Automatic Camera Pose Estimation by Key-Point Matching of Reference Objects

Zeng, Jinchen ; Butler, Rick; van den Dobbelsteen, John J.; Hendriks, Benno H.W.; van der Elst, Maarten; Dauwels, Justin

DOI
10.1109/ICASSP49357.2023.10095197

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10095197

AUTOMATIC CAMERA POSE ESTIMATION BY KEY-POINT MATCHING OF REFERENCE OBJECTS

Jinchen Zeng[⋆] *Rick Butler*[⋆] *John J. van den Dobbelsteen*[⋆] *Benno H. W. Hendriks*[⋆†]
Maarten Van der Elst[⋆§] *Justin Dauwels*[⋆]

[⋆]Delft University of Technology, Delft, The Netherlands
[†]Philips Research Laboratories, Eindhoven, The Netherlands
[§]Reinier de Graaf Groep, Delft, The Netherlands

ABSTRACT

In this paper, we aim to design an automatic camera pose estimation pipeline for clinical spaces such as catheterization laboratories. Our proposed pipeline exploits Scaled-YOLOv4 to detect fixed objects. We adopt the self-supervised key-point detector SuperPoint in combination with SuperGlue, a key-point matching technique based on graph neural networks. Thus, we match key-points on input images with annotated reference points. Reference points are chosen on fixed objects in the scene, such as corners of door posts or windows. The point-correspondences between the image coordinates and the 3D coordinates are applied to the Perspective-n-Point algorithm to estimate the pose of each camera. Compared with other camera pose estimation methods, the proposed pipeline does not require the construction of 3D point-cloud model of the scene or placing a polyhedron object in the scene before each required calibration. Using videos from real procedures, we show that the pipeline can estimate the camera pose with high accuracy.

Index Terms— Camera calibration, camera pose estimation, Perspective-n-Point, 3D geometry

1. INTRODUCTION

The operating room (OR) is constantly evolving with the introduction of new technologies and surgical procedures. One of the key areas of the improvement is surgical workflow analysis [1, 2], which affects patient safety, working conditions, and hospital efficiency. The 3D position & poses of personnel are descriptive of the surgical workflow, while camera extrinsics calibration is necessary to extract 3D information from 2D footages [3, 4, 5]. However, in dynamic environments like catheterization laboratories, camera poses can be affected by personnel and equipment movement, resulting in inaccurate localization. It is thus crucial to calibrate the camera regularly to ensure reliable results.

Our study focuses on camera pose estimation (a.k.a. extrinsic calibration of the camera), which involves estimating the orientation and position of the camera. Traditional methods use 3D-2D correspondences between 3D points and image pixels [6, 7, 8], while deep learning-based methods often use convolutional neural networks (CNNs) to regress the camera pose [9, 10, 11, 12]. However, traditional methods require calibration patterns to be placed in the scene, while deep learning-based methods need a large dataset of annotated images and poses, which can require a labor-intensive process.

Therefore, we aim to automate camera pose estimation for clinical spaces like ORs and catheterization laboratories. In Fig. 1, we propose an automatic camera pose estimation pipeline. We combine the object detection model Scaled-YOLOv4 [13], a neural-network based key-point detector SuperPoint [14] and key-point matcher SuperGlue [15] to obtain the 3D-2D correspondences between the corners of the fixed objects in the scene and image pixels for each view individually. Next we apply EPnP [6] algorithm to solve the Perspective-n-Point problem [8], resulting in the extrinsic parameters of the camera. The proposed pipeline achieves the lowest 5.79 pixel reprojection error and the lowest 3.28 cm Euclidean distance error averaged on 27 key-points. In summary, our proposed pipeline has two main contributions. Firstly, it employs advanced neural networks to achieve high accuracy and efficiency. It eliminates the need for constructing a 3D point-cloud of the scene or manually placing calibration patterns, significantly reducing the required time and resources. Secondly, it can be easily deployed to other catheterization laboratories and other indoor localization applications.

2. BACKGROUND

Camera calibration is the estimation of camera parameters including intrinsics, extrinsics and distortion coefficients [8]. Intrinsic- or internal parameters mainly represent the inherent properties of the camera including the focal length, the optical center, and the skew parameter. Extrinsic- or external parameters represent the camera rotation and translation. The camera projection matrix P maps a point $\mathbf{X} \in \mathbb{R}^3$ in world

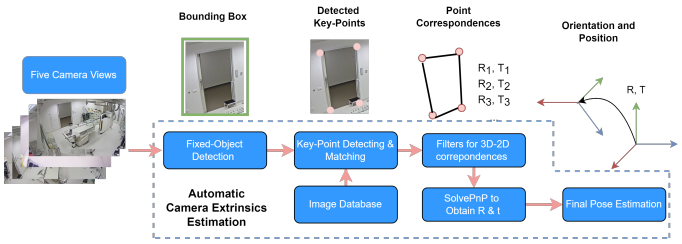


Fig. 1: Automatic camera pose estimation pipeline

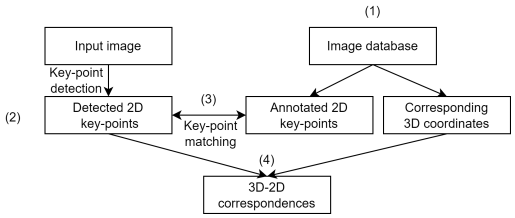


Fig. 2: The steps of key-point detection and matching with the reference points stored in the image database.

coordinates to a point $\mathbf{x} \in \mathbb{R}^2$ in image coordinates. This mapping and the projection matrix are written as $\mathbf{x} = P\mathbf{X}$, $P = K[R|t]$, where the matrix K is the camera projection matrix, also known as the intrinsics, and R and t represent the camera rotation and translation. K is expressed as:

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

where the focal length is (f_x, f_y), the optical center (c_x, c_y) and the skew coefficient s. $[R|t]$ represents the rigid transformation from the world coordinate to the camera coordinate. The transformation is done with a rotation matrix $R \in \mathbb{R}^{3\times3}$ and a translation vector translation vector $t \in \mathbb{R}^{3\times1}$. Together, they form the extrinsic matrix $[R|t] \in \mathbb{R}^{3\times4}$.

The camera distortion is usually represented as radial distortion and tangential distortion. Radial distortion occurs when the light ray bends more at the edges of the lens than in the center, which causes straight lines near the edge of the image to appear curved. This can be modelled as: $(x_{\text{distorted}}, y_{\text{distorted}}) = (x(1 + k_1r^2 + k_2r^4 + k_3r^6), y(1 + k_1r^2 + k_2r^4 + k_3r^6))$, where x, y represent the undistorted pixel coordinates. x and y are (normalized) pixel coordinates. $r^2 = x^2 + y^2$. k_1, k_2, and k_3 are radial distortion coefficients.

Tangential distortion occurs when the lens and the image plane are not aligned. It is modelled as: $(x_{\text{distorted}}, y_{\text{distorted}}) = (x + [2p_1xy + p_2(r^2 + 2x^2)], y(1 + k_1r^2 + k_2r^4 + k_3r^6))$, where p_1, p_2 are tangential distortion coefficients.

In our study, we have retrieved the intrinsic parameters and distortion coefficients $(k_1,\ k_2,\ p_1,\ p_2,\ k_3)$ by a chessboard calibration pattern after installation of the cameras [16]. This paper is about estimating the extrinsics $[R|t]$ in an automated manner from reference objects in the visual scene.

3. METHODS

The proposed pipeline mainly relies on obtaining 3D-2D correspondences between the reference points of fixed objects in the scene and the image pixel coordinates. The fixed objects include windows, doors, patslide, working bench, lights, and the monitor screen. As shown in Fig. 1, the proposed pipeline starts with the detection of the fixed objects in the scene. We create a image database consisting of the annotated key-points and 3D measurements of the key-points on the reference objects. Then we apply a key-point detection and matching technique to extract the 3D-2D correspondences between 2D image key-points and 3D coordinates annotated in the database images. After filtering the key-point matches, we apply the EPnP algorithm [6] to the resulting 3D-2D correspondences to solve Perspective-n-Point problem and obtain the orientation and position of the camera. In the following, we explain each of those steps.

3.1. Fixed Object Detection

Object detectors seek to recognize instances of a series of predefined object classes from given images or videos, and estimate the 2D positions of each detected object with a bounding box. We adopt the one-stage object detector Scaled-YOLOv4 [13], known for its fast speed and high accuracy. In [13], it is shown that Scaled-YOLOv4 outperforms other state-of-the-art object detectors in both speed and accuracy. In our scenario, we need a high-accuracy object detector and Scaled-YOLOv4 suits our application.

3.2. Key-Point Detection & Matching

To obtain the 3D-2D correspondence pairs, we implement the following steps (see in Fig. 2): (1) Create an image database that contains the annotated key-point coordinates with the corresponding measured 3D coordinates; (2) Detect key-points on input images by SuperPoint [14], a self-supervised framework specializing in tackling multiple-view geometry problems in 3D computer vision field; (3) Match the annotated key-points in database images with the input images by SuperGlue [15], a key-point matcher that learns priors over geometric transformations and 3D world regularities through end-to-end training from picture pairs; (4) Obtain 3D-2D correspondences between the measured 3D coordinates on database images and the 2D key-points on input images.

The image database consists of images taken from multiple angles. Each database image contain at least one fixed-object. Object key-points—the 3D positions of which are known—are annotated in the 2D images. We annotate in CVAT [17] the corners or vertices of fixed objects as the key-points, because their 3D coordinates are easy to measure.

3.3. Filtering

The obtained key-point matches can be wrong due to the complicated scene settings and the wrong inferences by SuperPoint or SuperGlue. Therefore, we integrate the bounding boxes inferred by Scaled-YOLOv4 as our first filter block to remove wrongly matched key-points. If the coordinate of the key-point is not inside of a bounding box with a corresponding label, we remove the key-point. Since one input image will be matched to multiple database images, the single label on the input image is likely to be matched to multiple slightly different coordinates. Thus, we choose either of the following two filtering algorithms as the second filter block: Nearest Centroid Distance (NCD) and Reprojection Error Minimization (RPEM). If matched key-points of one label suggest different 3D coordinates, NCD calculates the centroid of the resulting set of 3D coordinates, next selects the key-point that has the shortest Euclidean distance to this centroid and removes the others. Given M coordinates matched to the same key-point label, we denote the coordinates as $v_1, v_2, ..., v_M$, $v_n = (x_n, y_n)$. NCD finds and selects the key-point v_{selected} which is closest to the centroid $v_{\text{selected}} = \arg\min \|v_i - \sum_{i=1}^{M} v_i/M\|_2, i = 1, 2, ..., M$.

RPEM is designed to minimize the reprojection error (see Section 3.6). Although multiple key-points can be matched with the same label, we initialize with 4 key-points that have the highest confidence scores. Based on the rank of confidence scores, we add another key-point correspondence pair at a time and calculate the reprojection error. Consequently, we find the suboptimal combination of key-points that generates the minimized reprojection error.

At last, we apply the EPnP algorithm on the filtered 3D-2D correspondences to estimate the camera pose.

3.4. Camera Pose Estimation

The Perspective-n-Point (PnP) refers to estimating the orientation and position of a camera, given n correspondences between 3D world coordinates and projected 2D image points [8]. Popular algorithms to solve the PnP problem include Direct Linear Transform (DLT) [8], P3P [7], EPnP [6], and Bundle Adjustment (BA) [8]. We adopt EPnP because of its robustness and better performance during our testing. Filtered 3D-2D correspondence sets serve as the input for the PnP algorithm to estimate the pose of each camera.

3.5. Experimental Setup

Five cameras are installed at different locations in the catheterization laboratory at the Reinier de Graaf Gasthuis, Delft, NL. The installed cameras are named after their positions in the room: CornerNW (northwestern corner), CornerSW (southwestern corner), CornerSE (southeastern corner), WallS (southern wall) and WallW (western wall). The image frame size is 1920×1080 pixels for all cameras.

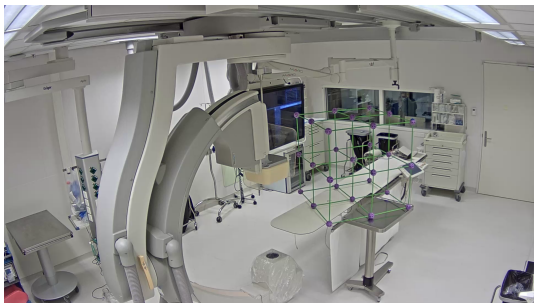


Fig. 3: The cube wireframe is placed on the operating table (view of CornerNW camera).

We use two recorded sessions for training and testing Scaled-YOLOv4. The two recorded sessions include one session for camera intrinsics calibration and the other session for a complete surgery procedure. We name them: calibration session and surgery session. Each recorded session has five videos for the five views. In a total of 10 video clips, we annotate 12 classes on 995 image frames and 3195 image frames. We use CVAT [17] for manual bounding box annotation.

A cube wireframe with 27 vertices is placed near the center of each view to evaluate the estimated pose (see Fig. 3). The length of each edge, and therefore the 3D coordinates of each vertex, are known. These positions are used as the ground truth in the evaluation.

3.6. Evaluation

In object detection, the Intersection over Union (IoU) describes the extent of overlap between the predicted and the annotated bounding box. IoU serves as a threshold to determine whether the prediction is a true positive. Average Precision (AP) is defined as the area under the precision-recall curve (PR-curve), and is calculated separately per class. Mean Average Precision (mAP) is defined as the AP values averaged over all classes. We use mAP@.5 and mAP@[.5:.95] specifically to evaluate the model performance: mAP@.5 represents the mean average precision by setting the IoU threshold to 50%, and mAP@[.5:.95] represents the mAP averaged over different IoU thresholds from 50% to 95% with a step of 5%.

To evaluate the estimated camera pose, we adopt two evaluation metrics: reprojection error (RPE) and Euclidean distance error (EDE). RPE refers to the error obtained by comparing the estimated key-point pixel coordinates (observed projection locations) to the locations obtained by projecting the 3D points according to the currently estimated camera pose [8]. For a 3D world coordinate $\mathbf{X_i}$, $\tilde{x}_i$ represents the projection of $\mathbf{X_i}$ on the image. RPE for $\mathbf{X_i}$ can be expressed as $\text{RPE}_i = \|\widetilde{x}_i - \mathbf{x_i}\|_2$. The RPE can be averaged over the N 3D-2D correspondences used for the estimation of the camera pose, i.e., $\text{RPE} = \sum_{i=1}^{N} \|\widetilde{x}_i - \mathbf{x_i}\|_2/N$. EDE refers to the error between the ground truth 3D coordinates and the triangulated 3D coordinates. In our case, EDE represents the

Table 1: The average precision results of Scaled-YOLOv4 tested on two recorded sessions: calibration session and surgery session.

	Calibration Session			Surgery Session		
Class	Targets	mAP@.5	mAP@.5:.95	Targets	mAP@.5	mAP@.5:.95
all	991	0.995	0.953	2760	0.995	0.992
door1	35	0.995	0.894	244	0.995	0.993
door2	115	0.995	0.993	269	0.995	0.995
window2	114	0.995	0.99	381	0.995	0.995
working bench	164	0.995	0.929	520	0.995	0.995
switch	61	0.994	0.921	269	0.995	0.99
wall_screen	87	0.995	0.988	126	0.995	0.995
patslide	148	0.995	0.956	217	0.995	0.995
charging stand	40	0.995	0.995	137	0.995	0.994
window1	73	0.995	0.993	244	0.995	0.995
light3	35	0.995	0.894	119	0.995	0.992
light2	35	0.995	0.995	119	0.995	0.985
light1	35	0.995	0.889	119	0.995	0.978

Euclidean distance differences averaged over 27 3D points on the cube wireframe (see Fig. 3). Since the triangulation relies on the estimated camera pose, EDE reflects the performance of our pose estimation directly. EDE can be written as EDE $= \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{X^i} - \mathbf{X^i_{\text{triangulated}}}\|_2$, where $\mathbf{X^i_{\text{triangulated}}}$ represents the i-th 3D point triangulated by 5 cameras.

4. RESULTS

4.1. Object Detection

For the calibration session, we use 796 frames for training and 199 frames for testing. For the surgery session, there are 2556 frames for training and 639 frames for testing. We start with a pre-trained YOLOv4-P5 (one of the Scaled-YOLOv4 architectures). The number of epochs and batch size are set to 50 and 32 respectively.

The average precision result for two recorded sessions is shown in Table 1. For the calibration session, the mean average precision (mAP@.5) averaged over all classes is 0.995, while mAP@.5:.95 is 0.939. AP@.5 for all the detected objects is above 0.99. In the surgery session compared to the calibration session, the result shows the same pattern. While mAP@.5 over all classes is 0.995 and mAP@.5:.95 is 0.992, AP@.5 for each detected object is all around 0.995.

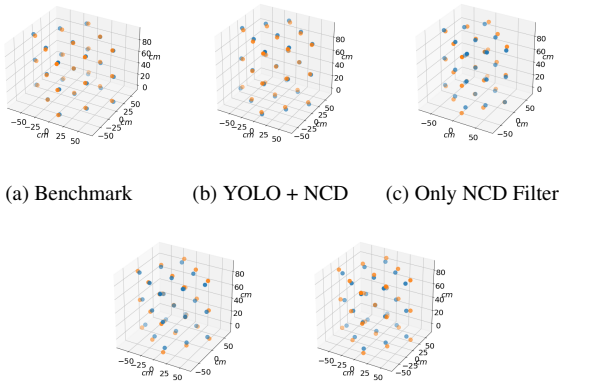
4.2. Camera Pose Estimation

For our benchmark, we manually annotate the 2D key-points that are visible in the input images and manually match them with the corresponding 3D coordinate measurements. We apply the same EPnP algorithm to retrieve the camera pose.

We compare the RPE and EDE in different configurations, including the benchmark (mannual key-point annotation), with/without Scaled-YOLOv4 detection, and with/without the NCD filter or the RPEM filter. The RPEM filter yields the lowest reprojection error of 5.79 pixels, since the RPEM is designed to minimize RPE. Except for the benchmark where key-points were annotated and matched manually, the NCD filter with Scaled-YOLOv4 detection shows the lowest

Table 2: The results of camera pose estimation, the metrics include reprojection error and Euclidean distance error.

		RPE (pixel)						EDE (cm)
		CornerNW	CornerSE	CornerSW	WallW	WallS	Average	Average
	Benchmark	6.91	20.18	17.21	11.20	15.35	14.16	2.73
NO YOLO Detection	NCD Filter	4.58	36.00	9.13	152.61	19.96	56.83	8.77
With YOLO Detection	NCD Filter	4.58	38.89	9.56	30.48	16.27	19.95	3.28
NO YOLO Detection	RPEM Filter	1.45	5.67	4.74	6.86	10.94	5.93	5.26
With YOLO Detection	RPEM Filter	1.45	5.67	5.24	6.86	9.74	5.79	4.97



(a) Benchmark (b) YOLO + NCD (c) Only NCD Filter

(d) YOLO + REPM (e) Only REPM Filter

Fig. 4: Euclidean Distance Error (cm). The orange dots represent the ground truth cube vertices, while the blue dots represent the triangulated 3D points. "YOLO" in the subplots refers to the bounding box filtering inferred by Scaled-YOLOv4.

EDE of 3.28 cm. However, if Scaled-YOLOv4 detection is removed, the NCD filter shows the worst performance.

5. DISCUSSION

In our study, there could be overfitting in the object detection due to fixed camera angles and the fixed positions of objects. The most effective approach to address the overfitting could be introducing other image datasets in similar scene setups such as catheterization laboratories at other hospitals. Secondly, the proposed approach still requires one session of manual measurements and annotations when the cameras are installed for the first time. Indeed key-points on reference objects need to be measured and stored in a database. However, once such database of key-points has been created, no further measurements are required and the camera pose estimation can be conducted automatically.

6. CONCLUSION

In this paper, an automatic pipeline for camera pose estimation is proposed and implemented. The proposed pipeline shows the ability for automating the camera pose estimation without any calibration patterns in a catheterization laboratory. This approach might also be applied to indoor localization in other contexts.

7. REFERENCES

[1] Tobias Blum, Hubertus Feußner, and Nassir Navab, "Modeling and segmentation of surgical workflow from laparoscopic video," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2010, pp. 400–407.

[2] Kevin Cleary, Ho Young Chung, and Seong K Mun, "Or2020 workshop overview: operating room of the future," in *International Congress Series*. Elsevier, 2004, vol. 1268, pp. 847–852.

[3] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua, "Worldwide pose estimation using 3d point clouds," in *European conference on computer vision*. Springer, 2012, pp. 15–29.

[4] Torsten Sattler, Bastian Leibe, and Leif Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 9, pp. 1744–1756, 2016.

[5] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson, "City-scale localization for cameras with known vertical direction," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1455–1461, 2016.

[6] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal Of Computer Vision*, vol. 81, pp. 155–166, 2009.

[7] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 8, pp. 930–943, 2003.

[8] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[9] Meng Xu, Youchen Wang, Bin Xu, Jun Zhang, Jian Ren, Stefan Poslad, and Pengfei Xu, "A critical analysis of image-based camera pose estimation techniques," 2022.

[10] Alex Kendall, Matthew Grimes, and Roberto Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," 2015.

[11] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2017.

[12] Mai Bui, Sergey Zakharov, Shadi Albarqouni, Slobodan Ilic, and Nassir Navab, "When regression meets manifold learning for object recognition and pose estimation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. may 2018, IEEE.

[13] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, "Scaled-yolov4: Scaling cross stage partial network," in *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 2021, pp. 13029–13038.

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[15] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[16] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[17] Boris Sekachev and Nikita Manovich et al., "opencv/cvat: v1.1.0," Aug. 2020.