RoboCam: Model-Based Robotic Visual Sensing for Precise Inspection of Mesh Screens

SIYUAN ZHOU, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
DUC VAN LE, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
LINSHAN JIANG*, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
ZHUORAN CHEN†, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
XIAOHUA PENG‡, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore
DAREN HO, HP Inc., Singapore
JIANMIN ZHENG, College of Computing and Data Science, Nanyang Technological University, Singapore
RUI TAN, College of Computing and Data Science, Nanyang Technological University, Singapore

The 3D-printed mesh screen with dense penetrating pores is a new structure for massive manufacturing of molded pulp package products. However, some of the pores may be clogged by the printing material powder during the printing process. Such defects negatively affect the quality of the pulp packages produced using the mesh screen mold. To pinpoint the defects, we design a model-based robotic visual sensing system, called RoboCam, which uses a robotic arm to carry a high-resolution camera for full inspection of a mold consisting of joined mesh screens. To inspect the entire mold, RoboCam plans the camera poses to capture multiple images of the mold and render synthesized images as references for identifying the clogged pores. In particular, we propose novel designs to rectify the inherent run-time pose errors of the robotic system for ensuring the reference quality and to accelerate the reference rendering for reducing inspection latency. Extensive evaluation shows that RoboCam's design outperforms various baselines, including three existing computer vision and convolution neural network-based inspection systems. RoboCam achieves a recall rate of 94.95% within 528 seconds latency for inspecting an entire mold with 13,000 designed pores.

CCS Concepts: • **Computer systems organization → Sensor networks**.

*Linshan Jiang is now with the National University of Singapore.
†Zhuoran Chen is now with the Hong Kong University of Science and Technology.
‡Xiaohua Peng is now with the Aputure.

Authors' addresses: Siyuan Zhou, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore, siyuan.zhou@ntu.edu.sg; Duc Van Le, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore, levanduc@ieee.org; Linshan Jiang, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore, LINSHAN001@e.ntu.edu.sg; Zhuoran Chen, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore, zhuoran.chen@ntu.edu.sg; Xiaohua Peng, HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University, Singapore, XIAOHUA001@e.ntu.edu.sg; Daren Ho, HP Inc., Singapore, kok-loon.ho@hp.com; Jianmin Zheng, College of Computing and Data Science, Nanyang Technological University, Singapore, ASJMZheng@ntu.edu.sg; Rui Tan, College of Computing and Data Science, Nanyang Technological University, Singapore, tanrui@ntu.edu.sg.

ACM 1550-4859/2025/1-ART
https://doi.org/XXXXXXX.XXXXXXX

Additional Key Words and Phrases: Industrial IoT, smart manufacturing, visual sensing

1 INTRODUCTION

Additive manufacturing, also known as 3D printing, has been increasingly adopted for fast prototyping various intermediary or final products involving complex 3D structures. However, it is challenging to perform thorough quality inspection of the printed products. In this paper, we consider a representative target application of inspecting our industry partner's 3D-printed mesh screen molds [4] with a sample shown in Fig. 1. The mold is used to mass-produce pulp packages (e.g., containers, dinnerware, packaging boxes). A mesh screen on the mold is a complex 3D structure with many penetrating pores for draining unused pulp away. The major defects with the mesh screens are the pores clogged with the printing material powder. Due to the clogging, the molded pulp products may have non-uniform thicknesses and become fragile [37]. Therefore, it is critical to detect and locate the clogged pores during an inspection process. Human eye-tracking with a microscope is a possible solution. However, such manual inspection is subjective, error-prone, and laborious. Moreover, as the inspection is planned as an essential function of our industry partner's automated cleaning system [3] which eradicates the clogging through vibration, vacuum, and air blasting, automated inspection is desirable to automate the whole process.

In this paper, we design and implement a model-based robotic visual sensing system, called *RoboCam*, which employs a robotic arm to carry a camera for automated and precise inspection of the 3D-printed mesh screens. RoboCam counts the clogged pores and locates them on the 3D-printed mesh screens. While the target mesh screen inspection application is important because of the increasing molded pulp package manufacturing, RoboCam also provides insights into the capability limit of a robotic visual sensing system in addressing practical industrial inspection tasks. In what follows, we discuss the three main challenges in the design of RoboCam and the basic ideas to address these challenges.

First, the pore sizes are at sub-millimeter level. Thus, to effectively image the pores, wave signals with short wavelengths should be used to avoid undesirable diffraction and interference effects when they go through the pores. The existing industrial defect inspection systems employ various sensing modalities including visible light [15], ultrasound [43], terahertz wave [20], X-ray [17], and laser [26]. RoboCam uses visible light due to its short enough wavelengths and the sensing equipment's advantages of low cost, easy deployment and operation. X-ray is also effective for imaging sub-millimeter-level pores, but it is hazardous and the X-ray imaging systems are expensive (e.g., up to 90k USD [7]). Terahertz systems are even more expensive (e.g., up to 215k USD [25]). In contrast, RoboCam's main hardware components, including a robotic arm, a high-resolution camera, and a computer workstation, cost a total of 26k USD only. Moreover, although laser can work in the visible spectrum, the laser emitter needs to point to the pore exactly and the inspection can only be performed on a pore-by-pore basis, which is inefficient. Ultrasound has centimeter-level wavelengths, which are incapable of imaging the pores.

Second, the pores are densely and non-uniformly distributed on the mesh screens, providing no simple patterns to follow when inspecting them. For instance, as shown in Fig. 1, a mesh screen consists of about 2,000 non-uniformly distributed pores. As such, the manual inspection with an *ad hoc* nature has limited/no basis to achieve the trustworthiness required by the industrial applications. In addition, as evaluated in this paper, the recent convolutional neural network (CNN)-based object detection [23, 44, 48] and image segmentation [15, 38] approaches cannot detect the clogged pores with satisfactory performance. The main reason is that detecting clogged pores requires comparison between actual pores and designed pores; however, CNN-based approaches aim to learn the feature patterns of clogged pores. Since the features of individual clogged pores are similar to the structures of mesh screens, object detection methods cannot accurately differentiate the clogged and unclogged pores. Meanwhile, image segmentation might not accurately segment out clustered clogging with cluster patterns unseen in the training dataset. Furthermore, due to the small size and high density of pores, labeling the mesh

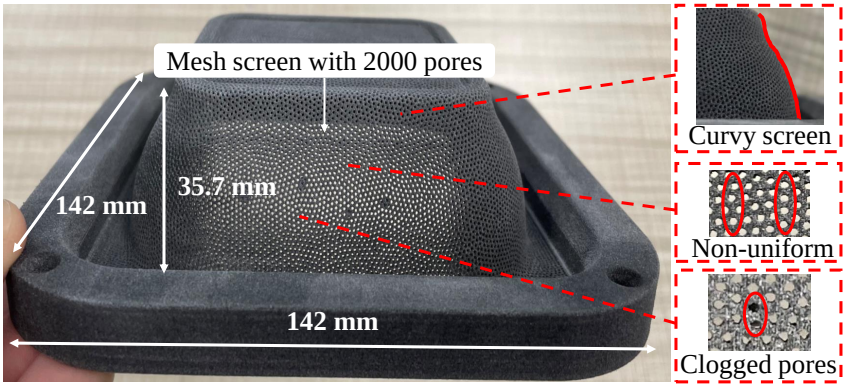


Fig. 1. A 3D-printed pulp package mold (length: 142mm, width: 142mm, height: 35.7mm) consisting of joined mesh screens. A screen has up to 2,000 non-uniformly distributed, sub-millimeter-sized pores (0.283mm^2) on its curvy surface.

screen images for training the CNNs is tedious and labor-intensive. Instead of using the training-based approaches, RoboCam utilizes the mesh screen's computer-aided design (CAD) model, camera's pose and intrinsics to render references (i.e., images of the defectless mesh screen). Then, RoboCam compares the real images captured by the camera and the rendered references to detect the clogged pores. This inspection task is a post-processing step for the 3D-printed mold manufacturing which produces the mesh screens according to their CAD model. Thus, the assumption on the availability of the CAD model holds in our problem context.

Third, the 3D-printed mesh screens can be curvy. As a result, a single image captured from a certain camera's view angle may not cover all the pores on the curvy surface. To fully inspect the mesh screen, the camera needs to move and capture multiple images from various view angles. One possible approach is a precalibrated multi-camera array to inspect customized 3D-printed mesh screens from multiple fixed view angles [32]. However, given the limited quantity of each customized screen (even down to a single piece), frequently calibrating the multi-camera array to different customized mesh screens leads to significant overheads. Differently, RoboCam uses a robotic arm to move the camera by following computed waypoints for fully covering the mesh screen. Thus, it adapts to the customized screen readily. However, any robotic arm has stochastic movement errors because of its mechanical deformation, assembly, and machining deviations [13, 27]. Thus, the robotic arm cannot reach the planned camera pose exactly. Our profiling experiments show that the Euclidean distance component of the pose error is normally within 1cm to 2cm. Such pose errors cause discrepancies between the rendered reference and the captured image, which negatively affect the inspection performance. To achieve high accuracy, we propose a rectification approach that performs in both the physical and digital domains, aiming at eliminating the discrepancies.

Based on the above component designs, we construct RoboCam that inspects a mesh screen in various poses via multiple iterations. Each iteration begins with planning a virtual 6-degree-of-freedom (6DoF) camera pose using the CAD model of the inspected mesh screen. The planned 6DoF pose allows the camera to inspect the largest region among the mesh screen regions which have not been inspected in the previous iterations. After the robotic system moves the camera to the planned pose with an unknown pose error and captures an image, it performs the following two-domain rectification. First, it determines a rectified pose as a new movement target of the robotic system for compensating the pose error. The robotic system further moves the camera to the rectified pose to complete the physical-domain rectification. Then, it captures a new image and uses the mesh screen's CAD model, camera intrinsics, and the estimated camera pose of the newly captured image to render a reference

image. This forms the digital-domain rectification. Finally, RoboCam compares the captured and rendered images to pinpoint the clogged pores. The inspection results of all iterations until full coverage of the mesh screen are aggregated as the final output.

We implement RoboCam on a real robotic arm mounted with a high-resolution camera and extensively evaluate its performance for inspecting 3D-printed molds with both flat and curvy mesh screens. We evaluate RoboCam by comparing its component designs with the respective baselines, including three approaches for camera pose planning, three for camera pose estimation, one for system error rectification, three for image rendering, and four for image processing-based inspection. The results show that RoboCam achieves a recall rate of 94.95% within 528 seconds latency for inspecting an entire mesh screen mold with 13,000 designed pores. Besides the detection of clogged pores, RoboCam can also measure the sizes of the unclogged pores, which form another metric of the 3D printing quality of the mesh screen.

Our contributions are summarized as follows:

- We design and implement RoboCam which employs a novel model-based approach for automated and precise inspection of 3D-printed mesh screens. RoboCam provides a useful reference for the development of other 3D-printed object inspection systems in which the prior 3D geometric information can be utilized to guide the 2D imaging with high-precision requirement.
- We propose a two-domain rectification approach to minimize the discrepancy between the captured image and its rendered reference image. In the rectification process, we adopt a model-based PnP method without pre-marking to automatically estimate the camera pose. Moreover, we propose a greedy-based parallel rendering approach that achieves 0.3 seconds per image, which is 10 times faster than a widely adopted rendering engine.
- We set up a real testbed to evaluate RoboCam and compare its key component designs with various respective baselines. Specifically, RoboCam's reference-based inspection achieves 1.51x and 1.32x improvement in terms of recall and precision, respectively, compared with an existing CNN-based approach [15].

The remainder of this paper is organized as follows. §2 reviews related work. §3 describes the hardware setup and profiling experiments. §4 presents the detailed designs of RoboCam. §5 presents the evaluation results. §6 concludes the paper.

2 RELATED WORK

■ **Industrial visual inspection:** The existing industrial visual inspection systems [15, 19, 31, 38, 44, 48, 49] can be divided into the following three categories. The first category [31, 44, 48] employs the object detection algorithms to inspect the manufactured products. For instance, the technical note [31] described a CNN-based smart camera system that was designed to detect air bubbles for inspecting ink cartridges on the manufacturing line. The study in [44] developed a real-time inspection system using CNN to detect the surface defects of the molded pulp products moving on the conveyor belts. Moreover, the study in [48] used Mask R-CNN [24] to detect and assess oil leaks on freight trains. The second category [15, 38] implements CNN-based image segmentation to segment out the defect surface regions of the inspected products. For instance, in [15], a visual inspection system adopted a CNN-based segmentation framework, called PSPNet [50], to estimate the air volume in the manufactured ink bags. The study in [38] used U-Net, a CNN-based segmentation algorithm, for metal product surface defect inspection. However, as shown by our experiments in §5, these two categories of approaches have inferior performance in inspecting the clogged pores in our application.

Similar to RoboCam, the third category [19, 49] employs the reference-based approaches to inspect the defects of industrial products such as printed circuit boards [19] and fabrics [49]. They adopted real images of the defectless products that are pre-captured offline from predefined camera's view angles as references. RoboCam moves the camera and captures multiple images from various view angles to fully inspect the curvy mesh screen

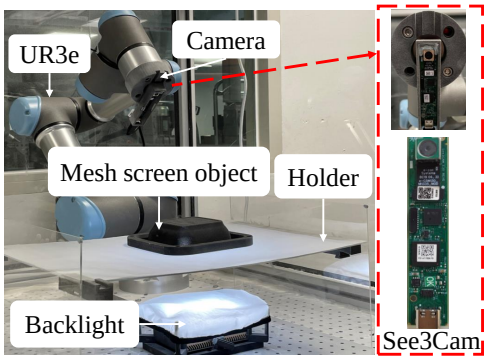


Fig. 2. Hardware setup.

via multiple iterations. Thus, pre-capturing the reference images is infeasible to RoboCam since the view angles are dynamically determined, and thus unpredictable. The study [36] employed a 3D CAD-based direct comparison method. Specifically, it used a laser scanner to capture point cloud data of the piping system, then compared geometrically with the design CAD model. Similarly, our solution adopts a CAD-based method to construct 2D defect-free references for comparison. Therefore, RoboCam uses the product's CAD model and camera parameters to render references online.

■ **Robotic visual sensing:** Several existing industrial inspection systems [27, 34] employ the robotic arm-mounted cameras for inspecting the product defects. For instance, in [34], a robotic arm system with a high-resolution camera was prototyped to monitor the rim's defects. Moreover, the authors in [27] proposed a K-means-based region segmentation algorithm to design a motion planning approach for scanning the specular surfaces for defects. Specifically, such robotic camera systems can perform highly flexible motions and adapt to product structures under inspection. However, the robotic system errors negatively affect the inspection performance. To address this issue, the existing studies have incorporated supplementary setups and equipment such as marker images [13] and depth cameras [27], for online error rectification. Differently, we rectify the errors without the need of additional equipment.

3 ROBOCAM'S HARDWARE & SYSTEM PROFILING

3.1 Hardware Setup of RoboCam

Fig. 2 shows the main hardware components of RoboCam. For sensing, we choose a high-resolution USB camera See3Cam [6] with 13 megapixels CMOS sensor (1/3.2-inch) which can capture images with 3840×2160 resolution at 30 frames per second (fps). We use a 3D-printed connector to mount the camera on the end-effector of a collaborative robotic arm called UR3e [5]. The UR3e has six rotation joints which allow moving the camera and capturing an image of the inspected 3D-printed object from an arbitrary view angle. A single-board computer (i.e., Raspberry Pi) is used to control the image capturing. Moreover, the inspected 3D-printed object is placed on a translucent resin holder near the robotic arm. We add a battery-based LED backlight under the holder to illuminate the pores during the image capturing process. A workstation computer is used to execute the pore inspection software pipeline.

3.2 Robotic System Error Profiling

The robotic system errors include the constant transformation error, denoted by S, and the dynamic movement error.[1] The transformation error is caused by the physical-virtual coordinate transformation. The S can be

[1]A notation table is provided in Appendix.

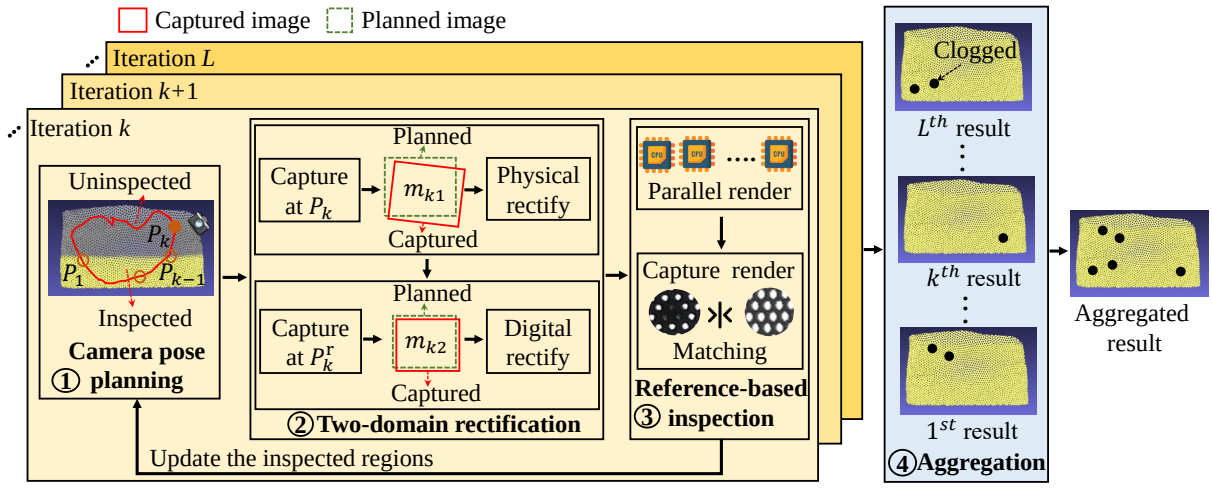


Fig. 3. Design overview of RoboCam.

estimated offline during system preparation and rectified during robotic arm movement planning. Specifically, to reach a target pose, denoted by P_k, the robotic arm should be directed to the pose $P_k - S$ to capture the image. While addressing S is straightforward, addressing the dynamic and stochastic movement errors is non-trivial.

We calibrate our robotic system using the hand-eye calibration approach [47]. Specifically, the hand-eye calibration approach aims to compute the unknown spatial transformation X, including translation and rotation between the mounted camera's pinhole center and the robotic arm's end-effector by solving $AX = XB$ matrix formulation, where A and B represent the poses of the end-effector and camera, respectively. We collect 15 sets of A and B to compute X. The end-effector poses A are provided by the robotic arm system, while the camera poses B are estimated using the checkerboard and perspective-n-point (PnP) method [35]. To obtain the camera's ground truth pose during movement, the camera captures an image containing a checkerboard with a 6×9 pattern of known dimensions and position, which is used to estimate the actual pose. Specifically, given the captured image, we use the PnP approach to calculate the ground truth of the camera's actual pose using the 2D-3D correspondences. We use the checkerboard corner detection approach [16] to detect the 2D pixel positions of the checkerboard's corner points. These 2D pixel positions, along with their known 3D positions, serve as the 2D-3D correspondences for the PnP algorithm. When the camera is planned to move to the target pose P_{cam}, the robotic arm's target pose P_{arm} can be calculated as follows: $P_{\text{arm}} = X^{-1} \cdot P_{\text{cam}}$. Specifically, starting from an initial 6DoF pose P_e $(x_e, y_e, z_e, \phi_e, \theta_e, \varphi_e)$, the robotic arm moves the camera and captures images at 15 randomly selected poses. The new pose P_n is defined as: $P_n = (x_n, y_n, z_n, \phi_n, \theta_n, \varphi_n)$, where: $(x_n, y_n, z_n, \phi_n, \theta_n, \varphi_n) = (x_e + \delta_n, y_e + \delta_n, z_e + \delta_n, \phi_e + \delta_{\phi_n}, \theta_e + \delta_{\theta_n}, \varphi_e + \delta_{\varphi_n})$. The translation components δ_x, δ_y, and δ_z are randomly selected from the ranges [-15cm, 15cm], [-15cm, 15cm], and [-20cm, 20cm], respectively, while the rotational variations δ_{ϕ_e}, δ_{θ_e}, and δ_{φ_e} are randomly chosen from the ranges $[-45°, 45°]$, $[-45°, 45°]$, and $[-45°, 45°]$, respectively. The estimated camera pose P_n and its end-effector pose are used as inputs for the hand-eye calibration, with the output being the X between the camera and the end-effector.

In this section, we conduct a set of profiling experiments to investigate the dynamic movement errors of our robotic system. Specifically, we define an initial 6DoF camera pose, denoted by P_o, with three Cartesian coordinates, denoted by (x_o, y_o, z_o), and three Euler angles, denoted by $(\phi_o, \theta_o, \varphi_o)$. Then, the robotic system repeatedly moves the camera from P_o to 25 target poses, denoted by $P_m = (x_m, y_m, z_m, \phi_m, \theta_m, \varphi_m)$ where $m = 1, \ldots, 25$, to capture

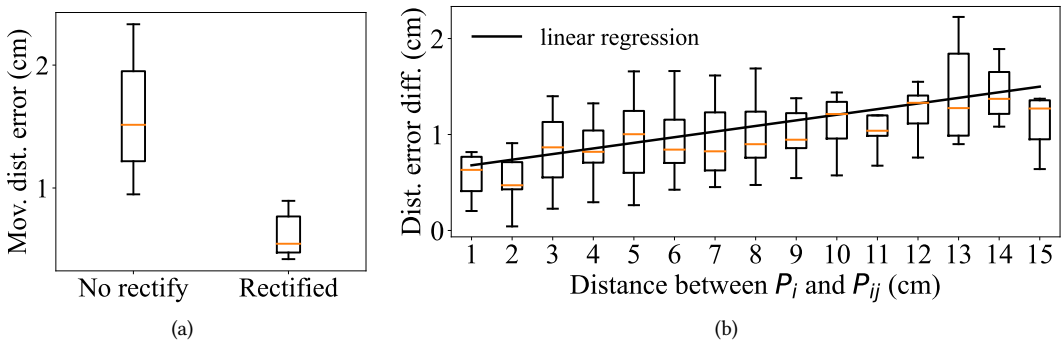


Fig. 4. Profiling the robotic movement errors: (a) Distance errors; (b) Difference between the distance errors of two camera poses.

25 images. We define P_m as $(x_m, y_m, z_m, \phi_m, \theta_m, \varphi_m) = (x_o + \delta_x, y_o + \delta_y, z_o + \delta_z, \phi_o, \theta_o, \varphi_o)$, where the δ_x, δ_y, and δ_z are randomly selected in ranges of [0, 30cm], [0, 30cm], and [0, 40cm], respectively. We estimate the actual camera pose, denoted by $\tilde{P}_m = (\tilde{x}_m, \tilde{y}_m, \tilde{z}_m, \tilde{\phi}_m, \tilde{\theta}_m, \tilde{\varphi}_m)$, from the image captured in the target pose P_m using a above checkerboard method [35]. Then, we consider the Euclidean distance, denoted by d_m, and cosine distance, denoted by C_m, between P_m and $\tilde{P}_m$ as the robotic movement distance and orientation errors. Specifically, $d_m = \sqrt{(x_m - \tilde{x}_m)^2 + (y_m - \tilde{y}_m)^2 + (z_m - \tilde{z}_m)^2}$, and $C_m = 1 - \frac{\phi_m \cdot \tilde{\phi}_m + \theta_m \cdot \tilde{\theta}_m + \varphi_m \cdot \tilde{\varphi}_m}{\sqrt{\phi_m^2 + \theta_m^2 + \varphi_m^2} \cdot \sqrt{\tilde{\phi}_m^2 + \tilde{\theta}_m^2 + \tilde{\varphi}_m^2}}$. The first boxplot in Fig. 4(a) shows the distribution of the movement distance errors d_m of the 25 target poses, which are mostly distributed in a range of [0.9cm, 2.2cm]. These results show that the robotic movement errors are dynamic and vary with the target pose. Meanwhile, the cosine distance C_m of the 25 target poses is concentrated in a narrow range around 0.01, which indicates that the planned and actual poses have similar orientations angles, i.e., little orientation errors. Thus, we focus on reducing the movement distance errors.

Furthermore, we investigate the difference between the movement distance errors of two target poses. In particular, we define ten initial poses, denoted by P_{i0} where $i = 1, \ldots, 10$. For each P_{i0}, the robotic system moves the camera to P_{i0} (with movement error) to capture an image. Then, it further moves the camera from P_{i0} along a certain axis to capture images in 15 more target poses, denoted by P_{ij} where $j = 1, \ldots, 15$. The distance between two target poses $P_{i(j-1)}$ and P_{ij} is 1cm. In total, the camera captures 160 images. The actual camera pose of each image is estimated. Each boxplot in Fig. 4(b) shows the distribution of error differences given the distance d^i_{jk} between the planned P_{ij} and P_{ik} as shown on the x-axis. Note that an error difference is the difference between the movement distance errors of P_{ij} and P_{ik} where $j \neq k$. We can see that the error difference shows a slight increasing trend as presented by the fitting line in Fig. 4(b). However, the difference is mostly less than 1cm when $d^i_{jk} \leq$ 5cm. Thus, the close target poses on the straight trajectory have similar movement distance errors. This provides insights to guide the design of our two-domain rectification in §4.

To better understand the robotic arm system error, we further break down the robotic system errors into two cases: the robotic arm without a camera and the robotic arm with a camera. We collect 15 poses of the robotic arm to measure its error without a camera. Specifically, a 3D-printed cone tool is attached to the robotic arm's end-effector to indicate the centroid point positions along the x-axis and y-axis, as shown in Fig. 5. A ruler is used to measure the ground truth positions of the centroid points along the x-axis and y-axis. The system error is defined as the Euclidean distance between the measured and target positions along these axes. Moreover, we re-measure the robotic arm system error using a camera by capturing an additional 15 poses. The

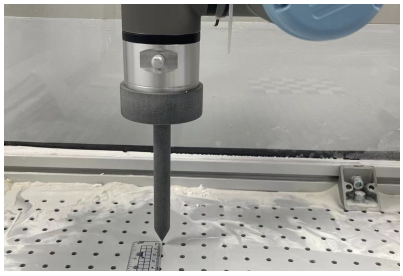


Fig. 5. Robotic arm moving error measurement.

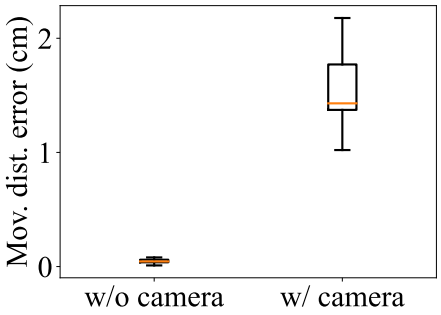


Fig. 6. Robotic arm system error.

Euclidean distance between the target camera poses and the ground truth camera poses is computed as the system error. The ground truth pose is obtained using the checkerboard and PnP method mentioned above. Fig. 6 shows the distribution of robotic system errors both without and with the camera. The robotic arm without the camera has sub-millimeter errors, while the robotic arm with the camera exhibits an error range of [0.9cm, 2.2cm]. The errors of the robotic system with the camera arise from the pose accuracy obtained through the checkerboard and PnP methods [41]. Specifically, vibrations at the end of the robotic arm can reduce the accuracy of checkerboard corner detection, which in turn affects the camera's pose estimation. The differences between the intrinsic parameters provided by the camera specifications and those of the actual manufactured camera can affect the PnP calculation accuracy. Additionally, accumulated errors in the robotic arm's end-effector poses and the camera's pose estimation during calibration can affect the overall accuracy of the calibration. These factors present challenges in achieving precise calibration of the robotic arm system.

4 DESIGN OF ROBOCAM

To fully inspect a 3D-printed mesh screen object, we adopt a 3D mesh segmentation method [9] to divide the object into several mesh screens. Each mesh screen consists of multiple 3D faces with similar normal vector directions. To detect the clogged pores on each mesh screen, RoboCam iteratively runs the inspection pipeline consisting of the following four steps: camera pose planning, two-domain rectification, reference-based inspection, and result aggregation, as illustrated in Fig. 3. In what follows, we present the detailed designs of these four steps.

4.1 Camera Pose Planning

At the beginning of each iteration k ($k \geq 1$), RoboCam selects the largest mesh screen region w_k (a set of vertices and faces), which has not been inspected in the previous iterations. Then, it plans a 6DoF pose, denoted by P_k, in which the camera can capture the image for inspecting the pores in w_k.

4.1.1 Selection of w_k. In our application, each mesh screen consists of up to one million 3D points in its CAD model. Specifically, a 3D mesh point, denoted by $p_i = (x_i, y_i, z_i, \phi_i, \theta_i, \varphi_i)$ including Cartesian coordinates (x_i, y_i, z_i) and Euler angles $(\phi_i, \theta_i, \varphi_i)$. Searching and segmenting out w_k from the massive 3D points is extremely compute-intensive [42]. Thus, to reduce the latency, RoboCam employs a 3D-2D projection-based approach to select w_k.

Specifically, we adopt the pinhole camera model [40] to project the entire 3D mesh screen to a 2D image plane, denoted by C, as illustrated in Fig. 7. Let $K = (c_x, c_y, f_x, f_y)$ denote the camera's intrinsic parameters: pixel coordinates (c_x, c_y) for the principal point and focal lengths (f_x, f_y) in both axes. These parameters can be obtained from camera specifications or offline measurements. Then, the 3D point $p_i = (x_i, y_i, z_i, \phi_i, \theta_i, \varphi_i)$ is

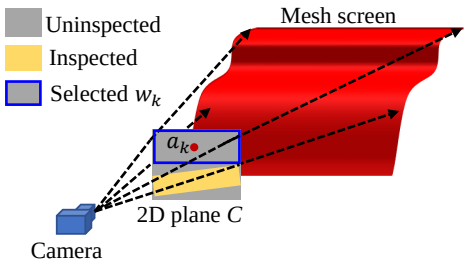


Fig. 7. Project the 3D mesh screen to the 2D plane.

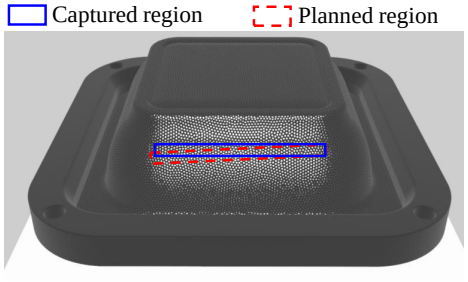


Fig. 8. Mismatch between planned and captured regions.

projected to a 2D image point in C, denoted by p_j with pixel coordinates (x_j, y_j). The pixel coordinates can be derived as $x_j = f_x \frac{x_i}{z_i} + c_x$ and $y_j = f_y \frac{y_i}{z_i} + c_y$. This 3D-2D projection preserves perspective and geometric information while losing depth information. As a result, multiple 3D points with the same Cartesian coordinates but different Euler angles are projected to the same point in C. Nevertheless, this projection greatly reduces the number of the points to be processed to determine w_k.

We implement the following approach to determine w_k in C. For instance, as illustrated in Fig. 7, the gray and yellow 2D plane regions represent the uninspected and inspected mesh regions, respectively. To decide w_k, we divide C into multiple square windows of $W \times W$ pixels, where W is the window size. Then, we gradually extend each window by increasing its width and height with a step size of one pixel until the window reaches the inspected pixel. Finally, the window with the largest number of the gray pixels is selected as w_k.

In the first iteration, w_1 is selected as the rectangle window that contains the entire mesh screen. However, due to the curvy surface, the first iteration can only inspect the pores on the inspectable mesh screen regions. We will discuss our approach for identifying these inspectable regions in §4.3.2. Similarly, the iteration k $(k > 1)$ may not be able to inspect all pores on its w_k. Thus, RoboCam repeats multiple iterations to inspect the mesh screen. The iteration stops when there is no further improvement in inspection coverage.

4.1.2 Determination of P_k. Given w_k, the camera should maintain a distance, denoted by d_{co} (i.e., the camera-object distance), from w_k and look at w_k's 3D mesh screen region. Let (x_k, y_k, z_k) and $(\phi_k, \theta_k, \varphi_k)$ denote the Cartesian coordinates and Euler angles of the 6DoF pose P_k, respectively. Calculating P_k requires the Cartesian coordinates and the Euler angles of w_k center point's projection on the 3D mesh screen, which are denoted by (x_a, y_a, z_a) and $(\phi_a, \theta_a, \varphi_a)$, respectively. RoboCam employs the ray-casting technique [12] to map the center point, denoted by a_k, of the 2D region w_k to the 3D mesh screen. Let R_ϕ, R_θ, and R_φ denote the 3×3 rotation matrices corresponding to the rotations of ϕ_a, θ_a, and φ_a, respectively. Then, the parameters of the P_k are determined by $(x_k, y_k, z_k)^\top = (x_a, y_a, z_a)^\top + d_{\text{co}} R_\phi R_\theta R_\varphi (0, 0, 1)^\top$ and $(\phi_k, \theta_k, \varphi_k)^\top = -(\phi_a, \theta_a, \varphi_a)^\top$.

4.2 Two-Domain Rectification

In this step, the robotic system moves the physical camera to the planned pose P_k and captures an image, denoted by m_{k1}. However, due to the movement errors of the robotic arm, the actual pose of the camera when taking m_{k1}, denoted by P_{k1}, may be different from P_k. Thus, the rendered image with the camera at P_k may be an invalid reference for m_{k1}. Moreover, m_{k1} may not fully cover the planned region w_k, as illustrated in Fig. 8. Based on the movement error profiling results presented in §3.2, we design an approach that rectifies the errors in both the physical and digital domains as illustrated in Fig. 3. It first uses a camera pose estimation method based on the perspective-n-point (PnP) algorithm [18] to estimate the camera's pose P_{k1} from m_{k1}. Then, the P_{k1} is used to determine the rectified pose P_k^{r}. The robotic system further moves the camera to the new target pose P_k^{r} to

complete the physical-domain rectification. Then, the camera takes the second image and yields the camera pose estimated by the PnP algorithm to complete the digital-domain rectification.

4.2.1 Estimation of the camera pose. We design a model-based camera pose estimation approach based on the PnP algorithm [18] which has been widely adopted to estimate the camera's pose from a 2D image captured by the camera. Compared with existing PnP algorithm, our approach fully automates camera pose estimation without the need for pre-marking. End-to-end deep learning methods [10] also utilize 2D images for pose estimation. However, the lack of depth information in 2D images limits their ability to achieve accurate 3D pose estimation. To calculate the six parameters of P_{k1} from m_{k1}, the PnP algorithm requires at least six 2D-3D corresponding pairs, each consisting of a 2D image pixel point and its corresponding 3D point in the mesh screen's CAD model. More 2D-3D corresponding pairs are useful to improving pose estimation accuracy. The main challenge in applying the PnP algorithm lies in the identification of these 2D-3D corresponding pairs. A possible approach is to pre-mark at least six anchor points on the inspected mesh screen area inside the camera view. Then, the 2D handcrafted features of these anchor points are extracted to identify the 2D-3D corresponding pairs. However, as RoboCam iteratively captures multiple images in various poses, many anchor points will be needed.

To avoid the pre-marking, our model-based PnP approach renders a 2D image, denoted by m_k, with the planned pose P_k for fully automated identification of 2D-3D corresponding pairs. Then, we adopt the Akaze feature extraction approach [11] to extract the contour local features of the 2D points in both the rendered and captured images, i.e., m_k and m_{k1}. Although the two images have different camera poses, they still contain overlapped pixel points. Thus, we identify the 2D-2D corresponding pairs among the overlapped points by comparing their features in terms of Euclidean distance. The obtained set of the 2D-2D corresponding pairs is denoted by $\Phi_{\text{2D-2D}}$. Then, we apply ray casting to obtain the 3D mesh screen points corresponding to the rendered 2D points in the set $\Phi_{\text{2D-2D}}$. Finally, these obtained 3D mesh points and their corresponding captured 2D points are used to estimate the camera pose P_{k1} of the image m_{k1} with the PnP algorithm.

4.2.2 Calculation of the rectified camera poses. To compensate for the robotic dynamic movement errors, a possible method is to capture multiple images at various poses close to P_k, and then select the image whose estimated camera pose is closest to P_k. However, given the robotic arm's six degrees of freedom and fine-grained movement at sub-millimeter level, this approach incurs a significant image capturing workload which is highly undesirable in the online inspection system. In what follows, we present our proposed approach which is based on the assumption that P_k and P_k^{r} will be close to each other. Thus, they have similar movement distance errors as seen from our profiling experiment results in §3.2.

Specifically, we define the ground-truth camera pose of the image m_{k1} as $P_{k1} = P_k + T_{Pk1}$, where T_{Pk1} is the unknown movement error. Let $\tilde{P}_{k1}$ denote the estimated pose of the image m_{k1}, which is the output of the PnP approach described in §4.2.1. The movement error T_{Pk1} can be approximated by $\tilde{T}_{Pk1} = \tilde{P}_{k1} - P_k$, which includes the pose estimation error of the PnP approach. The rectified camera pose is calculated as $P_k^{\text{r}} = P_k - \tilde{T}_{Pk1}$. Then, the robotic arm moves the camera to the pose P_k^{r} to capture an additional image m_{k2}. As the movement error for P_k^{r} is similar to that for P_k, the camera pose with P_k^{r} as the movement target camera pose is closer to P_k, compared with P_{k1}. The second boxplot of Fig. 4(a) shows the distribution of the distance errors d_m between the planned pose and the actual pose of image captured in the rectified pose with 25 target poses in our profiling experiments. From Fig. 4(a), the above physical-domain rectification reduces the distance error range to [0.5cm, 0.8cm]. However, the distance error cannot be completely eliminated due to its stochasticity. Thus, the digital rectification estimates the camera pose of m_{k2} which will be used for rendering the reference image for m_{k2} in the reference-based inspection step.

In summary, our proposed two-domain rectification approach reduces the discrepancy between the images captured in the physical system and its rendered reference image. A simpler approach is to only involve the digital

rectification to estimate the camera pose of the first image m_{k1} for rendering the reference image. However, our experiments in §5.3.1 show that this digital-rectification-only approach requires more inspection iterations (i.e., higher inspection latency) and cannot fully inspect the entire mesh screen. The main reason is that the image m_{k1} may not fully cover the selected region w_k, and cover the regions which have been inspected in the previous iterations. This causes redundant work.

4.3 Reference-based Inspection

In this step, RoboCam renders a reference image based on the mesh screen's CAD model, camera intrinsic parameters, and the estimated pose of the captured image m_{k2}. Then, the reference-based inspection employs Hungarian matching [30] to perform the pore-wise association to detect the clogged pores. Moreover, due to the curvy surface of the 3D mesh screen, the pores of the selected mesh regions which are not perpendicular to the camera pose may appear distorted and occluded on the captured image, reducing the accuracy of the reference-based inspection. To preserve accuracy, we only inspect the pores in the perpendicular/inspectable regions of the captured image. Thus, at the end of each iteration, the inspection results of the inspectable regions are used as the sensing feedback to update the inspected regions on the mesh screen as illustrated in Fig. 3.

4.3.1 Parallel image rendering. We perform image rendering by the rasterization technique [1]. It takes the mesh screen's CAD model, camera intrinsics parameters, and the estimated pose of the image m_{k2} as inputs to render the reference for m_{k2}.

However, rendering a mesh screen with millions of facets is computationally intensive. RoboCam needs to render multiple images to inspect the entire mesh screen via multiple iterations. Thus, the image rendering latency should be minimized to reduce the end-to-end inspection latency. We notice that the rasterization in Blender, an open-source 3D computer graphics software, does not fully utilize all available processor cores. To address this, we propose a greedy parallel rasterization method that maximizes multi-core usage. By employing a greedy task assignment, we balance workloads across workers, reducing end-to-end rendering time.

Specifically, the image m_{k2} is divided into multiple sub-images. We design a greedy approach to assign the rendering tasks of these sub-images to the multiple workers. The end-to-end rendering latency is the highest latency among the workers' latencies. Meanwhile, each sub-image may vary in the number of pores and mesh screen contours/edges, with more pores and edges resulting in higher rendering latency. Thus, the main objective of the rendering task assignment is to balance the rendering workload among the workers such that the end-to-end latency can be minimized. To this end, we build and fit a quadratic polynomial model offline, denoted by Υ. The model takes the image features (including the contrast, brightness, and number of object edges) of the sub-image as input variables to estimate the sub-image rendering latency. In general, higher contrast and brightness values indicate higher pore density, while more edges represent more contours of the mesh screen in the sub-image.

To assign the sub-images of m_{k2} to n workers, we employ an image processing algorithm to extract the contrast, brightness, and number of edges of the sub-images. Then, the trained model Υ is used to estimate the rendering latency of these sub-images based on the extracted features. The sub-images are sorted into an ascending queue according to their estimated rendering latency. The first n sub-images of the queue are assigned to the n workers. These assigned images are removed from the queue. The latency of each worker is set to the total latency of its assigned sub-images. Then, the first sub-image in the queue (i.e., the sub-image with the lowest latency among the unassigned sub-images) is assigned to the worker which has the lowest latency. The iteration stops until all sub-images are assigned. The workers render the references for their assigned sub-images simultaneously. The outputs of all workers are aggregated to yield the entire reference.

4.3.2 Inspectable region selection. We select the inspectable regions which are perpendicular to the camera pose of m_{k2} on the rendered image. Let $\boldsymbol{f}_{\mathbf{u}} = R_\phi R_\theta R_\varphi (0, 0, 1)^\top$ represent the camera pose vector. Here, R_ϕ, R_θ, and R_φ

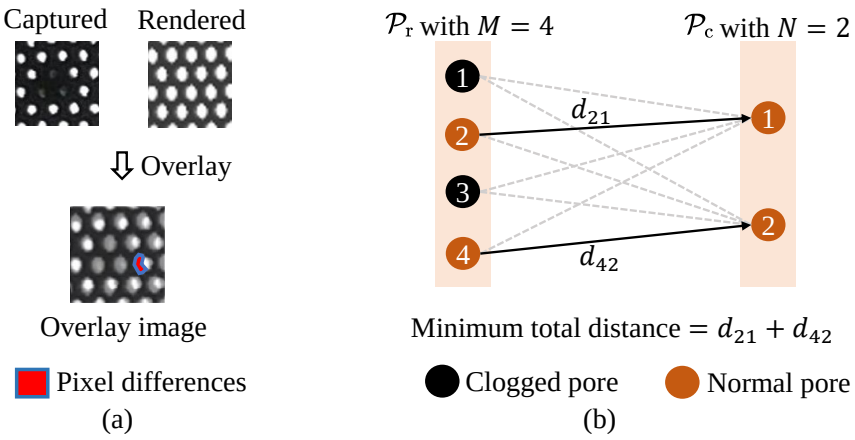


Fig. 9. Reference-based inspection: (a) Pixel-wise comparison; (b) Pore-wise comparison as a matching problem.

are the 3×3 rotation matrices corresponding to the Euler angles θ_{k2}, ϕ_{k2}, and φ_{k2} of the camera pose for m_{k2}, respectively. Then, an inspectable region consists of multiple connected mesh screen faces whose normal vectors are parallel to $\boldsymbol{f}_u$. However, the mesh screen's CAD model may have millions of faces, making it challenging to search all connected faces regions. Note that the connected faces searching needs to start from one face to continuously check the neighbor faces until no connected faces or the face normal vector is not parallel to $\boldsymbol{f}_u$, and then start the next searching. To reduce the latency, we adopt a one-time matrix operation to select the proper normal vectors of the mesh screen instead of the connected face searching. Specifically, given the CAD model face normal vector set $\mathbf{S_o}$, we perform the matrix operation to select the faces set $\mathbf{S}$ as: $\mathbf{S} = \begin{cases} \mathbf{S} \cup s_i & \|-\boldsymbol{u}_i - \boldsymbol{f}_u\| \leq T_l \\ \mathbf{S} & \|-\boldsymbol{u}_i - \boldsymbol{f}_u\| > T_l \end{cases}$, where the s_i is the i^{th} face of the $\mathbf{S_o}$, the $\boldsymbol{u}_i$ is the normal vector of s_i and the T_l is a threshold value. Each s_i is selected to $\mathbf{S}$ if the Euclidean distance between $\boldsymbol{u}_i$ and $\boldsymbol{f}_u$ is less than T_l. The set $\mathbf{S}$ may include several disconnected regions; the pores located in regions far from the camera pose may appear distorted in the captured image. To remove the distorted pores, we project $\mathbf{S}$ to the 2D image plane of rendered image. Given the projected image plane and the rendered image, RoboCam uses bitwise operations to select regions within the rendered image that overlap with the projected 2D image plane. Then, the largest inscribed rectangle of the selected 2D image regions is denoted as the inspectable region.

4.3.3 Clogged pore detection. The clogged pores can be detected by performing pixel-wise comparison between the inspectable regions of the rendered and captured images. However, due to differences in illumination and texture between the rendered and captured images, the pores in the rendered image may not be consistent with those in the captured image. For instance, Fig. 9(a) shows the pixel-level differences on the same pore between rendered and real images. As a result, the pixel-wise comparison of the two images generates excessive false positives. Differently, we employ a pore-wise comparison approach which extracts the pores from both images for comparison. It is presented in the following.

Our design uses the following prior knowledge from the 3D printer's yield quality assurance. First, the printer will not print a pore that is not in the design. Second, the displacement of a pore from its designed position is at most the pore diameter. From our subsequent analysis on the matching results, the displacement is at most 0.56mm and the maximum pore diameter is 0.7mm. Thus, the quality claim is supported by our results. In our design, we process both the rendered and captured images to extract the pores' centroids as pores' positions. Let

M and N denote the number of pores' positions extracted from the rendered and captured images, respectively. Due to the potential clogging, we have $M \geq N$. If $M = N$, the inspectable mesh screen region has no clogged pores. Otherwise, the mesh screen has M-N clogged pores in the inspectable region. Let $\mathcal{P}_\mathrm{r}$ and $\mathcal{P}_\mathrm{c}$ denote the sets of rendered and captured pores' positions, respectively. Now, the objective is to identify the positions of the M-N pores which appear in $\mathcal{P}_\mathrm{r}$, but disappear in $\mathcal{P}_\mathrm{c}$ due to the clogging.

We formulate the clogged pore detection problem as a matching problem which aims to match M pores in $\mathcal{P}_\mathrm{r}$ with N pores in $\mathcal{P}_\mathrm{c}$ with the objective of minimizing the total distance of the matched pore pairs as illustrated in Fig. 9(b). We adopt the Hungarian method to solve the matching problem for finding the clogged pores. Specifically, the method constructs an $M \times N$ cost matrix consisting of the distances of the pore pairs in $\mathcal{P}_\mathrm{r}$ and $\mathcal{P}_\mathrm{c}$. Then, the matrix is refined through row and column operations to pinpoint the N matched pore pairs with the minimum total distance. Finally, the M-N unmatched pores are detected as the clogged pores.

4.4 Result Aggregation

The inspectable regions in different iterations may overlap. As a result, a certain number of pores may be repeatedly inspected in multiple iterations. Thus, we apply a distance-based approach to identify these repeated pores. If the distance between a pore in an iteration and another pore in another iteration is less than a distance threshold (e.g., pore diameter), they are considered as the same pore being repeatedly inspected. Then, we employ the majority voting to yield the majority of the iterations' inspection results for the same pore as the final result for the pore. This voting improves the quality of the detection results.

5 EVALUATION

5.1 Evaluation Setup

We implement RoboCam on a real testbed as presented in §3.1. All computing for the inspection pipeline is executed on a workstation computer with an Intel(R) Xeon(R) silver 4214 CPU of 3.2 GHz frequency and a 32GB RAM. We use the python library *Trimesh* 3.2 to implement the image rendering, ray casting, and CAD model processing. The 2D image processing and PnP-based pose estimation algorithms are implemented using *OpenCV* 3.4. Moreover, we use *Scikit-learn* 1.12 to implement the Hungarian matching and image rendering latency model fitting. We evaluate RoboCam for inspecting two different samples of the mesh screen molds as shown in Fig. 10. Specifically, Sample A and B include about 13,000 and 5,800 pores, respectively. Sample A consists of four curvy and one flat mesh screens, while Sample B has one flat screen. We manually create the clogged pores with different distributions on the mesh screens of the Samples A and B for evaluation. We adopt the following settings for the design parameters of RoboCam. The camera-object distance d_co and threshold T_l are set to 20cm and 0.2, respectively, according to the experiments in the Appendix A.1 and A.2. For finding w_k, the window size W is set to 20 pixels, which is the diameter of a single pore in captured images. In what follows, we evaluate the performance of RoboCam's component designs and the whole pipeline in comparison with respective baseline approaches using various metrics.

5.2 Performance of RoboCam's Component Designs

5.2.1 Camera pose estimation. We evaluate the proposed model-based PnP approach for estimating the camera pose of the captured images. Ideally, the pose estimation accuracy can be measured by the distance difference between the ground-truth and estimated poses. However, it is non-trivial to obtain the ground-truth poses of the captured images. Instead, we adopt an accuracy metric called 2D mask error [21]. To calculate the 2D mask error, we apply the proposed model-based PnP approach to estimate the camera's pose of the captured image. Then, the estimated pose is used to render a reference image. We perform pixel-wise comparison between the captured and rendered images. Two pixels with identical coordinates in two images are deemed unmatched if

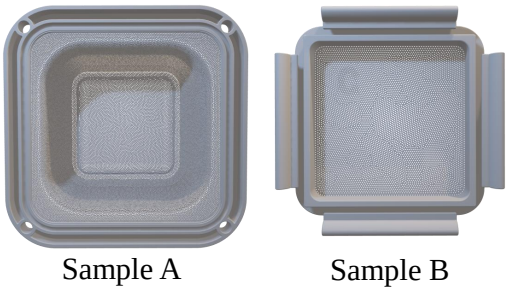


Fig. 10. Mesh screen samples.

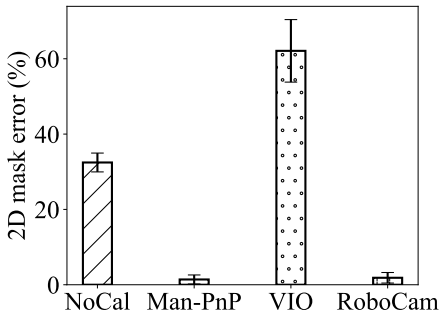
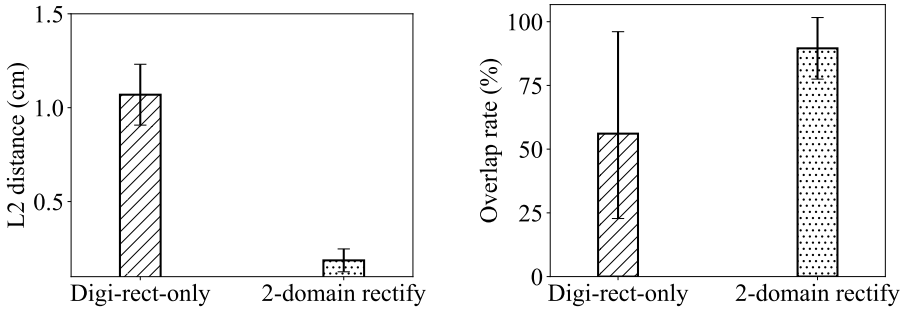


Fig. 11. Pose estimation accuracy.

their absolute pixel value difference exceeds a threshold of 20. The 2D mask error is calculated as the ratio of the number of unmatched pixels to the total number of captured image pixels. Lower mask error indicates higher estimation accuracy.

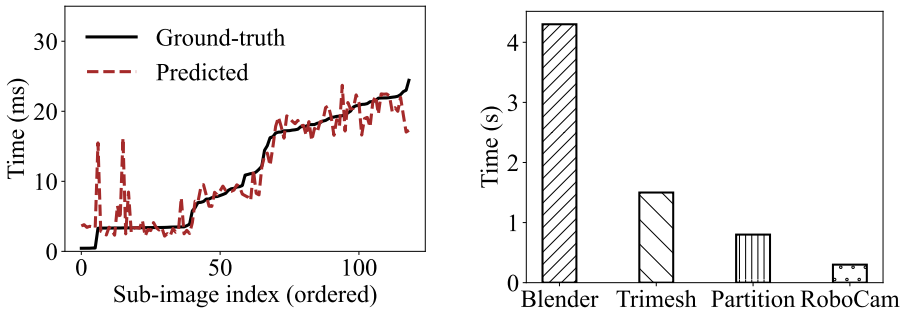
We compare our model-based PnP approach with three baseline approaches, including the no-calibration, manual PnP, and visual-inertial optimization (VIO). The no-calibration uses the planned pose P_k to render the reference image, instead of the estimated pose. In the manual PnP, we perform human eye-tracking to obtain eight 2D-3D corresponding pairs as inputs for the PnP algorithm to calculate the camera's pose. In the VIO method, since the camera and robotic arm do not include an integrated IMU, we attach an external IMU sensor [28] to the camera. We formulate the camera pose estimation as a camera registration problem, solved using a least-squares method, as is typical in study [33], using sequential external IMU sensor data (acceleration and angular velocity) and camera frames as inputs to estimate the camera's movement trajectory. Fig. 11 shows the average 2D mask error of three approaches for estimating the poses of ten Sample B's images with a resolution of 2104×1560 pixels, captured in ten different poses. From Fig. 11, the model-based and manual PnP approaches have the similar mask error of around 1.9% and 1.4%, respectively, while the no-calibration has much higher mask error of about 40%. These results show that our model-based PnP achieves similar performance as the manual PnP. However, the manual PnP requires higher labor costs due to the use of the human eye-tracking for identifying the 2D-3D corresponding pairs. As shown in Fig. 11, the VIO method exhibits higher camera pose estimation error compared with other three approaches. This is because the IMU may introduce noise and cumulative errors with movement, which negatively affect the camera pose estimation. The inertial data can reduce robotic system errors when the environment or the object's properties cause significant variations in visual data during pose estimation. However, the VIO method requires high-quality inertial data to improve camera registration. The IMU sensor typically exhibits errors at the level of tens of centimeters [39], which is similar to the movement scale in our application (sub-meter). However, this level of accuracy may not be sufficient for our application scenarios.

5.2.2 Two-domain rectification. In this section, we evaluate the performance of our two-domain rectification approach for inspecting a Sample A's curvy mesh screen in multiple iterations. We compare RoboCam with its variant, called *digital-rectification-only* which does not rectify the physical camera pose. For each inspection iteration, it only captures the image m_{k1} and uses the estimated pose of the m_{k1} for the image rendering. Figs. 12(a) and (b) show the distance error between the planned and captured poses, and the overlap rate between the captured image and the selected region w_k, respectively, achieved by the digital-rectification-only and two-domain rectification. In particular, the overlap rate is defined as a ratio of the number of pixels of the w_k covered by the captured image to the total number of pixels of w_k. Fig. 12(a) shows that our proposed two-domain rectification



(a) Pose distance errors. (b) Overlap rate.

Fig. 12. Distance errors and overlap rate over six iterations.



(a) Prediction of rendering latency. (b) Rendering latency.

Fig. 13. Image rendering performance. In (a), the prediction root mean square error (RMSE) between the ground-truth and predicted latencies of 120 testing sub-images is 2.1 ms.

approach can achieve a 3x distance error reduction, compared with digital-rectification-only. From Fig. 12(b), we can see that with the smaller distance error, our approach achieves a 1.4x average overlap rate improvement.

5.2.3 Parallel image rendering. We evaluate our greedy-based parallel image rendering approach. Specifically, we create a training dataset of 400 mesh screen sub-images (100×100 pixels) to fit the quadratic polynomial model offline, which takes the contrast, brightness, and number of object edges of the sub-image to estimate the image rendering latency. Fig. 13(a) shows the ground-truth and predicted latencies of 120 testing sub-images. The predictions track the ground truths well. These results imply that our model can predict the image rendering latency with high accuracy.

We also compare our parallel image rendering approaches with the three baseline approaches including the Blender, Trimesh, and partition approaches. Specifically, the Blender [1, 22] is a widely adopted open-source 3D computer graphics software, while the Trimesh is an image rendering function available in the python library Trimesh. We utilize Blender's python API, along with the BSDF shader and the Eevee engine, for rendering execution. The partition approach aims to assign the similar number of the sub-images to the workers without considering the rendering latency. We create 24 processes in the workstation to serve as the workers and evaluate the performance by six reference images. Fig. 13(b) shows the averaged latencies of the three approaches for

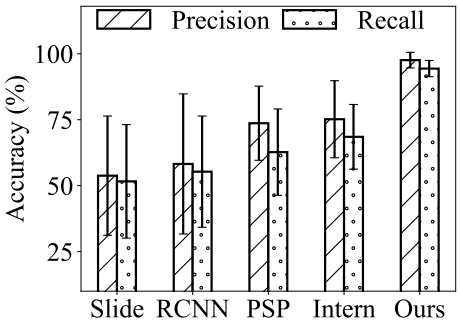


Fig. 14. The precision and recall over five mesh screens.

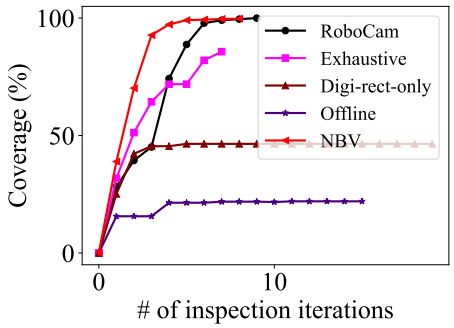


Fig. 15. End-to-End inspection coverage and latency.

rendering six Sample A's mesh screen images with a resolution of 2104 × 1560. Each image is divided into 100 sub-images which are assigned to 24 workers (i.e., the CPU cores). From Fig. 13(b), the Blender approach has the highest rendering latency time since it fully renders the geometry and complex lighting information. The partition approach has lower latency than the Trimesh approach due to the exploitation of the parallel rendering. Meanwhile, RoboCam achieves a low latency of around 0.3 seconds, making it 10x faster in image rendering than Blender.

5.2.4 Reference-based inspection. In this section, we compare RoboCam's reference-based inspection approach with the following four approaches. (1) *Sliding window* is a computer vision-based defect inspection approach proposed in [46]. It applies the image threshold processing and sliding window with a size of 10×10 pixels to scan the images line by line. Then, the pores located in a window are classified as clogged if the window has more black 0-pixels than white 1-pixels. (2) *Mask R-CNN* proposed in [48] employs R-CNN to extract the bounding boxes of the clogged pores. Then, it uses a fully connected layer to segment out the clogged pores from the extracted boxes. (3) *PSPNet* is an industrial defect inspection approach proposed in [15], which employs the deep image segmentation framework, called PSPNet, to segment out the clogged pore regions. Both learning-based approaches (i.e., the Mask R-CNN and PSPNet) use a ResNet50 as the backbone CNN model. The backbone is pre-trained using the widely used image object dataset COCO [2]. (4) *InternImage* is a state-of-the-art image segmentation model pre-trained using the ADE20K dataset [8], utilizing the deformable convolution operators [45]. We re-train the models for clogged pore detection in our application, using a mix of real and synthesized images. Specifically, 200 synthetic images (2104 x 1560 pixels), with a random distribution of clogged pores, are generated from a CAD model. Additionally, we collect and manually label 40 real images from mesh screens of Samples A and B, categorizing each pixel as clogged or unclogged. Moreover, we employ the precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$) rates as the clogged pore inspection accuracy metrics, where TP, FP, and FN represent the numbers of true positives, false positives and false negatives, respectively.

Fig. 14 presents the average and standard deviation of the precision and recall rates of three baseline and RoboCam approaches in inspecting five mesh screens of the Samples A and B. RoboCam achieves the highest average precision and recall rates with the lowest standard deviations among the approaches. For instance, RoboCam has the precision and recall rates of 97.59% and 94.37%, respectively, which are 1.32x and 1.51x higher than those of the PSPNet approach. Moreover, RoboCam achieves 1.29x and 1.37x higher precision and recall, respectively, compared with the InternImage model. This is because the image segmentation might not accurately segment out clustered clogging with cluster patterns unseen in the training dataset. However, our approach directly compares the actual pores in the captured image with the designed pores in the rendered image to accurately detect the clogged pores.

Table 1. End-to-end accuracy

Method	Accuracy	
	Precision (%)	Recall (%)
RoboCam (Aggregation + Voting)	92.35	94.95
Union	85.17	79.29

5.3 Overall Performance

5.3.1 End-to-end coverage. We assess RoboCam's end-to-end inspection coverage and latency on a Sample A's curvy mesh screen which sizes $3.5 \times 6.5\text{cm}^2$ and consists of 2,287 pores. We compare RoboCam with its variant (i.e., the digital-rectification-only approach) and the following three baselines. (1) *Exhaustive search* moves the robotic arm in a grid search manner from bottom to top with a 1-cm interval to cover the target mesh screen. (2) *Offline plan* divides the mesh screen into 15 equal-sized blocks, and defines the camera pose for each block. (3) *NBV* [14] selects the next pose by choosing the one with the largest inspectable region from all possible poses. Specifically, the NBV method generates a candidate pose set P_c by generating one pose to each pixel in the uninspected region of the 2D image plane C. The NBV method uses a brute-force approach to select the camera pose with the largest inspectable region as the next inspection pose, continuing this process until full coverage is achieved.

The above approaches iteratively inspect the mesh screen via multiple iterations. We define the inspection coverage by the ratio of the number of the inspected pixels to the total number of pixels in the whole 2D image plane C. Fig. 15 shows the coverage of RoboCam, exhaustive search, offline plan, and digital-rectification-only approaches along the iterations. Moreover, RoboCam has the highest coverage among the four approaches. For instance, after six iterations, RoboCam achieves 98% coverage, while the exhaustive search achieves 82.6% coverage. Specifically, RoboCam's variant (i.e., the digital-rectification-only approach) reaches the maximum coverage of about 41% after three iterations. Then, its coverage remains the same along the iterations. This result indicates that with the digital-rectification-only, the mesh screen may not be fully inspected even with more iterations. The NBV method can achieve full coverage in a smaller number of iterations than RoboCam. For example, NBV requires only 4 inspection iterations to reach 97.5% coverage, compared with RoboCam's 7 iterations. However, as illustrated in Fig. 16, the time overhead for a single NBV iteration is nearly 21 hours, whereas a single RoboCam iteration takes only 0.35 seconds. This is because the NBV method operates in a large search space in our scenario, with size to 222,224. Note that in our scenario, the uninspected region C of a mesh screen with dimensions 3.5×6.5 cm^2 contains 222,224 pixels. Differently, in the existing study [14], which uses NBV, the search space size is only 15. As such, the end-to-end latency reported in [14] is only 0.153 seconds. These results show that RoboCam's pose planning is effective in reducing the inspection time.

5.3.2 End-to-end accuracy. Table 1 shows the end-to-end precision and recall rates of RoboCam in detecting the clogged pores of the above Sample A's mesh screen. We can see that RoboCam can achieve 92.35% precision and 94.95% recall. We also compare RoboCam with a baseline approach called Union which merges the iterations' inspection results without removal of repeated results. From Table 1, RoboCam achieves higher precision and recall rates than those of the Union since the removal of repeatedly inspected pore results helps reduce the number of false positives and false negatives in detecting clogged pores in overlapped inspectable regions. We further obtain the precision-recall curve by adjusting the threshold value for the extraction of pores' centroids from the captured images. As shown in Fig. 17, the precision-recall curve indicates that the clogged pores inspection model achieves high precision (i.e., 0.8923 to 1) over a wide range of recall values from 0 to 0.9553.

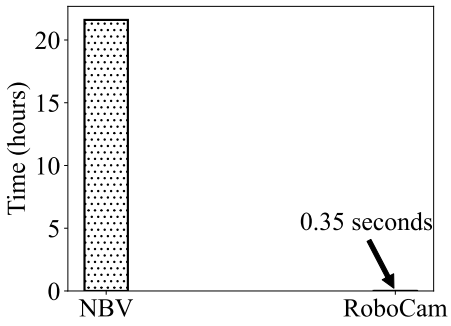


Fig. 16. NBV method performance compared with RoboCam.

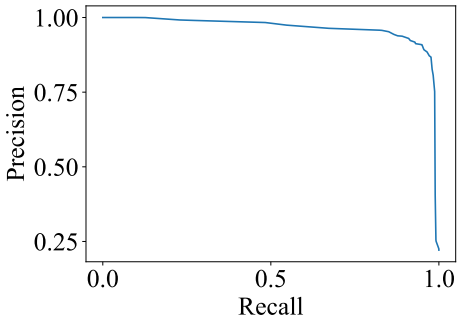


Fig. 17. Precision-recall curve.

5.3.3 End-to-end latency. We measure the end-to-end latency of RoboCam which includes the latencies of the individual processes including the computation, data transmission, and robotic arm moving. Specifically, RoboCam sends image data to the backend computing workstation using 510Mb/s Wi-Fi. Our measurement results show that RoboCam spends about a total latency of 528 seconds to fully inspect the entire mold Sample A in 33 iterations. For Sample B, one iteration can cover the full flat mesh screen, which has a latency of about 16 seconds. Furthermore, we also study how the mesh screen size affects the inspection latency of RoboCam. Specifically, we select an initial square region of the Sample A's mesh screen with a size of 7cm^2 for the pore inspection. Then, we expand the initial region to create larger regions with the size up to 23cm^2 and repeat the inspections. Fig. 18 presents the total inspection latency under the different mesh screen sizes. From Fig. 18, the latency generally increases with the mesh screen size. This is because RoboCam needs more iterations to inspect larger mesh screen regions, resulting in higher end-to-end latency. Moreover, the reason of the non-linear increase of the latency is that the mesh screen regions with a different size may have different curvy property.

5.4 Pore size measurements

In this paper, the design of RoboCam mainly focuses on clogged pore detection, which is a key inspection task for ensuring the quality of the mesh screen products. We provide an additional function to perform the automated fine-grained pore size measurement, which can be essential in certain 3D-printed mesh screen manufacturing systems. For instance, the pore size of the mesh screens used for the oil and water separation greatly affects the oil absorption efficiency [29]. Such a size measurement task is beyond the capability of manual inspection, due to the small size and high density of the mesh screen pores.

We employ an image processing algorithm to extract pore contours within the inspectable region of the captured images. The pore sizes are determined by counting the number of pixels within each contour, averaging 10 pixels. We then convert the pore size from pixels to area units using a pixel-to-area conversion baseline. This baseline is obtained by capturing an image of a ruler under the same camera settings, ensuring accurate conversion. However, as the robotic arm continuously moves the camera, the pixel-to-area conversion may vary. To address this, we utilize the area unit information from the CAD model to obtain a pixel-to-area conversion baseline for each captured image. Specifically, we calculate the baseline by comparing the area size of the inspectable region in the CAD model with its pixel size in the captured image. This conversion is then applied to convert the pore sizes from pixels to area units.

We further evaluate the performance of pore size measurements. A pre-calibrated USB digital microscope is utilized for manual measurement of each pore's size to validate the accuracy of pore size measurements. Fig. 19 presents the scatter plot comparing the ground-truth pore sizes with the measured sizes of 62 pores, illustrating a

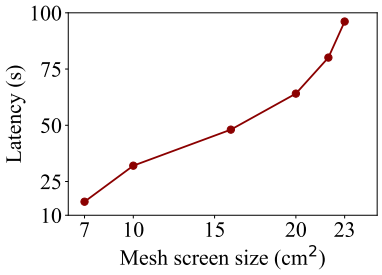


Fig. 18. Inspection latency.

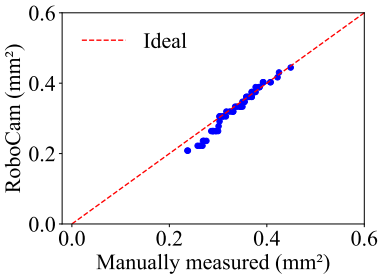


Fig. 19. Pore size measurement accuracy (RMSE=0.017).

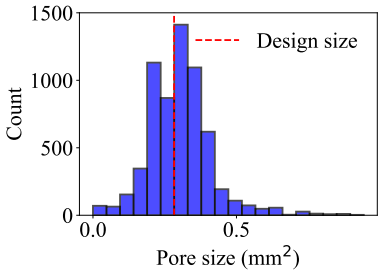


Fig. 20. Pore size distribution.

close match with a root mean square error of 0.017. This result shows that our method is capable of precisely measuring pore sizes at a sub-millimeter scale. Fig. 20 presents a histogram of the pore size distribution for Sample A's mesh screens. Fig. 20 shows that most pore sizes are close to 0.283mm^2, the designed pore size of the manufactured molds. These measurement results can be useful for the further improvement of the 3D printer's configuration.

6 CONCLUSION

This paper presents RoboCam, a model-based robotic visual sensing system designed for sub-millimeter pores inspection on the 3D-printed mesh screens. To inspect sub-millimeter, densely, and non-uniform pores, RoboCam compares captured images with rendered images to detect clogged pores. To fully inspect the customized curved mesh screen, RoboCam uses a robotic arm to move the camera along computed waypoints. With the proposed rectification approaches in both the physical and digital domains, the inconsistencies between the rendered and captured images are rectified at run time. This addresses a key challenge caused by the inherent movement errors of the robotic system. RoboCam achieves a recall rate of 94.95% within 528 seconds latency for inspecting an entire mesh screen mold with 13,000 designed pores. It also outperforms a CNN-based approach by more than 1.51x in terms of inspection accuracy.

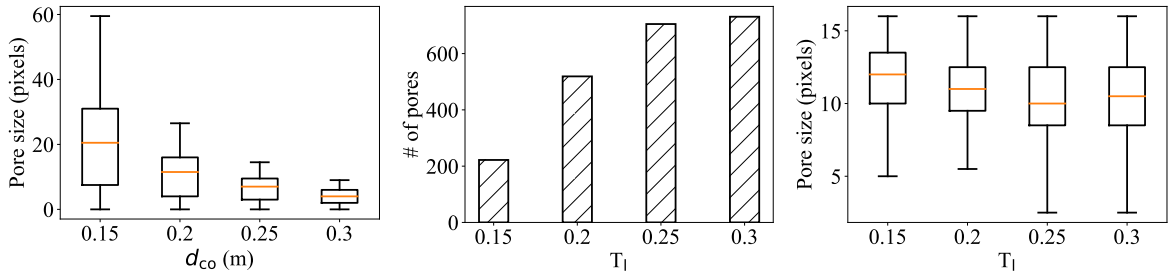
ACKNOWLEDGMENTS

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner, HP Inc., through the HP-NTU Digital Manufacturing Corporate Lab.

REFERENCES

[1] [n. d.]. Blender. https://www.blender.org/
[2] [n. d.]. COCO datasets. https://cocodataset.org/home
[3] [n. d.]. HP and AM Solutions Present the Innovative HP Jet Fusion 5200 Series 3D Automatic Unpacking Station. https://shorturl.at/ovzPS
[4] [n. d.]. Molded fiber. https://www.hp.com/usen/printers/3d-printers/molded-fiber-tooling.html
[5] [n. d.]. Robotic arm. https://www.universal-robots.com/
[6] [n. d.]. See3cam camera. https://www.e-consystems.com/13mp-autofocus-usb-camera.asp
[7] [n. d.]. X-ray sensors. https://shorturl.at/uvzPY
[8] ADE20k. 2024. https://groups.csail.mit.edu/vision/datasets/ADE20K/
[9] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 2007. 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Des. Appl.* 4, 6 (2007).
[10] M.Grimes A.Kendall and R.Cipolla. 2015. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*.

[11] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. 2011. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Pattern Analysis Machine Intell.* 34, 7 (2011).
[12] A. Appel. 1968. Some techniques for shading machine renderings of solids. In *Spring Joint Computer Conference*.
[13] A. K. Bedaka, S.-C. Lee, A. M. Mahmoud, Y.-S. Cheng, and C.-Y. Lin. 2021. A camera-based position correction system for autonomous production line inspection. *Sensors* 21, 12 (2021).
[14] Andreas Bircher, Mina Kamel, Kostas Alexis, Helen Oleynikova, and Roland Siegwart. 2016. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*. 1462–1468.
[15] J. Chen, D. Van Le, R. Tan, and D. Ho. 2024. A Collaborative Visual Sensing System for Precise Quality Inspection at Manufacturing Lines. *ACM Transactions on Cyber-Physical Systems (TCPS)* (2024).
[16] Corners. 2024. https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a
[17] W. Du, H. Shen, and J. Fu. 2020. Automatic defect segmentation in X-ray images based on deep learning. *IEEE Trans. Ind. Electron.* 68, 12 (2020).
[18] M. A. Fischler and R. C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981).
[19] Y. Fridman, M. Rusanovsky, and G. Oren. 2021. ChangeChip: A reference-based unsupervised change detection for PCB defect detection. In *IEEE Physical Assurance Inspection Electron.*
[20] R. Fukasawa. 2015. Terahertz imaging: Widespread industrial application in non-destructive inspection and chemical analysis. *IEEE Trans. Terahertz Sci. Technol.* 5, 6 (2015).
[21] H. Germain, V. Lepetit, and G. Bourmaud. 2021. Neural reprojection error: Merging feature learning and camera pose estimation. In *CVPR*.
[22] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, and C. et al. Herrmann. 2022. Kubric: A scalable dataset generator. In *CVPR*.
[23] H. Ha and J. Jeong. 2021. CNN-based defect inspection for injection molding using edge computing and industrial IoT systems. *Appl. Sci.* 11, 14 (2021).
[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. 2017. Mask R-CNN. In *Proceedings of the International Conference on Computer Vision (ICCV)*.
[25] T. Hochrein. 2015. Markets, availability, notice, and technical performance of terahertz systems: Historic development, present, and trends. *Journal of Infrared, Millimeter, and Terahertz Waves* 36 (2015).
[26] W. Huang and R. Kovacevic. 2011. A laser-based vision system for weld quality inspection. *Sensors* 11, 1 (2011).
[27] S. Huo, D. Navarro-Alarcon, and D. T. W. Chik. 2021. A robotic defect inspection system for free-form specular surfaces. In *ICRA*.
[28] IMU. 2024. https://docs.arduino.cc/hardware/nano-33-ble/
[29] J. J. Koh, G. J. H. Lim, X. Zhou, X. Zhang, J. Ding, and C. He. 2019. 3D-printed anti-fouling cellulose mesh for highly efficient oil/water separation applications. *ACS Appl. Materials Interfaces* 11, 14 (2019).
[30] H. W. Kuhn. 2005. The Hungarian method for the assignment problem. *Naval Research Logistics* 2, 1-2 (2005).
[31] Duc Van Le, Joy Qiping Yang, Siyuan Zhou, Daren Ho, and Rui Tan. 2023. Design, deployment, and evaluation of an industrial AIoT system for quality control at HP factories. *ACM Transactions on Sensor Networks (TOSN)* 20, 1 (2023).
[32] C. de Leo and B. S. Manjunath. 2014. Multicamera video summarization and anomaly detection from activity motifs. *ACM Transactions on Sensor Networks (TOSN)* 10, 2 (2014).
[33] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. 2015. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research* 34, 3 (2015), 314–334.
[34] W.-L. Mao, Y.-Y. Chiu, B.-H. Lin, C.-C. Wang, Y.-T. Wu, C.-Y. You, and Y.-R. Chien. 2022. Integration of Deep Learning Network and Robot Arm System for Rim Defect Inspection Application. *Sensors* 22 (2022).
[35] Gaku Nakano. 2016. A versatile approach for solving PnP, PnPf, and PnPfr problems. In *ECCV*. 338–352.
[36] C. H. P. Nguyen and Y. Choi. 2018. Comparison of point cloud data and 3D CAD data for on-site dimensional inspection of industrial plant piping systems. *Automation in Construction* 91 (2018).
[37] A. Niini, P. Tanninen, V. Leminen, I. Jönkkäri, and M. Horttanainen. 2022. Press-forming Molded Pulp from Repulped Liquid Packaging Board: Role of Heat Input, Pressing Force, and Defect Formation. *BioResources* 17, 4 (2022).
[38] S. Niu, B. Li, X. Wang, and Y. Peng. 2021. Region-and strength-controllable GAN for defect generation and segmentation in industrial images. *IEEE Trans. Ind. Informatics* 18, 7 (2021).
[39] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE transactions on robotics* 34, 4 (2018), 1004–1020.
[40] V. Usenko, N. Demmel, and D. Cremers. 2018. The double sphere camera model. In *International Conference on 3D Vision*.
[41] Eugene Valassakis, Kamil Dreczkowski, and Edward Johns. 2022. Learning eye-in-hand camera calibration from a single image. In *Conference on Robot Learning*. PMLR, 1336–1346.



(a) Pores size (pixels) vs. d_{co}. (b) # of pores vs. T_l. (c) Pores size (pixels) vs. T_l.

Fig. 21. Impact of the camera-object distance d_{co} and T_l settings on the pore quantity and size. With $d_{co} < 0.2$m, the model-based PnP fails to estimate the camera poses of the captured images.

[42] M. Vieira and K. Shimada. 2005. Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Des.* 22, 8 (2005).
[43] R. K. W. Vithanage, E. Mohseni, Z. Qiu, C. MacLeod, Y. Javadi, N. Sweeney, G. Pierce, and A. Gachagan. 2020. A phased array ultrasound roller probe for automated in-process/interpass inspection of multipass welds. *IEEE Trans. Ind. Electron.* 68, 12 (2020).
[44] H. Wang, Z. Shi, Y. Qiao, F. Yang, Y. He, D. Xuan, and W. Zhao. 2023. Autonomous and Cost-effective Defect Detection System for Molded Pulp Products. In *ICCPS*.
[45] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, et al. 2023. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14408–14419.
[46] Y. Wen, K. Fu, Y. Li, and Y. Zhang. 2021. A sliding window method to identify defects in 3D printing lattice structure based on the difference principle. *Meas. Sci. Technol.* 32, 6 (2021).
[47] Jin Wu, Yuxiang Sun, Miaomiao Wang, and Ming Liu. 2019. Hand-eye calibration: 4-D procrustes analysis approach. *IEEE Transactions on Instrumentation and Measurement* 69, 6 (2019), 2966–2981.
[48] L. Xiao, B. Wu, and Y. Hu. 2020. Surface defect detection using image pyramid. *Sensors* 20, 13 (2020).
[49] Z. Zeng, B. Liu, J. Fu, and H. Chao. 2021. Reference-based defect detection network. *IEEE Trans. Image Process.* 30 (2021).
[50] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. 2017. Pyramid scene parsing network. In *CVPR*.

A APPENDIX

A.1 Setting of d_{co}

The camera-object distance d_{co} affects the size of the mesh screen pores appeared in the captured and rendered images. Fig. 21(a) shows the distributions of the pore size in the rendered images of a flat mesh screen under various settings of d_{co}. Note that we employ a CV-based contour extraction approach on the rendered images to extract the pore regions, and then count the number of pixels inside each extracted region as the pore size. From Fig. 21(a), the pore size decreases with d_{co}. In general, the pores with a larger size are easier to inspect. Thus, a lower value of d_{co} is desirable to achieve high inspection accuracy. However, with $d_{co} < 0.2$m, the proposed model-based PnP approach (cf. §4.2.1) fails to estimate the camera poses of the captured images. This is because the mesh screen contours may fall outside the camera's field of view when the camera maintains a too close distance from the mesh screen. As a result, the 2D-3D corresponding pairs can not be obtained for the camera pose estimation. Thus, RoboCam adopts the d_{co} of 0.2m which leads to a good balance between the pore size (i.e., inspection accuracy) and the pose estimation accuracy.

A.2 Setting of T_l

The setting of T_l affects the size of the inspectable region which is selected for detecting the clogged pores in each inspection iteration. Fig. 21(b) and Fig. 21(c) show the quantity and size distribution of the pores in the inspectable regions of a curvy mesh screen under various settings of T_l. From Fig. 21(b), the number of pores increases with T_l. This is because with a higher value of T_l, more CAD model faces are selected to form the inspectable region with more pores. Thus, a higher value of T_l may help reduce the end-to-end inspection latency because the larger mesh screen regions are inspected in the iterations. Meanwhile, from Fig. 21(c), the distribution range of the pore sizes varies under different settings of T_l. The mesh screen's CAD model is designed to generate the pores with an identical size. Thus, the pore sizes of the printed mesh screen should be concentrated in a narrow range. A larger distribution range of the pore size indicates that the selected regions may include more pores which appear distorted on the captured image. Inspecting such distorted pores may reduce the recall rate in detecting the clogged pores. Therefore, the setting of T_l should be chosen to achieve a narrow distribution range of the pore size. For RoboCam, we set the T_l to 0.2 which leads to a good balance between the number of the inspected pores and their distribution size range.

A.3 Notation

Notation in this paper is summarized in Table 2.

Received 8 February 2024; revised 29 September 2024; accepted 21 Jan 2025

Table 2. Notation

Notation	Definition
S	the constant transformation error of robotic system
P_k	the planned camera pose in iteration k
P_o	an initial camera pose for dynamic movement error profiling
P_m	the m^{th} target camera pose for dynamic movement error profiling
$\tilde{P}_m$	the estimated camera pose from the image captured in P_m
d_m	the euclidean distance between P_m and $\tilde{P}_m$
C_m	the cosine distance between P_m and $\tilde{P}_m$
P_{ij}	the j^{th} target camera pose for i^{th} initial camera pose for investigating difference between the movement distance errors
d^i_{jk}	the distance between P_{ij} and P_{ik} for a given i
w_k	the largest uninspected mesh screen region before iteration k
C	the projected 2D image plane of w_k
K	the camera's intrinsic parameters
W	the window size used for dividing C
d_{co}	the camera-object distance
P_{k1}	the actual camera pose when the robotic system moves to the planned pose P_k
m_k	the rendered image based on the planned pose P_k
m_{k1}	the image captured at P_{k1}
$\Phi_{\mathrm{2D-2D}}$	the set of 2D-2D corresponding pairs between m_k and m_{k1}
P^{r}_k	the rectified camera pose based on P_{k1}
T_{Pk1}	the movement error when the robotic system moves to P_k
$\tilde{P}_{k1}$	the estimated pose of the image m_{k1}
$\tilde{T}_{Pk1}$	the estimated movement error when the robotic system moves to P_k
m_{k2}	the captured image when the robotic system moves to P^{r}_k
Υ	a quadratic polynomial model estimates sub-image rendering latency
n	the number of workers for parallel sub-image rendering
$\boldsymbol{f}_{\mathrm{u}}$	the camera pose vector for image m_{k2}
$\mathbf{S}_{\mathrm{o}}$	the CAD model face set
s_i	the i^{th} face of the $\mathbf{S}_{\mathrm{o}}$
$\boldsymbol{u}_i$	the normal vector of s_i
$\boldsymbol{S}$	the set of selected faces for the inspectable region
T_{l}	a threshold value for the selection of $\boldsymbol{S}$
M	the number of rendered image pores
N	the number of captured image pores
$\mathcal{P}_{\mathrm{r}}$	the set of rendered image pores' positions
$\mathcal{P}_{\mathrm{c}}$	the set of captured image pores' positions