TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Bauingenieurwesen Institut für Bauinformatik

PROJECT WORK

Design a workflow for robotic arms to recognize and grasp irregular objects

Submitted by
Xianglin Zhang
06.07.2023 in Dresden
Matriculation number: 5049423

Responsible permanent academic staff and first examiner:

Prof. Dr.-Ing. habil. Karsten Menzel

Second examiner: Prof. Dr.-Ing. Raimar Scherer

Supervisor: Dipl.-Ing. Shaowen Han

Dresden, 06. 07. 2023

Declaration of Originality

I confirm that this assignment is my own work and that I have not sought or used the inadmissible help of third parties to produce this work. I have fully referenced and used inverted commas for all text directly quoted from a source. Any indirect quotations have been duly marked as such.

This work has not yet been submitted to another examination institution - neither in Germany nor outside Germany - neither in the same nor in a similar way and has not yet been published.

Name: Xianglin

Vorname: Zhang

Matrikelnummer: 5049423

Dresden, 06. 07. 2023

Abstract

As the construction industry grapples with escalating labor costs and increasing demands for construction efficiency and quality, the integration of advanced technologies, including Artificial Intelligence (AI) and robotics, offers promising solutions. This paper proposes a novel workflow that empowers robotic arms to recognize and install irregularly shaped slabs, an essential process in the construction industry. The proposed workflow design comprises various stages, including image segmentation, feature extraction, shape matching, and final instructions for robotic arm movements. This method effectively tackles critical issues inherent in each stage, yielding promising results in terms of accuracy, flexibility, and robustness. Furthermore, the developed application has excellent portability, allowing operation on standard industrial robotic arms, and includes a graphical user interface for ease of adaptation to different construction environments. The proposed system was validated through simulations and real-world experiments, demonstrating high performance in terms of accuracy and stability, hence promising significant enhancements in construction efficiency, labor cost reduction, and safety improvement. This research paves the way for a broader application of robotics in the construction industry, with the potential to revolutionize traditional construction processes.

Keywords: Construction robot; Artificial Intelligence (AI); object detection; Automation; Shape Matching; Irregular Shape Recognition

Contents

1 Introduction

In light of the accelerated advancements in technology, Artificial Intelligence (AI) and robotics have attained groundbreaking progress across various domains. In the construction industry, AI and robotic technologies have demonstrated immense potential for enhancing construction efficiency, reducing labor costs, and ensuring construction quality (Yue Pan and Limao Zhang 2021).

Concurrently, as modern construction projects continuously demand higher standards for construction environments and accuracy, the construction industry faces unprecedented challenges. Under these circumstances, the technology for robotic slab installation has become increasingly critical, as it offers effective solutions to these issues. Utilizing robots in the production process can substantially shorten production cycles and elevate efficiency while simultaneously reducing error rates and ensuring construction quality. This can be attributed to the high precision and stability of robots, which can circumvent human-induced errors. In on-site construction tasks, employing robots can alleviate repetitive labor and manual labor intensity, thereby improving workers' efficiency. Robots are also capable of completing a multitude of tasks within short periods, significantly reducing project durations and saving time and costs for enterprises. Moreover, robots possess the ability to work for extended periods in complex, harsh construction environments, meaning they can continuously operate under adverse conditions such as high altitudes, high temperatures, and high humidity, thus diminishing the risk of personnel injuries. In comparison to humans, robots are better equipped to adapt to these stringent environments while maintaining stable working conditions. Researching how to utilize robots for replacing manual labor in floor slab installations has emerged as a crucial development trend in the construction industry. Various advanced AI and robotic technologies are gradually being integrated into the construction industry, providing construction enterprises with an efficient, low-cost, and safe construction scheme (Bock 2007).

At present, top-down processes are predominantly employed in the field of construction robots, owing to their simplicity, minimal requirement for complex sensory programs (such as computer vision), and execution of minimal specialized knowledge (Parascho 2023). Although this process is highly effective for certain construction techniques, unexpected issues frequently arise during the manufacturing process, particularly considering the complexities of construction sites. Material tolerances may accumulate, minor errors in robot settings may

likewise result in significant discrepancies between designs and actual structures, and changes in structural environments may lead to unforeseen collisions or inaccuracies. By entirely predefining processes, people cannot respond to any variations (Parascho 2023). Current market slab installation robots are primarily limited to performing repetitive labor, with weaker capabilities for recognizing and installing irregularly shaped slabs. This hinders the promotion and application of robots in broader contexts. Therefore, designing a workflow that enables robotic arms to recognize and install irregularly shaped slabs is of paramount necessity (Parascho 2023).

In order to address this issue, several aspects warrant further research and exploration, which are delineated as follows: firstly, the design of a workflow is essential to ascertain the tasks that need to be accomplished. Subsequently, the selection of appropriate computer vision algorithms is required, based on the tasks, to guide the robotic arm in grasping and installing floor slabs. Following the selection of suitable algorithms, the development of corresponding applications should be undertaken to fulfill construction requirements. These applications should possess a graphical user interface, allowing for convenient fine-tuning according to varying construction environments, and exhibit excellent portability, enabling operation on established industrial robotic arms. Lastly, the accuracy and robustness of the developed applications should be verified through simulations and experiments.

By implementing this innovative workflow, the advantages of robots in the construction domain can be fully exploited, improving construction efficiency, reducing labor costs, minimizing safety hazards, and satisfying the requirements for high-quality construction. This will usher in new opportunities for the development of the construction industry, creating greater value for society.

2 State of the art

2.1 Construction Robotic Arm

The construction industry accounts for 10-20% of the Gross Domestic Product in most countries, making it the largest sector of the economy in terms of employment; Construction work is labor-intensive and often carried out under hazardous conditions, with frequent changes in tasks and materials. Robotic arms, however, can contribute to improved work efficiency, reduced work errors,

diminished risk of worker injury, lower labor costs, and environmental protection (Gharbia et al. 2020). Consequently, the technology of robotic arms in construction represents a significant shift from traditional construction methods. The utilization of robots may present a range of opportunities that can alter the way we design and construct buildings.

To better understand the trends and trajectory of robotic arm applications in on-site construction, Marwan Gharbia and colleagues conducted a systematic review of 52 articles identified through the PRISMA protocol and meta-analysis (Gharbia et al. 2020). The results indicate that the technology of robotic arms in on-site construction is a continuously evolving application field, with additive manufacturing (AM), automated installation systems, automated robot assembly systems, autonomous robot assembly, and robotic bricklaying appearing to be the most researched subjects, and potentially influencing the development of research on construction robotic arms (Gharbia et al. 2020).

In the field of robotic arm construction, the current approach is predominantly through human-machine collaborative mode (Gharbia et al. 2020). For example, the glass ceiling installation robot, composed of a mobile platform and a robotic arm, includes hardware and software such as human-machine interaction interface devices, designed to tackle the challenges encountered when installing glass panels at high altitudes (Lee et al. 2008). Another example is an automated facade installation system for construction sites, combining a multi-degree-of-freedom robotic arm with a small excavator (Lee et al. 2007). Moreover, there are studies utilizing the flexibility of robotic arms, creating complex walls composed of 3D printed materials using mobile and multi-jointed robots (Furet et al. 2019).

Hence, there currently exists a research gap regarding the automated installation of traditional construction materials. This paper will design a workflow for identifying and installing irregular construction objects such as roof panel with a combination of semantic segmentation method and conventional computer vision methods, and develop an application based on VisionPro that can be used with KUKA robotic arms. This will achieve basic irregular model recognition and matching and output the pose of the robotic arm.

The task of the robotic arm to automatically identify irregular plate-shaped objects and install them according to the preset template can be decomposed into four parts: 1) Camera and Hand-Eye Calibration; 2) Foreground segmentation; 3) Object detection; 4) Compare the object with the template and output the mechanical the position of the arm grabbing and placing, and the five parts will be introduced separately next.

2.2 Camera and Hand-Eye Calibration

Camera calibration and hand-eye calibration for robotic arms are two key processes that play vital roles in ensuring accurate operations when visual systems are employed. The objectives of these processes are to align the movements of the robotic arm with the image data acquired by the vision system, thus enabling the robotic arm to accurately position itself and interact with the target object.

Camera calibration is aimed at determining the intrinsic and extrinsic parameters of the camera. The intrinsic parameters include factors such as focal length and image center, which reflect the camera's own properties, while the extrinsic parameters include rotation and translation parameters, reflecting the camera's position and orientation with respect to the world coordinate system. Through camera calibration, we can convert image coordinates to camera coordinates, thus understanding the position of an object in the camera's coordinate system (Remondino and Fraser 2006).

Hand-eye calibration is the process of determining the relative position and orientation between the end of the robotic arm (the "hand") and the camera (the "eye"). The crux of this process lies in finding a transformation matrix that can convert points in the camera coordinate system to the robotic arm coordinate system. Therefore, when the robotic arm moves, we can understand the position of the object observed by the camera in the robotic arm coordinate system, allowing the robotic arm to accurately move to the target position (Enebuse et al. 2022).

Therefore, the relationship between camera calibration and hand-eye calibration is that both are designed to solve the problem of spatial correspondence in the visual system of the robotic arm. Specifically, we first need to use camera calibration to determine the position of an object in the camera coordinate system, then use hand-eye calibration to convert this position to the robotic arm coordinate system, so that the robotic arm can accurately execute tasks. Consequently, these two steps are often conducted together; they depend on each other and jointly ensure the accurate operation of the robotic arm.

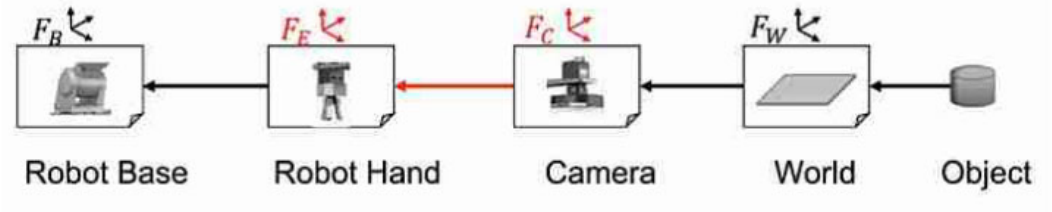

Fig 1 The relationship between coordinaes (Enebuse et al. 2022)

2.2.1 Camera Calibration

Currently, the commonly used camera calibration algorithms include Tsai's method, Heikkila & Silven's method, and Zhang's method. Next, we will introduce their principles, application scenarios, as well as advantages and limitations respectively.

a) Tsai Camera Calibration Method

Tsai Camera Calibration Method is a well-established technique for determining the parameters of a pinhole camera model, which has been widely used due to its efficiency and accuracy. It is proposed by Roger Y. Tsai in 1987 (Tsai 1987) and it is a two-step procedure.

First, it decouples the intrinsic and extrinsic parameters by using the perspective projection's properties, allowing the radial lens distortion to be ignored initially. Second, it estimates the lens distortion parameters. Tsai's method models the camera as a combination of a linear perspective projection followed by a nonlinear radial distortion. The method first establishes the linear part using a set of control points with known 3D positions, and their corresponding 2D projections in the image. This model includes the camera's internal parameters (such as focal length, principal point, and aspect ratio), as well as its external parameters (rotation and translation). Once the linear model is established, the method estimates the radial distortion parameters using the residuals (i.e., the difference between the projected 2D positions and the measured 2D positions).

Tsai's method is useful in situations where the camera setup is static, the radial distortion is not extreme, and you have a control object with known geometry to calibrate against. It's also beneficial when you need a fast, relatively accurate calibration without user intervention.

However, there are a few limitations to Tsai's method:

1. It assumes a pinhole camera model, which might not be accurate for wide-angle lenses with significant distortion.
2. It assumes the radial distortion is symmetrical around the principal point, which may not be the case if the lens is not correctly aligned with the image sensor.
3. It might not provide the most accurate results if the scene depth variation is significant, as the camera's perspective distortion might not be accurately modeled with a single set of parameters.
4. The accuracy of Tsai's method depends heavily on the control points'

quality. If these points are not accurately chosen, the calibration results can be significantly off.

5. It assumes that the image plane (sensor) is not skewed, which might not be accurate for all cameras, especially older ones.
6. It is sensitive to measurement noise. Small errors in the control point coordinates can lead to significant errors in the estimated camera parameters.
7. It assumes a single viewpoint for all control points. If there are occlusions, or the control points are not all visible from the same viewpoint, the method cannot be applied. Despite these limitations, Tsai's method remains a useful tool for quick and efficient camera calibration in many practical situations.

b) Heikkila & Silven Calibration Method

Heikkila and Silven Camera Calibration Method is another prominent approach for camera calibration, which considers a wider range of distortions and achieves more accurate results in many cases. It is proposed by Janne Heikkila and Olli Silven in 1997 and it is an effective camera calibration method. It handles both radial and tangential distortions, making it more suitable for wide-angle lenses and other lenses where the radial and tangential distortions are significant (J. Heikkila and O. Silven 1997).

This method is applicable to various machine vision applications, but it is most beneficial in camera-based 3D measurement and robot vision because high-precision geometric accuracy is required. The program uses an empirical inversion model for image correction, which can accurately compensate for radial and tangential distortion. Finally, the author also provides a Matlab toolbox for performing this calibration process, which can be obtained through the internet (J. Heikkila and O. Silven 1997).

Heikkila and Silven's method models the camera by an extended pinhole model, which includes five distortion parameters: two radial distortion parameters and three tangential distortion parameters. The calibration is achieved through a non-linear optimization process that minimizes the sum of squared differences between the observed and predicted image points (J. Heikkila and O. Silven 1997).

The method starts by initializing the parameters with an approximation based on a direct linear transformation (DLT) from control points with known 3D positions and their corresponding 2D projections in the image. Then, the method iteratively refines the parameters by minimizing the residuals, that is, the difference between the predicted and measured 2D positions (Heikkila and

Silvén).

The Heikkila and Silven method is particularly useful for cameras with significant radial and tangential distortions, such as wide-angle lenses, fisheye lenses, and lenses that might not be perfectly aligned with the image sensor. It can provide more accurate results than methods that only consider radial distortion, such as Tsai's method.

However, the Heikkila and Silven method also has some limitations: 1)It requires a non-linear optimization process, which can be more computationally intensive and slower than methods that only involve linear operations. 2) It might be sensitive to the initial approximation of the parameters. If the initial approximation is too far from the true parameters, the optimization process might not converge to the optimal solution. 3)The accuracy of the method still depends on the quality of the control points. If these points are not accurately chosen, the calibration results can be significantly off. The method assumes that the image plane (sensor) is not skewed, which might not be accurate for all cameras, especially older ones.

c) Zhang’s Calibration Method

Zhang’s Calibration Method is a well-known camera calibration method proposed by Zhengyou Zhang in 1998 (Zhang 2000). This method is widely adopted due to its simplicity, efficiency, and high precision. It’s a versatile approach that requires the camera to observe a planar pattern from at least two different orientations. The pattern does not need to be manufactured with high precision, allowing for great flexibility in practical use. It can be a chessboard pattern, dot pattern, or any other planar pattern with discernible features.

The key idea behind Zhang's method is to use a planar homography to relate the image points with the model points. By observing the planar pattern from at least two different orientations, the homography between the image points and the model points can be estimated. This homography can then be used to extract the camera's intrinsic and extrinsic parameters, including focal length, principal point, skew coefficient, and distortion coefficients (Zhang 2000).

Zhang's method is essentially a two-step process. The first step is the calibration, where the homographies are estimated and the camera parameters are extracted. The second step is the refinement, where the parameters are further refined using non-linear optimization to minimize the re-projection error (Zhang 2000).

Zhang's method is a very flexible and robust calibration method that can be used in a variety of situations. It's particularly suitable when the camera's

distortion is significant, or when the camera is observing a scene from multiple orientations. It is widely used in computer vision applications, including 3D reconstruction, image-based rendering, and robotic vision.

However, Zhang's method also has some limitations: 1) It assumes that the lens distortion is radially symmetric, which may not be accurate for all cameras. 2) The precision of the calibration depends on the number and distribution of the control points. If these points are not uniformly distributed across the image, the calibration results can be inaccurate. 3) The method requires the observation of a planar pattern from at least two different orientations. If this condition is not met, the method cannot accurately calibrate the camera.

Based on the preceding discussion, it can be deduced that Zhang's calibration methodology exhibits robustness to a diverse range of positions and orientations for chessboard captures. This implies that Zhang's method can still provide relatively accurate calibration results even under suboptimal capture conditions such as limited chessboard mobility or minor alterations in the chessboard's attitude. In contrast, the methods proposed by Tsai and Heikkila & Silven might necessitate broader movements and attitude changes for the chessboard to achieve precise outcomes. Moreover, Zhang's approach requires only a single chessboard calibration plate, making it exceptionally convenient for users. The calibration process merely entails photographing the chessboard at different locations and attitudes, without requiring precise knowledge of the chessboard's position and orientation. This is a clear departure from Tsai's and Heikkila & Silven's methods, which might demand complicated setup and meticulous measurements. Practically speaking, Zhang's method often results in a superior level of calibration precision. The method considers all internal parameters, including radial and tangential lens distortion, and uses a more intricate optimization process to ascertain these parameters. In contrast, Tsai's approach accounts only for radial lens distortion, while Heikkila & Silven's method, despite considering tangential distortion, employs a relatively simplistic optimization procedure. Therefore, considering potential variations in illumination conditions on construction sites and the possibility of uneven terrain leading to uncertainty in the calibration board's position and attitude, the camera calibration for the workflow will be based on Zhang's calibration algorithm.

The RCNN, as the earliest deep learning-based detector, operates as a two-stage object detection framework. It leverages region proposal methods to generate potential bounding boxes within an image and subsequently employs a classifier to analyze and classify these proposed boxes (Girshick, R., Donahue, J., Darrell, T., & Malik, J. 2014, 2015). The architecture of RCNN can be conceptually divided into two distinct stages.

Stage 1: Feature Extraction from Region Proposals: By utilizing selective search, 2000 to 3000 region proposals are generated for images (Van de Sande, K. E., Uijlings, J. R., Gevers, T., & Smeulders, A. W. 2011, November). These candidate region proposals undergo a rescaling process and are subsequently passed through a CNN model, resulting in the extraction of a high-dimensional feature vector of 4096 dimensions.

Stage 2: Classification and localization: Linear SVM classifiers are employed within the RCNN framework to predict the presence of objects within each region proposal and perform object category recognition tasks (Chang, Y. W., & Lin, C. J. 2008, December).

As the pioneer in two-stage detectors, RCNN demonstrated a substantial improvement in mean Average Precision (mAP), elevating it from 33.7% (achieved by DPM-v5) to a notable 58.5%. (Ren, X., & Ramanan, D. 2013). However, owing to the extensive number of region proposals and the involved training process of RCNN, this method necessitates substantial time and storage memory resources.

In order to overcome these inherent limitations, SPP-Net effectively mitigated the challenges by incorporating the principles of Spatial Pyramid Matching (SPM) and introducing a novel CNN architecture to solve the problem of content losses and unwanted geometric distortions due to the cropping or warping operation during feature extraction process of RCNN (He, K., Zhang, X., Ren, S., & Sun, J. 2015; Lazebnik, S., Schmid, C., & Ponce, J. 2006, June). SPP-Net achieves superior outcomes by accurately estimating region proposals at various scales, and concurrently enhances detection efficiency during the testing phase through shared computation cost among different proposals before the SPP layer.

While SPP-Net has successfully enhanced the detection speed, it is not devoid of certain limitations. Firstly, its speed is still limited by the two-stage architecture. Secondly, in the case of SPP-Net, the fine-tuning process is exclusively applied to its fully connected layers, neglecting any adjustments to the preceding layers. Similar to SPP-Net, Fast RCNN also processes the entire image using convolutional layers to generate feature maps. (Girshick 2015). In the Fast R-CNN framework, both the image and multiple regions of interest (RoIs) are fed into a fully convolutional network for processing and subsequent analysis. (Moghaddam, B., Biermann, H., & Margaritis, D. 1999, June). Every region of RoI within the Fast R-CNN framework undergoes pooling operations to generate fixed-size feature maps. Then these feature maps are subsequently mapped to feature vectors through fully connected layers. The network produces two distinct output vectors for each RoI: SoftMax probabilities indicating class probabilities and bounding-box regression offsets specific to each class. (Rottmann, M., Colling, P., Hack, T. P., Chan, R., Hüger, F., Schlicht, P., & Gottschalk, H. 2020, July; Dickerson 2017). Fast R-CNN

facilitates end-to-end training of all network layers by employing a multitask loss function. It effectively addresses the need for additional storage space while enhancing both accuracy and efficiency through the adoption of more rationalized training schemes. However, the speed of detection is still constrained by the proposal detection phase, which imposes limitations on overall performance.

Faster-RCNN solved the above problem by changing extraction method of region proposals (Ren, S., He, K., Girshick, R., & Sun, J. 2015), the selective search in Fast-RCNN is replaced by region proposal network (RPN). The introduction of Faster R-CNN signifies a significant breakthrough in the field of object detection, as it enables the training of region proposal-based CNN architectures in an end-to-end manner, which enables Faster-RCNN to be capable of achieving near-real-time performance. Meanwhile, a remarkable frame rate of 5 frames per second on a GPU is achieved, while concurrently delivering state-of-the-art object detection accuracy as demonstrated by impressive mAP scores of 73.2% on PASCAL VOC07 and 70.4% on VOC12 datasets. (Zeiler, M. D., & Fergus, R. 2014). Nevertheless, it is important to note that the training algorithm of Faster R-CNN remains computationally intensive, and the RPN tends to generate object-like regions that encompass both objects and backgrounds, rather than specifically identifying individual object instances, or confronting with objects exhibiting extreme scales or unconventional shapes.

More advanced detectors, such as R-FCN (Dai, J., Li, Y., He, K., & Sun, J. 2016; Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. 2017) and Feature Pyramid Networks (FPN) (Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. 2017), but they are more or less limited in the detection speed because of their inherent two-stage framework. Region proposal-based frameworks typically consist of multiple interrelated stages, encompassing region proposal generation, CNN-based feature extraction, classification, and bounding box regression. These stages are commonly trained as separate entities. Despite advancements such as Faster R-CNN, the need for alternative training methods persists to achieve shared convolutional parameters between the Region Proposal Network (RPN) and the detection network. Consequently, the time required to handle these distinct components becomes a limiting factor in real-time applications. To ameliorate this circumstance, one-stage detectors have been proposed.

2.2.2 HAND-EYE CALIBRATION

After performing camera calibration, it is necessary to perform hand-eye calibration for the robotic arm. In the multifaceted field of robot control guided by machine vision, the process of hand-eye calibration occupies a significant position. Hand-eye calibration is aimed at precisely estimating the relative position and

orientation between the robot's end effector—referred to as the "hand"—and the vision system, or the "eye". This calibration process is paramount for a myriad of applications such as part assembly, bin picking, and inspection operations, as it enables the robot to accurately obtain environmental information for precise task execution (Enebuse et al. 2022).

This chapter will commence with a comprehensive exploration of the importance and applications of hand-eye calibration, elucidating its role in robot control guided by machine vision. Subsequently, we will introduce the calibration procedure for an "eye-in-hand" setup, which involves estimating the position and orientation of the camera within the robot's end effector through a sequence of steps. This process involves complex mathematical calculations, such as solving homogeneous transformation equations, which may involve rotational and translational noise that needs to be rectified through appropriate calibration processes.

Following this, we will focus on the principles and procedure of "eye-to-hand" calibration, where the camera is positioned external to the robot arm, and the robot performs operations within the field of view. We will investigate the characteristics of hand-eye calibration in this configuration, including its advantages and challenges in application. This process also presents several challenges, including how to deal with rotational and translational noise, and how to consider the impact of the robot's motion range on calibration accuracy.

a) eye-in-hand

   "Eye-in-hand" calibration refers to a setup where a camera is attached directly to the robot's end-effector, essentially being the "eye" in the robot's "hand". This setup allows the robot to perceive the environment directly from its point of action, providing a close-up view of the tasks it is performing, such as object manipulation or precise assembly tasks (Enebuse et al. 2022).

   The calibration process of this setup is crucial as it aims to accurately estimate the pose (position and orientation) of the camera relative to the robot's end-effector. This pose is typically represented by a homogeneous transformation matrix that describes the rotation and translation of the camera relative to the end-effector (Enebuse et al. 2022).

   The calibration process involves capturing images of a calibration object with known geometry from different poses of the robot. The 2D-3D correspondences between the image points and the known object points are then used to compute the camera poses. Simultaneously, the robot poses are obtained from the robot's joint encoders. With pairs of camera and robot poses, the hand-eye calibration problem becomes solving a homogeneous transformation equation, which can be more challenging due to the noise

present in rotation and translation components of the pose (Enebuse et al. 2022).

Various methods have been proposed to solve this problem, broadly classified into two categories: separate and simultaneous methods (Enebuse et al. 2022). The separate methods first estimate the rotation component and then the translation component, based on the estimated rotation. However, errors from rotation estimates directly propagate to the translation estimates in these methods, and they also lose the inherent coupling between rotation and translation. Therefore, simultaneous methods were proposed that estimate both components at the same time, providing more accurate results in the presence of noise.

The Separated Methods (Shiu and Ahmad 1989; Tsai and Lenz 1989) is a common approach to addressing the hand-eye calibration problem in robot vision guidance systems. In this method, the rotation and translation parameters are estimated separately. Here are the basic steps of the Separated Methods:

1. Estimation of Rotation Parameters: First, the rotation parameter is estimated from the relationship between the movements of the robot arm and the camera. This is typically done by comparing the relative rotations of the arm and the camera recorded at two different points in time.

2. Estimation of Translation Parameters: After estimating the rotation parameters, these parameters are then used to estimate the translation parameters. This step is done by applying the rotation parameters to the relative movement between the robot arm and the camera, and then comparing the relative translations recorded at two different points in time.

It's important to note that in the Separated Methods, errors in the rotation estimates directly affect the estimation of translation parameters, so in practical applications, the rotation parameters need to be estimated as accurately as possible. Furthermore, since the Separated Methods treat rotation and translation parameters separately, this method may lose the inherent coupling between these two parameters. Therefore, in some applications that require a high degree of accuracy, the Simultaneous Methods might need to be considered, which deals with rotation and translation parameters at the same time (Enebuse et al. 2022).

The Simultaneous Methods (H. Chen 1991) is another approach for addressing the hand-eye calibration problem in robot vision guidance systems. Unlike the Separated Methods, the Simultaneous Methods estimates the rotation and translation parameters at the same time. Here are the basic steps of the

Simultaneous Methods:

1. Simultaneous Estimation of Rotation and Translation Parameters: In this step, both the rotation and translation parameters are estimated simultaneously from the relationship between the movements of the robot arm and the camera. This is typically done by comparing the relative transformations (both rotation and translation) of the arm and the camera recorded at two different points in time.

The Simultaneous Methods can be more complex to implement than the Separated Methods, but they have the advantage of preserving the inherent coupling between rotation and translation parameters. This means that they can be more accurate in applications where this coupling is significant. Additionally, because the rotation and translation parameters are estimated simultaneously, errors in the rotation estimates are not directly propagated to the translation estimates, potentially making the Simultaneous Methods more robust to errors in rotation estimation. Some examples of the Simultaneous Methods include the screw motion approach and the dual quaternion method, among others (Enebuse et al. 2022).

b) Eye-on-base

The "eye-on-base" or "eye-to-hand" calibration procedure refers to the process where the camera (eye) is positioned externally or away from the robot arm (hand). The robot operates within the field of view of the camera (Jiang et al. 2022).

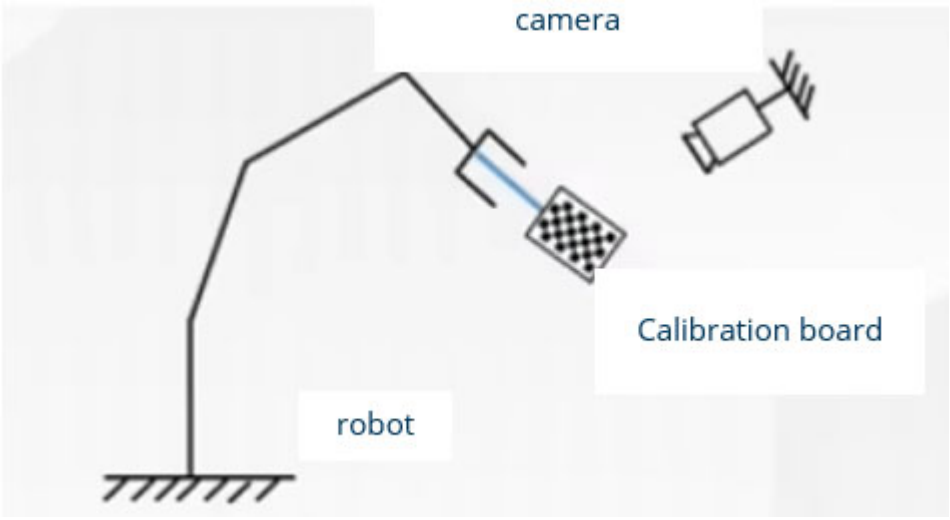

Fig 2 Schematic diagram of eye-to-hand calibration

The KUKA educate ready2 was used for simulation and experimentation in this project. The camera position was fixed on a base, so the eye-on-base calibration method was used to obtain the positional relationship between the robotic arm

and the camera.

By affixing a calibration board at the end of the robotic arm actuator, we maneuver the arm to present the board in two distinct positions and orientations within the camera's field of view, capturing images at each point. Utilizing these captured images for camera calibration, we derive the internal parameters of the camera and correct for lens distortions. Following this, the images of the calibration board at two locations, combined with the angles of each joint of the robotic arm, allow for the calculation of two homogeneous transformation matrices from the camera coordinates to the calibration board's pose: ${}^{End}_{Robot1}T$ and ${}^{End}_{Robot2}T$. Additionally, while setting the end-effector positions, we can also compute the homogeneous transformation matrices from the robot base to the calibration board: ${}^{End}_{Robot1}T$ and ${}^{End}_{Robot2}T$. The aforementioned four known quantities maintain the following relationship:

$$ {}^{End}_{Robot1}T * {}^{Robot1}_{Camera1}T * {}^{Camera1}_{object}T = {}^{End}_{Robot2}T * {}^{Robot2}_{Camera2}T * {}^{Camera2}_{object}T \quad (1.) $$

This can be further simplified as:：

$$ {}^{End}_{Robot2}T^{-1} * {}^{End}_{Robot1}T * {}^{Robot1}_{Camera1}T = {}^{Robot2}_{Camera2}T * {}^{Camera2}_{object}T * {}^{Camera1}_{object}T^{-1} \quad (2.) $$

Where ${}^{Robot1}_{Camera1}T$ and ${}^{Robot2}_{Camera2}T$ represent the transformation matrices from camera coordinates to the base coordinates, which are the precise matrices we seek to determine through the hand-eye calibration process.

2.3 Foreground Segmentation

This section is dedicated to an in-depth examination of concepts and methodologies pertaining to Foreground Segmentation, a fundamental technique within the domain of computer vision and image processing. The objective of foreground segmentation is to segregate foreground objects from the background milieu within an image. This facilitates a concentrated focus on objects of interest, thereby enabling a comprehensive analysis and interpretation.

We shall commence with an exploration of the theoretical underpinnings, expounding upon the nature of foreground segmentation and the imperatives driving its application. Subsequently, we will present a survey of prevalent foreground segmentation algorithms, including but not limited to threshold-based segmentation, cluster-based segmentation, and edge detection-based segmentation, while concurrently discussing their respective strengths, limitations, and appropriate application scenarios.

Proceeding further, informed by the specific requirements of our project, we will elucidate our rationale for selecting a particular foreground segmentation algorithm, and its practical implementation and efficacy within the project's context.

2.3.1 Threshold-based segmentation

Threshold segmentation represents one of the ubiquitous methods employed in image processing. This technique differentiates foreground from the background by comparing image grayscale values with a predetermined threshold (Pare et al. 2020). Following are the commonly employed variations of threshold segmentation:

a) Global threshold-based segmentation: This method treats the entire image as a holistic entity, segmenting the image into foreground and background through the application of a uniform global threshold. This method demonstrates efficacy when there are distinct disparities in grayscale values between the foreground and the background.

b) Local threshold-based segmentation: This approach subdivides the image into numerous smaller regions, each associated with a corresponding local threshold. Such an approach is particularly adept at handling images with uneven illumination or complex backgrounds. It is primarily applicable when the image is characterized by uneven lighting or a complex backdrop.

c) Adaptive threshold-based segmentation: This technique adaptively selects varying thresholds for segmentation, contingent upon the grayscale characteristics of different regions within the image. It exhibits superior handling of illumination changes and noise. The primary applicability of this method arises when the image is subject to changes in illumination, noise, and similar problems.

d) Multi-threshold segmentation: This methodology applies multiple distinct thresholds to carry out multiple segmentation iterations, thereby achieving more refined results. It is most suitably employed when there is a requirement for a higher degree of detail in the results.

e) Cluster-based segmentation: This approach clusters similar pixel points within the image, thereby achieving effective segmentation. It is best suited for scenarios where the image requires effective segmentation and contains similar pixel points.

In the experiment of this project, we have adopted a specialized adaptive thresholding technique known as the Dynamic Soft Thresholding Method. With

this approach, the threshold is no longer a fixed value but is dynamically adjusted according to the local characteristics of the image. This method is particularly suitable for handling situations where the threshold is difficult to determine due to factors such as changes in illumination and noise. However, it should be noted that the term "soft threshold" may imply a degree of flexibility or ambiguity in the selection of the threshold. For instance, rather than simply classifying pixels as foreground or background, a "membership score" may be assigned based on the proximity of the pixel value to the threshold. This can allow the model to better handle pixels that are close to the threshold (Pare et al. 2020).

By coordinating with controlled ambient lighting and background conditions, this method has achieved satisfactory results in the segmentation of the foreground.

2.3.2 Semantic segmentation

However, in a construction environment, there are often interferences such as background and lighting. Using threshold segmentation for foreground segmentation is clearly insufficient. To eliminate the effects of factors like lighting and background, semantic segmentation can be used. Semantic segmentation is a task within computer vision that seeks to understand images at a pixel level, determining to which category or object each pixel belongs. Specifically, semantic segmentation divides an image into multiple regions, each representing an object or background with specific semantics (Guo et al. 2018). Semantic segmentation has a wide range of applications in numerous fields. For instance, in autonomous driving, semantic segmentation can identify different objects, such as roads, pedestrians, and vehicles, helping autonomous vehicles understand their surroundings (Yurtsever et al. 2020). In medical image analysis, semantic segmentation can identify and separate lesion areas in images, assisting physicians in making diagnoses (Shen et al. 2017). In remote sensing image processing, semantic segmentation can identify different types of ground covers, such as buildings, bodies of water, and vegetation from satellite images (Yuan et al. 2021). The three main methods of semantic segmentation are region-based semantic segmentation (Uijlings et al. 2013b), semantic segmentation based on fully convolutional neural networks (Long et al. 2015) and weakly supervised semantic segmentation (Pathak et al. 2016).

Fully convolutional neural networks (FCNs) typically outperform many other methods in semantic segmentation tasks for several reasons (Pathak et al. 2016):

a) End-to-end training and prediction: Compared to some region-based or sliding window methods, FCNs can process an entire image at once, classifying pixels across the whole image. This not only reduces computational requirements

but also mitigates boundary effect issues that arise from partitioning or sliding windows.

b) Efficiency: FCNs complete predictions by learning a specific feature hierarchy, providing more efficient computation than other methods that require repeated calculations of the same features.
c) Preservation of spatial information: After conducting convolution operations, FCNs typically use up-sampling (deconvolution) operations to restore the original resolution, thereby preserving spatial information throughout the network, a critical factor in pixel-level tasks such as semantic segmentation.
d) Scalability: By adding more convolution layers or employing more complex structures (such as skip architectures and residual connections), FCNs can enhance their performance, making them more adaptable to complex semantic segmentation tasks.
e) Rich feature learning: Owing to the characteristics of deep learning, FCNs can learn more complex and rich image features, enabling superior performance in complex tasks compared to traditional methods based on manual features.

However, FCNs also have their limitations. For instance, they require a large number of annotated data for training, the acquisition of which can pose a challenge. Furthermore, although FCNs can process an entire image, high-resolution images may require significant computational resources. Due to these limitations, there is a need for appropriate annotation and training for different segmentation objects in engineering applications.

However, the Segment Anything Model (SAM) substantially addresses this issue, proposing a new image segmentation task, model, and dataset. This model, based on the FCN architecture, supports real-time segmentation mask output and can perform zero-sample transfer to new image distributions and tasks via hints. The dataset includes over 100 million masks and 11 million images, making it one of the largest image segmentation datasets to date (Kirillov et al. 2023).

Researchers used two experiments to evaluate this model: segmentation quality assessment and zero-sample transfer experiments. In the segmentation quality assessment experiment, researchers conducted extensive evaluations of SAM using 23 different segmentation datasets. The results showed that SAM could generate high-quality masks from a single foreground point, with the segmentation quality often only slightly lower than the manually annotated standard. Additionally, human studies were conducted to evaluate the similarity between the masks generated by SAM and those generated by humans. The results indicated a high degree of similarity between the two (Kirillov et al. 2023), demonstrating that SAM could effectively mimic human segmentation behavior.

In the zero-sample transfer experiment, researchers utilized multiple downstream tasks, including edge detection, object proposal generation, instance

segmentation, and text-to-mask prediction. The results showed that, through the use of hints, SAM could be directly used to solve these tasks without any additional training (Kirillov et al. 2023). Furthermore, some exploratory experiments were conducted to assess SAM's representation learning capabilities. The results demonstrated that SAM's representation could be used for multiple purposes such as data annotation, dataset content understanding, and feature extraction for downstream tasks.

Therefore, in this project, SAM could be applied to segment images captured by the camera, yielding masks of irregular graphics to be grabbed. Fig 3 compares the segmentation effects of the threshold segmentation method and the SAM model on irregular polygons in images under conditions of shadow. Fig 4 presents the masks obtained by applying SAM to segment the objects to be grabbed in the robotic arm simulation software, after removing the pure black background. It is evident that using this model can effectively eliminate the impact of illumination and background on image segmentation, and it demonstrates good transferability.

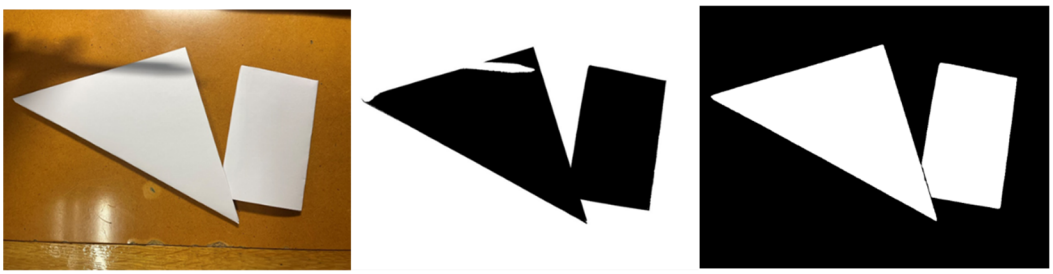

Fig 3 Threshold Segmentation Method and SAM Applied to Objects with Uneven Illumination

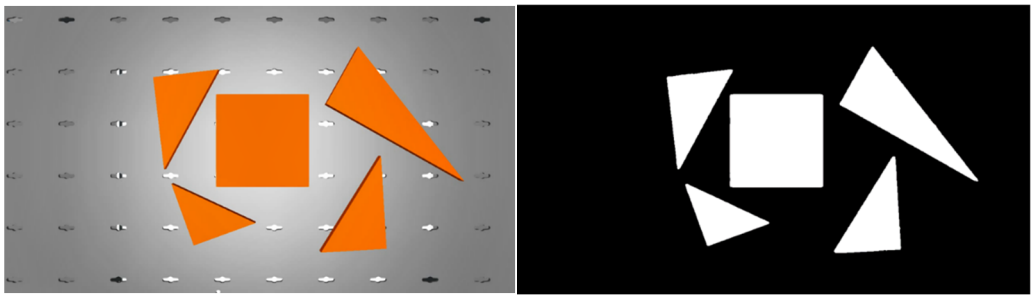

Fig 4 SAM Segmentation of Objects Against Complex Backgrounds.

2.4 Object detection

The history of object detection methods can be divided into three stages (Zou et al. 2023): traditional methods, machine learning-based methods, and deep learning-based methods; 1) Traditional methods: In the early days of computer vision, object detection mainly relied on hand-crafted feature extraction

algorithms and simple classifiers. Representatives of traditional methods include sliding window-based methods, region extraction-based methods, and segmentation-based methods. The advantages of traditional methods are their relative simplicity and lower computational complexity, but they tend to have lower accuracy and robustness; 2) Machine learning-based methods: With the development of machine learning technology, object detection methods began to employ machine learning algorithms for classification. These methods still use hand-crafted features, but the performance of classifiers has been significantly improved. Representatives of machine learning-based methods include Support Vector Machines (SVMs) and Random Forests. The advantages of these methods are their relatively higher accuracy and robustness, but feature extraction relies on manual design, and their generalization ability is limited; 3) Deep learning-based methods: With the breakthroughs in deep learning technology, particularly the emergence of Convolutional Neural Networks (CNNs), object detection methods have undergone revolutionary changes. Deep learning-based methods can automatically learn high-level features of images, improving the accuracy and robustness of object detection; Representatives of this class of methods include the R-CNN series, YOLO series, and SSD. The advantages of deep learning methods are high accuracy and real-time performance, suitable for various complex scenarios, but they have higher computational complexity and require strong hardware support (Zou et al. 2023).

In the following sections, this chapter will introduce the general principles and applica-tions of various object detection methods and select the most suitable object detection method for detecting irregular shapes. We will conduct a detailed comparison and analysis of these methods to provide an appropriate solution for the irregular panel detection task.

2.4.1 Sliding window detection

Sliding window is a conventional detection method in computer vision. The basic idea of sliding window is to slide a fixed-size window over the image and classify each sub-image within the window as object or non-object. The size of the window and the stride (the amount by which the window moves) can be adjusted to detect objects of different sizes. The advantage of sliding window is that it can detect objects at different scales and locations in an image. It is also a simple and intuitive method that can be easily implemented. However, it can be computationally expensive, especially when dealing with large images or searching for objects at multiple scales. To address this issue, researchers have developed various techniques to speed up sliding window detection, such as using feature pyramids

to reduce the number of windows to be evaluated, or using deep learning models to learn more efficient representations of images (Amit et al. 2020).

Based on this approach, two typical and far-reaching methods have been invented. These two methods are the Histogram of Oriented Gradients (HOG) and the Deformable Part Model (DPM). The Histogram of Oriented Gradients (HOG) is a feature descriptor used for object detection in computer vision. It extracts features by calculating the gradient direction and magnitude of each pixel in the image and combines this information into a histogram (Dalal and Triggs 2005). HOG detectors use multiple scaling of the input image to detect objects of different sizes and have become an important foundation for many object detectors and computer vision applications. The HOG detector employs a sliding window to move through the image, calculating feature vectors at each position, and then using a classifier to classify the vector to determine whether an object is present (Dalal and Triggs 2005).

The Deformable Part Model (DPM) is a traditional object detection method that uses deformable parts to describe an object's shape and appearance. DPM is considered one of the paradigms of traditional object detection methods. It represents an object as a collection composed of multiple parts and uses a Support Vector Machine (SVM) classifier to classify each part (Felzenszwalb et al. 2008). It utilizes a sliding window to search for objects in the image and represents them as collections of multiple parts. DPM has been widely applied to various computer vision tasks, such as pedestrian detection and vehicle detection (Hsiao et al. 2009).

2.4.2 CNN-Based detection

Traditional object detection methods typically employ hand-crafted features to represent images, which are often a combination of low-level information such as edges, colors, and textures. However, this approach presents several issues, such as the tedious process of manual feature design requiring specialized knowledge and the difficulty in representing higher-level semantic information of objects (Yamashita et al. 2018).

In contrast, object detection algorithms based on deep Convolutional Neural Networks (CNNs) possess superior feature representation capabilities. CNNs can automatically learn image features and better represent information such as the shape and texture of objects (Yamashita et al. 2018). Additionally, CNNs can extract higher-level semantic information through multi-level abstraction. Ross Girshick and colleagues proposed a simple and scalable detection algorithm (Girshick et al. 2014) that utilizes deep Convolutional Neural Networks (CNNs) to

extract image features and employs these features for object detection. Experimental results demonstrate that, on the PASCAL VOC 2012 dataset, this algorithm improves the mean Average Precision (mAP) by more than 30%, reaching 53.3%. Compared to previous best results, this algorithm exhibits higher detection accuracy and better scalability.

Based on CNN, the RCNN (Regions with Convolutional Neural Networks method) was proposed to further improve detection accuracy. RCNN, a convolutional neural network-based object detection algorithm, is one of the first methods to apply deep learning to object detection. It mainly consists of two stages: candidate region generation and feature extraction and classification. In the candidate region generation stage, RCNN employs the Selective Search algorithm to generate candidate regions that may contain target objects. Selective Search is an image segmentation-based algorithm that can divide an image into multiple regions and merge them into candidate regions that may contain target objects based on the similarity between these regions. In the feature extraction and classification stage, RCNN uses a convolutional neural network (CNN) to extract features from each candidate region and classifies them using a support vector machine (SVM). Specifically, RCNN first scales each candidate region to a fixed size and extracts its features through a CNN model (such as AlexNet). Then, for each category, a binary SVM classifier is trained to determine whether the category is present in the current candidate region (Zou et al. 2023).

Support Vector Machine (SVM) is a binary and multi-classification algorithm used to identify which category a data point belongs to among two categories. It is achieved by finding an optimal separating hyperplane that maximally disperses samples of the two categories. SVM implements classification by finding an optimal hyperplane in a given dataset. The hyperplane is a decision boundary that divides data points into two categories in some sense. SVM requires the decision boundary to be as far away from data points as possible, resulting in better classification performance. An important concept in support vector machines is the "support vector." Support vectors are data points with the shortest distance to the decision boundary. These points are used to determine the position of the decision boundary and are therefore referred to as anchor points of the decision boundary. SVM constructs the decision boundary by solving a convex optimization problem. This convex optimization problem requires finding a hyperplane that maximizes the distance between the decision boundary and support vectors. This is the core idea of support vector machines and the reason they are called "support vector machines" (Evgeniou and Pontil 2001).

Jasper R. R. Uijlings and colleagues proposed an object detection algorithm called Selective Search (Uijlings et al. 2013a). This algorithm is an image segmentation-based method that generates a large number of candidate regions potentially

containing target objects for subsequent feature extraction and classification. Specifically, Selective Search employs multiple complementary image segmentation strategies to generate candidate regions and combines these regions into hierarchical region sets. This method can capture various types of objects, including rigid, non-rigid, and amorphous objects. The advantage of the Selective Search algorithm is its ability to generate high-quality, data-driven, class-independent target locations, and it can improve the performance of machine learning techniques and appearance models in object recognition by reducing the number of candidate locations. Moreover, Selective Search can be combined with powerful bag-of-words models to further enhance object recognition performance. Finally, the algorithm has been publicly released and widely applied in the computer vision field.

RCNN performs well in detection accuracy, but it is relatively slow. Subsequently, algorithms such as Fast RCNN and Faster RCNN have improved upon it, achieving improvements in both speed and accuracy (Arulprakash and Aruldoss 2022).

In practical applications, object detectors may encounter varying backgrounds, dataset discrepancies, insufficient generalization to new datasets, and threats from adversarial examples. This implies that when the object detector operates in a novel environment, it may fail to accurately recognize targets or generate false alarms. Moreover, adversarial examples may deceive the object detector by introducing subtle modifications to the images. Some object detection algorithms necessitate the use of region proposal methods, which could result in high computational costs and lengthy processing times. Region proposal methods are techniques for generating candidate object locations within an image. However, these methods require multiple iterations over the entire image, thus incurring high computational costs and extended processing times. Some algorithms struggle with the localization of small objects. Due to the limited number of pixels small objects occupy in an image, they are often challenging to accurately locate and identify. This is because many object detection algorithms rely on feature maps for object detection and may fail to generate sufficient feature maps for small objects. Certain algorithms require a sequential training framework for classification and bounding box regression, which may lead to extended training times. These algorithms necessitate classification and bounding box regression for each object, potentially consuming considerable computational resources and time. Additionally, since these algorithms employ sequential training, they may require longer training durations (Arulprakash and Aruldoss 2022).

For irregular panel laying tasks, the dimensions and locations of each panel are given in advance. When there are no interfering factors in the background, after performing foreground segmentation, the foreground becomes the target of the robotic arm operation. Moreover, when detecting targets, if a neural network-

based object detection method is adopted, a large dataset is required for training due to the diversity of panel materials.

Irregular panels can be abstracted as irregular polygons, and the features of polygons, such as centroids and areas, can be extracted using conventional object detection methods with sufficient accuracy. Therefore, after segmenting the foreground, this project will use the blob analysis algorithm to extract the geometric features of the foreground. Then, by comparing the geometric features with the template (Template matching), we will determine the target of a single operation of the robotic arm.

2.5 Template Matching

Template matching finds wide application in the fields of image processing, computer vision, medical image analysis, and brain imaging. It is a classic and fundamental method used to evaluate the similarity between different objects or images. Template matching can be used in conjunction with other object recognition methods as a preprocessing or postprocessing step to enhance the accuracy of target detection and recognition. Template matching can be divided into region-based methods and feature-based methods (Hashemi et al. 2016).

Region-based template matching methods are commonly referred to as correlation methods or template matching, originally developed by Fonseca et al (Fonseca and Manjunath 1996). This approach performs well when there are no strong features between the template and the image, as it directly operates on pixel values. The matching is measured by computing the intensity values of the image and the template. Matching scores are extracted using methods such as squared difference, correlation-based approaches, optimization techniques, and mutual information. In some template matching problems, direct matching of the template to the target image is not feasible. Region-based methods are suitable for images with no evident details but have distinct feature information, such as color/grayscale rather than shape/size (Fonseca and Manjunath 1996).

On the other hand, feature-based methods are applicable in situations where matching is based on structural information rather than intensity information. Feature-based template matching methods are suitable for structural information matching rather than intensity-based matching. This approach typically employs feature detection algorithms to extract key points and descriptors from the image, which are then matched with key points and descriptors from the template. These descriptors can be extracted using algorithms such as SIFT(scale-Invariant Feature Transform) (Lowe 2004), SURF(Speeded Up Robust Features) (Herbert Bay et al. 2008), ORB(Oriented FAST and Rotated BRIEF) (Rublee et al. 2011), etc. During the

matching process, distance metrics such as Euclidean distance, Hamming distance, etc., are commonly used to calculate similarity scores between two descriptors. Finally, by comparing all scores and selecting the highest scoring match, the best match is determined. Feature-based template matching methods perform well when dealing with images that have complex textures and shape variations, exhibiting higher robustness and accuracy (Hashemi et al. 2016).

In this project, Hu moments is used as a feature of the object to compare with the template. Hu moments are seven moment invariants generated from the raw moments of an image, also known as Hu Moment Invariants. They were first proposed by Ming-Kuei Hu in his 1962 paper "Visual Pattern Recognition by Moment Invariants" (Hu 1962). These seven invariants remain constant with respect to image translation, scaling, and rotation, thereby possessing significant application value in the field of computer vision and image analysis. For instance, Hu moments are widely employed in image processing, with the most common applications being object recognition and image matching. In the context of object recognition, Hu moments can be used to identify specific objects within an image. For image matching, Hu moments are used to compare the similarity between two images (Yang et al. 2013). Hence, Hu Moment Invariants can represent the geometric information of a polygon, and then be utilized for matching the polygon with a template.

Before delving further into Hu moments, it is crucial to first understand the concept of image moments. In a two-dimensional function, a moment is obtained by multiplying the pixel intensity with a specific function of pixel coordinates and then summing up the results. Image moments can be divided into zero-order moments, first-order moments, second-order moments, etc., based on their order. Specifically, the zero-order moment represents the total pixel intensity of the image, the first-order moment is related to the image's centroid, while the second-order moments and higher are related to the shape of the image (Hu 1962). The seven invariants of Hu moments can be derived from Eqs. (3-9) (Yang et al. 2013):

$$\emptyset_1 = \eta_{20} + \eta_{02} \tag{3.}$$

$$\emptyset_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{4.}$$

$$\emptyset_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{21} - 3\eta_{03})^2 \tag{5.}$$

$$\emptyset_4 = (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2 \tag{6.}$$

$$\begin{aligned}\emptyset_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} - \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]\end{aligned} \tag{7.}$$

$$\emptyset_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{21} + \eta_{03})(\eta_{30} + \eta_{12}) \quad (8.)$$

$$\emptyset_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad (9.)$$

Among them, η_{pq} is the normalized version of the central moment, which is defined as follows:

$$\eta_{pq} = U_{pq}/U_{00}^{[\frac{p+q}{2}] + 1} \quad (10.)$$

U_{pq} is the center distance.

2.6 COMPUTER VISION LIBRARY

Upon determining the appropriate object detection algorithm, the subsequent step involves developing computer vision applications. To develop efficient and reliable computer vision applications, it is crucial to comprehend mainstream computer vision libraries. Computer vision libraries consist of a series of pre-written code modules that provide developers with various functionalities for image processing, object detection, and other machine vision tasks. By utilizing these libraries, developers can expedite the development process while ensuring code quality and stability.

Numerous mainstream computer vision libraries currently exist in the market, including but not limited to OpenCV, VisionPro, TensorFlow, and PyTorch. These libraries offer a wealth of tools and functions, encompassing various aspects of the computer vision field, such as image processing, feature extraction, object detection, and tracking.

2.6.1 OpenCV

OpenCV is a computer vision library that provides various image processing functions, including image pre-processing, feature extraction, image analysis, and recognition (Bradski and Kaehler 2008). In the field of robotic arms, OpenCV's image processing functions can be used in various application scenarios to enhance the intelligence and operation efficiency of the robotic arms. For example, the pre-processing function of OpenCV can be used to pre-process image data to make it have better features, so that the control program of the robotic arm can more effectively identify targets (Pásztó and Hubinský 2010).

Upon completion of image pre-processing, the application of OpenCV for the recognition of irregular polygons, and subsequent guidance for robotic arm manipulation, becomes feasible. This process primarily leverages two functionalities offered by OpenCV, namely edge detection and polygon approximation. These two algorithms, which are foundational to the operation of OpenCV in this context, will be briefly outlined as follows:

a) Edge detection is a technique in the field of image processing, aiming to identify the edges and contours in an image. The fundamental idea is to calculate the second derivative, or the gradient, of the image to pinpoint significant transitions within it. To be more specific, convolution is applied to the image, wherein the convolution kernel calculates the image gradient, yielding information regarding the direction of the image edges, gradient intensity, and the position of pixels on the edges. Subsequently, a threshold is selected based on gradient strength and direction to eliminate noise within the image that does not contribute to the edge information. Finally, the identification of the edges in the image is achieved through noise suppression. The Canny edge detection algorithm, a commonly used image processing approach, is proficient in extracting edge information from an image. The implementation process of this algorithm can be broadly divided into the following stages: Gaussian Smoothing: The image noise is eliminated using a Gaussian filter, smoothing the image. Gradient Calculation: The Sobel operator is applied to the image, resulting in the gradient magnitude and direction of the image. Non-Maximum Suppression: Redundant information in the gradient magnitude image is removed by performing non-maximum suppression. Double Thresholding: The gradient magnitude image undergoes processing based on two preset thresholds, retaining only the edges with higher intensity. Edge Linking: The edges with high intensity are linked, resulting in the final edge image (Bradski and Kaehler 2008).

b) The polygon fitting algorithm in OpenCV is a method used to fit curves and contours in input images. Polygon fitting usually requires edge detection on the input image, with continuity constraints applied to eliminate irrelevant edges. There are many algorithms for polygon fitting, including RANSAC (random sample consensus), Hough transform, and bilinear interpolation. These algorithms have different principles and implementation processes but generally have some robustness that can adapt to noise and deformation in the image. In OpenCV, cv2.fitLine() function can be used to fit line segments in an image, cv2.approxPolyDP() function can be used to fit curves in an image, cv2.minEnclosingCircle() function can be used to fit circles in an image, etc. The usage and parameter configuration of these functions need adjustment based on specific tasks and image features. Through polygon fitting, we can obtain the geometric shape of target objects in the image which enables us to achieve tasks

such as positioning recognition segmentation etc. In robotic arm control systems, polygon fitting technology can be utilized for identifying object shapes and positions while determining end-effector position accordingly (Bradski and Kaehler 2008).

2.6.2 VisionPro

VisionPro is a machine vision software developed by Cognex. It aims to provide a simple and efficient way to solve various machine vision tasks. VisionPro provides a range of powerful tools and features, including image analysis, feature extraction, object recognition, detection and positioning, as well as support for various machine vision hardware devices. VisionPro uses an object-based development environment which allows users to quickly create and test complex visual detection programs through drag-and-drop configuration of visual tools. This enables developers to focus more on solving actual visual detection problems rather than writing low-level code. In addition, VisionPro also supports programming languages such as C# and VB.NET which allows users to develop custom visual solutions flexibly. Furthermore, VisionPro provides a series of professional vision libraries that include various specialized tools for processing different types of visual tasks such as PmAlign (Pattern Matching Alignment), Blob (region analysis or connected component analysis), 1D/2D barcode reading, color processing, machine learning etc. Additionally, VisionPro supports various common industrial camera interfaces including GigE、USB、CameraLink etc., making it easy to integrate with different types of visual hardware devices. The following will introduce the visionpro algorithms mainly used in the design workflow of this article:

a) PMAlign

PMAlign is a tool used for image alignment and positioning in Cognex VisionPro software, whose full name is Pattern Matching Alignment. It is based on Cognex's patented pattern matching technology, also known as PatMax. This technology can be used to find and locate specific patterns or features in an image, regardless of whether these features are difficult to recognize due to rotation, scaling or slight deformation. The core function of PMAlign is to locate objects in the image by comparing the current image with a predefined pattern (Pattern). This predefined pattern can be an image fragment or a model that describes a specific shape or feature. When PMAlign finds an area in the current image that matches the preset pattern, it can generate a result describing the position and direction of that area, which can be used for image alignment and positioning. PMAlign has strong robustness and can handle various changes in images including rotation, scaling,

deformation and lighting changes. This makes it very useful in many practical industrial applications such as machine vision inspection, automatic assembly and robot positioning. For example, PmAlign can be used to design a visual positioning system for placing machines, utilizing machine vision technology to achieve high-speed and precise component placement (Wei and Jiao 2008).

b) Blob Analysis

The Blob (Binary Large OBjects) tool in VisionPro is a crucial module for image analysis and processing. It is a tool designed for handling and analyzing connected regions (referred to as "blobs" or "regions") in binary images. The Blob tool can extract individual connected regions from binary images and perform a variety of analyses and measurements on these regions. For instance, the Blob tool can calculate various geometric and shape features of each connected region, such as area, perimeter, centroid location, orientation, rectangularity, roundness, symmetry, and more. For example, in manufacturing, parts on the assembly line need to undergo visual inspection to ensure the accuracy of their shape and size. These parts have diverse shapes, indeterminate positions, and multiple parts may even appear in the field of view simultaneously. In this case, the Blob tool in VisionPro plays a vital role. Initially, by capturing and binarizing images from the assembly line, the Blob tool can identify and extract each connected region. Subsequently, the Blob tool computes geometric and shape features of each connected region, such as area, perimeter, centroid location, and orientation. Based on these features, we can infer the likely type of part that each connected region corresponds to. For example, features related to area and shape can assist us in determining whether the part is circular or rectangular, while the centroid location and orientation can help us ascertain the part's position and direction. As such, the Blob tool can enable automatic visual inspection of parts on the assembly line, thereby enhancing production efficiency and quality (Moeslund 2012).

Moreover, Cognex has entered into a collaboration with KUKA, a manufacturer of industrial robotic arms. The vision application used by KUKA robotic arms, known as VisionTech, is developed based on VisionPro. To enable the portability of tools to KUKA's robotic arm teach pendants, this paper will provide a detailed explanation of how to develop a program based on VisionPro for identifying and installing irregular roof panels. This program will leverage the precise control of KUKA robotic arms and the visual recognition capabilities of VisionPro to achieve automated installation of irregular roof panels.

3 Methodology

Through preceding discussions, the preliminary design of the workflow can be outlined as follows:

a) Initially, we affix the Cognex industrial camera at a predetermined position, establish a network setting, and connect it with the host computer. Throughout this process, it is critical to ensure the camera is installed securely and its field of view fully covers all possible working areas. In practical applications, issues such as disconnection of the camera or obstruction in its field of view might arise. If these situations occur, the program should issue error alerts, and operators should promptly inspect and restore the connection between the camera and the host computer while also ensuring an unobstructed view.

b) Subsequently, we proceed with camera calibration to determine the coordinate transformation relations and the conversion relationship between pixel space and actual space. During this stage, the precision of calibration is paramount, as any discrepancy can directly impact the effectiveness of all subsequent steps. If calibration results are found to be inaccurate, recalibration is necessary until satisfactory outcomes are achieved.

c) Next, we apply a dynamic soft thresholding or Segment Anything Model to pre-process the captured image to segment the foreground.

d) We then extract geometric features of the irregular target shapes from the pre-processed image. Precise feature extraction algorithms are required in this step to prevent influencing the shape matching results in later stages. If feature extraction fails or yields inaccurate results, the image quality and feature extraction algorithms need to be examined and adjusted accordingly.

e) Afterward, we read the vertices of template polygons from a CSV file and calculate the geometric features of irregular shapes in the template. This is because the CSV format, a simple text file format, can be readily opened and edited with various text editors and spreadsheet software, such as Microsoft Excel. This facilitates efficient management and reading of these files without requiring specific software or library support. If there are file errors, the program should issue alerts.

f) Having completed the feature extraction, we traverse through the features of different target irregular shapes, comparing them with the geometric features of shapes in the template to identify the most compatible template. This process necessitates precise matching algorithms, or it may lead to erroneous matching outcomes.

g) Upon identifying a compatible template, we compute the centroids and the principal axis direction of the paired shapes. The results of this step are crucial as they directly influence the subsequent robotic arm movement. If there are errors in the calculation of the centroid or principal axis direction, a detailed examination of the computational process and parameters is required to ensure accuracy.

h) We then calculate the translation and rotation angles of the target shape relative to the template. This step also demands precise computation, as any error could impact the performance of the robotic arm in terms of picking up and placing objects.

i) Finally, based on the calculated translation and rotation angles, we control the robotic arm to execute precise pick-and-place movements.

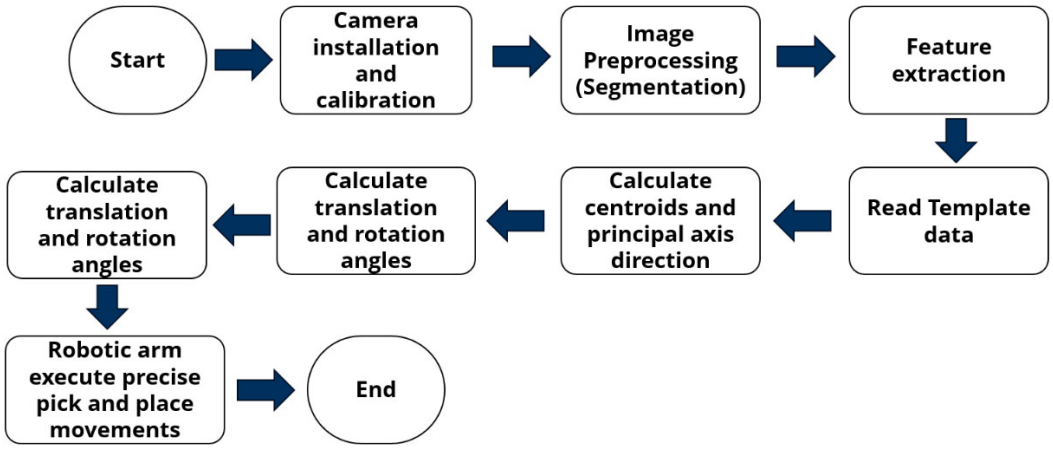

Fig 5 Preliminary design of the workflow

3.1 Camera Calibration

Given that the images captured by the camera are partitioned into small squares, each square referred to as a pixel. Each pixel has a unique coordinate, indicating its position in the image. On the other hand, the robotic arm's positioning uses actual coordinates. Therefore, it is necessary to calibrate the camera before undertaking vision-guided robotic arm movement tasks to establish the conversion relationship from pixel coordinates to actual coordinates, and to correct the camera's distortion.

In this study, we employ Zhang's calibration method for camera calibration. In this method, we first capture multiple images using a calibration board with a known plane. Next, we detect the feature points on the calibration board in these images, acquiring the positional information of these feature points. Subsequently, an

enclosed form of solution is utilized to estimate the camera's five intrinsic parameters and all extrinsic parameters. This solution is based on the geometric constraints of the known planar calibration board and matches the feature points in multiple images to yield the camera's intrinsic and extrinsic parameters. To enhance the accuracy of parameter estimation, the algorithm further optimizes all parameters, including lens distortion parameters, by minimizing an error function. This error function is defined by comparing the disparity between the projection positions of feature points on the calibration board in different images and their positions in the real world (Zhang 2000).

In VisionPro, camera calibration has been encapsulated into a tool block object called CogCalibCheckerboardTool. After completing the calibration steps in the graphical interface of the tool block, the image collected by the camera can be used as input. The tool block processes this input image and outputs an image that has corrected camera distortions, as well as a transformation matrix that converts pixel coordinates to real-world coordinates.

During calibration, the calibration board is first placed within the camera's field of view. Then, within the tool block, the Calibration Mode is set to linear, and in the Calibration Plate, the size of each grid of the calibration board chessboard is set in "Tile Size X" and "Tile Size Y". After these settings are completed, clicking "Run" completes the camera calibration.

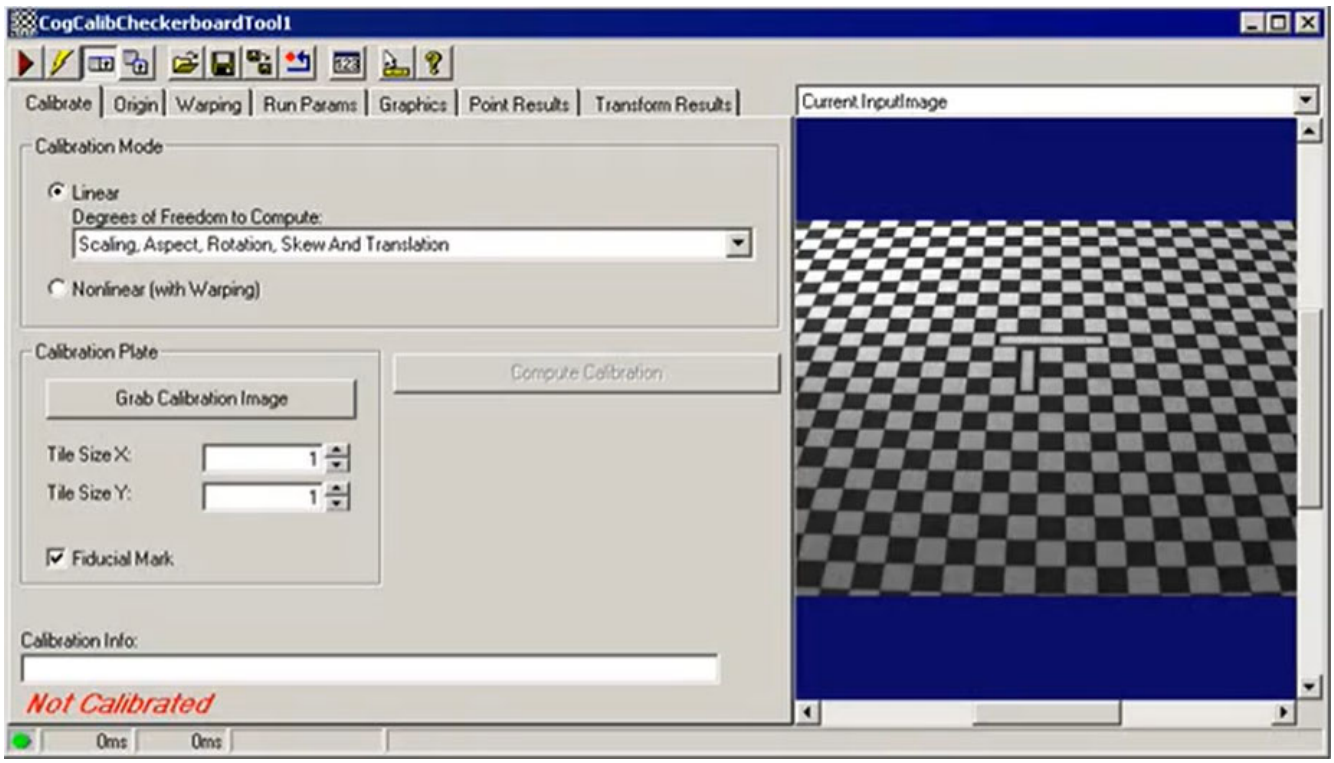

Fig 6 The graphical interface of the CogCalibCheckerboardTool

3.2 Image Preprocessing

In order to extract geometric information of irregular objects using an algorithm, it is necessary to pre-process the images captured by the camera. This includes binary image processing to separate the foreground and the background, reducing noise, emphasizing the geometric features such as the shape edges of the object, and simplifying image data. These steps make the Blob analysis more accurate.

The primary component used in this project is the Blob tool from VisionPro, which is responsible for the analysis and extraction of geometric feature information from irregular figures. Within the Blob analysis tool of VisionPro, a preprocessing step using threshold segmentation is already included. The tool employs a threshold method to differentiate objects (foreground) from the rest of the image (background), converting pixels below the threshold into white (representing objects), and pixels above the threshold into black (representing the background). This generates a binary image that serves as the foundation for further Blob analysis to extract geometric characteristics of the object. The process entails several preprocessing steps, including:

a) Image Cropping: The images are cropped to encompass the area of interest (in this case, the black background area). This procedure aids in diminishing irrelevant background information, thereby enhancing the precision of Blob analysis.
b) Binarization: As Blob tools typically handle binary images, the input images need to be binarized. The selection of the threshold for binarization can significantly affect the outcome of Blob analysis. In this context, we employed the 'Hard Threshold (Dynamic)' method for image binarization.
c) Morphological Operations: After the binary images have been processed, some morphological operations might be necessary. These operations, such as dilation, erosion, opening, and closing, can improve the shape characteristics of the image, allowing the Blob analysis to identify target objects more accurately. In this case, we used a filter to exclude noise by discarding spots with an area smaller than one hundred pixels.

Fig 7 Image Cropping in Blob Toolblock

However, for scenarios with complex backgrounds and lighting, the built-in threshold segmentation method of the blob tool may not process the image accurately. In this case, the previously introduced Sagment Anything Model can be employed to preprocess the image under complex lighting and background conditions, separating the foreground and background to generate a mask image. This mask image can then be fed into VisionPro for further processing. The image below contrasts the foreground segmentation results of irregular objects in a complex background simulated by KUKA.simpro, using both Otsu's threshold segmentation method and the Sagment Anything Model. As can be observed, the Sagment Anything Model effectively distinguishes the foreground from the background in the case of irregular objects within complex backgrounds.

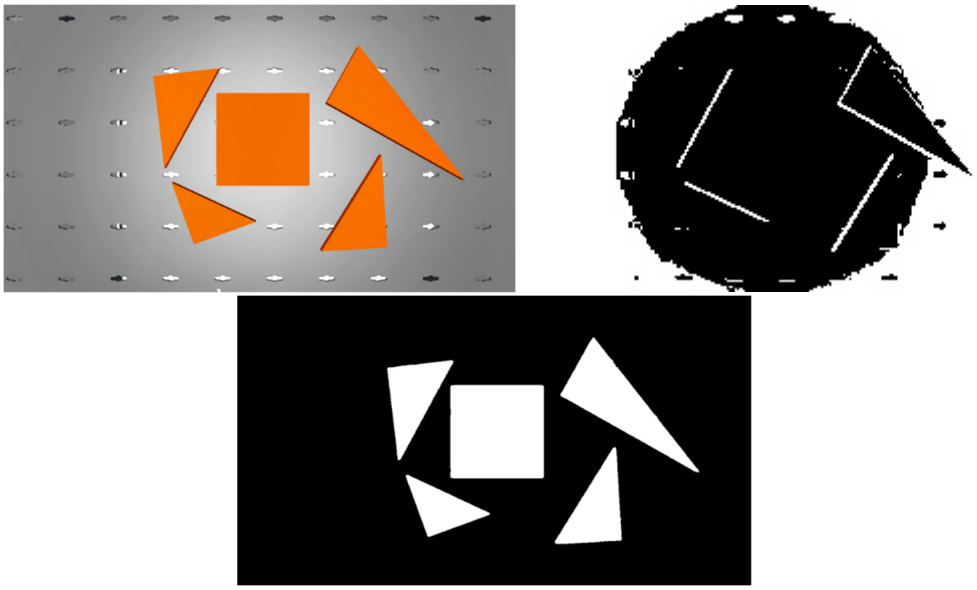

Fig 8 Otsu's threshold segmentation and SAM Segmentation of Objects Against Complex Backgrounds.

3.3 Irregular Objects Detection and Matching

After segmenting the foreground, interfering factors caused by illumination and background have been eliminated. The task now is to identify the objects to be grasped from the segmented images and assemble them according to the template. To accomplish this task, an algorithm needs to be selected to determine the suction point when the robotic arm grasps a single object with a suction cup, and place the object in the corresponding position in the template.

In a 2D scenario, irregular building boards can be abstracted as polygons of varying shapes and sizes. Therefore, the recognition and matching of irregular objects fundamentally involve identifying polygonal sections from images captured by the camera, extracting feature information of these polygons, and comparing it with the feature information of polygons in the templates. Upon locating the matching target polygon, the translation and rotation from the actual polygon to the target one are calculated to guide the robotic arm for picking and placement.

In the process of addressing this problem, three key issues need to be resolved:

1. How to extract target polygon information such as boundaries and vertices.

2. In the presence of multiple targets of different shapes, which algorithm should be used to match polygons and output the corresponding translation and rotation.

3. How to convert the output translation and rotation into a data type that the robotic arm can utilize and write the corresponding robotic arm motion program.

Given these three issues, this section first compares two target detection methods: Patmax and Blob Analysis, ultimately selecting Blob Analysis for the target detection task. Subsequently, it selects a shape matching algorithm based on Hu moments and develops the program.

3.3.1 Irregular Objects Detection

Following the discussions in section 2.1, this project adopts conventional Computer Vision (CV) techniques for irregular object detection tasks. In the realm of machine vision, there are principally two algorithms utilized for object detection: PatMax and Blob Analysis. The upcoming discussions will individually analyze these two methodologies, eventually leading to a selection between the two as per the task requirements for irregular object detection.

a) PatMax
   Section 2.2.2 introduced the pmAlign tool in the VisionPro software package, the core algorithm of which is PatMax. PatMax is a sophisticated geometric pattern matching or object positioning technology launched by Cognex in 1997.

It was developed to resolve object localization issues in image processing, particularly demonstrating high robustness in scenarios requiring handling of perspective changes, partial occlusion of the object, or variable distance between the object and the camera (Connolly 2003).

The core philosophy of the PatMax algorithm starkly contrasts traditional pixel-based model matching techniques. Rather than directly comparing pixel correlation with pixels in the training model, it extracts a set of boundary curves from the training model as features and then searches for these similar shapes in the image under processing. By such means, even if the target object undergoes rotation, scaling, or partial occlusion, as long as its boundary curves are still captured by the camera, PatMax can successfully localize the object.

Another notable characteristic of PatMax is its high precision. It can locate the direction of an object with an accuracy of 0.028 degrees (Connolly 2003).

The process of using PatMax for image matching generally involves the following steps:

1. Model Creation: First, an area is selected in the known reference image as a model. This model contains the main geometric features of the object, i.e., the features used for matching.
2. Feature Extraction: PatMax extracts a set of boundary curves from the selected model as geometric features. These features not only contain the shape information of the object but also the texture or detail information of the object.
3. Image Search: In the new image, PatMax uses the previously extracted features to search for objects similar to the model in the image. Because PatMax is based on geometric features, it can find objects in different directions and scales and can work effectively even when the object is partially occluded.
4. Result Reporting: Once an object matching the model is found, PatMax will report the position, size, and direction of the found object. In multi-object recognition scenarios, PatMax will report the information of all matching objects.

Therefore, if the PatMax algorithm is used as the algorithm for irregular target detection tasks, a reference image must be given in advance to extract features for detection. Therefore, PatMax can only perform single target detection. In this project, it is necessary to detect multiple targets with different shapes simultaneously, so PatMax may not be applicable.

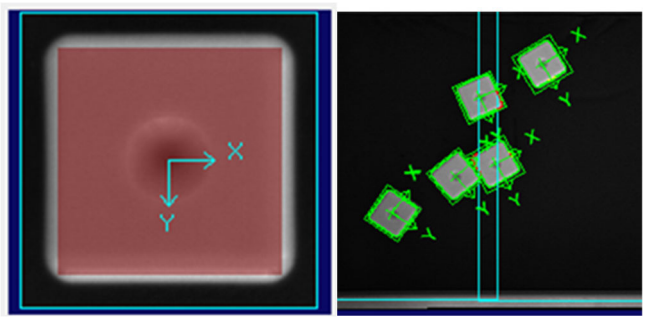

Fig 9 Positioning with PMAlign's Pattern

b) Blob Analysis

Blob analysis is an image processing technique utilized for detecting and measuring "blobs" (or connected regions) within an image. Composed of a series of processing operations and analytic functions, it facilitates the extraction of information from any 2D shape present in the image (Pandit and Rangole 2014).

The application of a Blob analysis algorithm typically commences with the calculation of adjustable lower and upper thresholds, which are then employed for segmenting objects from the background. This process often encompasses the computation of histogram information to identify suitable thresholds. The thresholding process is of utmost importance as it ensures appropriate distinction between regions of interest (blobs) and the background.

Following the thresholding process, the Floodfill method is typically used to populate the object areas, marking or separating regions within the image. The Floodfill method is a seed-based region filling algorithm that starts from a seed point and spreads outward, marking all pixels connected to this point as the same region. In situations where there are distinct, separable blobs in the image post-thresholding, this method proves particularly useful. The fundamental idea of Floodfill is to start from a seed pixel and progressively fill the connected region of this pixel with the same color or label (Nisha and Varshney 2017).

Here are the basic steps of the Floodfill algorithm:

1. Select a seed point: Firstly, select a point in the image as the seed point. This seed point is usually a point within the area where you wish to start filling.
2. Set the target and replacement colors: Determine the target color you wish to fill (usually the color of the seed point) and the new color you want to fill with.
3. Check the seed point: Examine if the color of the seed point is the target color. If it's not, end the algorithm as there is no area needing filling. If it is, change the color of the seed point to the replacement color.

4. Inspect the neighbors: Next, look at the four or eight neighborhoods of the seed point (different neighborhoods can be chosen based on the specific application) and apply the previous steps to each neighbor. In other words, for each neighbor, you will check if its color is the target color. If it is, change its color to the replacement color and consider this neighbor as the new seed point, applying the Floodfill algorithm recursively.
5. Recursive filling: By recursively applying this process, you can fill the connected region of the seed point. This process will continue until no more pixels can be filled.

After using the Floodfill algorithm to fill the target areas, the Blob analysis algorithm is used to detect and measure blobs. Blobs are regions composed of neighboring pixels with the same logical state (such as foreground or background). The Blob analysis algorithm can extract information about the blob's size, shape, location, etc. It assigns labels to each individual blob and calculates attributes such as area, centroid, bounding box, etc., providing important insights into the nature and location of the detected objects.

For this project, Blob analysis has the following advantages compared to the PatMax algorithm:

1. Multi-target detection: Blob analysis can simultaneously detect multiple targets in the image, while PatMax is typically used for the recognition and location of a single target. This makes Blob analysis more advantageous in scenarios requiring detection of multiple objects in an image. Therefore, for irregular shapes that differ from each other, the Blob analysis algorithm is more applicable for target detection.
2. Computational efficiency: For processing large volumes of data, Blob analysis is usually faster than PatMax. This is because the computational complexity of Blob analysis typically increases linearly with the number of pixels, whereas the computational complexity of PatMax may increase exponentially as the search area enlarges.
3. Shape insensitivity: Blob analysis does not depend on the specific shape of the target; as long as there is connectivity between pixels, it can recognize and measure. This enables it to handle objects of various shapes. Conversely, PatMax requires a pre-defined pattern, which may limit its ability to handle different shapes.
4. measuring capabilities: Blob analysis can provide many measurements about the detected objects, such as area, perimeter, centroid, orientation, etc. These metrics are helpful for understanding the properties and location of objects, and can be used for matching with the shapes in the template. PatMax, on the other hand, primarily provides information about the object's location and orientation.

5. Flexible processing capability: Blob analysis can adjust methods like threshold processing and blob segmentation to adapt to various application scenarios. On the other hand, PatMax tends to rely on clear, high-contrast images, and might not handle low-quality or complex images ideally. Furthermore, in situations with complex backgrounds and lighting that cannot be handled by threshold segmentation, semantic segmentation models like "Segment Anything" can be used to preprocess the image, transforming it into a binary image containing only the foreground and background, after which Blob analysis can be performed.

Therefore, this project employs the Blob tool in the VisionPro software package to implement multi-target detection and develops a secondary script in C# for the tool to output irregular shape information such as centroid and geometric moments for subsequent shape-matching tasks. Below are the steps for target detection using VisionPro's Blob tool:

1. Load the image: The output image from the camera calibration tool block in section 3.1 is used as the input for the blob tool block.
2. Set the threshold: To distinguish between the target and the background, you need to set one or more thresholds. The Blob tool will determine whether each pixel belongs to the target object based on whether the pixel value is within the threshold range. Typically, you can determine an appropriate threshold by observing the grayscale histogram of the image.
3. Execute Blob analysis: Run the Blob tool for analysis. The Blob tool will identify all the connected regions (Blobs) in the image that meet the threshold conditions, and calculate a series of attributes for each region, such as area, centroid location, bounding box, etc.
4. Filter and sort: Set a numerical value to filter blobs based on the connected area, eliminating noise caused by lighting or the background. In this case, you can set a filter condition so that the Blob tool only reports the Blobs that meet the conditions. Additionally, you can specify a sorting rule so that the reported Blobs are sorted according to attributes such as area, location, etc.
5. Check the results: Finally, check the output results of the Blob tool to confirm whether the detected objects meet expectations. If the results are unsatisfactory, you can return to the previous steps, adjust parameters, and run the Blob tool again.

Upon configuring the Blob tool, it can be incorporated into the program. The Blob tool facilitates the acquisition of the x and y coordinates of each pixel within the Blob in the image. By averaging these coordinates, we can obtain the centroid coordinates of the irregular object. Within the script, one can utilize the command 'blobResults.GetBoundary()' to garner the boundary information of the irregular object. The acquired boundary corresponds to a CogPolygon object. By utilizing

'CogPolygon.CenterOfMassX' and 'CogPolygon.CenterOfMassY', we can ascertain the x and y coordinates of the irregular shape's centroid.

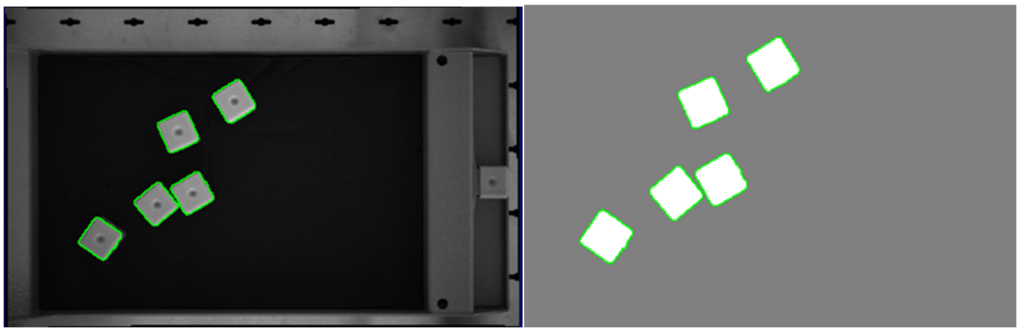

Fig 10 Extraction of Geometric Features of an Object using the Blob Tool

3.3.2 Template Matching

Following the acquisition of geometric information for each polygon in the image through blob analysis, it is necessary to utilize this information to perform matching with the polygons within the template. Consequently, the translation and rotation required for the polygons to move to the target position specified by the template can be computed, which guides the robotic arm for grasping and placement tasks. For the task of template matching, two algorithms are employed. The first one is based on polygon fitting and properties of the polygon including vertex angles and side lengths. The second algorithm operates on the principles of Hu Moment Invariants, as introduced in Section 2.5.

a) Polygon Matching Algorithm Basedon Edge Lengths and Angles
   Irregular shapes can be abstracted into individual polygons, with the fundamental characteristics of a polygon being its vertices, edge lengths, and interior angles. It is widely recognized that given the interior angles at each vertex and the lengths of each side, a polygon can be uniquely determined. Therefore, theoretically, an algorithm can be designed to calculate the interior angles at each vertex and the corresponding side lengths using the coordinates of the polygon vertices. Then, by comparing these with the polygons in the template, it is possible to compute the translation and rotation necessary to move the polygons to the target positions specified in the template.

   This algorithm can be broken down into three general steps:

   1. Use the set of all pixel points that constitute the polygon boundary provided by blob analysis to fit the polygon and obtain the coordinates of the polygon vertices.

2. Compare the vertex information of the polygon with the vertex information of each polygon provided by the template, and obtain a one-to-one correspondence of vertices for matching polygons.
3. Calculate the translation and rotation of the polygons in the image to the target positions using the corresponding vertices.

For the first task, the cv2.approxPolyDP() method from OpenCV can be used to extract the vertex coordinates from the large set of points describing the edges of the polygon. cv2.approxPolyDP() is a function in OpenCV that is used to approximate contours. It takes two parameters: the contour itself, and the approximation accuracy. The latter represents the ratio of the approximation accuracy to the original contour's perimeter. This value typically lies between 1-5% (Poda and Qirici 2018). The function returns a set of points approximating the polygon, which can be used in applications for shape detection and classification. Furthermore, this algorithm is based on the assumption that curves can be approximated by a series of short line segments, and the resulting approximate curve is composed of a subset of points defined by the original curve. This operator is developed based on the Ramer-Douglas-Peucker algorithm, a method widely used in computer graphics and computational geometry to reduce the number of points in a curve approximated by a series of points (Douglas and Peucker 1973). This is known as curve simplification or line simplification. The aim of the algorithm is to find a similar curve with fewer points. The algorithm cannot guarantee that the simplified curve is the best approximation, but it provides a trade-off between simplicity and proximity to the original curve.

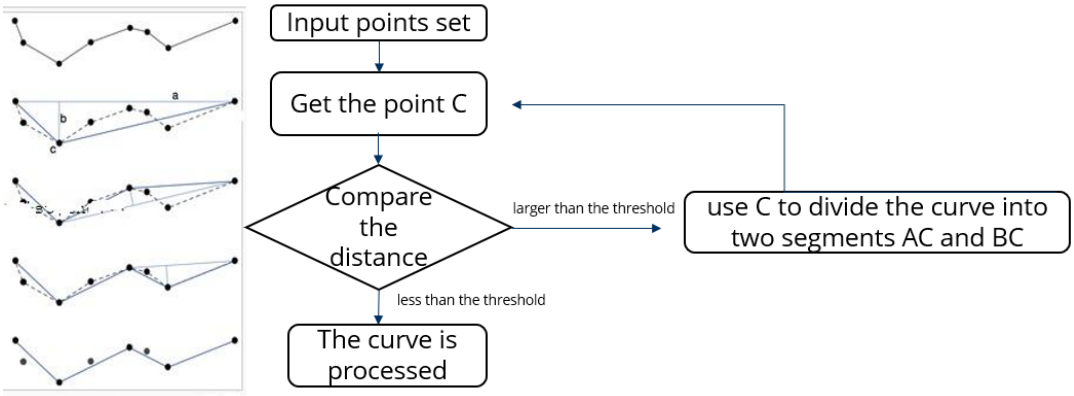

Fig 11 Ramer-Douglas-Peucker algorithm flow chart

The detailed steps of the Ramer-Douglas-Peucker algorithm are as follows:

1. Begin with a curve: The algorithm takes a curve composed of continuous points connected by line segments and a tolerance value ε as input.

2. Mark the endpoints: Mark the first and last points of the curve as "important" and consider all other points as "unimportant".
3. Find the point furthest from the line segment: For each line segment drawn between every pair of "important" points, consider all "unimportant" points between them. Find the point that is the furthest from the line segment.
4. Check the relationship between the point furthest from the line segment and the tolerance: If the distance from the point furthest from the line segment to the line segment is greater than ε, mark this point as "important".
5. Repeat this process: Repeat steps 3 and 4 for the line segments generated by the points newly marked as "important".
6. End when no more points can be marked: Repeat this process until no additional points can be marked as "important".

The result of the algorithm is a simplified version of the original curve, containing only the "important" points. This algorithm is particularly useful in Geographic Information Systems (GIS), computer graphics, and data compression, as it helps to reduce the amount of data required to represent a curve without significantly compromising the quality of the representation. Therefore, the edge point set of the polygon obtained by blob analysis can be used as input for cv2.approxPolyDP() to obtain the vertex coordinates of the polygon for subsequent matching.

Upon obtaining the vertex information of the polygons in the pixel image using cv2.approxPolyDP(), it is necessary to perform template matching based on the coordinates of the vertices of the polygons. As the polygons in the template are also given by vertex coordinates, a preliminary selection can be carried out first by comparing the number of vertices. The polygons in the template that have the same number of vertices as the polygons in the camera image are selected in this first round of screening. Subsequently, the lengths of the sides for all polygons are calculated and arranged in ascending order. The ascending sequence of side lengths of the target polygon is compared with the side lengths of the polygons obtained from the first round of screening. If the difference in all side lengths is smaller than a predetermined threshold, it is recognized that the target polygon matches the respective polygon in the template. This process is repeated until all polygons are paired with polygons in the template.

For two paired polygons, it is also necessary to calculate the rotation angle and translation of the target polygon to the position of the corresponding paired polygon in the template. The translation can be determined by the vector formed by the centroids of the two polygons, while the rotation angle needs to be calculated by first determining the corresponding relationship between

the vertices of the two paired polygons and then calculating the angle between a pair of side lengths.

However, this method has certain limitations. The number of fitted polygon vertices obtained when using the cv2.approxPolyDP() function to fit the polygon may vary from the original polygon due to the influence of lighting or background, which can affect the matching of the polygon with the template. Therefore, there is a need for a matching method that does not rely on polygon side lengths and vertices. As a result, the matching algorithm based on Hu moments is introduced.

b) Polygon Matching Algorithm Based on Hu Moment Invariant

Hu moments are seven moment invariants generated from the raw moments of an image, also known as Hu Moment Invariants. They were first proposed by Ming-Kuei Hu in his 1962 paper "Visual Pattern Recognition by Moment Invariants" (Hu 1962). These seven invariants remain constant with respect to image translation, scaling, and rotation, thereby possessing significant application value in the field of computer vision and image analysis. For instance, Hu moments are widely employed in image processing, with the most common applications being object recognition and image matching. In the context of object recognition, Hu moments can be used to identify specific objects within an image. For image matching, Hu moments are used to compare the similarity between two images (Yang et al. 2013). Hence, Hu Moment Invariants can represent the geometric information of a polygon, and then be utilized for matching the polygon with a template.

According to eq.3-10, the polygon matching steps based on the Hu moment invariant are as follows:

1. Obtain the geometric moments of the objects: Using blob analysis, the raw moments (m_{pq}) are calculated. Blob analysis helps us to identify distinct regions in an image that are separated by a background.
2. Obtain the geometric moments of the templates: Creating a list of polygons with the verticies read form the template file and calculate the geometric moments of them.
3. Compute the centroid of the objects and templates: The centroid (x', y') is calculated using the raw moments. Typically, this would be done using the equations: $x' = m_{10}/m_{00}$ and $y' = m_{01}/m_{00}$, where m_{00} is the zeroth moment (essentially, the total mass of the blob), m_{10} is the sum of the x coordinates of all pixels, and m_{01} is the sum of the y coordinates.
4. Compute the central moments of the objects and templates: The central moments (U_{pq}) are computed using the raw moments and the centroid of the image. The central moments are translation invariant, i.e., they don't change if the image is moved around in space.

5. Compute scale invariant moments of the objects and templates: These are also known as normalized central moments (η_{pq}), and are calculated from the central moments. The scale invariant moments don't change if the object in the image is scaled up or down.
6. Compute the seven Hu moments of the objects and templates: Finally, the seven Hu moments are computed from the scale invariant moments with Eqs.(3-10). The Hu moments are invariant to translation, scale, and rotation.
7. Select a polygon in the image and calculate the Euclidean distance between its Hu moment and the Hu moments of polygons in the template. The one with the smallest distance is considered as the matching polygon in the template.
8. Repeat step 7 until all polygons are matched.

This methodology does not necessitate the approximation of the polygon, thereby eliminating matching errors caused by inaccurate fitting. Furthermore, according to the logic of this approach, it is only required to identify the polygon in the template that has the shortest Euclidean distance to the Hu moments of the target polygon.

Within the software environment of VisionPro, I developed a tool block called "ReadTemplate" which was specifically designed to read CSV files to provide templates.

This tool block uses the CSV file path as its input and reads the UTF-8 encoded CSV file located at the specified path. It iterates through each line of the CSV file, with each line representing a polygon. The polygon coordinates are read in an "x-y-x-y..." sequence, storing vertex coordinates in two separate lists that respectively hold the x and y coordinates of the vertices in an ordered fashion. After completing the read for each line, a new CogPolygon object is created using the lists containing the vertex coordinates.
The CogPolygon, a construct within the VisionPro software environment, represents polygons and describes planar figures constituted by a closed chain or polygonal loop formed by a finite number of line segments. This process is repeated until all lines in the CSV file have been read. Ultimately, these CogPolygon objects are stored in the 'TemplatePolygons' list for output.

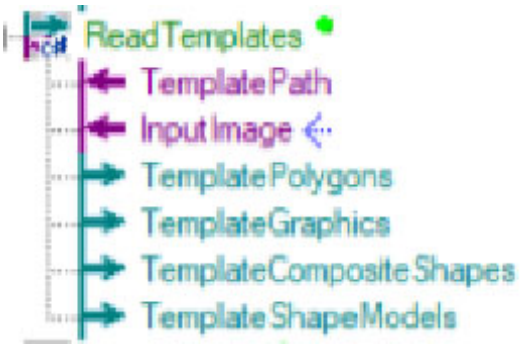

Fig 12 ReadTemplate.vpp

Upon completing the template reading, it becomes necessary to extract graphical information from the images collected by the camera. By employing the method 'blobResults.GetBoundary()', we sequentially acquire the edges of each irregular shape in the image, with the edges presented in the form of CogPolygon objects. Subsequently, the function 'ShapeMatcher(List<Cogpolygon>, Cogpolygon)' is invoked, taking as input the list of CogPolygons output from 'ReadTemplate.vpp' and the edge of an individual blob. Through the method of Hu Moment Invariants Matching, it identifies the polygon from the list that is most similar to the edge shape of the blob and outputs its index in the list.

The centroid coordinates and primary axis angle of both this blob edge shape and the polygon in the template that is most similar to it are then calculated using 'CogPolygon.CenterOfMassX', 'CogPolygon.CenterOfMassY', and 'CogPolygon.Angle' respectively.

The 'ShapeMatcher(List<Cogpolygon>, Cogpolygon)' function iterates through the input list of CogPolygons. In each iteration, it calculates geometric moments using the 'CogPolygon.AreaMoments2' method, then computes the central moments of the polygon. Thereafter, it calculates the normalized central moments based on these central moments. Following this, it calculates Hu moments using these normalized central moments, and then computes the Euclidean distance between the Hu moments of the two polygons.

If this Euclidean distance happens to be the current minimum value, it temporarily saves the polygon from the list as the template polygon that most closely resembles the shape of the blob's edge. After completing the iteration through the list, it thus retrieves the polygon from the list that most closely approximates the shape of the blob's edge.

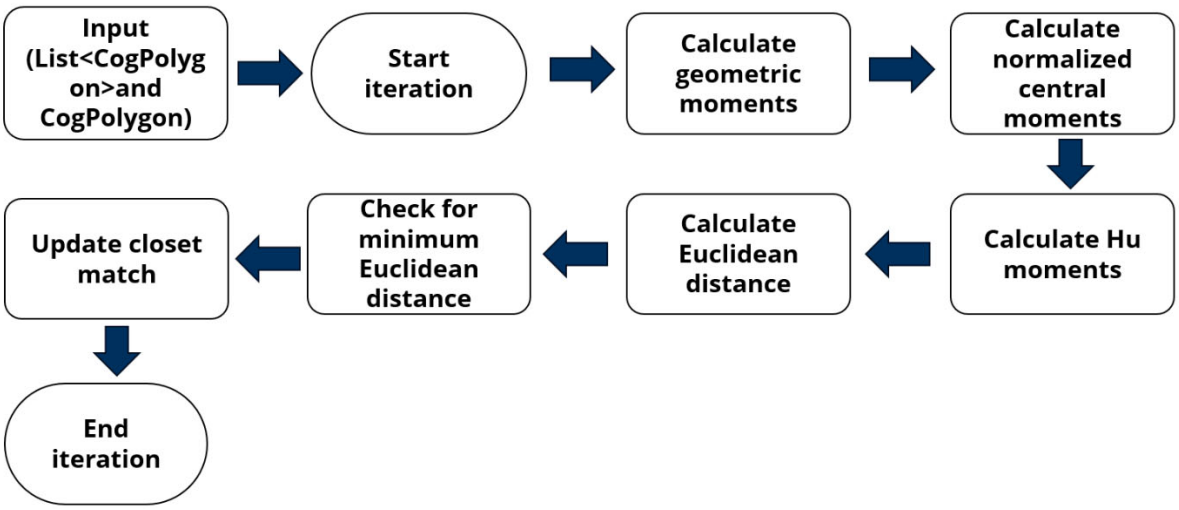

Fig 13 Flowchart of Template Matching Based on Hu Moment Invariants

Following the acquisition of the corresponding centroid coordinates and primary axis angles, these data must be transformed into 'VisionTechResult2D' objects. In order to achieve this goal, a 'CogTransform2DLinear' object is initially created. This object typifies a 2D geometric transformation and encapsulates affine transformations that can be applied to points, regions, and other geometric objects within the VisionPro environment.

Subsequently, the previously obtained centroid coordinates and primary axis direction are respectively assigned to the 'TranslationX', 'TranslationY', and 'Rotation' properties of this 'CogTransform2DLinear' object. This assignment enables the robotic arm to utilize these data to move to the corresponding position.

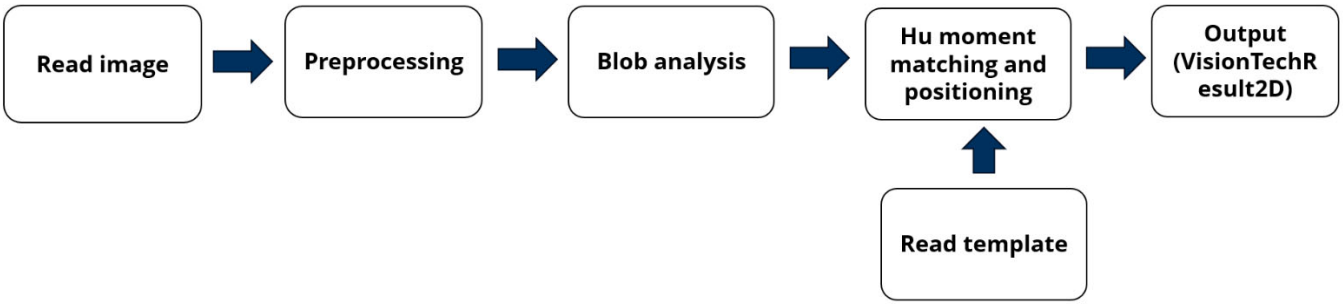

Fig 14 Flowchart of the AutoSlabAlign.vpp Toolblock

The calibration tool, blob tool, and the template reading tool are combined, and then the script is used to implement the template matching of the results obtained by the blob tool through the above process, and output VisionTechResult2D, which is the final tool block AutoSlabAlign.vpp. Install this tool block on the Smartpad of the KUKA robot arm to guide the robot arm visually to recognize and assemble irregular objects.

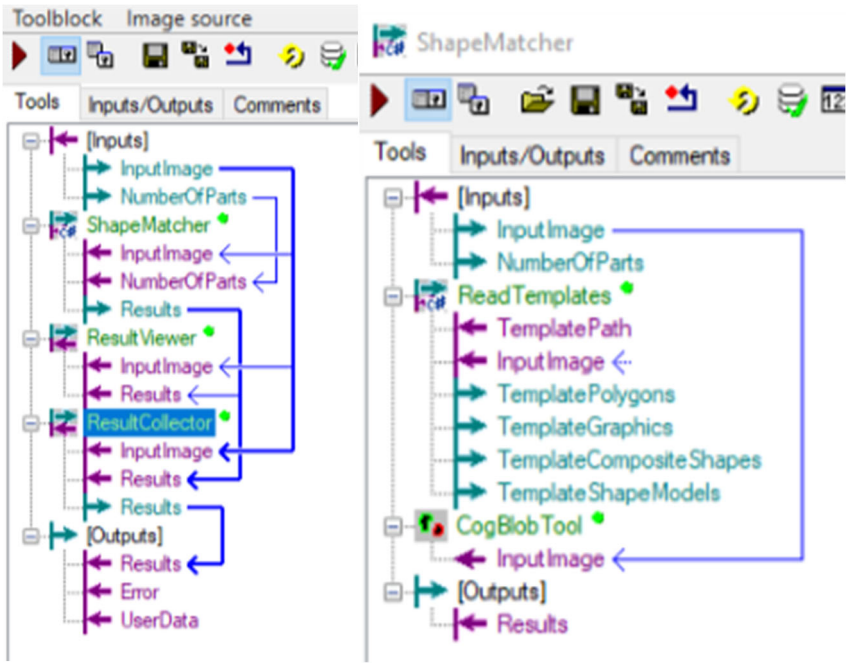

Fig 15 Architecture of the AutoSlabAlign.vpp Tool Block

Based on the background study, the project aims to monitor the window construction progress of a building at the university, i.e., Beyer-Bau as presented in Fig. 4, using YOLO as the object detection algorithm. By employing YOLO for this task, high accuracy and real-time performance can be achieved, which is critical for effective monitoring of the window building progress. In the subsequent sections, a detailed description of the specific targets, methodology, outcomes, and evaluations of the project will be provided.

4 Validation of Workflow Rationality

In this chapter, we will provide an illustrative example of a program which is capable of recognizing and grasping irregular objects in a two-dimensional environment and subsequently arranging them according to a predefined template. This use case is grounded on the development within the Cognex

VisionPro platform, representing a significant application of machine vision in practical production scenarios. Through in-depth understanding and implementation of this specific application, we not only experience the robust capabilities of the VisionPro platform in the realm of machine vision but also gain profound insights into the challenges and solutions pertaining to the grasping and assembly of irregularly shaped objects. Crucially, through this hands-on application, we observe the concrete manifestation of theoretical principles and algorithms in real-world scenarios, enabling us to further comprehend their value, significance, and to validate the effectiveness of our code.

This chapter is structured into two key sections: simulation and experimentation. In the simulation section, we will initially introduce the simulated environment—KUKA's SimPro, followed by a detailed explanation of the robotic arm model employed. Subsequently, we will elaborate on the process of constructing models of irregular objects and positioning the camera within the simulated environment. Furthermore, we will discuss the placement of the calibration board within the simulation and the subsequent camera calibration process. Lastly, we will detail the simulation of the robotic arm's motion program and present the resultant outcomes.

In the experimental section, we will firstly delineate the experimental environment—KUKA Educate Ready2, and provide a comprehensive explanation of the camera calibration process implemented therein. Following this, we will describe how to install the tool blocks developed by us, and detail the programming process for the vision-guided motion program of the robotic arm, which will utilize KUKA's Robotic Language (KRL). Finally, we will showcase the results of the experiment.

4.1 Design of workflow

According to our previous discussion, to implement the identification and grabbing of irregular objects in two dimensions, the workflow can be designed as follows:

a) Image preprocessing: First, preprocess the captured image. The goal of preprocessing is to clean up the image, reduce noise, enhance the object edges, simplify data, and finally distinguish the foreground and background. You can use the Sagment Anything Model to preprocess images with complex lighting and backgrounds, segment the foreground and background to get a mask image, then pass the mask image to visionpro for further processing. If the background and lighting are controllable, such as using uniform lighting and a pure black background, threshold segmentation can be used for image

preprocessing.

b) Hand-eye calibration and camera calibration: If the relationship between the camera coordinate system and the robot arm base coordinate system is known, perform the robot arm hand-eye calibration first, then use checkerboard calibration to calibrate the camera to get the conversion relationship between pixel coordinates and actual coordinates, and correct the camera distortion. The corrected image serves as the input for subsequent steps.

c) Blob analysis: Next, use the Blob tool to analyze the preprocessed image to get the geometric characteristics of the blob, such as edge centroid and major axis. Calculate the geometric moment of the irregular object based on these data as the input for the subsequent Hu Moment Invariant matching step.

d) Template reading: The tool block reads the vertex information of the template polygon from the csv file and calculates the Hu moment of the irregular object as the input for the subsequent Hu Moment Invariant matching step.

e) Template matching: Match the template based on the analyzed geometric characteristics to determine the placement of the object. Use the Blob tool to calculate the Hu Moment Invariant of each irregular shape, which consists of seven components. Then take an irregular shape and calculate the Euclidean distance between its Hu moment and all template polygons. The template polygon with the smallest distance is the placement for this irregular shape. Repeat these steps until all irregular shapes have been matched.

f) Output of VisionTechResult2D object: After the matching is complete, calculate the centroid and major axis of each irregular shape and its matched template polygon in turn as the coordinates and direction for the robot arm to grab and place. After the calculation is complete, use these coordinates and directions to construct the VisionTechResult2D object, which is used to guide the movement and grabbing of the KUKA robot arm.

g) Vision-guided robot arm: The robot arm grabs and places according to the obtained coordinates and directions, using point-to-point, linear and other movement methods to assemble irregular objects according to the template.

This workflow automates various steps of vision guidance, including image preprocessing, camera calibration, Blob analysis, template matching, etc., reducing the complexity of operation. In addition, each step is encapsulated into an independent module, and each module can be independently updated and upgraded, making it easy to maintain and upgrade. The subsequent chapters will validate the rationality of this workflow through simulation and experiment.

4.2 Simulation

In the upcoming section on simulation, we will take a deep dive into the simulation environment utilized for this study, namely, the KUKA SimPro software. The selection of this particular software will be elaborated in detail, showcasing its superior compatibility with the KUKA series of robots, in comparison to other simulation software options. We will present a comprehensive understanding of the software's features, focusing on the process of image acquisition including the location and resolution of the images. We will also delve into how camera calibration is conducted within this software, translating pixel coordinates into actual spatial coordinates. Additionally, we will elucidate the programming commands used in KUKA SimPro such as Point-To-Point (PTP) and Linear (LIN) movements and how the robotic arm's motion program is developed. The section will culminate in the presentation of our simulation results.

4.2.1 Simulation Environment

In the realm of robotics simulation software, KUKA SimPro emerges as a robust, versatile, and user-friendly platform. It serves as an integral part of this study owing to its unique attributes that align well with the demands of our research. KUKA SimPro, developed by KUKA, a renowned manufacturer of industrial robots and solutions for factory automation, is a specialized software that facilitates the simulation and offline programming of KUKA robot systems. This enables us to virtually design and verify the robotic system and processes before deploying it in the real world.

A primary advantage of using KUKA SimPro lies in its excellent compatibility with the KUKA series of robots. It supports a wide range of KUKA robots, making it an ideal platform for our study. This exceptional adaptability significantly simplifies the integration of various robot models into the simulation environment, thereby improving the overall efficiency of the design process.

The true strength of KUKA SimPro is its ability to generate real robot programs directly from the simulation. Once the robot's movements and tasks are simulated and verified in the software, the program can be exported and run on the actual robot controller without the need for any further modifications. This feature not only accelerates the development cycle but also minimizes the risk of errors during the transition from the simulation to the actual execution.

Another noteworthy feature of KUKA SimPro is the ease of use. Its intuitive interface allows even novice users to quickly grasp the basic operations and start developing simulations. The software also includes advanced features for more

experienced users, offering a well-rounded tool for developing sophisticated robot simulations.

Through its blend of compatibility, functionality, and user-friendliness, KUKA SimPro provides us with a practical and efficient platform for developing and testing our robot programming. The ability to design, simulate, and validate robotic applications within a single software environment streamlines our research process, contributing to the thoroughness and accuracy of our results. Based on the aforementioned discussions, this project will be carried out in KUKA simpro 3.1.2.

4.2.2 Models

In the simulation of this project, the model used is KUKA Ready2EducateKR3. This model includes a KR 3 R540 robotic arm, along with a corresponding workstation and background panel. The assembly location is consistent with the equipment used in the experiments. The robotic arm's gripper has been replaced with a suction cup, as shown in Fig 12.

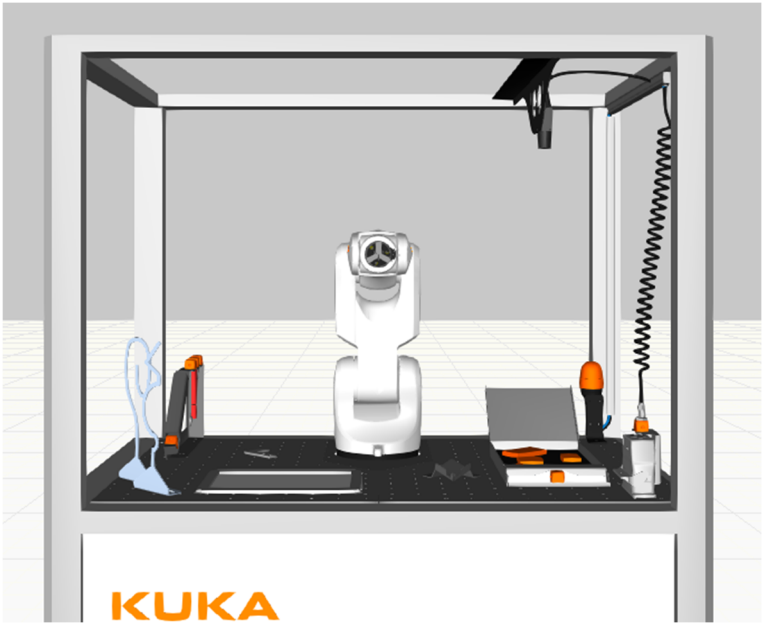

Fig 16 Kuka Educate Ready2 Model（KUKA sim pro）

For the irregular objects, each is modeled as a rectangle with distinct lengths of sides. This is done to better set and demonstrate the major axis direction of the polygons within the software. The model contains five irregular objects, which are placed on a pure black background. The dimensions of these irregular objects are as shown in the table below:

Model number	Length(mm)	Width(mm)	Height(mm)
1	88	50	10
2	50	38	10
3	50	50	10
4	88	25	10

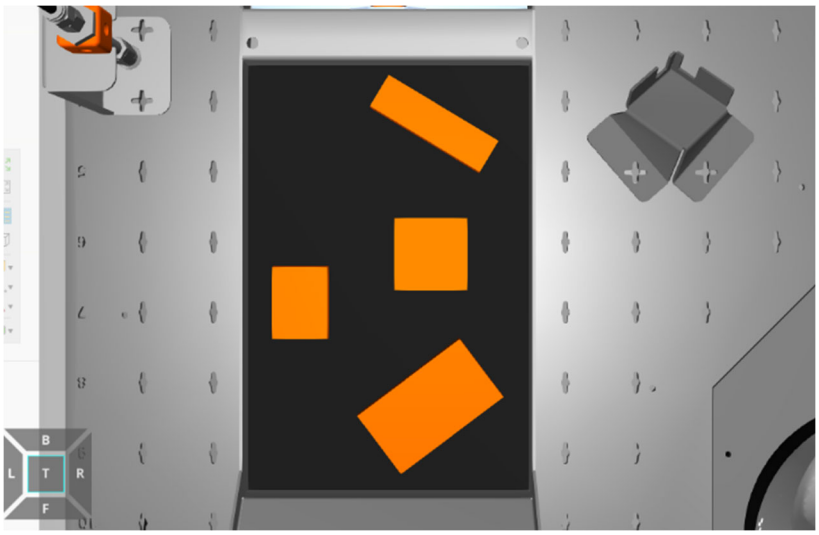

Fig 17 Irregular shapes used in Simulation (KUKA sim pro)

These irregular objects can be assembled into a rectangle measuring 163mm88mm10mm, as shown in the figure.

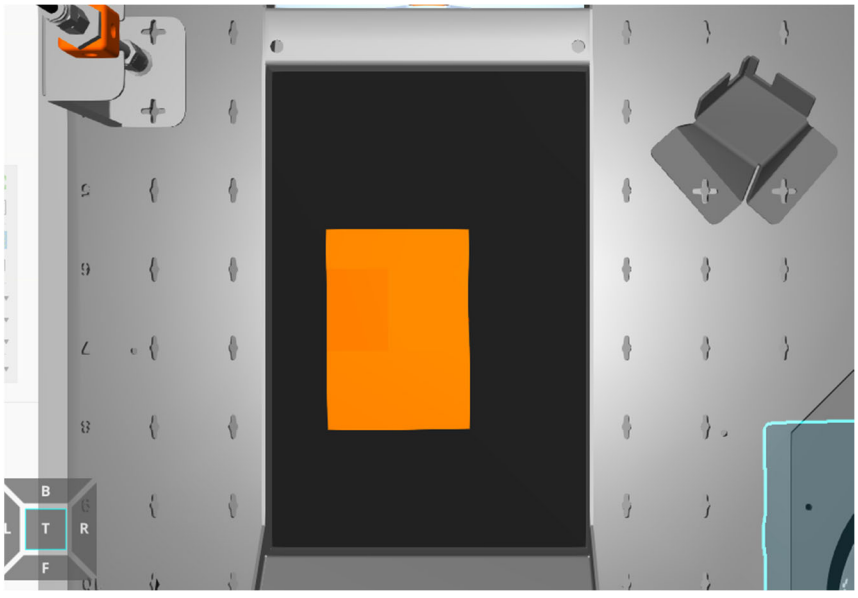

Fig 18 Irregular shapes used in Simulation after Assembling (KUKA sim pro)

4.2.3 Calibration

Following the placement of the irregular objects, we need to calibrate the simulated camera. This process is identical to real-world camera calibration. The camera is positioned directly above the background panel, and every image capture is a top view of the panel with a resolution of 1116*632.

After obtaining the top view images via the simulated camera, we employ a calibration plate for calibration, converting pixel coordinates into real-world coordinates. This step uses the Calibration tool block provided by VisionPro, which allows the use of a chessboard calibration plate to calibrate the camera. This tool block plays a key role in transforming images based on pixel coordinates into actual dimensions. Once the camera is correctly calibrated, the tool block can parse input from the pixel image and produce the corresponding real-world dimensions. The calibration process is not only crucial for accurate object recognition and measurement but also helps to minimize distortion and provide a clear and actual view of the objects to be recognized or assembled.

The calibration plate used in this simulation is shown in Figure 1. The origin of the pixel coordinates and the directions of the x and y axes are shown in Fig15, and the origin and directions of the x and y axes after calibration are shown in Figure 16.

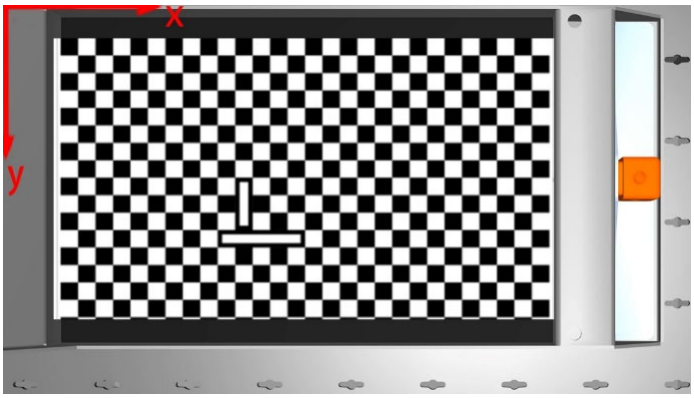

Fig 19

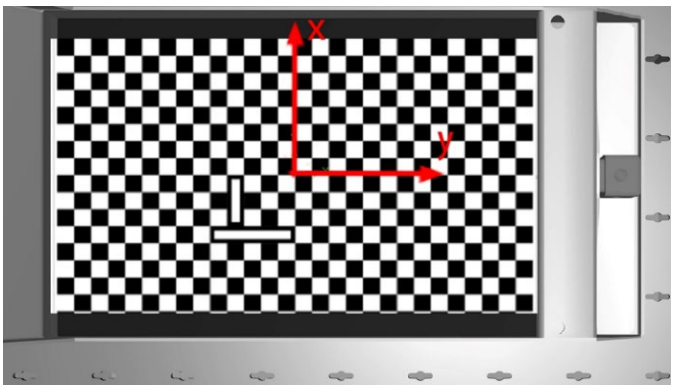

Fig 20

4.2.4 Signal simulation

In the KUKA Sim Pro simulation environment, we can simulate I/O signals by creating and configuring "Signal" objects. This mode of simulation allows us to simulate and test interactions between the robot arm and peripheral devices without actual hardware, providing theoretical support for applications in real environments.

KUKA Sim Pro offers a series of predefined signal types, including digital input signals, digital output signals, analog input signals, and analog output signals, etc. These signal types can meet various application scenarios. For example, when executing some logical judgments or device switch controls, we can choose to use digital input and output signals. When we need to simulate sensor or actuator values, we can opt for analog input and output signals.

In this project, our primary focus is on the simulation of digital signals. This is because we need to control the suction cup's gripping and releasing via signals. The specific operation process is as follows: when the robot arm moves to the pickup or placement location, we use the '$out' command to output a digital signal. This signal will be sent to the suction cup's controller to control the action of the suction cup based on the signal's state. When the signal is 'true', the suction cup is triggered to grip the object; when the signal is 'false', the suction cup is triggered to release the object.

In this way, we can simulate and control the suction cup's action by controlling the state of the digital signal, achieving the gripping and placing of irregular objects in the simulation environment.

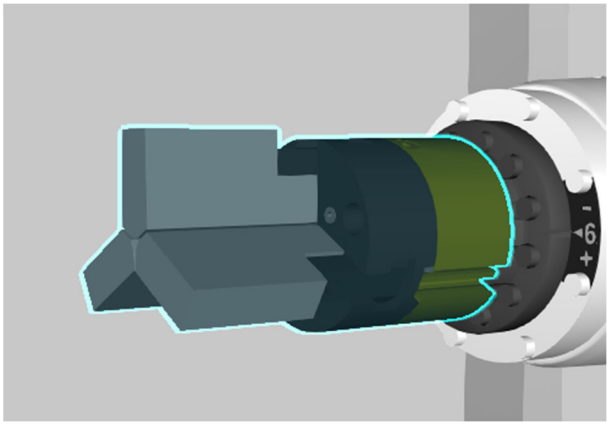

Fig 21 The gripper used in the simulation (KUKA sim pro)

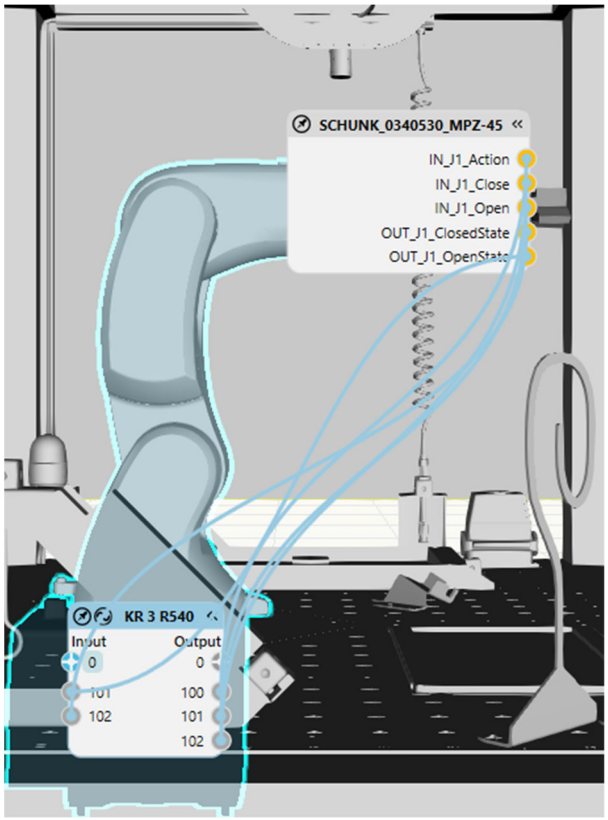

Fig 22 Signal connection between the robotic arm and the gripper (KUKA sim pro)

4.2.5 Application of the Custom Tool Block

After adding the Calibration tool block into the custom tool block, it is necessary to place the template file containing the vertex coordinates of each irregular object after assembly in the specified path of the tool block. Subsequently, images can be imported to start processing images collected in the simulation software, obtaining the centroid coordinates and main axis angles of each irregular object before and after assembly. The results after processing by the custom tool block 'AutoSlabAlign' are shown in Fig 17, where the edges and centroids of the irregular objects are marked in green. Fig 17 shows the result of the program execution, which is output in KUKA Vision2D format. Here, X, Y, and A represent the position and angle of the origin of the tool coordinates for a particular pick or place pose

of the robot arm, respectively. The data is output in pairs in sequence, representing the pick and place positions respectively.

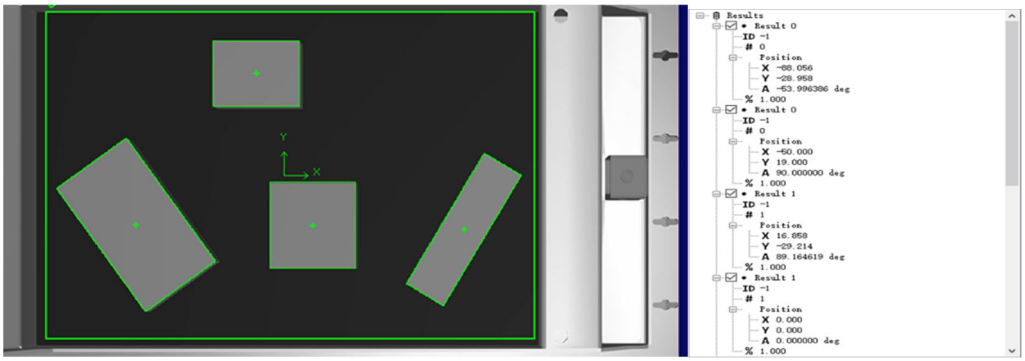

Fig 23 The results after processing by the custom tool block AutoSlabAlign

After obtaining the grasping and placing posture of the manipulator, the tool block can save the posture information into a csv file, and then read these data into KUKA simpro through the KUKAsimproAPI plug-in and convert it into PTP, LIN of the manipulator Wait for the movement command.

KUKA's SimPro uses KRL (KUKA Robot Language), a proprietary, high-level language developed specifically for programming KUKA robots. Two critical commands used frequently in KRL are PTP (Point-to-Point) and LIN (Linear). PTP - Point to Point movement: The PTP command moves the robot arm from one point to another in space. The robot will take the shortest path to the destination, but the path taken between the two points is not defined, and the orientation of the end-effector can also change during the movement. This kind of movement is usually used when the path between two points is unobstructed and the precise path taken by the robot arm is not important. LIN - Linear movement: The LIN command instructs the robot to move in a straight line from its current position to a specified point. Unlike the PTP command, LIN ensures a linear path and a constant orientation of the end-effector during the entire motion. This command is typically used when there is a need for a precise path, such as in painting or welding applications. In KUKA SimPro, these commands can be used to program complex robot arm movements, and are often critical to the success of the task. By simulating these movements beforehand, the user can ensure safe and efficient operations.

The custom plugin reads the grabbing and placing pose information from a CSV file and then adds a PTP command at a certain height directly above this position. Subsequently, the arm moves to this position using a LIN command before executing the grab or placement. After importing the file, the motion commands

for the robotic arm are shown in Fig 18, and Fig 19 shows the process of assembly executed by the robotic arm based on the commands.

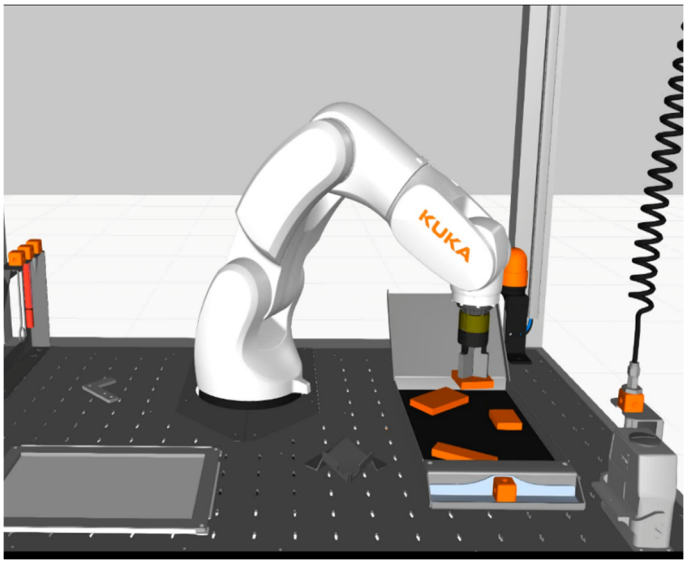

Fig 24 The Process of Assembly Executed by the Robotic Arm Based on the Commands

It can be seen from the simulation that the robotic arm can better recognize the irregular graphics in the image and automatically complete the assembly according to the template. Fig 20 shows the effect of the assembled robotic arm in KUKAsimpro.

Fig 25 The Result of Assembly Executed by the Robotic Arm Based on the Commands

4.3 Experiment

In the forthcoming section on experimental application, we will delve into the application of our developed algorithm on a real-world robotic system, the KUKA educate ready2. This robotic system, designed for educational purposes, provides an authentic experience of industrial-grade automation processes, and it is crucial for our study to test our developed solutions in such a realistic environment.

This chapter will commence by providing a comprehensive understanding of the KUKA educate ready2 system. We will explore its various components, functionalities, and their significance in the context of our application. Further, we will elucidate on how our developed solution is integrated into the teach pendant of the system. This step is crucial as it transforms our theoretical and simulated concepts into a tangible operation, underlining the practical relevance of our study.

Next, we will illustrate the calibration process in a real-world setup, which might encompass certain challenges not encountered in the simulated environment. Calibration is an essential step to ensure the accuracy and reliability of our vision-guided robotic operation. We will provide an in-depth understanding of this process, focusing on its methodologies, techniques, and relevance.

Following calibration, we will expound on the programming of the vision-guided robotic arm movement. We will elaborate on the various instructions, commands required to successfully implement this motion control.

4.3.1 Experiment environment

The experiments for this project were conducted on the KUKA Educate Ready2 (as shown in Fig 21), a compact, industrial-grade robotic system designed specifically for educational and research applications. Its objective is to provide users with hands-on experience of real-world automation and robotics technology, bridging the gap between theoretical knowledge and practical application. It consists of a compact, lightweight yet powerful KR 3 R540 robotic arm (as shown in Fig 22) and a mobile teaching table equipped with the robot arm and relevant control hardware. The KR 3 R540, a compact six-axis robot, is particularly suitable for small working spaces, with a maximum working radius of 541mm and a maximum payload capacity of 3kg. Its precision of approximately ±0.02mm makes it an excellent tool for tasks that require high precision. The KUKA Educate Ready2 is powered by the KUKA KR C4 compact controller, a comprehensive controller solution tailored for its bundled KR 3 R540 robotic arm. Researchers control the robotic arm via the KUKA smartPAD-2 (as shown in Fig 23).

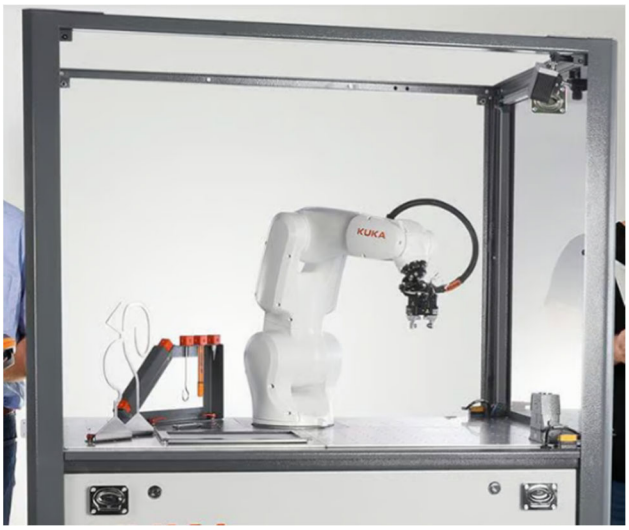

Fig 26 KUKA Educate Ready2 (https://www.kuka.com/)

Fig 27 KR 3 R540 robotic arm (https://www.kuka.com/)

For our specific project, we have opted for the VCXG-25M industrial camera due to the following justifications (Baumer_VCXG-25M_EN_20221110_DS):

a) High resolution and frame rate: The VCXG-25M provides a high resolution of 1920*1200 pixels and a frame rate of 40 frames per second, enabling us to capture clear, detailed images and process them in real-time.

b) Stability and reliability: The VCXG-25M is specifically designed for industrial applications and can operate reliably under diverse, harsh environmental conditions typically found at construction sites.

c) Advanced features: This camera model offers advanced features like hardware triggering and image preprocessing, enhancing control over the image capture process and improving image processing efficiency.

d) Appropriate interface: The VCXG-25M utilizes the GigE Vision interface, which ensures high-speed, stable data transmission, and offers advanced features such as synchronization and triggering.

e) Flexibility and adaptability: Importantly, the applications developed for the VCXG-25M, thanks to its standardized GigE Vision interface, can be readily adapted for use with other industrial-grade cameras having the same interface. This flexibility and compatibility make it a more versatile choice, adding another layer of justification for selecting this specific model.

Fig 28 VCXG-25M industrial camera

Due to the fact that the gripper supplied with the KUKA Educate Ready2 cannot suction and grab irregular objects like the suction cup used in the simulation, in this experiment, the gripper is only used to indicate the pick-up and placement positions of irregular objects. The irregular objects are replaced with cardboard of the same rectangular size with different lengths and widths as used in the simulation (as shown in Figure 24), solely to verify the feasibility of the methods and codes proposed in this project.

Fig 29 Smart pad

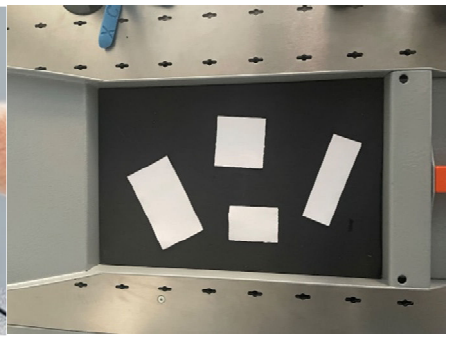

Fig 30 Irregular shapes used in experiment

4.3.2 Calibration and Tool block setup

Upon introducing the experimental equipment and tools, this section will delve into the calibration of the camera used in the experiment, as well as the installation and debugging of the tool block.

Since the calibration of the camera in KUKA Educate Ready2 is performed separately from the tool block, the first step in the experiment involves calibrating the camera to derive the transformation relationship between pixel coordinates and actual coordinates. This is then followed by the installation and debugging of the tool block. The calibration process for the camera is the same as in the simulation, with the camera being fixed directly above the black background board. The calibration board used is a checkerboard calibration board with dimensions of 240mm*180mm*6mm (as shown in Fig 25). Once the calibration

board is properly positioned, calibration can be performed using the SmartPAD. The transformation relationship between the pixel coordinates and actual coordinates is saved in a custom file after calibration.

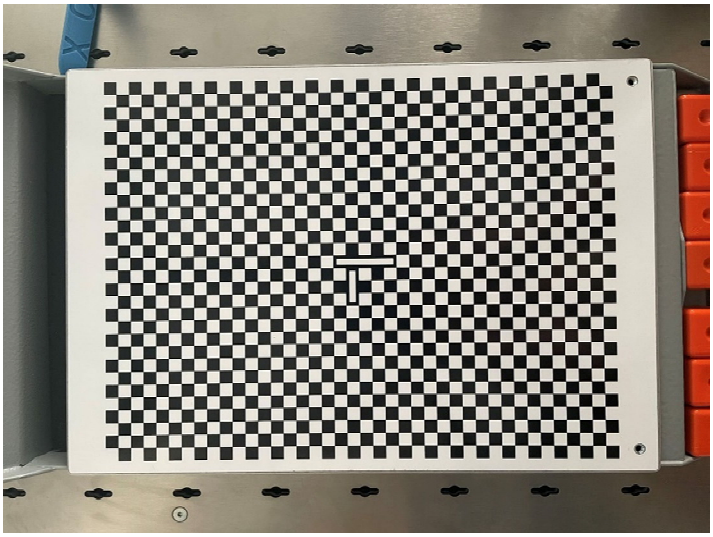

Fig 31 Calibration Board used in Experiment

The tool block developed for this project is based on VisionPro. As long as the KUKA visiontech plug-in is installed on the SmartPAD, the tool block can be easily installed by using the AutoSlabAlign.vpp file in SmartPAD. Once installed, it needs to be run once to adjust parameters such as the camera's exposure and focus, and an image should be collected for blob analysis area (as shown in Fig 26). The analysis area should be within the black background board area to avoid influences from other background factors.

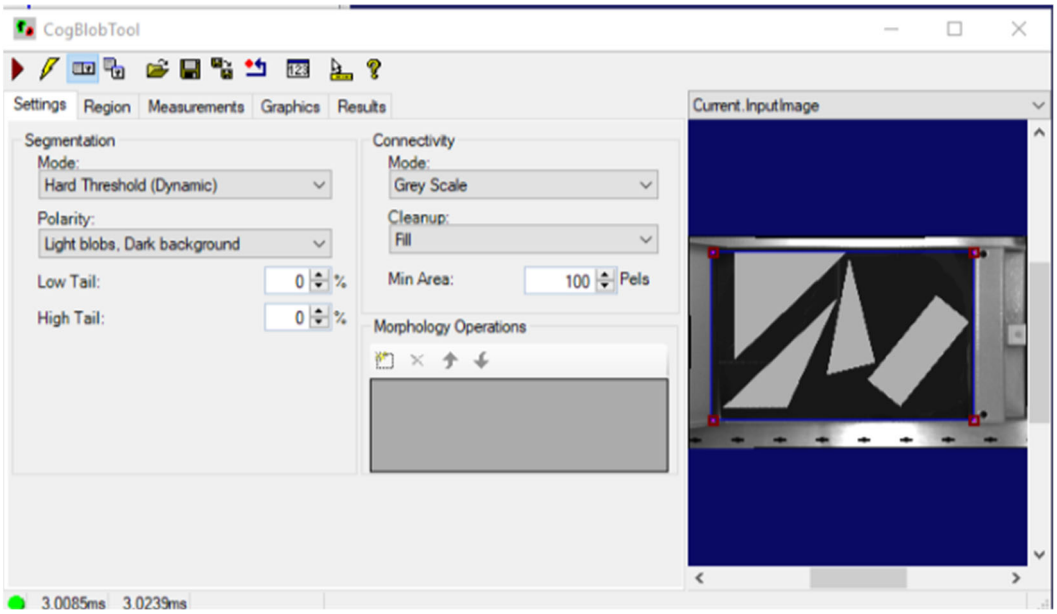

Fig 32 Graphical User Interface of the Blob Tool

Once the tool block has been adjusted, it can be copied onto the SmartPAD. The tool block installation can then be conducted at the "Task Configuration" section in VisionTech, with the previously obtained calibration file selected during the installation process.

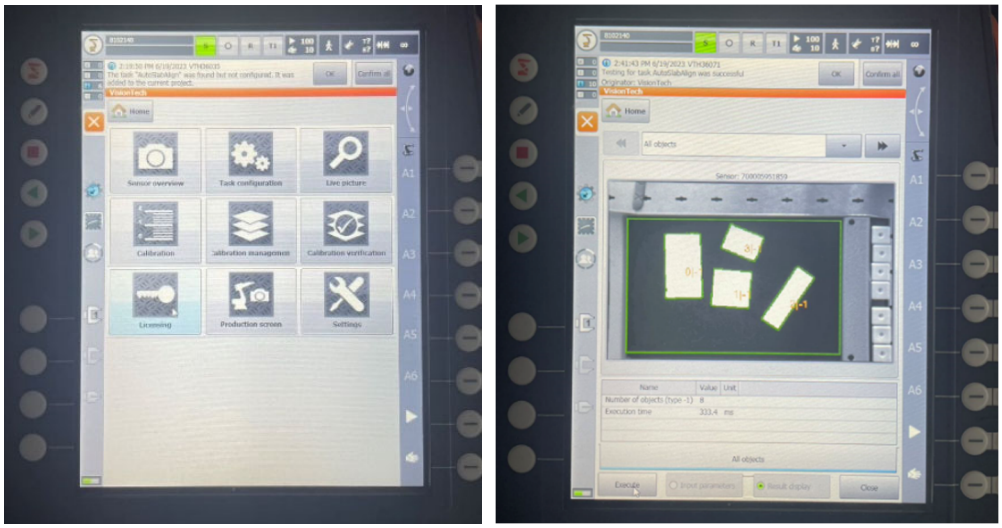

Fig 33 Running results of AutoSlabAlign.vpp on Smartpad

4.3.3 Vision-guide KUKA robotic arm motion programming

Upon establishing the appropriate settings for the camera and tool block, the stage is set for the creation of the KUKA robotic arm's motion program. The programming language employed by the KUKA robotic arm is the proprietary KUKA Robot Language (KRL) developed by KUKA. KRL is a procedural programming language that embodies a fundamental structure composed of the declaration segment (variable definition), the data segment (data block declaration), and the task segment (inclusive of program codes). It also incorporates control structures for loops and conditional judgments.

In the context of this project, the KRL command 'VT.TRIGGER' is invoked to trigger a visual task (vt). Subsequently, the 'VT.WAITFORRESULT' command is called upon to wait for the visual system to return the results of recognition and localization. Finally, the 'VT.LOOPRESULTS' command is implemented to iterate through all the results returned from the visual task, and to independently carry out the grasping and placement for each irregular object. This sequence of command utilization ensures that each object is appropriately handled and manipulated as intended,

thereby driving the success of the automation process. In addition, the 'SPTP' and 'LIN' commands are also called to control the robot arm tool coordinates to move to the target point with the fastest path and the straight path respectively.

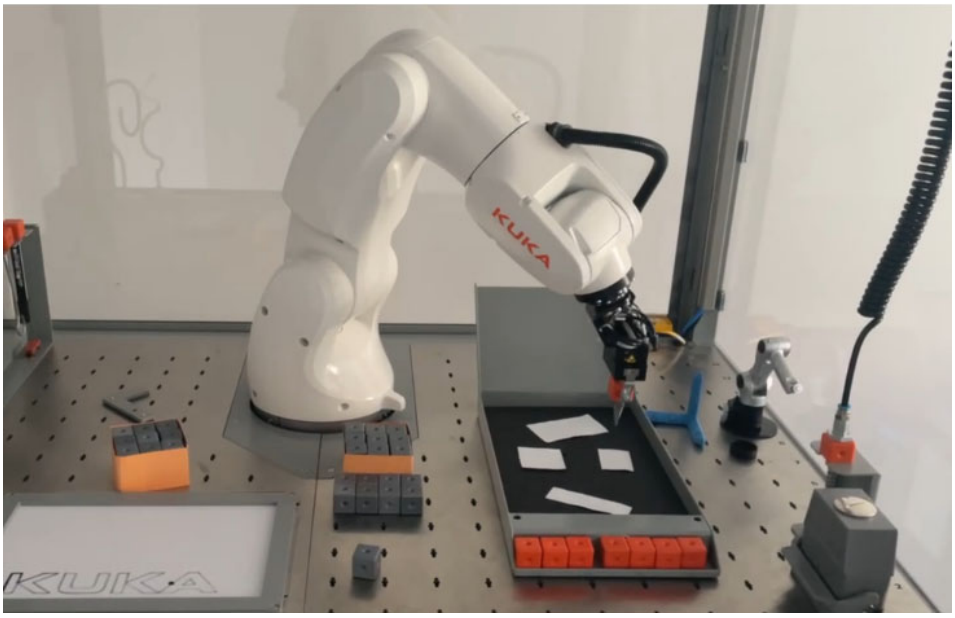

Fig 34 The robotic arm, guided by the vision program, moves to the irregular object's placement location.

5 Conclusion

In this project, we have pioneered an automated workflow that enables a robotic arm to autonomously recognize and assemble irregular shapes according to a provided template. At the heart of this achievement lies the creation and implementation of a tool block named "AutoSlabAlign". Built around the Hu Moment Invariants shape matching algorithm, this tool is powerful in effectively processing and analyzing image data of irregular shapes, and extracting key geometric characteristic information therein. The tool uses collected PNG-format images and CSV files (which contain location data of template polygons) as input, and outputs the centroid coordinates and principal axis directions of these irregular shapes, as well as the centroid coordinates and principal axis directions of the corresponding shapes in the template.

These pieces of information are then transformed into 'VisionTechResult2D' objects, and a 2D geometric transformation is implemented via a 'CogTransform2DLinear' object, achieving precise control over the robotic arm. The algorithms used include binary threshold selection, morphological operations

(such as dilation, erosion, opening, and closing), and the Hu Moment Invariants matching method.

The project encountered challenges in binary image processing due to factors such as illumination and background. The introduction of the "Segment Anything Model" significantly mitigated these challenges, ensuring the acquisition of stable binary images under various environments and imaging conditions.

To validate the effects of the designed workflow and model, we conducted simulations with KUKA simpro and empirical tests on the KUKA educate ready platform. The results showed high feasibility of our method, highlighting the potential of this work in revolutionizing automated operations in the industry dealing with irregular shape components.

However, we acknowledge there is still room for improvement. Despite the advancements, there are certain limitations to the current workflow, which are subsequently discussed.

5.1 Limitation

While our workflow demonstrates excellent performance in handling 2D scenarios, it does have some constraints. First and foremost, the current model is designed for two-dimensional images. For problems involving depth information or operations in a three-dimensional environment, our workflow might require significant adjustments or additional modules to handle these extra dimensions. When dealing with the assembly of irregular shapes in three-dimensional space, the factors to consider become more complex, including but not limited to object's location, shape, rotation angle, lighting conditions, and background environment.

Secondly, the existing workflow relies primarily on the robot arm moving directly to the target location using ptp (point-to-point) or lin (linear) commands. This method does not take into account any possible obstacles along the path. If obstacles are present, the robot arm may collide, causing equipment damage or operation errors. This is especially important in real environments, as workplaces often undergo dynamic changes, and the appearance of obstacles is often unpredictable.

Moreover, the project assumes that the irregular shapes being handled are consistent in thickness, and the height at which the robot arm grasps needs to be pre-set, or a suction cup gripper with flexibility is required for grasping. This

assumption may cause problems when dealing with objects of uneven thickness or complex shapes. To address this issue, new grippers might need to be designed, or more sophisticated algorithms might need to be employed to estimate the grasp posture.

To overcome these issues, future work might include developing tools and algorithms suitable for three-dimensional image processing, introducing more complex path planning algorithms to avoid obstacles along the path, and designing new grippers or developing more advanced grasp posture estimation algorithms. This might involve hardware upgrades (such as introducing 3D sensing devices like depth cameras or radar), as well as the corresponding software development and debugging work. This would broaden the application range of our workflow, but it would also bring about higher development and implementation costs.

In addition to the aforementioned limitations, there are some unique constraints to consider with the "Segment Anything Model" we use. This model requires a preliminary prompt to help identify the object to be segmented. In our project, we set the number of objects to be segmented in advance, and then extract the highest scoring results from the global segmentation mode of the "Segment Anything Model". However, this method might lead to inaccuracies, as the highest-scoring segmentation is not always the most suitable.

To address this issue, we might consider implementing advanced object detection models such as YOLO to provide more accurate prompts for the "Segment Anything Model", achieving more precise image segmentation. For instance, we could first use an object detection model to identify the target objects in the image, then set the prompt for the "Segment Anything Model" based on the position and shape information of these objects to achieve more accurate segmentation.

Moreover, such advanced object detection models could also achieve real-time target detection and tracking, enabling our workflow to adapt to more complex and dynamic environments. However, this requires substantial computational resources and meticulous model training and fine-tuning, which is a consideration for future research.

Consequently, future work may also involve: 1) the development and integration of advanced object detection models to improve the accuracy and flexibility of image segmentation; 2) optimizing the "Segment Anything Model" to enhance its segmentation performance under various conditions; 3) a deep optimization and integration of the entire workflow to adapt to a broader and more complex range of applications.

5.2 Application prospect

Automated Material Handling: On construction sites, materials that require handling often come in different, irregular shapes, such as stones, bricks, and precast components. Utilizing the technology developed in this project to automatically identify and handle irregularly shaped objects can automate the collection, classification, and placement of these materials, significantly improving the efficiency and accuracy of material handling.

Automated Assembly: During the assembly process of precast components, it is necessary to accurately install components of various shapes and sizes at specific locations. By using the tools we have developed, automatic identification, positioning, and assembly of components can be achieved, thus greatly reducing the complexity and error rate of manual operations, and enhancing the efficiency and quality of assembly.

Construction Robots: With the technology developed in this project, we can design and develop construction robots capable of automatically recognizing and handling irregularly shaped objects. These robots can autonomously perform various construction tasks in complex construction environments, such as material transportation, component assembly, construction site cleanup, etc. This significantly improves the level of construction automation, alleviates the intensity of manual labor, and enhances the efficiency and quality of construction.

References

Publication bibliography

Amit, Yali; Felzenszwalb, Pedro; Girshick, Ross (2020): Object Detection. In : Computer Vision: A Reference Guide. Cham: Springer International Publishing, pp. 1–9.

Arulprakash, Enoch; Aruldoss, Martin (2022): A study on generic object detection with emphasis on future research directions. In *Journal of King Saud University - Computer and Information Sciences* 34 (9), pp. 7347–7365. DOI: 10.1016/j.jksuci.2021.08.001.

Baumer_VCXG-25M_EN_20221110_DS.

Bock, Thomas (2007): Construction robotics. In *Auton Robot* 22 (3), pp. 201–209. DOI: 10.1007/s10514-006-9008-5.

Bradski, Gary; Kaehler, Adrian (2008): Learning OpenCV: Computer vision with the OpenCV library: " O'Reilly Media, Inc.".

Chang, Y. W., & Lin, C. J. (2008, December): Feature ranking using linear SVM. In Causation and prediction challenge (pp. 53-64). PMLR.

Connolly, Christine (2003): Latest developments in machine vision – a review of image processing packages. In *Sensor Review* 23 (3), pp. 193–201. DOI: 10.1108/02602280310481805.

Dai, J., Li, Y., He, K., & Sun, J. (2016): R-fcn: Object detection via region-based fully convolutional networks. Advances in neural information processing systems, 29.

Dalal, N.; Triggs, B. (2005): Histograms of oriented gradients for human detection. In : 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, 886-893 vol. 1.

Dickerson, N. L. (2017): Refining bounding-box regression for object localization (Doctoral dissertation, Portland State University).

Douglas, David H.; Peucker, Thomas K. (1973): Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In *Cartographica: the international journal for geographic information and geovisualization* 10 (2), pp. 112–122.

Enebuse, Ikenna; Ibrahim, Babul K. S. M. Kader; Foo, Mathias; Matharu, Ranveer S.; Ahmed, Hafiz (2022): Accuracy evaluation of hand-eye calibration techniques for vision-guided robots. In *PloS one* 17 (10), e0273261. DOI: 10.1371/journal.pone.0273261.

Evgeniou, Theodoros; Pontil, Massimiliano (2001): Support Vector Machines: Theory and Applications. In G. Goos, J. Hartmanis, J. van Leeuwen, Georgios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos (Eds.): Machine Learning and Its Applications, vol. 2049. Berlin,

Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 249–257.

Felzenszwalb, Pedro; McAllester, David; Ramanan, Deva (2008): A discriminatively trained, multiscale, deformable part model. In : 2008 IEEE conference on computer vision and pattern recognition. IEEE, pp. 1–8.

Fonseca, Leila M. G.; Manjunath, B. S. (1996): Registration techniques for multisensor remotely sensed imagery. In *PE & RS- Photogrammetric Engineering & Remote Sensing* 62 (9), pp. 1049–1056.

Furet, Benoit; Poullain, Philippe; Garnier, Sébastien (2019): 3D printing for construction based on a complex wall of polymer-foam and concrete. In *Additive Manufacturing* 28, pp. 58–64. DOI: 10.1016/j.addma.2019.04.002.

Gharbia, Marwan; Chang-Richards, Alice; Lu, Yuqian; Zhong, Ray Y.; Li, Heng (2020): Robotic technologies for on-site building construction: A systematic review. In *Journal of Building Engineering* 32, p. 101584. DOI: 10.1016/j.jobe.2020.101584.

Girshick, R. (2015): Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

Girshick, Ross; Donahue, Jeff; Darrell, Trevor; Malik, Jitendra (2014): Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In : 2014 IEEE Conference on Computer Vision and Pattern Recognition. 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Columbus, OH, USA, 2014/6/23 - 2014/6/28: IEEE, pp. 580–587.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014): Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2015): Region-based convolutional networks for accurate object detection and segmentation. IEEE transactions on pattern analysis and machine intelligence, 38(1), 142-158.

Guo, Yanming; Liu, Yu; Georgiou, Theodoros; Lew, Michael S. (2018): A review of semantic segmentation using deep neural networks. In *Int J Multimed Info Retr* 7 (2), pp. 87–93. DOI: 10.1007/s13735-017-0141-z.

H. Chen (1991): A screw motion approach to uniqueness analysis of head-eye geometry. In : 2013 IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos, CA, USA: IEEE Computer Society, 145,146,147,148,149,150,151. Available online at https://doi.ieeecomputersociety.org/10.1109/CVPR.1991.139677.

Hashemi, Nazanin Sadat; Aghdam, Roya Babaie; Ghiasi, Atieh Sadat Bayat; Fatemi, Parastoo (2016): Template Matching Advances and Applications in Image Analysis. Available online at http://arxiv.org/pdf/1610.07231v1.

He, K., Zhang, X., Ren, S., & Sun, J. (2015): Spatial pyramid pooling in deep

convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.

Heikkila, Janne; Silvén, Olli: A four-step camera calibration procedure with implicit image correction. In : Proceedings of IEEE computer society conference on computer vision and pattern recognition: IEEE, pp. 1106–1112.

Herbert Bay; Andreas Ess; Tinne Tuytelaars; Luc Van Gool (2008): Speeded-Up Robust Features (SURF). In *Computer Vision and Image Understanding* 110 (3), pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014.

Hsiao, Edward; Felzenszwalb, P.; McAllester, D.; Ramanan, D. (2009): A discriminatively trained, multiscale, deformable part model.

Hu, Ming-Kuei (1962): Visual pattern recognition by moment invariants. In *IEEE Trans. Inform. Theory* 8 (2), pp. 179–187. DOI: 10.1109/TIT.1962.1057692.

J. Heikkila; O. Silven (1997): A four-step camera calibration procedure with implicit image correction. In : Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1106–1112.

Jiang, Jianfeng; Luo, Xiao; Luo, Qingsheng; Qiao, Lijun; Li, Minghao (2022): An overview of hand-eye calibration. In *Int J Adv Manuf Technol* 119 (1-2), pp. 77–97. DOI: 10.1007/s00170-021-08233-6.

Kirillov, Alexander; Mintun, Eric; Ravi, Nikhila; Mao, Hanzi; Rolland, Chloe; Gustafson, Laura et al. (2023): Segment Anything. Available online at http://arxiv.org/pdf/2304.02643v1.

Lazebnik, S., Schmid, C., & Ponce, J. (2006, June): Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06) (Vol. 2, pp. 2169-2178). IEEE.

Lee, Seung Yeol; Lee, Kye Young; Lee, Sang Heon; Kim, Jin Woo; Han, Chang Soo (2007): Human-robot cooperation control for installing heavy construction materials. In *Auton Robot* 22 (3), pp. 305–319. DOI: 10.1007/s10514-006-9722-z.

Lee, Seungyeol; Lee, Jungil; Han, Changsoo; Lee, Kyeyoung; Lee, Sangheon (2008): Human robot cooperative control and task planning for a glass ceiling installation robot.

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2017): Light-head r-cnn: In defense of two-stage object detector. arXiv preprint arXiv:1711.07264.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017): Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

Long, Jonathan; Shelhamer, Evan; Darrell, Trevor (2015): Fully convolutional networks for semantic segmentation.

Lowe, David G. (2004): Distinctive Image Features from Scale-Invariant Keypoints. In *Int J Comput Vis* 60 (2), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94.

Moeslund, Thomas B. (2012): Introduction to video and image processing. Building real systems and applications. London, Heidelberg: Springer (Undergraduate topics in computer science).

Moghaddam, B., Biermann, H., & Margaritis, D. (1999, June): Defining image content with multiple regions-of-interest. In Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL'99) (pp. 89-93). IEEE.

Nisha, Nisha; Varshney, Sapna (2017): A Review: Polygon Filling Algorithms Using Inside-Outside Test. In *IJAERS* 4 (2), pp. 73–75. DOI: 10.22161/ijaers.4.2.15.

Pandit, Amruta; Rangole, Jyoti (2014): Literature review on object counting using image processing techniques. In *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 3 (4), pp. 8509–8512.

Parascho, Stefana (2023): Construction Robotics: From Automation to Collaboration. In *Annual Review of Control, Robotics, and Autonomous Systems* 6, pp. 183–204.

Pare, S.; Kumar, A.; Singh, Girish Kumar; Bajaj, V. (2020): Image segmentation using multilevel thresholding: a research review. In *Iranian Journal of Science and Technology, Transactions of Electrical Engineering* 44, pp. 1–29.

Pásztó, Peter; Hubinský, Peter (2010): Application of a Visual System for mobile robot navigation (OpenCV). In *AT&P Journal Plus* 1, pp. 62–64.

Pathak, Deepak; Krahenbuhl, Philipp; Donahue, Jeff; Darrell, Trevor; Efros, Alexei A. (2016): Context encoders: Feature learning by inpainting. In : Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2536–2544.

Poda, Xhensila; Qirici, Olti (2018): Shape Detection and Classification Using OpenCV and Arduino Uno. In *RTA-CSIT* 2280, pp. 128–136.

Remondino, Fabio; Fraser, Clive (2006): Digital camera calibration methods: Considerations and comparisons. With assistance of Hans-Gerd Maas, D. Schneider.

Ren, S., He, K., Girshick, R., & Sun, J. (2015): Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

Ren, X., & Ramanan, D. (2013): Histograms of sparse codes for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3246-3253).

Rottmann, M., Colling, P., Hack, T. P., Chan, R., Hüger, F., Schlicht, P., & Gottschalk, H. (2020, July): Prediction error meta classification in semantic

segmentation: Detection via aggregated dispersion measures of softmax probabilities. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-9). IEEE.

Rublee, Ethan; Rabaud, Vincent; Konolige, Kurt; Bradski, Gary (2011): ORB: An efficient alternative to SIFT or SURF. In : 2011 International Conference on Computer Vision, pp. 2564–2571.

Shen, Dinggang; Wu, Guorong; Suk, Heung-Il (2017): Deep learning in medical image analysis. In *Annual review of biomedical engineering* 19, pp. 221–248.

Shiu, Y. C.; Ahmad, S. (1989): Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB. In *IEEE Transactions on Robotics and Automation* 5 (1), pp. 16–29. DOI: 10.1109/70.88014.

Tsai, R. (1987): A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. In *IEEE J. Robot. Automat.* 3 (4), pp. 323–344. DOI: 10.1109/JRA.1987.1087109.

Tsai, R. Y.; Lenz, R. K. (1989): A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. In *IEEE Transactions on Robotics and Automation* 5 (3), pp. 345–358. DOI: 10.1109/70.34770.

Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; Smeulders, A. W. M. (2013a): Selective Search for Object Recognition. In *Int J Comput Vis* 104 (2), pp. 154–171. DOI: 10.1007/s11263-013-0620-5.

Uijlings, Jasper R. R.; van de Sande, Koen E. A.; Gevers, Theo; Smeulders, Arnold W. M. (2013b): Selective search for object recognition. In *Int J Comput Vis* 104, pp. 154–171.

Van de Sande, K. E., Uijlings, J. R., Gevers, T., & Smeulders, A. W. (2011, November): Segmentation as selective search for object recognition. In 2011 international conference on computer vision (pp. 1879-1886). IEEE.

Wei, Luosi; Jiao, Zongxia (2008): Visual Location System for Placement Machine Based on Machine Vision. In : 2008 Fifth IEEE International Symposium on Embedded Computing. 2008 Fifth IEEE International Symposium on Embedded Computing (SEC). Beijing, China, 2008/10/6 - 2008/10/8: IEEE, pp. 141–146.

Yamashita, Rikiya; Nishio, Mizuho; Do, Richard Kinh Gian; Togashi, Kaori (2018): Convolutional neural networks: an overview and application in radiology. In *Insights into imaging* 9 (4), pp. 611–629. DOI: 10.1007/s13244-018-0639-9.

Yang, Shuai; Premaratne, Prashan; Vial, Peter (2013): Hand gesture recognition: An overview. In : 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology. 2013 5th IEEE International Conference on Broadband Network & Multimedia Technology (IC-BNMT 2013). Guilin, China, 2013/11/17 - 2013/11/19: IEEE, pp. 63–69.

Yuan, Xiaohui; Shi, Jianfang; Gu, Lichuan (2021): A review of deep learning methods for semantic segmentation of remote sensing imagery. In *Expert Systems with Applications* 169, p. 114417.

Yue Pan; Limao Zhang (2021): Roles of artificial intelligence in construction engineering and management: A critical review and future trends. In *Automation in Construction* 122, p. 103517. DOI: 10.1016/j.autcon.2020.103517.

Yurtsever, Ekim; Lambert, Jacob; Carballo, Alexander; Takeda, Kazuya (2020): A survey of autonomous driving: Common practices and emerging technologies. In *IEEE access* 8, pp. 58443–58469.

Zeiler, M. D., & Fergus, R. (2014): Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13 (pp. 818-833). Springer International Publishing.

Zhang, Z. (2000): A flexible new technique for camera calibration. In *IEEE Trans. Pattern Anal. Machine Intell.* 22 (11), pp. 1330–1334. DOI: 10.1109/34.888718.

Zou, Zhengxia; Chen, Keyan; Shi, Zhenwei; Guo, Yuhong; Ye, Jieping (2023): Object Detection in 20 Years: A Survey. In *Proc. IEEE* 111 (3), pp. 257–276. DOI: 10.1109/JPROC.2023.3238524.