REVIEW ARTICLE

A Review of Research and Applications of OpenCV in Robotic Arm Grasping Vision Systems

Rui Du[1,*]

[1] College of Engineering, Yancheng Institute of Technology, Yancheng 224051, China

Abstract

This paper reviews the OpenCV-based vision system for robotic-arm grasping across three core modules—camera calibration, image preprocessing and recognition, and binocular 3D reconstruction—together with lightweight deployment on embedded platforms. Using the calib3d module, monocular, binocular, and hand–eye calibration procedures are described, which correct lens distortion and unify coordinate frames and thereby address grasping-positioning errors. Image processing algorithms, including grayscale transformation, Gaussian and median filtering, Otsu segmentation, and feature matching, are compared for static and dynamic assembly-line scenarios. Binocular stereo matching generates disparity maps for depth recovery and 6-DoF pose estimation, supporting peg-in-hole assembly and obstacle avoidance. For embedded platforms (Raspberry Pi, TQ210, and STM32-based systems), optimization strategies such as library pruning and reduced image resolution or disparity search range are summarized to improve real-time performance. The review indicates that OpenCV-based solutions are open-source, low-cost, and portable, and are suitable for conventional grasping tasks; however, their robustness remains limited under strong illumination and occlusion. Integrating deep learning, multi-sensor fusion, and online hand–eye calibration are suggested to extend their applicability to complex environments.

Keywords: OpenCV, robotic arm, machine vision, camera calibration, lightweight embedded deployment.

1 Introduction

With the rapid development of intelligent sorting, precision assembly, and mobile grasping robots, robot vision has become the core perceptual means for robotic arms to achieve autonomous operation. As an open-source, lightweight computer vision library, OpenCV (Open Source Computer Vision Library) integrates a full set of algorithms, including camera calibration, image filtering, feature extraction, binocular stereo matching, and 3D reconstruction [1]. It runs on general-purpose and embedded Linux platforms such as the Raspberry Pi and industrial PCs, whereas microcontroller boards such as Arduino usually act as the servo or motion controller in the same system, and it has been widely applied in the development of vision systems for various robotic arms [2, 3].

In the development of low-cost monocular grasping equipment, the built-in checkerboard calibration

Submitted: 08 July 2026
Revised: 08 September 2026
Accepted: 15 September 2026
Published: 24 September 2026

Vol. 1, **No.** 3, 2026.
10.62762/TICPS.2026.513798

***Corresponding author:**
✉ Rui Du
1229390784@qq.com

Citation
Du, R. (2026). A Review of Research and Applications of OpenCV in Robotic Arm Grasping Vision Systems. *ICCK Transactions on Intelligent Cyber-Physical Systems*, 1(3), 104–112.

© 2026 ICCK (Institute of Central Computation and Knowledge)

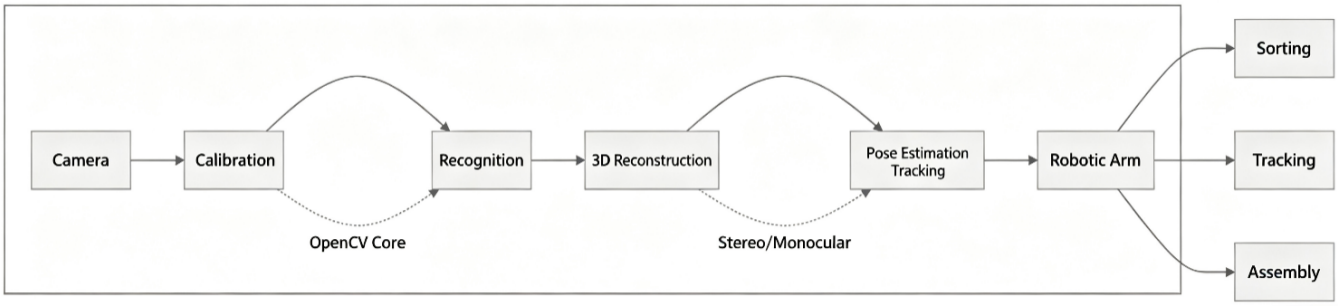

Figure 1. Overall workflow of an OpenCV-based robotic-arm grasping vision system reviewed in this paper, from image acquisition and calibration to recognition, 3D reconstruction, pose estimation, and the grasping applications (sorting, tracking, and assembly).

tools of OpenCV [1, 4] suffice for lens-distortion correction; subsequently, basic image processing operations such as threshold segmentation and contour matching can identify planar workpieces, followed by the transformation from pixel coordinates to the robotic-arm base coordinate frame, thereby establishing a simple vision system for material sorting that meets the operational requirements of desktop small robots [5, 6]. In scenarios involving standard industrial parts, the region-of-interest (ROI) processing and edge-detection functions of OpenCV can stably identify regular threaded components, effectively improving the recognition efficiency of assembly-line grasping [7]. For embedded mobile robots, OpenCV can be pruned and ported by removing redundant modules to reduce computational overhead, enabling real-time image acquisition and target localization on low-power hardware [2, 3]. In binocular vision systems, OpenCV is used for stereo calibration, and the disparity computation module recovers 3D spatial information of objects; on the one hand, this supports trajectory solving for precision peg-in-hole assembly [8], and on the other hand, it can reconstruct environmental point clouds for obstacle avoidance [9]. Furthermore, combining image-based target recognition with 3D point cloud registration has been shown to improve object localization and 6-DoF pose estimation for robotic grasping in cluttered and partially occluded environments [10, 11].

Currently, most studies on robotic-arm vision adopt monocular, binocular, or embedded vision schemes in isolation, lacking a systematic integration of the full OpenCV technical workflow. This paper focuses on robotic-arm grasping scenarios and systematically summarizes the implementation approaches of OpenCV in three major aspects: camera calibration, image preprocessing and recognition, and binocular 3D reconstruction. It also compares the application differences among various vision architectures and compiles lightweight deployment optimization methods, thereby providing a systematic technical summary for the design of OpenCV-based vision systems for robotic arms. The overall workflow reviewed in this paper is shown in Figure 1.

2 OpenCV-Based Camera Calibration Implementation Scheme

Camera calibration is a fundamental prerequisite for vision-based robotic grasping, because the accuracy of transforming pixel coordinates into the 3D spatial coordinate system depends essentially on calibration quality. Numerous existing monocular and binocular mobile grasping devices rely on the calib3d module of OpenCV to accomplish the calibration procedures, which can be categorized into three technical routes: monocular calibration, binocular calibration, and hand–eye calibration [1, 12].

2.1 Monocular Camera Calibration: Zhang's Method and Its OpenCV Implementation

Zhang's checkerboard-based calibration method [4] is the prevailing solution for low-cost machine vision systems. The complete procedure consists of three major steps: checkerboard corner extraction, iterative solving of the intrinsic and extrinsic parameters, and image distortion correction. At the hardware level, monocular grasping systems built on low-power platforms such as the Raspberry Pi (often with an Arduino board as the servo controller) adopt this scheme: the `findChessboardCorners` function coarsely detects the interior corners of the checkerboard, and `cornerSubPix` refines them to sub-pixel accuracy, thereby reducing corner-extraction errors [1].

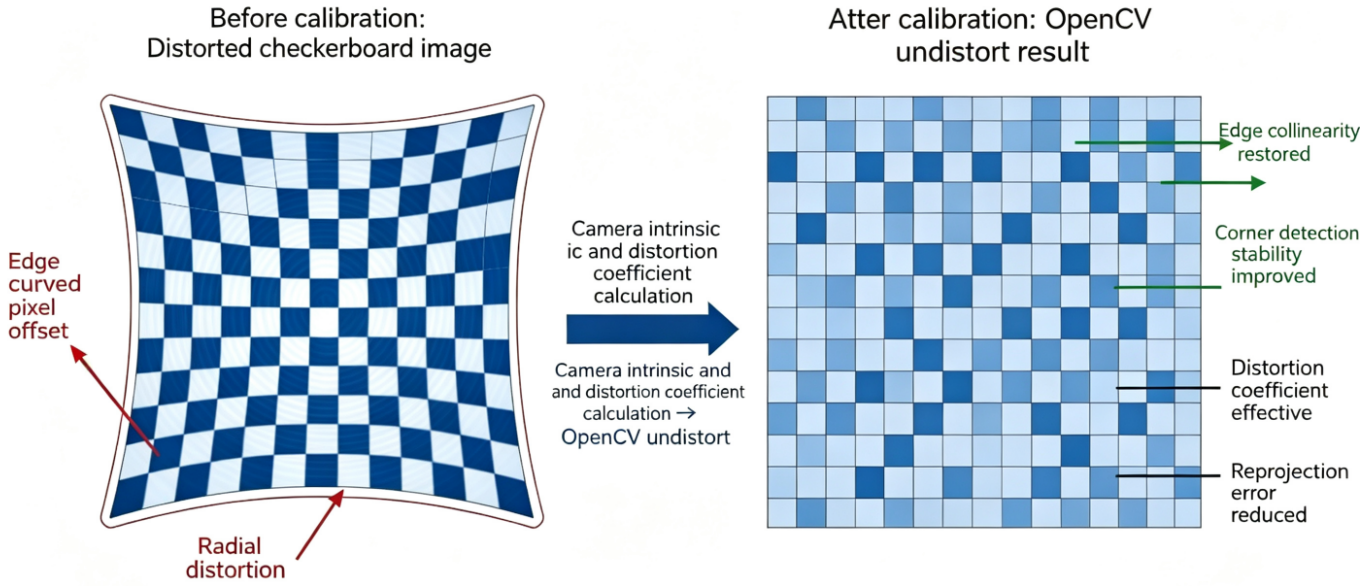

Figure 2. Schematic comparison of a checkerboard image before and after camera calibration and distortion correction with the OpenCV `undistort` function.

After acquiring checkerboard images from multiple viewpoints, the `calibrateCamera` function is invoked to iteratively solve the camera intrinsic matrix and the radial and tangential distortion coefficients, and the distorted images are then corrected with the `undistort` function. Calibration quality is usually reported as the root-mean-square reprojection error, in pixels. Uncalibrated devices typically exhibit noticeable planar positioning deviations caused by lens distortion; calibration removes this systematic error, so that the grasping errors for desktop threaded workpieces and small components can be reduced [4], which makes this approach suitable for simple sorting robotic-arm platforms. Unstructured field grasping systems also adopt monocular calibration as a fundamental preprocessing step, performing feature matching and coarse target localization after image correction. The effectiveness of calibration can be intuitively assessed by comparing images before and after correction; Figure 2 presents a schematic comparison of checkerboard images before and after distortion correction.

2.2 Binocular Stereo Calibration: Implementation Logic in OpenCV

Monocular vision alone cannot directly acquire the depth of the target; scenarios such as peg-in-hole assembly and obstacle avoidance therefore require binocular vision. Binocular calibration based on OpenCV is mainly implemented in two steps. First, independent checkerboard calibration is performed on the left and right cameras to obtain their respective intrinsic parameters and distortion coefficients. Second, the `stereoCalibrate` function is called to solve for the rotation and translation between the two cameras, followed by `stereoRectify` to complete stereo rectification, which aligns the epipolar lines of the left and right images so that the disparity search becomes one-dimensional, significantly reducing the computational overhead [1]. After rectification, the `StereoSGBM` algorithm, an OpenCV implementation of semi-global block matching (SGBM) derived from the semi-global matching (SGM) algorithm of Hirschmüller [13], is used to generate a disparity map. For a rectified image pair, the depth of a pixel follows from

$$Z = \frac{f\,B}{d}, \tag{1}$$

where Z is the depth (m), f is the focal length (pixels), B is the baseline between the two cameras (m), and d is the disparity (pixels). The depth map can be converted into point clouds of the workpieces and the surrounding environment (e.g., with `reprojectImageTo3D`), which is applicable to collision-free path planning for robotic arms and precision peg-in-hole mating [8, 9]. This binocular calibration workflow is widely adopted in embedded Raspberry Pi systems and industrial binocular grasping platforms, achieving a balanced trade-off between computational cost and 3D reconstruction accuracy [14].

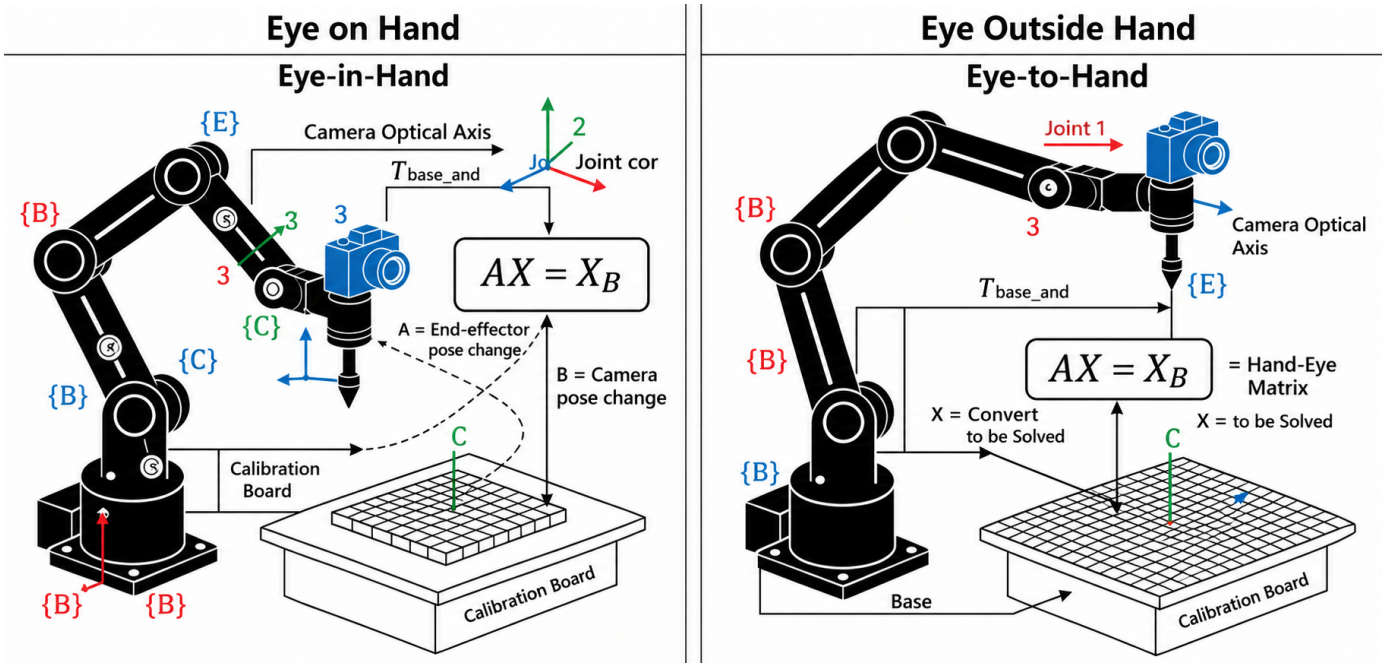

Figure 3. Hand–eye calibration configurations: eye-in-hand (left) and eye-to-hand (right). The unknown hand–eye matrix X is solved from $A_iX = XB_i$.

2.3 Hand–Eye Calibration: Solution Strategy in OpenCV

Efficient robotic-arm grasping requires the coordinated unification of three coordinate frames: the camera, the end-effector, and the robot base. Depending on the camera mounting, two configurations exist, eye-in-hand (camera mounted on the end-effector) and eye-to-hand (camera fixed in the workspace), as shown in Figure 3. In both cases the core objective is to solve the homogeneous transformation equation

$$A_iX = XB_i \tag{2}$$

for the unknown hand–eye matrix X [12]. Let T_e^b denote the end-effector-to-base transform obtained from the joint readings and T_t^c the calibration-target-to-camera transform, both being 4×4 homogeneous matrices, and let i and $i+1$ index two robot poses. For the eye-in-hand configuration, $X = T_c^e$ is the camera-to-end-effector transform, $A_i = (T_{e,i+1}^b)^{-1}T_{e,i}^b$ is the relative motion of the end-effector, and $B_i = T_{t,i+1}^c(T_{t,i}^c)^{-1}$ is the relative motion of the target observed by the camera. For the eye-to-hand configuration, $X = T_c^b$ is the camera-to-base transform and $A_i = T_{e,i+1}^b(T_{e,i}^b)^{-1}$, while B_i is unchanged.

The operating procedure is as follows: the robotic arm moves to several viewpoints from which the checkerboard is observed, and the end-effector pose is recorded at each position (at least three poses whose relative rotation axes are not parallel are required). OpenCV extracts the checkerboard corners from each image and computes the target-to-camera extrinsic parameters (e.g., with `solvePnP`). The multi-pose data are then integrated to solve Eq. (2), for example with the `calibrateHandEye` function of OpenCV, which offers several solvers including the method of Tsai and Lenz [12]. After hand–eye calibration is completed, pixel coordinates within the image can be mapped to the robot base frame, which reduces grasping failures caused by coordinate misalignment. Whether on low-cost servo robotic arms or on field-deployed 6-DoF grasping devices, this establishes a complete coordinate transformation chain for visual grasping [11].

3 OpenCV-Based Traditional Image Processing and Object Recognition

3.1 Basic Workflow of Image Preprocessing

In robotic-arm vision-based grasping systems, OpenCV provides a large number of low-level image processing functions that support workpiece recognition and coordinate extraction. Various monocular and binocular embedded robotic-arm systems rely on OpenCV for preprocessing operations such as image denoising, color-space conversion, and threshold segmentation, which suppress interference from illumination and lens noise and provide high-quality image data for subsequent contour extraction, feature matching, and target

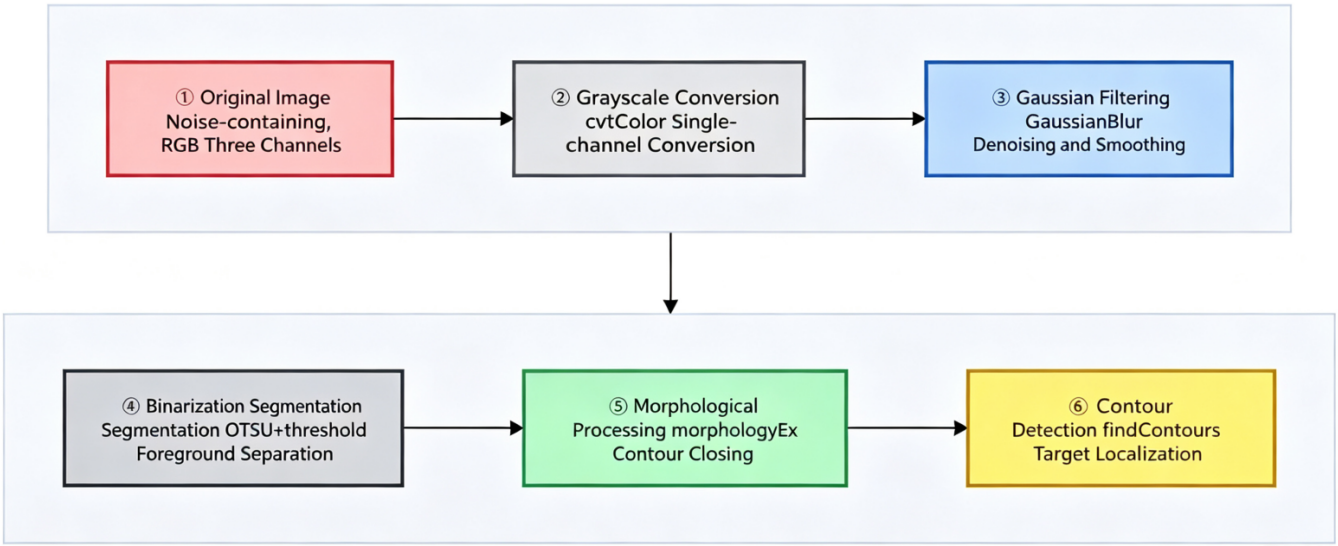

Figure 4. Preprocessing pipeline for workpiece contour extraction: grayscale conversion, Gaussian filtering, Otsu binarization, morphological closing, and contour detection.

localization [6].

Grayscale conversion is the first step of the preprocessing pipeline. Existing studies generally adopt the weighted-average method to convert RGB color images into single-channel grayscale images,

$$Y = 0.299\,R + 0.587\,G + 0.114\,B, \tag{3}$$

where R, G, and B are the red, green, and blue channel intensities and Y is the gray level. This reduces the data volume while preserving the contour details of workpieces to the greatest extent, and it avoids the loss of contrast and detail associated with the simple averaging and maximum-value methods [1].

Filtering is mainly divided into two mainstream schemes: Gaussian filtering and median filtering. Gaussian filtering is a linear low-pass filter that is separable and fast, so it is preferred for dynamic grasping scenarios on industrial assembly lines, where per-frame latency matters; it suppresses Gaussian-distributed sensor noise at the cost of slightly blurring workpiece edges. Median filtering is a nonlinear filter that removes salt-and-pepper (impulse) noise while preserving edges better, and it is therefore preferred where impulse noise dominates, such as static sorting stations, although its per-pixel cost is higher for large kernels [1].

After denoising, Otsu's automatic threshold selection method [15] is employed to separate the foreground workpieces from the background, which removes the need to set a fixed threshold manually and adapts to global changes of illumination in workshop environments; for non-uniform illumination, local adaptive thresholding is required instead. For the holes and scattered noise pixels remaining in the binary image, morphological opening and closing operations are used to fill internal voids within objects and remove isolated pixels, thereby smoothing the outer contours of workpieces. The influence of each preprocessing stage on workpiece contour extraction is illustrated in Figure 4: after the original image undergoes grayscale conversion, Gaussian filtering, Otsu binarization, and morphological closing, noise is progressively eliminated and the target regions are completely filled. Finally, the `findContours` function is employed to achieve accurate contour extraction and grasping localization of the workpieces.

3.2 Algorithm Selection for Static Workpiece Recognition

After preprocessing, contour detection is employed to achieve coarse target localization. The `findContours` function is invoked to extract connected regions, and workpieces of different sizes are distinguished by combining the contour area and the circumscribed-circle dimensions, thereby enabling material classification and grasping decisions [14]. For regular static threaded workpieces, template matching can be applied directly for recognition, offering fast computation and strong real-time performance [1]. In scenarios involving workpiece stacking and partial occlusion, the Scale-Invariant Feature Transform (SIFT) [16] is adopted, and correspondences are screened by the Euclidean distance between feature descriptors (usually with a nearest-neighbor ratio

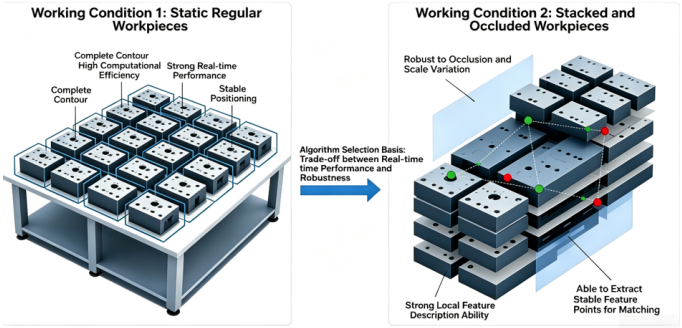

Figure 5. Schematic comparison of algorithm selection for static regular workpieces (template matching) and stacked or occluded workpieces (SIFT feature matching).

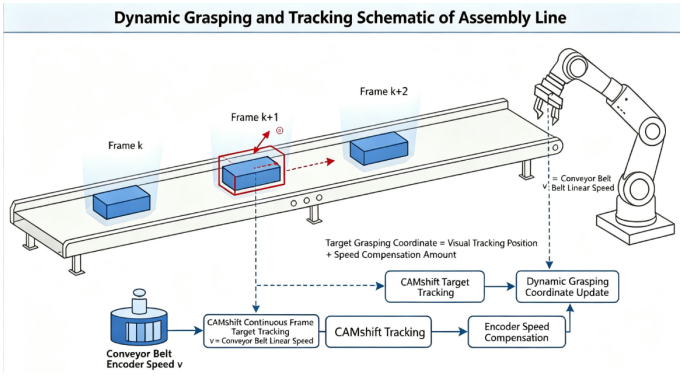

Figure 6. Dynamic grasping and tracking on an assembly line: CamShift tracks the workpiece over consecutive frames, and the conveyor encoder speed v is used to compensate the grasping coordinate.

test), which enhances recognition robustness in complex environments. On a Raspberry Pi-based robotic arm, for instance, SIFT has been reported to reach 97–100% matching accuracy, whereas a FAST+SURF combination is the fastest [3]. For some low-cost embedded devices, SIFT is therefore replaced by hue–saturation–value (HSV) color thresholding to reduce computational consumption, using defined color intervals to rapidly filter spherical or color-coded materials [5, 6]. Figure 5 presents a comparative schematic of template matching and SIFT feature matching. The left panel illustrates the scenario of regular static workpieces, where template matching is suitable for fast localization; the right panel illustrates the scenario of stacked and occluded workpieces, where SIFT feature matching is suitable for robust recognition. In practice, the algorithm selection should be weighed comprehensively against the task requirements for real-time performance, localization accuracy, and occlusion resistance.

3.3 Dynamic Grasping Tracking and Embedded Adaptation Optimization

For dynamic grasping scenarios such as conveyor belts, OpenCV can be combined with the Continuously Adaptive Mean Shift (CamShift) algorithm [17], a mean-shift-based visual tracking method, to continuously lock onto the moving workpiece over consecutive frames; the tracked position is then compensated with encoder velocity feedback so that the grasping trajectory of the robotic arm can be planned in advance, as illustrated in Figure 6. Denoting by $\mathbf{p}(t)$ the tracked target position in the robot base frame at time t (m), by $\mathbf{v}$ the conveyor velocity measured by the encoder (m/s), and by Δt the total latency of image acquisition, processing, and arm motion (s), the predicted grasping point is

$$\mathbf{p}_g = \mathbf{p}(t) + \mathbf{v}\,\Delta t. \tag{4}$$

Tracking and grasping of moving objects with a hand–eye system have been studied since the early 1990s [18].

Embedded hardware platforms such as the Raspberry Pi, the TQ210 board (a Cortex-A8 development board), and STM32-based systems can use pruned, lightweight builds of OpenCV that retain only the basic modules for grayscale conversion, filtering, and contour processing, thereby meeting the real-time image processing requirements of low-computing-power devices [2, 3]. Overall, traditional OpenCV-based image processing solutions are low-cost and easy to deploy; however, relying solely on 2D pixel information, the object depth cannot be obtained directly without additional constraints such as a planar workspace or a known object height. Consequently, recognition accuracy degrades significantly in scenarios with stacked workpieces of varying heights, and binocular calibration or deep learning algorithms are needed to compensate for this limitation.

4 OpenCV-Based Binocular 3D Reconstruction

Monocular vision provides only 2D planar pixel information without target depth or spatial orientation, which makes it inadequate for scenarios such as peg-in-hole assembly and stacked-workpiece grasping. An OpenCV-based binocular vision system, however, accomplishes 3D reconstruction through stereo calibration and disparity computation (Section 2.2 and Eq. (1)), thereby enabling the estimation of 6-DoF (six degrees of freedom) object poses, and has become the core vision solution for robotic-arm grasping under complex working conditions [11, 14]. The dense disparity map yields a 3D point cloud of the scene [8], which can be used directly for environmental obstacle extraction and

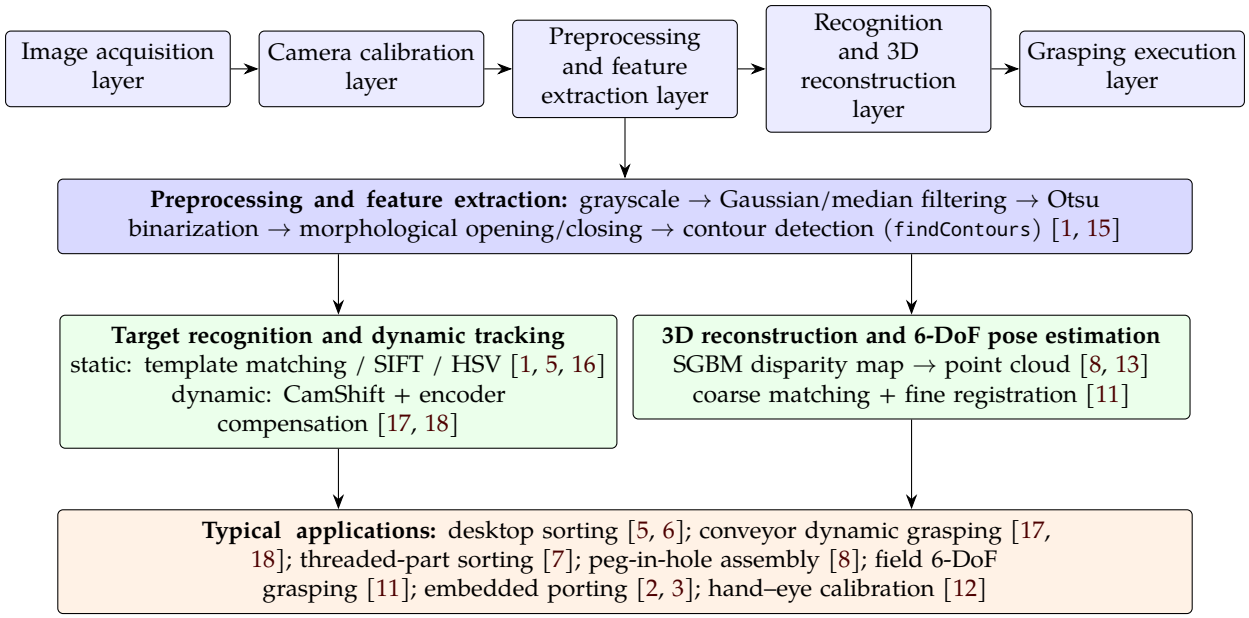

Figure 7. Overall technical framework of OpenCV in robotic-arm grasping vision systems, with the representative studies cited for each stage and application.

provides spatial constraints for collision-free trajectory planning of the robotic arm [9].

For 6-DoF object pose estimation, existing studies have established a two-tier architecture of "coarse matching + fine registration." In the coarse localization stage, SIFT features [16] are extracted from both the left and right views with OpenCV, and 2D correspondences are established through feature matching; combined with the binocular depth information, the coarse spatial coordinates of the target are obtained. The fine pose optimization stage relies on a registration strategy in which the 3D template model is matched against the scene point cloud to solve the six-parameter rigid transformation (three translations and three rotations) [10, 11], which outputs the spatial position of the target as well as its pitch, roll, and yaw angles. In dynamic grasping scenarios on assembly lines, the binocular reconstruction results can be fed back to the motion controller in real time, and a proportional–integral–derivative (PID) controller is used to adjust the grasping angle of the robotic-arm end-effector, thereby reducing dynamic grasping offset errors.

For embedded binocular devices, OpenCV stereo algorithms require lightweight adaptation, namely reducing the disparity search range and the image resolution, to ensure real-time performance on platforms such as the Raspberry Pi [2, 3]. However, current OpenCV-based binocular solutions have limitations: under strong light and occlusion, the disparity maps contain holes, which degrades the point-cloud completeness and the 6-DoF pose accuracy. Future integration of monocular features with depth data may improve robustness. Figure 7 summarizes the overall technical logic chain of OpenCV in robotic-arm grasping vision systems. It integrates the core technical modules of each stage with the corresponding literature contributions, collectively reflecting the overall architecture, key advances, and application boundaries of existing research.

5 Conclusion

This paper takes various robotic-arm grasping scenarios as the research subject and systematically reviews the complete application framework of the OpenCV vision library, covering calibration, image processing, 3D reconstruction, and embedded deployment. First, three types of calibration methods, namely monocular, binocular, and hand–eye calibration, are introduced; with the OpenCV tools, lens-distortion correction and the unification of multiple coordinate frames are accomplished, which reduces positioning deviations in grasping tasks. Second, image processing techniques including grayscale transformation, filtering, contour detection, and feature matching are compared, and differentiated recognition strategies for static workpieces and dynamically moving objects on assembly lines are distinguished. Meanwhile, the binocular stereo matching workflow is elaborated, in which the object depth is recovered from disparity data and the 6-DoF pose is computed, supporting complex operations such

as peg-in-hole assembly and obstacle-aware grasping. Finally, for low-computing-power hardware platforms such as the Raspberry Pi and ARM development boards (with microcontrollers such as Arduino and STM32 serving as controllers), optimization methods such as library pruning, reduced image resolution, and lightweight feature replacement are compiled, which are reported to allow vision programs to run on small embedded devices. The overall solution is open-source, easy to implement, and low-cost in hardware, and it is suitable for conventional, structured industrial scenarios such as assembly-line sorting and small-part grasping.

However, the current traditional OpenCV-based vision grasping solution still has notable shortcomings. Recognition performance degrades significantly under strong illumination and workpiece stacking or occlusion, and the existing lightweight strategies are mainly suitable for simple fixed-station operations. Future work may integrate deep learning algorithms with traditional OpenCV vision to enhance the stability of target recognition under complex working conditions. Online hand–eye calibration can be incorporated to reduce the coordinate drift caused by prolonged robotic-arm operation. In addition, the fusion of depth sensors and multimodal perceptual data can compensate for the insufficient depth information inherent in monocular vision and the limited robustness of binocular vision, further extending the applicability of this vision system to complex and variable industrial production lines.

Data Availability Statement

Not applicable.

Funding

This work was supported without any funding.

Conflicts of Interest

The author declares no conflicts of interest.

AI Use Statement

The author declares that DeepSeek was used for translation of the manuscript from Chinese into English. The authors have carefully reviewed, revised, and verified the AI-assisted output and take full responsibility for the content of the manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA, USA: O'Reilly Media. ISBN 978-0-596-51613-0. https://books.google.com/books?id=seAgiOfu2EIC

[2] Pulli, K., Baksheev, A., Kornyakov, K., & Eruhimov, V. (2012). Real-time computer vision with OpenCV. *Communications of the ACM, 55*(6), 61-69. [CrossRef]

[3] Kaymak, C., & Uçar, A. (2018). Implementation of object detection and recognition algorithms on a robotic arm platform using Raspberry Pi. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. IEEE. [CrossRef]

[4] Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence, 22*(11), 1330-1334. [CrossRef]

[5] Abdullah-Al-Noman, M., Eva, A. N., Yeahyea, T. B., & Khan, R. (2022). Computer vision-based robotic arm for object color, shape, and size detection. *Journal of Robotics and Control (JRC), 3*(2), 180-186. [CrossRef]

[6] Titu, M. F. S., Haque, S. R., Islam, R., Hossain, A., Qayum, M. A., & Khan, R. (2024). Experiments with cooperative robots that can detect object's shape, color and size to perform tasks in industrial workplaces. *International journal of intelligent robotics and applications, 8*(1), 179-192. [CrossRef]

[7] Jain, T., Mushtaq, F., Ramesh, K., Deshmukh, S., Ray, T., Parimi, C., ... & Jha, P. K. (2023). Development of machine vision approach for mechanical component identification based on its dimension and pitch. *arXiv preprint arXiv:2310.01995*. [CrossRef]

[8] Wang, Z., Li, P., Zhang, H., Zhang, Q., Ye, C., Han, W., & Tian, W. (2023). A binocular vision method for precise hole recognition in satellite assembly systems. *Measurement, 221*, 113455. [CrossRef]

[9] Wang, X., Yang, C., Ju, Z., Ma, H., & Fu, M. (2017). Robot manipulator self-identification for surrounding obstacle detection. *Multimedia Tools and Applications, 76*(5), 6495-6520. [CrossRef]

[10] Le, T.-T., Le, Q.-D., Chen, Y.-R., Vidal, J., & Lin, C.-Y. (2021). 6D pose estimation with combined deep learning and 3D vision techniques for a fast and accurate object grasping. *Robotics and Autonomous Systems, 141*, 103775. [CrossRef]

[11] Liang, J., Zhang, J., Pan, B., Xu, S., Zhao, G., Yu, G., & Zhang, X. (2021). Visual reconstruction and localization-based robust robotic 6-DoF grasping in the wild. *IEEE Access, 9*, 72451-72464. [CrossRef]

[12] Tsai, R. Y., & Lenz, R. K. (1989). A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Transactions on robotics and automation, 5*(3), 345-358. [CrossRef]

[13] Hirschmüller, H. (2008). Stereo processing by

semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(2), 328-341. [CrossRef]

[14] Vo, C. D., Dang, D. A., & Le, P. H. (2022). Development of multi-robotic arm system for sorting system using computer vision. *Journal of Robotics and Control (JRC), 3*(5), 690-698. [CrossRef]

[15] Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics, 9*(1), 62-66. [CrossRef]

[16] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision, 60*(2), 91-110. [CrossRef]

[17] Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, 2nd Quarter.

[18] Allen, P. K., Timcenko, A., Yoshimi, B., & Michelman, P. (1993). Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation, 9*(2), 152-165. [CrossRef]

Rui Du received his bachelor's degree from Yancheng Institute of Technology, Yancheng, China. He is currently pursuing the master's degree in Electrical Engineering at Yancheng Institute of Technology. His research interests include robotic grasping, simultaneous localization and mapping (SLAM), and navigation of wheeled robots. (Email: 1229390784@qq.com)