ConNect: Exploring Augmented Reality Service using Image Localization and Neural Network Object Detection

Tong Li
Randy H. Katz, Ed.
David E. Culler, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-64
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-64.html

May 11, 2018

ConNect: Exploring Augmented Reality Service using Image Localization and Neural Network Object Detection

Abstract

ConNect: Exploring Augmented Reality Service using Image Localization and Neural Network Object Detection

by

Tong Li

Master in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Randy H. Katz, Chair

Conventional Mobile Augmented Reality (MAR) systems, which identify objects using an image localization based instance identification system perform poorly in certain situations, such as when the background is too simple to track, or the query image is taken too close to the object. Some of these situations could be resolved by the recent improvement of neural network object detection. This report proposed two ways of using neural network object detection to complement computer vision based MAR system. We are able to achieve higher accuracy, nearly 100% in certain situations, and lower false positive rate, down to 0% in most cases, without introducing too much additional latency, around 30 ms (20%).

Contents

List of Figures

Acknowledgments

This work would not have been possible without the guidance from Kaifei Chen and Professor Randy H. Katz. Many thanks to Professor David E. Culler for being my second reader of this work. Xin Wang and Kaiqian Zhu also helped me in understanding and deploying Neural Network in to my system.

Chapter 1

Introduction

Augmented Reality (AR) has seen wide adoption, with the recent advances in computer vision and robotics. The basic concept of AR is to show a user additional information overlaid on the original view when she sees the real world through the screen. A user expects to see the augmented information both accurately and in real time, while moving her device. To this end, an AR device must perform intense computations on a large amount of (visual) data with imperceptible latency, from capturing an image to extracting the corresponding information and overlaying it on the proper location of the screen.

While this challenge has been addressed by using powerful and expensive AR-oriented hardware platforms, such as Microsoft HoloLens [10] and Google Tango [22], another approach has been also investigated: AR with regular mobile devices, such as a smartphone, called Mobile AR (MAR). MAR is attractive in that it does not require users to purchase and carry additional devices [6]. However, MAR has its own challenges since mobile devices have significantly less storage and computation speed than AR-oriented devices. There are relatively lightweight MAR software tools, such as Google ARCore [2] and Apple ARKit [3], but they operate only on the latest generation of smart phones and consume considerable energy, resulting in limited applicability.

This work investigates how to enhance MAR on ordinary mobile devices. Specially, we aim to provide a stable annotation-based MAR service that understands objects, at which the device points, and overlays the corresponding annotations on the relevant location of the screen. This application is viable even with the restricted capabilities of mobile devices, in contrast to more demanding AR that naturally embeds 3D objects into a 3D model using a game engine [10]. More importantly, it has a variety of practical usage scenarios, such as navigation through physical spaces like museums, grocery stores, and airports [12, 7], Through the annotation service, a user can obtain information about an object by pointing her phone at it. Furthermore, she is able to interact with (or control) the object by touching annotations on the screen. In this report, annotation and label are used interchangeably as the information of object that is being identified.

Some representative ways to support MAR includes image retrieval and image localization. Both of these have their own limitations. They have poor performance when the image is captured very close to the object. An image capture form a location that is too close to the objects provides less surrounding features to localize or retrieve the image. On the other hand, since they mainly rely on K-Nearest Neighbors (K-NN) [7] to find the approximate closest result to the query image, their false positive rate is very high when there is no correct result in the dataset.

On the other hand, Neural Network Object Detection (NNOD) has improved significantly in recent years, due to the improvement in hardware support (e.g., better GPUs), as well as a deeper understanding of neural networks and more complex and sophisticated neural network architectures. The limitation of NNOD is that it only supports category detection, where the categories are pre-defined during the training phase of neural network models. This is resolved by tuning an existing model to have its own custom categories. That is, each device is mapped to one category during the tuning phase. However, this approach fails when two identical devices exist in the environment.

In this report, we design and implement ConNect, a prototype MAR system that can quickly and accurately identify an object among hundreds to thousands of (possibly similar) objects, in a building, based on a query image from an arbitrary location and orientation. ConNect identifies objects by first processing a query image using both image localization and NNOD. Then, ConNect fuses the result of the two to achieve higher accuracy and lower false positive rate without sacrificing too much latency.

Our contributions are as follow:

- We demonstrate an end-to-end annotation-based MAR service on ordinary mobile devices with high accuracy.
- We propose two novel ways of fusing image localization based objected identification and NNOD based object identification.
- We evaluate two fusion algorithm proposals and analyze the advantages and disadvantages of them.

In this report, background knowledge of image localization and neural network object detection are introduced in Section 2. The architecture and workflow of ConNect is explained in Section 3. Section 4 provides detained explanation of two fusion algorithms proposed in ConNect, including their pros and cons, as well as the trade-off decisions made. Section 5 provides details of implementing ConNect. In Section 6, ConNect is evaluated and compared to a baseline system, SnapLink [7], that only uses image localization to achieve similar MAR service. Section 7 is the summary and future work of ConNect.

Chapter 2

Background

2.1 Image localization

An image localization-based object identification system (Figure 2.1) contains two phases. First, the construction phase involves constructing a database in the form of a 3D model of a building and one labeled point for each object in the 3D space [20]. Second, the query phase takes a query image as its input, it localizes the query image in the 3D-space database using the most likely location and angle, captures the image of that specific location/angle (i.e., a small fraction) from the 3D space, and finds the object label point closest to the center of the captured image.

Several aspects make image localization a better fit for our target application compared to image retrieval. First, the system deployment process is extremely simple, because a 3D model that covers various angles can be built by capturing a video with a modern modeling tool, such as Project Tango [22], during a building walk through. Labeling a 3D model is also simple because every object only needs to be labeled once as a single point in the 3D space, irrespective of the number of images containing that object in the database. Second,

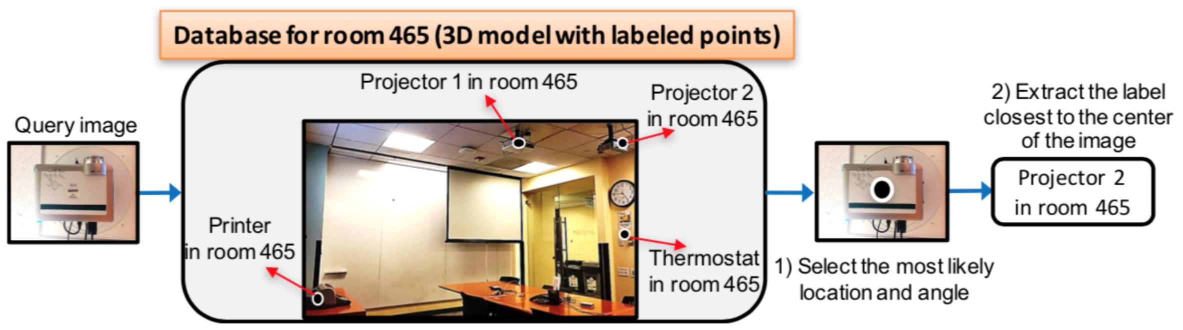

Database for room 465 (3D model with labeled points)
Query image
Projector 1 in room 465
Projector 2 in room 465
Printer in room 465
Thermostat in room 465
1) Select the most likely location and angle
2) Extract the label closest to the center of the image
Projector 2 in room 465

Figure 2.1: Image Localization

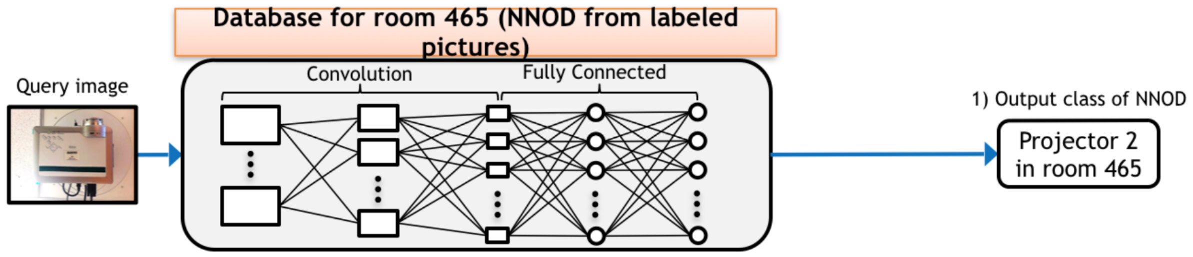


Figure 2.2: Neural Network Object Detection

because a 3D model incorporates all image features into a 3D space, it can represent all angles and distances, resulting in a higher identification accuracy with an arbitrary query image compared to image retrieval. As an example, in a room that contains 100 objects, each of which needs to be captured from 10 different viewing locations, image retrieval would require 1,000 reference images, whereas image localization only requires a video and 100 labeled points. Thus, our system uses image localization instead of image retrieval.

However, the false positive rate of image localization is very high. This is caused by two reasons. First, it finds the most likely location and angle of the query image from the 3D model. Therefore, even if the image is not taken in any place that belongs to the 3D model, the image localization based identification will still return something. Second, it retrieves the labels of items on the image based on the re-projection from the position and angles. Therefore when the position and angle are wrong, it localizes an incorrect position and angle, and it will still return the contents from that wrong position and angle. This is getting worse when the position is localized incorrectly from a extreme far distance, which causes the image being considered to contain all the objects in the 3D model. This can be extremely dangerous in some use cases where the objects are controllable such as electronic appliances in the building.

2.2 An Alternative Approach: Neural Network Object Detection

An NNOD based object identification system (Figure 2.2) contains two phases. First, the training and tuning phase. The first part involves training a convolutional neural networks (CNNs) that is able to classify detect general category objects such as cats, dogs, person, orange, apple, and etc. The second part involves changing the NNOD output categories to specific objects that we want to identify. Second, the query phase, also known as inference, takes in a query image and outputs the identified objects category and their x-y position in the query image.

For the training part, it usually takes a significant amount of time and effort in design-

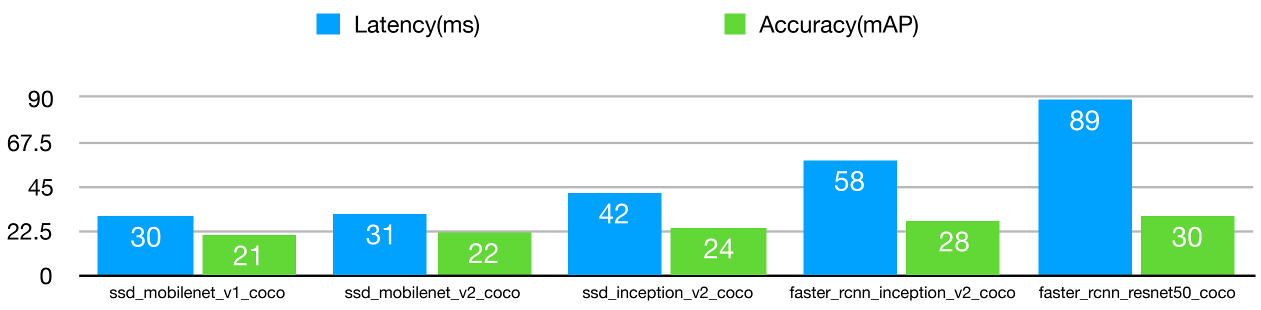


Figure 2.3: Different pre-trained models performance

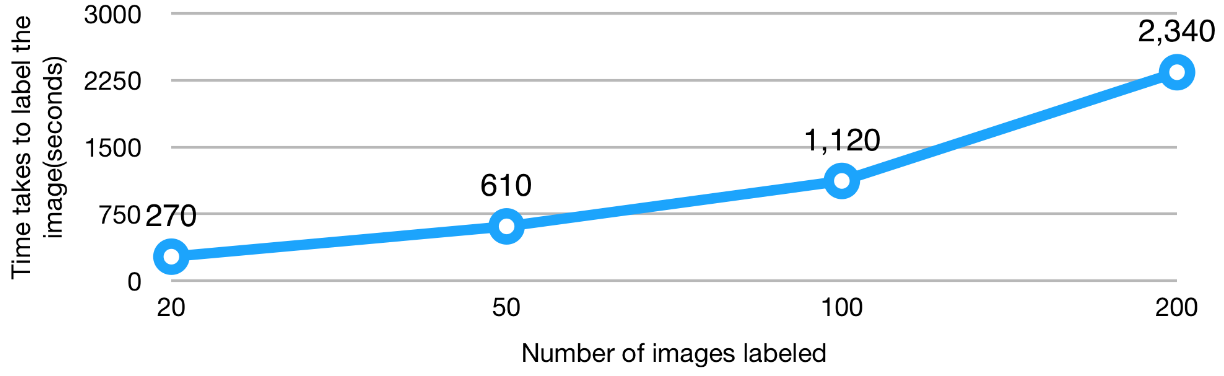


Figure 2.4: Labeling time for different sizes of tuning images

ing the NNOD architecture, gathering and labeling the training images, as well as training the parameters of the NNOD model. However, we can skip this step in most of the cases because there are many well-trained NNOD models that can be download from the internet. For example, in the official Tensorflow github [23], object detection models such as ssd_mobilenet_v1_coco [11], ssd_inception_v2_coco [21], faster_rcnn_inception_v2_coco [8] all allow object detection in a fast and accurate manner. Among these pre-trained object detection models, we choose faster_rcnn_inception_v2_coco as our base NNOD model to do the experiment and evaluation since it provides better accuracy with satisfactory latency. As shown in Figure 2.3, faster_rcnn_inception_v2_coco has 58 ms of inference latency and 28 mAP inference accuracy using an Nvidia GeForce GTX TITAN X card [24].

For the tuning part, since we want our own custom categories to be identified, we need to provide some custom labeled training images, as well as training the NNOD model with some additional epoch (one forward pass and one backward pass of all the training images). As shown in Figure 2.4, the manually labeling time is linear to the number of images manually labeled, which is approximately 10-15 seconds per image. The accuracy of NNOD object identification depend on the number of training image, as well as the number of epoch ran

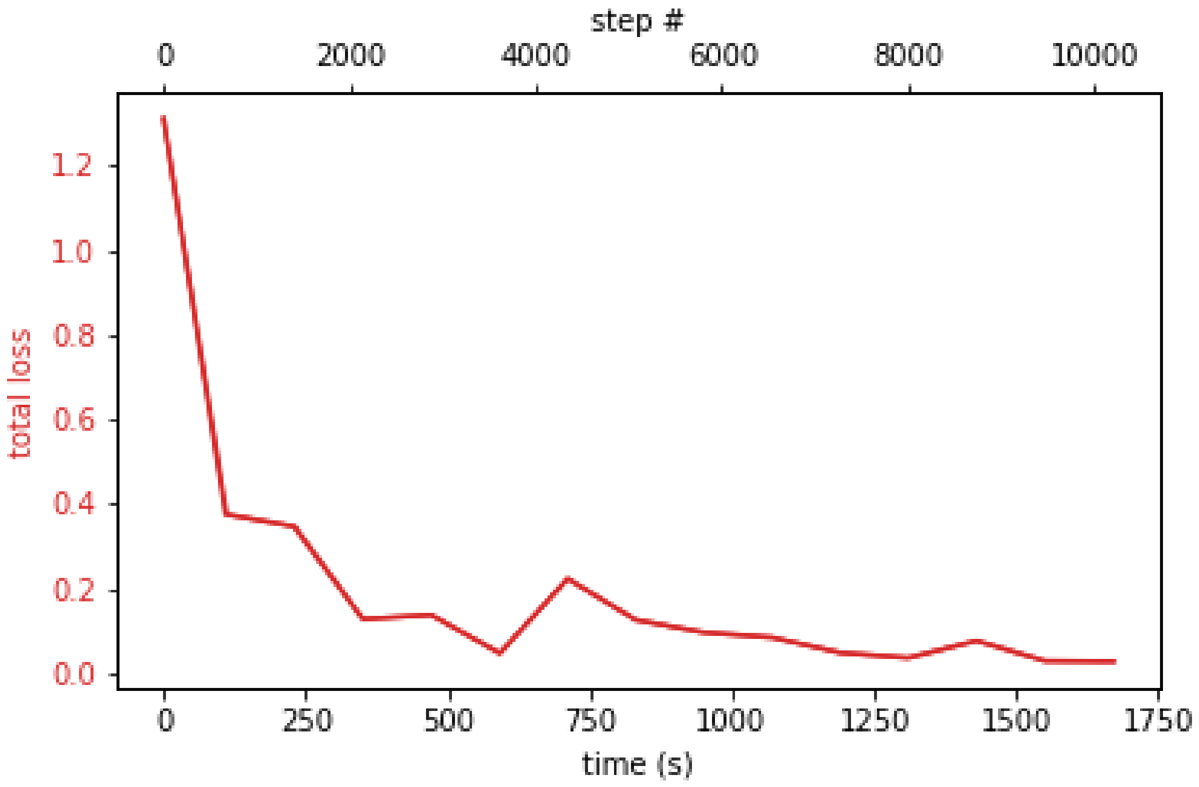


Figure 2.5: Labeling time for different size of tuning images

during train. As shown in Figure 2.5, in our use case, with 200 training images, the loss for custom objects detection is converged to 0.03 after 1671 secs (10212 steps) of training on GPU NVIDIA Quadro M4000 [16].

Chapter 3

ConNect Overview

3.1 System Overview

ConNect is a MAR system that allows a user to point their phone towards an object in a building, and overlay labels of the objects in view on the screen. As shown in Figure 3.1, ConNect comprises the **Offline Deployment Phase**, and the **Online Identification Phase**.

Offline Deployment Phase: First, a general category or tuned category NNOD model will be prepared as stated in Section 2.2. From now on, we define **general objects** to be the type of objects that could recognized by NNOD model and are likely to exist in the building.

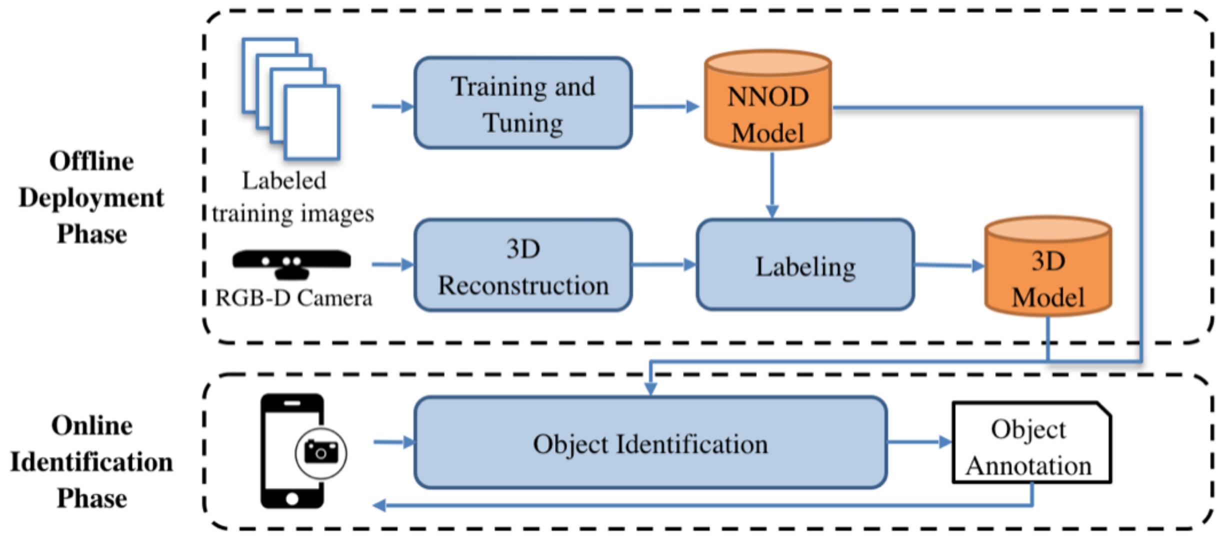


Figure 3.1: Overview of ConNect

Second, a 3D point cloud will be build in this phase, and objects in this 3D model will be labeled as stated in Section 2.1. There will be two types of objects in the 3D model, one of them are the general objects that the NNOD could identify, the other are the **specific objects**, in which future users are interested in but the NNOD models dose not have the specific category to identify. The specific and general objects are labeled manually and easily with the ConNect labeling tool (e.g., click at the object in the image and type in its label). In the case that the NNOD model is tuned to have each specific objects as its category, then we call them **unified objects** to avoid confusion in later Sections.

Online Identification Phase: It consists of two parts, the client and the server. The client, simply a smartphone, constantly previews the video of what the camera sees on the screen. In the background, it sends the camera captures in the form of query images to the server. Therefore, a user, by pointing her phone's camera at the target objects that she wants to get the label, is able to validate from the screen that the camera is indeed pointing at the target. The server, on the other hand, with the query images as input, does two things in parallel. It uses the image to do image localization, as well as NNOD. Both of the two algorithms will tell the server the contents in the query image. The server will then return both results to the client. The client then fuses the results based on which fusion algorithm future user chooses to use, and display the identified label on the screen. Future user chooses which fusion algorithm to use base on how much effort she is willing to spend on the offline deployment phase, as well as how much accuracy does she want to achieve in identifying the objects.

3.2 Goals and Design Choice

In designing ConNect, there are two goals we want to achieve. One goal is to achieve greater accuracy than what is possible solely using image localization (SnapLink [7]) or NNOD. The other goal is to reduce the false positives. That is, when we are unconfident of where a certain object exists in the query image, we would rather not allow a user to see the unconfident annotation. This is a better strategy than allowing a user to see the annotation of a certain object that actually does not exist in the screen/query image. We choose this design because if the annotation of the objects are confidential, or the annotation is able to allow a user to control the object such as electronic appliance, a false positive is much more dangerous than a false negative for security reasons.

Chapter 4

Fusion algorithms

4.1 Image Localization with general NNOD model

As stated in Section 2, there are many available well-trained object detection models to use. These models will identify some general category objects such as chairs, tables, TVs. However, to allow it to identify all the specific objects in our use case, we need to spend additional significant amount of time and effort to tune the model (hours of labeling and training). Therefore, if time and computation resources (e.g., GPUs) are concerns for our use case, we developed a fusion algorithm that require no tuning of the NNOD model.

The pseudocode of the first fusion algorithm is shown below. This algorithm is designed based on two assumptions. First, false positive of NNOD is low, that is, NNOD will unlikely say there is a general object O in the image (x, y), but actually there is nothing there. Second, the false positive of image localization is higher than the false positive of NNOD, as discussed in Section 2.1. This is why in the fusion algorithm we trust NNOD more than image localization when there is a mismatch.

```
Algorithm 1: fusion algorithm 1
  Data: U is a hash table of the identified result from image localization
        V is a hash table of the identified result from NNOD
        Each entry of U and V are in the form of (name, (x, y))
  Result: Identified objects' labels after fusion
1 $conflict \longleftarrow false$;
2 $fused\_result \longleftarrow []$;
3 if (V not contains general objects) and (U contains general objects) then
4     return fused_result;
5 for $v \in V$ do
6     if v.name not in U then
7         $conflict \longleftarrow true$;
8         break;
9     else
10        if distance(v, U.get(v.name)) > threshold then
11            $conflict \longleftarrow true$;
12            break;
  /* only return objects when 2 results are not conflicting */
13 if conflict = false then
14     for $u \in U$ do
15         if u.name is specific object then
16             fused_result.add(u);
17 return fused_result;
```

Let's say our targeting scenario is the conference room in Figure 4.1. The items that a user is really interested in are the two drawings on the left and right of the wall. However, we only have a general NNOD model that can detect a TV, projector, and whiteboard, but not drawings.

There are four possible outcomes:

1. The image localization returns a correct result, and the NNOD also returns a correct result. Since the localization result consists of both general and specific objects, we compare the location of general objects in the localization result with the location of identified objects in NNOD result. Their difference will be insignificant since both of them are identified correctly. Then we can return all the specific objects for the user to see the annotation.

2. The image localization returns an incorrect result, but the NNOD returns a correct result. Then, there will be a mismatch. Based on the algorithm, the query image will be discarded, and an empty list of objects will be returned. As a result, the user will not see any annotation on the screen.

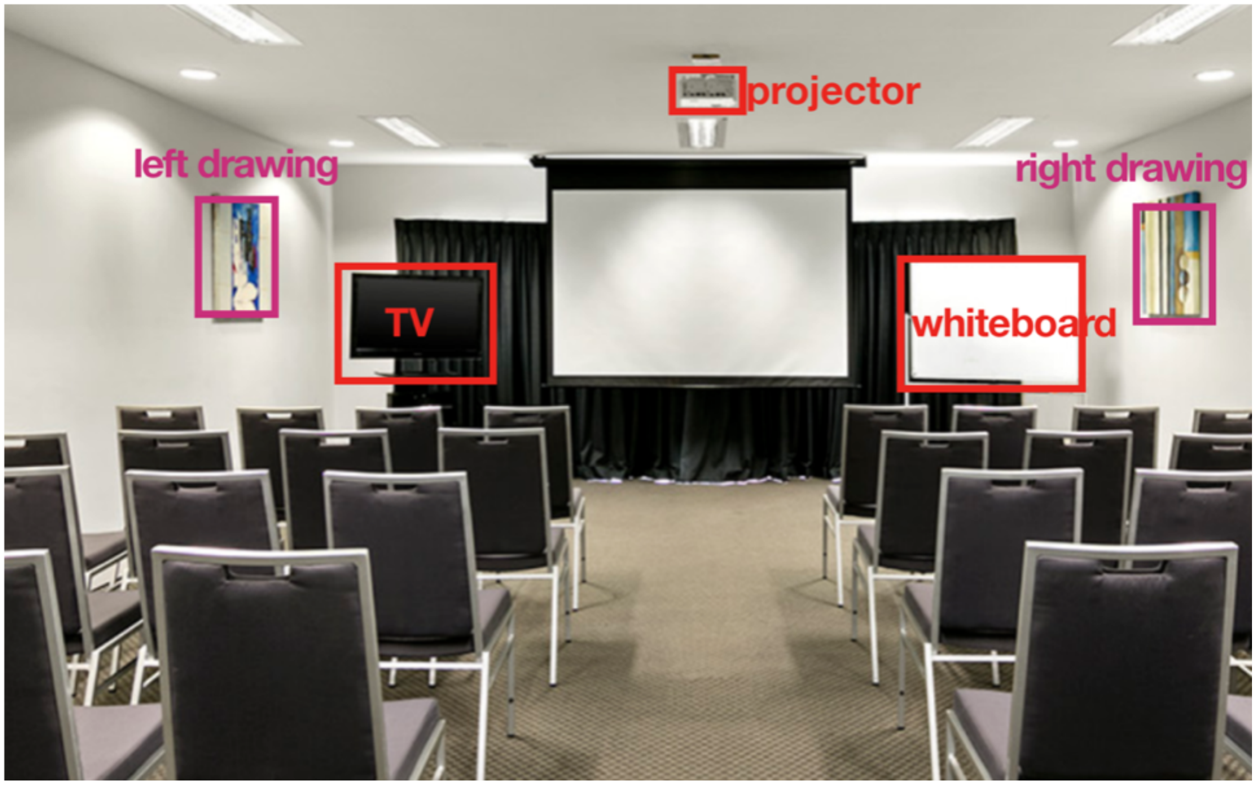


Figure 4.1: Scenario of using ConNect fusion

3. The image localization returns a correct result, but the NNOD returns an incorrect result. Based on our previous assumption, NNOD's error will be caused by a false negative instead of a false positive. That is, it is incorrect by, Case 1, missing some general objects or, Case 2, missing all general objects. In Case 1, the fusion algorithm will take the remaining NNOD identified general objects. It will then calculate the distance between these objects identified via NNOD and identified via image localization. Since image localization is identifying correctly and NNOD is identifying general objects partial correctly, the distance will be less than a threshold. The correct specific objects will be returned for the user to see annotation on the screen. In Case 2, based on the fusion algorithm, the query image will be discarded and the user will not see any annotation on the screen even though image localization returns the correct result. However, this is the trade-off we have to make as discussed at the end of Section 3. We allows prefer false negatives to false positive.

4. The image localization returns an incorrect result, and the NNOD also returns an incorrect result. In this case, it is very unlikely that the two incorrect results will match (In our experiments of hours of testing on ConNect, this never ocurred). Therefore a empty list of objects will be returned and the user could not see any annotation on the

screen.

By doing so, we do not improve the accuracy of the system by only applying image localization based object identification. Rather, we provide a method to change most of the false positive case of image localization to NULL results. This is a significant contribution as discussed in Section 3. In addition, our Fusion Algorithm 1 provides a minimal way of combining image localization based and NNOD based object identification. It does not require a person to train or tune the NNOD model offline.

4.2 Image Localization with tuned NNOD model

If time and computation resources are not constrained to the person who is doing the offline deployment phase, then ConNect have the capability of improving the accuracy after fusing the results of image localization and NNOD, by applying our Fusion Algorithm 2 shown below.

Algorithm 2: fusion algorithm 2

```
Data: U is a hash table of the identified result from image localization
      V is a hash table of the identified result from NNOD
      Each entry of U and V are in the form of (name, (x, y))
Result: Identified objects' labels after fusion
$conflict \longleftarrow false$;
$fused\_result \longleftarrow []$;
if V is empty and U is not empty then
    return fused_result;
for $v \in V$ do
    if v.name not in U then
        $conflict \longleftarrow true$;
        break;
    else
        if distance(v, U.get(v.name)) > threshold then
            $conflict \longleftarrow true$;
            break;
/* when there is no conflict, return all the objects */
if conflict = false then
    fused_result.add(U);
/* when there is conflict, only return objects identified by NNOD */
if conflict = true then
    fused_result.add(V);
return fused_result;
```

The scenario is the same as in the previous sub-section, but this time, both labels in the image localization 3D model and NNOD categories identify the same two unified objects, left-drawing, and right-drawing, as shown in Figure 4.2. When we are running ConNect and using Fusion Algorithm 2, again there are four possible outcomes:

1. The image localization returns a correct result, and the NNOD also returns a correct result. Then both of the results will contain the same set of unified objects, and their difference will be insignificant since both of them are identified correctly. Then we can return all of the specific objects for user to see the annotation.

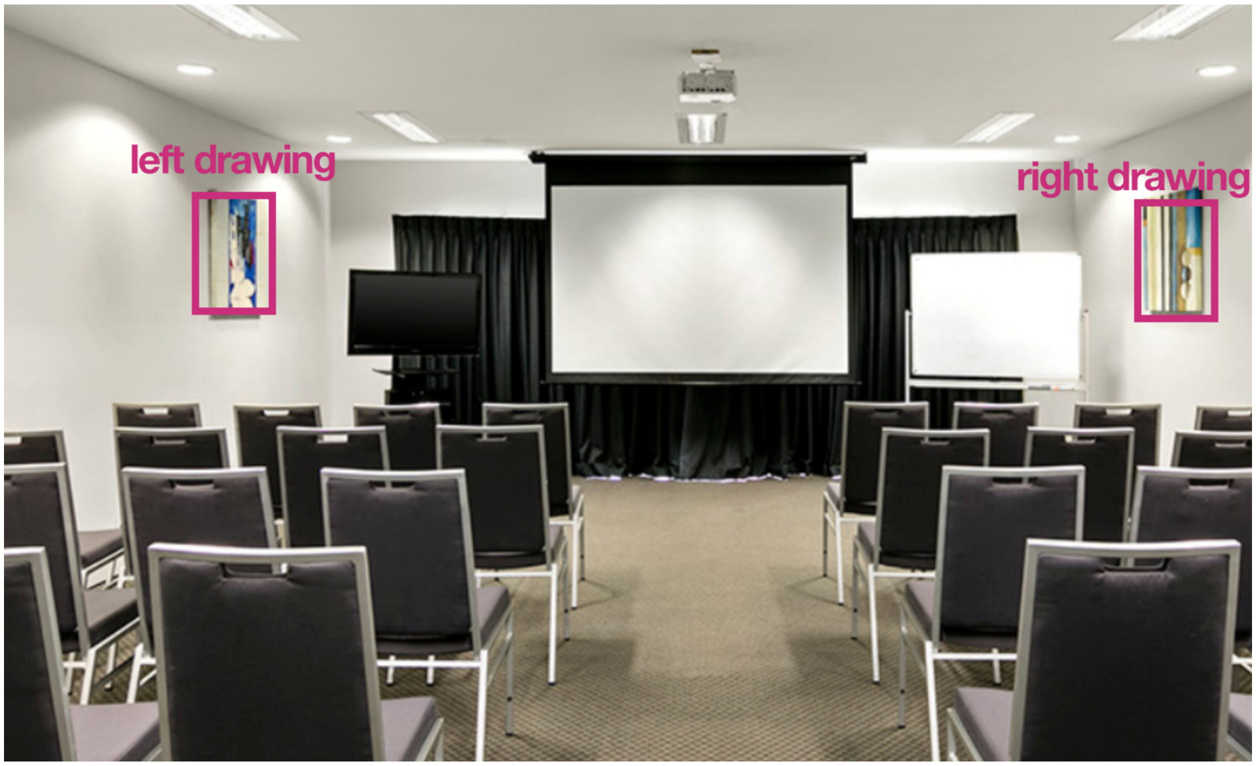


Figure 4.2: Scenario of using ConNect fusion2

2. The image localization returns an incorrect result, but the NNOD returns a correct result. There will be a mismatch. The conflict in the algorithm will be set to true, and all the unified objects identified in NNOD will be returned to the user. Again, we are making this decision based on the assumption that the false positive of NNOD is very low. Whenever NNOD returns some identified objects, then the object must be there. Therefore, in this case, part or all the unified objects in the query image will be annotated on the screen. The important thing here is that there is no false positive; The user should never see a annotation that should not appear on the screen.

3. The image localization returns a correct result, but the NNOD returns an incorrect result. As discussed above, it happens only when a subset of unified objects are identified in the NNOD. In the case of NNOD identify at least one unified object, the identified objects will be used to compare to the image localization result. It will eventually return all the annotations in its image localization result, which is the desired outcome. On the other hand, in the case that NNOD identifies no object, our fusion algorithm choose to return an empty list. That is because of prioritizing false negative over false positive.

4. The image localization returns an incorrect result, and the NNOD also returns an

incorrect result. In this case, again, it is very unlikely that the two incorrect results will be matched (Again, in our experiments of hours of testing on ConNect, this never happened). Therefore a empty list of objects will be returned and the user could not see any annotation on the screen.

Chapter 5

System Implementation

5.1 Offline deployment phase

For the offline deployment phase, when we are preparing 3D models for image localization, we choose to use RTABMap [14] because it is open source and supports various types of hardware such as stereo camera (camera with two or more lenses with a separate image sensor or film frame for each lens) and RGB-Depth camera. RTABMap allows us to use a RGB-Depth camera to take a video of a room as input. It will then output the 3D model of that room for image localization to use. We also implemented a ConNect labeling tool in 400+ lines of C++ code, using Qt as the GUI library. It currently reads RTABMap [14] databases and presents images to users for labeling. Labels are saved to the original database file. In the future, we plan to add adapters to other popular 3D reconstruction tools, such as Google Tango [22] and Bundler [1]. When we are preparing NNOD models, we use LabelImg [15]. It is a open sourced labeling tool to provide labeled images for NNOD to train on. We use Tensorflow Object Detection API [18] to train and tune our NNOD model.

5.2 Online identification phase

For the online identification phase, we implemented our ConNect image localization runtime in 3200+ lines of C++ code. It receives queries in a RESTful interface using GRPC. We use the Qt event system to construct the image localization pipeline. In the pipeline, we use OpenCV [5] to extract SURF [4], solve perspective-n-point (PnP) [9] problems, and project points between 2D and 3D. Perspective-n-Point is the problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the rotation (roll, pitch, and yaw) and 3D translation of the camera with respect to the world. We use the randomized KD-Tree implementation in FLANN [17] and transformation utilities from PCL [19]. On the other hand, we implemented our NNOD run time in 200+ lines of Python code. It also receives queries in a RESTful interface using

GRPC. On the client side, we developed an Android client using 1700+ lines of Java code. It provides an ability to access images from the camera, and to send the images to the server, as well as getting the result from the server and to fuse the result.

5.3 Detailed workflow of ConNect

The detailed workflow of ConNect using the tools mentioned above is summarized in Figure 5.1.

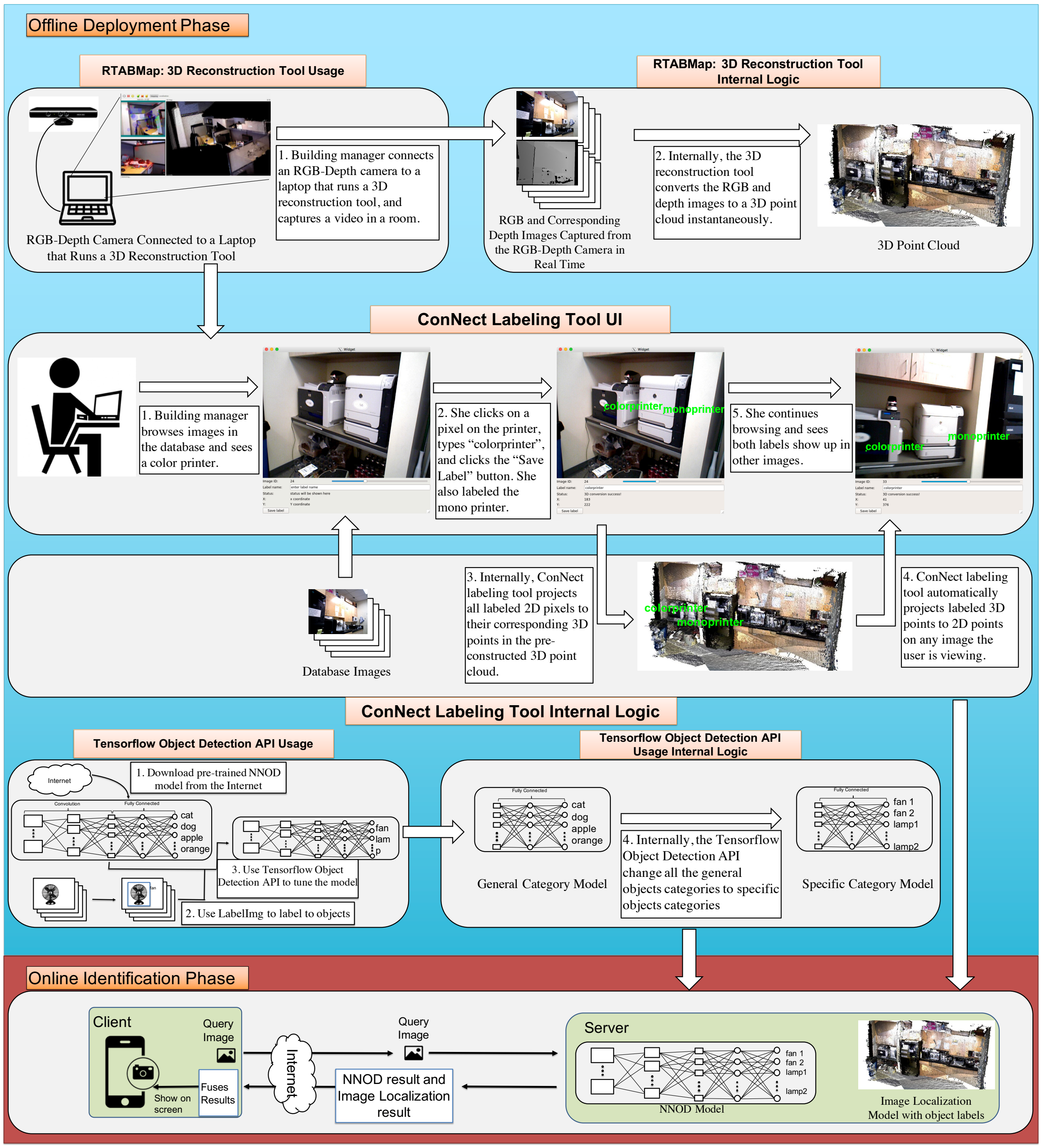

Offline Deployment Phase
RTABMap: 3D Reconstruction Tool Usage
RTABMap: 3D Reconstruction Tool Internal Logic
1. Building manager connects an RGB-Depth camera to a laptop that runs a 3D reconstruction tool, and captures a video in a room.
RGB-Depth Camera Connected to a Laptop that Runs a 3D Reconstruction Tool
RGB and Corresponding Depth Images Captured from the RGB-Depth Camera in Real Time
2. Internally, the 3D reconstruction tool converts the RGB and depth images to a 3D point cloud instantaneously.
3D Point Cloud
ConNect Labeling Tool UI
1. Building manager browses images in the database and sees a color printer.
2. She clicks on a pixel on the printer, types "colorprinter", and clicks the "Save Label" button. She also labeled the mono printer.
5. She continues browsing and sees both labels show up in other images.
colorprinter
monoprinter
Database Images
3. Internally, ConNect labeling tool projects all labeled 2D pixels to their corresponding 3D points in the pre-constructed 3D point cloud.
4. ConNect labeling tool automatically projects labeled 3D points to 2D points on any image the user is viewing.
ConNect Labeling Tool Internal Logic
Tensorflow Object Detection API Usage
Tensorflow Object Detection API Usage Internal Logic
Internet
1. Download pre-trained NNOD model from the Internet
Convolution
Fully Connected
cat
dog
apple
orange
fan
lamp
3. Use Tensorflow Object Detection API to tune the model
2. Use LabelImg to label to objects
General Category Model
4. Internally, the Tensorflow Object Detection API change all the general objects categories to specific objects categories
fan 1
fan 2
lamp1
lamp2
Specific Category Model
Online Identification Phase
Client
Query Image
Show on screen
Fuses Results
Internet
Query Image
NNOD result and Image Localization result
Server
NNOD Model
Image Localization Model with object labels

Figure 5.1: Detailed workflow of ConNect

Chapter 6

Evaluation

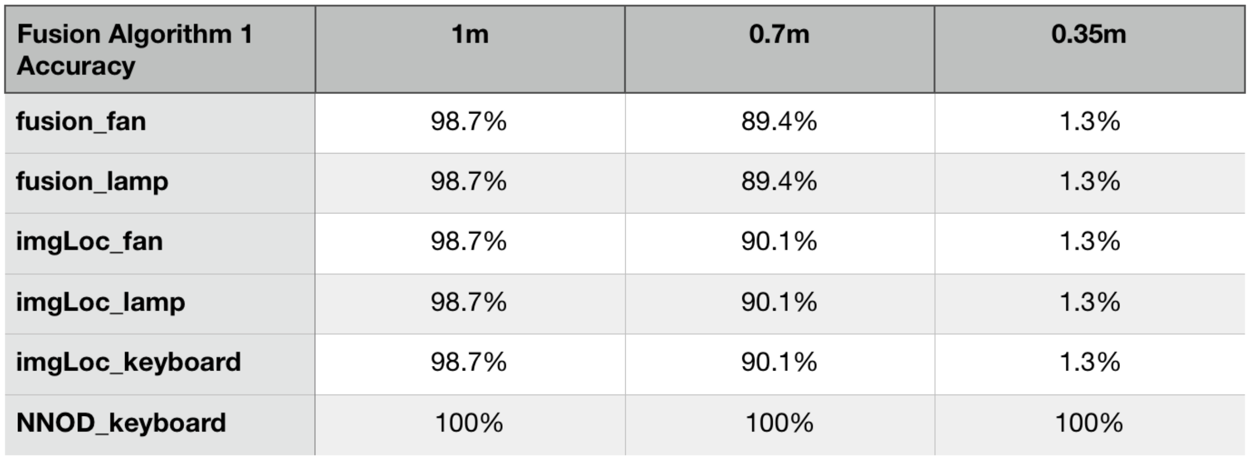
6.1 Accuracy

We evaluate the accuracy for both Fusion algorithms. The experiment is conducted as follows: the user wanted to identify two objects that are close to each other, a fan, and a lamp. User tried to identify them using her phone at distance 1m, 0.7m, 0.35m. For each distance from the objects (fan and lamp), the client software of ConNect constantly sends to the server around 150 query images in total. After receiving the results from server and fuse them using the Fusion Algorithm, client software showed all three type of results (image localization result, NNOD result, fusion result) on the screen for evaluation purpose. Then we manually labeled and calculated the correctness for all of them. Based on extensive experimentation, 1m is the distance that image localization performed very well. 0.7m is the distance that image localization has lower accuracy. 0.35m is the distance that image localization gives completely unreliable results. For the threshold in both algorithms, we use 50 pixels distance. This allows image localization and NNOD to agree with each other even though their results are slightly different in a acceptable range, that is, 50 pixels distance in our experiments. The reason why we choose 50 pixels distance is because we believe that if the result is off by 50 pixels from its true location, it should be considered as a wrong result.

Accuracy of Fusion Algorithm 1

For Fusion Algorithm 1, NNOD is only capable of identifying general class objects. Therefore it could not improve the accuracy on its own. The point of this subsection is to show that, in most cases, Fusion Algorithm 1 does not lower the accuracy compared to solely using image localization. For the experiment conducted in this subsection, we made sure fan, light, and keyboard all appears in the screen view.

As shown in Table 6.1, at distance 1m, the accuracy of image localization is 98.7%. On the other hand, the accuracy of Fusion Algorithm 1 is also 98.7%. This shows that Fusion Algorithm 1, using NNOD to complement image localization, does not lower the accuracy.



Fusion Algorithm 1 Accuracy	1m	0.7m	0.35m
fusion_fan	98.7%	89.4%	1.3%
fusion_lamp	98.7%	89.4%	1.3%
imgLoc_fan	98.7%	90.1%	1.3%
imgLoc_lamp	98.7%	90.1%	1.3%
imgLoc_keyboard	98.7%	90.1%	1.3%
NNOD_keyboard	100%	100%	100%

Figure 6.1: Accuracy for Fusion Algorithm 1 at different distances from the objects

A similar thing occurred at distance 0.7m (Table 6.1). The image localization accuracy is lowered to 90.1% compared to its accuracy at distance 1m. However, Fusion Algorithm 1 also yields similar accuracy, 89.4%. There is one frame that image localization identified correctly, but the Fusion Algorithm discarded the result. That is because the distance between the NNOD's detected keyboard position and the image localization detected keyboard position are slighter greater than the threshold. This could be resolved in the future when the threshold is tuned to be slightly larger.

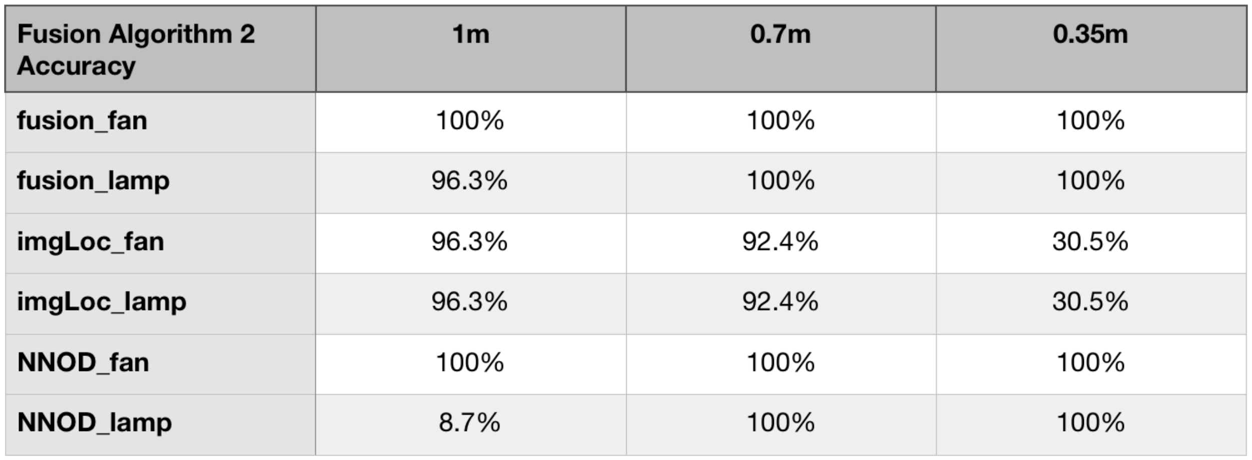
At distance 0.35m, as shown in Table 6.1, image localization yield random results. Therefore the accuracy is down to 1.3%. We can see that this Fusion Algorithm does not lower the accuracy in the case that the distance is 0.35.

Accuracy of Fusion Algorithm 2

For Fusion Algorithm 2, NNOD is capable of identifying the unified objects. Therefore, different from the accuracy Table for Fusion Algorithm 1, we record NNOD_fan and NNOD_lamp, which is the accuracy of NNOD correctly identifying unified objects.

For distance 1m, we see in Table 6.2 that both imgLoc_fan and imgLoc_lamp are 96.3% accurate. On the other hand, NNOD is identifying a fan 100% accuratly but performs poorly on identifying the lamp. However, after Fusion Algorithm 2, the fused result for fan is 100% accurate due to the 100% accuracy of NNOD fan identification. And yet fusion_lamp achieved 96.3% accuracy due to 96.3% accuracy of imgLoc_lamp. We can observe that there is accuracy improvement after fusion.

For distance 0.7m, we saw in Table 6.2 that both imgLoc_fan and imgLoc_lamp are 92.4% accurate. On the other hand, NNOD is identifying a fan and a lamp with 100% accuracy. Then undoubtedly, after Fusion Algorithm 2, both the fused result for fan and lamp are 100% accurate due to the 100% accuracy of NNOD. We can observe that accuracy is improved



Fusion Algorithm 2 Accuracy	1m	0.7m	0.35m
fusion_fan	100%	100%	100%
fusion_lamp	96.3%	100%	100%
imgLoc_fan	96.3%	92.4%	30.5%
imgLoc_lamp	96.3%	92.4%	30.5%
NNOD_fan	100%	100%	100%
NNOD_lamp	8.7%	100%	100%

Figure 6.2: Accuracy for Fusion Algorithm 2 at different distances from the objects

significantly in this case.

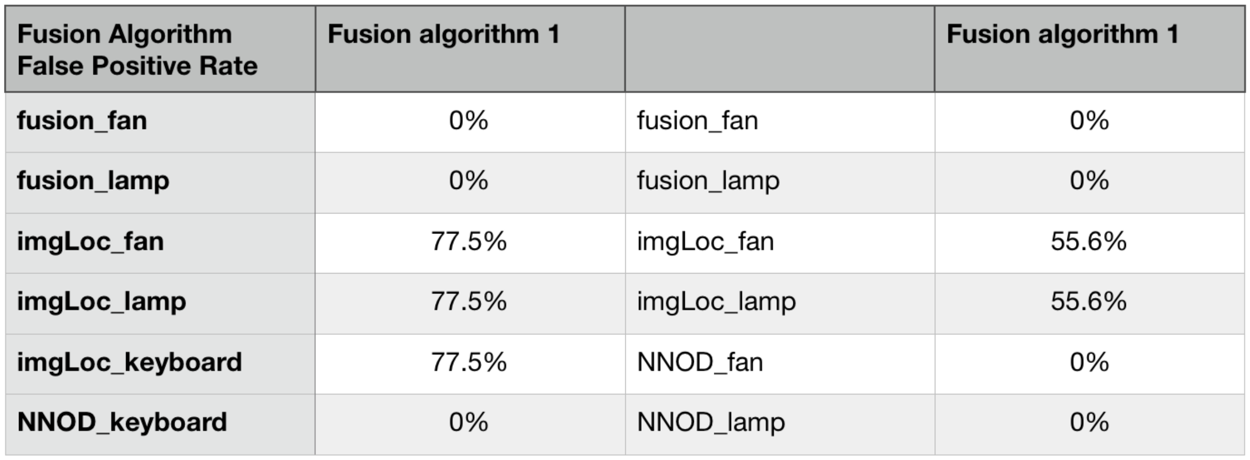
For distance 0.35m, we saw in Table 6.2 that both imgLoc_fan and imgLoc_lamp are 30.5% accurate. On the other hand, NNOD is identifying a fan and a lamp with 100% accuracy. Then undoubtedly, after Fusion Algorithm 2, both the fused result for fan and lamp are 100% accurate due to the 100% accuracy of NNOD. We can observe that accuracy is improved significantly in this case.

6.2 False positive rate

We also evaluate the false positive rate for both Fusion Algorithms. The experiment was conducted as follow. User pointed his phone in a direction where there is nothing there to identify. Then client software of ConNect sent around 150 query images to the server and got the identified result for both image localization and NNOD. Then the client software fused the result and showed all of them on the screen for evaluation purpose. We manually calculated the false positive rate for all of them.

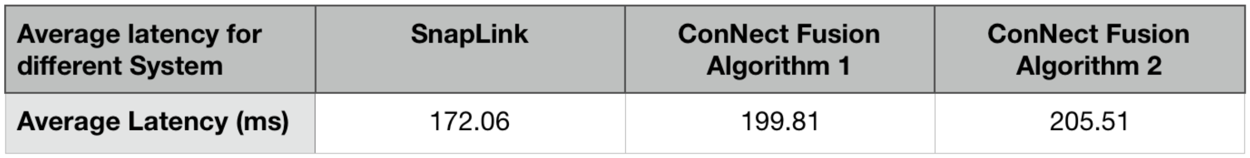
False positive rate of Fusion Algorithm 1

As shown in Table 6.3, when the phone is pointed to a direction with no objects, image localization still has a very high chance, 77.5%, of falsely identifying a fan and a lamp. However, since NNOD never identified a keyboard at all, the false positives of image localization is discarded. Therefore, our Fusion Algorithm 2 yielded no false positives, which is desired since the phone is pointing at nothing, the user should not see anything being identified.



Fusion Algorithm False Positive Rate	Fusion algorithm 1		Fusion algorithm 1
fusion_fan	0%	fusion_fan	0%
fusion_lamp	0%	fusion_lamp	0%
imgLoc_fan	77.5%	imgLoc_fan	55.6%
imgLoc_lamp	77.5%	imgLoc_lamp	55.6%
imgLoc_keyboard	77.5%	NNOD_fan	0%
NNOD_keyboard	0%	NNOD_lamp	0%

Figure 6.3: False positive rate for both Fusion Algorithms



Average latency for different System	SnapLink	ConNect Fusion Algorithm 1	ConNect Fusion Algorithm 2
Average Latency (ms)	172.06	199.81	205.51

Figure 6.4: Latency comparison between SnapLink (image localization) and ConNect two Fusion Algorithms

False positive rate of Fusion Algorithm 2

As we can see from Table 6.3, again, image localization gives a very high false positive rate. However, our Fusion Algorithm 2 is able to use NNOD results to correct all the false positives suggested by image localization.

6.3 Latency

End-to-end average latency is measured for ConNect's two Fusion Algorithms, as well as SnapLink, which only uses image localization to do object identification. As we can see from Table 6.4, the latency of both ConNect Fusion Algorithms are higher than SnapLink. This is not surprising because ConNect is doing everything SnapLnik does, with the addition of NNOD object identification and fusion. For Fusion Algorithm 1, the latency is 27.75 ms (16.1%) higher than SnapLink. For Fusion Algorithm 2, the latency is 33.45 ms (19.4%) higher than SnapLink.

Chapter 7

Summary and Future work

In this paper, we present ConNect, an accurate and responsive vision-based object identification system that enables users to see annotation of the many objects in the building by using ubiquitous smartphones. This work proposes two innovative ways of fusing image localization with Neural Network Object Detection to achieve higher accuracy and lower false positive rate in identifying the objects. We showed that for Fusion Algorithm 1, where less work is needed in the offline deployment phase, achieves the same accuracy as image localization, but much less false positives. We also showed that for Fusion Algorithm 2, where more work need to be done in the offline deployment phase, achieves higher accuracy and much less false positives than image localization. Two Fusion Algorithms introduce additional latency (less than 20%). Comparing the problem of misidentifying objects, we believe the additional latency is acceptable.

As for future work, we plan to move ConNect toward a smart building service, where many more applications can benefit from object detection inside buildings. In addition, we also plan to deploy ConNect's server on Edge devices, such as the NVIDIA Jetson TX2 Module [13], a power-efficient embedded AI computing device. This will potentially reduce the latency of ConNect's online identification phase. In the past, we spent some time on deploying ConNect's server on Jetson TX2 Module, but gave up after 2 month of trying. This is because it is a new product released by NVIDIA and the insufficient software support makes it difficult to develop ConNect on it. However, we believe in the near future, edge devices such as Jetson TX2 Module will become the perfect machine to deploy ConNect.

Bibliography

[1] Sameer Agarwal et al. "Building rome in a day". In: *Communications of the ACM* 54.10 (2011), pp. 105–112.

[2] *Google ARCore*. `https://developers.google.com/ar/`. 2017.

[3] *Apple ARKit*. `https://developer.apple.com/arkit/`. 2017.

[4] Herbert Bay et al. "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[5] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.

[6] Dimitris Chatzopoulos et al. "Mobile Augmented Reality Survey: From Where We Are to Where We Go". In: *IEEE Access* (2017).

[7] Kaifei Chen et al. "SnapLink: Fast and Accurate Vision-Based Appliance Control in Large Commercial Buildings". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.4 (2017), 129:1–129:27.

[8] Xinlei Chen and Abhinav Gupta. "An implementation of faster RCNN with study for region sampling". In: *arXiv preprint arXiv:1702.02138* (2017).

[9] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Readings in computer vision*. Elsevier, 1987, pp. 726–740.

[10] *Microsoft HoloLens*. `https://www.microsoft.com/en-us/hololens`. 2017.

[11] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[12] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. "Overlay: Practical mobile augmented reality". In: *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM. 2015, pp. 331–344.

[13] *NVIDIA Jetson TX2 Module*. `https://developer.nvidia.com/embedded/buy/jetson-tx2`. 2018.

[14] Mathieu Labbe and Francois Michaud. "Appearance-based loop closure detection for online large-scale and long-term operation". In: *Robotics, IEEE Transactions on* 29.3 (2013), pp. 734–745.

[15] *LabelImg*. `https://github.com/tzutalin/labelImg`. 2018.

[16] *NVIDIA Quadro M4000*. `http://www.pny.com/nvidia-quadro-m4000`. 2018.

[17] Marius Muja and David G Lowe. "Scalable nearest neighbor algorithms for high dimensional data". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2227–2240.

[18] *Tensorflow Object Detection API*. `https://github.com/tensorflow/models/tree/master/research/object_detection`. 2018.

[19] Radu Bogdan Rusu and Steve Cousins. "3d is here: Point cloud library (pcl)". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1–4.

[20] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. "Fast image-based localization using direct 2d-to-3d matching". In: *2011 International Conference on Computer Vision*. IEEE. 2011, pp. 667–674.

[21] Christian Szegedy et al. "Inception-v4, inception-resnet and the impact of residual connections on learning." In: *AAAI*. Vol. 4. 2017, p. 12.

[22] *Google Project Tango*. `https://get.google.com/tango/`. 2017.

[23] *Tensorflow detection model zoo*. `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md`. 2018.

[24] *NVIDIA TITAN X*. `https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/`. 2018.