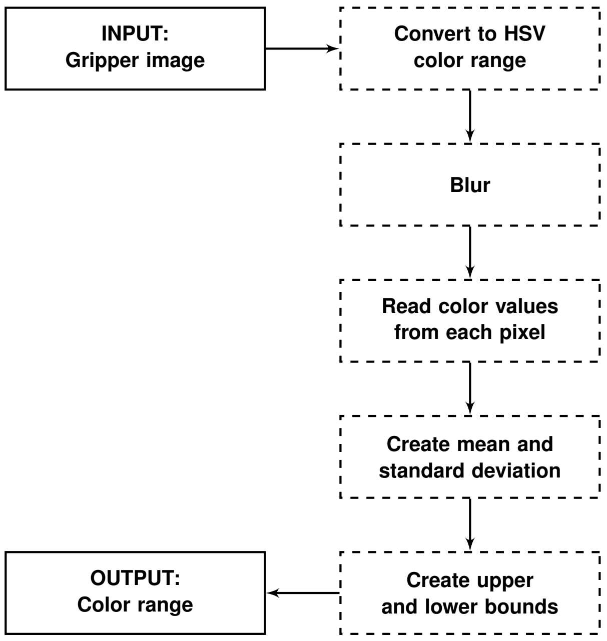

INPUT:
Gripper image
Convert to HSV color range
Blur
Read color values from each pixel
Create mean and standard deviation
Create upper and lower bounds
OUTPUT:
Color range

Figure 4.1: CV pipeline for creating the color range from an gripper image.

4.2 Adaptive thresholding techniques

To decide what part of an image represents a components or their background, segmentation techniques are applied. One of the most common segmentation techniques is thresholding [Dey et al., 2014]. Thresholding is used repeatedly throughout the implementation of this computer vision pipeline in the preparation of raw image data, enabling contour detection or template matching. In its essence the thresholding process fixes all pixels that are passing an set threshold to a foreground value while all other pixels are set to a background value [Dey et al., 2014]. Most thresholding techniques apply global threshold values for the entire image that is processed [Dey et al., 2014]. While this leads to adequate results when the pictures taken are highly reproducible in content and quality, this includes lightning distribution and shadows, results tend to very once component sizes or color change. The advantage of adaptive thresholding techniques is that the threshold value does not have to predefined for the entire picture but instead changes dynamically while processing the image data [Dey et al., 2014]. The adaptive thresholding technique used in this pipeline uses Otsus Method [Otsu, 1979]. Otsus method assumes that the image is bimodal and thus can be seperated into two classes of pixels, background and foreground [Dey et al., 2014]. Bimodal pictures can be described as pictures with two peaks in its histogram [Dey et al., 2014]. The threshold values can then be calculated as the median of these peaks [Dey et al., 2014]. A downside of this approach is that Otsus method needs this bimodal distribution of grey values to correctly calculate the threshold value [Dey et al., 2014]. Due to the static background that all components share on either gripper or tray the resulting pictures are consistent enough for Otsus method to produce steady results, independent of the component at hand.

4.3 Morphological Transformations

After creating a binary version of the tray image morphological transformations are applied to clean the search image without loosing the sharp corners of the component. This is achieved by applying opening and closing on the image. While opening describes the process of eroding and then dilating an image to remove white spots or enclosures, closing applies dilating and then eroding to achieve the opposite effect [Brahmbhatt, 2013]. The combination of both transformations allows to remove the noise from the binary image without loosing too much details. This makes opening and closing the image more favorable then applying a blur filter that softens and thereby obfuscates the contours of the component before creating the binary image.

4.4 Convex hull

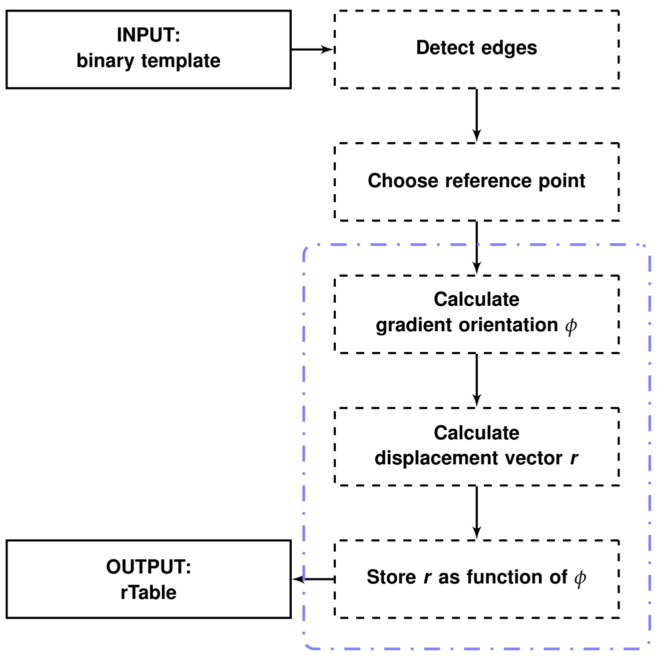
To transform the contours detected in the image into a center location the existing pipeline used rotated bounding rectangles. While simplifying the shape of the components contour into the form of a rectangle makes calculation the center of mass particular easy it also looses a lot of shape information that is necessary to precisely grip the component from the tray. This is especially true when working with irregularly shaped components like NPN photo transistors. Replacing the bounding rectangle with a convex hull creates more robust outer bounds of the component while still simplifying the shape enough to ease calculating the center of mass.

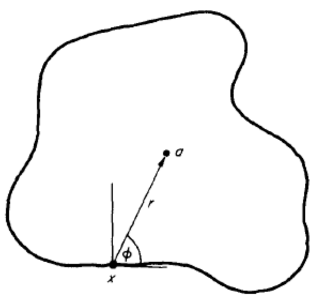
4.5 Template matching using Generalized Hough Transform

While originally developed to detect analytically defined shapes like lines, circles or ellipse by P.V.C. Hough, as stated in his 1962 patent, a extended version known as generalized Hough Transform was developed and published by Duda and Hart in 1972 which can also be applied to detect more elaborate known shapes in search images[Duda and Hart, 1972].

The basic idea of Hough Transform describes that, given a set of Points **P**, every pixel $p = [x, y] \in \mathbf{P}$ could be part of a line [Treiber, 2013]. To detect all lines in **P**, every p has to vote for all lines that pass through that specific pixel [Treiber, 2013]. The line that accumulates the most votes is thereby the most likely candidate.

To make this work the search image has to be turned into an edge image to reduce the amount of calculations to only include contours points. While lines in 2D space can be described in the simplest way by a pair consisting of slope and position, this description leads to problems regarding horizontal lines. Duda and Hart used the Hesse normal form $r = x\cos\theta + y\sin\theta$ for line representation in which every line going through a point x, y can be described by the distance r to the origin and the angle θ between the normal of the line and the x axis





Displacement vector r for an arbitrary boundary point x [Ballard, 1981]

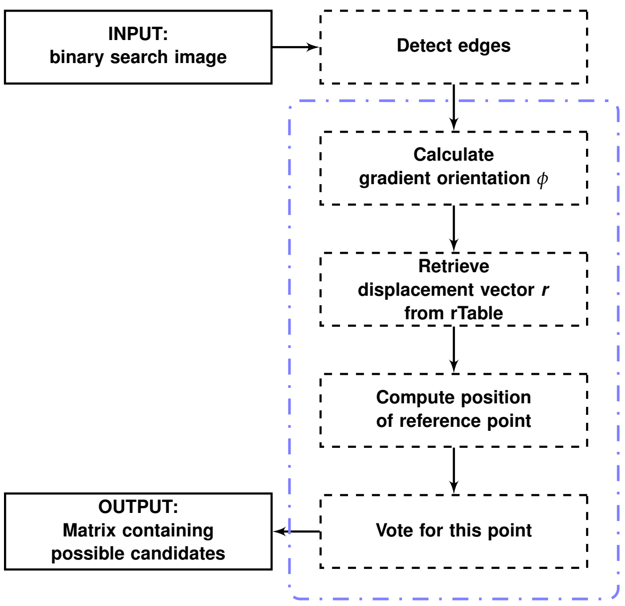
Figure 4.2: Building the template model that is used to find the component in the search image. Steps in the blue box have to be repeated for every edge pixel.

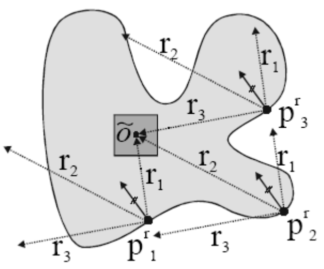
[Duda and Hart, 1972]. With these information it is now possible to describe each line going through a point by a pair of variables (r/θ). By calculating these values with r being the distance between the origin and a single contour point of the edge image and θ varying angles, the accumulated values create a plane referred to as Hough space which is increased by one at the position (r/θ) for every value of r [Duda and Hart, 1972].

To recognize arbitrary shapes that are more complex, the generalized Hough Transform [Ballard, 1981] requires an extra step. Before the shape can be recognized in the image a contour model has to be created and trained which then can be used to retrieve occurrences of the model in the search image [Treiber, 2013]. Only after this initial training phase the model can be applied and used to find the shape inside a search image in the recognition phase [Treiber, 2013].

4.5.1 Training the model

Before the model, in our case the contour template of an SMD component, can be recognized in the search image it first has to be trained. The information that are obtained during this training process are stored in a structure called the rTable [Ballard, 1981]. *Step1*: At first the binary image of the model has to be transformed into a edge image by applying an edge detection algorithm [Ballard, 1981]. *Step2*: A point o has to be specified that is used as a reference for all further calculations [Ballard, 1981]. *Step3*: For every contour pixel x of the edge image, the gradient orientation $\phi(x)$ and the displacement vector $r = o - x$ from the pixel position x to the reference point o have to be calculated [Ballard, 1981]. The vector r is then stored as a function of ϕ in the rTable [Ballard, 1981]. This step is visualized in figure 4.2


INPUT:
binary search image
Detect edges
Calculate gradient orientation ϕ
Retrieve displacement vector *r* from rTable
Compute position of reference point
Vote for this point
OUTPUT:
Matrix containing possible candidates



Reconstruction of the reference origin [Ulrich et al., 2003]

Figure 4.3: Applying the model to the search image and retrieving possible candidates. Steps in the blue box have to be repeated for every edge pixel.

inside the blue box and has to be repeated for every countour pixel.
It is also important to notice that the rTable is vector valued which means that every value of *phi* can reference multiple values of *r* [Ballard, 1981].

4.5.2 Recognizing the model

Once a model based on the template is trained the problem of finding the shape of the template inside a search image can be simplified to finding the translation, rotation and scaling transformations necessary to map the model into the picture. The following description of the recognition process is based on [Ballard, 1981] and [Treiber, 2013]. The recognition phase starts similar to the training phase, in which the search image has to be turned into a edge image before it can be processed. Once a binary edge image has been created, the gradient orientation $\phi(x)$ can be obtained for each pixel x of the edge image. The appropriate displacement vector r can now be retrieved from the rTable that was created during the training phase of the model. Based on the pixel position of x and the displacement vector r a assumed reference point $\tilde{o}$ can be calculated. After the possible reference point is calculated a accumulator array is increased, or in other terms, voted for. This process is once again represented in figure 4.3 within the blue box. The result of the recognition phase is a matrix containing all possible candidates of transformations that map the model into the search image with the most likely candidate being the one that accumulated the most votes. If the model in question is only translated without any further transformations, the process described so far would suffice to retrieve the models location from the search image and return a two dimensional matrix capable of containing all possible candidates. Taking into consideration that the model can not only be translated but also rotated or even scaled, the process has to be repeated for each of transformations, retrieving alternative center points for every possible location, orientation or size of the model. To accumulate the votes for these transformations the size of the matrix has to be increased to four dimensions as well.

4.5.3 The best candidate

After accumulating all votes in a four dimensional matrix the most likely candidate has to be decided on. In this specific use case only a single occurrence of the model in the search image is to be expected and the candidate with the most votes can be classified as the best candidate. Still every single vote in the matrix represents a possible candidate with its own set of transformations [Ballard, 1981]. In a scenario in which multiple copies of the same model can occur in the same search image the Generalized Hough Transform can retrieve all of these occurences in a single run [Ballard, 1981].

4.5.4 Drawbacks and runtime optimization

Applying Generalized Hough Transform to find arbitrary shapes in search images is both computational as well as memory intensive [Ulrich et al., 2003]. This is especially of concern regarding the hardware setup used in most consumer grade or even professional 3D printers. Machines like the Kühling&Kühling Reprap 3D printer utilize a single board computer to host and run the majority of their 3D printing software. With the additional workload of running Generalized Hough Transform directly on the machine it is necessary to optimize the process and input data without loosing the accuracy that Generalized Hough Transform provides.

- *Optimizing the process*

  To decrease runtime of the recognition phase during shape detection both scale and rotational ranges can be narrowed down to only cover specific predefined values. With a fixed distance between camera and component as well as the knowledge about the components dimensions it is possible calculate the expected pixel size of the component within the picture. Setting the minimum and maximum scale size close to this value cuts down on scaling steps during the recognition phase.
  Similar to scaling, the rotation that has to be covered by the Generalized Hough Transform can be narrowed down by expecting the component to be supplied in the component tray within a maximum rotational offset.

- *Optimizing the input data*

  The raw image data that is captured by the cameras has a high resolution. While this is ideal to detect even small details in the image it also slows down the recognition process with a high amount of edge pixels that have to covered by the algorithm. Scaling images to half of their original size reduces the amount of pixels that have to be covered to one fourth of its original value. While providing the search image in such a considerable

```
<part id="2" name="ATTiny2">
  <position box="2"/>
  <size height="1.87" width="5.38"/>
  <shape>
    <point x="-2.6" y="-2.6"/>
    <point x="-2.6" y="2.6"/>
    <point x="2.6" y="2.6"/>
    <point x="2.6" y="-2.6"/>
  </shape>
  <pads>
    <pad x1="-2.155" y1="-4.0" x2="-1.655" y2="-2.054"/>
    <pad x1="-0.895" y1="-4.0" x2="-0.395" y2="-2.054"/>
    <pad x1="0.375" y1="-4.0" x2="0.875" y2="-2.054"/>
    <pad x1="1.645" y1="-4.0" x2="2.145" y2="-2.054"/>
    <pad x1="-2.155" y1="2.054" x2="-1.655" y2="4.0"/>
    <pad x1="-0.885" y1="2.054" x2="-0.385" y2="4.0"/>
    <pad x1="0.385" y1="2.054" x2="0.885" y2="4.0"/>
    <pad x1="1.655" y1="2.054" x2="2.155" y2="4.0"/>
  </pads>
  <destination x="20" y="10" z="0" orientation="45"/>
</part>
```

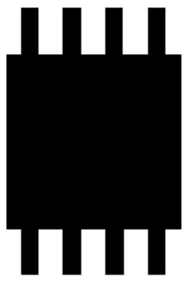


Figure 4.4: The source G-Code file containing the shapes that define the components outline (left) and the binary image of the tempalte that is generated on basis of the shapes contained in the source file.

lower resolution does reduce runtime it also loses fine details which can lead to mismatched or misaligned templates. If applied, this optimization has to be used carefully with regard to a efficiency to quality of output ratio.

4.5.5 The template

To find a shape inside a search image the shape detection algorithm has to be provided with a template or reference image. The template is used to train the model to find the shape of interest inside the search image. While Generalized Hough Transform expects to be provided with a binary edge image of the original template it would be rather inefficient to store image data for each component on the machine itself. Ideally the template should be easy to obtain and use without having to rely on an extensive backup library of component pictures.

- *Obtaining a template*

  Electronic design automation software (EAD) like Autodesk EAGLE is a popular choice in the design and production process of printed circuit boards (PCB). A added bonus is the inclusion of comprehensive component libraries that provide information about dimension, or at least landing pattern, of specific components by different manufacturers and package sizes. These information are usually provided in the form of geometrical shapes for the component body and, if present, soldering pads. These information can be exported directly through the extended feature set of a specialized version of the Slic3r software [Slic3r, 2019], modified by Daniel Ahlers to include the capability of designing electronic circuits into 3D objects [Ahlers, 2015]. Slic3r is used to convert these 3D models into layer information that can be saved as G-Code files and interpreted by the 3D printer. By the addition of electronic components, we can extract all necessary information about shape and size directly from the G-Code file and import them into the computer vision pipeline.

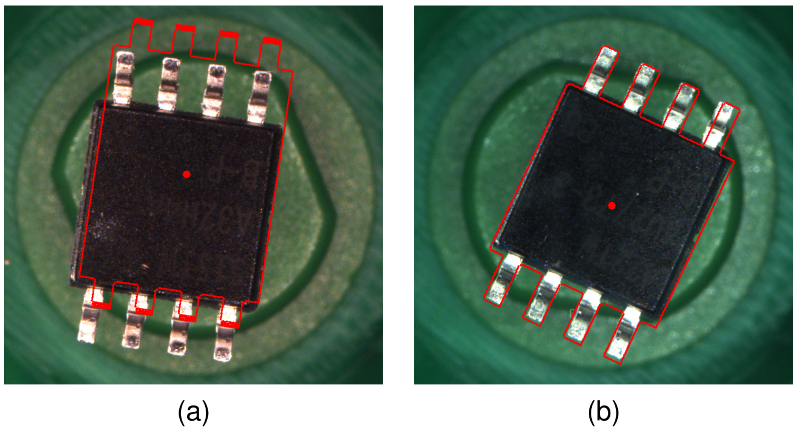


Figure 4.5: Comparison of template matching precision of the shape exported from the EAD software (a) and the shape measured and modified by hand (b). Both images show the shape and the center point mapped into the search image.

- *Converting the template*

  To make the shape exported into the G-Code file useable for Generalized Hough Transform it first has to be converted into a binary image. This is achieved by reading the geometrical shapes from the G-Code file and directly drawing them into a binary image using black for the component and white for the background as seen in figure 4.4. The resulting image can then be used to train the model for the shape detection phase of Generalized Hough Transform.

Weaknesses and optimization

The template that is obtained by exporting the landing pattern from the EAD software rather describes a generalized form of the component in which it can be placed on the PCB. While this abstraction does make sense in the field of printed circuit boards in which it creates a margin of tolerance for pins of micro controllers, it obfuscates the actual shape in such a degree that it hinders any attempt of precise template matching.

As seen in figure 4.5 (a) the resulting template that is exported from the software does not match the physical shape of the component and thus leads to imprecise detection of location and orientation in the image. The success rate of Generalized Hough Transform is dependent on the quality of the provided images. This is true for both the search image as well as the template. If the template is provided in a more accurate form, for example by manually measuring the component and adapting the G-Code to fit these values, the template matching results can be drastically improved. Figure 4.5 (b) shows that such a template with more accurate dimensions does improve the results.

4.6 Picking

The modified process for picking the components from the tray follows a similar approach to the one we have already seen in the existing computer vision pipeline. The image taken by

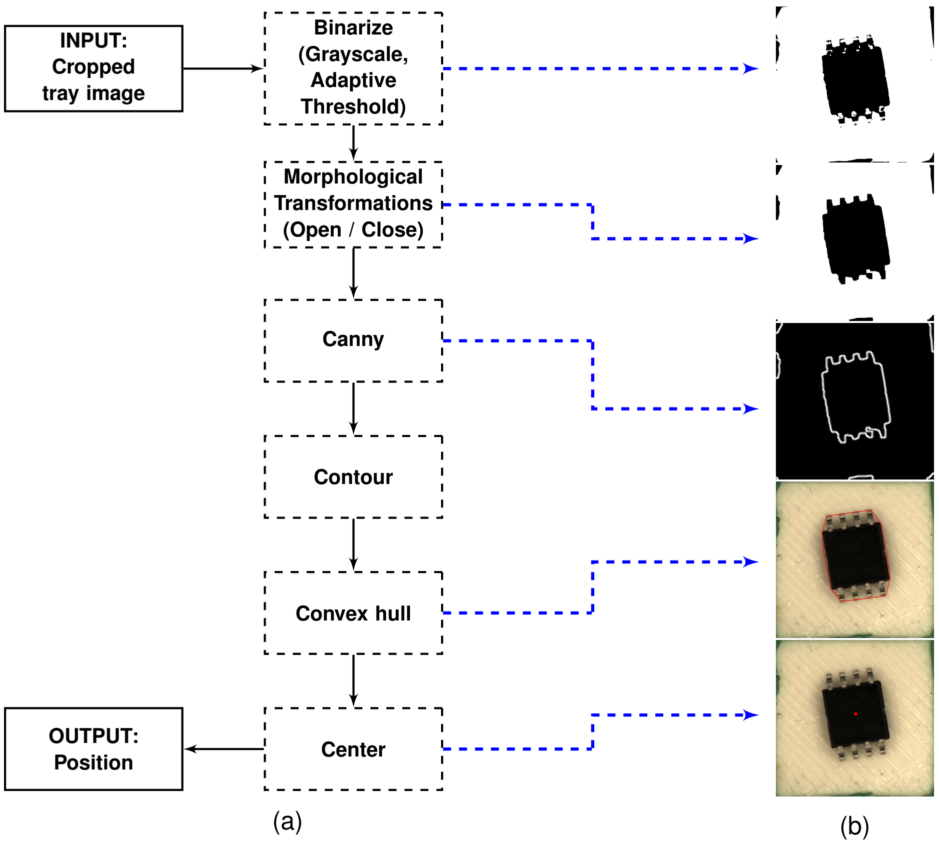

INPUT: Cropped tray image
Binarize (Grayscale, Adaptive Threshold)
Morphological Transformations (Open / Close)
Canny
Contour
Convex hull
Center
OUTPUT: Position
(a)
(b)

Figure 4.6: Modified cv pipeline for detecting the component position in a provided tray image (a) along with output images generated during the steps of the cv pipeline (b).

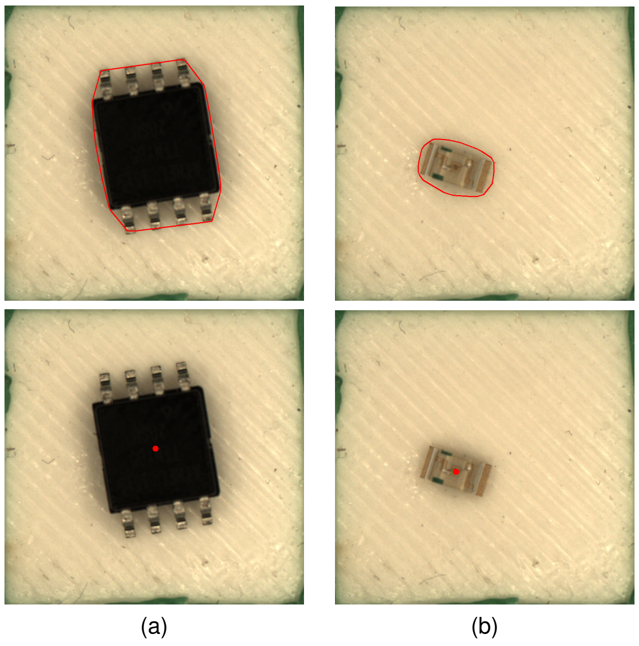


Figure 4.7: Sample results after applying the modified cv pipeline on tray images containing an ATTiny85 (a) and a white LED (b). The images show the detected bounding boxes as well as position.

the print head camera is cropped to the size of the tray before it is binarized to retrieve the contours contained in the image. In contrast to the original pipeline, this step is performed using adaptive thresholding and is followed by morphological transformations to ease the creation of homogeneous results of diverse components as described in section 4.2 and 4.3. The contours are then used to calculate the center of mass of the component contained in the picture. The next step is to create a convex hull (see section 4.4) that closer resembles the actual shape of the component. This shape is then used to calculate the center of mass of the component.

The Pictures shown in figure 4.7 demonstrate exemplary output of the modified image processing pipeline for picking a component from the tray. The red convex hull approximates the shape of the component more precise then the rotated bounding box could and provides a more accurate center of mass. When considering the output for the white LED in figure 4.7 (b) we can see that even parts that are blending in with the background color and are obfuscated through noise created by the components shadow still can be picked up more precisely then before.

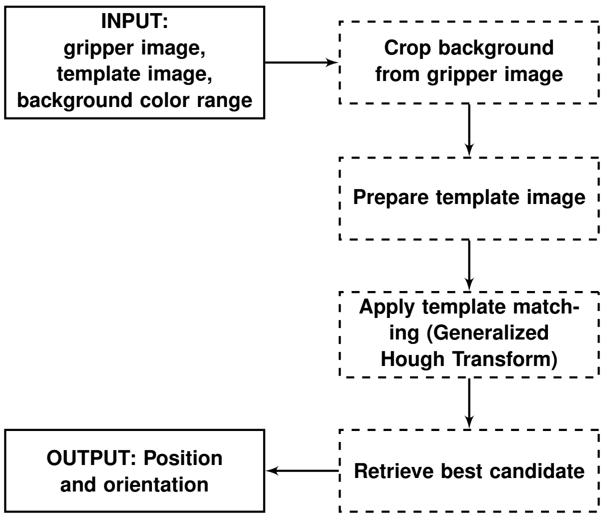

INPUT: gripper image, template image, background color range
Crop background from gripper image
Prepare template image
Apply template matching (Generalized Hough Transform)
Retrieve best candidate
OUTPUT: Position and orientation

Figure 4.8: CV pipeline for detecting the component orientation and position on a gripper image.

4.7 Placing

The process of placing a component, once it is picked from the tray, can be broken down into three phases:

Initial recognition phase
First the orientation of the component, that is attached to the gripper nozzle, has to be recognized in the picture taken by the bed camera. The offset between the current and the expected orientation can now be calculated based on these information.

Alignment phase
If necessary the component can now be rotated by the gripper nozzle based on the previously calculated offset. Once the component is realigned, another check of the current orientation is needed. This extra step validates the orientation or repeats the alignment process until the desired orientation is achieved. Is the component oriented correctly, the offset of the center of mass to the gripper nozzle center is calculated. This positional offset compensates for imprecise picking of the component by readjusting the component placement position on the printbed.

Placement phase
Finally the component can be placed on the printbed using the calculated center offset.

To prepare the gripper image for the template matching process it first has to be segmented. To achieve more consistent results this segmentation step has been modified to use a automatically generated color range based (see section 4.1). Instead of simply removing the background color the resulting image is binarized based on the provided color range. The accuracy of the placed component is mainly dependent on the precision of which the components orientation and offset is recognized and realigned with. The existing pipeline used rotated bounding boxes to calculate the center of mass as well as the orientation. As described in chapter chapter 3 this attempt is vulnerable to background noise, light or color mismatches or even shape deformation caused by objects that are falsely picked up from the nozzle background. To improve upon the existing placement strategy Generalized Hough Transform for template matching is implemented as described in 4.5. This method is more robust against background noise and less dependent on the detection of the precise outline of the component in the picture.

5 Evaluation

This chapter describes the method and shows the execution of the evaluation test that was developed to compare the results of the existing and modified computer vision pipelines to draw conclusions about changes in efficiency, precision and reliability.

5.1 Overview

The idea behind the test is to create measurable results to compare the precision and efficiency of both computer vision pipelines. To achieve this the test has to be easily reproducible, comparable and applicable to both pipelines without the need of any additional modification to the pipelines itself. It should also be as close as possible to a real world task that the pick and place machine would have to perform. In addition the test has to be completely handled from within the modified 3D printer and has to be manageable only using the available hardware, much in the sense of a calibration cube that can be printed to test the settings of a regular 3D printer. Building on this premise, the test has to contain features to test picking up, analysing and placing components while also documenting the individual steps by saving data of

- the detected position in the component tray,
- the analysed shape, rotation and positional offset on the gripper nozzle,
- the placed component itself
- as well as saving the time it takes to complete the entire placement test.

5.2 Preparation

Performing the afore mentioned tests require the preparation of a reproducible testing environment with a fixed set of parameters. Those parameters are the used components, the test file that is executed on the 3D printer and the test ground the components are placed on, which will be referred to in this section as the paper template.

5.2.1 Template

To be able to decide whether the placed components were positioned correctly, a stencil like template had to be prepared on which the components can be placed. While it would be possible to create this template by 3D printing a predefined shape onto the printbed in which the components have to be placed, this would also introduce another margin of error which is not part of the pick and place mechanic that is to be evaluated. Considering the amount of

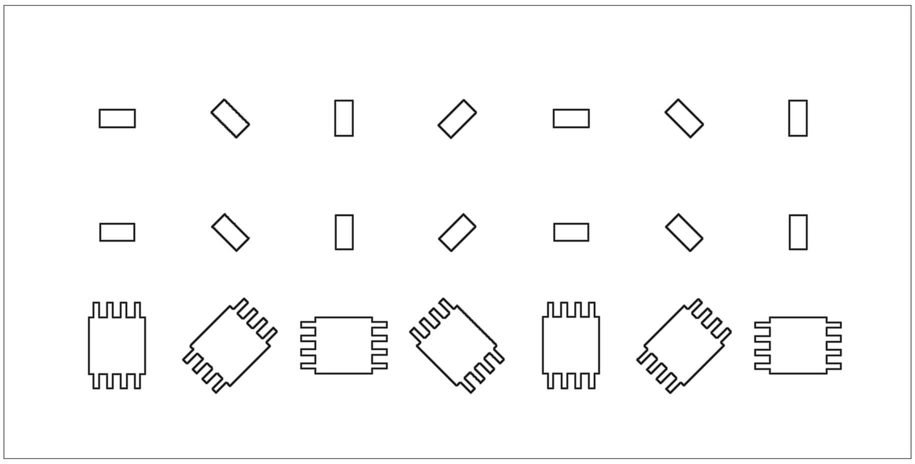


Figure 5.1: The paper template containing outlines of all components used for testing.

components that have to be placed it would also be rather time consuming. The test template, as seen in figure 5.1, can be printed onto a sheet of paper and contains precise outlines of the components that are going to be placed. The lower left corner of the template serves as point of origin. All component positions on the template are measured in relation to this point. When attaching the paper template to the printbed it has to be carefully aligned with the 3d printers x and y origin so that components will be placed in the assumed positions. The origin describes a point in 3D space in which both X and Y axis maintain a value of zero. As the printbed itself describes the zero position for the Z axis, this criteria is met as long as the paper template remains attached flush to the printbed. To help ease this process the printhead camera can be moved to the origin position itself and provide visual on screen feedback while aligning the paper template. Once the template is positioned correctly a thin layer of adhesive is applied to the surface of the template to ensure that the placed components won't be accidentally moved through vibration, airflow or other external influences.

5.2.2 Components

The parts used for this test are common SMD components and were selected based on their individual properties. The resistor 5.2 (a) is the smallest components available. While its shape is rather simple it has to be precisely detected and gripped from the tray to be successful. The light emitting diode 5.2 (b) is slightly bigger then the resistor. The dome shaped epoxy lense that sits on top of the component narrows the available space to grip the component down even further. The back of the component contains the polarity markings which indicate the direction the current can pass through the diode. These markings partly blend in with the background color of the gripper and provide an added challenge for the template matching process. Lastly the ATTiny85 5.2 (c), which provides a complex shape as well as reflective pins that make detecting the outline more challenging on both picking and placing operations.

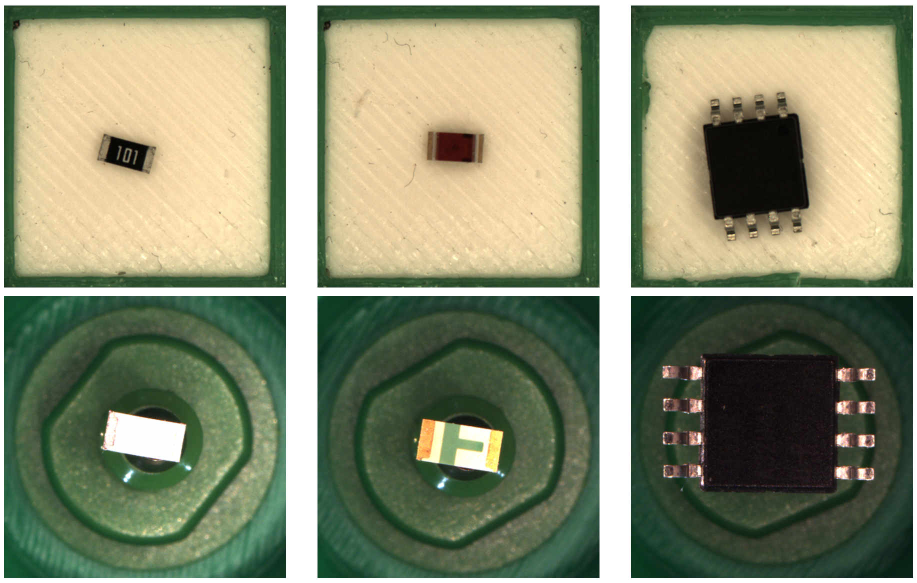

(a)Resistor (b)LED (c)ATTiny85

Figure 5.2: The components used for testing.

5.2.3 G-Code

The 3D printer has to be able to execute the pick and place test without the need for further hard- or software. Given that in a real world scenario all information about component type, rotation and position will be provided within the G-Code file it does make sense to use a modified G-Code file for the test setup as well. The only vital information the G-Code needs is the tray location of the individual components, the dimensions and shape of the component as well as the location and rotation the component has to be placed at.

```
<part id="1" name="ATTiny85">
  <position box="1"/>
  ; Height and width of the component in millimeters
  <size height="1.87" width="5.38"/>
  <shape>
    ; Geometric shape definitions for the body of the component
  </shape>
  <pads>
    ; geometric shape definitions for the individual soldering pads
  </pads>
  ; X, Y and Z placement position of the component in millimeters
  ; Placement orientation in degree
  <destination x="20" y="10" z="0" orientation="45"/>
</part>
```

The placement of the components can then be initialized by calling the custom G-Code hook *M361* passing the part ID *P<ID>* as an argument:

```
M361 P1
```

5.3 Documentation

To create comparable test results, three images of each placed component need to be saved on each test run. One of the detected shape and center of mass of the component inside the tray, one of the detected shape, orientation and center of mass of the component on the gripper and one of the final placement of the component on the paper template. The first two images can be taken during the image processing stages inside the computer vision pipelines. These images can then be used to find the cause for misaligned or misplaced components. The third image however can only be obtained after the component has been placed. To take consistent pictures of all components after the test run, another custom G-Code hook has been implemented. Once this hook is parsed and interpreted by the 3D printer, the head camera will align and take pictures of all part locations contained within this G-Code file.

5.4 Execution

The structure of the testing process is as follows:

- First the G-Code file containing all placement and template information as well as the G-Code hook to document the component placement has to be loaded into OctoPrint.
- Next all the components have to be placed inside the tray according to the tray layout in the OctoPNP UI as seen in figure 5.3.
- Now the test procedure is started by printing the G-Code file.
- Once the pick and place process stops the printing time is recorded and the component placement is automatically documented by the printhead camera. These steps are repeated for every individual test run.

When the last test run for the original computer vision pipeline was completed, the OctoPrint plugin, OctoPNP [OctoPnP, 2019], was replaced with the modified plugin [Thesis Source, 2019] which runs the computer vision pipeline described in this thesis. No changes to the printer hardware, paper template, camera configuration or other aspects of the test setup were carried out that could in any way impact the results of either of the pipelines.

5.5 Analysis

To evaluate the output of the pick and place test, a set of rules have to be declared on which the resulting images can be graded upon. The individually placed components will be classified based on these rules and separated into successfully placed or misaligned components as well as complete failures.

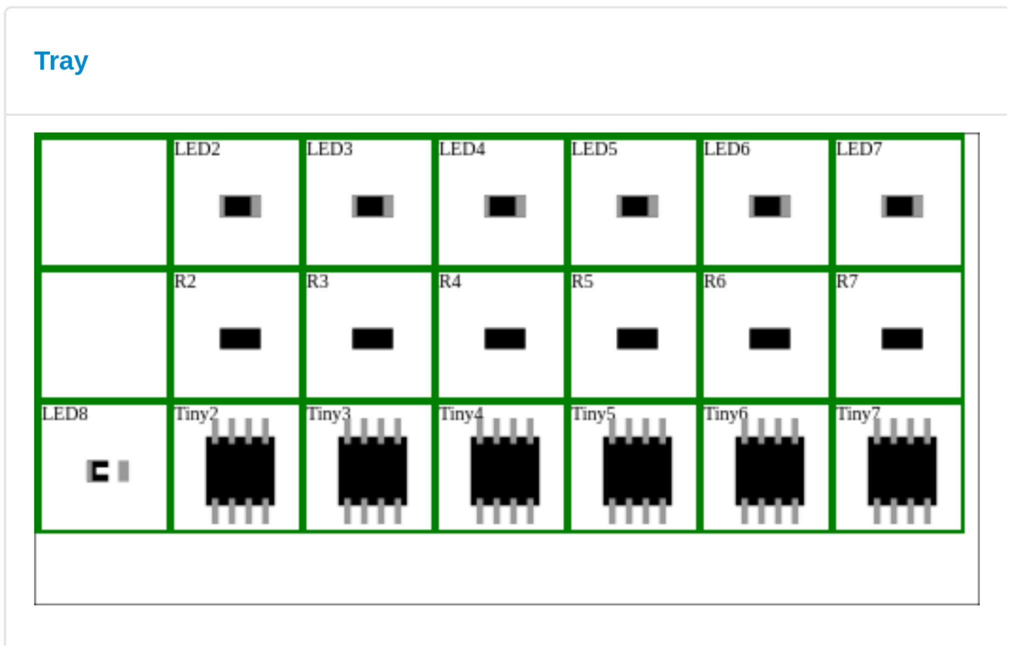


Figure 5.3: OctoPNP [OctoPnP, 2019] user interface showing the component tray as well as the components that have to be placed inside the trays.

5.5.1 Evaluation process

The following section describes the criteria on which the components will be categorized.

Successfully placed
All components that are placed close or even within the borders of their outline on the paper template count as successfully placed. These components show little to no translation or rotation offsets and would lead to a successful print in a real world scenario as seen in figure 5.4(a).

Misaligned
Misaligned components, as seen in figure 5.4(b), can be split into components with *rotation* or *translation* offsets. These components overstep their template borders and could, depending on the magnitude of the respective offset, impact the functionality of the final print.

Complete failures
Any component that is either not placed at all or placed in such a way that it would severely impact the print are classified as complete failures. The resulting print will definitely not work and might even impact the printing process itself. Figure 5.4(c) shows a component placed in such a way.

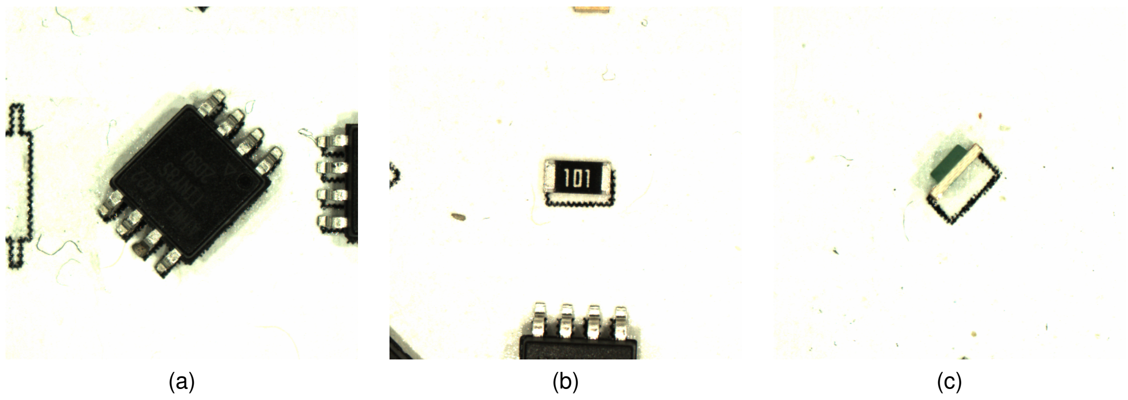


Figure 5.4: Components placed on the paper template during testing. These are samples showing components that were placed successfully (a), with an offset to the target position (b) as well as components that was placed sideways (c).

5.5.2 Results

After running five tests with 18 components each 90 individual placement results for each computer vision pipeline were accumulated. The bars in figure 5.5 represent the original (blue) and modified (red) pipeline respectively and show the distribution of all 90 components per pipeline on the categories *successfully placed*, *misaligned* (split into translation and rotation offsets) and *failures*. Looking at the test results alone there are a few observations that can be made.

Failure and success

Just looking at the amount of successfully placed components, 20 for the original and 31 for the modified pipeline, as well as the amount of complete placement failures, 26 for the original and 2 for the modified pipeline, it seems that the modified pipeline is capable of producing more reliable results and therefor more consistent prints. However, to get a better understanding of the rate of success it is important to take a thorough look at those components that were not placed correctly.

Original pipeline

Analysing the output images for the failed placement attempts on the original pipeline shows, that all failures share a similar cause. During the detection of the component in the tray image, background noise or small irregularities on the border (as seen in figure 5.6(a)) are picked up as part of the component. The resulting bounding rectangle that is supposedly enclosing the component (see chapter 3.1) is then used to calculate a mismatched center position. This way the component is either not picked up at all or, like in figure 5.6(b), picked up on the far edge of the component. The component is then placed either far off its target position or not placed at all.

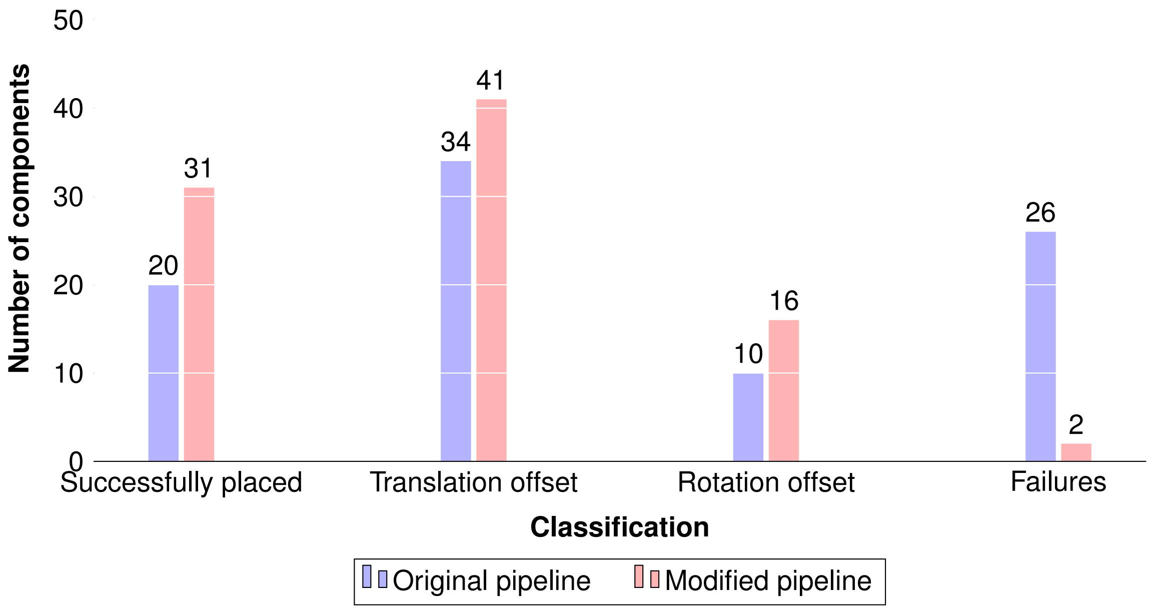


Figure 5.5: Graph showing the accumulated results of the 90 placed components per cv pipeline configuration.

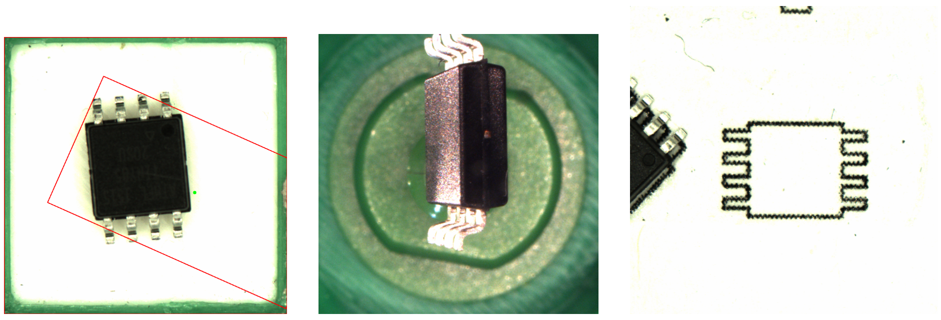


Figure 5.6: Documented pick and place process of an ATTiny85 that was picked up sideways and could not be placed on the paper template.

Modified pipeline

Because of the small amount of complete failures on the modified pipeline, all instances can be shown and discussed in detail. The first failure was caused by not being able to mechanically pick up the component inside the tray. While the position of the component was detected successfully in figure 5.7(a) the gripper image 5.7(b) shows that the resistor was not picked up and therefor not placed on the paper template 5.7(c). The second failed attempt shows that once again the component could not be picked up. In comparison to the first failure the component is located too close to the border of the tray to be detected as an individual object. The detected center point sits in the upper right corner of the tray image in figure 5.8(a), far from the actual component. In addition the components color, a green LED, makes it indifferential from the green border and increases the difficulty of being seperated from the background even further. Running the tray image seen in figure 5.8(a) through the pipeline again we can

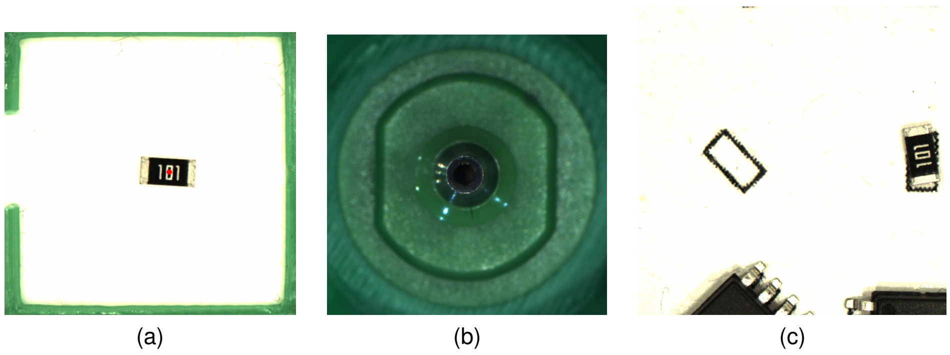


Figure 5.7: Resistor that was successfully detected in the tray image (a) but could not mechanically picked up by the gripper nozzle (b) so it could not be placed on the paper template (c).

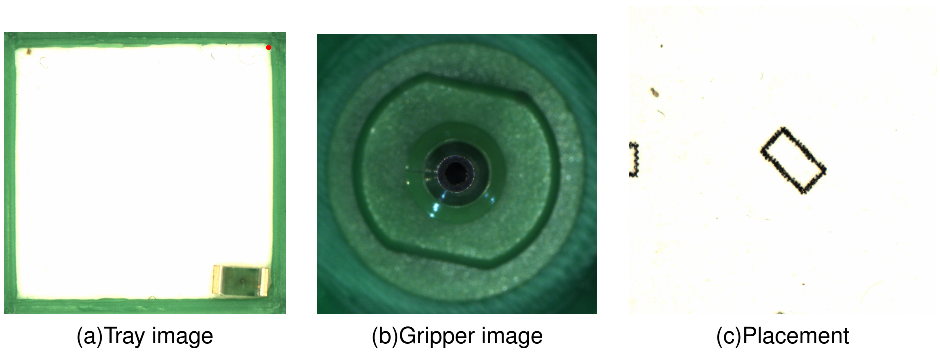


Figure 5.8: LED that is placed too close to the tray border which leads to a faulty detected component position (a). Thus, the component can not be picked up (b) or placed on the paper template (c).

observe this behavior during the binarization step (see chapter 4.6) in figure 5.9. Misplaced components like this are usually caused either by mechanical vibration of the tray during the placement procedure or by airflow caused by the 3D printer fans.

Misalignment

In both aspects, rotation and translation offsets, the modified pipeline has a higher count of misaligned components then the original pipeline. It is also worth mentioning that both pipelines peek at components with translation offsets compared to the rotation offsets.

Original pipeline

Once again the problems of translation and rotation offsets can be narrowed down to a similar cause. In both cases, parts of the background are picked up as part of the component, most prominently the black nozzle in the center of the gripper. The resulting bounding rectangle is therefor malformed and leads to offsets on either position or orientation. The test results show that in most cases these issues lead to translation offsets while the orientation is picked up with higher precision. As seen in figure 5.10(a) the remaining shape of the component, that

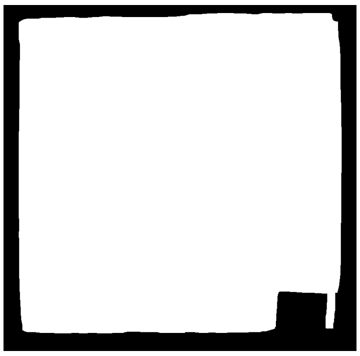


Figure 5.9: Binary version of 5.8(a) showing that the component blend in with the tray border.

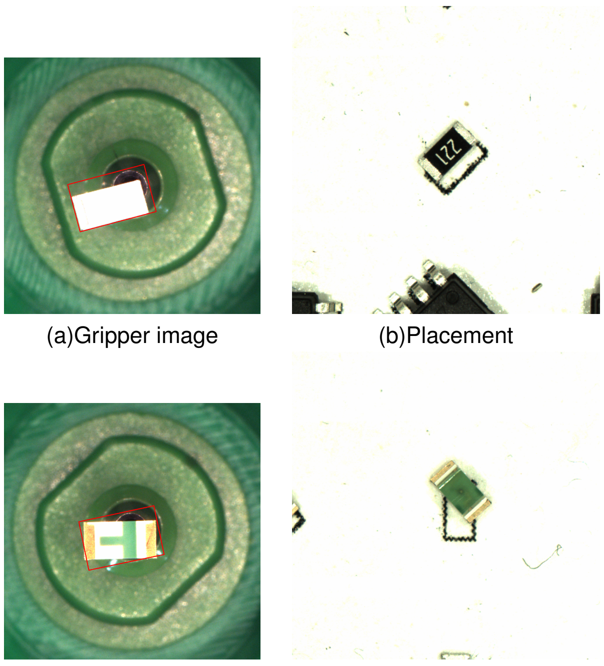
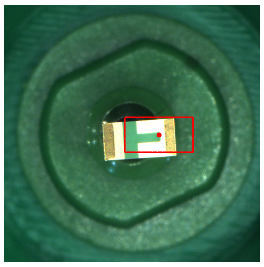


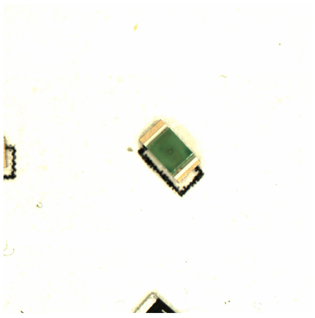
Figure 5.10: Components placed on the paper template showing translation (a,b) and rotation (c,d) offsets.

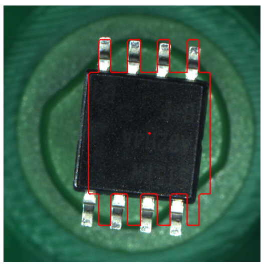
is not obfuscated by the gripper nozzle, shows enough of the components contour to align the bounding rectangle correctly while the bounds are extended to include the gripper nozzle. However, if the image contains more background noise that further obscures the contour of the component as seen in figure 5.10(c), the chances of rotational offset increase as well.

Modified pipeline

Similar to the original pipeline, the modified version has more components with translation then rotation offsets even though they use rather different approaches to calculate orientation and the center of mass. In this version the provided template is matched with detected edges of the original gripper image. Rotation and center are retrieved based on the provided template. As seen in chapter 4.5.5, the quality of the result is therefor directly connected to the quality of the template. Figure 5.11 shows that in both cases the template was matched with a


(a)Gripper image


(b)Placement on template


(c)Gripper image

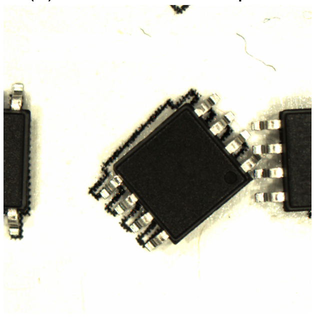
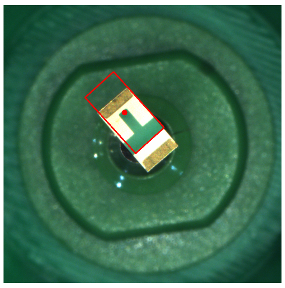

(d)Placement on template

Figure 5.11: Coponents placed on the paper template showing translation (a,b) and rotation (c,d) offsets.


(a)Plain rectangular template matched to gripper image

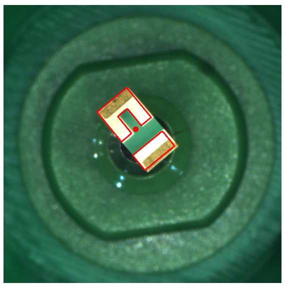

(b)Template with more details matched to same gripper image

Figure 5.12: Comparing template matching results improved by using a modified template with higher degree of details.

specific feature of the component in the picture. While the template of the LED was matched to the pattern of the polarity markings creating a translation offset, the ATTiny85 template was aligned to the slightly bend upper left leg of this micro controller. In both cases these were the position and rotation that accumulated the most votes during the recognition phase 4.5.2 and therefor the most likely candidate. Taking a look at the distribution of mismatched templates among the three components we find that the most mismatches happened on resistors with 23 out of the total of 57 misaligned components, followed by the green LED with 19 and lastly the ATTiny85 with 15 misaligned components. Both the LED and resistor use similar plain rectangular templates. The ATTiny85 however is provided with a more complex representation of its outer shape and tends to be aligned with higher precision. This correlation can be seen when the pipeline is supplied with a more matching template as seen in figure 5.12.

Runtime
During each test run the time needed to place all 18 components was documented. After completing all tests the average placement time per component was calculated based on these values. In the end the original pipeline took 10.4658 seconds per component while the modified pipeline took 10.4475 seconds. Based on these values there is no noticeable increase in placement time through the implementation of Generalized Hough Transform for template matching.

5.6 Summary

Conducting and analysing these tests showed that both pipelines have their strengths and weaknesses which are not adequately represented by numbers alone. While the bounding rectangle approach tends to work well for picking up position and orientation on gripper images as long as the components are big enough to cover the gripper nozzle, they are easily falsified by background noise. The modified pipeline on the other hand performs rather well on locating parts in the tray while the precision of which component orientation and position are detected on gripper images is highly dependent on the quality of the provided template.

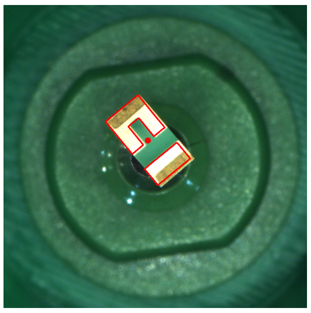


Figure 6.1: Gripper image of an LED with mapped template. The template was modified by hand and contains polarity markings.

6 Conclusion

The lack of established hard- and software that combines pick and place machines with 3D printers is a problem that the modified Kühling&Kühling Reprap 3D printer [Wasserfall, 2015] tries to solve. It shows that even though attempts at combining these machines are still experimental it is feasible and can produce consistent results. Computer vision based image processing pipelines for pick and place machines like OpenPNP [OpenPnP, 2019] or FireSight [FireSight, 2019] as well as the two approaches described in chapters 3 and 4 demonstrate that there are more then one valid approach to solve the problem of detecting SMD components in images. The use of template matching to detect known shapes in images for industrial applications like production, quality control or robotics is an established practice [Ulrich et al., 2003]. Chapter 5 shows that the use of template matching for SMD pick and place machines does have its benefits and can still be improved upon.

6.1 Further thoughts

Eventhough it is not realized in this thesis it is easy to see more practical applications for shape detection during the pick and place process. The shape detection method discussed in this chapter can be modified and used to not only find outer bounds of the components in the search image but also to read component specific markings directly from the image as seen in figure 6.1. This would make it possible to detect and align symmetrically build components with polarity or pin layout that would otherwise not be possible to detect from its outer shape alone. An added complexity is that manufacturers rarely ever follow any established standard for these markings, with bigger exceptions like the pin zero dot marking. These markings are versatile and do tend to very in shape, color and location. To realize this the Generalized

Hough Transform does not only need to be supplied with precise templates for each individual marking but also to be executed during the picking phase of the computer vision pipeline as well. The marking information then have to be saved and analysed before the component is realigned in the placement phase.

Bibliography

[Ahlers, 2015] Ahlers, D. (2015). Development of a software for the design of electronic circuits in 3d-printable objects. `https://tams-www.informatik.uni-hamburg.de/publications/2015/BSc_Daniel_Ahlers.pdf`.

[Ayob and Kendall, 2008] Ayob, M. and Kendall, G. (2008). A survey of surface mount device placement machine optimisation: Machine classification. *European Journal of Operational Research*, 186(3):893–914.

[Ballard, 1981] Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. In *Pattern Recognition*, pages 13(2):111–122.

[Brahmbhatt, 2013] Brahmbhatt, S. (2013). *Practical OpenCV*. Apress.

[Dey et al., 2014] Dey, N., Dutta, S., Dey, G., Chakraborty, S., Ray, R., and Roy, P. (2014). Adaptive thresholding: A comparative study.

[Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.

[Espalin et al., 2014] Espalin, D., Muse, D. W., Macdonald, E., and Wicker, R. B. (2014). 3d printing multifunctionality: structures with electronics. *The International Journal of Advanced Manufacturing Technology*, 72(5-8):963–978.

[FireSight, 2019] FireSight (Accessed: 11.08.2019). OpenCV based image processing pipeline. `https://github.com/firepick1/FireSight`.

[Gokulnath et al., 2018] Gokulnath, A. R., Chandrakumar, S., and Sudhakar, T. D. (2018). Open source automated smd pick and place machine. *Procedia Computer Science*, 133:872–878.

[Kruth et al., 1998] Kruth, J.-P., Leu, M., and Nakagawa, T. (1998). Progress in additive manufacturing and rapid prototyping. *CIRP Annals*, 47(2):525–540.

[Kumar et al., 2014] Kumar, R., Lal, S., Kumar, S., and Chand, P. (2014). Object detection and recognition for a pick and place robot. *Asia-Pacific World Congress on Computer Science and Engineering*.

[Li et al., 2017] Li, C., Cao, C.-Q., and Gao, Y.-F. (2017). Visual Servoing based object pick and place manipulation system. In *Shawn X. Wang (editor) Current Trends in Computer*

Science and Mechanical Automation Vol.2: Selected Papers from CSMA2016, pages 334—-341.

[Li et al., 2015] Li, Y., Zeng, G., and Wang, W. (2015). The research on end-effector position and orientation error distribution of scara industrial robot. *International Journal of Research in Engineering and Science*, 3(6).

[Macdonald et al., 2014] Macdonald, E., Salas, R., Espalin, D., Perez, M., Aguilera, E., Muse, D., and Wicker, R. B. (2014). 3D Printing for the Rapid Prototyping of Structural Electronics. *IEEE Access*, 2:234–242.

[Moyer and Gupta, 1996] Moyer, L. K. and Gupta, S. M. (1996). Smt feeder slot assignment for predetermined component placement paths. *Journal of Electronics Manufacturing*, 06(03):173–192.

[OctoPnP, 2019] OctoPnP (Accessed: 20.01.2019). OctoPrint plugin for camera based pick and place operations. `https://github.com/platsch/OctoPNP`.

[Octoprint, 2019] Octoprint (Accessed: 20.01.2019). Web interface for your 3D printer. `https://octoprint.org/`.

[OpenPnP, 2019] OpenPnP (Accessed: 20.01.2019). Open Source SMT pick and place system. `http://openpnp.org/`.

[Open Source Initiative, 2019] Open Source Initiative (Accessed: 20.01.2019). GNU Affero General Public License version 3. `https://opensource.org/licenses/AGPL-3.0`.

[Otsu, 1979] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66.

[Slic3r, 2019] Slic3r (Accessed: 20.01.2019). Open source 3D printing toolbox. `https://slic3r.org/`.

[Thesis Source, 2019] Thesis Source (Accessed: 10.09.2019). The modified OctoPnP branch described in this thesis. `https://github.com/platsch/OctoPNP/tree/ba-thesis-kolwa`.

[Treiber, 2013] Treiber, M. (2013). *An introduction to object recognition selected algorithms for a wide variety of applications*. Springer.

[Ulrich et al., 2003] Ulrich, M., Steger, C., and Baumgartner, A. (2003). Real-time object recognition using a modified generalized Hough transform. In *Pattern Recognition*, pages 36(11):2557–2570.

[Valentine et al., 2017] Valentine, A. D., Busbee, T. A., Boley, J. W., Raney, J. R., Chortos, A., Kotikian, A., Berrigan, J. D., Durstock, M. F., and Lewis, J. A. (2017). Hybrid 3d printing of soft electronics. *Advanced Materials*, 29(40):1703817.

[Wasserfall, 2015] Wasserfall, F. (2015). Embedding of SMD populated circuits into FDM printed objects. In *Proceedings of the 26th International Solid Freeform Fabrication Symposium*, pages 180–189.

[Wilkinson et al., 2019] Wilkinson, N. J., Smith, M. A. A., Kay, R. W., and Harris, R. A. (2019). A review of aerosol jet printing—a non-traditional hybrid process for micro-manufacturing. *The International Journal of Advanced Manufacturing Technology*.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Bachelorstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.
Ich bin mit einer Einstellung in den Bestand der Bibliothek des Fachbereiches einverstanden.

Hamburg, den __________ Unterschrift: _________________________