

Development and Study of an Automatic 2D Nesting System for Parts on Sheet Materials Using Computer Vision

by

Abylaikhan Myrzakhanov

Submitted to the Department of Robotics and Mechatronics
in partial fulfillment of the requirements for the degree of

Master of Science in Robotics

at the

NAZARBAYEV UNIVERSITY

Apr 2026

© Nazarbayev University 2026. All rights reserved.

Author
Department of Robotics and Mechatronics
Apr 17, 2026

Certified by
Atakan Varol
Professor
Thesis Supervisor

Accepted by
Dr. Elizabeth Arkhangelsky
Dean, School of Engineering and Digital Sciences

Development and Study of an Automatic 2D Nesting System for Parts on Sheet Materials Using Computer Vision

by

Abylaikhan Myrzakhanov

Submitted to the Department of Robotics and Mechatronics
on Apr 17, 2026, in partial fulfillment of the
requirements for the degree of
Master of Science in Robotics

Abstract

This thesis addresses the problem of automatic 2D cutting and nesting on sheet materials with irregular and unpredictable shapes. Conventional CAD/CAM-oriented nesting systems typically assume a predefined digital model of the available material and are therefore less suitable for small workshops and production environments in which leftover or nonstandard sheet material must be reused. To address this limitation, this thesis proposes an integrated system that combines computer vision, geometric modeling, heuristic nesting, and projection-based visual feedback within a unified workflow linking the physical workspace to computational layout generation.

The proposed system uses an overhead camera and four ArUco markers to define the workspace, perform metric reconstruction, and detect the available material. After camera calibration with a ChArUco board, the workspace is rectified by homography, and a hybrid segmentation pipeline is applied to extract the material contour and reconstruct its geometry in real dimensions. Based on this representation, a multi-start raster-based heuristic nesting algorithm places regular and irregular parts within the segmented material region, and the resulting layout is projected directly onto the work surface to assist the operator during cutting.

Experimental evaluation demonstrated both geometric accuracy and practical feasibility. Camera calibration achieved an RMS reprojection error of 0.1501 px, while segmentation produced an average geometric error of approximately 0.5 cm relative to ground-truth measurements. On regular benchmark families, the proposed nesting algorithm achieved average utilization rates of 99.7% on BENG, 94.6% on CGCUT, 92.8% on NGCUT, and 91.7% on WANG. On irregular-shape benchmarks, the achieved utilization rates were 63.0% on Albano, 43.91% on Jakobs1, and 56.94% on Marques. Although the packing density for irregular instances remains below that of specialized state-of-the-art methods, the algorithm consistently generated feasible non-overlapping layouts. End-to-end experiments on real segmented materials further confirmed the practical applicability of the proposed system and the usefulness of projection-based guidance in real cutting scenarios.

Overall, the results show that the proposed approach provides a feasible and

practically relevant framework for integrating real material geometry with automatic nesting in workshop environments. At the same time, they indicate the need for more robust segmentation under challenging lighting conditions and for stronger layout-improvement strategies for irregular parts.

Thesis Supervisor: Atakan Varol

Title: Professor

Acknowledgments

I would like to express my sincere gratitude to Prof. Huseyin Atakan Varol for his invaluable guidance, mentorship, and continuous support throughout the course of this research. His insightful feedback, encouragement, and academic expertise greatly contributed to the development of this thesis.

I am also deeply thankful to the Institute of Smart Systems and Artificial Intelligence (ISSAI) for providing me with a stimulating and supportive environment, valuable experience, and opportunities for both academic and personal growth. My time at ISSAI has been truly meaningful and has played an important role in shaping my research journey.

Finally, I would like to thank everyone who supported me during my master's studies and the preparation of this dissertation.

Contents

1	Introduction	11
1.1	Problem statement	12
1.2	Motivation and research gap	14
1.3	Research objectives	15
2	Literature Review	17
2.1	Nesting methods	17
2.1.1	Precise methods	17
2.1.2	Geometric foundations of irregular nesting	18
2.1.3	Constructive heuristics	19
2.1.4	Meta-heuristics and hybrid methods	20
2.2	Computer vision in nesting tasks	22
2.2.1	Classical segmentation methods	22
2.2.2	Segmentation through deep learning	24
2.2.3	Fiducial markers for workspace localization	26
2.2.4	Camera calibration	27
2.3	Comparative analysis of existing works	28
3	Methodology	32
3.1	Overall system architecture	32
3.2	Hardware and software configuration	34
3.3	Camera calibration and metric reconstruction	36
3.3.1	Intrinsic camera calibration	36

3.3.2	Workspace definition using four ArUco markers	39
3.4	Segmentation of sheet material	42
3.5	Geometric description of segmented material	46
3.6	Geometric representation of input parts	48
3.7	Nesting algorithm	50
3.8	Projection-based visual feedback	54
4	Results	56
4.1	Camera calibration quality	56
4.2	Geometric evaluation of segmentation results	57
4.3	Nesting algorithm efficiency	57
4.3.1	Evaluation of the layout of regular-shaped parts	58
4.3.2	Evaluation of the layout of irregular-shaped parts	64
4.4	Comprehensive assessment of the system	68
5	Conclusion	70
5.1	Future work	71
A		73
A.1	HybridNest pseudocode	73

List of Figures

1-1	Overview of C&P classes according to the typology of Wascher et al. (adapted from [70]).	12
1-2	Illustration of the irregular two-dimensional packing problem (adapted from [37]).	13
1-3	Example of an ArUco fiducial marker.	14
2-1	No-Fit Polygon of two irregular shapes A and B (adapted from [18]).	18
2-2	Illustration of the Bottom-Left (BL) and Bottom-Left-Fill (BLF) placement.	19
2-3	Illustration of the working principles of three metaheuristic optimization methods.	21
2-4	Examples of classical segmentation methods used in computer vision: Canny edge detection, Otsu's thresholding, and connected components analysis.	23
2-5	Examples of deep learning-based segmentation.	24
2-6	Architecture of the U-Net model (adapted from [58]).	25
2-7	Architecture of the Mask R-CNN framework for instance segmentation (adapted from [39]).	25
2-8	Examples of fiducial marker families.	26
2-9	Difference between a chessboard and a ChArUco board.	28

2-10	Workflow of the PacCAM system: (a) source material is captured, (b) the captured material is digitized, (c) the layout is arranged in an interactive packing interface, (d) the optimized layout is fabricated, and (e) the resulting parts are assembled (adapted from [61]).	29
2-11	Overview of the ProjecTables workflow: (a) a piece of scrap material is selected, (b) the material is placed on the preparation table for contour capture or (c) directly on the cutting surface, (d) CAD parts are projected onto the material for interactive positioning and cutting, and (e) the resulting model is assembled (adapted from [56]).	30
3-1	Overall architecture of the proposed system.	33
3-2	Experimental setup of the proposed system.	34
3-3	Workspace definition using four ArUco markers.	40
3-4	Example of object segmentation.	43
3-5	Example of manual creation of an irregularly shaped part on the segmented material.	49
3-6	Examples of layouts produced by the proposed automatic 2D nesting algorithm for different sheet material shapes.	50
3-7	Examples of projection-based visual guidance in the proposed automatic 2D nesting system.	55
4-1	Example of part placement produced by the proposed HybridNest algorithm for a regular-shaped benchmark instance.	59
4-2	Representative layout obtained by the proposed method for a CGCUT benchmark instance.	60
4-3	Example of the layout obtained by the proposed method for one of the NGCUT benchmark instances.	62
4-4	Example of the layout obtained by the proposed method for one of the WANG benchmark instances.	63
4-5	Layout obtained by the proposed method for the Albano benchmark instance.	65

4-6	Layout obtained by the proposed method for the Jakobs1 benchmark instance.	66
4-7	Layout obtained by the proposed method for the Marques benchmark instance.	67

List of Tables

2.1	Comparative analysis of existing research works.	31
3.1	Technical specifications of the OmniVision OV5693 camera.	35
4.1	Comparison of reprojection error with published camera calibration results.	57
4.2	Dimensional comparison of segmented objects with ground truth. . .	58
4.3	Comparison of the proposed method with representative results from the literature on the BENG benchmark family.	58
4.4	Results of the proposed method on the BENG benchmark family. . .	58
4.5	Results of the proposed method on the CGCUT benchmark family. .	60
4.6	Comparison of area utilization on CGCUT benchmark instances. . . .	61
4.7	Results of the proposed method on the NGCUT benchmark family. .	61
4.8	Results of the proposed method on the WANG benchmark family. . .	62
4.9	Comparison with representative literature results on the Albano benchmark.	64
4.10	Comparison with representative literature results on the Jakobs1 benchmark.	65
4.11	Comparison with representative literature results on the Marques benchmark.	66
4.12	End-to-end evaluation results for four consecutive placement stages on segmented material.	69

Chapter 1

Introduction

The efficiency of the placement and optimal cutting of elements, ranging from small parts to large blanks, on sheet material (plywood, fabric, cardboard, etc.) is a crucial factor. The amount of waste depends on this, and thus the cost of the production process. In medium and large-scale production, even a small optimization of cutting leads to savings in money and material. However, in small and medium-sized industries, the cutting process is completely manual. The engineer evaluates the free area on the sheet material, places the necessary elements “by eye”, takes into account the limitations and residues, and then performs cutting. Of course, this is a familiar method for an engineer because it is very simple, but if we consider it from the point of view of material usage, this is a very suboptimal method. It leads to excessive use of material and the need for additional trimming.

In the scientific literature, this problem is referred to as the “cutting & packing (C&P) problem.” This problem includes specific tasks, such as conveyor or pallet loading, packaging, and optimizing the placement of parts (nesting). In his work, Dyckhoff proposed a classification that systematizes these tasks by purpose and type, focusing on the combinatorial nature of optimal shape placement [28]. An expanded typology of C&P tasks is shown in Fig. 1-1.

Increasing attention is being paid to the optimal placement of irregular shapes. The main difficulty lies in choosing a discrete order and orientation, and in performing complex geometric checks. Geometric checks include checking intersections, account-



NU_SEDS_MS_Template_Adapted_From_MIT/images/cp_problems.png

Figure 1-1: Overview of C&P classes according to the typology of Wascher et al. (adapted from [70]).

ing for boundaries, and technical gaps. A review and analysis of existing research show high computational complexity. Therefore, precise methods and similar solutions are not very effective for industrial-scale tasks. In practice, heuristic and hybrid approaches dominate [35, 9].

1.1 Problem statement

The main practical difficulty lies in the fact that the ideal geometry of the cutting task rarely coincides with the actual geometry of the production process. Most cutting and packaging algorithms work under ideal conditions: (i) the material (sheet, cloth, plate, cardboard) has known dimensions and shape; in most cases, it is a simple shape (square, rectangle); (ii) the parts are represented as exact polygons obtained from CAD systems or vector contours; and (iii) the production parameters (clearances,

tolerances, orientation constraints) are strictly defined and unchanged.

However, in small workshops and production environments, when working with leftover material (scraps), these conditions are often violated. The work area may be partially occupied, the shape of the remaining material may be arbitrary, and local defects, shadows, and glare may complicate the accurate assessment of the available cutting area. For this reason, in practice, an engineer often returns to manual cutting of materials, even if specialized cutting software is available.



Figure 1-2: Illustration of the irregular two-dimensional packing problem (adapted from [37]).

From a computational point of view, most problems of optimal placement or cutting of two-dimensional objects belong to the class of NP-hard problems [31]. This means that as the number of parts increases, the number of possible combinations of permutations and orientations increases dramatically. An example of the irregular packing problem is shown in Fig. 1-2. If we additionally consider objects of arbitrary shape, then the algorithm requires multiple geometric checks, including checking for non-overlap and placing the object in an acceptable area with a minimum gap. As a result, as mentioned earlier, solutions to such problems rely on heuristics, metaheuristics, and hybrid schemes that allow us to obtain fairly good results in a reasonable amount of time.

1.2 Motivation and research gap

Despite the extensive theoretical basis for solving cutting problems, as mentioned earlier, one of the key problems is obtaining the boundaries of the available area where cutting takes place. A slight error in the contour or scale can lead to an accumulation of collisions and a loss of placement quality. The geometric aspects of irregular nesting are discussed in detail in the literature, which emphasizes the high complexity of developing checks for the feasibility of object placement [8].

Several studies show that the introduction of computer vision can help close the gap between the physical workspace and the digital task model. A camera can determine the boundaries and current locations of objects, and then provide these data to placement algorithms. Studies of packaging systems using computer vision showed that this approach has practical value by providing a compromise between quality and speed [37, 55].



Figure 1-3: Example of an ArUco fiducial marker.

In order to accurately model a workspace, flat markers (fiducial markers) are often used. They are effective for accurately determining the position of a plane and providing stable coordinate anchoring, even if the image has been distorted by perspective. Among such markers, ArUco-type markers stand out, which have proven

effective in determining the position of a camera or a plane, as well as for robot navigation [33]. An example of such a marker is shown in Fig. 1-3. However, in order to obtain accurate data (in millimeters), the camera must be additionally calibrated. This is necessary to take into account the intrinsic parameters and distortions of the camera. The classical camera calibration method proposed by Zhang remains one of the most popular and reliable methods [74].

After camera calibration, the most important step is segmentation. At this stage, the algorithm selects the relevant free area and the necessary objects on a given surface. Classical segmentation methods are often used for this purpose: edge detection (Canny detector [20]), binarisation (Otsu method [52]), followed by the analysis of contours or connected components (edge-tracking algorithm, topological analysis of binary images) [67]. In situations where it is necessary to determine moving objects against a stable background, background subtraction and adaptive modeling of pixel distributions (including mixtures of Gaussians) are used, which are robust to gradual frame changes.

Research notes that recent placement improvements have mostly been due to incremental improvements to existing algorithms and hybrid approaches [17]. However, the fundamental engineering challenge remains unresolved: how to ensure stable interaction between the geometry of real materials and computational algorithms for the placement of objects. This creates the need for practical solutions that (i) accurately simulate the workspace, (ii) operate on standard equipment, and (iii) provide understandable intermediate data for process control.

1.3 Research objectives

The primary aim of this work is to design, implement, and evaluate an integrated system for automatic 2D nesting of parts on sheet materials with irregular and unpredictable geometry, intended for use in small workshops and production environments where leftover or nonstandard materials must be reused without predefined CAD models.

The object of research is the process of automatic cutting layout generation on physically captured sheet materials of arbitrary shape.

The subject of research is the methods and algorithms for workspace reconstruction via computer vision, geometric contour extraction, heuristic part placement, and projection-based operator guidance within a unified end-to-end system.

To achieve the stated aim, the following specific objectives are formulated:

1. To develop and evaluate a camera calibration and homography-based workspace reconstruction pipeline capable of providing sub-pixel reprojection accuracy and centimeter-level geometric precision under real workshop conditions.
2. To design and implement a hybrid classical segmentation method for extracting the contour of sheet material from a live camera feed, and to quantitatively evaluate its geometric accuracy against ground-truth measurements on objects of varying shape and size without large computational resources.
3. To develop a nesting algorithm capable of placing both regular and irregular parts within a segmented material region of arbitrary shape, and to evaluate its effectiveness.
4. To implement a projection-based visual feedback module that maps the computed layout onto the physical work surface.

The next section of the Literature Review will analyze existing methods for solving cutting and placement problems (including irregular shapes, geometric tools, and heuristics), as well as placement methods using computer vision and feedback systems.

The Methods section will describe in detail the methodology developed and the methods used to solve the key issues raised above.

The “Results” section summarizes the quantitative and qualitative results, as well as the limitations and observations that we encountered.

In the “Conclusion” section, we summarize the results, draw the main conclusions, make recommendations for subsequent research, and indicate the directions of our future work.

Chapter 2

Literature Review

Questions of how to better position and cut objects have always been closely related to two key areas: combinatorial optimization and computational geometry. In CAM/CAD systems, this is reflected in two aspects mentioned earlier: firstly, the development of algorithms that are able to create efficient and dense layouts while taking into account the given conditions; secondly, the connection with the real production environment. Next, we analyze the most common approaches and their limitations, especially when working with irregular objects and leftover materials.

2.1 Nesting methods

2.1.1 Precise methods

Precise methods such as branch-and-bound, integer linear programming (ILP), and dynamic programming are able to find optimal solutions. However, their applicability is limited to small problem instances. A study by Alvarez-Valdes et al. [2] shows that even with restrictions on the orientation of objects, the solution time increases exponentially as the number of objects increases. The practical application of precise methods does not exceed, as a rule, 15–20 irregular objects, which makes them ineffective for industrial scenarios where the number of parts can reach hundreds [8].

2.1.2 Geometric foundations of irregular nesting

The key difference between irregular placement and two-dimensional packing of rectangular objects is that the efficiency and speed of algorithmic solutions largely depend on geometric factors. In the work of Bennell et al. [8], the authors note that the high complexity of developing a geometric “core” is one of the main reasons why research in the field of irregular shape placement has historically received less attention than research on classical algorithms for the placement of rectangular objects.

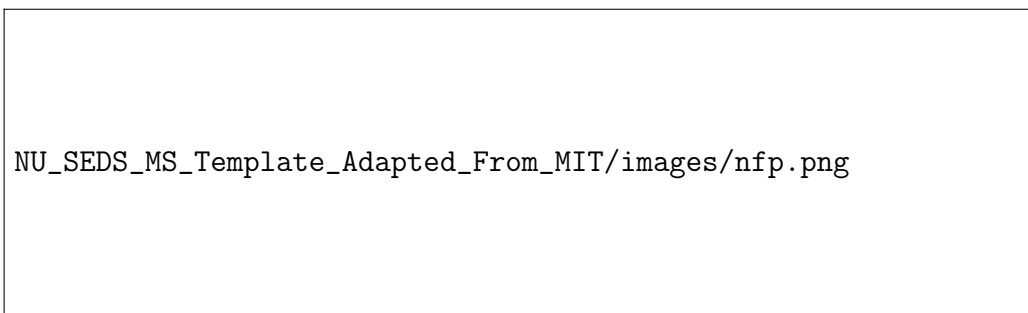


Figure 2-1: No-Fit Polygon of two irregular shapes A and B (adapted from [18]).

The No-Fit Polygon (NFP) method is considered a key tool for optimizing the placement of irregular objects. For a pair of shapes A and B , the method is based on the set of positions of B relative to A at which they touch but do not overlap. An example of the NFP for two irregular shapes is shown in Fig. 2-1. This simplifies the task of checking overlap by reducing it to checking whether an object belongs to a feasible area, thereby helping to generate placement options efficiently. This algorithm operates in $O(n)$, which is much faster than direct pixel-by-pixel intersection analysis. However, the literature notes that the implementation of NFP is difficult, because there may be empty spaces in the form of “holes,” narrow “locks,” internal edges, and numerical errors. In the work of Burke et al. [18], the authors proposed a robust method for implementing NFP. This work demonstrates success in handling complex, degenerate cases, including the occurrence of holes and mutual concavities.

There are alternative approaches to calculating NFP based on the application of Minkowski sums. Bennell et al. [10] describe a more mathematical approach, but at the same time indicate significant practical difficulties in its implementation when

working with real polygons.

In addition, other geometric methods are used: trigonometric intersection tests, ϕ -functions, raster approaches (mask, raster point), and polygon decomposition. These approaches are often integrated into hybrid systems, where, for example, coarse raster filtering is combined with precise polygonal verification. The resilience of the system to problems related to inaccurate contour segmentation and small-scale errors depends on the choice of geometric approach [9, 37].

2.1.3 Constructive heuristics

In real-world applications, constructive methods dominate. In these methods, new objects are placed in sequential order according to established rules. The key advantages are high speed, predictability of results, and the ability to integrate production constraints: acceptable turning angles, control over technical gaps, and preferences in object orientation.

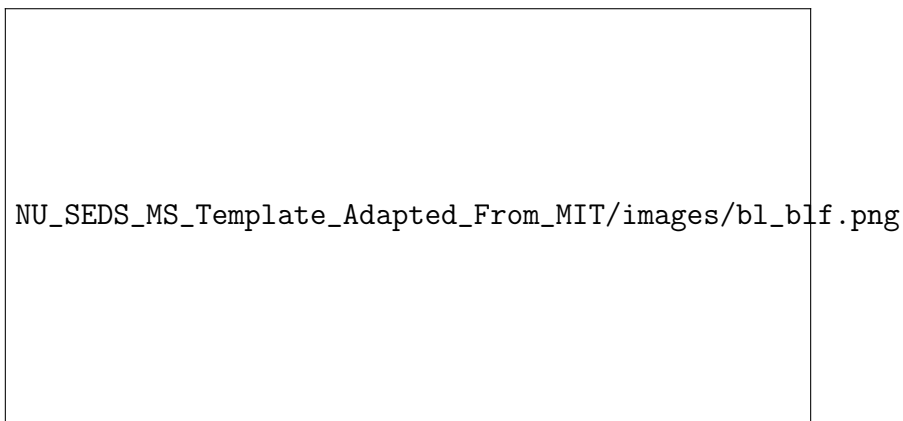


Figure 2-2: Illustration of the Bottom-Left (BL) and Bottom-Left-Fill (BLF) placement.

Numerous heuristics are based on NFP. Examples include the Bottom-Left (BL), Bottom-Left-Fill (BLF), and Largest-Area-First heuristic methods. BL/BLF works by finding a valid position for the next object in the lowest-left part of the free space; see Fig. 2-2. This method provides high packing density for rectangular objects, but the result is suboptimal for complex irregular shapes [4].

In the Largest-Area-First method, the largest area is placed first, followed by the smaller areas. In Hopper’s work, the author used this method for rectangular placement and achieved an area utilization rate of up to 85%. However, using this method with irregular shapes results in a reduction in efficiency to 65–70% due to difficulties in filling angular gaps [40].

The Shelf/Level and Skyline methods divide the space into horizontal levels. At each level, the objects are arranged from left to right until there is no more space in width. When there is no more space, a new higher level is opened. The height is fixed at each level. This simplifies the geometry, and therefore such methods perform well in terms of speed [5]. Examples of level-oriented algorithms are Next Fit Decreasing Height (NFDH) and First Fit Decreasing Height (FFDH). The idea of the NFDH algorithm is to sort the rectangles by non-increasing height, then place them in the current level until the free space is exhausted [44]. The idea of FFDH is to try to insert a rectangle into the lowest level where it fits in width; a new level is created only if this object does not fit anywhere. The problem with these approaches is that they work only with regular shapes, and under strong contour noise they may produce high error, since the algorithm can detect a bad configuration of voids early and it will be difficult to fix it.

2.1.4 Meta-heuristics and hybrid methods

Metaheuristic approaches search for possible solutions stochastically, but do not guarantee optimality. The advantage of metaheuristics is the ability to escape local minima. This is especially useful if we need to place a large number of parts. Metaheuristic methods include Simulated Annealing (SA), Genetic Algorithms (GA), and Tabu Search (TS). The workflows of these algorithms are illustrated in Fig. 2-3.

Genetic Algorithms (GA) model the search process in the form of biological evolution. The idea is that every solution to a problem is an individual in the population. Each individual is encoded into a chromosome (a string of numbers), for example, a random list of parts and their orientations. After that, two operations are applied: (i) crossover, where two parent chromosomes are taken and a new offspring chromosome

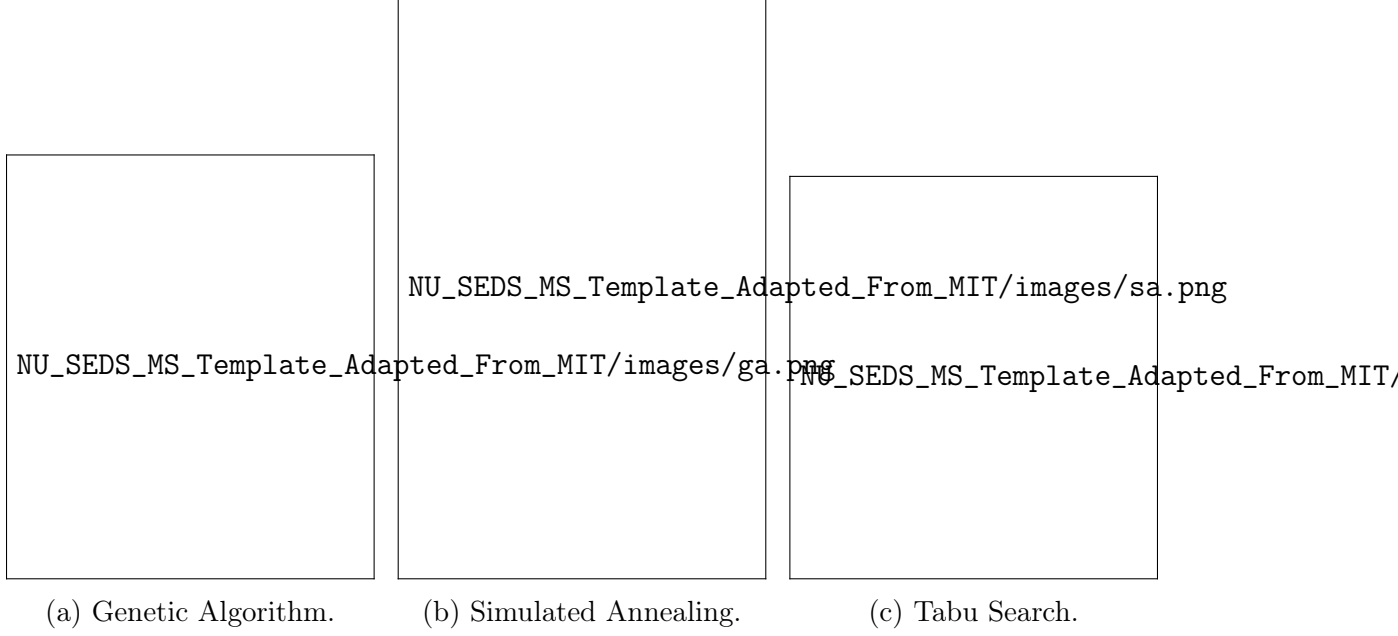


Figure 2-3: Illustration of the working principles of three metaheuristic optimization methods.

is obtained from them, and (ii) mutation, where one or several values in the chromosome change randomly. This approach helps the algorithm avoid getting stuck in a local optimum.

In the work of Jakobs [43], the author was one of the first to apply GA to 2D nesting and, as a result, demonstrated a steady improvement in solution quality from generation to generation, but the computational cost of evaluating each chromosome remains high.

Burke et al. [19] suggested an improvement so that the set of chromosomes is immediately checked by the NFP algorithm to avoid defective chromosomes.

Simulated Annealing (SA) starts with randomly placed parts at the beginning, and then a small random change is made at each iteration. If the new solution is better than the previous one, then it is used. In the work of Gomes et al. [36], the application of this algorithm with Linear Programming (LP) is considered; it is used as a local optimizer. If the parts intersect after the iteration, then the algorithm tries to push them apart. If the layout has become acceptable, LP tries to compact it. The SA and GA algorithms are similar, but the main difference is that GA takes the

best of different solutions and combines them with each other, while SA gradually improves one solution.

Tabu Search (TS) is similar in structure to SA; it also improves the current solution and iteratively introduces improvements. The difference is that the algorithm keeps a list of forbidden moves, and those changes that have been made recently become temporarily prohibited from repeating. This allows the algorithm not to return to the same search path and helps it explore new solutions. A comparative analysis in the work of Leung et al. [47] showed that TS is, on average, superior to SA in terms of the quality of the final solution.

2.2 Computer vision in nesting tasks

The use of video cameras and personal computers in the 1990s gave rise to the application of computer vision in the industrial sector. Since that time, the video camera has ceased to be just a device that records video; now it is perceived as a device capable of transmitting accurate geometric information about objects or the working environment to algorithms in real time.

In cutting and placement tasks, computer vision is an essential tool for connecting the physical workspace and computational algorithms. Segmentation methods are used to obtain data about the workspace (which space is available and which objects are located in the workspace). Segmentation is a set of methods that analyze a frame and highlight objects of interest and area boundaries in the frame. Examples of classical segmentation methods are shown in Fig. 2-4.

2.2.1 Classical segmentation methods

The Canny operator [20] allows us to find the boundaries of objects and is one of the most widely used methods due to its good balance between sensitivity and noise suppression. The Canny algorithm works in several steps: (i) smoothing the frame using a Gaussian filter to reduce noise, (ii) calculating the intensity gradient using the Sobel method, (iii) non-maximum suppression to thin the boundaries, and (iv)

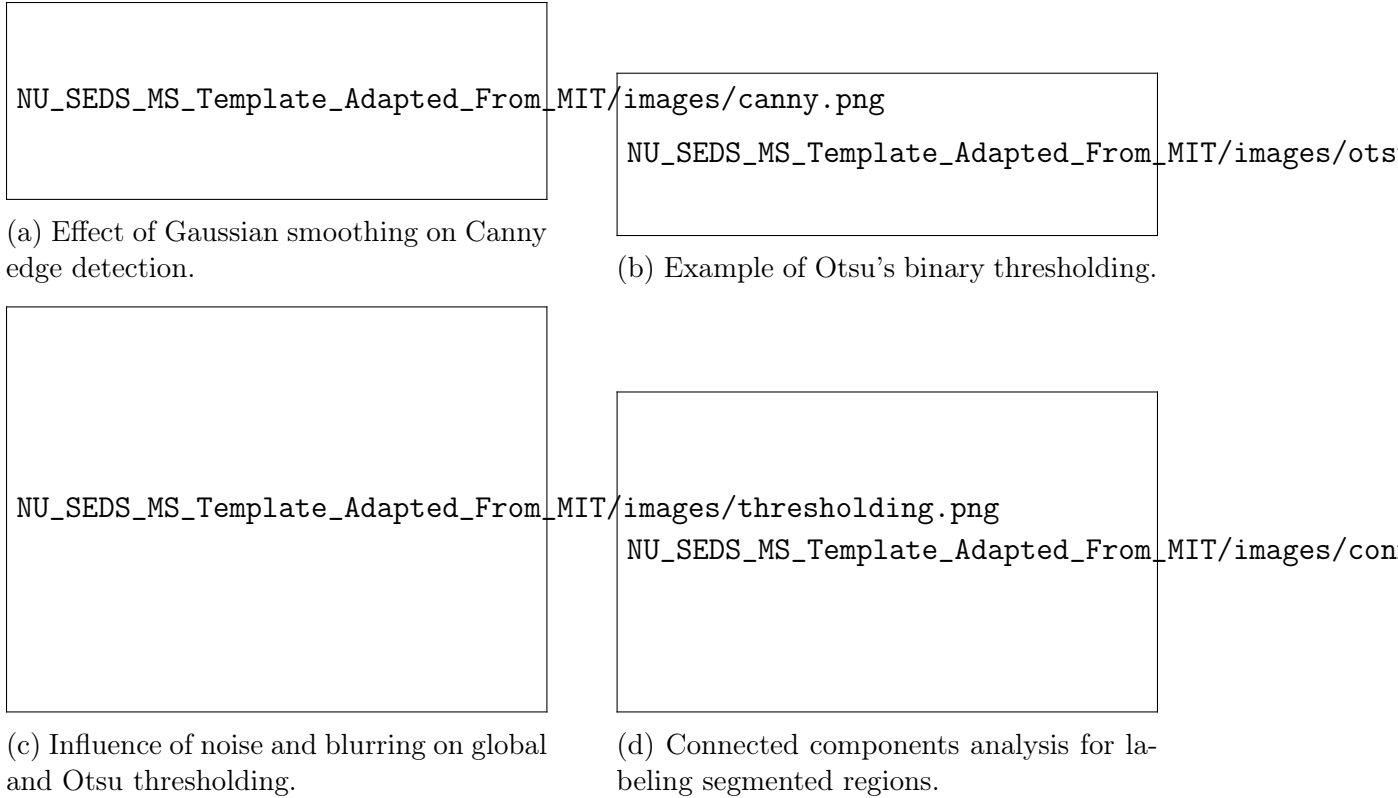


Figure 2-4: Examples of classical segmentation methods used in computer vision: Canny edge detection, Otsu's thresholding, and connected components analysis.

double-threshold processing with hysteresis to separate weak and strong boundaries, followed by edge linking. The smoothing parameters and the setting of the upper and lower thresholds significantly affect the segmentation result. When processing sheet materials with different textures, such as plywood or fabric, additional adaptive adjustment is required [21].

The method that automatically adjusts the pixel separation threshold between the foreground and background is called the Otsu binarization method [52]. This method is effective for segmenting objects against a homogeneous background. However, with uneven lighting, the Otsu threshold gives fragmented segmentation. In this case, as well as with the Canny operator, an adaptive method can be used, namely adaptive binarization [62].

Background subtraction is an alternative approach in which the algorithm receives a frame from an “empty” workspace and highlights the changes as potential objects.

The most common implementation of this approach is the Mixture of Gaussians (MoG), which was described by Stauffer et al. [66]. In this implementation, each pixel is described by a mixture of K Gaussians with adaptive parameters. The disadvantage of this implementation is instability under rapid changes in the workspace.

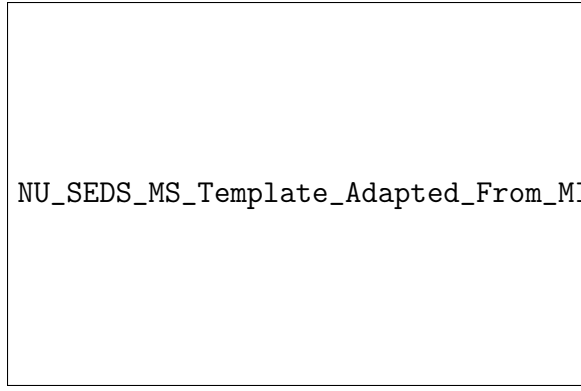
There is also the Connected Components Analysis (CCA) method, in which the algorithm binarizes the frame, that is, each pixel becomes either white or black. The algorithm goes through all the white pixels and groups them. Each such group is a separate object in the image. However, this method gives a rough idea of the object, so the next step is to apply Contour Analysis (CA), which was described in the work of Suzuki et al. [67]. The algorithm builds a hierarchy of contours, which helps to identify the external contours of objects and internal cavities. Next, the resulting contours are approximated by polygons using the Douglas–Peucker method [26]. The algorithm recursively removes unnecessary points, keeping only those that deviate significantly from a straight line.

2.2.2 Segmentation through deep learning



Figure 2-5: Examples of deep learning-based segmentation.

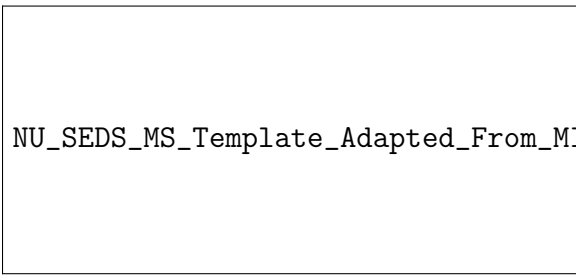
Due to the popularity of deep learning methods in recent decades, classical segmentation methods have been actively replaced. The deep learning approach allows us to detect objects and highlight their contours without explicitly setting the parameters manually. Examples of segmentation through deep learning are shown in Fig. 2-5.



NU_SEDS_MS_Template_Adapted_From_MIT/images/unet.png

Figure 2-6: Architecture of the U-Net model (adapted from [58]).

The U-Net architecture, which was described in the work of Ronneberger et al. [58], has become one of the most popular methods for pixel-by-pixel segmentation. U-Net has an encoder that consistently reduces spatial resolution by extracting features, and the decoder restores resolution using skip connections to preserve fine details of the frame. The overall architecture of U-Net is illustrated in Fig. 2-6. This method is superior to classical segmentation methods such as Canny [20] and Otsu [52] in uneven lighting conditions.



NU_SEDS_MS_Template_Adapted_From_MIT/images/maskrcnn.png

Figure 2-7: Architecture of the Mask R-CNN framework for instance segmentation (adapted from [39]).

The Mask R-CNN architecture proposed by He et al. [39] solves the problem of instance segmentation. That is, it not only marks pixels as belonging to an object, but also separates individual instances of objects of the same class. A schematic overview of the Mask R-CNN framework is presented in Fig. 2-7.

Despite the excellent results, deep learning has a number of limitations: (i) the model requires a large labeled dataset (hundreds or thousands of images with manually drawn object masks), (ii) the models do not generalize well to new conditions,

and (iii) computational requirements are significantly higher than when using classical segmentation methods. Using Mask R-CNN in real time requires a Graphics Processing Unit (GPU), while Canny, MoG, and CA work stably on a regular Central Processing Unit (CPU) at 1080p resolution. At the same time, the high resolution of the video camera affects the accuracy of determining the size of objects.

2.2.3 Fiducial markers for workspace localization

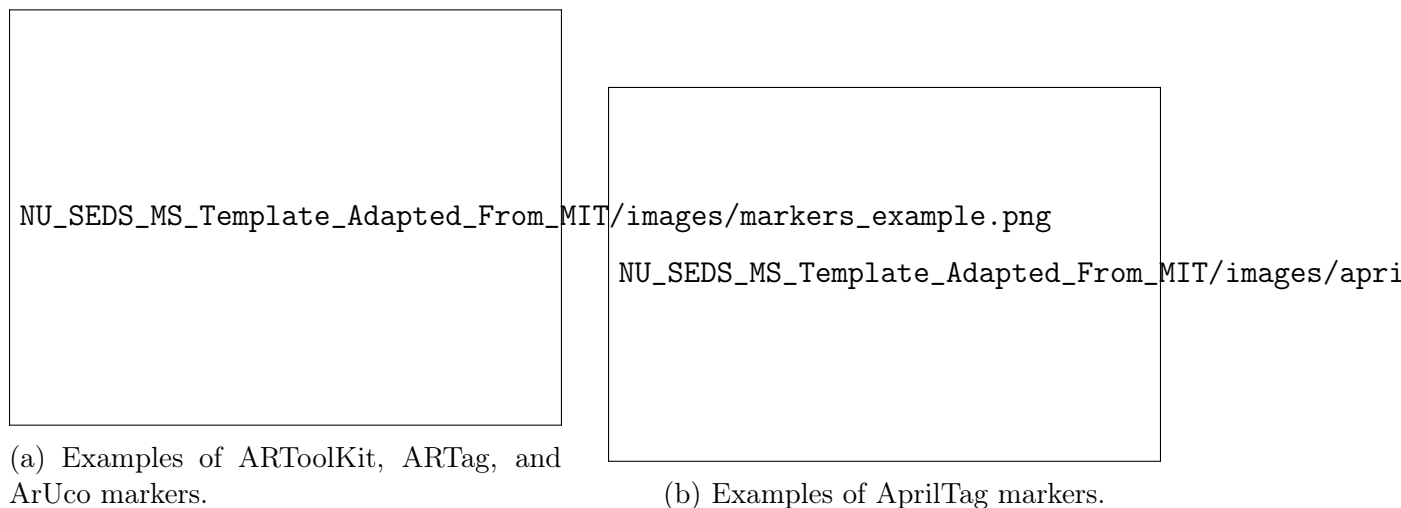


Figure 2-8: Examples of fiducial marker families.

Fiducial markers are specialized artificial visual markers. They serve as reference patterns for computer vision algorithms to determine three-dimensional position, orientation, and scale from two-dimensional images. Among the various families of markers, such as ARToolKit, ARTag, and AprilTag, shown in Fig. 2-8, ArUco markers [33] are the most popular; an example of such a marker is shown in Fig. 1-3. This marker is a square with a unique binary identifier encoded in a matrix pattern. The advantage of ArUco markers is their resistance to perspective distortion. Markers are detected at image angles up to 60–70 degrees from the normal to the plane.

In the work of Romero-Ramirez et al. [57], the authors demonstrated that ArUco marker detection surpasses AprilTag in processing speed, up to 60 frames per second (FPS) on the CPU for 1080p images. The ArUco marker is found in the frame in several stages: (i) the image is binarized to find quadrilateral contours, (ii) after that,

the bit matrix inside the found squares is read, (iii) the results are compared with the initially known markers, and (iv) if a match is found, the marker is considered detected, and the algorithm accurately determines its coordinates.

If we know the actual physical distance between the markers in millimeters and their position in the image, then we can estimate a homography. A homography is a transformation matrix that converts pixel coordinates into real coordinates on the plane; thereby, we can determine the exact dimensions of objects in the frame.

This matrix describes the projective relationship between the plane of the scene and the frame. Also, the transformation matrix is recalculated every frame, so small vibrations or offsets are automatically compensated.

2.2.4 Camera calibration

However, homography alone is not enough to obtain a metrically correct representation of the workspace. Any camera has an optical system that introduces distortion. Because of this, straight lines at the edges of the image may look slightly curved, and objects in the vicinity may appear larger than in the center. Therefore, when working with flat fiducial markers, calibration is especially important, since a small error in the camera parameters leads to an accumulation of errors in estimating the position of markers, coordinates, and exact dimensions [74, 45]. To take these nuances into account, camera calibration is required, in which the internal parameters of the camera (focal length, coordinates of the principal point) are evaluated and distortion coefficients (radial and tangential) are calculated.

The classical method described in the work of Zhang [74], which is based on obtaining a large number of images by observing a flat template (checkerboard) in various orientations, allows calculating camera parameters by minimizing projection error.

An alternative to the chessboard is the ChArUco board, which is a chessboard with ArUco markers on each black square; see Fig. 2-9. This allows the calibration algorithm to reliably identify corner points when the board is partially occluded. Compared to the classic chessboard, a ChArUco board requires fewer images and is

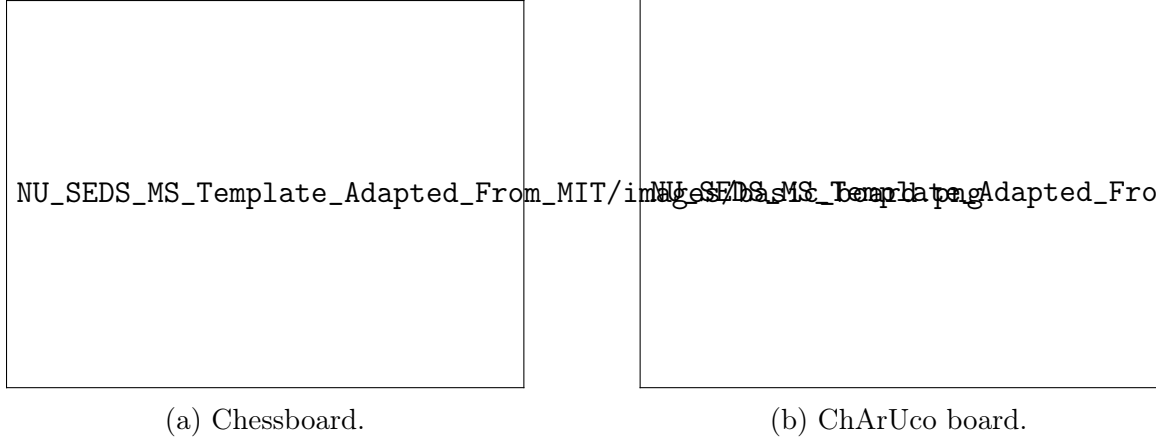


Figure 2-9: Difference between a chessboard and a ChArUco board.

resistant to glare. In a study by Romero-Ramirez et al. [57], the authors showed that for a camera with 1080p resolution and a focal length of 4–8 mm, 20–30 images will be enough to obtain a projection error of less than 0.5 pixels. This means that the metric accuracy is about 1–2 mm with a shooting height of 2–2.5 m. After calibration, we need to perform image undistortion, after which we can calculate the homography of the workspace. These methods significantly improve the accuracy of homography estimation.

2.3 Comparative analysis of existing works

As part of the analysis of scientific research, several research directions can be identified that are relevant to the context of our system under development. It should be emphasized that existing solutions in scientific research focus on optimizing cutting algorithms, either by providing a user-friendly interface or by integrating computer vision. However, combining solutions from different areas into a single system is less common.

The first direction includes CAD/CAM-oriented nesting systems. Examples of this direction are the works by Xie et al. [73] and Weng et al. [72]. The first paper presents a computer-aided nesting (CAN) system. This scientific paper combines several algorithms: rectangle enclosure, BL placement, and GA to increase the material

utilization rate. In the second system, the parts are placed using the BLF method with an additional discrete enumeration of the rotation angles. The advantage is that these systems integrate well into the production process, as the parts are obtained from CAD systems. However, the main limitations are that the geometric parameters of the workspace and the dimensions and models of the parts in CAD must be known and prepared in advance. This is a big disadvantage for small workshops that often work with leftover materials.

NU_SEDS_MS_Template_Adapted_From_MIT/images/pac_cam.png

Figure 2-10: Workflow of the PacCAM system: (a) source material is captured, (b) the captured material is digitized, (c) the layout is arranged in an interactive packing interface, (d) the optimized layout is fabricated, and (e) the resulting parts are assembled (adapted from [61]).

The second area includes systems using computer vision. Examples of this trend are the works of Bouganis et al. [14] and Saakes et al. [61]. In the first paper, the authors presented a system in which the algorithm first receives a frame of a sheet and automatically chooses where to place the next part/object, so that the parts do not intersect and the occupied surface area is minimal. It is important to note that the parts do not overlap during the capture of the frame and the background color differs well from the parts. The packing density metric was used as an estimate. Packing density is the ratio of the total area of the placed objects to the total area. The higher the value, the more effective the layout is. The main value of this work is that the part is not a CAD contour, but a set of pixels in a binary image. In the second paper, the authors proposed the PacCAM system. A general workflow of the PacCAM system is illustrated in Fig. 2-10. They used two methods to obtain a workspace. The first method is to view the workspace through a calibrated camera mounted above the desktop. The authors note a mean error of 0.83 mm with a standard deviation of 0.5 mm, which indicates good accuracy in obtaining metric data. The second method uses a photodiode that captures the reflection of a low-power laser from the surface

of the sheet. This method is more accurate, but slower. The limitation of this work is that it does not solve the problem of fully automatic placement of parts based on the captured scene. The user decides how to arrange the parts. The value of this work is to reduce the gap between physical and digital interaction by obtaining the shape of the sheet or the remnants of the material and using them when planning the layout.

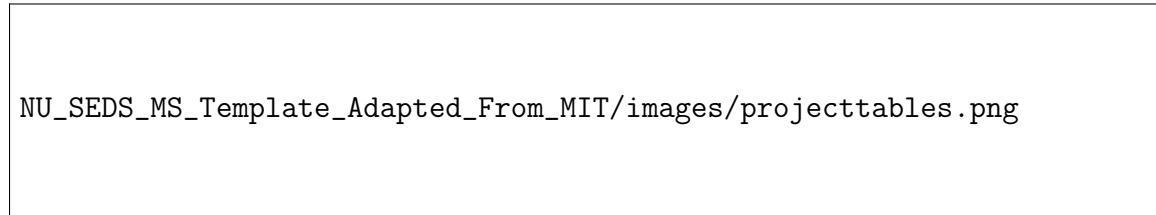


Figure 2-11: Overview of the ProjecTables workflow: (a) a piece of scrap material is selected, (b) the material is placed on the preparation table for contour capture or (c) directly on the cutting surface, (d) CAD parts are projected onto the material for interactive positioning and cutting, and (e) the resulting model is assembled (adapted from [56]).

The third direction is visual guidance of the user directly on the work surface. In the work of Ojer et al. [51], the authors proposed an Augmented Reality (AR) system for manually assembling electronic components on printed circuit boards. The proposed system uses a projector and a calibrated camera. The projector displays the work instructions on the Printed Circuit Board (PCB), and the camera is used to capture the work scene, track the PCB, and recognize components. The authors note that screen-based visualization often forces the user to constantly shift their gaze between the screen and the workspace. The presented system can speed up the work and reduce the number of errors. The second scientific work that relates to this field is the work of Rasmussen et al. [56]. The authors demonstrated the ProjecTables system; its workflow is illustrated in Fig. 2-11. The system uses a calibrated camera and a projector. First, the camera captures a sheet or remnants of material, then the Canny edge detection algorithm [20], morphological closure, and contour selection are applied. Then the user independently places the necessary parts on the work surface using the system interface. After that, the projector projects the found boundaries directly onto the physical material, then the system projects a CAD model on a 1:1

scale. After that, the user manually places the part in the desired position using the computer mouse.

Table 2.1: Comparative analysis of existing research works.

System / Work	Auto nesting	Vision Workspace capture	Use of leftover material	Proj./AR guidance	CAD/CAM integration
Xie et al. [73]	+	−	−	−	+
Weng et al. [72]	+	−	±	−	+
Bouganis et al. [14]	+	+	±	−	−
Saakes et al. [61]	−	+	+	−	±
Ojer et al. [51]	−	±	−	+	−
Rasmussen et al. [56]	−	+	+	+	+

A comprehensive analysis of existing works, summarized in Table 2.1, shows that each of the considered areas only partially contributes to solving the overall problem. At the same time, their integration into a single system remains incomplete. Thus, it can be concluded that there is a research gap between obtaining a workspace in real time, algorithms for the placement of objects, and effective user interaction. The current work is aimed at reducing this gap by combining the mentioned components into a single system.

Chapter 3

Methodology

3.1 Overall system architecture

The proposed work is a specialized system designed to automatically place parts on sheet material with visual feedback for the operator who adjusts the material. The system consists of five components: (i) a camera, (ii) a projector, (iii) a computing module, (iv) a desktop with four ArUco markers, and (v) an information module containing information about the input parts. Using these components in a single system allows us to move from the digital design of cutting material to physical implementation.

The camera is positioned above the desktop surface and captures images of the workspace in real time. This position of the camera is very convenient, as this arrangement allows us to obtain information about the position of the markers, the boundaries of the workspace, and the shape of the material placed on the table without interfering with the operator. There are four ArUco markers[33] on the edges of the desktop. They form a workspace; in our example, the workspace was a rectangle. Their role is to set the reference coordinate system and define the boundaries of the workspace. Markers of this type are easily detected in the frame; they are responsible for linking calculations to the real plane of the table and for obtaining the orientation and position of objects in space. They are also very convenient for creating a workspace. If we need to change the shape or size of the workspace, we just need to

move the markers to the desired position.

The computing module is a key component and plays a central role in the architecture of the system. Its functions are to: (i) receive a video stream from a camera in real time, (ii) process frames, (iii) identify the workspace, (iv) segment sheet materials or other objects, (v) extract geometric characteristics, and (vi) run an algorithm to automatically place parts. The input data describing which parts should be placed are given in the form of simple geometric shapes (square/rectangle/circle) or complex geometric parts with a prepared geometric description. Based on the input data, the computing module places the parts inside the detected material, taking into account its shape and available area. After the placement is completed, the resulting layout scheme of the parts is converted from the internal coordinate system to the coordinates of the actual workspace and projected using a projector. The projector is a feedback device. It projects the result of the placement of the parts onto the surface of the table, directly onto the cutting material, at a scale of 1:1.

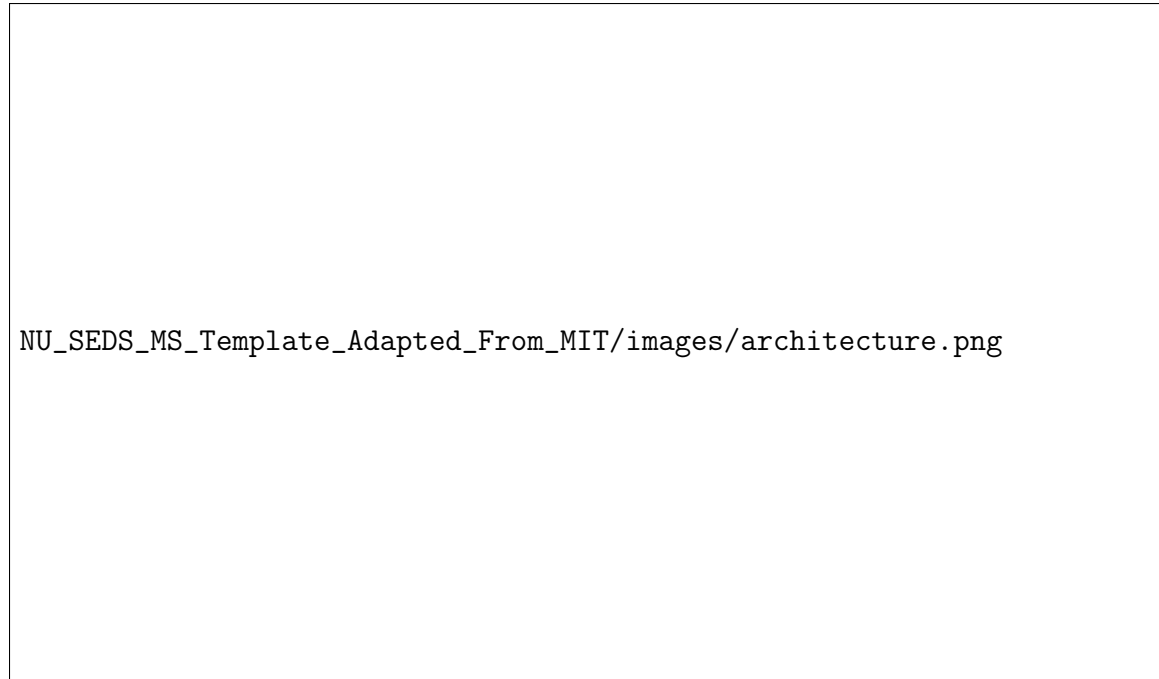


Figure 3-1: Overall architecture of the proposed system.

The general logic of the specialized system consists of several sequential stages, shown in Fig. 3-1: (i) marker detection, (ii) formation of a normalized workspace, (iii)

segmentation of sheet material, (iv) extraction of geometric contours of the material, (v) obtaining information about the parts to be placed, (vi) running an algorithm for compact part layouts, and (vii) projecting the result onto a physical work surface. As a result, the operator sees how the necessary parts should be cut on the sheet material with the least amount of waste. Unlike many other studies, this paper provides a direct link between the computational layout and the physical workspace. Thus, the proposed specialized system combines computer vision, nesting algorithms, and projection visualization.

3.2 Hardware and software configuration



Figure 3-2: Experimental setup of the proposed system.

The specialized system includes a workspace, in our case a 1600×450 mm table, an OmniVision OV5693 camera, a TouYinger X21 projector with a maximum resolution

Table 3.1: Technical specifications of the OmniVision OV5693 camera.

Parameter	Specification
Pixels	5 MP
Sensor	OV5693
Interface	USB 2.0
Sensor size	1/4 inch
Aperture (F)	2.1
Focal length (EFL)	2.5 mm
Field of view (FOV)	135° (D), 95° (H), 70° (V)
Focusing	Autofocus
Format support	MJPEG, YUY2
Static image resolution	2592 × 1944
Video recording	MJPEG:
	2592 × 1944 @ 15 FPS
	1920 × 1080 @ 15 FPS
	1280 × 720 @ 15 FPS
	640 × 480 @ 15 FPS
	YUY2:
	2592 × 1944 @ 3 FPS
	1920 × 1080 @ 5 FPS
Audio	1280 × 720 @ 10 FPS
	640 × 480 @ 15 FPS
Audio	Onboard dual microphones
Operating voltage	5 V ± 5%
Operating current	500 mA
Camera size	18.00 × 18.00 × 19.69 mm
Module size	18.00 × 40.00 mm

of 1280×720 px, a personal computer that served as the computing module, and four ArUco markers measuring 120×120 mm. A general view of the experimental setup used to implement the proposed system is shown in Fig. 3-2. The camera and projector were fixed above the desktop at a height of 2.5 m from the floor. The camera has a CMOS sensor with a resolution of 5 MP, autofocus, and a maximum video recording resolution of 2592×1944 at 5 FPS. Detailed camera specifications can be found in Table 3.1. However, to obtain a real-time video stream without delay,

a resolution of 1280×720 at 15 FPS was used. All experiments were conducted on a personal computer with an Intel Core i9-7900X CPU, which contains 10 computing cores and 20 threads with a frequency of 3.30 GHz, and an NVIDIA GeForce RTX 2080 Ti GPU with 11 GB of VRAM and 120 GB of RAM.

The software part of the system was written in the Python programming language. The OpenCV library was used for computer vision, and NumPy was used for computational operations. To speed up calculations, a JIT compiler was used to translate Python code into fast machine code. Multithreading was also used, which allowed several parallel workflows to search for an effective layout of the material, and GPU acceleration was used for mathematical operations, although this is optional in the system.

In general, the chosen configuration is focused on solving three tasks: (i) obtaining sufficiently accurate geometric contours of objects, (ii) finding the optimal layout scheme with a small delay, and (iii) providing visual feedback.

3.3 Camera calibration and metric reconstruction

3.3.1 Intrinsic camera calibration

The camera was calibrated to obtain the geometry of the scene correctly. The purpose of the calibration was to determine the internal parameters of the camera, including the focal length along the image axes, the coordinates of the principal point, and optical distortion coefficients. A ChArUco board was used for calibration (see Fig. 2-9b), which combines a chessboard pattern and ArUco markers. Using this type of board ensures stable calibration even with partial visibility of the board and allows us to obtain precisely localized corner points suitable for subsequent optimization of camera parameters[33, 49]. The ChArUco calibration board consisted of 6×9 cells of 29 mm and ArUco markers measuring 21 mm. A *DICT_4X4_50* dictionary was used for detection[33].

The mathematical calibration model was based on the standard pinhole camera

model[74]. The relationship between a 3D point in the scene, $\mathbf{P} = (X, Y, Z)^\top$, and its projection onto the image plane, $\mathbf{p} = (u, v)^\top$, is described by

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad (3.1)$$

where s is a scale factor, $\mathbf{R}$ and $\mathbf{t}$ are the camera extrinsic parameters, and $\mathbf{K}$ is the camera intrinsic matrix. In general, the intrinsic matrix $\mathbf{K}$ is written as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.2)$$

where f_x and f_y denote the effective focal lengths expressed in pixels, and (c_x, c_y) denote the coordinates of the principal point. These parameters are estimated during intrinsic calibration. The adopted formulation follows the standard camera calibration model widely used in computer vision[74].

Due to lens distortion in the camera imaging system, radial and tangential distortion coefficients were additionally estimated during calibration[16]. For normalized image coordinates in the camera coordinate system,

$$x = \frac{X_c}{Z_c}, \quad y = \frac{Y_c}{Z_c}, \quad (3.3)$$

and

$$r^2 = x^2 + y^2, \quad (3.4)$$

the radial distortion component can be written as

$$\begin{aligned} x_r &= x \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6}, \\ y_r &= y \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6}, \end{aligned} \quad (3.5)$$

while the tangential distortion component is given by

$$\begin{aligned}x_t &= 2p_1xy + p_2(r^2 + 2x^2), \\y_t &= p_1(r^2 + 2y^2) + 2p_2xy.\end{aligned}\tag{3.6}$$

The distorted normalized coordinates are then computed as

$$x_d = x_r + x_t, \quad y_d = y_r + y_t,\tag{3.7}$$

and finally converted into pixel coordinates according to

$$u = f_x x_d + c_x, \quad v = f_y y_d + c_y.\tag{3.8}$$

Using a rational distortion model allows us to accurately account for nonlinear distortions, especially with noticeable perspective deformation and a wide field of view[23].

During camera calibration, 50 images were taken at different positions and tilts relative to the camera. To ensure high-quality calibration, the frames passed the following selection criteria: (i) at least 14 ChArUco corners had to be detected in the frame, (ii) the Laplacian-variance sharpness score had to be at least 25[53], (iii) the angular offset had to satisfy the specified threshold, (iv) the board offset relative to the previous frame had to be at least 25 px in order to avoid repeated frames, and (v) the frame was considered suitable if a stable position of the board was maintained for 1 second. Such filtering made it possible to exclude blurred, too similar, or geometrically insufficiently informative frames.

To quantify the camera parameters, optimization was performed on a set of correspondences between the three-dimensional points on the calibration board and their observed locations in the two-dimensional image. During the optimization process, the stop criterion $\varepsilon = 10^{-9}$ was used, with the maximum number of iterations being 100. For each accepted image, a set of corresponding pairs

$$(\mathbf{X}_{ij}, \mathbf{x}_{ij})\tag{3.9}$$

was obtained, where $\mathbf{X}_{ij}$ denotes the known coordinate of the j -th ChArUco point in the board coordinate system, and $\mathbf{x}_{ij}$ denotes its measured position in the i -th image. The calibration problem was then reduced to minimizing the total reprojection error[74]:

$$\min_{\mathbf{K}, \mathbf{d}, \mathbf{R}_i, \mathbf{t}_i} \sum_{i=1}^N \sum_{j=1}^{M_i} \|\mathbf{x}_{ij} - \pi(\mathbf{K}, \mathbf{d}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_{ij})\|^2 \quad (3.10)$$

where $\mathbf{d}$ is the vector of distortion coefficients, and $\pi(\cdot)$ denotes the distortion-aware projection function.

The calibration quality was assessed by the root mean square (RMS) reprojection error. For N images and the total number of observations Q , it can be expressed as

$$e_{\text{RMS}} = \sqrt{\frac{1}{Q} \sum_{i=1}^N \sum_{j=1}^{M_i} \|\mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij}\|^2} \quad (3.11)$$

where $\hat{\mathbf{x}}_{ij}$ are the coordinates of the point obtained by back-projection after optimizing the camera parameters. In addition, an individual reprojection error was calculated for each frame, which made it possible to analyze the average, median, minimum, and maximum error values. After obtaining the individual reprojection error, frames were filtered using a threshold of no more than 0.19 px. As a result of calibration, the intrinsic camera matrix given in formula (3.2) and the vector of distortion coefficients $\mathbf{d} = [k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6]^\top$ were determined, which were used for image correction and metrically consistent analysis of the working scene.

3.3.2 Workspace definition using four ArUco markers

In order to obtain the boundaries of the workspace, four ArUco[33] markers were used. The image from the camera is shown in Fig. 3-3. When an input frame was received, the image was preprocessed, after which the markers were detected. Preprocessing included obtaining a real-time video stream from the camera with a resolution of 1280×720 at 15 FPS. Each frame was converted from BGR to a grayscale image. After that, contrast-limited adaptive histogram equalization (CLAHE)[75] was ap-

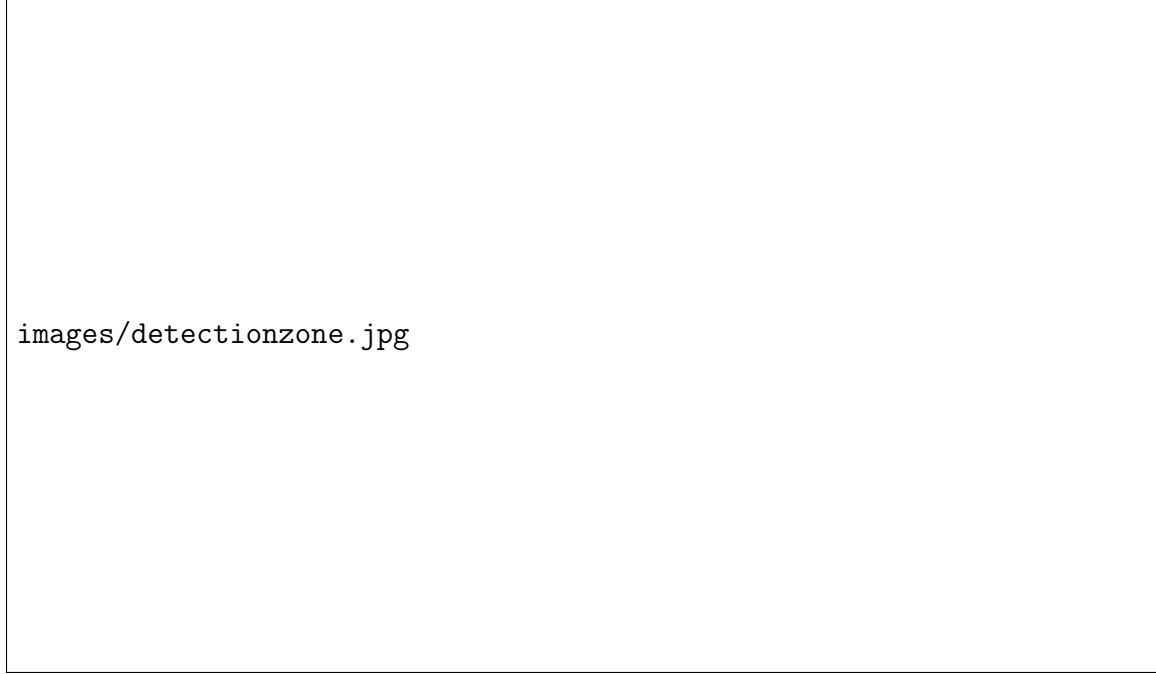


Figure 3-3: Workspace definition using four ArUco markers.

plied. This method enhances the visibility of borders and markers even under uneven lighting conditions. The workspace was considered ready for use when four markers were detected in the frame. This restriction allows us to form the boundaries of the workspace only from a complete and geometrically consistent set of reference points.

Let the detected markers be indexed as $k = 1, 2, 3, 4$, and let each marker be defined by four corner points in the image:

$$\mathbf{m}_{k,j} = (x_{k,j}, y_{k,j}), \quad j = 1, 2, 3, 4 \quad (3.12)$$

where $\mathbf{m}_{k,j}$ are the coordinates of the j -th corner of the k -th marker in the image plane. The system combines all the detected corner points and creates an outer quadrilateral based on them, which limits the workspace. This approach allows us to define the workspace more precisely and easily change its dimensions and shape, inside which the desired object is segmented for cutting.

The set of all corner points can be written in the form

$$\mathcal{P} = \{\mathbf{m}_{k,j} \mid k = 1, \dots, 4, j = 1, \dots, 4\} \quad (3.13)$$

In order to obtain the correct order of the markers, extreme values of the sums and differences of coordinates are used. The vertices of the workspace are defined as follows:

$$\mathbf{z}_1 = \arg \min_{\mathbf{p}=(x,y) \in \mathcal{P}} (x + y) \quad (3.14)$$

$$\mathbf{z}_2 = \arg \min_{\mathbf{p}=(x,y) \in \mathcal{P}} (y - x) \quad (3.15)$$

$$\mathbf{z}_3 = \arg \max_{\mathbf{p}=(x,y) \in \mathcal{P}} (x + y) \quad (3.16)$$

$$\mathbf{z}_4 = \arg \max_{\mathbf{p}=(x,y) \in \mathcal{P}} (y - x) \quad (3.17)$$

where $\mathbf{z}_1$ is the upper-left, $\mathbf{z}_2$ is the upper-right, $\mathbf{z}_3$ is the lower-right, and $\mathbf{z}_4$ is the lower-left corner. The order of the vertices is important to eliminate ambiguity in the sequence and to correctly construct the geometric transformation of each frame.

The average side length of a marker in pixels is calculated as follows:

$$\bar{\ell}_{\text{px}} = \frac{1}{16} \sum_{k=1}^4 \sum_{j=1}^4 \|\mathbf{m}_{k,j^+} - \mathbf{m}_{k,j}\| \quad (3.18)$$

where $j^+ = j + 1$ for $j = 1, 2, 3$, and $j^+ = 1$ is assumed for $j = 4$. Then the pixel-to-scale ratio is defined as:

$$s_{\text{mm/px}} = \frac{L_{\text{ArUco}}}{\bar{\ell}_{\text{px}}} \quad (3.19)$$

where L_{ArUco} is the known length of each side of the marker. Thus, the coefficient $s_{\text{mm/px}}$ defines the metric scale of the rectified workspace and makes it possible to convert all subsequent geometric measurements from pixels to millimeters. After obtaining this coefficient, the length and width of the quadrilateral in pixels are calculated:

$$W_{\text{px}} = \|\mathbf{z}_2 - \mathbf{z}_1\| \quad (3.20)$$

$$H_{\text{px}} = \|\mathbf{z}_4 - \mathbf{z}_1\| \quad (3.21)$$

In metric dimensions, the workspace is defined as follows:

$$W_{\text{mm}} = s_{\text{mm/px}} \cdot W_{\text{px}} \quad (3.22)$$

$$H_{\text{mm}} = s_{\text{mm/px}} \cdot H_{\text{px}} \quad (3.23)$$

After all operations, the workspace is transformed into a canonical area using a perspective transformation (homography)[38]. This helps to rectify the workspace so that perspective distortions are reduced. Let $\mathbf{H}$ be the homography matrix that maps the vertices of the workspace as follows:

$$\mathbf{H} : \{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4\} \rightarrow \{(0, 0), (W_{\text{px}} - 1, 0), (W_{\text{px}} - 1, H_{\text{px}} - 1), (0, H_{\text{px}} - 1)\} \quad (3.24)$$

Then the rectified image of the workspace can be represented as:

$$I_{\text{rect}}(u, v) = I \left(\mathbf{H}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \right) \quad (3.25)$$

This approach to detecting the workspace is very convenient for small and medium-sized workshops, where the configuration of the workspace is constantly changing, or where the dimensions of the sheet material are constantly changing.

3.4 Segmentation of sheet material

Segmentation is a very important part of the proposed system. The purpose of segmentation is to detect and highlight the boundaries of the sheet material in the workspace. The proposed system used a hybrid approach consisting of (i) selection of the boundaries of an object, (ii) background subtraction relative to the reference image (a backup method), (iii) morphological postprocessing of a binary mask, and (iv) filtering connected components by area. However, before the segmentation stage, preprocessing of the frame was performed.

In order to get rid of noise and small textural distortions, each frame was first converted to grayscale, and then it was subjected to soft Gaussian smoothing [20]. This transformation can be written as:

$$\mathbf{I}_t^{\text{gray}} = \mathcal{G}(\mathbf{I}_t^{\text{rect}}), \quad \mathbf{I}_t^{\text{blur}} = \mathcal{H}_\sigma * \mathbf{I}_t^{\text{gray}}, \quad (3.26)$$

where $\mathbf{I}_t^{\text{rect}}$ is the rectified frame of the workspace at time t , $\mathcal{G}(\cdot)$ is the grayscale conversion operator, and $\mathcal{H}_\sigma$ is the Gaussian smoothing filter. In the implemented system, a 5×5 kernel was employed, which effectively suppressed high-frequency noise without introducing noticeable blurring of the material boundaries.



Figure 3-4: Example of object segmentation.

After preprocessing the frame, the first stage of segmentation was performed: selection of the boundaries of the object. This method works by detecting sudden changes in brightness between the workspace and the object. The Canny operator

was applied to the smoothed frame, which formed the contours of the objects[20]:

$$\mathbf{E}_t(x, y) = \text{Canny}(\mathbf{I}_t^{\text{blur}}, T_{\text{low}}, T_{\text{high}}), \quad (3.27)$$

where T_{low} and T_{high} are the lower and upper thresholds of the edge detector. After applying the operator, morphological dilation of the detected edges was performed using a small elliptical structuring element[64]; this step is necessary in order to turn thin contour lines into more stable connected objects. The resulting edge-based mask is denoted by $\mathbf{M}_t^{\text{edge}}$.

A fallback segmentation method is background subtraction relative to the reference image. This approach compares the reference frame with the received frame[54]. The reference frame was created when the segmented objects were not present in the workspace, and it could also be updated at the user's request. This approach can be described as follows:

$$\mathbf{B} = \mathcal{H}_\sigma * \mathcal{G}(\mathbf{I}_{t_0}^{\text{rect}}), \quad (3.28)$$

where $\mathbf{B}$ is the reference image, and t_0 is the moment of its registration. Based on this, the difference between the input frame and the reference frame was calculated:

$$\mathbf{D}_t(x, y) = |\mathbf{I}_t^{\text{blur}}(x, y) - \mathbf{B}(x, y)|, \quad (3.29)$$

after that, a binary foreground mask was formed:

$$\mathbf{M}_t^{\text{bg}}(x, y) = \begin{cases} 1, & \mathbf{D}_t(x, y) \geq T_{\text{bg}}, \\ 0, & \text{otherwise.} \end{cases} \quad (3.30)$$

where T_{bg} is the binarization threshold. This approach allows us to highlight segmented objects even in cases where their visual difference from the workspace surface is not sufficiently large[54].

However, the main method for obtaining a binary mask is selection of the object boundaries, but the user can change the segmentation method manually. With this

in mind, the binary segmentation mask is set as follows:

$$\mathbf{M}_t^{(0)} = \begin{cases} \mathbf{M}_t^{\text{edge}}, & \text{if the edge channel is active,} \\ \mathbf{M}_t^{\text{bg}}, & \text{otherwise.} \end{cases} \quad (3.31)$$

After that, morphological postprocessing was performed. Median smoothing was used to suppress single-pixel noise, followed by morphological opening using a small elliptical structuring element[64]:

$$\mathbf{M}_t^{(1)} = \gamma\left(\text{Med}\left(\mathbf{M}_t^{(0)}\right)\right). \quad (3.32)$$

If it was necessary to combine closely located objects, the system could additionally perform morphological closing[64]. However, morphological closing was not used in the proposed system, since when working with objects of complex shape, a slight coalescence of neighboring areas can lead to segmentation errors.

The final stage is binary-mask filtering using connected component analysis[59]. For each segmented object, the area in pixels was calculated, and only those objects that exceeded the minimum area threshold were included in the final mask:

$$\mathbf{M}_t^{\text{seg}}(x, y) = \begin{cases} 1, & (x, y) \in \mathbf{C}_k \text{ and } A(\mathbf{C}_k) > A_{\min}, \\ 0, & \text{otherwise,} \end{cases} \quad (3.33)$$

where $\mathbf{C}_k$ is the k -th connected component, $A(\mathbf{C}_k)$ is the area of the object, and $A_{\min}$ is the minimum allowable area of the object. Filtering helps to remove false detections, glare, or changes in lighting. An example of the segmentation result is shown in Fig. 3-4.

Segmentation in the proposed system is not just the detection of an object, but the detection of its real geometric shape. Therefore, the result of segmentation is an exact binary mask of the segmented object, which is then used to obtain its geometric parameters.

3.5 Geometric description of segmented material

After obtaining the binary mask of the segmented object, the next step is to obtain its geometric parameters. The outer contour of the material is extracted from the binary mask[67], then approximated by a compact polygonal model[26]. A minimum-area bounding rectangle is calculated for each segmented object[32]. It is used for: (i) obtaining a stable local coordinate system, (ii) estimating dimensions, and (iii) obtaining a stable orientation of the object, but it does not replace the real contour. Unlike a conventional axis-aligned bounding rectangle, such a container takes into account the actual orientation of the object in the plane and therefore better corresponds to the geometry of the real material[32]. If we denote this rectangle by $\mathcal{R}^*$, then it is defined as:

$$\mathcal{R}^* = \arg \min_{\mathcal{R} \supset \mathcal{C}} S(\mathcal{R}), \quad (3.34)$$

where $S(\mathcal{R})$ is the area of the rectangle covering the contour $\mathcal{C}$. Let the vertices of the found rectangle be ordered as:

$$\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4. \quad (3.35)$$

Then its main dimensions in millimeters are determined by the lengths of the corresponding sides:

$$W = \|\mathbf{p}_2 - \mathbf{p}_1\| s, \quad H = \|\mathbf{p}_4 - \mathbf{p}_1\| s, \quad (3.36)$$

where s is the scale factor in millimeters per pixel obtained during workspace identification.

After constructing the oriented container, the contour is translated into a local coordinate system associated with the segmented object itself. The vertex $\mathbf{p}_1$ is taken as the origin, and the axes of the local system are directed along the two main sides of the rectangle. Then the unit basis vectors of the local system can be set as follows:

$$\mathbf{e}_u = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|}, \quad \mathbf{e}_v = \frac{\mathbf{p}_4 - \mathbf{p}_1}{\|\mathbf{p}_4 - \mathbf{p}_1\|}. \quad (3.37)$$

For each point of the original contour $\mathbf{q}_i$, the local coordinates (u_i, v_i) are obtained by projection onto these directions:

$$u_i = ((\mathbf{q}_i - \mathbf{p}_1) \cdot \mathbf{e}_u) s, \quad v_i = ((\mathbf{q}_i - \mathbf{p}_1) \cdot \mathbf{e}_v) s. \quad (3.38)$$

As a result, a local polygon of the object is formed:

$$\mathcal{P}_{\text{loc}} = \{(u_i, v_i)\}_{i=1}^N. \quad (3.39)$$

The coordinates of this polygon are already expressed in millimeters. This method is convenient because it directly relates the shape of an object to its physical dimensions.

The area of a segmented object is calculated not from the area of the bounding rectangle, but from the local polygon. The Gauss area formula is used for this[15]:

$$S_{\text{piece}} = \frac{1}{2} \left| \sum_{i=1}^N (u_i v_{i+1} - v_i u_{i+1}) \right|, \quad (u_{N+1}, v_{N+1}) = (u_1, v_1). \quad (3.40)$$

The value S_{piece} reflects the true area of the object, taking into account its actual shape. Also, to describe the geometric contour, the lengths of the sides of the polygonal approximation are calculated. If the neighboring vertices of the local polygon are defined by the points (u_i, v_i) and (u_{i+1}, v_{i+1}) , then the length of the corresponding edge is defined as:

$$l_i = \sqrt{(u_{i+1} - u_i)^2 + (v_{i+1} - v_i)^2}, \quad i = 1, 2, \dots, N, \quad (3.41)$$

with cyclic closure of the last edge to the first vertex. The set of values l_i is one of the simple descriptors of the object shape.

As a result of this stage, (i) an external contour is formed for each object, (ii) a simplified polygonal representation is obtained, (iii) a minimum-area bounding rectangle is determined, (iv) a local polygon in millimeters is constructed, and (v) the geometric parameters of the object are calculated. The results obtained form the basis for the segmented-object placement algorithm.

3.6 Geometric representation of input parts

In the proposed system, the input parts are represented by a unified geometric model. The system allows us to create a layout scheme for objects of simple and arbitrary shape. In general, the input object can be written as:

$$\mathbf{P}_i = (\tau_i, \mathcal{G}_i, \mathcal{M}_i, \mathcal{A}_i), \quad (3.42)$$

where τ_i denotes the object type, $\mathcal{G}_i$ is a geometric description, $\mathcal{M}_i$ is a set of metric parameters, and $\mathcal{A}_i$ is a set of additional attributes necessary for storage and subsequent placement.

The simplest way to specify parts is for the user to select the object type: (i) rectangle or (ii) circle, as well as enter its geometric parameters. For a rectangular object type, the geometric parameters are set by the width $\mathbf{w}_i$ and height $\mathbf{h}_i$:

$$\mathcal{G}_i^{\text{rect}} = \{\mathbf{w}_i, \mathbf{h}_i\}, \quad (3.43)$$

For a circular part, the geometric parameters are set by the diameter $\mathbf{d}_i$:

$$\mathcal{G}_i^{\text{circ}} = \{\mathbf{d}_i\}. \quad (3.44)$$

This method is very convenient when working with simple workpieces and requires only a minimal set of parameters.

In order to place objects of arbitrary shape, a method is proposed in which the user independently draws the objects required for cutting on the segmented material, focusing on the shape of the material. An example of this process is shown in Fig. 3-5. In this case, the parts are represented as a closed polygon, which is defined by an ordered set of vertices in the metric coordinate system:

$$\mathcal{G}_i^{\text{poly}} = \{\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \dots, \mathbf{v}_{i,n_i}\}, \quad \mathbf{v}_{i,k} = (x_{i,k}, y_{i,k}). \quad (3.45)$$

It is important to note that the proposed system has a function for storing user-



Figure 3-5: Example of manual creation of an irregularly shaped part on the segmented material.

defined shapes, which allows us to reuse user-created objects in the future. The shape repository contains geometric descriptions of contours in millimeters, including polygon vertices, the closure property, border segments, and the dimensions of the bounding container.

Additionally, the proposed system implements a batch upload mode for parts in .csv format. This mode is needed in order to place a pre-formed list of objects, which is automatically sent to the placement algorithm. Each entry contains information about the type of part, its geometric parameters, additional attributes, and placement status.

Thus, the use of a hybrid approach in the representation of input data makes it suitable for standard production tasks, where both a typical set of objects and objects of arbitrary shape are used.

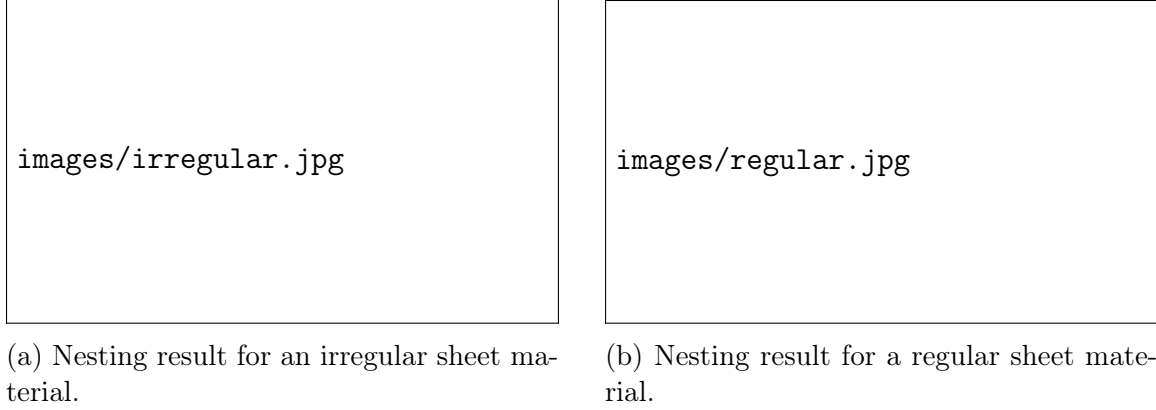


Figure 3-6: Examples of layouts produced by the proposed automatic 2D nesting algorithm for different sheet material shapes.

3.7 Nesting algorithm

The nesting problem is formulated in the standard form as the maximization of material utilization over an admissible placement region Ω :

$$\max_{\Omega} \eta = \frac{\sum_{i=1}^N z_i S_i}{S_{\Omega}}, \quad (3.46)$$

where $z_i \in \{0, 1\}$ is the placement indicator of part i , S_i is the area of the i -th part, and S_{Ω} is the area of the admissible material region. The placement must satisfy the feasibility constraints

$$F_i(\mathbf{r}_i, \theta_i) \subseteq \Omega, \quad F_i(\mathbf{r}_i, \theta_i) \cap F_j(\mathbf{r}_j, \theta_j) = \emptyset, \quad i \neq j, \quad (3.47)$$

where F_i is the geometry of part i , $\mathbf{r}_i$ is its position, and θ_i is its rotation angle.

In this work, however, Ω is not a predefined rectangular container. Instead, it is obtained directly from the segmented contour of the real sheet material and represented in local metric coordinates. The implemented algorithm, denoted *HybridNest*, is therefore a practical raster-based multi-start heuristic designed for segmented material regions of arbitrary shape. The general problem formulation and the ideas of bottom-left placement, greedy ordering, and local compaction follow the nesting literature[8, 9, 27, 17, 36]. The proposed contribution of this work is their integra-

tion with: (i) a segmented irregular workspace, (ii) adaptive raster resolution, (iii) a multi-start schedule of ordering strategies, and (iv) an interactive operating mode.

Before layout generation, all input parts are filtered by geometric feasibility. Circular parts are accepted only if their diameter fits into the admissible region dimensions. Rectangular and polygonal parts are checked in the default and rotated orientations whenever such rotation is allowed. This stage removes obviously infeasible parts and reduces the size of the search space before the actual nesting stage.

After filtering, the admissible region Ω is discretized into a binary raster mask A^Δ , where Δ is the grid cell size in millimetres. Intuitively, Δ controls the balance between geometric precision and computational cost: smaller values preserve narrow concavities and small gaps more accurately, whereas larger values accelerate the search but coarsen the geometry. In the implementation, Δ is adapted to the current instance using the part dimensions and the expected packing density. Let $d_{\min}$, d_{med} , and $d_{\max}$ denote the minimum, median, and maximum characteristic part sizes, and let

$$\rho = \frac{\sum_{i=1}^N S_i}{S_\Omega} \quad (3.48)$$

be the expected density of the instance. First, a precision-oriented base step is defined as

$$\Delta_p = \max \left(\Delta_{\min}, \frac{d_{\min}}{30} \right), \quad (3.49)$$

where $\Delta_{\min} = 0.15$ mm. Next, the size-variation coefficient

$$\nu = \frac{d_{\max}}{d_{\min}} \quad (3.50)$$

is used to compute an intermediate step

$$\Delta_0 = \begin{cases} 1.2 \Delta_p, & \nu < 2, \\ 0.75 \max \left(\Delta_p, \frac{d_{\text{med}}}{25} \right), & \nu > 5, \\ \Delta_p, & \text{otherwise.} \end{cases} \quad (3.51)$$

A density correction factor is then applied:

$$\alpha(\rho) = \begin{cases} 0.85, & \rho > 0.7, \\ 1.15, & \rho < 0.3, \\ 1.0, & \text{otherwise,} \end{cases} \quad (3.52)$$

and the final raster step is obtained as

$$\Delta = \min \{ \Delta_{\max}, \max(\Delta_{\min}, \alpha(\rho)\Delta_0) \}, \quad (3.53)$$

where $\Delta_{\max} = 1.0$ mm. For irregular segmented material regions, the implementation additionally restricts the raster step to at most 0.35 mm in the quality-oriented mode in order to preserve narrow geometric details. It should be noted that all numerical constants present in the algorithm — including resolution bounds, scaling coefficients, and density thresholds — are empirically determined heuristic parameters, chosen to yield satisfactory practical performance rather than obtained through formal analytical optimization. These constants are presented as configurable parameters, which allows the user to configure them independently.

The search itself is organized as a multi-start procedure. Instead of relying on a single fixed part order, HybridNest evaluates several alternative orderings of the same input set. In the full configuration, the implementation defines up to 15 ordering strategies, including best-fit decreasing, area-to-perimeter ordering, difficulty-first ordering, aspect-ratio-biased ordering, width-oriented ordering, first-fit decreasing height, longest-side-first ordering, and mixed large-small schedules. For one layout computation, the quality-oriented configuration evaluates at most 16 attempts. When multiprocessing is enabled, these attempts are distributed across up to 14 worker processes. In the interactive fast mode, the schedule is reduced to four fast orderings and the attempt budget is reduced to approximately 2–8 attempts depending on the instance size; for irregular segmented regions in live operation, the attempt budget is capped at 2 in order to preserve responsiveness.

Within each attempt, the parts are placed sequentially according to the selected ordering. For each part, a set of orientation candidates is generated. Circles have a single orientation candidate, rectangles use 0° and 90° , and polygonal library parts use the allowed angles stored in the corresponding shape description, with $\{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$ used as the default set. Each orientation is converted into a binary footprint on the raster grid.

Candidate positions are then generated using a hierarchy of placement rules. First, a true bottom-left scan is performed to identify the lowest feasible row and the left-most feasible position on that row. Second, stable corner positions are checked. Third, positions adjacent to already placed parts are examined in order to obtain tighter local packing. Fourth, skyline-type positions are tested in the quality-oriented mode. Finally, a coarse residual scan is performed over the remaining admissible raster cells. Every candidate that satisfies both the occupancy mask and the admissible-region mask is greedily shifted towards the origin while feasibility is preserved. This local sliding step reduces free gaps and acts as a lightweight compaction operator directly during placement. Among all feasible candidates, the implementation selects the one minimizing the bottom-left priority key

$$k(x, y) = y + 0.35x, \tag{3.54}$$

while the heuristic candidate score is used only as a tie-breaker.

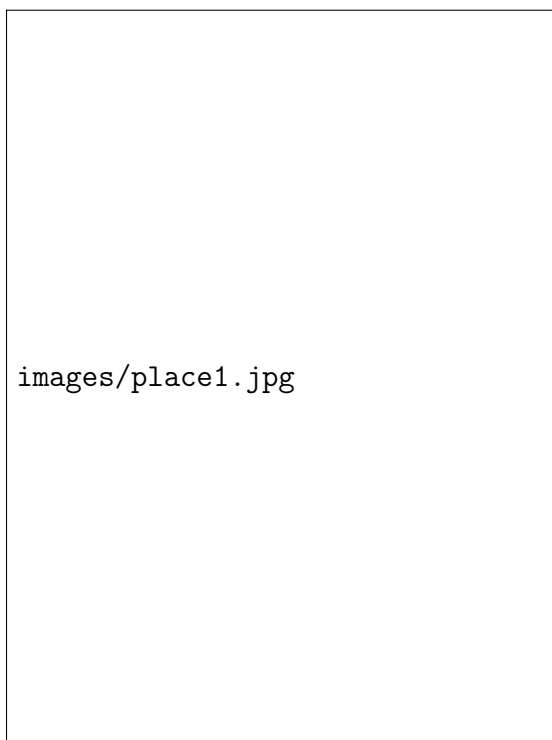
If none of the preferred candidates is feasible, a fallback raster scan is executed. After the first full pass through the ordered parts, the algorithm performs up to two additional gap-filling passes on the remaining unplaced parts. In these passes, the residual set is reordered to favor smaller pieces and circles that can occupy local cavities. After that, a post-processing stage shifts already placed parts again towards the origin in order to shrink the occupied envelope and improve compactness.

Each attempt is evaluated using the total placed area, the number of placed parts, the used envelope length, and the compactness of the occupied region. For fixed material regions, the best solution is selected lexicographically by maximizing placed

area first, then maximizing the number of placed parts, and finally minimizing the occupied envelope. Thus, HybridNest does not search the continuous placement space directly. Instead, it approximates the feasible domain on an adaptive raster and compensates for the resulting discretization by combining multiple orderings, local sliding, gap filling, and post-compaction. This makes the method suitable for segmented real material regions, where geometric irregularity and response time are more important than exact global optimality. The complete packing procedure is summarized in Algorithm 1, and the single-piece insertion routine is detailed in Algorithm 2 (see Appendix A.1).

3.8 Projection-based visual feedback

The main difference from other studies is that the proposed system provides feedback in the form of a projection directly onto the workspace[56, 51, 13]. After obtaining an effective layout, the user sees on the computer monitor a visual display of all the placed parts on the selected segmented material. The user then captures a frame containing only the workspace, and the result is projected directly onto the work surface through a projector with a resolution of 1280×720 px. In this case, the selected segmented object is displayed in green, and the unselected ones are outlined in yellow. The placed parts on the segmented object are highlighted in red. An example of such projection support is shown in Fig. 3-7. To analyze the placement algorithm, the frame is accompanied by visual information about the material utilization rate and the number of placed parts relative to their total number. This form of feedback helps the operator to immediately assess the quality of the placement, reduce the operator’s cognitive load during material cutting, and make the cutting process more controlled[51, 13, 69].



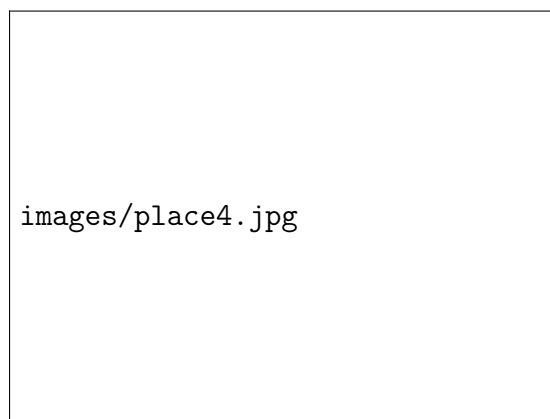
(a) Projection of the nesting result onto the selected irregular material in the physical workspace.



(b) Projection of the nesting result onto the selected regular material in the physical workspace.



(c) Projection of the nesting result onto the selected regular material in the physical workspace under low-light conditions for better visibility.



(d) Projection of the nesting result onto the selected irregular material in the physical workspace under low-light conditions for better visibility.

Figure 3-7: Examples of projection-based visual guidance in the proposed automatic 2D nesting system.

Chapter 4

Results

This section evaluates the results of the proposed system according to the key stages of its operation. The analysis was performed sequentially in four areas: (i) camera calibration quality, (ii) segmentation quality, (iii) efficiency of the layout algorithm, and (iv) efficiency of the entire system. This assessment method allows us to evaluate the accuracy of each stage and the entire system as a whole.

4.1 Camera calibration quality

Obtaining geometric parameters of segmented materials depends directly on the calibration quality. The calibration quality was estimated by the RMS reprojection error according to formula (3.11). This value characterizes the discrepancy between the actually detected control points and the points projected onto the image. The smaller the RMS error, the higher the geometric consistency[74].

As a result of calibration, the value of e_{RMS} was 0.1501 px. Additionally, the average error was 0.149 px, the median was 0.153 px, the standard deviation was 0.016 px, the minimum value was 0.128 px, and the maximum was 0.183 px. The low RMS error, combined with the small inter-frame variation, shows that the estimated intrinsic camera parameters and the distortion model consistently describe the real geometry. Thus, the calibration can be considered sufficiently accurate for subsequent steps. Comparing this result with the data from other studies presented in Table 4.1,

Table 4.1: Comparison of reprojection error with published camera calibration results.

Work	Calibration setup	Error, px	Comment
This work	Monocular camera, 1280×720 , 29 accepted frames, ChArUco board	0.1501	e_{RMS} , subpixel reprojection error
Geiger et al. [34]	Camera-to-camera calibration, checkerboard-based setup	0.18	Corner reprojection error
Sels et al. [63]	Standard checkerboard calibration	0.25	Average reprojection error
An et al. [3]	Omnidirectional camera calibration using ChArUco board	0.37	Average reprojection error
Roshan et al. [60]	Thermal camera calibration using ChArUco board	< 0.4	Overall mean reprojection error

we can conclude that the achieved accuracy is in the high-precision subpixel range[34, 3, 60].

It should also be noted that a direct quantitative comparison of reprojection errors between studies is only indicative, because the final result is influenced by the type of camera, resolution, distortion model, number of frames, and shooting conditions.

4.2 Geometric evaluation of segmentation results

To evaluate the quality of segmentation, the obtained geometric parameters were compared with ground-truth measurements. We used 10 objects of different shapes and sizes, and the deviation from ground truth was approximately 0.4–0.6 cm. The results show that the proposed approach provides fairly accurate geometric descriptions of segmented objects; the measurements for each object are summarized in Table 4.2.

4.3 Nesting algorithm efficiency

In order to evaluate the placement algorithm, open benchmarks were used. The quality of the algorithm was evaluated in two directions: (i) the layout of regular-

Table 4.2: Dimensional comparison of segmented objects with ground truth.

Object	W^{gt} , cm	H^{gt} , cm	W^{seg} , cm	H^{seg} , cm	e_i , cm
1	18.0	12.0	18.3	11.5	0.40
2	20.5	14.0	20.1	14.5	0.45
3	16.0	10.5	15.5	10.0	0.50
4	22.0	15.5	22.6	15.0	0.55
5	19.0	13.0	18.3	13.5	0.60
6	24.5	16.0	24.1	15.6	0.40
7	14.0	9.5	14.4	9.0	0.45
8	21.0	11.5	20.5	10.9	0.55
9	17.5	12.5	17.0	12.0	0.50
10	23.0	14.5	22.4	13.9	0.60
Mean	19.55	12.90	19.32	12.59	0.50

shaped parts and (ii) the layout of irregular-shaped parts.

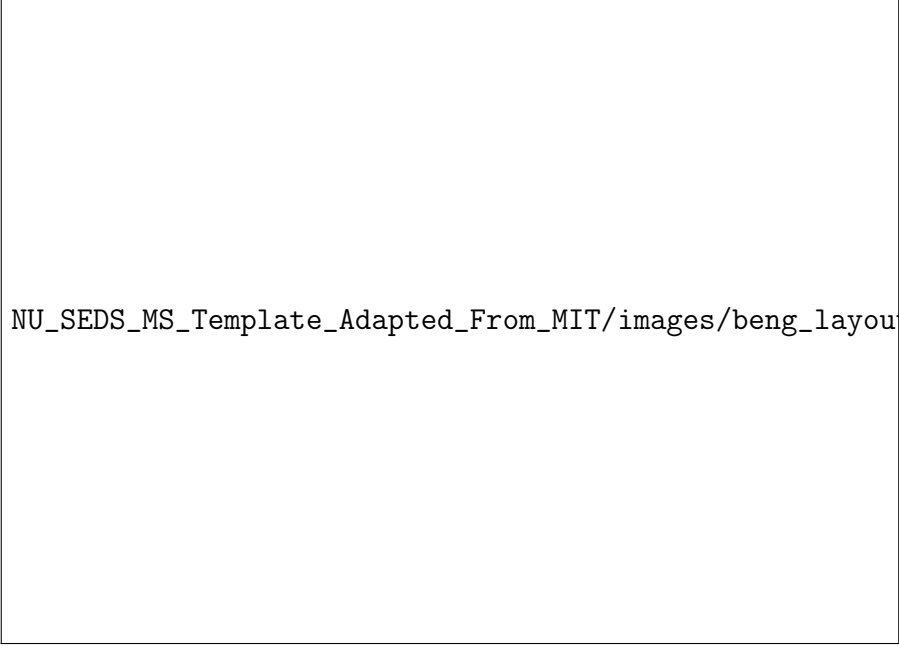
4.3.1 Evaluation of the layout of regular-shaped parts

Table 4.3: Comparison of the proposed method with representative results from the literature on the BENG benchmark family.

Method	Reference	Avg. filling rate (%)	Avg. time (s)
LWF	[71]	99.3778	6.07
HSA	[46]	99.3312	6.05
IBHP	[65]	99.4326	–
HAGA	[48]	99.3429	228.12
Proposed HybridNest	This work	99.7200	18.28

Table 4.4: Results of the proposed method on the BENG benchmark family.

Instance	Container, mm	Available items	Placed items	Utilization, %	Trim loss, %	Time, s
beng_01	25×10	20	9	98.80	1.20	1.50
beng_02	25×10	40	8	99.60	0.40	2.42
beng_03	25×10	60	9	100.00	0.00	3.37
beng_04	25×10	80	10	100.00	0.00	4.49
beng_05	25×10	100	13	100.00	0.00	5.43
beng_06	40×25	40	22	98.80	1.20	7.59
beng_07	40×25	80	28	100.00	0.00	20.95
beng_08	40×25	120	28	100.00	0.00	33.18
beng_09	40×25	160	35	100.00	0.00	46.08
beng_10	40×25	200	29	100.00	0.00	57.76
Average	–	–	19.1	99.72	0.28	18.28



NU_SEDS_MS_Template_Adapted_From_MIT/images/beng_layout.png

Figure 4-1: Example of part placement produced by the proposed HybridNest algorithm for a regular-shaped benchmark instance.

The four benchmarks used to evaluate the layout of regular-shaped parts were[42]: (i) BENG[7], (ii) CGCUT[22], (iii) NGCUT[6], and (iv) WANG[12]. In these benchmark families, the number of available items may exceed the number of items actually placed, since the objective is to maximize the utilization of a fixed container rather than to place all input items. Therefore, the utilization values reported in this study refer to the fraction of the container area occupied by the placed parts, not to the fraction of items placed.

The BENG benchmark consists of 10 instances[7, 42]. All instances consist of rectangular objects that need to be placed in a known container. A 25×10 mm container was used for instances 1 to 5, and a 40×25 mm container was used for instances 6 to 10. The number of parts increases from 20 to 200, which makes this benchmark suitable for evaluating algorithmic scalability. The number of part types varies from 20 to 40. The BENG benchmark is used to evaluate area utilization as task size increases[42]. According to the results obtained in the proposed system, the algorithm showed high placement efficiency. The average area utilization rate was 99.7%. The average computation time was 18.28 seconds, and for the largest instance

it was 57.76 seconds. A comparative analysis with representative methods from the literature is presented in Table 4.3. The detailed quantitative results obtained for all BENG instances are summarized in Table 4.4. An example of the layout produced by the proposed algorithm is shown in Fig. 4-1.

Table 4.5: Results of the proposed method on the CGCUT benchmark family.

Instance	Container, mm	Available items	Placed items	Utilization, %	Trim loss, %	Time, s
cgcut_01	15×10	16	9	100.00	0.00	1.24
cgcut_02	40×70	23	12	95.00	5.00	8.56
cgcut_03	40×70	62	3	88.75	11.25	5.24
Average	–	–	8	94.58	5.42	5.02



Figure 4-2: Representative layout obtained by the proposed method for a CGCUT benchmark instance.

The CGCUT benchmark consists of 3 instances[22, 42]. The number of parts in the set varies from 16 to 62, and the number of different types varies from 7 to 19. The containers for the three instances have sizes of 15×10 , 40×70 , and 40×70 mm. This benchmark is the most demanding in terms of geometric density and combinatorial complexity. The utilization rates for the 1st, 2nd, and 3rd instances were 100%, 95%, and 88.7%, respectively; the average value was 94.5%. The average execution time of the algorithm was 5.02 seconds. The quantitative results of the proposed method on

all instances of the CGCUT family are given in Table 4.5. A visual example of the resulting layout scheme for one of the CGCUT family instances is shown in Fig. 4-2. A comparison of the proposed method with representative results from the literature on the CGCUT benchmark is presented in Table 4.6.

Table 4.6: Comparison of area utilization on CGCUT benchmark instances.

Method	Area Utilization, %			Avg., %
	cgcut_01	cgcut_02	cgcut_03	
Christofides & Whitlock [22]	96.20	89.30	81.40	88.97
Fekete & Schepers [30]	98.10	92.50	85.60	92.07
Clautiaux et al. [24]	99.40	93.80	86.90	93.37
Iori et al. [42]	99.70	94.10	87.50	93.77
This work	100.00	95.00	88.75	94.58

Table 4.7: Results of the proposed method on the NGCUT benchmark family.

Instance	Container, mm	Available items	Placed items	Utilization, %	Trim loss, %	Time, s
ngcut_01	10 × 10	10	5	95.00	5.00	0.86
ngcut_02	10 × 10	17	9	100.00	0.00	1.07
ngcut_03	10 × 10	21	9	100.00	0.00	1.20
ngcut_04	15 × 10	7	6	92.00	8.00	0.85
ngcut_05	15 × 10	14	5	99.33	0.67	0.94
ngcut_06	15 × 10	15	8	96.00	4.00	1.13
ngcut_07	20 × 20	8	8	43.75	56.25	0.85
ngcut_08	20 × 20	13	8	94.75	5.25	1.40
ngcut_09	20 × 20	18	8	100.00	0.00	1.66
ngcut_10	30 × 30	13	6	96.89	3.11	1.23
ngcut_11	30 × 30	15	9	97.00	3.00	2.10
ngcut_12	30 × 30	22	9	99.00	1.00	2.19
Average	–	–	7.5	92.81	7.19	1.29

The NGCUT benchmark consists of 12 instances[6, 42]. The containers have sizes of 10 × 10 mm, 15 × 10 mm, 20 × 20 mm, and 30 × 30 mm. The number of items varies from 7 to 22, with five to ten different types, and each instance has a profit weight. This benchmark evaluates the quality of the layout in cases where optimal solutions for complete coverage are geometrically impossible, which requires an effective strategy for selecting the placed parts. The results show that the average area utilization rate for all instances was 92.8%. The average running time of the algorithm was 1.29 seconds. The quantitative results of the proposed method for all instances of the NGCUT family are given in Table 4.7. A visual example of the

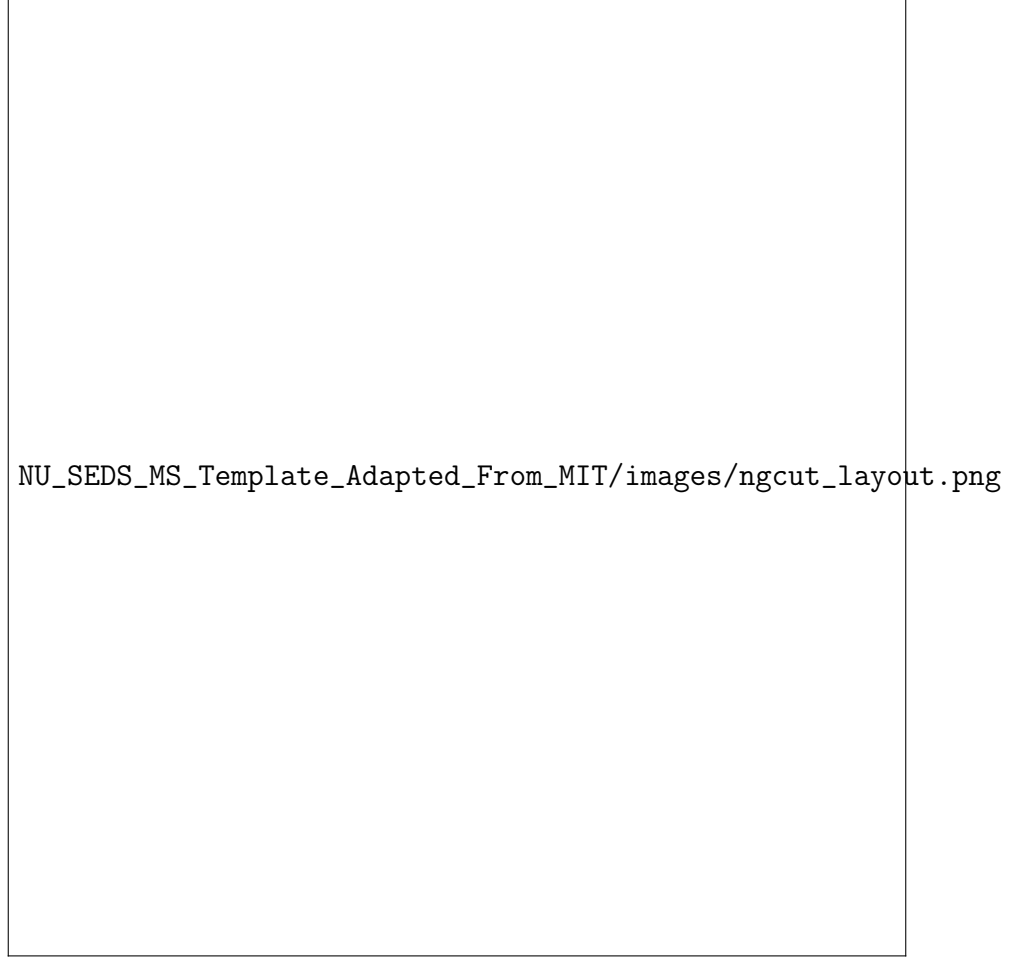


Figure 4-3: Example of the layout obtained by the proposed method for one of the NGCUT benchmark instances.

resulting layout scheme for one of the NGCUT instances is shown in Fig. 4-3.

Table 4.8: Results of the proposed method on the WANG benchmark family.

Instance	Container, mm	Available items	Placed items	Utilization, %	Trim loss, %	Time, s
wang_01	33×69	42	3	100.00	0.00	3.43
wang_02	39×70	42	7	86.45	13.55	4.55
wang_03	40×70	42	3	88.75	11.25	4.64
Average	—	—	4.3	91.73	8.27	4.21

The WANG benchmark includes 3 instances[12, 42]. The benchmark uses a set of 42 parts. The container sizes are 33×69 mm, 39×70 mm, and 40×70 mm. This benchmark is used to evaluate how the algorithm is affected by small changes in the size of the original container. According to the results, the proposed algorithm provided an area utilization rate of 100.0% for the 1st instance, 86.4% for the 2nd,



Figure 4-4: Example of the layout obtained by the proposed method for one of the WANG benchmark instances.

and 88.7% for the 3rd. The average area utilization rate was 91.7%. The quantitative results of the proposed method for all instances of the WANG family are given in Table 4.8. A visual example of the resulting layout scheme for one of the WANG family instances is shown in Fig. 4-4.

In general, the evaluation of all four benchmarks shows that the proposed algorithm provides competitive area utilization for the placement of rectangular parts. The average utilization was 99.7% for BENG, 94.5% for CGCUT, 92.8% for NGCUT, and 91.7% for WANG. The average running time across these four benchmark families was approximately 7.2 seconds. These results confirm that the placement algorithm maintains high spatial efficiency, although the exact value depends on the container size and the number of items.

4.3.2 Evaluation of the layout of irregular-shaped parts

Three classical benchmark sets were used to evaluate the layout of irregularly shaped parts [42]: (i) Albano [1], (ii) Jakobs1 [43], and (iii) Marques [50]. In all cases, the parts were placed in a strip-packing setting, where one strip dimension is fixed and the objective is to arrange the parts without overlap while minimizing the extent of the occupied strip. The comparative evaluation was based on the material utilization rate (%), defined as the ratio of the total area of the parts to the area of the occupied portion of the strip. This metric was chosen because it directly reflects packing density and provides a normalized basis for comparison across studies, whereas absolute strip length values may not be fully comparable due to differences in geometric scaling or unit conventions used in benchmark representations.

Table 4.9: Comparison with representative literature results on the Albano benchmark.

Method	Reported utilization, %	Reference
Gomes and Oliveira (SAHA)	87.43	[41]
Egeblad et al. (2DNest)	87.44	[41]
Imamichi et al. (ILSQN)	88.16	[41]
Domović et al.	85.17	[25]
Beam Search (ESICUP reference)	87.88	[11]
This work	63.00	–

The Albano benchmark contains 24 parts from 8 types of polygonal shapes, with an average of about 7.25 vertices per part, and acceptable orientations of 0° and 180° [1, 41, 8]. In the experiments, the proposed algorithm was able to place all 24 parts. The strip utilization rate was 63%, and the material loss was 37%. The average computation time for this instance was 97.3 seconds. Since the test set consists of repeated parts and a limited number of rotation angles, the benchmark allows us to evaluate how well the algorithm uses geometric contour compatibility. The results show that the proposed method is able to build a feasible layout for the complete set of parts; however, in terms of packing density, it is still inferior to specialized algorithms. A comparative analysis with the results of other studies is given in Table 4.9. A visual example of the resulting layout scheme for an Albano instance is shown in Fig. 4-5.

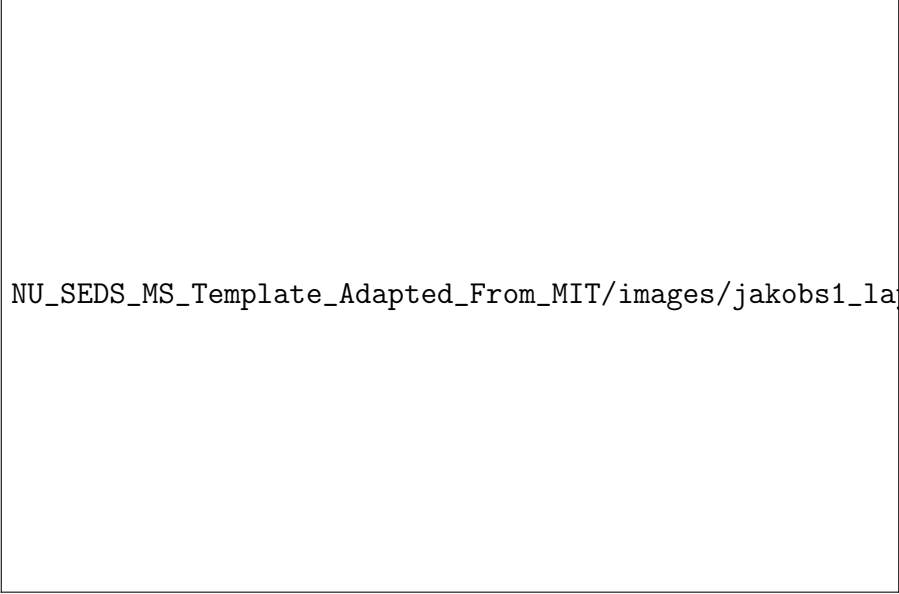


Figure 4-5: Layout obtained by the proposed method for the Albano benchmark instance.

Table 4.10: Comparison with representative literature results on the Jakobs1 benchmark.

Method	Utilization, %	Reference
Gomes and Oliveira (SAHA) [36]	78.89	[41]
Gomes and Oliveira (GLSHA) [36]	81.67	[41]
Burke et al. (BLF) [17]	82.60	[41]
Imamichi et al. (ILSQN) [41]	86.89	[41]
Egeblad et al. (2DNest) [29]	89.07	[41]
This work	43.91	—

The Jakobs1 benchmark contains 25 parts, with each part representing a different type[43, 42]. The average number of vertices per part is about 5.6. Rotations of 0° , 90° , 180° , and 270° [68] are allowed. Due to the absence of repeated objects and the presence of four acceptable orientations, this benchmark is a convenient test



NU_SEDS_MS_Template_Adapted_From_MIT/images/jakobs1_layout.png

Figure 4-6: Layout obtained by the proposed method for the Jakobs1 benchmark instance.

for assessing how effectively the algorithm handles both placement sequencing and orientation selection for irregular parts. The comparative results for this benchmark are presented in Table 4.10, and a representative layout is shown in Fig. 4-6.

The Marques benchmark contains 24 parts belonging to 8 different types[50, 42]. The average number of vertices in the polygonal description is about 7.37. Rotations of 0° , 90° , 180° , and 270° are allowed[41]. This benchmark is used to evaluate how efficiently the algorithm selects the placement order and part orientations and constructs a dense layout.

Table 4.11: Comparison with representative literature results on the Marques benchmark.

Method	Utilization, %	Reference
Gomes and Oliveira (SAHA) [36]	88.14	[41]
Burke et al. (BLF) [17]	86.50	[41]
Imamichi et al. (ILSQN) [41]	89.03	[41]
Egeblad et al. (2DNest) [29]	89.17	[41]
This work	56.94	—

In the experiments, the algorithm was able to place all 24 parts. The strip utilization rate was 56.94%, and the material loss was 43%. The average computation

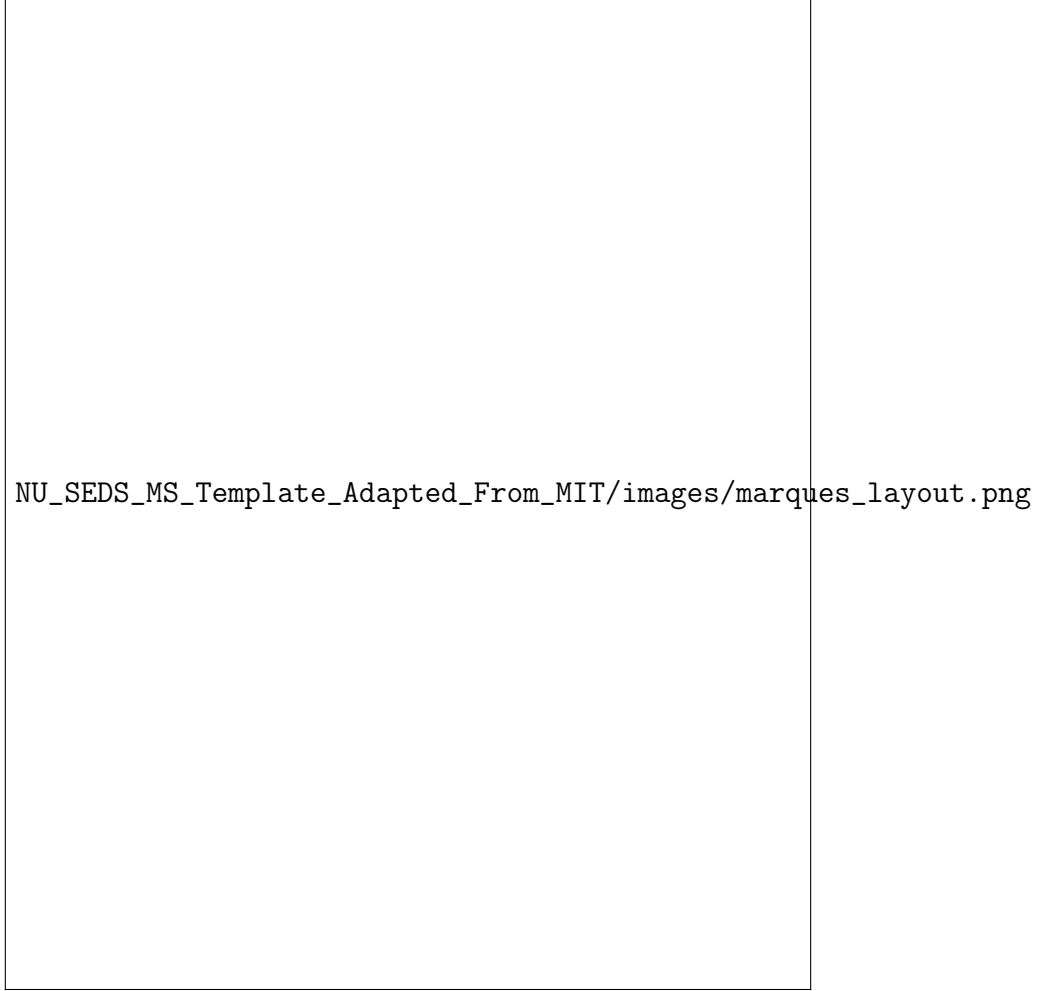


Figure 4-7: Layout obtained by the proposed method for the Marques benchmark instance.

time for this instance was 2.61 seconds. The results show that, as in the previous experiment on the Jakobs1 benchmark, the algorithm can arrange all the parts, but the packing density is inferior to that reported in other studies. The comparative quantitative results for Marques are given in Table 4.11. A visual example of the resulting layout scheme for the Marques instance is shown in Fig. 4-7.

According to the evaluation results on the three benchmarks, the algorithm lost from 37% to 56% of the material. The material utilization rate was 63% on the Albano[1] benchmark, while the best result reported in other studies was 88%; on the Jakobs1[43] benchmark, our result was 43.91% versus 89.07%; and on the Marques[50] benchmark, our result was 56.94% versus 89.17%. The results show that the algorithm

is able to arrange all the parts in an acceptable area without intersections, but the packing density requires improvement. The main limitation is the lack of iterative-improvement local search and contour-fitting heuristics that help other algorithms achieve high packing density.

4.4 Comprehensive assessment of the system

To evaluate the entire system, we created a custom test set consisting of 65 parts to be placed on the segmented material: 30 irregularly shaped parts and 35 regular-shaped parts. Because not all parts can fit onto a single segmented area simultaneously, the placement was carried out in four consecutive stages, with the remaining unplaced parts from each stage passed on to the next. Table 4.12 summarizes the results across these four stages. It is important to note that Instances 1–4 do not represent four independent experiments; rather, they represent sequential stages of a single placement process applied to the same original set of 65 parts. At each subsequent stage, only the parts that were not placed in the previous stage are considered, so the number of candidate parts decreases progressively. At Stage 1, the system attempted to place 65 parts and successfully placed 18 of them, corresponding to a placement rate of 27.69%. The candidate set sizes considered at Stages 2, 3, and 4 were 47, 36, and 21 parts, respectively, yielding placement rates of 23.40%, 41.67%, and 100.00%. Here, the placement rate is defined as the ratio of successfully placed parts to the total number of parts considered at a given stage, i.e., $n_{\text{placed}}/n_{\text{total}} \times 100$. These results indicate that the main limitation of the proposed system is the rapid expansion of the search space as the number of candidate parts increases.

Regarding material usage, the utilization metric must be interpreted independently from the placement rate. While the placement rate reflects how many of the candidate parts were successfully placed, utilization measures the proportion of the segmented material area occupied by the placed parts. Consequently, a high utilization does not imply a high placement rate, and vice versa. This distinction is clearly illustrated by comparing Stage 1 and Stage 4. At Stage 1, only 18 of the 65 parts

were placed, yielding a placement rate of 27.69% (computed as $18/65 \times 100$), yet these parts fully occupied the available material area, resulting in a utilization of 100.00% and zero trim loss. At Stage 4, all 21 remaining parts were successfully placed, corresponding to a placement rate of 100.00% (computed as $21/21 \times 100$) in just 3.37 seconds; however, since the total area of these 21 parts did not fully cover the available material, 21.51% of the material area remained unused, resulting in a trim loss of 21.51%. Overall, the evaluation demonstrates that the proposed system is capable of generating feasible layouts on segmented sheet material across all four stages, while its computational efficiency decreases with increasing numbers of candidate parts.

Table 4.12: End-to-end evaluation results for four consecutive placement stages on segmented material.

Stage	n_{total}	n_{placed}	Placement rate (%)	Utilization (%)	Trim loss (%)	Time (s)
1	65	18	27.69	100.00	0.00	389.82
2	47	11	23.40	98.83	1.17	78.98
3	36	15	41.67	86.60	13.40	53.63
4	21	21	100.00	78.49	21.51	3.37

Chapter 5

Conclusion

This thesis addressed the problem of automatic 2D nesting on physically captured sheet materials of irregular and unpredictable shape in small workshop and production settings. In contrast to conventional CAD/CAM-oriented workflows, where the geometry of the material is known in advance, the proposed approach operates directly on real materials acquired from the workspace through computer vision. To address this problem, an integrated end-to-end system was developed that combines camera-based workspace reconstruction, contour extraction and geometric modeling of segmented materials, heuristic nesting of regular and irregular parts, and projection-based visual guidance for the operator.

The obtained results demonstrate that the proposed system is technically feasible and suitable for practical deployment under controlled workshop conditions. The calibration stage achieved a reprojection error of 0.1501 pixels, providing a reliable metrical basis for subsequent reconstruction and measurement. The segmentation stage produced geometric descriptions with an average error of 0.5 cm relative to ground-truth measurements, indicating that the extracted contours are sufficiently accurate for downstream nesting. The proposed HybridNest algorithm was able to generate valid layouts for both regular and irregular material regions, while benchmark experiments confirmed strong performance for regular instances and practical applicability for segmented real-world objects. In addition, the projection module established a direct connection between the computed layout and the physical work

surface, thereby improving the usability of the system and reducing the operator’s need to mentally transfer digital plans to the cutting material.

From a research perspective, the main contribution of this work lies not only in the implementation of individual modules, but in their integration into a unified operational pipeline. The study demonstrates that automatic layout generation on real leftover materials can be achieved by coupling workspace reconstruction, segmentation, geometric representation, heuristic placement, and 1:1 projection-based feedback within a single system. This closes part of the gap identified in the literature between digital nesting methods and physically grounded, interactive cutting workflows.

At the same time, the study revealed several important limitations. First, the segmentation stage remains sensitive to illumination changes, glare, and reflections, which can degrade contour quality and introduce false detections. Second, although the proposed nesting strategy produces valid and practically useful layouts, its packing efficiency on difficult irregular benchmarks remains below that of highly specialized state-of-the-art methods designed exclusively for offline optimization. These limitations indicate that, while the proposed system is effective as an integrated proof of concept and practical prototype, further improvements are needed to increase its robustness and optimization quality in more challenging production scenarios.

5.1 Future work

Future research should focus first on improving the robustness of material segmentation under realistic workshop conditions. In the current system, strong illumination changes, glare, and surface reflections remain the main causes of contour instability and false-positive detections. This limitation may be addressed through more advanced illumination normalization, adaptive thresholding, reflection-aware preprocessing, or learning-based segmentation approaches. In addition, controlled lighting and simple hardware measures for reflection suppression could further improve the reliability of scene acquisition.

A second important direction is the further development of the nesting algo-

rithm for irregularly shaped parts and segmented material regions. Although the current heuristic is sufficiently fast and produces valid layouts, its packing density can be improved by incorporating stronger refinement stages, such as iterative local search, contour-aware compaction, or hybrid optimization strategies that combine fast constructive placement with subsequent improvement procedures. Parallel and multi-stage optimization schemes are also promising for larger and more geometrically complex instances.

Finally, the proposed system should be evaluated on a wider variety of materials, textures, and industrial scenarios in order to better assess its generalizability. Future work should also include user-oriented studies of projection-based guidance, with particular attention to operator accuracy, task completion time, and cognitive load. Such investigations would make it possible to evaluate the system not only as a computational method, but also as a practical human-centered tool for real cutting workflows.

Appendix A

A.1 HybridNest pseudocode

The complete HybridNest packing pipeline is summarized in Algorithm 1, while the compact single-piece insertion routine is given in Algorithm 2.

Algorithm 1 HybridNest multi-start packing procedure

Require: Object $O = (W, H, \Omega)$, pieces $\mathcal{P}$, seed s
Ensure: Best layout L^*

- 1: $(\mathcal{P}_{fit}, \mathcal{P}_{unfit}) \leftarrow \text{SPLITCUTSTOFIT}(W, H, \mathcal{P})$
- 2: **if** $\mathcal{P}_{fit} = \emptyset$ **then**
- 3: **return** $\emptyset$
- 4: **end if**
- 5: sort $\mathcal{P}_{fit}$ by decreasing area
- 6: $\Delta \leftarrow \text{CALCULATEADAPTIVECELLSIZE}(\mathcal{P}_{fit}, W, H)$
- 7: build allowed raster mask A^Δ from Ω
- 8: determine number of attempts T
- 9: initialize best statistics (C^*, A^*, U^*, K^*) and best layout L^*
- 10: **for** $t = 1$ to T **do**
- 11: $\pi_t \leftarrow \text{GETSORTINGSTRATEGY}(t, \mathcal{P}_{fit}, s)$
- 12: initialize packer B_t with $(W, H, \Omega, \Delta, A^\Delta)$
- 13: $A_t \leftarrow 0, I_t \leftarrow \emptyset$
- 14: **for all** $p \in \pi_t$ **do**
- 15: **if** $\text{INSERTPIECE}(B_t, p)$ succeeds **then**
- 16: update placed area A_t and placed-id set I_t
- 17: **end if**
- 18: **end for**
- 19: **for** $g = 1$ to G **do**
- 20: $\mathcal{R} \leftarrow$ pieces not yet placed
- 21: **if** $\mathcal{R} = \emptyset$ **then**
- 22: **break**
- 23: **end if**
- 24: reorder $\mathcal{R}$ for gap filling
- 25: $q \leftarrow 0$
- 26: **for all** $p \in \mathcal{R}$ **do**
- 27: **if** $\text{INSERTPIECE}(B_t, p)$ succeeds **then**
- 28: update A_t, I_t , and q
- 29: **end if**
- 30: **end for**
- 31: **if** $q = 0$ **then**
- 32: **break**
- 33: **end if**
- 34: **end for**
- 35: **if** post-compaction is enabled **then**
- 36: $\text{SHRINKENVELOPE}(B_t)$ and remove accidental overlaps
- 37: **end if**
- 38: compute:
 $C_t =$ number of placed pieces
 $U_t =$ used strip length / horizontal span
 $K_t =$ bounding-box compactness term
- 39: **if** $\text{BETTERCANDIDATE}(C_t, A_t, U_t, K_t; C^*, A^*, U^*, K^*)$ **then**
- 40: update L^* and best statistics
- 41: **end if**
- 42: **if** early-stop condition holds **then**
- 43: **break**
- 44: **end if**
- 45: **end for**
- 46: record final metrics
- 47: **return** L^*

Algorithm 2 Compact insertion of one piece in HybridNest

Require: Packer state B , piece $p = (w, h, s, \text{id}, \text{name})$

Ensure: true if placed, else false

```
1: build footprint candidates  $\mathcal{F}$ :
2: if  $s = \text{circle}$  then
3:   create one disk footprint
4: else if  $s \in \{\text{poly}, \text{library}\}$  then
5:   create polygon footprints for allowed rotation angles
6: else
7:   create rectangular footprint at  $0^\circ$ 
8:   if rotation is allowed and  $w \neq h$  then
9:     add rectangular footprint at  $90^\circ$ 
10:  end if
11: end if
12: for all  $F \in \mathcal{F}$  do
13:    $\mathcal{C} \leftarrow \text{FINDPOSITIONS}(B, F)$ 
                                      $\triangleright$  bottom-left scan, corner positions, adjacency, skyline, quick scan
14:   if  $\mathcal{C} = \emptyset$  then
15:     continue
16:   end if
17:   best candidate  $\leftarrow \emptyset$ 
18:   for all  $(x, y, \sigma) \in \mathcal{C}$  up to an evaluation cap do
19:     convert  $(x, y)$  from grid cells to millimeters
20:     if piece exceeds container bounds in millimeters then
21:       continue
22:     end if
23:      $(\hat{x}, \hat{y}) \leftarrow (x, y)$ 
24:     if  $x > 0$  or  $y > 0$  then
25:        $(b_x, b_y) \leftarrow \text{SLIDE TOWARDS ORIGIN}(x, y, F)$ 
26:       if  $(b_x, b_y)$  stays within bounds and fits mask then
27:          $(\hat{x}, \hat{y}) \leftarrow (b_x, b_y)$ 
28:       else if  $(x, y)$  does not fit mask then
29:         continue
30:       end if
31:       else if  $(x, y)$  does not fit mask then
32:         continue
33:       end if
34:       define candidate key:
35:        $\kappa = (\hat{y} + 0.35\hat{x}, -\sigma)$ 
36:       if no best candidate stored yet or  $\kappa$  is better then
37:         store  $(\hat{x}, \hat{y})$  as current best
38:       end if
39:     end for
40:     if a best candidate exists then
41:       if  $\text{PLACE}(B, F, \hat{x}, \hat{y}, p)$  succeeds then
42:         return true
43:       end if
44:     end if
45:   perform fallback raster scan with adaptive step
46:   for all fallback positions  $(x, y)$  do
47:     if position is inside millimeter bounds and fits mask then
48:       if  $\text{PLACE}(B, F, x, y, p)$  succeeds then
49:         return true
50:       end if
51:     end if
52:   end for
53: return false
```

Bibliography

- [1] Antonio Albano and Giuseppe Sapuppo. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(5):242–248, 1980.
- [2] R. Alvarez-Valdés, A. Martínez, and J. M. Tamarit. A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, 145(2):463–477, 2013.
- [3] Gwon Hwan An, Siyeong Lee, Min-Woo Seo, Kugjin Yun, Won-Sik Cheong, and Suk-Ju Kang. Charuco board-based omnidirectional camera calibration method. *Electronics*, 7(12):421, 2018.
- [4] Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980.
- [5] Brenda S. Baker and Jerald S. Schwarz. Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12(3):508–525, 1983.
- [6] J. E. Beasley. An exact two-dimensional non-guillotine cutting tree search procedure. *Operations Research*, 33(1):49–64, 1985.
- [7] Bengt-Erik Bengtsson. Packing rectangular pieces—a heuristic approach. *The Computer Journal*, 25(3):353–357, 1982.
- [8] Julia A. Bennell and José F. Oliveira. The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397–415, 2008.
- [9] Julia A. Bennell and José F. Oliveira. A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, 60(S1):S93–S105, 2009.
- [10] Julia A. Bennell and Xiang Song. A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computers & Operations Research*, 35(1):267–281, 2008.
- [11] Julia A. Bennell and Xiang Song. A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, 16(2):167–188, 2010.
- [12] Judith O. Berkey and Peter Y. Wang. Two-dimensional finite bin-packing algorithms. *Journal of the Operational Research Society*, 38(5):423–429, 1987.

- [13] Tim Bosch, Gu van Rhijn, Frank Krause, Reinier Könemann, Eline S. Wilschut, and Monique de Looze. Spatial augmented reality: a tool for operator guidance and training evaluated in five industrial case studies. In *Proceedings of the 13th ACM International Conference on Pervasive Technologies Related to Assistive Environments*, pages 1–7, 2020.
- [14] Alexandros Bouganis and Murray Shanahan. A vision-based intelligent system for packing 2-D irregular shapes. *IEEE Transactions on Automation Science and Engineering*, 4(3):382–394, 2007.
- [15] Bart Braden. The surveyor’s area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.
- [16] Duane C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.
- [17] Edmund K. Burke, Robert S. R. Hellier, Graham Kendall, and Glenn Whitwell. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, 54(3):587–601, 2006.
- [18] Edmund K. Burke, Robert S. R. Hellier, Graham Kendall, and Glenn Whitwell. Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research*, 179(1):27–49, 2007.
- [19] Edmund K. Burke, Graham Kendall, and Glenn Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671, 2004.
- [20] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
- [21] Keong-Hun Choi and Jong-Eun Ha. An adaptive threshold for the canny edge with weak label. *Applied Sciences*, 15(22):12158, 2025.
- [22] N. Christofides and C. Whitlock. An algorithm for two-dimensional cutting problems. *Operations Research*, 25(1):30–44, 1977.
- [23] David Claus and Andrew W. Fitzgibbon. A rational function lens distortion model for general cameras. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 1, pages 213–219, 2005.
- [24] François Clautiaux, Jacques Carlier, and Aziz Moukrim. A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, 183(3):1196–1211, 2007.
- [25] Daniel Domović, Tomislav Rolich, and Marin Golub. Evolutionary hyper-heuristic for solving the strip-packing problem. *The Journal of The Textile Institute*, 110(6):825–840, 2019.

- [26] David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973.
- [27] Kathryn A. Dowsland, Subodh Vaid, and William B. Dowsland. An algorithm for polygon placement using a bottom-left strategy. *European Journal of Operational Research*, 141(2):371–381, 2002.
- [28] Harald Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.
- [29] Jens Egeblad, Benny K. Nielsen, and Allan Odgaard. Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 183(3):1249–1266, 2007.
- [30] Sándor P. Fekete and Jörg Schepers. A general framework for bounds for higher-dimensional orthogonal packing problems. *Mathematical Methods of Operations Research*, 60(2):311–329, 2004.
- [31] Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.
- [32] Herbert Freeman and Ruth Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, 18(7):409–413, 1975.
- [33] S. Garrido-Jurado, Rafael Muñoz-Salinas, Francisco J. Madrid-Cuevas, and Manuel J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014.
- [34] Andreas Geiger, Frank Moosmann, Omer Car, and Bernhard Schuster. Automatic camera and range sensor calibration using a single shot. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, 2012.
- [35] A. Miguel Gomes. Irregular packing problems: Industrial applications and new directions using computational geometry. In *IFAC Workshop on Intelligent Manufacturing Systems*, volume 46, pages 378–383, 2013. Published in IFAC Proceedings Volumes.
- [36] A. Miguel Gomes and José F. Oliveira. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, 171(3):811–829, 2006.
- [37] Baosu Guo, Yu Zhang, Jingwen Hu, Jinrui Li, Fenghe Wu, Qingjin Peng, and Quan Zhang. Two-dimensional irregular packing problems: A review. *Frontiers in Mechanical Engineering*, 8:966691, 2022.

- [38] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.
- [39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.
- [40] E. Hopper and B. C. H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2d packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.
- [41] Takashi Imamichi, Mutsunori Yagiura, and Hiroshi Nagamochi. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. *Discrete Optimization*, 6(4):345–361, 2009.
- [42] Manuel Iori, Vinícius Loti de Lima, Silvano Martello, and Michele Monaci. 2dpacklib: a two-dimensional cutting and packing library. *Optimization Letters*, 16:471–480, 2022.
- [43] Stefan Jakobs. On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, 88(1):165–181, 1996.
- [44] Edward G. Coffman Jr., Michael R. Garey, David S. Johnson, and Robert E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.
- [45] José Marcato Junior, Marcus Vinícius Antunes de Moraes, and Antonio Maria Garcia Tommaselli. Experimental assessment of techniques for fisheye camera calibration. *Boletim de Ciências Geodésicas*, 21(3), 2015.
- [46] S. C. H. Leung, D. Zhang, C. Zhou, and T. Wu. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. *Computers & Operations Research*, 39(1):64–73, 2012.
- [47] Stephen C. H. Leung, Defu Zhang, and Kwang Mong Sim. A two-stage intelligent search algorithm for the two-dimensional strip packing problem. *European Journal of Operational Research*, 215(1):57–69, 2011.
- [48] Yanbo Li, Xing Zhao, Liang Wang, Peng Zhang, and Yan Du. An improved adaptive genetic algorithm for two-dimensional rectangular packing problem. *Applied Sciences*, 11(1):413, 2021.
- [49] Wenwei Lin, Peidong Liang, Guantai Luo, Ziyang Zhao, and Chentao Zhang. Research of online hand-eye calibration method based on charuco board. *Sensors*, 22(10):3805, 2022.
- [50] V. M. M. Marques, C. F. G. Bispo, and J. J. S. Sentieiro. A system for the compaction of two-dimensional irregular shapes based on simulated annealing. In *Proceedings of IECON '91: 1991 International Conference on Industrial Electronics, Control and Instrumentation*, pages 1911–1916, 1991.

- [51] Marco Ojer, Hugo Alvarez, Ismael Serrano, Fátima A. Saiz, Iñigo Barandiaran, Daniel Aguinaga, Leire Querejeta, and David Alejandro. Projection-based augmented reality assistance for manual electronic component assembly processes. *Applied Sciences*, 10(3):796, 2020.
- [52] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.
- [53] Said Pertuz, Domenec Puig, and Miguel A. García. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415–1432, 2013.
- [54] Massimo Piccardi. Background subtraction techniques: A review. In *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.
- [55] Xiao ping Liao, Junyan Ma, Chengyi Ou, Fengying Long, and Xiangsha Liu. Visual nesting system for irregular cutting-stock problem based on rubber band packing algorithm. *Advances in Mechanical Engineering*, 8(6):1–15, 2016.
- [56] Troels A. Rasmussen and Timothy Merritt. Projectables: Augmented CNC tools for sustainable creative practices. *International Journal of Architectural Computing*, 16(3):227–242, 2018.
- [57] Francisco J. Romero-Ramirez, Rafael Muñoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38–47, 2018.
- [58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.
- [59] Azriel Rosenfeld and John L. Pfaltz. Sequential operations in digital picture processing. *Journal of the ACM*, 13(4):471–494, 1966.
- [60] Mariadas Capsran Roshan, Mats Isaksson, and Adrian Pranata. A geometric calibration method for thermal cameras using a charuco board. *Infrared Physics & Technology*, 138:105219, 2024.
- [61] Daniel Saakes, Thomas Cambazard, Jun Mitani, and Takeo Igarashi. PacCAM: Material capture and interactive 2d packing for efficient material usage on furniture design. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, pages 441–446, 2013.
- [62] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.
- [63] Seppe Sels, Bart Ribbens, Steve Vanlanduit, and Rudi Penne. Camera calibration using gray code. *Sensors*, 19(2):246, 2019.

- [64] Jean Paul Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
- [65] K. Shiangjen, J. Chaijaruwanich, W. Srisujjalertwaja, P. Unachak, and S. Somhom. An iterative bidirectional heuristic placement algorithm for solving the two-dimensional knapsack packing problem. *Engineering Optimization*, 50(2):347–365, 2018.
- [66] Chris Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999.
- [67] Satoshi Suzuki and Keiichi Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.
- [68] Shunji Umetani, Mutsunori Yagiura, Shinji Imahori, Takashi Imamichi, Koji Nonobe, and Toshihide Ibaraki. Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research*, 16(6):661–683, 2009.
- [69] Pieter Vanneste, Yi Huang, Jung Yeon Park, Frederik Cornillie, Bart Decloedt, and Wim Van den Noortgate. Cognitive support for assembly operations by means of augmented reality: an exploratory study. *International Journal of Human-Computer Studies*, 143:102480, 2020.
- [70] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.
- [71] Lijun Wei, Dongfeng Zhang, and Qingsheng Chen. A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research*, 36(5):1608–1614, 2009.
- [72] Wei-Chu Weng and Hsin-Chuan Kuo. Irregular stock cutting system based on AutoCAD. *Advances in Engineering Software*, 42(9):634–643, 2011.
- [73] Sheng-Quan Xie, Gary Wang, and Yan Liu. Nesting of two-dimensional irregular parts: An integrated approach. *International Journal of Computer Integrated Manufacturing*, 20(8):741–756, 2007.
- [74] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.
- [75] Karel J. Zuiderveld. Contrast limited adaptive histogram equalization. In Paul S. Heckbert, editor, *Graphics Gems*, pages 474–485. Elsevier, 1994.