applied sciences

MDPI

Article

Visual-Based Localization Using Pictorial Planar Objects in Indoor Environment

Yu Meng [1,2,*], Kwei-Jay Lin [2,3], Bo-Lung Tsai [2], Ching-Chi Chuang [3] and Yuheng Cao [2] and Bin Zhang [1,*]

1 College of Computer Science and Engineering, Northeastern University, Shenyang 110819, China
2 Department of Electrical Engineering and Computer Science, University of California, Irvine, CA 92697, USA; klin@uci.edu (K.-J.L.); blt@uci.edu (B.-L.T.); yskyerda@uci.edu (Y.C.)
3 NTU IoX Center, National Taiwan University, Taipei 10617, Taiwan; C.Chuang-1@student.tudelft.nl
* Correspondence: mengyu@stugmail.neu.edu.cn (Y.M.); zhangbin@mail.neu.edu.cn (B.Z.)

Received: 30 September 2020; Accepted: 24 November 2020; Published: 30 November 2020

Abstract: Localization is an important technology for smart services like autonomous surveillance, disinfection or delivery robots in future distributed indoor IoT applications. Visual-based localization (VBL) is a promising self-localization approach that identifies a robot's location in an indoor or underground 3D space by using its camera to scan and match the robot's surrounding objects and scenes. In this study, we present a pictorial planar surface based 3D object localization framework. We have designed two object detection methods for localization, ArPico and PicPose. ArPico detects and recognizes framed pictures by converting them into binary marker codes for matching with known codes in the library. It then uses the corner points on a picture's border to identify the camera's pose in the 3D space. PicPose detects the pictorial planar surface of an object in a camera view and produces the pose output by matching the feature points in the view with that in the original picture and producing the homography to map the object's actual location in the 3D real world map. We have built an autonomous moving robot that can self-localize itself using its on-board camera and the PicPose technology. The experiment study shows that our localization methods are practical, have very good accuracy, and can be used for real time robot navigation.

Keywords: indoor localization; visual-based localization; picture matching; computer vision; object detection

1. Introduction

Device localization is an important technology for future distributed indoor IoT applications with autonomous moving devices, such as industrial robot, intelligent hotel assistant, and smart office courier. Next generation smart services must be location-based and self-location-aware. For example, smart movers in large warehouses can help workers collect customer orders from different shelves into one shipment package. Mobile hospitality assistants can lead customers in different corners of big hotels or shopping centers to specific locations or give directions. Autonomous moving couriers can be used in offices to deliver documents and supplies without human contacts. Many distributed devices can be deployed together to provide a coordinated smart service network.

Visual-based localization (VBL) [1,2] is one of the promising self-localization technologies that have received a growing interest. VBL identifies a device's location in a target space by using its camera to see the device's surroundings, without the dependency on GPS which is designed for outdoor usage and also subject to signal bouncing and interference. Therefore, VBL is especially attractive for devices that must work indoor, underground, or in an enclosed space. Moreover, with the advancement and popularity of low-cost, low-energy, high-quality cameras, VBL has become very attractive and easy to

adopt. Many indoor devices and smart equipments can be equipped with small cameras and instantly become location-aware by connecting them to servers for VBL processing.

1.1. Visual Matching and Mapping

Different approaches and methods for producing the location information from camera captured images have been developed. Research projects on VBL [1] have shown a big diversity on the tradeoff between operation simplicity and result quality. VBL methods can be divided into two general classes: matching only methods and matching with mapping methods.

Matching only VBL can be modeled as an image look up and retrieval problem [3]. This class of methods produces a camera's current location by searching for similar images from a known image collection when compared with scenes of the camera view. The camera location is given by the image(s) with the most resemblance to a view using image matching techniques. The location is a simple lookup without any mapping or positional conversion. Due to the limited precision, this class of methods can be used by some simple, approximate context-aware applications, such as mobile scenic guide and ROI (region of Interest) searching.

To achieve more accurate localization, researchers have studied VBL using the matching with mapping approach. Such methods generate the precise location and orientation of a camera by mapping the camera view to a 3D position using the spatial relationship between the camera and its view on the 3D map. The 3D map may be built either prior to runtime or on-the-fly depending on the application scenario. The location produced by such mapping-based methods can be very precise due to the adoption of sophisticated 2D-to-3D conversions. These methods are better for indoor navigation and route planning of moving devices since the location produced from VBL is a precise 3D coordinate on the 3D map.

1.2. Object-Based Mapping

Extracting from a camera view the image pixels corresponding to certain positions in a 3D map is the first step for mapping-based localization. It has two subproblems: what information it can gather from a captured view, and how to map the view information to the actual 3D map.

Image information identification and extraction can be done by pixel-based methods or object-based methods. Pixel-based methods try to detect significant feature pixels from the camera view to match with features of some images in the image collection. They usually report only the simlarity level between them without using any 3D mapping. In comparison, object-based methods try to identify some actual objects in their views, such as tables and chairs. Each object has a 3D model with an actual size which can be used to build a 3D map. Using real objects' shapes and sizes, object-based methods can derive the distances between a camera and visible objects in the 3D model. As long as a camera can accurately see and identify known objects, these objects' 3D models provide the intermediary knowledge between the camera view and its real world 3D location.

Object detection and 3D matching are therefore the foundation technologies for object-based localization's accuracy and efficiency. Some earlier projects [4–6] have studied 3D model objects, such as tables and buildings, as objects for detection, map building and matching. However, 3D object model building and matching have a high computational complexity and are very time consuming. It may be difficult for low-cost IoT devices to support such computations.

For fast detection and simplified object model building, artificial or computer-generated objects have been designed in some projects [7–9]. These artificial, easy-to-identify objects (Figure 1a) can be placed in an environment for a camera to easily detect them as references for localization. However, many artistic or high-end indoor places, such as museums, galleries and hotels, may not want to put such artificial objects on their walls. Using natural or artistic objects is more desirable for VBL technologies.

Our VBL study for natural object identification and mapping proposes to use 2D matching models rather than 3D models, since there are many 2D image matching techniques much more efficient than

3D techniques. Moreover, many natural objects have some planar surface, e.g., books, furnitures and wall-hanging art pieces. We can identify an object using only its planar surface without taking its complete 3D features and model. By using 2D planar object detection and then applying 2D planar to 3D object construction, the 3D position of a carefully modeled object can be produced efficiently as long as its planar surface can be correctly detected and located.

(**a**) ArUco (**b**) ArPico (**c**) PicoPose

Figure 1. Planar object examples.

1.3. Pictorial Planner Object Based Localization

In this paper, a pictorial planar surface based VBL framework is presented for producing the 3D pose of indoor devices with a camera. An example robot with our VBL support is shown in Figure 2; the robot can localize itself whenever some pre-trained pictorial planar surface is visible by its camera. We have designed and implemented two object detection methods for centimeter-precision-level localization, ArPico and PicPose (Figure 1b,c). Our methods are motivated by an open-source project, ArUco [10], which uses fiducial markers (Figure 1a) for AR processing. ArUco produces 2D barcodes with a black frame and an inner grid of binary blocks. The black frame is used to detect the marker, and the inner blocks are used to recognize its ID.

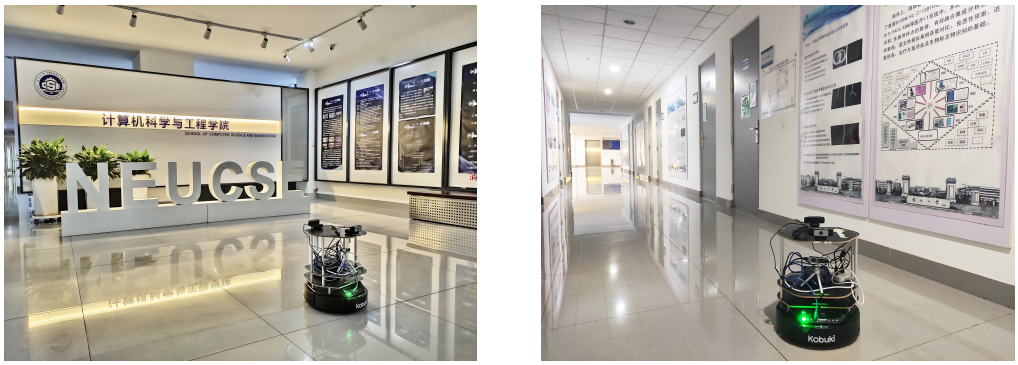


Figure 2. Robot navigating using PicPose localization.

The pictorial planner object based VBL framework presented in this paper has three components: (a) Offline object learning and library creation; (b) libraries and map management; and (c) real-time device localization. This framework is applicable to both ArPico and PicPose methods. The ArPico method detects and recognizes framed pictures (Figure 1b) before converting pictures into vectors of binary blocks (similar to ArUco markers) for matching with known marker codes in a library [11] . It then uses the picture's corner points to identify the camera's pose relative to the picture's actual position stored in the 3D map. In contrast, the PicPose method detects the pictorial planar surface (Figure 1c) of an object without requiring it to have any specific shape or frame. PicPose is designed to extract feature points of a pictorial planar object from a camera view, and produce the camera's pose

by matching those feature points on the planar surface of a 3D object with its real-world 3D models stored in the 3D map. Compared with some earlier localization works, our VBL methods have the following benefits:

- Our methods only use a simple monocular camera, no need for high-resolution or multiple cameras. It is a low cost solution for localization.
- Our methods can learn to recognize any user-selected pictorial object, not restricted to system-generated fiducial markers or patterns. It is a flexible solution for any user space.
- The PicPose method can produce a successful localization with a partially visible picture, and with any known shape. It provides a robust solution in the practical setting.
- Our methods are not subject to radio interference or blocking concern, or other sensing signal disturbances. It provides an ideal indoor and underground solution for localization.

The contribution of this paper is that we present an attractive localization support that uses existing natural pictorial objects in a target environment as pictorial planar markers. Our approach is novel since we do not place system-generated fiducial markers in an environment [10] but instead pre-learn distinctive objects already present in the environment and use them for producing robot localization. It is similar to the human localization perception strategy of referencing to known visual objects from our past visual knowledge. Our approach is marker-based but uses pre-runtime object learning to collect good strong markers from the operational environment for runtime recognition and to produce localization.

To improve the localization speed and precision, we have designed two new techniques to improve the efficiency and precision of pictorial planar object recognition. We have developed an algorithm to reduce the number of extracted feature points from a robot view before matching them to a pictorial object in the library. By inspecting the relative positions of feature points and selecting just a sufficient number of useful points, we can reduce the number as well as improve the quality of feature points for picture matching. Our study shows that this technique can reduce the average matching time by 50%. Another new technique is for filtering out weak matched feature point (FP) pairs. Two filters are designed to inspect the matching result produced by FLANN and can greatly reduce the number of useful matched FP pairs before we use them in the homography calculation. These filters can drastically decrease the FP matching output (e.g., from 1656 pairs to 48 pairs in an example in Section 5.3). This results in a big saving of the average pose homography calculation time (from 30 ms to 1 ms, or about 95% improvement in our experiment). Both new techniques can significantly improve state-of-the-art object matching technologies.

The rest of this paper is organized as follows: Related works on visual based localization are presented in Section 2. Section 3 shows the design of our planar surface based 3D object localization framework and the main components in it. Based on this framework, in Section 4, we present the ArPico method which converts framed pictures into ArUco-like markers for localization. Section 5 shows the design of PicPose. It first detects pre-learned picture from camera view using deep learning approach, and then extract feature points from the picture to produce the camera pose. We have built a moving robot that uses ArPico and PicPose for localization. The performance data collected from our prototype and VBL study are presented in Section 6. Section 7 concludes the paper.

2. Related Work

2.1. Marker-Based Localization

Unlike outdoor devices, indoor devices cannot use satellite-based positioning systems, i.e., GPS, to get precise locations due to poor satellite radio signal reception inside of buildings. To implement indoor localization, many projects have tried to deploy indoor local active or anchoring devices sending various signal media, such as ultrasound, ultra wide band (UWB) radio, Wireless Fidelity (WiFi), etc. for individual devices to receive and compute their locations. One issue with radio and sound signals is that they are subject to signal bouncing, noisy transmission, and interference from random

signals generated by other indoor objects like microwave ovens and washers. Such signals can make localization very imprecise. In comparison, line of sight is usually much less susceptible to external inteference. As long as a subject is visible, VBL can use it as an information source for localization.

Using cameras to produce localization has been shown to be an effective and low-cost approach [1,2]. The location of a camera can be derived by using relative positions to some pre-located visible objects or surfaces with known patterns. This is the foundation for marker-based localization for indoor devices. Several marker-based pose estimation approaches have been proposed to calculate the location of a camera using specific 2D markers [8,12,13]. Two-dimensional markers used for camera localization are extensively designed as monochrome codes with specific patterns for fast detecting and recognizing [14–16]. These methods have high efficiency on detecting and recognizing markers from a camera view. Their technologies have been used in 3D indoor model building [17], location estimation and navigation for UAV or robots [9,18,19]. One important goal for the projects is that markers are designed to have specific patterns so that systems can identify the marker patterns and poses very efficiently. However, the markers (e.g., Figure 1a) produced by specially designed algorithms may look odd in most indoor living environments. Moreover, a camera is required to fully capture a marker for correct detection. The detection would fail if a marker is only partially visible in a camera view. Our work in this paper has addressed the issues of unnatural markers and marker occlusion.

2.2. Marker-Less Visual Based Localization

In order to avoid using unnatural markers for localization, other works have studied how to extract key information from natural camera views to calculate the camera pose. It can be divided into two categories of approaches.

One is the frame retrieval approach using pixel feature extraction, such as SLAM [20,21], InLoc [22]. In these approaches, the camera frame, which contains the feature points used to construct a 3D map, is stored to build a key-frame library. By matching the camera frame with the key frames, the feature points in current frame can be mapped into the 3D map to estimate the pose of the camera.

Their advantage is that the features or objects used for real-time frame matching are not pre-trained so that it can be applied in any unknown environment. On the other hand, the historical map is not able to be reused when there is something changed in the environment since it relies on the key frame library; the map may need to be re-built if there is any view change in the target space. The camera can gain its pose only when the camera orientation is similar to that used in map building as the features of the environment in other orientations are impossible to be captured and detected from the historical frames. Moreover, the camera cannot get real-world scale from marker-less based localization since the real scale of the point cloud is unknown in the 3D map.

The other approach is visual object based localization that calculates camera pose from the object detected in the view. Some researches designed object detection and recognization algorithms [23–26] to extract significant points from camera view frames to match with that in the pre-built model for efficient pose estimation. However, they are not always reliable when one or more similar objects appearing in a frame and when the object are not completely captured by the camera. To achieve higher accuracy and robustness, DNN (Deep Neural Network) has been involved in more and more research [27,28]. However, the exact pose of the camera cannot be computed as it is hard to gain the detailed outlines or poses of the objects by from DNN approach. In our work, we have added the capabilities to find the detail outlines of identified objects in order to calculate their exact positions in a 3D map model.

2.3. Two-Dimensional vs. Three-Dimensional Object Model and Identification

Three-dimensional object recognition is to identify and produce the 3D information, such as pose and shape, of some objects from a picture frame or video stream. Many previous works have designed algorithms to identify 3D objects in pictures [4–6,29]. However, most 3D object recognition algorithms have high computational complexities and require large memory space. For example, [30]

recommends systems with 200 GB free storage, 10 GB memory and 5 GB of GPU memory for running their programs. It may be too costly for low-end IoT devices.

In comparison, 2D picture recognition is a problem with much more efficient solutions. Unlike 3D objects, 2D pictures usually do not deform in camera views and its features do not suffer physical change when captured from different angles since it is a planar surface. Therefore, the complexity of 2D picture detection and matching algorithms is much lower than 3D object recognition. Traditional 2D pictures matching approaches are usually based on color thresholding [31,32] or corner/contour feature extraction methods [33,34]. These matching methods have a high efficiency and accuracy on pictures that have similar sizes and simple backgrounds. However, they may have problems when a picture has illumination change, motion blur and complex background in the view since both color and texture features may be different from the original picture. It is especially difficult when there are other similar pictures or objects inside of a camera view since it may not differentiate similar objects [35].

To enhance the robustness of picture detection and matching, researchers have recently worked on training 2D pictures as 2D objects using deep learning based methods [36]. Multiple photos that capture the target 2D picture under different angles, illuminations, backgrounds and motions are used as samples for training. In general, deep learning based 2D picture detection has a higher robustness on picture detection than traditional methods, whereas traditional methods have better capabilities on extracting detailed pixel-level patterns and features from a camera view.

2.4. VBL Methods

A comprehensive comparison of different VBL methods is given in Table 1. Compared with matching only and pixel-based localization methods (top two rows of Table 1), ArUco, ArPico and PicPose can produce more precise locations with a real scale output. Among planar object methods, ArUco markers are pre-coded black-and-white cells, ArPico detects pictures that have a clear rectangular border, while PicPose can handle all planar objects without any border or shape restriction. Moreover, PicPose is able to use partially visible objects for localization while the other planar object based methods cannot. Due to the neural-network based object detection complexity, the total processing time of PicPose is longer than ArPico, but still reasonably short (<100 ms in our project) to support real-time localization of moving devices. We will present detailed performance comparisons in this paper.

Table 1. Comparison of Visual-Based Localization (VBL) Methods.

Method		Precision	Speed	Planar Object	Object Shape	Object Content	Actual Scale	Partial View
Pixel Matching Only (CNN [37,38])		Low	∝ size of image library	-	-	-	No	Yes
Pixel Matching and Mapping (ORB-SLAM [39,40])		Medium	Medium	-	-	-	No	Yes
Planar Object Based	ArUco ([8,12])	High	High	Yes	Square blocks	Patterned marker	Yes	No
	ArPico	High	High	Yes	Clear border	Framed picture	Yes	No
	PicPose	High	Medium - High	Yes	Any	Any picture	Yes	Yes

3. Pictorial Planner Object Based VBL

In this section, we present an overview of our pictorial planner object based VBL framework as shown in Figure 3. The framework has 3 stages: offline object learning and libraries building, libraries and map management, and real-time device localization.

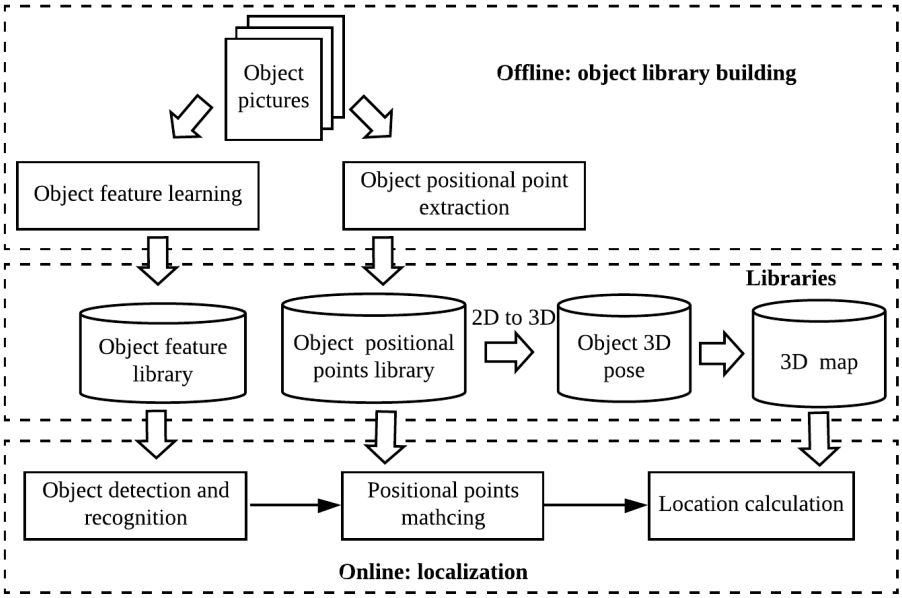


Figure 3. Pictorial Planner Object Based VBL Framework.

The object library building stage is the offline preparation process for the localization support. The VBL system scans all object pictures captured by cameras from different positions and angles, and generates the object feature libraries to be used for object detection. There are two main object learning components, object feature learning and object positional point extraction. The features on the planar surface of each object used for detection are learned from object picture collections. In addition, to calculate the pose of a recognized object, we also need to identify certain pixel points on the object picture to match and compare with their relative positions in the original object picture. The object positional point extraction part is used to find the positional points that not only have a unique feature but also can be used for mapping from a 2D picture surface to its 3D model. For instance, the positional points of a rectangular picture are its four corner points of the rectangle, and for other shapes, either some corner points on the picture or special ORB feature points on opposite sides of the surface.

Three object libraries are used in our VBL framework: the object feature library, the positional points library, and the 3D object model. The object feature library stores the features of each object produced from the object feature learning component. Among the features, certain positional points of each object are identified and stored in the positional points library. Each positional point has a corresponding 3D point in the object's 3D model. The coordinate (x, y, z) of each point in the 3D model is represented by the actual scale of the object. The x and y are measured by the real scale from the positional point in the original object picture. The z value is 0 in the 3D model since all objects to be matched in our framework are flat surfaces. Unlike image sequence libraries used in other localization methods like SLAM, the total data size of the 3 libraries is small and can be stored in most mobile IoT devices, or small edge servers if ncessary.

For each target area, the 3D map of known objects in the area must be built before a device localization can be conducted. We can use some offline tools to record each object's 3D location in the real world environment. The ArUco project [10] has built an easy-to-use video analytic tool that can automatically map ArUco markers in a 3D area by shooting videos to link the relative positions of neighboring markers. We can extend such a tool and replace ArUco markers with pictorial planars to build the 3D map for all objects with planar surfaces. The 3D map will be looked up during the localization process.

The online localization layer is designed to conduct object detection and recognition, positional points matching and location calculation. Firstly, the planar object must be detected from the camera

view. After an object recognition, another image processing component will extract the positional points for the object, and then match them with the corresponding positional points stored in the library. If the matching is successful, the location of the camera relative to the object can be estimated by calculating the translation relationship between the positional points on the camera view and those in the original 3D model.

Based on this framework, we have designed two object detection and localization methods, namely, ArPico and PicPose. They are presented in the next two sections. On the other hand, the VBL framework presented in this section is general and applicable to other planar object based localization methods.

4. ArPico VBL

The goal of object detection and recognition in the localization process is to identify a likely object in the camera view, and match it with any of the known objects stored in the library. Object identification algorithms only need to differentiate a detected object from all others in the dictionary rather than recognizing all detailed features. Therefore, A simple picture transformation method, ArPico, is proposed for fast detection and identification by converting colored pictures (Figure 1b) to 2D barcode-like binary markers.

4.1. ArPico to ArUco Transformation

An example of the ArPico marker conversion for pictorial objects is shown in Figure 4. ArPico are always displayed with a black frame (Figure 1b). This black frame should be cropped off from the picture before the marker identification (the upper-left picture on Figure 4). Then convert the inner picture to greyscale (the upper-right picture on Figure 4), and divide it into blocks. In each block, we use the average of grey value as its feature, thus an average grey block matrix is created (the lower-left picture on Figure 4). The median of the matrix is set as the threshold to define each block as 0 or 1 (black or white). In this way, an ArPico pictorial object is converted to a 2D code pattern (the lower-right picture on Figure 4) just like an ArUco marker (Figure 1c).

Figure 4. ArPico Marker Conversion Example.

4.2. ArPico Object Properties

Each ArPico pictorial object captured in a camera view will be compared with all markers in a library to see if it matches with one of them. Due to the light and visibility conditions, sometimes a captured picture may not be very clear. It is important to have clearly distinguishable markers in an ArPico library to reduce erroneous identifications. In this section, we present some desirable properties for ArPico markers. Any picture that cannot meet these conditions should not be used for ArPico VBL.

Before a pictorial object is added into ArPico library, its distinctiveness and uniqueness are required to be verified using the binary marker code generated in Section 4.1. Then we check the self-transition T and rotation difference $D(m_i)$ of the binary code to ensure the marker is unique in four 90-degree rotations. Meanwhile, it also need to compare with other existing codes in the dictionary to minimize the possibility of detection errors.

The self-transition T represents how many pairs of neighbour binary codes in a pictorial object have different values. A good code should have many transitions from 0 to 1 or from 1 to 0 that give it many significant features. Assuming the binary block is a $n \times m$ matrix, the self-transition T can be represented by:

$$T = \sum_{x=1}^{n} \sum_{y=1}^{m-1} P(x,y) \otimes P(x,y+1) + \sum_{x=1}^{n-1} \sum_{y=1}^{m} P(x,y) \otimes P(x+1,y) \quad (1)$$

where P(x,y) is the binary code value at block (x,y). T is a threshold to inspect if the code is unique enough.

The rotation difference $D(m_i)$ is used to check the differences of the picture when it rotates 90, 180 and 270 degrees.The rotation difference $D(m_i)$ can be defined as:

$$D(m_i) = \min_{m_k \in A(m_i)} \{H(m_i, m_k)\} \quad (2)$$

where $H(m_i, m_j)$ is the hamming distance of two markers and $A(m_i)$ is a set of rotation markers of m_i. The rotation difference $D(m_i)$ need to be large enough so that the marker can be detected correctly in different rotations.

The last step is to check the global uniqueness of the picture. In order to avoid detection mistakes, the humming distance between the new candidate and the other existing pictures in the library needs to be larger than a threshold. After the marker can pass this test, the pictorial object will be included in the object library; otherwise it will not be used for ArPico VBL.

4.3. ArPico Image Processing

Figure 5 shows the workflow of the ArPico image processing. When an ArPico pictorial object is detected in a camera view, the detected image may be rescaled to improve the computing efficiency. Resizing the detected picture is to make it to have a lower resolution, as long as it can show some recognizable ArPico objects, and also to avoid many trivially invalid objects in the camera view to improve the computational efficiency. Once the picture with optimal scale is generated, we will find all areas with border contours in the frame image and produce all pictorial object candidates surrounded by some border contours. Some of those candidates may be recognized as valid ArPico objects included in the ArPico object library.

Among all pictorial object candidates produced from an image, some of them can be easily removed from consideration. We apply two filters to get reliable candidates. First, the undersize candidate will be filtered out as they may not contain enough information for recognition. We use the candidate's perimeter to determine which one is smaller than an acceptable size.

The second filter is designed as a minimum distance between a candidate's corners and the camera frame's edges. It will have high distortion and information losing when the picture is captured near the frame edge. So we remove those candidates which are located closed to the frame edge. Suppose an image's width is w pixels, its length is l pixels. Let the distance ratio be r. We have the following constraints for every pixel (x, y) in a valid candidate:

$$w \cdot r \leq x \leq w - w \cdot r \quad (3)$$

$$l \cdot r \leq y \leq l - l \cdot r \quad (4)$$

After the filtering, we will convert candidate objects to their binary codes and check if some of them can be matched with a valid marker in the library. The acceptable maximum error between a candidate and its potential marker in the library can be represented as:

$$T\lfloor (l-1)/2 \rfloor - 1 < m \leq T\lfloor (l-1)/2 \rfloor \quad (5)$$

where $\lfloor (l-1)/2 \rfloor$ is the smallest hamming distance between any two markers in the library. T is a tolerance rate with the range of $(0, 1)$. If the hamming distance between a candidate and a existing marker is smaller than m, the candidate will be identified as that marker.

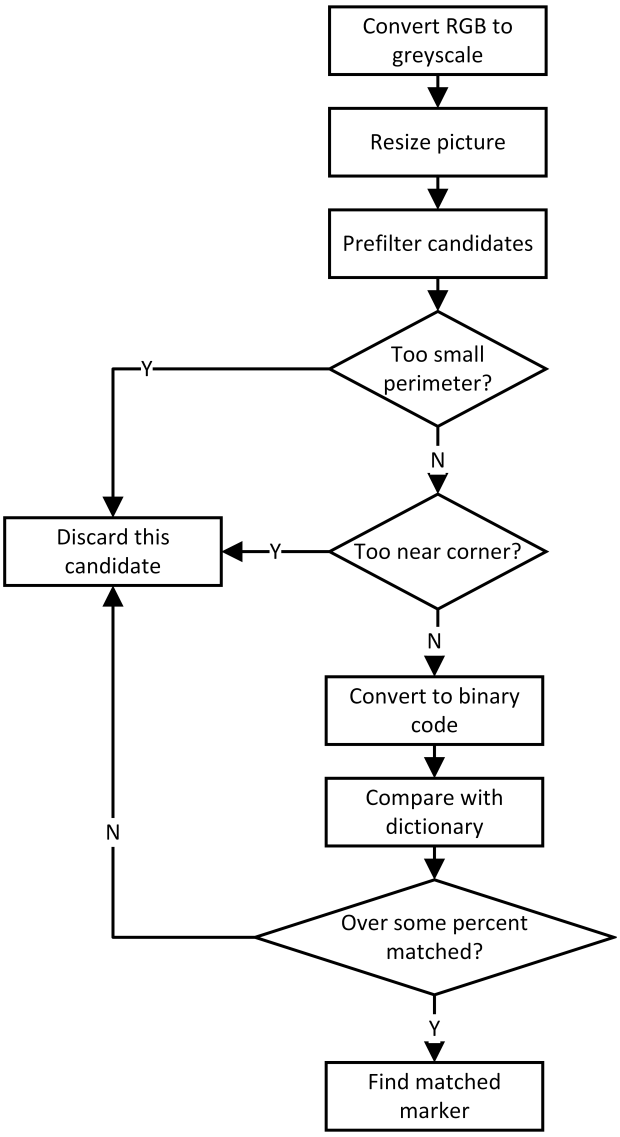


Figure 5. ArPico Image Processing Workflow.

5. PicPose: A Picture-Based Pose Method

To recognize pictures without borders and utilize their fine visual details, we have designed the PicPose planar object (Figure 1c) based pose technology. PicPose matches the feature points extracted from an object, which is detected from a camera view, to a corresponding original picture in the library for pose calculation.

The PicPose workflow has four steps (Figure 6): planar object detection, feature extraction, feature matching, and pose calculation. The planar object detection component is to detect re-trained objects from a camera view. In order to obtain a high accuracy and robustness on object detection, a deep learning based object detection technology, SSD, is employed. The second step is to extract the detailed feature points from detected objects. Feature matching is the third and pivotal component for pose estimation as a translation matrix describing pose variation from a model to a detected object is generated. The quality of feature points matching determines the accuracy of the last step, pose calculation.

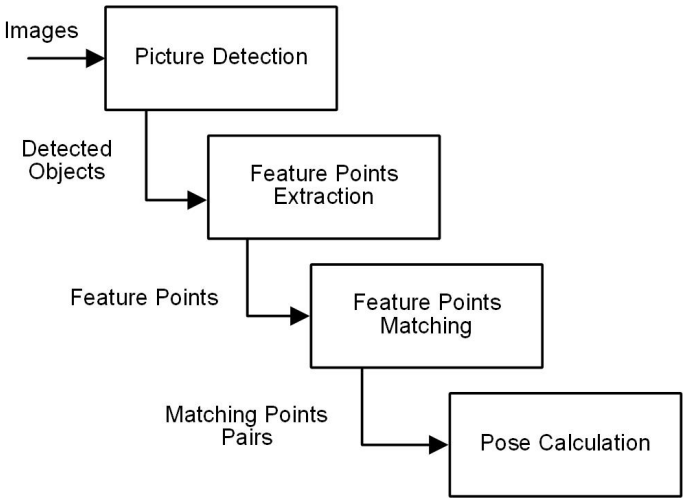


Figure 6. PicPose Workflow.

In this section, the detail of each conponent is presented using the PicPose workflow.

5.1. PicPose Picture Detection

The challenge of PicPose picture detection is to locate known planar objects in the image captured by a camera. PicPose pictures has no clear borders thus cannot be detected using simple border detection algorithms. Therefore, image learning technology is used in PicPose for picture detection. We adopt an object detection framework based on Single Shot MultiBox Detector (SSD). SSD, however, brings a higher demand on the computing capability. Therefore, we decide to use the lightweight neural network model MobileNet, which consists of streamlined separable convolutions, as the training and regression network in this project.

5.1.1. Offline Dataset Preparation and Model Training

Pictures which are used for pose calculation in a target environment must be involved in network training before runtime detection. The dataset is composed of two parts, images and their attributes describing the information of target pictures in the image, such as corner location, labels and size. In practice, images for the dataset can be chosen from a series of photos or videos in which the target pictures are captured with different backgrounds and angles to enhance the training robustness. MobileNet-SSD applied with Caffe deep learning framework is implemented with pre-trained weights network VOC0712. Several parameters need to be configured before training. The most important parameter is the batch size that indicates the number of target examples utilized within one iteration during training. It greatly affects convergence speed, computing time of each iteration, and training accuracy. In general, a batch size of 32 is a good starting point before increasing it to 64, 128, and 256. Other batch sizes (lower or higher) may be used for specific datasets since it also depends on the size of dataset.

5.1.2. Online SSD Detection

The bounding boxes that contain detected pictures are produced using Caffe-based SSD when the trained neural network model is applied. Each bounding box has four coordinates, $(P_{1x}, P_{1y}), (P_{2x}, P_{2y}), (P_{3x}, P_{3y}), (P_{4x}, P_{4y})$. If there are many target pictures detected in a camera image, picture i has four corner coordinates: $(P^i_{1x}, P^i_{1y}), (P^i_{2x}, P^i_{2y}), (P^i_{3x}, P^i_{3y}), (P^i_{4x}, P^i_{4y})$.

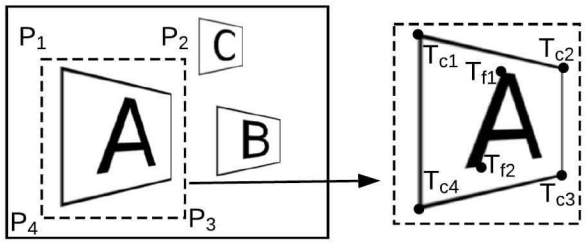
Computational speed is always the pain point of deap learning based object detection in real-time services. There are two possible ways to speed up the detection. One is using hardware to provide more powerful computing capability for deep neural network algorithm such as Intel Movidius or commodity GPUs. Tuning the parameters of SSD, such as scale increasing rate, input size and batch

size, is another way not only to increase computing speed but also to improve accuracy. However, it is always a conflict of goals between accuracy and speed when tuning parameters. For example, if the detection speed is unable to ensure the efficiency of the whole system for real-time localization, some parameters must be modified to increase speed such as reducing the input image resolution, even though it may affect the accuracy of bounding boxes produced.

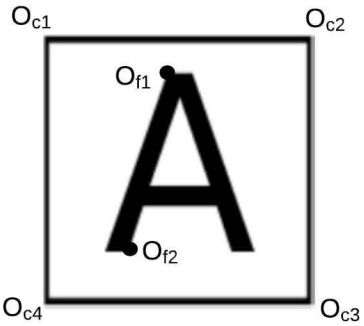
5.2. PicPose Feature Points Extraction

After a picture has been identified by SSD in an area of a camera view, we need to identify specific feature points for producing the camera pose. To find the pose in the 3D space, at least 4 non-colinear points of a pictorial object need to be extracted from the captured image to match with the corresponding points in the original picture.

The feature points identification issue is illustrated in Figure 7. The left-hand side of Figure 7a is a full image view captured by a camera, and the right-hand side shows the area of a pictorial object detected by SSD. P_1–P_4 show the detected area, inside of which we can find two types of feature points: corner points T_{ci} and internal feature points T_{fi}. Internal feature points are extracted by feature extraction algorithms and can be anywhere on a picture. Their corresponding feature points in original picture is shown as Figure 7b. In detected picture, each point T_i has one point O_i which has same relative position inoriginal picture with the detected picture. Thus, (O_i, T_i) is a matching pair which can map the point frome original picture to camera view. In order to obtain the feature point pairs as acrrcate as possible, two types of feature points extraction methods are discussed below.



(a) The pictorial object detected from a camera view



(b) The feature points in original picture

Figure 7. PicPose Feature Points.

5.2.1. Corner Extraction by Contour Detection

Figure 8 shows the flow of corner points extraction. After a potential region of target picture is detected using SSD, its region size will be enlarged to make sure it can include the picture as much as possible. If the object is too big, it may be reduced in size in order to improve the processing efficiency. We can convert a picture to become smaller as long as it has enough details for pose calculation. Once the optimal bounding box is generated, the exact contours of the picutre will be detected.

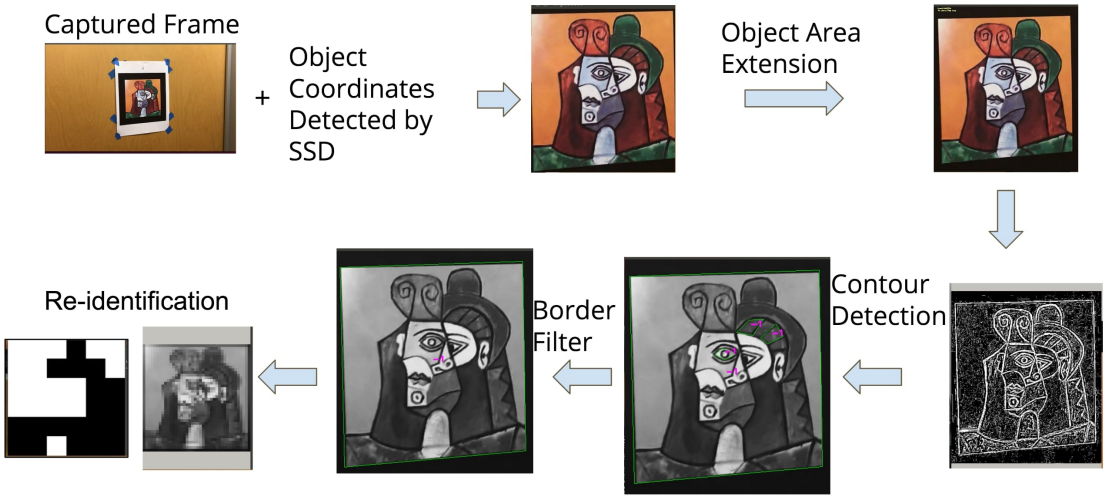


Figure 8. Corner Points Extraction.

During contour detection, many quadrilaterals on the pictorial area may be produced. We need to identify the real target picture object from these quadrilaterals. Two types of filter are used in our work. One is shape based filters. This filter removes quadrilaterals too tiny to be processed correctly and also quadrilaterals with unrealistic corner angles. The second filter is a content based filter. We use a content re-identification approach to convert a quadrilateral's picture into an ArPico marker code, and check if such a code exists in our marker librray. If we cannot find a planar object in the planar object library with the same code, we will reject that quadrilateral in the contour detection process.

5.2.2. Texture Feature Point Extraction

However, the contour of a picture may not always be detected if its edge is not clear enough to be differentiated from the background. Three common conditions may lead to a failure of corner detection. The first condition is a bluring edge which may be casued by a moving camera. The edge is also impossible to catch if a picture is captured partially from a camera. Finally, it is hard to detect if the edge of a target object is not a regular shape like square or rectangle. Therefore, using only contour deteciton technology is not good enough for extracting sufficient feature points from a moving camera for pose calculation.

Our project uses the Oriented FAST and Rotated BRIEF (ORB) feature detector to extract feature points from the original picture and corresponding feature points from a captured target picture. The architecture of proposed feature point pairs extraction model is shown in Figure 9. We first build a feature point library to store the feature extracted from original picture. The feature library will use as the ORB ground truth when mathcing the feature from camera view. The ground truth library stores picture ID, resolution and features points. There are two attributes to to describe a feature point, pixel coordinate and BRIEF descriptor. In the online process, the original picture is retrieved from the library using the picture ID recognized by SSD detector. Then we extract the feature points using ORB from the captured picture with the same parameters as those configured for generating the ground truth library. The ground truth feature points can also be obtained from the library, so that we can match the feature points of captured picture with that in the ground truth library. FLANN (Fast Library for Approximate Nearest Neighbors) is employed in our project to match the feature points of the detected picture with ground truth features.

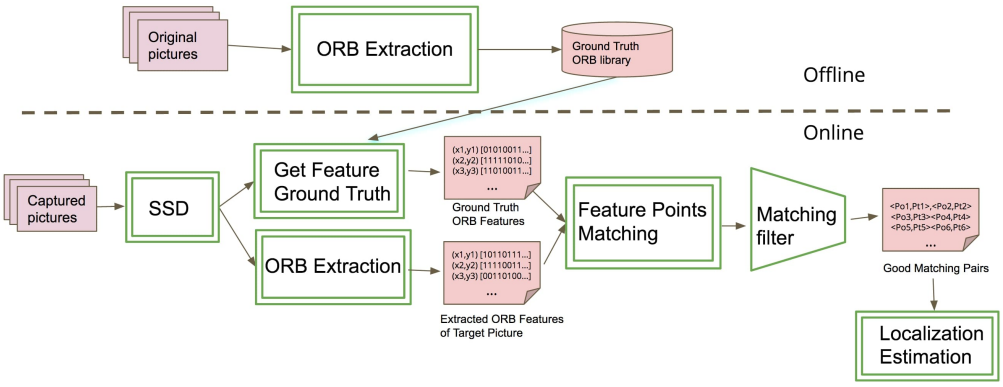


Figure 9. Feature-based Localization.

To further improve feature point extraction for a better matching, we have designed two techniques. The first one is to use a bilateral filter to avoid potential noise in the captured image. The second is to use the feature point density to reduce the number of feature points to reduce the matching time. They are presented below.

Feature Extraction with Noise Avoidance

The feature points of a picture are the prominent points in the picture, such as outline points, highlights in darker areas, and dark spots in brighter areas. In order to extract the feature points from a picture, ORB is used in our work. There are two elements that can uniquely define an ORB feature point, key point and descriptor. FAST (Features from Accelerated Segment Test), which is an efficient corner point detection algorithm, is used to detect key points in ORB. The main idea of FAST is to detect the greyscale pixel value of a circle around the candidate feature point. If there are enough pixels, which have significant difference with the candidate point on greyscale value, in the circle around the candidate point can be detected, the candidate point will be considered as a feature point. In general, the radius of the circle is 3 and there are total 16 pixels on the circle. If the greyscale value of continuous N pixels on the circumference is larger or smaller than that of the candidate point P, then P is considered as a feature. Normally N is set to 12. In order to speed up the extraction of feature points as well as screen out non-feature points quickly, the greyscale values at positions 1, 9, 5 and 13 are detected first. If P is a feature point, three or more pixel at these four locations should be larger or smaller than the grey values of P. If it is not satisfied, it will be discharged directly.

FAST is a pixel-sensitive corner point detection algorithm which may be affected by noises in an image. Bilateral filter is applied before FAST detection to avoid the noise. The bilateral filter adds the pixel value weight on the basis of Gaussian filtering. In that way, the influences of both distance factor and pixel difference are considered in the bilateral filter. The weight can be calculated as:

$$W_{x,y} = exp(\frac{-\left\|P_{x,y} - \overline{P_{x,y}}\right\|^2}{2\sigma_p^2}) * exp(\frac{-\left\|C_{x,y} - \overline{C_{x,y}}\right\|^2}{2\sigma_c^2})$$

where $P_{x,y}$ is the grey value of pixel (x,y), $\overline{P_{x,y}}$ is the mean value of the neighbours of (x,y), $C_{x,y}$ is the position of (x,y) and$\overline{C_{x,y}}$ is the position of its neighbours. σ_p and σ_c represent the standard deviations of pixel values and distances respectively. In the flat region, the weight of each pixel in the filter is similar, and the spatial distance weight dominates the filtering effect. In the edge region, the values on the same side of the edge are similar and far greater than those on the other side of the edge. At this time, the weight of the pixels on the other side has little effect on the filtering results, and the edge information is protected. When there are noise points in the flat area, the weights of the signals around the noise points are very small. After normalization, these weights are improved, and the noise points are also filtered.

After obtaining the key points, we need to describe the attributes of these feature points in some way. The output of these attributes is called feature descriptor. The feature descriptor is generated by BRIEF (Binary Robust Independent Elementary Features) algorithm in ORB. The descriptor produced by BRIEF is a set of binary codes which combines the differences of M paris of points surrounding a key point.The descriptor is represented as a binary sequence containing M bits, which can be generated without the need to compute a SIFT-like feature descriptor. A small region surrounding the key point is selected by BRIEF. Then for each pair of points (Pa, Pb) in the region, its discriptor bit is represented by comparing the grayscale values of this pair of points. The bit is 1 if $G(Pa) > G(Pb$, 0 otherwise, where G represents the greyscale value a point. Thus, all M pairs of points are compared to generate a binary vector, which can be presented as:

$$d_p = \sum_{i=1}^{M} 2^{i-1} b_{(Pa_i, Pb_i)}$$

Normally M takes the value of 128, 256 or 512, and we use 256 bits descriptor in this model.

Density Aware Feature Point Reduction

In order to guarantee the precision of pose estimation, the goal of feature point extraction is to find the location of each feature point in the ground truth library from that in the detected picture as precisely as possible. The quality feature matching highly determines the accuracy of picture pose estimation. An acceptable pair of feature points needs to meet two criteria, complexity of feature descriptor and distribution of the location of feature points. However, the FAST key point selection is applied prior to BRIEF descriptor generation. Therefore, there is no chance to know what the descriptor is when selecting key points. So the uniqueness of descriptors is hard to guarantee, especially when there are similar patterns in the same picture. Therefore, we designed a density aware optimizer to distribute the selecetd feature points according to the density in a small region.

The picture is cropped into 16 (4×4) regions in which we calculate the density of feature points. Some feature points with similar descriptors will be removed from the region if the density is higher than a threshold. We use Hamming distance to evaluate the difference of each descriptor. The optimization algorithm is shown in Figure 10. In this algorithm, *Hi* represents the Hamming distances between the descriptor and its nearest 10 descriptors in the region. α is a factor that can control the maximum density of a block. The distributed feature points with more particular descriptors are gained via this optimization algorithm.

In our experimental study, the algorithm is able to significantly reduce the number of feature points produced. We first compute the average number of feature points in all 16 regions. We then control the number of points in each region to be close to the average (within the standard deviation). For many pictures with an uneven density, this may reduce the total points to be 25% less than the original number of points. This in turns may reduce the FLANN matching time by 40–50%.

5.3. *PicPose Feature Points Matching*

FLANN (Fast Library for Approximate Nearest Neighbors) is applied in our work to obtain the matching points between detected picture and the original picture stored in the library. FLANN can provide fast nearest neighbor searching in high dimensional spaces. However, it is hard to avoid the failed matching clustering in a small area. As a dominant process in the localization model, the accuracy of matching directly affects the localization performance. That is to say, too many false matches are not tolerant by localization calculation. Therefore, a matching filter is required to apply for removing the bad matching results.

There are two common situations which may cause matching errors, single point errors and clustered points errors. Single point error means there is only one matching error within a small region. Meanwhile, the clustered points error is that there are a lot of errors clustered in a region. A two-layer matching filter is proposed to handle those two kinds of errors.

Algorithm 1 Feature Points Density Optimization
Input: picture width w, height h, key points array P and corresponding descriptors array d_p
Output: Optimized key points array D^p and descriptor array D^d
Initialize a 2D array for distributed key points D^p, and a 2D array for descriptors D^d;
m = w/4
n = h/4
for i = 0 to 3 do
  for j = 0 to 3 do
    for k = 0 to P.size do
      if $i * m \leq P_i.x < (i+1) * m$ and $j * n \leq P_i.y < (j+1) * n$
        D^P_{i*4+j}.add(P_i)
        D^d_{i*4+j}.add(d_{pi})
        P.remove(P_i)
        d_p.remove(d_{pi})
      end if
    end for
  end for
end for
$\alpha = \min(\frac{1}{100}, \frac{w*h}{P.size})$
for i = 0 to 15 do
  if D^P_i.size > α*m*n
    for j = 0 to D^P_i.size
      H_i.add(Hamming distances of in D^d_{ij} and nearest 10 descriptor in D^d_i)
    end for
    H_i.rank()
    D^P_i.remove(points with Top(D^P_i.size - α*m*n) nearest Hamming distances in H_i)
  end if
end for

Figure 10. Algorithm for Reducing Feature Points.

We use Lowe's test as the first layer that can remove the single point error. Lowe's matching test compares the nearest neighbor's distance to the next nearest neighbor distance to eliminate those unmatched feature points due to image occlusion and background confusion. The feature points with the top 2 closest Euclidean distance of their descriptors are involved in a ratio test. The feature point will be accepted only when the ratio between the Euclidean distance of testing feature point and the top 2 closest feature points is smaller than a critical threshold. A huge number of matching errors can be removed through the nearest ratio test. The recommended threshold is 0.8, but our study shows that the ratio is the best between 0.4 and 0.6.

However the first layer is unable to find a set of errors clustered in one small region. We further designed the second layer, global distance test, for detecting those errors which have unreasonable space distribution in the picture. First of all, a distributed region is created by dividing the picture to a 4×4 grid. In each grid, the dispersion of a matching pair can be represented with the standard deviation of the descriptor's Euclidean distance. The ratio γ between the euclidean distance of a descriptor and the standard deviation of the grid can describe the space distributed deviation in the grid. The feature point pair can be accepted when γ is smaller than a threshold which means the space distribution of this pair does not deviate from the average to much in the region. In general, the threshold of γ can be set between 0.1 and 0.3.

An example of the matching pairs by applying our two-layer filter is shown as Figure 11. We show 3 matching layers in Figure 11. In each layer the picture on the left is the original picture stored in the library and the picture detected from the camera view is on the right. There is no filter applied in the first layer that produces 1656 matched pairs, although we can see that it contains a large number of errors. The second layer shows the matching pairs accepted by the first filter is 301 pairs. We can see that there are still some matching errors crossing the different regions in the picture. The third layer shows the matching pairs accepted by the global distance test. There are only 48 matching pairs last after two-layer filtering, but all of those pairs are matching correctly.

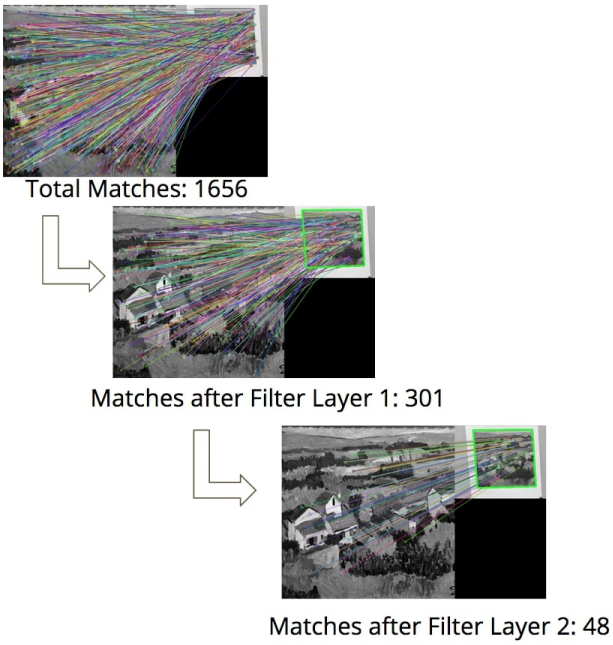


Figure 11. Matching Examples.

In this example of Figure 11, the filter reduces the number of matched pairs from 1656 to 48, keeping only 2.9% of the originally reported matches. This in turns will reduce the next stage of the localization workflow when the camera pose is produced using the homography algorithm. In our study, it shows the homography calculation time may be reduced from 30 ms to 1 ms, or reduce the pose calculation time to about 3.5% of the original time. This is a very signficant improvement on the localiztion efficiency without sacrisfying the pose precision.

5.4. *PicPose Pose Calculation*

The pose of a planar object can be produced by the positional transformation between a planar object detected from a camera and the original model. Homography algorithm is employed in this work to calculate the transition matrix.

The input data for the homography calculation are matched feature point pairs discussed in the last section. The matched pairs are saved in two coordinate matrices, one is matrix M_o with the points on the original picture and the other is M_t that has the corresponding points on the captured picture in the camera view:

$$M_o = [x_{oi}, y_{oi}, 1]^{\mathrm{T}}, M_t = [x_{ti}, y_{ti}, 1]^{\mathrm{T}}$$

The homography matrix H is to define the mapping from M_o to M_t:

$$M_t = HM_o$$

However, transformation errors may occur due to the matching error and picture scale difference. To minimize the transformation error, we use a robust regression method LMedS [41] which is designed to minimize the median of squared residuals. Points are divided into two classes, inliers and outliers, according to their transformation errors in LMedS. The transformation errors of points in the inliers class need to be small enough to produce a stable matrix H. LMedS will identify outlier points, by finding those points severely deviated from the current transformation. LMedS will produce a feasible result as long as the ratio of outliers is smaller than 50%.

The homography matrix can only solve the 2D–2D pose transformation problem. However, our goal is to get the camer's 3D position relative to the picture so that a reference system transformation solution is required.In PicPose, the RPP (Robust Planar Pose) [42] algorithm is applied to estimate the pose transferred from the detected picture. RPP uses a minimum of four coplanar but not collinear points on the target planar object to determine the pose of a camera uniquely. We will use a set of feature points of the planar object from inliers pairs produced by LMedS on the homography calculation step as the input data set for RPP. If there are at least 4 good non-colinear matching pairs in the inliers set, the camera's pose can be found by RPP. However, matching error is unavoidable in practice although we use a 2-layer filter (in Section 5.3). The more inliers points are used the more stable the result will be produced. Therefore, we feed all points in the inliers class into RPP to improve the reliability and robustness of the camera pose calculation.

The goal of RPP is to calculate the 6 DoF(Degrees of Freedom) of a camera's pose. There are 2 kinds of pose information in 6DoF, translation and rotation. Each of them contains 3DoF parameters. RPP can generate two vector, translation vector T and rotation vector R, using the homography matrix and the parameters of the camera.The pose projection is shown in Figure 12. The original picture model is on the left side, P_{oi} is the original feature point represented in scene coordinate system. The target picture on camera view is between original picture and camera. P_{ti} is P_{oi}'s corresponding matching point on target picture, which is represented in camera reference system. C_c is the center point of the camera, and it is also the origin in the camera reference system. The projection between P_{ti} and P_{oi} can be defined by:

$$P_{ti} = RP_{oi} + T$$

Since we know the real-world scale of the original picture, P_{oi} can be represented using the coordinates with a 3D real distance. P_{ti} is also able to be represented by a 3D real distance. In this way, the origin C_c's real-worlds 6DoF pose can be produced.

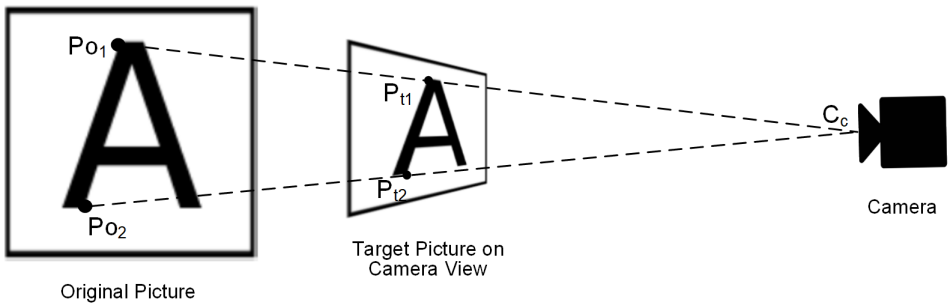


Figure 12. Matching Points Projection.

6. Performance Study

This section presents our performance study on the ArPico and PicPose pose accuracy and execution time. We have built a moving robot that uses the ArPico and PicPose localization to navigate in an indoor space. The robot positions produced by our localization software are compared with the ground truth positions identified from a camera mounted on the ceiling. We also compare the pose accuracy using different types of VBL markers from the same positions using the same robot camera.

The image resolution of the camera is 1920 × 1080 pixels and the markers for the ArUco, ArPico and PicPose comparison study have the same size of 18 cm × 18 cm. The system hardware for running the pose computation has an Intel Core i7, 2.6 GHz CPU and 16 GB memory with the OS of Ubuntu 16.04. All the source codes of testing programs are implemented in C++ and python based on the OpenCV library.

6.1. Robot Localization Accuracy

As shown in Figure 13, we have built a moving robot to test the localization accuracy. The performance study was conducted in a 3.5 × 3.5 m indoor space with 5 pre-trained pictures hanging on the surrounding walls. The top-left window shows the robot-top camera view in which the yellow boxes are the outputs from the SSD object detection and green boxes are produced by PicPose used for the robot's pose calculation from each pictorial object. The top-right window shows the picture matching results for each pictorial object: the left side of each matching pair is the original picture in the library and the detected picture from the robot view is on the right; each line between two picture objects connects the matched feature points. The bottom-left window shows the third-party view where the robot is facing the left wall. The bottom-right window shows the robot location and viewing direction on the 3D map (ceiling view).

The robot is programmed to cruise on a specific route and report the localization outputs to be compared with the ground truth locations. Comparisons on some cruising tests are shown in Figure 14. The blue dots on the figures show the ground truth locations and the red dots are the locations reported by the robot using PicPose. Only one picture marker is seen by the camera in Figure 14a and its MAE is 0.063 m. In Figure 14b, the robot moves from top-left to bottom-right and can see several picture markers. As a result, the MAE is improved to 0.054 m. In general, with the increasing distance to pictures, the errors become bigger since feature points are more difficult to identify with smaller pictures on the camera view. In Figure 14c, the robot moves in a circular route. Estimating the robot location when the robot is making turns presents a challenge since the camera view is often distorted by the constantly changing angles. In Figure 14c, the reported locations are not as close to the ground truth with an MAE of 0.073. On certain part of the route, no picture marker can be seen so that no location is reported.

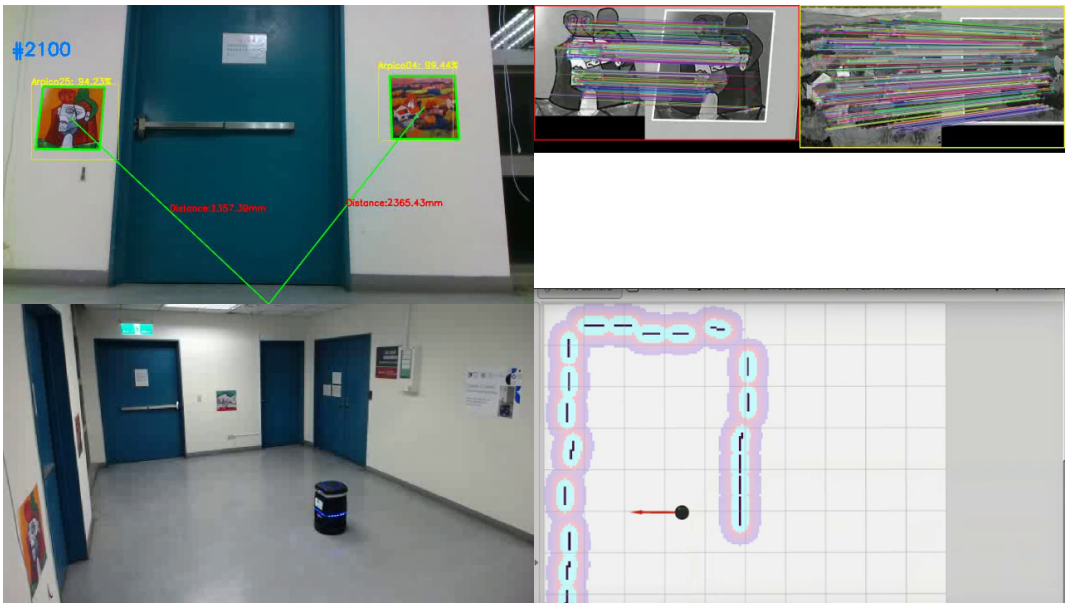
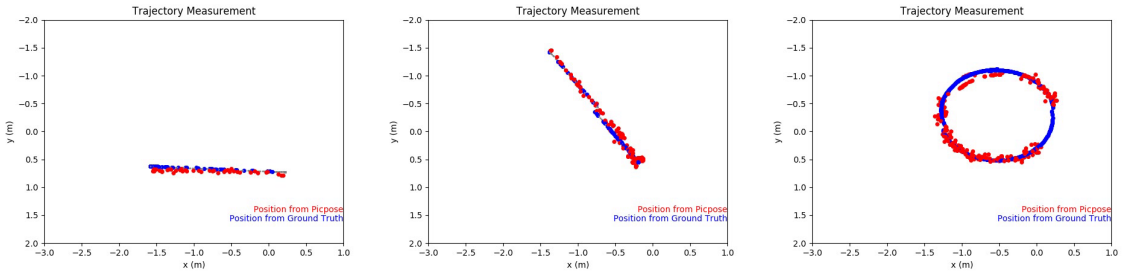


Figure 13. Robot Localization Study.



(**a**) Single marker localization (**b**) Multi-marker localization (**c**) Cicular motion localization

Figure 14. PicPose Robot Localization Accuracy.

6.2. Pose Correctness

To compare the VBL performance under different camera distances, environment brightness and noise conditions, we position the camera at different distances (100 cm, 150 cm, 200 cm and 300 cm) from the picture markers, and shoot video under normal, low brightness and noise conditions. We add gaussian noises to the images to simulate the noise condition. We use Three correctness criteria on evaluating the results. MAE refers to the mean absolute error (in pixels) between reported outline of the objects and their pose in ground truth. Diagonal is the sum of two diagonal distances of the pictorial object's corners. MAER, shows the matching error relative to the marker's size on the camera view, is the ratio of MAE and diagonal distances.

The test results are shown in Table 2. The performances of three methods are all good in a normal brightness environment with the camera distance of 100–150 cm. Whereas PicPose performs better in a low brightness environment as both ArUco and ArPico only detect the corner point to calculate the pose. When the picture is in a dark environment, its corner and contour may not easily be differentiated from the background. Compared with those two methods, PicPose uses texture features in the whole picture which can produce more features to be detected in a dark environment. The MAE and MAER of all three methods become bigger with increasing distances. PicPose is better than ArUco and ArPico for the distance of 300 cm, especially under low brightness conditions. As for the images with noise, the MAEs of both ArUco and ArPico are lightly affected, meanwhile PicPose performed better than former as it has Bilateral filtering to reduce noises.

Table 2. Pictorial Marker Matching Error.

Distance (cm)	Condition	ArUco		ArPico		PicPose	
		MAE/Diagonal	MAER	MAE/Diagonal	MAER	MAE/Diagonal	MAER
100	Normal	4/966	0.4%	4/954	0.4%	5/975	0.4%
100	Low Brightness	7/966	0.7%	7/954	0.7%	6/975	0.6%
100	With Noise	10/966	1%	11/954	1.2%	8/975	0.8%
150	Normal	6/651	0.9%	6/668	0.9%	5/657	0.8%
150	Low Brightness	9/651	1.4%	8/668	1.3%	6/657	0.9%
150	With Noise	16/651	2.5%	13/668	1.9%	10/657	1.5%
200	Normal	7/486	1.4%	7/481	1.5%	7/491	1.4%
200	Low Brightness	9/486	1.9%	10/481	2.0%	7/491	1.4%
200	With Noise	16/486	3.3%	17/481	3.5%	14/491	2.8%
300	Normal	5/326	1.5%	6/334	1.8%	4/325	1.2%
300	Low Brightness	7/326	2.1%	8/334	2.4%	5/325	1.5%
300	With Noise	18/326	5.5%	15/334	4.5%	10/325	3.0%

The second comparison shows the reported distance between the camera and markers using ArUco, ArPico and PicPose for the same experimental setup. The result is shown in Table 3. When the camera distance is 100–200 cm, the results of ArUco and ArPico are slightly better than that of PicPose. However, PicPose performs better when the camera distance is 300 cm since it can get more matching points than ArUco and ArPico as the picture size gets smaller in the camera view.

Table 3. Reported Pose Distance.

Truth Distance (cm)	Light Level	ArUco (cm)	ArPico (cm)	PicPose (cm)
100	Normal	100.8	101.2	101.5
100	low	101.6	101.1	101.3
150	Normal	149.6	150.2	149.2
150	low	148.1	149.5	148.7
200	Normal	200.2	200.3	200.8
200	low	201.9	201.1	201.7
300	Normal	302.7	303.8	302.5
300	low	306.4	306.2	305.9

6.3. Partially Visible Picture Test

In order to verify the capability of detecting the partially visible pictures, we designed the third experiment. The camera is placed in different positions to capture different percentages of visible picture. The comparison results on detecting partially visible picture is shown in Table 4. Neither ArUco nor ArPico can report any pose result since they need to see all four corners in the camera view. The visibility percentages on PicPose tests represent the ratio of visible parts of pictures appearing in the view. It can be seen in the table that only a few frames can be reported when the percentage of the visible part is 10%. The success rate of PicPose highly depends on SSD; if there are not enough features to be detected by SSD, the pose calculation will fail. The objects can be successfully detected when the visibility ratio increases to 50%. More than 92% of the frames are successfully used for pose calculation with the pictorial object visibility ratio of 90%.

Table 4. Partially Visible Picture Pose Success Ratio.

Distance (cm)	ArUco	ArPico	PicPose (10%-View)	PicPose (30%-View)	PicPose (50%-View)	PicPose (90%-View)
100	0/335	0/234	56/302 (18.5%)	143/336 (42.6%)	264/298 (88.6%)	321/321 (100%)
200	0/267	0/321	6/289 (2%)	114/321 (35.5%)	268/313 (85.6%)	303/307 (98.7%)
300	0/352	0/333	0/325 (0%)	30/316 (9.5%)	156/290 (53.8%)	297/323 (92.5%)

6.4. Execution Time

The last experiment is the comparison on computing speed. In Table 5, the total computing time values of ArUco, ArPico and PicPose are 27.3–29.4 ms, 20.5–34.4 ms and 80.9–85.7 ms respectively. SSD is the most time-consuming part in PicPose. The time spent on feature points extraction and matching reduces when the camera distance gets farther. This is because the feature points detected from the camera view will become less when the picture gets smaller. PicPose can produce 11–13 locations per second which is acceptable for a robot to make real-time navigation decisions. In general, we believe that both ArPico or Picpose can provide high accuracy and real-time localization services in robot systems.

Table 5. Localization Execution Time.

Distance (cm)	PicPose (ms)					ArUco (ms)	ArPico (ms)
	SSD	Extraction	Matching	Position	Total		
100	58.7	14.3	8.2	1.1	82.3	27.3	30.5
200	64.1	12.7	7.9	1.0	85.7	29.4	31.2
300	63.3	8.3	8.2	1.1	80.9	28.1	34.4

7. Conclusions and Future Work

In this paper, we have constructed a universal pictorial planner object based VBL framework. Based on this framework, we presented two centimeter-level localization methods using ArPico

and PicPose, which detect the objects with pictorial surfaces in an indoor environment to provide device location. Using our proposed technology, mobile devices only need to equip with an on-board monocular camera to achieve indoor localization. As long as there is a 3D map consisting of known planar surfaces, the location can be generated precisely and efficiently. This is a low cost solution which makes the visual based localization method very scalable and flexible. Our proposed localization technology can be applied by any moving device with a camera, such as robot, smartphone, and smart glasses, as long as clear views on the environmental objects can be captured.

Our VBL methods are built on top of existing image processing technology for picture identification and matching. We are enhancing many of them and also studying new machine learning models for picture recognition and matching. Our overall goal is to build an efficient VBL technology that can be deployed on light-weight indoor devices for future smart applications.

Author Contributions: Conceptualization, methodology, investigation, data curation, formal analysis, writing–original draft preparation, review and editing, Y.M. and K.-J.L.; software and validation, Y.M., B.-L.T., C.-C.C. and Y.C.; resources, Y.M., K.-J.L. and B.-L.T.; visualization, Y.M. and C.-C.C.; supervision and project administration, K.-J.L. and B.Z.; funding acquisition, Y.M., K.-J.L. and B.Z.. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 108-2633-E-002-001), National Taiwan University (NTU-108L104039), Intel Corporation, Delta Electronics, Compal Electronics, National Natural Science Foundation of China (U1908212) and the Fundamental Research Funds for the Central Universities (N2017013, N2017014). Yu Meng was supported by the China Scholarship Council (201706080026).

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Piasco, N.; Sidibé, D.; Demonceaux, C.; Gouet-Brunet, V. A survey on Visual-Based Localization: On the benefit of heterogeneous data. *Pattern Recognit.* **2018**, *74*, 90–109. [CrossRef]
2. Brosh, E.; Friedmann, M.; Kadar, I.; Lavy, L.Y.; Levi, E.; Rippa, S.; Lempert, Y.; Fernandez-Ruiz, B.; Herzig, R.; Darrell, T. Accurate Visual Localization for Automotive Applications. *arXiv* **2019**, arXiv:1905.03706.
3. Sadeghi-Niaraki, A.; Choi, S.M. A Survey of Marker-Less Tracking and Registration Techniques for Health & Environmental Applications to Augmented Reality and Ubiquitous Geospatial Information Systems. *Sensors* **2020**, *20*, 2997.
4. Se, S.; Jasiobedzki, P. Stereo-vision based 3D modeling and localization for unmanned vehicles. *Int. J. Intell. Control. Syst.* **2008**, *13*, 47–58.
5. Garcia, M.A.; Solanas, A. 3D simultaneous localization and modeling from stereo vision. In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004; Volume 1, pp. 847–853.
6. Se, S.; Ng, H.K.; Jasiobedzki, P.; Moyung, T.J. Vision based modeling and localization for planetary exploration rovers. In Proceedings of the International Astronautical Congress, Vancouver, BC, Canada, 4–8 October 2004; pp. 434–440.
7. Garrido-Jurado, S.; noz Salinas, R.M.; Madrid-Cuevas, F.; Medina-Carnicer, R. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognit.* **2016**, *51*, 481–491. [CrossRef]
8. Muñoz-Salinas, R.; Medina Carnicer, R. UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers. *Pattern Recognition.* **2020**, *101*, 107193, [CrossRef]
9. Acuna, R.; Li, Z.; Willert, V. MOMA: Visual Mobile Marker Odometry. In Proceedings of the 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Nantes, France, 24–27 September 2018; pp. 206–212. [CrossRef]
10. ArUco Project. Available online: https://www.uco.es/investiga/grupos/ava/node/26 (accessed on 11 October 2019).
11. Meng, Y.; Lin, K.; Peng, B.; Tsai, B.; Shih, C. ArPico: Using Pictures to Build Localization Service for Indoor IoT Applications. In Proceedings of the 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), Paris, France, 20–22 November 2018; pp. 105–112. [CrossRef]

12. Munoz-Salinas, R.; Marin-Jimenez, M.J.; Yeguas-Bolivar, E.; Medina-Carnicer, R. Mapping and localization from planar markers. *Pattern Recognit.* **2018**, *73*, 158–171. [CrossRef]
13. Hu, D.; DeTone, D.; Malisiewicz, T. Deep charuco: Dark charuco marker pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8436–8444.
14. Basiratzadeh, S.; Lemaire, E.D.; Dorrikhteh, M.; Baddour, N. Fiducial Marker Approach for Biomechanical Smartphone-Based Measurements. In Proceedings of the 2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART), Paris, France, 24–26 April 2019; pp. 1–4. [CrossRef]
15. Mutka, A.; Miklic, D.; Draganjac, I.; Bogdan, S. A low cost vision based localization system using fiducial markers. *IFAC Proc. Vol.* **2008**, *41*, 9528–9533. [CrossRef]
16. Xavier, R.S.; da Silva, B.M.; Gon, L.M. Accuracy analysis of augmented reality markers for visual mapping and localization. In Proceedings of the 2017 Workshop of Computer Vision (WVC), Natal, Brazil, 30 October–1 November 2017; pp. 73–77.
17. Hübner, P.; Weinmann, M.; Wursthorn, S. Marker-based localization of the microsoft hololens in building models. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *42*, 195. [CrossRef]
18. Awais, M.; Park, J.; Jung, J.; Choi, E.; Park, J.; Kim, C. Real-Time Vision-Based Localization of Planar Cable-Driven Parallel Robot. In Proceedings of the 2018 18th International Conference on Control, Automation and Systems (ICCAS), Daegwallyeong, Korea, 17–20 October 2018; pp. 462–465.
19. Germanese, D.; Leone, G.R.; Moroni, D.; Pascali, M.A.; Tampucci, M. Long-Term Monitoring of Crack Patterns in Historic Structures Using UAVs and Planar Markers: A Preliminary Study. *J. Imaging* **2018**, *4*, 99. [CrossRef]
20. Davison, A.J.; Reid, I.D.; Molton, N.D.; Stasse, O. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1052–1067. [CrossRef] [PubMed]
21. Pumarola, A.; Vakhitov, A.; Agudo, A.; Sanfeliu, A.; Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 4503–4508. [CrossRef]
22. Taira, H.; Okutomi, M.; Sattler, T.; Cimpoi, M.; Pollefeys, M.; Sivic, J.; Pajdla, T.; Torii, A. InLoc: Indoor Visual Localization With Dense Matching and View Synthesis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
23. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018.
24. Sobreira, H.; Pinto, M.; Moreira, A.P.; Costa, P.G.; Lima, J. Robust Robot Localization Based on the Perfect Match Algorithm. In *CONTROLO'2014–Proceedings of the 11th Portuguese Conference on Automatic Control*; Moreira, A.P., Matos, A., Veiga, G., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 607–616.
25. Yuan, C. Markerless pose tracking for augmented reality. In *International Symposium on Visual Computing*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 721–730.
26. Vlaminck, M.; Luong, H.; Philips, W. A markerless 3D tracking approach for augmented reality applications. In Proceedings of the 2017 International Conference on 3D Immersion (IC3D), Brussels, Belgium, 11–12 December 2017; pp. 1–7.
27. Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; Torralba, A. Learning Deep Features for Discriminative Localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2921–2929.
28. Chaplot, D.S.; Parisotto, E.; Salakhutdinov, R. Active Neural Localization. *arXiv* **2018**, arXiv:1801.08214.
29. Carvalho, L.E.; von Wangenheim, A. 3D object recognition and classification: a systematic literature review. *Pattern Anal. Appl.* **2019**, 22, 1–50. [CrossRef]
30. Soltani, A.A.; Huang, H.; Wu, J.; Kulkarni, T.D.; Tenenbaum, J.B. Synthesizing 3D Shapes via Modeling Multi-View Depth Maps and Silhouettes with Deep Generative Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1511–1519.
31. Allen, J.G.; Xu, R.Y.D.; Jin, J.S. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. In Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing, Sydney, Austria, 18–23 June 2004; pp. 3–7.

32. Maldonado-Bascon, S.; Lafuente-Arroyo, S.; Gil-Jimenez, P.; Gomez-Moreno, H.; Lopez-Ferreras, F. Road-Sign Detection and Recognition Based on Support Vector Machines. *IEEE Trans. Intell. Transp. Syst.* **2007**, *8*, 264–278. [CrossRef]
33. Bay, H.; Tuytelaars, T.; Van Gool, L. Surf: Speeded up robust features. In Proceedings of the European conference on computer vision. Graz, Austria, 7–13 May 2006; pp. 404–417.
34. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6 November–13 November 2011; pp. 2564–2571. [CrossRef]
35. Zheng, L.; Zhang, Y.; Thing, V.L.L. A survey on image tampering and its detection in real-world photos. *J. Vis. Commun. Image Represent.* **2019**, *58*, 380–399. [CrossRef]
36. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *arXiv* **2019**, arXiv:1905.05055.
37. Tolias, G.; Sicre, R.; Jégou, H. Particular object retrieval with integral max-pooling of CNN activations. *arXiv* **2015**, arXiv:1511.05879.
38. Radenović, F.; Tolias, G.; Chum, O. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 3–20.
39. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. *IEEE Trans. Robot.* **2015**, *31*, 1147–1163. [CrossRef]
40. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
41. Schweighofer, G.; Pinz, A. Robust Pose Estimation from a Planar Target. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *28*, 2024–2030. [CrossRef]
42. Meer, P.; Mintz, D.; Rosenfeld, A.; Dong, Y.K. Robust regression methods for computer vision: A review. *Int. J. Comput. Vis.* **1991**, *6*, 59–70. [CrossRef]