CamIoT: Recognizing and Interacting with Distant IoT Objects Using a Wrist-Worn Outward-Facing Camera

Anonymous CVPR submission

Paper ID 0017

Abstract

CamIoT is the first wrist-worn platform that uses an outward-facing camera to recognize and interact with IoT objects at a distance. CamIoT leverages the index finger's gesture to provides these novelties: (i) Utilizing the finger's orientation for guiding the recognition of an IoT object in the camera view; and (ii) Detecting finger's circumduction and flexion for the function selection of the chosen IoT object. Finger tracking plays an essential role in main parts of CamIoT system involving in both object recognition and interaction with appliances. Thus in this report, we mainly ***focus on three methods we implemented for finger tracking in CamIoT****. We then experimentally measure their performances and discuss their limitations. Other fundamental parts of our work are also introduced and presented briefly to explain the system thoroughly.*

Introduction

CamIoT (camera-based IoT) is a wrist-worn Internet of Things (IoT) device for recognizing (smart) home appliances and interacting with them. With IoT devices becoming so ubiquitous, the demand for better and smoother usability features proliferates. Often such interactions require extra efforts to retrieve apps from personal devices or instrument the environment for voice or gesture control. CamIoT's goal is to enable always-available instrumentation-free interaction by merely pointing and gesturing to IoT objects. The performance in this wrist-worn device consists of two main sections: object recognition and interaction with appliances. In the first part, this hardware/software platform recognizes the object the user is pointing at. Once the taken image is classified, CamIoT allows the user to choose a function based on the selected object by moving his/her index finger.

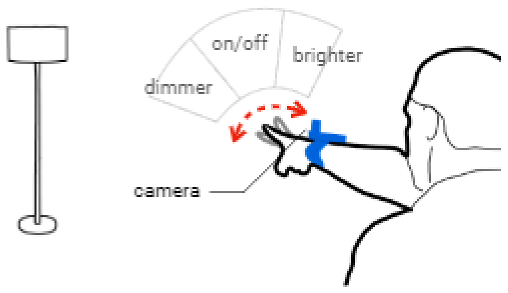


Figure 1: Camiot uses a wrist-worn outward-facing camera to recognize an IoT object as a user points at it, using finger flexion and circumduction gestures to interact with control shortcuts of a selected IoT.

System Design

Our hardware consists of a Raspberry Pi Zero W as the controller, an MPU 6050 IMU sensor for providing accelerometer and gyroscope data, and a Raspberry PI Camera Module V2 for capturing IoT objects and the user's finger. For audio feedback, we embedded a speaker under CamIoT's case. For the object recognition part only, the images taken are sent to a local server for classification (detailed in the following sections).

System Overview

The process starts when the user raises his arm to take a picture. CamIoT senses such arm movements through the IMU data and based on [4]'s method. The image is then sent to the local server for classification. Once the result is communicated back to the device, the voice feedback announces the result. In this step, if the user decides to hold his/her finger in the camera view, the finger's direction helps to crop the image (we refer to this technique as disambiguation in later sections). After recognition, CamIoT solely tracks the index finger's position in the camera view (finger circumduction). The current setup matches the finger's location to either the left, middle, or right direction. The user confirms the finger's direction by dropping his/her finger from the camera view (flexion). The chosen direction is then mapped to a particular function on the IoT device (e.g., turn down TV Volume).

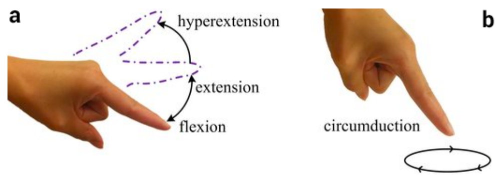


Figure 2: CamIoT uses Finger flexion and circumduction to select control functions of IoT devices. Image credit: Wang *et al*. [31]

Methods

Appliance (Object) Recognition

In this work, we carry out the few-shot learning of a ConvNet for appliance recognition, as ConvNets are better for capturing small objects from images with features of large receptive fields [8, 1]. Moreover, we propose to utilize the finger for disambiguation. And feed the model with the indicated portion of an image for prediction. Such a method can potentially benefit model performance because: Model attention can focus on the appliances with reduced background areas, thus reducing the probability of wrong classifications by eliminating other potential appliances from the background.

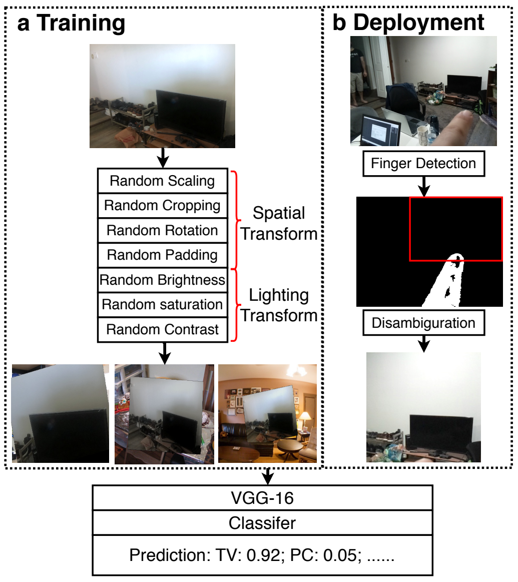


Figure 3: Object recognition network. The model consists of a shallow classifier of two fully connected layers and a frozen VGG-16 that pretrained on [8] for generating deep feature maps.

In the setup stage, five images of each appliance are taken by camIoT from various angles as training data. During the deployment (3b), the finger segmentation is first derived automatically from the query image. Then, guided by the finger's direction, the image is cropped to 0.6 of its size and fed into the model for inference.

Finger Detection

To recognize the circumduction and flexion of the figure, CamIoT performs automatic finger segmentation from the camera view. It uses the characteristics of skin color and edge detection. And it does the finger segmentation without any supervision.

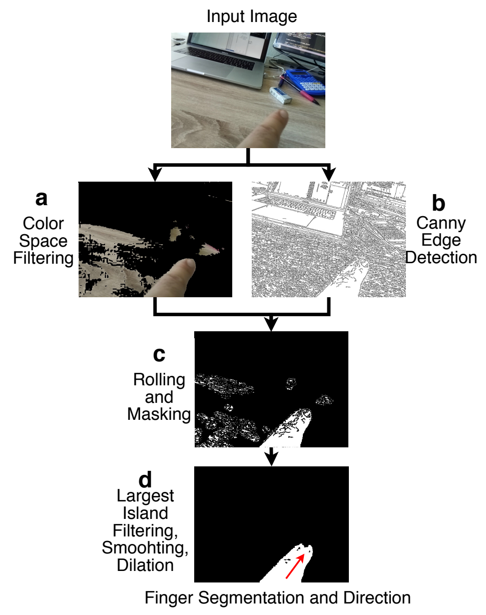

Input Image
a
Color Space Filtering
b
Canny Edge Detection
c
Rolling and Masking
d
Largest Island Filtering, Smoohting, Dilation
Finger Segmentation and Direction

Figure 4: Finger Detection Pipeline

We first derive a rough finger mask (Figure 4a) with the skin color model in YCbCr space [6]. Multiple false-positive regions can happen in the segmentation map due to background similarities. Thus an edge mask (Figure 4b) with Canny filter is generated and applied in this step (Figure 4c). We finally take the largest isolated region on the resulted finger mask that lies on the lower part of the image as the finger prediction, and further derive the finger direction by linear interpreting the row-wise midpoints of the segmentation (Figure 4d). If no region has an area size bigger than a preset threshold, the image is detected as no finger.

This finger detection approach was not always robust and effective in different setups with a bright background. Thus we explored two more methods, which are as follows: 1. Depth Map Measurement, 2. Template Matching. For these approaches, we modified our hardware and added another

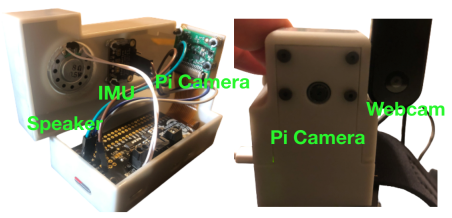


Figure 5: Adding a second camera to CamIoT and hardware re-design

camera (PC webcam) to our system (5).

Depth Map

For the depth map approach, the calculated depth map is utilized as the mask to separate the background and the fingers because fingers should be much closer to the camera. To find the depth map, we took images from both of the cameras. Although a depth map based on a single camera is recently reported [7, 3], it requires either video recording or better quality cameras with accessible and adjustable intrinsic features. So we continued with the two-camera option due to our camera conditions and the lower complexity of the dual-camera setup. Our depth map approach can be divided into two stages - calibration and calculation. In depth map procedure, unlike our case, usually, the two cameras are identical and only vary in their location. So we calibrated across the cameras using an 8x6 checkerboard. We then estimated the intrinsic parameters of the cameras, including focal length and the image distortion.

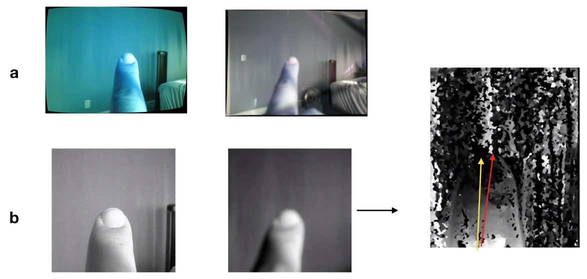


Figure 6: Depth Map Pipeline. (Green Arrow:Expected direction Red Arrow: Predicted direction)

With each camera's estimated intrinsic parameters, images were resized and scaled according to the parameters to provide proper depth map measurement (6a). After the calibration, the images were processed to estimate the depth map. The depth map estimation is based on Stereo Processing by Semi-Global Matching and Mutual Information (StereoSGBM) [2]. Once the depth map is estimated, we could utilize the depth information to attain further details, including point direction of the fingers and the mask of background and fingers (6b). Applying blob detection on the disparity (depth map) image (based on color and thresholding), we calculated the fingertip area. Assuming the finger end is fixed in all the frames (not much movement at its end is possible), a vector connecting the bottom portion to the center of the calculated blob is the finger direction.

Template Matching

This time we utilized the differences between the intrinsic parameters of the cameras. The webcam provided higher resolution and longer focal length. Thus its images were more zoomed in and focused; in a sense, more ideal to be treated as our template. We cropped the fingertip treating it as a kernel, 2D convoluted it through Pi camera image. Following different OpenCV template matching algorithms, our data set best results were generated by "TM SQDIFF NORMED":

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}.$$

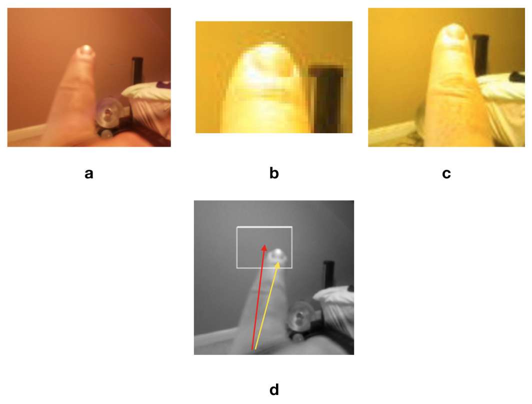
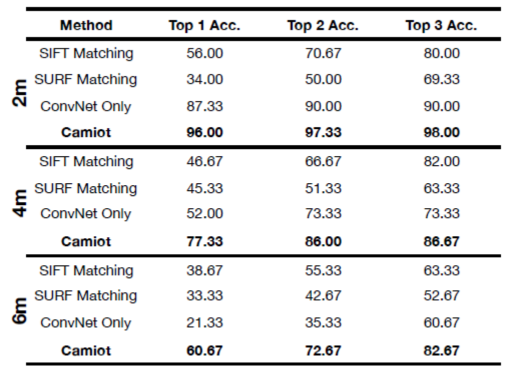


Figure 7: a.Pi Image b.The chosen template c.Webcam image 4.processed image (Green Arrow:Expected Direction Red Arrow:Predicted Direction)

Again, knowing the index finger's end doesn't move, we calculated its location in the training stage. Once we found the template position in the Pi image, a vector from the index finger's end to the detected template would indicate the finger direction (7).

Results

To evaluate the accuracy of selecting appliances from a distance, We built a data set, including ten appliances. First, the model was trained using five images per object taken from about 0.5 meters distance. The photos were also taken from various angles to profile the visual appearance of the objects comprehensively. The users were asked to point



	Method	Top 1 Acc.	Top 2 Acc.	Top 3 Acc.
2m	SIFT Matching	56.00	70.67	80.00
	SURF Matching	34.00	50.00	69.33
	ConvNet Only	87.33	90.00	90.00
	Camiot	**96.00**	**97.33**	**98.00**
4m	SIFT Matching	46.67	66.67	82.00
	SURF Matching	45.33	51.33	63.33
	ConvNet Only	52.00	73.33	73.33
	Camiot	**77.33**	**86.00**	**86.67**
6m	SIFT Matching	38.67	55.33	63.33
	SURF Matching	33.33	42.67	52.67
	ConvNet Only	21.33	35.33	60.67
	Camiot	**60.67**	**72.67**	**82.67**

Figure 8: Appliance Recognition Evaluation

at each object 20 times in random order in three distance ranges: $2m$, $4m$, $6m$.

We compared our algorithm against two template matching methods; one used SIFT (Scale-invariant feature transform) as the feature descriptor while the other applied SURF (Speeded-Up Robust Features). Figure 8 shows that our method achieves recognition accuracy of 96.00%, 77.33% and 60.67 for 2m, 4m, and 6m distances, respectively, which are the highest among the methods.
For Finger Detection, we first found the most natural thresholds of angles for dividing the virtual panel. The panel consisted of only three segments (left, middle, right), and we ran a pilot study to find the thresholds based on the users' performance. We asked each user to naturally point at each segment in a random order several times. We then derived the optimal thresholds from being the angles that best split different segments based on an exhaustive search. The resultant threshold are shown in 9

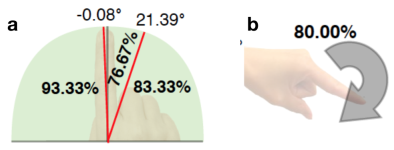


Figure 9: Finger Tracking segmentation, calculated thresholds and reported accuracy for flexion and circumduction.

For evaluation of finger circumduction and flexion, we informed the user of the segmentation thresholds, and again measured his performance while he was pointing at the directions in random order. Specifically, after each pointing, we asked the user to perform a flexion. The study includes all ten appliances and three pointing gestures that happened for each segment. Figure 9 indicates the detection accuracy of selecting different segments and flexing action (flexion is used in a confirmation manner in our system design). Our algorithm reached a mean accuracy of 84.44% while the overall flexion accuracy was around 80.00% (9).

Due to the COVID-19 situation and limited access to perform user study, our experiments only involved 1-3 participants. Furthermore, given the hardware limitation and the pandemic, we could not regenerate the above studies to test the depth map and template matching methods. Thus we took a different approach where we only focused on corner cases in the discussed methods. Our goal was to explore the possibility of improving the current design using the computational techniques introduced in the **depth map** and **template matching** sections. In this experiment, we only considered edge cases. Scenarios such as where the background is noisy or too bright, the skin color resembles the background color. Few other scenarios were when the finger is mostly out of camera view, or the back of the hand is partially occluding the field of view. With these assumptions in mind, we experimented 10-15 cases using depth map and template matching methods. Based on our findings, we might improve finger tracking by 12.4% via template matching or 7.5% by using depth map in the overall design.

Future Works

Although the computational methods seemed promising in terms of overall performance, their downside was our hardware capacity. To fully investigate such methods we needed preferably identical cameras and RTOS (better embedded systems and C/C++ coding platform instead of Python). Moreover the depth map from the two different cameras, generated a noisy output which significantly affected the blob detection in some cases. Therefore there's an demand for noise removal from depth maps [5]. As mentioned, the template matching approach was more effective. However, one of the challenges is to provide a proper template which can highly affect the outcome. The effective template area can be calculated in the setup stage through training images. Our proposed method was to define a reinforcement learning network with a policy to optimize the overall location of the finger tip across the training data set. Like any other methods computational approaches discussed here also have their own trade-offs. But the improvement shown in the results motivates more investigation in this domain. To avoid full implementation and its complexity, an interesting approach would be to combine these computational methods with the current finger tracking algorithms.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015. 2

[2] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. 3

[3] J. Jang and J. Paik. Dense depth map generation using a single camera with hybrid auto-focusing. In *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, pages 277–278, 2015. 3

[4] Runchang Kang, Anhong Guo, Gierad Laput, Yang Li, and Xiang 'Anthony' Chen. Minuet: Multimodal interaction with an internet of things. In *To Appear at the ACM symposium on Spatial user interaction*. ACM, 2019. 1

[5] S. Kim, M. Kim, and Y. Ho. Depth image filter for mixed and noisy pixel removal in rgb-d camera systems. *IEEE Transactions on Consumer Electronics*, 59(3):681–689, 2013. 4

[6] S Kolkur, D Kalbande, P Shimpi, C Bapat, and J Jatakia. Human skin detection using rgb, hsv and ycbcr color models. *arXiv preprint arXiv:1708.02694*, 2017. 2

[7] S. Lee, J. Lee, M. H. Hayes, A. K. Katsaggelos, and J. Paik. Single camera-based full depth map estimation using color shifting property of a multiple color-filter aperture. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 801–804, 2012. 3

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2