Vision-based Camera/Robot Pose Estimation using Both Semantic and Geometric Features on LEGO Baseplates

Shu-Hao Yeh, Shuangyu Xie, Wei Yan, and Dezhen Song

***Abstract*— We are exploring the possibility of using LEGO baseplate as artificial landmarks (ALs) for robots and cameras in calibration, navigation or Augment Reality (AR) applications. LEGO baseplates are rigid, widely-available, low-cost, and precisely-manufactured and appear to be great candidate for ALs. However, they are also monochromatic with low contrast and easily affected by lighting. To overcome those issues, we utilize geometric and semantic information in our algorithm design by leveraging grid pattern, circle stud shapes, and text patterns. Our algorithm has extensively utilized the information for cross validation in noise filtering and position refinement using robust estimation methods. We have implemented and successfully tested our algorithm. The results show that our algorithm can recover more than 95% stud centers as feature points which ensures pose estimation accuracy. Our experiments also show that LEGO baseplate produces significantly more accurate camera pose estimation results than that of existing state-of-the-art counterpart when both methods are deployed by users with no computer vision background.**

I. Introduction

Artificial landmarks/visual fiducial markers such as checkerboards, quick response (QR) codes, AprilTag [1] are often used in assisting precise pose estimation for robots or cameras in applications such as calibration, indoor navigation or Augmented Reality (AR). Despite being a low-cost, high-precision, and robust solution, deployment of such artificial landmarks still requires customized fabrication, extensive setup, and calibration, which requires computer vision expertise that an average person may not have. Inspired by the fact that LEGO baseplates are rigid, widely-available, low-cost, and precisely-manufactured, we are interested in developing algorithms to enable them as artificial landmarks in vision-based camera/robot pose estimation, which can take those advantages to reduce barriers in deploying such systems.

Fig. 1 shows that our algorithm takes advantage of both geometric and semantic information from a LEGO baseplate which has a rectangle baseboard with a grid of studs and "LEGO" text engraved on top of each stud. We extract four baseplate corners, apply circle detection for stud centers, and employ a deep learning approach to extract bounding boxes and classify 4 types of text patterns. The latter also allows us to detect LEGO baseplate orientation. We exploit the grid pattern by applying robust estimation method on stud center points obtained by using both rectified circle centers and text bounding box centers. Finally, we estimate camera pose using a Maximum Likelihood Estimation (MLE) approach. We have implemented and successfully tested our algorithm. The results show that our algorithm can recover more than 95% stud centers as feature points which ensures pose estimation accuracy. Our experiments also show that LEGO baseplate produces significantly more accurate camera pose estimation results than that of state-of-the-art AprilTag [1] when both methods are deployed by users with no computer vision background.

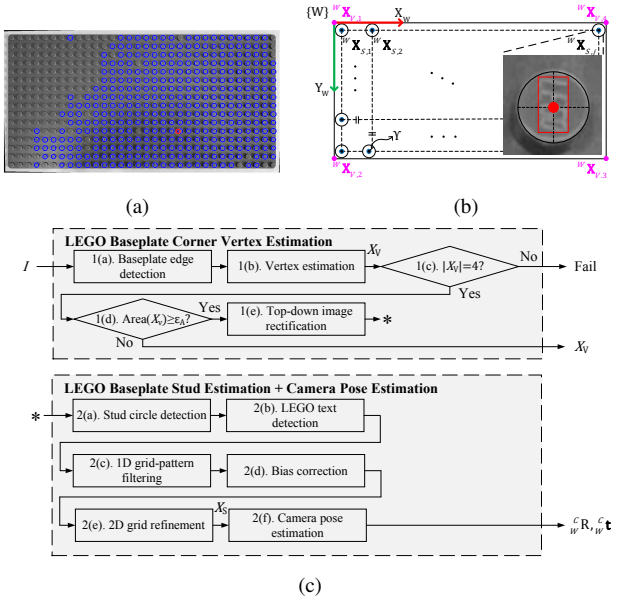


Fig. 1. (a) A LEGO baseplate in a rectified top-down view. (b) An illustration of LEGO baseplate with geometric features (circles and corner vertices) and semantic features (i.e. "LEGO" text on each circle). (c) LEGO baseplate feature estimation diagram.

II. Related Work

Vision-based camera pose estimation is a fundamental problem that enables localization in robotics [2] and 3D reconstruction in computer vision [3]. When a LEGO baseplate is used as an artificial landmark, the closely related areas are pose estimation fundamentals, features types, and approaches that use both semantic and geometric landmarks.

A camera pose estimation problem takes different forms with different inputs. If the input includes a single perspective image with known camera intrinsic parameters and a given set of 2D image points with known positions of the corresponding 3D points, this problem becomes Perspective-n-Point (PnP) problem [4]. If all 3D points are coplanar,

S. Yeh, S. Xie, and D. Song are with Computer Science and Engineering Department, Texas A&M University, College Station, TX 77843, USA. Emails: *ericex1015@tamu.edu*, *sy.xie@tamu.edu*, and *dzsong@cs.tamu.edu*.

W. Yan is with Architecture Department, Texas A&M University, College Station, TX 77843, USA. Email: *wyan@tamu.edu*.

This work was supported in part by National Science Foundation under IIS-2119549 and NRI-1925037, and TAMU Presidential Transformational Teaching Grants (PTTG).

then the camera pose can be solved using homography estimation and decomposition [5]. If the corresponding 3D points positions are not known but we have multiple perspectives, then this becomes 3D reconstruction that constructs both camera poses and 3D point positions [3]. Moreover, if the correspondence between 2D and 3D points are not known, then robust estimation method such as random sample consensus (RANSAC) [6] or its variation [7] can help establish the correspondence in any of the above scenarios. Our problem employs the PnP framework by focusing on challenges brought by LEGO baseplate landmarks.

Obtaining a set of 2D image points and their corresponding 3D points is nontrivial. These points are named as feature points because they are often distinctive points across views that can be easily detected and matched. They are also referred to as landmarks when used in robot navigation. We can classified them as two types: natural features and artificial landmarks. Natural features are distinctive geometric shapes or mathematical singularities such as corners [8]–[10], edges [11], circles, scale-invariant feature transforms [12], speeded up robust feature [13], ORB [14], planes [15], or feature combination [16]. The pros of natural features are that there is no need to modify environment which simplifies deployment. The cons are that indoor scenes often cannot ensure feature distribution uniformity which cause the algorithm to be unstable or infeasible.

Therefore, artificial landmarks or visual fiducial markers are often employed to increase localization or pose estimation robustness [17], [18]. The artificial landmarks must contain the distinguishable appearance to assist detection and position estimation [1], [19], [20]. To give the distinct identification of every artificial landmarks, a common design is to encode the identification into the QR code style "tag". ARToolKit [19] is the first visual tag system designed for the AR application. ARTag [20] is also a popular visual tag system, and utilizes the image gradient in the detection scheme to improve the marker detection. ARToolKitPlus [21] is the successor to the ARToolKit and extends the application to the mobile devices. AprilTag [1] is one of the state-of-the-art visual fiducial system.

However, printing and deploying exiting visual fiducial tags still require computer vision expertise which makes them less accessible to an average user. Our approach is to employ LEGO baseplate as a landmark to ease the deployment difficulty. Recently, LEGO baseplate has received attention in camera calibration [22] and AR [23]–[25]. On the other hand, we can learn from the recent progress made in nature feature detection to deal with the difficulty brought by LEGO baseplate. To be more specific, nature feature recognition has made good progress in combining semantic feature recognition using deep learning with traditional geometric feature recognition. Taira et al. [26] and Merrill et al. [27] combine the appearance, geometric and semantic features to assist in pose verification to enhance the localization accuracy and improve the robustness in the loop closure, respectively. Inspired by the existing works, our work also combine the geometric and semantic features to increase the detection rate of studs on the LEGO baseplate and improve the camera pose estimation accuracy.

III. Problem Definition

When a LEGO baseplate is in a camera field of view with sufficient resolution, we can use it to estimate camera pose.

A. Assumptions and Nomenclature

We have the following assumptions,

a.1 The camera is pre-calibrated and its lens distortion is removed from images.

a.2 Position noises of the points follow zero-mean Gaussian distribution with known variance σ^2 in each dimension and the noise in each dimension is independent.

Common notations in this paper are defined as follows.

K	the intrinsic matrix of the camera.
I	input image with its 2D coordinate system $\{I\}$.
$\{W\}$	3D world coordinate system (Fig. 1(b)), $\{W\}$ is defined on the LEGO baseplate with its origin located at the top-left corner of the LEGO baseplate, its X-axis aligned with the top edge (red arrow), and its Y-axis aligned with the left edge (green arrow).
$\{C\}$	3D camera coordinate system (CCS) for I where its origin is at the camera center, and its X-axis and Y-axis parallel to the horizontal and vertical axes of $\{I\}$, respectively.
$\mathbf{x}$	is a homogeneous vector describing a 2D point position in $\{I\}$, $\mathbf{x} \in \mathbb{P}^2$, 2D projective space.
$\mathbf{X}$	is a 3-vector describing a 3D point position, $\mathbf{X} \in \mathbb{R}^3$.
${}^C_W\mathrm{R}$	rotation matrix from $\{W\}$ to $\{C\}$, ${}^C_W\mathrm{R} \in \mathrm{SO}(3)$.
${}^C_W\mathbf{t}$	translation vector from $\{W\}$ to $\{C\}$, ${}^C_W\mathbf{t} \in \mathbb{R}^3$.

All 3D coordinate systems are right-handed system. Symbol $\tilde{}$ on a variable means that it is in inhomogeneous coordinate.

B. LEGO Baseplate

A LEGO baseplate is usually used as the base for LEGO building and widely-available in common household. It is a precisely-manufactured rectangle with a known size of $l_{\text{baseplate}} \times w_{\text{baseplate}}$, where $l_{\text{baseplate}}$ is length and $w_{\text{baseplate}}$ is width. They are rigid, flat, and monochromatic. On top of the base, it contains short cylinders which are called studs. Each stud has a radius of $r = 4.80$ mm. The studs form a grid/lattice pattern. There is a "LEGO" text on the top surface of each stud. The identification of LEGO baseplates can be easily encoded by its size and color. Fig 1(a) is a 16×32-stud gray baseplate. All these characteristics make it an ideal artificial landmark.

1) Geometric Features: There are two types of geometric features points and a grid pattern. First, the four vertexes of rectangular baseplate are robust features. Denote the i-th vertex as ${}^W\mathbf{X}_{V,i}$, where $i \in \{1, 2, 3, 4\}$, and ${}^W\mathbf{X}_{V,i}$ resides on the X-Y plane of $\{W\}$. These features can be detected even at a distance. Second, the centers of top circular surface of studs are abundant. Denote the j-th stud center feature as ${}^W\mathbf{X}_{S,j}$. The stud centers ${}^W\mathbf{X}_{S,j}$ are coplanar and the plane is parallel to X-Y plane of $\{W\}$ but with a height of 3.10

mm. We name this plane as stud plane. The grid-pattern has an equal distance of $\delta = 8.00$ mm between adjacent stud centers in both vertical and horizontal directions.

2) Semantic Features: Recall that there is a "LEGO" text on the top surface of each stud. It is worth noting that the center of the "LEGO" text co-locates with the center of the stud (e.g. red dot in the stud image in Fig. 1(b)) and is also located on the stud plane which provides a great semantic feature that can be translated to geometric coordinates. Denote the k-th text center location as ${}^{W}\mathbf{X}_{S,k}$.

To summarize, a LEGO baseplate provides feature sets: $\{{}^{W}\mathbf{X}_{V,i}, {}^{W}\mathbf{X}_{S,j}, {}^{W}\mathbf{X}_{S,k}\}, \forall i, j, k$, and a grid pattern.

C. Problem Definition

Our problem is defined as,

Definition 1: Given the input image I and LEGO baseplate parameters, estimate the camera pose ${}^{C}_{W}\mathrm{R}$ and ${}^{C}_{W}\mathbf{t}$.

IV. Algorithm

Our algorithm consists of two main blocks: (1) LEGO baseplate feature extraction (Fig. 1(c)) and (2) camera pose estimation. We begin with the former.

A. LEGO Baseplate Feature Extraction

1) LEGO Baseplate Corner Vertex Detection: The four corners of a LEGO baseplate provides a good starting set of features which can be detected as a distance. Since a LEGO baseplate is monochromatic, we can apply color-based segmentation to extract its silhouette. We then find the 4 edges of the LEGO baseplate by using the line fitting with RANSAC. Denote the j-th edge as $\mathbf{l}_{e,j}$, where $\mathbf{l}_{e,j} \in \mathbb{R}^3$ is the line model. The vertices are obtained by intersecting the line edges. Denote the i-th vertex as $\mathbf{x}_{v,i}$. Let $\mathbf{l}_{e,i_1}$ and $\mathbf{l}_{e,i_2}$ be the edges intersecting at $\mathbf{x}_{v,i}$. The vertex set $\mathcal{X}_V$ can be obtained by

$$\mathcal{X}_V := \{\mathbf{x}_{v,i} : \mathbf{l}_{e,i_1} \times \mathbf{l}_{e,i_2}\}. \tag{1}$$

The uncertainty of the vertex depends on the error distribution of edges, and is propagated through the edge intersection. We utilize the first order approximation of the covariance matrix [3] to estimate the covariance matrix of the vertex $\mathbf{x}_{v,i} \in \mathcal{X}_V$. Denote the covariance matrix of $\mathbf{x}_{v,i} \in \mathcal{X}_V$ as $\Sigma_{v,i}$. The first order approximation of $\Sigma_{v,i}$ is

$$\Sigma_{v,i} = J_{e,i_1}\Sigma_{e,i_1}J_{e,i_1}^{\mathsf{T}} + J_{e,i_2}\Sigma_{e,i_2}J_{e,i_2}^{\mathsf{T}}, \tag{2}$$

where $J_{e,i_1} = \frac{\partial \mathbf{x}_{v,i}}{\partial \mathbf{l}_{e,i_1}}$, and $J_{e,i_2} = \frac{\partial \mathbf{x}_{v,i}}{\partial \mathbf{l}_{e,i_2}}$. Σ_{e,i_1} and Σ_{e,i_2} are denoted as the covariance matrices of $\mathbf{l}_{e,i_1}$ and $\mathbf{l}_{e,i_2}$. Σ_{e,i_1} and Σ_{e,i_2} are directly related to the accuracy of the color-based segmentation and chosen empirically.

We know that a minimum solution for estimating camera pose requires

$$|\mathcal{X}_V| = 4. \tag{3}$$

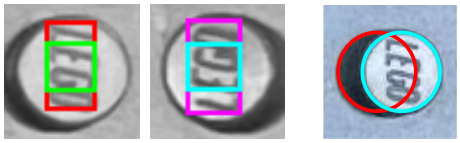
If not satisfied, we report a failure case. Otherwise, we then verify if the area of the baseplate in the image is bigger than an empirical threshold ϵ_A,

$$\mathrm{area}(\mathcal{X}_V) \geq \epsilon_A, \tag{4}$$

where $\mathrm{area}(\cdot)$ computes area of the polygon. If true, the image is sufficiently detailed for stud feature detection. $\mathcal{X}_V$ tells us the dimension and size of LEGO baseplate and allows us to estimate a homography transformation H, using Direct Linear Transform (DLT) algorithm in [3], to rectify the baseplate region into a top-down view (e.g. Fig. 1(a)) which is defined as I',

$$I' := \{\mathrm{H}\mathbf{x}_i : \forall \mathbf{x}_i \in I\}. \tag{5}$$

Only using the four LEGO baseplate corner vertices is often not accurate enough because boundary segmentation using edge detection often has large error due to baseplate thickness and shadow. Since image resolution is sufficient, we use more features from studs. With LEGO baseplate image rectified as a rectangle, we can partition each stud by using the known geometry of the stud distribution on the baseplate. For j-th stud, its segmented image is defined as I'_j. The stud images in Fig. 2 show examples of the segmented studs. We apply circle detection and perform text recognition on each I'_j to extract geometric features and semantic features, respectively.



(a) (b) (c)

Fig. 2. (a) and (b) LEGO text detection using 4 classes of patterns in different colors. (c) Illustration of bias induced from the stud height. The red circle is the false detection affected by the shadow while the light blue circle is the correct stud detection.

2) Stud Circle Detection: For circle detection, we apply Canny edge detector [11] to obtain the circular edges and apply circle Hough Transform (CHT) algorithm on the edge image to detect circles. Since each stud is a cylinder, which can create the shadow with circular shape. The shadow may cause CHT algorithm to detect false circles which usually have random radius. Since that I' is up to scale s of the physical LEGO baseplate, we can use the scale s and radius r to filter the false detection. Denote the j-th stud center detected from CHT in I'_j as $\mathbf{x}'_j$ with the radius r_j. The filtered stud set from CHT of all studs can be defined as

$$\mathcal{X}_{\text{circle}} := \bigcup_j \{\mathbf{x}'_j : |r_j - sr| \leq \epsilon_r\}, \tag{6}$$

where ϵ_r is the empirical radius difference threshold.

3) LEGO Text Detection: We apply convolutional neural network (CNN) based object detection to detect the semantic features from the "LEGO" text. We consider the "LEGO" text as a complete pattern instead of using 4 separate characters since the low contrast and poor lighting condition pose challenges in directly adopting the text recognition method such as optical character recognition. To enhance the robustness of the detection, we add additional "EG" class since "L" and "O" in "LEGO" may not be eligible due to wear and tear in LEGO usage. We also consider orientation of the "LEGO" text. There are 2 orientations

because "LEGO" text is along vertical direction, and the baseplate may be flipped upside-down in the rectified view. Therefore, we define 4 classes (Figs. 2(a) and 2(b)), and train a state-of-the-art level CNN object detection model Yolov4-tiny [28] as our detection algorithm. The center of output bounding box is the corresponding stud center.

Due to the monochromatic LEGO baseplate and poor lighting conditions, false detection is inevitable. Falsely detected text patterns have random width to height ratio for text bounding boxes. Therefore, we can use the scale s and the ratio of width to height of "LEGO" ρ to filter out false detections. Again, stud centers detected by Yolov4-tiny from I'_j are $\mathbf{x}'_j$ with the "LEGO" bounding box ratio of width to height ρ_j. For "EG" detection, we can scale the bounding box to obtain "LEGO" bounding box since "LEGO" and "EG" are co-centered, and their dimension is known. The filtered stud center set from Yolov4-tiny is defined as

$$\mathcal{X}_{\text{text}} := \bigcup_j \left\{ \mathbf{x}'_j : |\rho_j - s\rho| \leq \epsilon_b \right\}, \tag{7}$$

where ϵ_b is the empirical ratio difference threshold. The classes identification can help us identify the orientation of the LEGO baseplate by voting.

4) 1D Grid-Pattern Filtering: After we obtain $\mathcal{X}_{\text{circle}}$ and $\mathcal{X}_{\text{text}}$, we enforce the grid-pattern to filter out the falsely-detected features. The grid-pattern is composed by vertical and horizontal lines. We first validate the vertical lines and the horizontal lines, respectively.

Recall that the distance δ between the adjacent stud centers in LEGO baseplate and the scale s defined for I' are known. Since we compute vertical and horizontal lines separately, we perform 1-D lattice pattern fitting for each line group. Define $(\cdot)_n$ as the n-th element of vector. We apply RANSAC on $(\tilde{\mathbf{x}}'_j)_1$ to find the vertical lines in the 1D grid-pattern.

Assume that $\mathbf{x}'_p \in \mathcal{X}_{\text{circle}} \cup \mathcal{X}_{\text{text}}$ is the point selected by RANSAC with the most inlier support, and the corresponding 1D lattice pattern set can be represented by

$$\mathbf{X} := \left\{ x = (\tilde{\mathbf{x}}'_p)_1 + sn_x\delta : x/s \in [0, l_{\text{baseplate}}], n_x \in \mathbb{Z} \right\}, \tag{8}$$

where $n_x \in \mathbb{Z}$ is an integer multiplier for lattice vertices. The 1D lattice pattern $\mathbf{X}$ is considered as high quality when the inliers are sufficient

$$\sum_{x \in \mathbf{X}} \left\{ \sum_{\mathbf{x}'_j \in \mathcal{X}_{\text{circle}} \cup \mathcal{X}_{\text{text}}} 1_X(\mathbf{x}'_j, x) \right\} > \eta_g, \tag{9}$$

where η_g is the number threshold, $1_X(\mathbf{x}'_j, x)$ is an inlier indicator function to determine if x shares the same vertex with $(\tilde{\mathbf{x}}'_j)_1$,

$$1_X(\mathbf{x}'_j, x) := \begin{cases} 1, & \text{if} \quad \|(\tilde{\mathbf{x}}'_j)_1 - x\|_2^2 \leq \epsilon_g \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

$\|\cdot\|_2$ is the L2 norm, and ϵ_g is a distance error threshold.

Similarly, we can find the 1D lattice pattern for horizontal lines, and validate the lattice pattern by using the inlier number. Assume that $\mathbf{x}'_q \in \mathcal{X}_{\text{circle}} \cup \mathcal{X}_{\text{text}}$ is the point selected by RANSAC with the most inlier support. Define the validated 1D lattice pattern for horizontal line as set

$$\mathbf{Y} := \left\{ y = (\tilde{\mathbf{x}}'_q)_2 + sn_y\delta : y/s \in [0, w_{\text{baseplate}}], n_y \in \mathbb{Z} \right\}, \tag{11}$$

where $n_y \in \mathbb{Z}$ is an integer multiplier for lattice vertices. Similarly, we have an inlier indicator function $1_Y(\mathbf{x}'_j, y)$, and the thresholding function for determining inlier sizes which shares the same format as (9).

We use the two 1D lattice sets $\mathbf{X}$ and $\mathbf{Y}$ to filter the false studs. A stud is considered to be in the grid-pattern only if the stud center is the inlier for both $\mathbf{X}$ and $\mathbf{Y}$. Define s_X and s_Y as inlier scoring functions

$$s_X(\mathbf{x}'_j) = \sum_{x \in \mathbf{X}} 1_X(\mathbf{x}'_j, x) \quad \text{and} \quad s_Y(\mathbf{x}'_j) = \sum_{y \in \mathbf{Y}} 1_Y(\mathbf{x}'_j, y). \tag{12}$$

The grid-pattern stud center sets are defined as follows

$$\mathcal{G}_{\text{circle}} := \left\{ \mathbf{x}'_j \in \mathcal{X}_{\text{circle}} : \left(s_X(\mathbf{x}'_j) = 1\right) \bigwedge \left(s_Y(\mathbf{x}'_j) = 1\right) \right\}, \tag{13}$$

and we can define $\mathcal{G}_{\text{text}}$ similarly. Note that we still separate the grid-pattern studs to $\mathcal{G}_{\text{circle}}$ and $\mathcal{G}_{\text{text}}$. The reason behind that is that the stud $\mathbf{x}'_j \in \mathcal{G}_{\text{circle}}$ may contain positional bias which is introduced from the shadow caused by directional lighting conditions. The bias detection and correction are necessary.

5) Bias correction for $\mathcal{G}_{\text{circle}}$: It is necessary to detect and correct the the bias in $\mathcal{G}_{\text{circle}}$ because the bias can violate the assumption **a.2** of the zero-mean of the measurement error and cause the later MLE in the camera pose to output biased estimation. We employ $\mathcal{G}_{\text{text}}$ to assist in detecting the bias since the "LEGO" texts are on stud surface and do not suffer from the bias caused by lighting and shadow. Fig. 2(c) illustrates how bias is caused by stud shadow.

In an ideal scenario, the stud centers $\mathbf{x}'_j \in \mathcal{G}_{\text{circle}}$ and $\mathbf{x}'_j \in \mathcal{G}_{\text{text}}$ should share the same mean $\bar{\mathbf{x}}'_j$ representing the true stud center position. However, the mean of $\mathbf{x}'_j \in \mathcal{G}_{\text{circle}}$ may not be equal to $\bar{\mathbf{x}'}_j$ if a bias exists. Define $\mathcal{I}_G = \{j_g : g = 1, \cdots, n_g\}$ as the index set of the common studs between $\mathcal{G}_{\text{circle}}$ and $\mathcal{G}_{\text{text}}$, where j_g is the stud index of the g-th common stud, and $n_g = |\mathcal{G}_{\text{circle}} \cap \mathcal{G}_{\text{text}}|$ is the number of the common studs. To detect the bias, we design the following hypothesis testing based on Hotelling's T^2 test:

$$\mathbf{H_0} : \mu'_{\text{circle}} = \mu'_{\text{text}} \quad \text{vs.} \quad \mathbf{H_1} : \mu'_{\text{circle}} \neq \mu'_{\text{text}}, \tag{14}$$

where $\mu'_{\text{circle}} = \frac{1}{n_g} \sum_{\substack{\mathbf{x}'_j \in \mathcal{G}_{\text{circle}} \\ j \in \mathcal{I}_G}} \mathbf{x}'_j$ and $\mu'_{\text{text}} = \frac{1}{n_g} \sum_{\substack{\mathbf{x}'_j \in \mathcal{G}_{\text{text}} \\ j \in \mathcal{I}_G}} \mathbf{x}'_j$. Denote the error vector between the studs $\mathbf{x}'_{j_g} \in \mathcal{G}_{\text{circle}}$ and $\mathbf{x}'_{j_g} \in \mathcal{G}_{\text{text}}$ as $\mathbf{e}'_{j_g}$. The test statistic can be calculated by $T^2 = n_g \bar{\mathbf{e}}'^{\mathsf{T}}_g \Sigma_{e_g}^{-1} \bar{\mathbf{e}}'_g$, where

$$\bar{\mathbf{e}}'_g = \frac{1}{n_g} \sum_{j_g \in \mathcal{I}_g} \mathbf{e}'_{j_g} \text{ and } \Sigma_{e_g} = \frac{\sum\limits_{j_g \in \mathcal{I}_g} (\mathbf{e}'_{j_g} - \bar{\mathbf{e}}'_g)(\mathbf{e}'_{j_g} - \bar{\mathbf{e}}'_g)^{\mathsf{T}}}{n_g - 1}. \tag{15}$$

Define $F_{a,b}(x)$ as the cumulative distribution function of the F-distribution at value x with a numerator degree of freedom (DoF) and b denominator DoF. By setting the

significance level α, the p-value is obtained by $F_{a,b}^{-1}(1-\alpha)$. We consider that the bias $\mathbf{b}'$ exists by rejecting $\mathbf{H}_0$ when $T^2 > \frac{2(n_g-1)}{n_g-2} F_{2,n_g-2}^{-1}(1-\alpha)$.

If the hypothesis test confirms the existence of bias, we remove bias as follows,

$$\mathbf{x}'_j = \mathbf{x}'_j - \bar{\mathbf{e}}', \forall \mathbf{x}'_j \in \mathcal{G}_{\text{circle}}. \tag{16}$$

6) 2D Grid Refinement: So far, we estimate two 1D lattice sets $\mathbf{X}$ and $\mathbf{Y}$ separately. For better accuracy, we want to re-estimate the stud centers by simultaneously using the both vertical and horizontal lines in the grid-pattern. Define the inlier sets of $x \in \mathbf{X}$ and $y \in \mathbf{Y}$ as $\mathcal{G}_x := \{\mathbf{x}'_j \in \mathcal{G}_{\text{circle}} \cup \mathcal{G}_{\text{text}} : 1_X(\mathbf{x}'_j, x) = 1\}$ and $\mathcal{G}_y := \{\mathbf{x}'_j \in \mathcal{G}_{\text{circle}} \cup \mathcal{G}_{\text{text}} : 1_Y(\mathbf{x}'_j, y) = 1\}$. The grid-patterns $\mathbf{X}$ and $\mathbf{Y}$ are refined by

$$\min_{\substack{\forall x \in \mathbf{X} \\ \forall y \in \mathbf{Y}}} \sum_{x \in \mathbf{X}} \sum_{\mathbf{x}'_j \in \mathcal{G}_x} \|(\tilde{\mathbf{x}}'_j)_1 - x\|_2^2 + \sum_{y \in \mathbf{Y}} \sum_{\mathbf{x}'_j \in \mathcal{G}_y} \|(\tilde{\mathbf{x}}'_j)_2 - y\|_2^2. \tag{17}$$

After obtaining the refined $\mathbf{X}$ and $\mathbf{Y}$, we map the stud centers back to I by using inverse homography H^{-1}. Denote the j-th stud on the grid-pattern in I as $\mathbf{x}_{s,j}$. The stud center set on the grid can be obtained by

$$\mathcal{X}_S := \left\{ \mathbf{x}_{s,j} = \mathrm{H}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} : \forall x \in \mathbf{X} \quad \text{and} \quad \forall y \in \mathbf{Y} \right\}. \tag{18}$$

If the lattice vertices are too few $|\mathcal{X}_S| < \eta_S$ for a given threshold η_S due to insufficient features, we discard them and only use (1) to estimate camera pose. The uncertainty of the stud $\mathbf{x}_{s,j} \in \mathcal{X}_S$ depends on the two 1D lattice patterns and the homogrpahy H, and is propagated through the grid-pattern estimation. We utilize the first order approximation of covariance matrix [3] to estimate the covariance matrix of $\mathbf{x}_{s,j} \in \mathcal{X}_S$. Denote the covariance matrix of $\mathbf{x}_{s,j} \in \mathcal{X}_S$ as $\Sigma_{s,j}$. Lemma 1 details covariance matrix $\Sigma_{s,j}$.

Lemma 1: Under the Gaussian noise assumption, the covariance matrix $\Sigma_{s,j}$ is

$$\Sigma_{s,j} = J_h \Sigma_h J_h^\mathsf{T} + \sigma_x^2 J_x J_x^\mathsf{T} + \sigma_y^2 J_y J_y^\mathsf{T}, \tag{19}$$

where $J_h = \frac{\partial \mathbf{x}_{s,j}}{\partial \mathbf{h}}$, $\mathbf{h} \in \mathbb{R}^9 = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}^\mathsf{T}$ is the vector-format of $\mathrm{H} = \begin{bmatrix} \mathbf{h}_1^\mathsf{T} & \mathbf{h}_2^\mathsf{T} & \mathbf{h}_3^\mathsf{T} \end{bmatrix}^\mathsf{T}$, and Σ_h is the covariance matrix of $\mathbf{h}$. σ_x^2 is the variance of $x \in \mathbf{X}$, and $J_x = \frac{\partial \mathbf{x}_{s,j}}{\partial x}$. σ_y^2 is the variance of $y \in \mathcal{H}$, and $J_y = \frac{\partial \mathbf{x}_{s,j}}{\partial y}$.

Proof: H is estimated by the vertices $\mathbf{x}_{v,i} \in \mathcal{X}_V$. The covariance matrix Σ_h can be obtained by using backward propagation of covariance [3]

$$\Sigma_h = \left(\sum_i J_{v,i}^\mathsf{T} \Sigma_{v,i}^{-1} J_{v,i} \right)^+, \tag{20}$$

where $J_{v,i} = \frac{\partial \hat{\mathbf{x}}_{v,i}}{\partial \mathbf{h}}$ and $\hat{\mathbf{x}}_{v,i} = \mathrm{H}\mathbf{x}'_{v,i}$.

$x \in \mathbf{X}$ and $y \in \mathbf{Y}$ are estimated from $\mathbf{x}'_j \in \mathcal{G}_{\text{circle}} \cup \mathcal{G}_{\text{text}}$. Define $\mathcal{G}_{\text{circle},x}$ as $\mathcal{G}_{\text{circle},x} := \{\mathbf{x}'_{j \in \mathcal{G}_{\text{circle}}} : 1_X(\mathbf{x}'_j, x) = 1\}$. We also define $\mathcal{G}_{\text{text},x}$ as $\mathcal{G}_{\text{text},x} := \{\mathbf{x}'_{j \in \mathcal{G}_{\text{text}}} : 1_X(\mathbf{x}'_j, x) = 1\}$. Assume that the covariance matrices of $\mathbf{x}'_j \in \mathcal{G}_{\text{circle}}$ and $\mathbf{x}'_j \in \mathcal{G}_{\text{text}}$ are Σ_c and Σ_t, where Σ_c and Σ_t can be estimated empirically. The variance σ_x^2 can be obtained by using backward propagation of covariance

$$\sigma_x^2 = \left(\sum_{\mathbf{x}'_j \in \mathcal{G}_{\text{circle}},x} J_{x,j}^\mathsf{T} \Sigma_c^\mathsf{T} J_{x,j} + \sum_{\mathbf{x}'_j \in \mathcal{G}_{\text{text}},x} J_{x,j}^\mathsf{T} \Sigma_t^\mathsf{T} J_{x,j} \right)^{-1}. \tag{21}$$

Denote the error between $(\tilde{\mathbf{x}}'_j)_1$ and x as $e_{x,j} = (\tilde{\mathbf{x}}'_j)_1 - x$. $J_{x,j} = \frac{\partial e_{x,j}}{\partial \mathbf{x}'_j}$. Similarly, we define subsets $\mathcal{G}_{\text{circle},y}$ and $\mathcal{G}_{\text{text},y}$, and we can obtain σ_y^2 by using backward propagation of covariance

$$\sigma_y^2 = \left(\sum_{\mathbf{x}'_j \in \mathcal{G}_{\text{circle}},y} J_{y,j}^\mathsf{T} \Sigma_c^\mathsf{T} J_{y,j} + \sum_{\mathbf{x}'_j \in \mathcal{G}_{\text{text}},y} J_{y,j}^\mathsf{T} \Sigma_t^\mathsf{T} J_{y,j} \right)^{-1}. \tag{22}$$

Denote the error between $(\tilde{\mathbf{x}}'_j)_2$ and y as $e_{y,j}$. $J_{y,j} = \frac{\partial e_{y,j}}{\partial \mathbf{x}'_j}$. Because H, x and y are independent, the covariance matrix of $\mathbf{x}_{s,j}$ can be obtained from (19) by using the propagation of covariance. ■

B. Camera Pose Estimation

With the stud centers in $\{I\}$ and the associated covariance matrices derived, we are ready to utilize the 2D and 3D LEGO feature correspondences $\{\mathbf{x}_{v,i} \leftrightarrow {}^W\mathbf{X}_{V,i}\}$ and $\{\mathbf{x}_{s,j} \leftrightarrow {}^W\mathbf{X}_{S,j}\}$ to estimate the camera pose. We estimate the initial camera pose ${}^C_W\mathrm{R}$ and ${}^C_W\mathbf{t}$ by using PnP method [4]. We then employ the MLE to refine ${}^C_W\mathrm{R}$ and ${}^C_W\mathbf{t}$. Define the measurement vector as

$$\mathbf{e} := \begin{bmatrix} \widetilde{(\hat{\mathbf{x}}_{v,1})} - \tilde{\mathbf{x}}_{v,1} \\ \vdots \\ \widetilde{(\hat{\mathbf{x}}_{v,4})} - \tilde{\mathbf{x}}_{v,4} \\ \vdots \\ \widetilde{(\hat{\mathbf{x}}_{s,j})} - \tilde{\mathbf{x}}_{s,j} \\ \vdots \end{bmatrix} \in \mathbb{R}^m, \tag{23}$$

where $m = 2|\mathcal{X}_S| + 8$, $\hat{\mathbf{x}}_{v,i} = \mathrm{K}\left({}^C_W\mathrm{R}W\mathbf{X}_{V,i} + {}^C_W\mathbf{t}\right)$, $\hat{\mathbf{x}}_{s,j} = \mathrm{K}\left({}^C_W\mathrm{R}W\mathbf{X}_{S,j} + {}^C_W\mathbf{t}\right)$, $\widetilde{[x,y,z]}^\mathsf{T} = [x/z, y/z]^\mathsf{T}$ is vector de-homogenizing, and $\tilde{\mathbf{x}}_{v,1}, \tilde{\mathbf{x}}_{v,4}$, and $\tilde{\mathbf{x}}_{s,j}$ are de-homogenized versions of $\mathbf{x}_{v,1}, \mathbf{x}_{v,4}$, and $\mathbf{x}_{s,j}$, respectively. The MLE solves ${}^C_W\mathrm{R}$ and ${}^C_W\mathbf{t}$ by minimizing

$$\min_{{}^C_W\mathrm{R}, {}^C_W\mathbf{t}} \mathbf{e}^\mathsf{T} \Sigma_e^{-1} \mathbf{e}. \tag{24}$$

$\Sigma_e \in \mathbb{R}^{m \times m}$ is the covariance matrix of $\mathbf{e}$, where $\Sigma_e = \mathrm{Diag}(\Sigma_{\tilde{v},1}, \cdots, \Sigma_{\tilde{v},4}, \cdots, \Sigma_{\tilde{s},j}, \cdots)$. $\Sigma_{\tilde{v},i} \in \mathbb{R}^{2\times 2}$ is the covariance matrix of $\tilde{\mathbf{x}}_{v,i}$, and can be obtained by using forward propagation of covariance [3] from $\Sigma_{v,i}$ in (2) by dehomogenising. $\Sigma_{\tilde{s},j} \in \mathbb{R}^{2\times 2}$ is the covariance matrix of $\tilde{\mathbf{x}}_{s,j}$, and can be obtained by using forward propagation of covariance from $\Sigma_{s,j}$ in (19) by dehomogenising.

For uncertainty analysis, let us denote the unit quaternion vector of ${}^C_W\mathrm{R}$ as ${}^C_W\mathbf{q} \in \mathbf{R}^4$. We define the camera pose by a 7-vector $\mathbf{p} = \begin{bmatrix} {}^C_W\mathbf{q}^\mathsf{T} & {}^C_W\mathbf{t}^\mathsf{T} \end{bmatrix}^\mathsf{T}$ in SE(3) which consists

of the unit quaternion vector ${}^{C}_{W}\mathbf{q}$ and the translation vector ${}^{C}_{W}\mathbf{t}$. The covariance of $\mathbf{p}$ is denoted as Σ_p. The first order approximation of Σ_p is

$$\Sigma_p = \left(\sum_{\mathbf{x}_{v,i} \in \mathcal{X}_V} J_{v,i}^{\mathsf{T}} \Sigma_{\tilde{v},i}^{-1} J_{v,i} + \sum_{\mathbf{x}_{s,j} \in \mathcal{X}_S} J_{s,j}^{\mathsf{T}} \Sigma_{\tilde{s},j}^{-1} J_{s,j} \right)^{-1}. \tag{25}$$

Denote the error vector $\widetilde{(\hat{\mathbf{x}}_{v,i})} - \tilde{\mathbf{x}}_{v,i}$ as $\mathbf{e}_{v,i}$. Denote the error vector $\widetilde{(\hat{\mathbf{x}}_{s,j})} - \tilde{\mathbf{x}}_{s,j}$ as $\mathbf{e}_{s,j}$. $J_{v,i} = \frac{\partial \mathbf{e}_{v,i}}{\partial \mathbf{p}}$, and $J_{s,j} = \frac{\partial \mathbf{e}_{s,j}}{\partial \mathbf{p}}$.

To quantify the uncertainty of the estimate camera pose, we use the maximum eigenvalue of Σ_p. The maximum eigenvalue of Σ_p is

$$\lambda_{\max} = \max_i \lambda_i(\Sigma_p), \tag{26}$$

where $\lambda_i(\Sigma_p)$ is the i-th eigenvalue of Σ_p.

V. EXPERIMENTS

We have implemented our system in Matlab and Python. We first analyze the working distance for LEGO baseplate feature detection, and then evaluate the camera pose accuracy. In the experiments, we use the Do3Think industrial camera with a result of 2200×2200 pixels. For the experimental setup, we choose the LEGO 8×16-stud gray plate (See Fig. 3(a)) which can be used not only as the baseplate but also as the LEGO brick with a taller height of the stud plane 5 mm.

A. Features and Algorithm Setup in Comparison

We have 5 different feature setups in our experiments as detailed below.

- G: Only use circle centers, the geometric features with the 1D grid-pattern filter in Sec. IV-A.4.
- S: Similar to "G", but we use the semantic features returned from the text detection instead.
- GS: We combine "G" with "S" and apply the 1D grid pattern filtering in Sec. IV-A.4.
- GS+B: We run GS with the bias correction after the stud features obtained in "GS".
- GS+BG: We run GS+B with the 2D grid refinement in Sec. IV-A.6.

B. LEGO Baseplate Working Range Tests

In the tests, we want to find the LEGO baseplate working distance. We fix the LEGO baseplate on the vertical surface, and make the camera look perpendicular to the LEGO baseplate. We choose 6 different distance positions to show the change in the feature detection rate according to our experiment setup. The distance here is the perpendicular distance between camera center and the LEGO stud plane and calculated by the estimate camera pose. We begin with the distance 17.5 cm away from the LEGO stud plane since the LEGO 8×16 baseplate can not be included completely in the image when we move the camera closer than 17.5 cm. Every two positions are 2.5 cm apart to show the change of feature detection rate. In every position, we collect 20 images. We also ensure that the LEGO baseplate in the image is always close to the image center when we move the camera. For the feature setups used in the tests, we consider G, S and GS+BG here since G and S are the basis of all other three feature setups, and GS+BG is the final feature.

1) Evaluation Metric: Two metrics have been introduced to evaluate the LEGO baseplate working distance. i) First, we use the feature detection rate γ to track how many stud centers have been successfully recognized

$$\gamma = \frac{|\mathcal{X}_S|}{n_S}, \tag{27}$$

where $n_S = 8 \times 16$ is the number of the studs on the LEGO baseplate. ii) We also use the standard reprojeciton error ϵ in computer vision [3]. We calculate the reprojection error of stud centers using the estimated camera pose.

2) Experimental Results: Tab. I shows the experimental results under 6 different distance. We also provide the average stud circle radius r_s since the distance determines the pixel resolution of the studs, and affects the average detection rate. In the Tab. I, the average detection rate Avg(γ) of S is always higher than G. It is expected since the edge of the "LEGO" text is sharper than the stud circular edge. It is clear that the LEGO baseplate feature detection works more stable when the average stud circular radius is larger than 24.71 pixels, and in our case the perpendicular distance between the camera and the LEGO baseplate is smaller or equal to 25.0 cm. When the average stud circular radius is smaller than or equal to 22.50, the average reprojection error of stud centers increases significantly, which is directly affected by the insufficient feature detection of both geometric and semantic features with lower than 10% of Avg(γ) in both G and S.

Keep in mind that the actual working distance may vary for different camera lenses. The more important indicator is the average stud radius in pixel r_s.

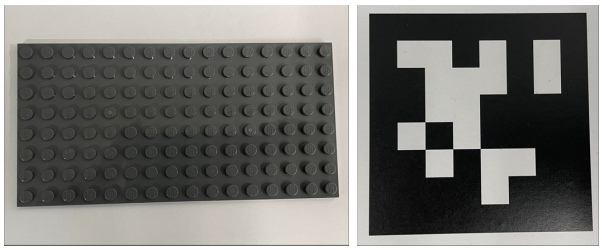
TABLE I

EXPERIMENTAL RESULTS OF WORKING DISTANCE TESTS. r_s AND ϵ ARE IN PIXELS.

Distance		17.5	20.0	22.5	25.0	27.5	30.0
Avg(r_s)		36.31	31.37	27.81	24.71	22.50	20.64
Avg(γ)%	G	69.4	54.2	53.2	54.4	29.6	6.7
	S	95.0	92.3	94.6	76.3	39.5	8.8
	GS+BG	99.5	99.0	98.9	95.3	74.1	42.2
Avg(ϵ)	GS+BG	**0.73**	**0.61**	**0.58**	**0.85**	1.47	2.08

C. Camera Pose Estimation Tests

We are interested in whether the using of geometric and semantics features makes a difference in motion estimation accuracy and feature detection rate in our setup. In addition to 5 feature setups: G, S, GS, GS+B and GS+BG (See Sec. V-A), we also compare with the AprilTag [1] which is one of the state-of-the-art artificial landmark (See Fig. 3(b)), and the detail setup is described in Sec. V-C.1. We include another camera into our tests, the iPhone 12 Pro camera with a result of 1440×1080 pixels, to show the wide adaption of using the features of the LEGO baseplate in motion estimation for different cameras.



(a) (b)

Fig. 3. Experiment setups for (a) LEGO 8×16-stud gray plate and (b) AprilTag [1].

1) AprilTag Setup in Comparison: The dimension of the AprilTag is 9 cm in each dimension, which makes it same area size with the LEGO 8×16-stud gray baseplate. We have 3 different deployment setups of the AprilTag in our experiments and summarize details in Tab. II. AprilTag* is the ideal setup with the high quality printing (laser printing on adhesive sheet) and the high rigidity deployed surface (mirror planar surface) to ensure the flatness. We also calibrate the dimension of the AprilTag in AprilTag* to increase the precision since cameras have higher precision in measuring compared with printing. This calibration only can be done if the flatness and rigidity of artificial landmark are satisfied, and it requires the computer vision expertise. In addition to the ideal setup, we also include another two setups: AprilTag^1 and AprilTag^2 to imitate how regular users without computer vision expertise construct the artificial landmarks in reality. For example, using plastic planar surface is common since it is easily accessible and its light weight can be easily adopted in different tasks. It is also common that the users print the AprilTag by themselves and adhere it on the targeted surface using tape or glue. It is worth noting that dimension calibration cannot be performed in this both setups since the flatness is not guaranteed, and regular users also do not have calibration knowledge.

TABLE II

APRILTAG SETUPS.

	Printing	Deployment	Surface material
AprilTag*	Laser (2400×2400 dpi)	Adhesive sheet	Mirror
AprilTag^1	Laser (2400×2400 dpi)	Adhesive sheet	Plastic
AprilTag^2	Inkjet (1200×600 dpi)	Tape + printing paper	Plastic

2) Evaluation Metrics: Three metrics have been introduced to measure our algorithm perform. i) The first is the feature detection rate γ in (27). ii) The second is the standard reprojection error ϵ of stud centers using the estimated camera pose. For the AprilTag, we calculate the reprojection error of the 4 corners using the estimated camera pose. iii) The third is $\lambda_{\max}$, the maximum eigenvalue of camera pose matrix (26).

3) Experimental Results: We have collected 100 images for both the LEGO baseplate and the AprilTag on each camera. Tab. III shows the experimental results. The upper part of the table are the results using Do3Think camera, and the lower part of the table are the results of the iPhone camera. In both cameras, the average of γ, Avg(γ), shows that combining geometric features with semantic features (GS, GS+B and GS+BG) significantly increases feature detection rate. Also, our bias removal step and utilizing grid structure to remove noise have been very effective, which can be shown from the reduce of the average reprojection error. The accuracy of camera pose estimated using the iPhone camera is better than the results of the Do3Think camera. It is expected since mobile phone cameras these days often have strong edge sharpening feature, which can benefit the LEGO baseplate feature detection. The overall algorithm design (GS+BG) can detect 95.0% stud centers when using the iPhone camera. The same trend is carried over to average reprojection error, standard deviation of reprojection error and $\lambda_{\max}$ in both cameras, which is not surprising because the number of features recovered almost directly impact estimation algorithm accuracy.

TABLE III

EXPERIMENTAL RESULTS OF STUD FEATURE TESTS. ϵ IS IN PIXELS.

Device	Feature	Avg(γ)%	Std(γ)%	Avg(ϵ)	Std(ϵ)	Avg($\lambda_{\max}$)
Do3Think	G	63.3	**7.8**	1.41	0.85	2.95×10^{3}
	S	77.0	25.3	1.07	0.60	1.53×10^{3}
	GS	83.5	13.0	1.43	0.85	1.11×10^{3}
	GS+B	85.5	13.3	1.22	0.72	1.11×10^{3}
	GS+BG	**89.0**	10.9	**0.75**	**0.51**	$\mathbf{2.15 \times 10^{4}}$
	AprilTag*	—	—	**0.30**	0.32	6.01×10^{4}
	AprilTag^1	—	—	2.29	2.01	3.26×10^{2}
	AprilTag^2	—	—	3.81	3.77	1.28×10^{1}
iPhone	G	81.1	12.1	1.13	0.69	2.58×10^{3}
	S	83.1	14.3	0.68	0.37	1.15×10^{3}
	GS	93.2	**11.0**	1.29	0.73	1.49×10^{3}
	GS+B	93.5	11.7	0.88	0.60	1.58×10^{3}
	GS+BG	**95.0**	11.7	**0.48**	**0.33**	$\mathbf{2.49 \times 10^{4}}$
	AprilTag*	—	—	**0.18**	0.17	1.40×10^{3}
	AprilTag^1	—	—	1.25	1.08	1.76×10^{2}
	AprilTag^2	—	—	1.72	1.66	4.26×10^{2}

For the AprilTag, we only consider the reprojection error and the maximum eigenvalue of camera pose matrix since the AprilTag is composed by black and white squares, which makes the detection less challenging. The average reprojection error of the ideal setup: AprilTag* performs slightly better than our GS+BG in both cameras. This is expected since we are limited by LEGO manufacturing accuracy and the 4 corners are well-defined by the boundary of the AprilTag, which provides good point configuration for the camera pose estimation.

In fact, AprilTag^1 and AprilTag^2 which represent the regular scenarios of AprilTag usage by users without computer vision background or lack of precise printing and mounting methods. The pose estimation results cannot achieve that accuracy of AprilTag* due to tag deformation caused by printing and deployment defects. The average reprojection error of AprilTag^1 and AprilTag^2 is significantly larger than the average reprojection error of GS+BG in both cameras: Avg(ϵ) of GS+BG is about 20% of AprilTag^2 in the Do3Think camera and about 28% of AprilTag^2 in the iPhone camera. In fact, 0.48 pixels in reprojection error means 0.044 mm in our experiment setup, and it also shows that our algorithm has performed well in camera pose estimation.

The results show that the benefits of using LEGO baseplate

as the artificial landmark since we do not need to concern the printing and deployment due to high precision in manufacturing and high rigidity of LEGO baseplate.

VI. Conclusion

We reported a camera/robot pose estimation algorithm design based on the LEGO baseplate recognition. The design attempts to enable the usage of widely-available LEGO baseplate as artificial landmarks in robotic or AR applications. Our algorithm leveraged both geometric features, e.g. circular stud centers, and semantic features, "LEGO" text on each stud, exploited the grid pattern, to recover features and estimate camera poses. The experimental results confirmed that our design was successful. Our experiment results confirmed that our methods produce significantly more accurate camera pose estimation results when deployed by users without computer vision background.

Acknowledgment

We thank Z. Shaghaghian for her insightful discussion. We are also grateful to D. Wang, A. Kingery, A. Angert, F. Guo, Y. Jiang, and C. Qian for their inputs and feedback.

References

[1] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4193–4198.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, 2nd Edition*. Cambridge University Press, 2004.

[4] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[5] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," Research Report, 2007. [Online]. Available: https://hal.inria.fr/inria-00174036

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[7] S. Yeh, Y. Lu, and D. Song, "Model quality aware ransac: A robust camera motion estimator," in *IEEE/RSJ International Conference on Intelligent Robots (IROS), Las Vegas, NV*, Oct. 2020.

[8] C. Harris, M. Stephens *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[9] J. Shi *et al.*, "Good features to track," in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*. IEEE, 1994, pp. 593–600.

[10] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[11] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[12] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*. Springer, 2006, pp. 404–417.

[14] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.

[15] Y. Lu, J. Lee, S.-H. Yeh, H.-M. Cheng, B. Chen, and D. Song, "Sharing heterogeneous spatial knowledge: Map fusion between asynchronous monocular vision and lidar or other prior inputs," in *Robotics Research*. Springer, 2020, pp. 727–741.

[16] Y. Lu and D. Song, "Visual navigation using heterogeneous landmarks and unsupervised geometric constraints," in *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 3, June 2015, pp. 736 —- 749.

[17] I. Loevsky and I. Shimshoni, "Reliable and efficient landmark-based localization for mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 520–528, 2010.

[18] G. Jang, S. Lee, and I. Kweon, "Color landmark based self-localization for indoor mobile robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 1037–1042.

[19] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99)*. IEEE, 1999, pp. 85–94.

[20] M. Fiala, "Artag, a fiducial marker system using digital techniques," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2. IEEE, 2005, pp. 590–596.

[21] D. Wagner and D. Schmalstieg, "Artoolkitplus for pose tracking on mobile devices," 2007.

[22] L. Baronti, M. Dellepiane, and R. Scopigno, "Using lego pieces for camera calibration: a preliminary study." in *Eurographics (Short Papers)*. Citeseer, 2010, pp. 97–100.

[23] T. V. Do and J.-W. Lee, "A multiple-level 3d-lego game in augmented reality for improving spatial ability," in *International Conference on Human-Computer Interaction*. Springer, 2009, pp. 296–303.

[24] T. Engelke, S. Webel, and N. Gavish, "Generating vision based lego augmented reality training and evaluation systems," in *2010 IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 2010, pp. 223–224.

[25] W. Yan, "Augmented reality applied to lego construction: Ar-based building instructions with high accuracy & precision and realistic object-hand occlusions," *arXiv preprint arXiv:1907.12549*, 2019.

[26] H. Taira, I. Rocco, J. Sedlar, M. Okutomi, J. Sivic, T. Pajdla, T. Sattler, and A. Torii, "Is this the right place? geometric-semantic pose verification for indoor visual localization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4373–4383.

[27] N. Merrill and G. Huang, "Calc2. 0: Combining appearance, semantic and geometric information for robust and efficient visual loop closure," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4554–4561.

[28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.