12-2020

Vision-based analytics for improved AI-driven IoT applications

Amit SHARMA
Singapore Management University

VISION-BASED ANALYTICS FOR IMPROVED AI-DRIVEN IoT APPLICATIONS

AMIT SHARMA

SINGAPORE MANAGEMENT UNIVERSITY
2020

Vision-based Analytics for Improved AI-driven IoT Applications

by

Amit Sharma

Submitted to School of Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Archan MISRA (Supervisor / Chair)
Professor of Information Systems
Singapore Management University

Qianru SUN
Assistant Professor of Information Systems
Singapore Management University

Hwee Pink TAN
Associate Professor of Information Systems
Singapore Management University

Rajeev RASTOGI
Vice-President, Machine Learning
Amazon Inc.

Singapore Management University
2020

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.

Amit Sharma

13 February 2021

Vision-based Analytics for Improved AI-driven IoT Applications

Amit Sharma

Abstract

Proliferation of Internet of Things (IoT) sensor systems, primarily driven by cheaper embedded hardware platforms and wide availability of light-weight software platforms, has opened up doors for large-scale data collection opportunities. The availability of massive amount of data has in-turn given way to rapidly growing machine-learning models e.g. You Only Look Once (YOLO), Single-Shot-Detectors (SSD) and so on. There has been a growing trend of applying machine learning techniques, e.g., object detection, image classification, face detection etc., on data collected from camera sensors and therefore enabling plethora of vision-sensing applications namely self-driving cars, automatic crowd monitoring, traffic-flow analysis, occupancy detection and so on.

While these vision-sensing applications are quite useful, their real-world deployments can be challenging for various reasons namely DNN performance drop on data collected in-the-wild, high energy consumption by vision sensors, privacy concerns raised by the captured audio/video data and so on. This dissertation explores how a combination of IoT sensors and machine-learning models can help resolve some of these challenges. It proposes novel vision-analytics techniques, aimed at improving the large-scale adoption of vision-sensing techniques, with their potential performance improvements demonstrated by using two different vision-sensing systems namely *SmrtFridge* and *CollabCam* .

First, this dissertation describes SmrtFridge system, which uses a combination of embedded RGB & Infrared (IR) camera sensors and a machine-learning model for automatic food item identification and residual quantity sensing. *SmrtFridge* adopts a user interaction-driven sensing approach which is triggered as and when a

user is interacting (adding/removing items) with any food item. Using two different processing pipelines, i.e., motion-vector based and IR based, *SmrtFridge* isolates the food item from the other background objects that might be present in the captured images. The segmented items are then assigned a food label by an image classifier. *SmrtFridge* shows that using these segmentation techniques can help convert the item identification problem from a complex object-detection problem to a relatively simpler object-classification problem. Also, *SmrtFridge* proposes a novel IR based residual quantity estimation technique which can quantify the residual content inside food item containers (transparent/opaque) of various shapes, sizes and material types.

Secondly, this dissertation presents CollabCam, a novel and distinct multi-camera collaboration framework for energy efficient visual (RGB) sensing in a large-scale camera deployment. *CollabCam* exploits the partially overlapping FoVs of cameras to selectively reduce imaging resolution in their mutually common regions. This resolution reduction can enable overall energy savings of a camera sensor by reducing the energy consumption in image capture, optional storage and network transmission. *CollabCam* proposes novel techniques for (a) autonomous and accurate estimation of overlapping regions between a pair of cameras (b) mixed-resolution sensing where selected regions of an image are captured at lower resolution, whereas the remaining regions are captured at default (higher) resolution and (c) collaborative object inference where a modified DNN model, called *CollabDNN*, utilizes the perspective of other collaborating cameras to enhance performance of object detection on low-resolution images. Application of *CollabCam* techniques on two publicly available datasets demonstrates the potential high energy savings for a multi-camera system and takes a step towards making energy efficient large-scale vision-sensing systems a reality.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The market for Internet-of-Things (IoT) devices is expected to witness explosive growth in the next decade [7]. Such growth is driven both by the proliferation of low-cost sensors (such as infra-red sensors [19, 6], visual (RGB) cameras [13] and even short-range radar [18]) and the increasing availability of powerful embedded device platforms (e.g., Nvidia's Jetson Nano platform [10]) that allows the execution of neural network models to extract real-time intelligence from such multi-modal sensor data. Indeed, it is interesting to note that even previously expensive and bulky sensing modalities are gradually becoming available in consumer-grade devices such as smartphones—e.g., Google's Pixel 4 smartphone includes an integrated radar sensor for short-range sensing of gestural motion [9], while Apple's has already integrated LiDAR sensors [1] in several of its products namely iPad pro, iPhone 12 pro. The pervasive deployment of such multi-modal sensing capabilities is ushering in a variety of novel applications across many domains, including smart city initiatives [56, 16, 15], smart healthcare [28, 105, 4], home automation [20], vision based gesture [26, 83], among others.

Visual sensing represents one of the most powerful sensing modalities for these emerging categories of IoT-based applications. The rapid adoption of visual sensing is driven by two key trends. First, there has been a dramatic increase in the quality (resolution), and a corresponding decrease in cost, of such visual sensors.

As an example, most flagship consumer smartphones now routinely embed multiple cameras, with resolution as high as 108 megapixels [2], and low-cost embedded platforms (e.g., Raspberry Pi & Arduino) now routinely support the integration of network-connected cameras [14]. Second, with the introduction of Deep Neural Network (DNN) models, there has been a spectacular improvement in the accuracy of computational models for a variety of vision-based tasks, such as object detection and classification [104, 58], object & face recognition [70, 59], medical image classification [81, 77] and human activity recognition [116, 75]. As an illustrative exemplar, the field of object identification (a crucial component of many vision-based applications) has been revolutionized by the (a) increased public availability of large annotated datasets (e.g., ImageNet [99], Pascal VOC [41], COCO [73]) and (b) the development of increasingly sophisticated and deeper DNN models (such as faster-RCNN [48, 47, 95], Single-Shot-Detector (SSD) [74] and YOLO [94, 93]). Indeed for many applications, machine-based vision intelligence now offers performance comparable to, and even superior than, human baselines [113].

Notwithstanding these impressive achievements, there are still several limitations before such DNN-based visual sensing pipelines can be effectively employed to support a wide variety of pervasive IoT applications. The proposed work in this thesis is motivated by the identification of some of these limitations, especially related to the use of DNN-based vision pipelines on data from vision sensors embedded in pervasive devices deployed in *in-the-wild* settings (such as homes, campuses and markets). To motivate my research direction, I first list out some of these key limitations.

1.0.1 Key Challenges with Pervasive Visual Sensing

While the performance of various DNN based object identification techniques have improved significantly in recent times, practical adoption of vision based applications on pervasive IoT devices can still prove out to be challenging for many reasons,

some of which are discussed below.

1. **Performance Degradation under Diverging Training & Test Images**: State-of-the-art DNN based object detection models are trained on a set of representative images for each object class supported by the model. While additional DNN attributes such as the structure of the neural network model or the dynamics of hyper-parameter tuning also affect the model's performance, generically speaking, the model performance is heavily influenced by the *quality* and *representativeness* of the training images. In particular, DNN-based models report high accuracy when the test images (gathered by a visual sensor embedded in an in-the-field IoT device) are similar to the images in the training corpus. However, for many pervasive scenarios, the images captured during the 'test phase' are not as *clean* as the images utilized for training. This discrepancy arises due to a variety of artifacts, such as, but not limited to, variable lighting/illumination conditions, significant variations in the nature of the background or levels of clutter, diversity in viewing angles and zoom levels (which often change the size of the object of interest relative to other objects) and partial/occluded views. If pervasive applications are to be widely deployed, it is important to develop techniques that allow such DNN models to perform *robustly*, across these myriad deployment instance-specific variations, as it is clearly infeasible to train instance-specific models with an explicitly curated *deployment-specific* set of training images. For example, consider a smart fridge that uses an outward-facing camera sensor to identify food items—such identification should be possible in spite of the significant differences in the background scenery (across houses, office spaces and commercial establishments) and should work whether or not items are viewed individually or in groups. Unfortunately, as we shall show in Section 2.7.4, state-of-the-art object recognition models suffer significant performance degradation under such real-world background artifacts.

2. **Energy Consumption:** Pervasive devices often have limited computation power and limited energy resources (e.g., smartwatches, smartphones, etc.). Given the energy consumption requirements for vision based applications, a pervasive device application might not be able to sustain for long. Though recent works have tried to optimize energy consumption in various ways, the energy consumption on pervasive IoT devices remains a formidable bottleneck for many reasons. First, in many instances, field deployments of such devices preclude the availability of a grid-powered energy source, and limitations on device form factors can limit the size and capacity of battery power. Second, as tens of thousands of such devices get pervasively deployed, it will be operationally impractical to periodically retrieve and replace batteries on such devices. Finally, battery-based energy sources present an increasing challenge to the goal of *sustainability*, as the annual production and disposal of billions of such batteries can lead to considerable pollution. Not surprisingly, there is an increased interest in developing ultra-low power sensing techniques (e.g., Glimpse [84]) that can eventually lead to the desired goal of battery-less sensor device operation [55, 53, 54]. This is a particularly difficult challenge for vision-based sensing, as vision sensing pipelines are among the most energy-intensive sensing modalities on pervasive devices [71, 72].

3. **High Computation Latency:** Recent increases in perceptual accuracy for vision tasks have almost always been accomplished by sharp increases in the complexity of DNN models. For example, while the initial YoLo v1 model had 26 layers and a benchmark accuracy of 63.4% [92], the state-of-the-art YoLo v3 model has a much deeper DNN model (consisting of 106 layers) even though it achieves a significant 15.9% improvement [63] in benchmark accuracy [94]. Sensing-based applications now routinely include computationally-complex DNN based processing of video and audio streams: for example, performing object recognition on mobile video [84]

or emotion/speaker recognition on mobile audio [67]. However, such heavy-weight computing is usually offloaded to a remote cloud platform. In spite of recent advances [84, 58], real-time execution of high-fidelity DNNs is still not possible locally on pervasive devices. To ensure their adoption in many real-world pervasive applications, it is necessary to perform such computation quickly (in near real-time) on resource-limited embedded platforms (such as an Nvidia Jetson Nano device), as opposed to execution on clusters of GPU-equipped server machines hosted in data centers.

4. **Object Identification Accuracy under Viewing Impairments:** Sensors in IoT devices are often embedded in objects of daily use—such as kitchen appliances, vehicles, on-body/wearable devices and even children's toys [42, 5]. Due to a variety of commonplace usage artifacts, it is possible that objects of interest may not be properly visible (and sometimes completely occluded for significant periods of time) to such embedded vision sensors. In general, the accuracy of DNN based vision models is seen to degrade fairly significantly when confronted with problems such as object occlusion, partial capture of objects and insufficient/uneven ambient lighting.

5. **Security & Privacy Concerns:** The use of vision-based approaches inevitably lead to privacy concerns, especially when such vision sensors are embedded in IoT devices deployed or used in relatively private spaces (such as within one's home). As an example, visual-microphone[39] uses high-speed video of objects kept nearby a sound source to decode the sound being emitted by the source. A variety of techniques have recently been proposed to tackle some of these concerns, either via machine-to-machine communication of privacy preferences (e.g., iPic [22], intelligent control of external objects such as smart lighting (e.g., LiShield [114]), or by using DNN-based models to obscure objects of interest without affecting end-application objectives (such as memorability [102]). In general, achieving the proper balance

between privacy and usability remains an open and challenging problem. Besides privacy, the use of vision-based sensing on pervasive devices also lead to considerable security challenges (e.g., [25], especially because such devices often lack the systems-level support (e.g., trusted hardware) commonly available on higher-end computing platforms. Such challenges are exacerbated if the applications require *cooperation* or sharing of video streams between multiple cameras.

While the list above is not intended to be comprehensive, it captures several of the key and interesting problems related to visual sensing-based IoT applications that emerged during my research and analysis. Out of this list of problems, my dissertation work tackles problems 1 & 2, and does not explicitly attempt to address problems 3, 4 & 5.

1.1 Motivating Scenarios

I now utilize the following scenarios to motivate the specific research challenges and application constraints that I propose to address in my dissertation.

1.1.1 Scenario 1: IoT based Food Item Sensing by a "Smart Fridge"

Knowing the contents inside a refrigerator can often be very useful. Alice is a person leading a busy life with most of her daily time invested in work and family-related chores. After work, she often visits a supermarket near her house to buy necessary food items. However, she is often unable to precisely recall what items are already present in her refrigerator, thus making it difficult for her to decide what to buy. She doesn't want to buy unnecessary items, especially because many redundant items stay untouched (and eventually expire) in her fridge for long time. She wishes to have a technology-based solution which can automatically & unobtrusively track all

the food items currently in her fridge, as well as compute the residual quantity of food remaining in food containers. Such residual quantity estimation can help her be aware of items that need to be replenished. Though state of the art technologies have demonstrated prototypes of a smart fridge based on various sensing modalities (such as RFID / barcode scanners [85, 82], embedded camera sensors [8] and even smartphones [97]) , none of the existing systems work effectively without explicit human action. While RFID/barcode scanners & smartphone based solutions require the user to explicitly scan the items, direct applications of camera-based sensing approaches result in poor recognition accuracy, primarily because the captured images often contain the food item as a small, often partially-visible part within a complex background. Moreover, pure visual approaches are unsuitable for estimating the residual content inside food containers, when the containers are opaque or only semi-transparent.

1.1.2 Scenario 2: Energy Efficient Collaborative Multi-Camera Sensing

Bob is a computer vision entrepreneur offering camera based monitoring systems to various entities like universities, offices, homes and so on. With the increasing proliferation of network connected cameras & advent of smart cities, he wishes to expand his systems to large-scale, e.g., city wide deployment. However, the high energy consumption of each individual visual sensing platform presents a significant technical challenge, as it is infeasible to perform very frequent battery replacement on tens of thousands of spatially dispersed nodes. Bob thus very much desires solutions that continue to offer high levels of object detection, but significantly reduce the power consumption of individual camera sensors.

From his deployment experience, Bob observes that such networked camera deployments usually have significant spatial redundancy, i.e., a particular geographical area is often simultaneously observed by multiple cameras. Bob wonders if there

might be ways to support *collaborative operation* across such partially-overlapping cameras to improve their power efficiency. State-of-the-art techniques (e.g., [80, 89] for multi-camera collaborative operation (for tracking & surveillance) propose novel DNN pipelines for improved object detection, but do not explicitly tackle the energy overhead challenge. In particular, such approaches do not actually *modify the parameters of the camera sensor*, but instead modify the subsequent image processing pipeline. To help reduce the energy overheads of multi-camera operation, Bob would desire a solution that can take advantage of these multiple views to reduce the overall camera energy overhead, possibly by reducing either the camera resolution or the sampling (frame) rate. To develop and deploy such a solution, we need mechanisms that (a) autonomously compute the spatially overlapping areas between multiple cameras, (b) determine the minimum acceptable image resolution (either for the entire image, or for selected parts of the image frame) and (c) execute such *mixed-resolution* sensing, followed by appropriate storage and networked data transmission, on individual cameras.

1.2 Thesis Statement

Previous sections highlight the importance of vision sensing in IoT ecosystem and discuss various challenges in deployments of vision sensing applications on pervasive devices. Based on these observations, I propose to demonstrate the following thesis in my dissertation:

I demonstrate that it is possible to improve the applications of AI-based vision sensing for IoT applications by employing multiple techniques, including :

(a) using additional sensing modalities to better isolate objects that need to be visually sensed;

(b) using collaboration among multiple spatially-overlapping visual sensors to significantly reduce the overall energy consumption of sensing-based applications while maintaining object detection accuracy;

This dissertation intends to establish these points in following manner:

- First, to demonstrate the benefits of using additional inexpensive sensing for better isolation of object of interest in an image, I build a prototype of a consumer grade smart refrigerator system (called *SmrtFridge*) which uses an Infrared (IR) camera sensor along with normal visible light camera to isolate cold objects from its surrounding. This unobtrusive isolation is triggered every time some item is being added or removed from the refrigerator and it is capable of object isolation under random object movements and varying levels of object occlusions as well. Apart from object isolation, the usage of an IR camera also enables coarse-grained, residual quantity estimation of contents inside a food container (transparent or opaque).

- Second, to demonstrate the use of collaboration among vision sensors to achieve a significant improvement in energy-vs-accuracy trade-offs, I create *CollabCam* - a multi-camera collaboration system for energy-efficient sensing. The core functionality of *CollabCam* includes an autonomous approach to accurately estimate the spatially overlapping regions between multiple field-deployed cameras and then use such region estimates to intelligently reduce the imaging resolution of individual cameras. Each collaborating camera captures a mixed-resolution image, whereby portions of the image that

overlap with the sensing field of other cameras, are captured at progressively lower resolution while non-shared regions are captured at the default (highest possible) resolution. I show that the subsequent use of customized collaborative DNN models permit such vision sensors to maintain object detection accuracy at a level comparable to a baseline of non-collaborative operation, while reducing the total energy consumption in image capture, storage and network transfer by $\sim$50-93%.

Chapter 2

Food Item Sensing System

In this chapter I describe the *SmrtFridge* system. *SmrtFridge* is an IoT-driven sensing system that demonstrates the capability of automatic & unobtrusive sensing of food items. *SmrtFridge* uses sensors embedded inside a regular refrigerator for (a) identifying the label of a food item, and (b) estimating the residual quantity of food inside a container. It follows an interaction-driven model, i.e., it activates its sensing pipeline only when a user actively interacts with the fridge. In particular, it attempts to identify the food item whenever it is being either added to, or removed from, the fridge. It also tries to estimate the residual quantity of food inside a container whenever a container is being inserted into the fridge. A lab prototype, based on a real refrigerator, was created and the system was tested using 20 different participants.

2.1 *SmrtFridge* Overview

The notion of a *smart fridge*, which uses embedded sensors to track the usage and quantity of stored food items, is a staple part of the vision of "Connected Devices" or **IoT** (Internet of Things) [60]. An Internet-connected fridge that can automatically track the *identity* and *quantity* of items placed inside it (with this information subsequently being exposed via Web APIs) can enable several useful applications– such as allowing a consumer to ascertain commonly-used items that need to be

replenished (when their quantity is low) or purchased (if they are no longer present in the fridge) while visiting the supermarket.

SmrtFridge uses a small number of commodity sensors to provide two novel features needed in an eventual smart fridge:

- *Interaction-Driven Capture of Food Items:* Using a combination of infra-red (IR) and optical camera sensors, it is able to automatically visually *isolate* & *extract* the specific brand/type of food item container that a user either places inside or removes from a fridge, without requiring any special per-object tags (e.g., RFID tags). For each user-object interaction, *SmrtFridge* captures the whole interaction and then it first identifies & extracts the relevant item image(s) from the captured interaction. By then feeding such extracted item images to "standard", state-of-the-art DNN-based object recognition pipelines, *SmrtFridge* can then identify the food item objects with high accuracy. *SmrtFridge's interaction-driven* paradigm, where the sensing pipeline is activated only during user-object interactions, is notable as it both (i) reduces the likelihood of visual occlusion (compared to prior approaches that focus on recognizing *stationary* items inside the fridge) by using multiple images and (ii) improves recognition accuracy by supplying the DNNs with cropped, *foreground* images of food items, using just a single camera[1].

- *Track Residual Fractional Quantity of Individual Containers:* After capturing the user-object interaction, *SmrtFridge* uses the IR images, captured by the appropriately embedded IR sensor during the interaction, to determine the approximate residual amount of content inside a food item container (either transparent or *opaque*). Estimated quantity is expressed as a fraction of the total container capacity, for example, 30%, 60% and so on. Quantity estimation is automatically triggered every time a container is *re-inserted* in the fridge.

[1]This is contrast to technologies, such as Amazon Go™, which reportedly use multiple store-mounted cameras & special product tags to identify individual items

SmrtFridge uses two novel capabilities, namely natural interaction-driven image capture and residual quantity estimation, to perform sensing of *individual food item containers* (e.g., the amount of milk remaining in a milk carton) and requires no overt user action or additional object-level tagging. This makes *SmrtFridge* distinct from prior smart fridge technologies (surveyed in Related Work), which either assume some form of explicit user interaction or object tagging, or rely purely on analysis of static RGB images of content *inside* a fridge.

2.1.1 Key Challenges

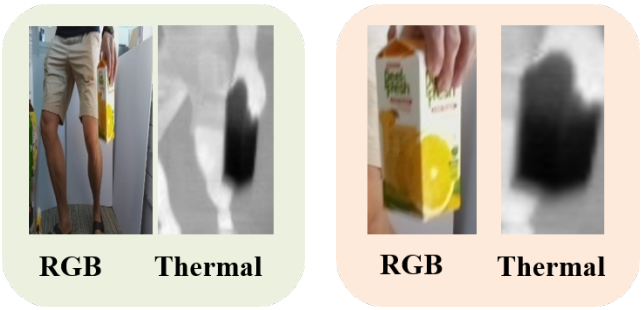
To build a smart fridge that uses such natural item-level interactions to identify food container items and their residual content, we must address several challenges:

- *How to Extract a Food Item:* Individual users interact with individual food item containers using a variety of different, transient gestures–e.g., an individual might hold a single milk container in the middle while removing it from the fridge, while another grasp the same object with two hands. A key challenge is to look at the sequence of image frames captured by a camera sensor, during such natural interaction episodes, and isolate/extract food item images, with an accuracy that is suitable for state-of-the-art image classifiers.

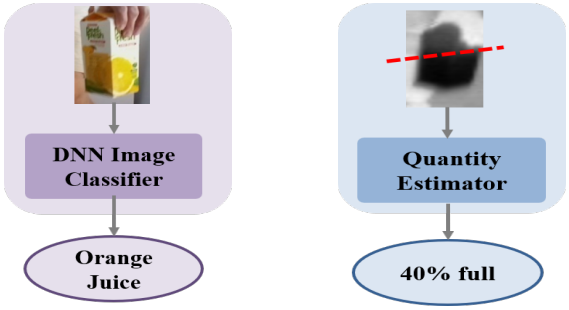
- *How to Overcome Varying Levels of Object Occlusion:* As a user extricates or inserts an individual food item into the fridge, it is quite likely that the camera's field of view will be obscured by the user's body parts (e.g., the user's hand or fingers) at different points of the motion trajectory, thereby occluding the food container object. An important research question thus is: How do we utilize a single static camera to robustly *recognize* an individual food item object and *reconstruct* its shape, even under occasional partial visibility?

- *How to Estimate the Content of a Container:* To generate proactive alerts for specific conditions (e.g., when a milk carton is becoming almost empty),

we need a system that is capable of recognizing and tracking changes in the content of such containers. Past approaches (e.g., [35]) have suggested the use of weight or visual sensors, but these cannot handle opaque containers or track multiple items. Accordingly, we tackle the question: How do we identify the changes in the occupancy level of individual, potentially non-transparent, containers, across such natural user interactions?



(1) Interaction Capture (2) Item Extraction



(3) Object Recognition (4) Quantity Estimation

Figure 2.1: High-Level Steps in *SmrtFridge*

2.1.2 Functional Overview

Figure 2.1 shows the high-level idea of *SmrtFridge*. Once the fridge door is opened, (1) the fridge-mounted cameras start capturing images of the user-object interaction. This includes multiple RGB & IR images spanning the whole user-object interaction. (2) Subsequently, a combination of thermal & optical flow-based approaches is used to identify & extract the image segment corresponding to a food item. This

step extracts the relevant parts of the images and removes the unnecessary parts, e.g., human body, background objects etc., from the captured images. (3) The extracted sub-images (only RGB) are then fed to a Deep Neural Network (DNN) based classifier to visually identify the food item label. The DNN outputs a most likely label, e.g., orange juice, for each segmented image. The final item label (for one full user-item interaction) is estimated based on a weighted fusion of multiple labels returned by the DNN. (4) Finally, when the user subsequently *re-inserts* the item back in the fridge, a quantity-estimation pipeline operates over IR image(s) of the extracted food item to quantify the fraction of the container that is empty. The estimated quantity is expressed as a percentage of the full capacity, e.g., 40%, 60% etc.

2.2 Design Goals

We believe that for a system like *SmrtFridge* to be usable in a day to day life, it should satisfy certain design requirements as mentioned below.

- *Accurate & Precise Identification of Item Labels: SmrtFridge* must be able to accurately identify and label the individual food items with which a user interacts. Also, the label given to each item should be precise and not generic. For example, generic item-agnostic alerts would be of the form 'item was extracted from the fridge at time t' and are useful only for tracking fridge usage & doesn't provide any actionable input to the user. In contrast, an alert of the form 'Juice Product X, with approx. 40% content remaining, has been in your fridge for the past two weeks' provides a user targeted, actionable feedback. These kind of alerts would help a user decide what to do with that specific juice box. In order to provide these kind of alerts, the *SmrtFridge* needs to be able to accurately identify food labels as well as estimate residual quantity of the container products.

- *No Additional Human Effort:* A fridge is a commonly used household item, with which an individual or household interacts multiple times a day. To avoid making such interactions cumbersome, it is important that *SmrtFridge* operates unobtrusively, i.e., it should not impose any requirements for additional explicit user action, beyond what she presently does with a conventional fridge. This implies that *SmrtFridge* cannot employ approaches where fridge items are augmented, for example, by attaching RFID tags to each item, utilize additional infrastructure such as Barcode scanners [97] or involve manually entered product logging [76] to obtain additional insights. Moreover, *SmrtFridge's* image-based item recognition pipelines must work in-the-wild, i.e., with images of items that are not necessarily centred or placed vertically.

- *Need to Estimate Residual Amount in a Container:* Past studies [64] have shown that users who are aware of the amount of unconsumed food items in their fridge make less wasteful consumption decisions. To support such insights, *SmrtFridge* must be able to estimate the fraction of remaining content inside specific containers. Now, a fridge may have different types of containers, i.e., containers of different materials, shapes, sizes etc. So, *SmrtFridge* should be able to estimate quantity for a variety of containers. From a practical perspective, it is imperative to perform such content estimation when the user is inserting an item back into the fridge (as the user would have typically consumed some fraction of the existing content) so that the user can subsequently track (without inspecting the refrigerator) the residual quantity of food.

2.3 System Architecture

SmrtFridge's key novelties (compared to prior work) are in developing processing pipelines to (a) *extract* a food item's sub-image (a bounding box) from individual

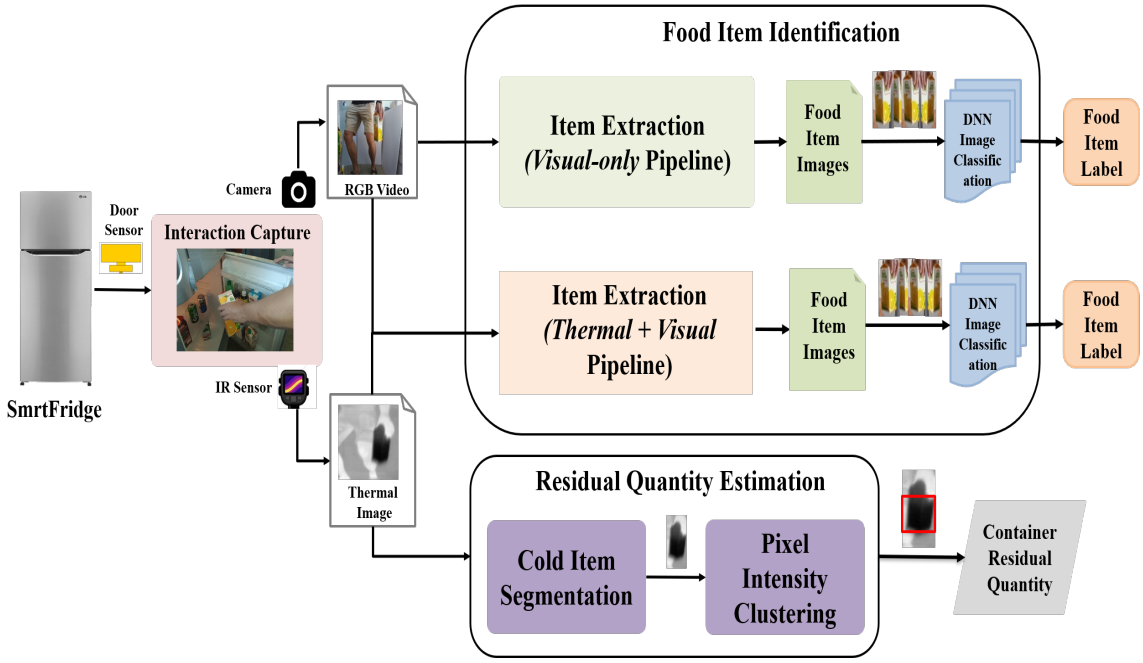
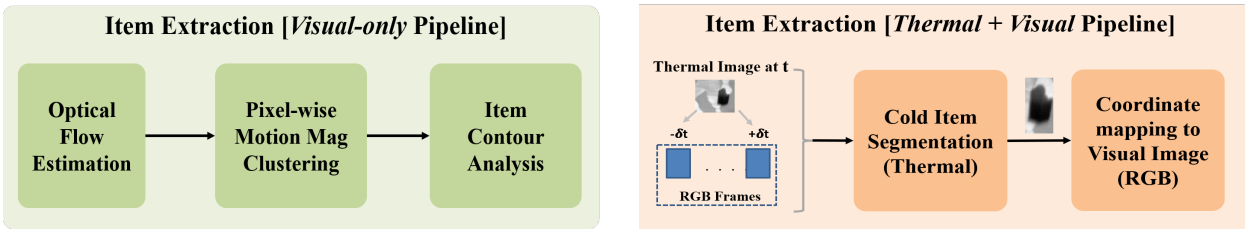


Figure 2.2: Overview of *SmrtFridge's* Workflow.



(a) Visual-Only Image Extraction (b) IR-driven Image Extraction

Figure 2.3: Item Extraction Pipelines

RGB image frames (so that it can then be recognized using state-of-the-art DNNs), and (b) *estimate* the residual food quantity in such food containers.

Figure 2.2 shows detailed architecture of *SmrtFridge* system. As illustrated in Figure 2.2, *SmrtFridge's* sensing substrate includes: (a) an RGB camera that visually captures the item-level interactions that an individual performs with the fridge; (b) an infrared (IR) camera that is used to both aid in food item extraction and quantity estimation; and (c) a magnetic reed switch, attached to the door, which detects the opening and closing of the fridge door (and thus triggers the sensing pipeline).

The *SmrtFridge* workflow consists of the following steps:

1. *Episode Segmentation:* The door contact sensor (magnetic reed switch) helps

to identify the *start* and *end* of a single interaction *episode*. An episode may involve the user inserting/removing one or more (and possibly even none) items in/from the fridge. So, the door contact sensor acts as a trigger to the IR and RGB camera pipelines. These cameras start capturing images whenever the door is open until the user closes the door subsequently. At the end of an interaction, the *SmrtFridge* would have a sequence of images (both RGB and IR) covering the whole interaction. This step makes *SmrtFridge* an interaction-driven system, i.e., a system that only captures data as and when there is any user interaction happening.

2. *Item Image Extraction:* This process concurrently executes two different pipelines. The first pipeline, as shown in Fig 2.3(a), uses only the visual (RGB) camera data to first compute object motion vectors using the *sequence* of captured images. This is followed by clustering and thresholding (motion magnitude) of such vectors to extract the image of the food item. The second pipeline, as shown in Fig 2.3(b), uses the thermal (IR) camera driven approach. It first obtains the relevant coordinates of objects that are significantly colder than the ambient thermal values. Once colder object coordinates are estimated, then next step is to map these coordinates to the RGB image coordinates. These re-mapped RGB camera coordinates are then used to extract the pixels corresponding to the image of the food item.

3. *Image-based Food Item Recognition:* The extracted RGB images (or sequence of images), ideally corresponding to a food item, are then passed through a CNN-based recognizer. The CNN is pre-trained by an external entity (e.g., an image analytics company) with a, preferably large, corpus of representative images of various food items. For each image frame, the CNN then outputs the likely label (along with the confidence values). Because the item-specific user interaction (within an episode) lasts for several seconds, the extraction process retrieves a sequence of multiple (typically 30-40) images,

of which 5-10 contain the food item. This series of CNN output labels are then further fed through a classifier that uses the frequency of occurrence and associated confidence levels to output the food item label with the *highest likelihood, above a minimum threshold*.

4. *Residual Food Quantity Estimation:* In parallel to the above *Food Item Identification* process, IR images are also fed through a quantity estimation pipeline. This pipeline works on the principle of *differential heating of the container vs the food content inside them*. Because such a temperature differential is most likely absent when the user is taking out a currently refrigerated item, this pipeline is triggered only when the user is *inserting* items into the fridge. For a given IR image, this pipeline also extracts the image segment corresponding to the cold item (using the same thresholding scheme above). The extracted food item's IR image is fed through an unsupervised classifier that demarcates the container pixels into one or two spatial clusters (based on whether the container is completely or partially full). The partial area of the *colder cluster* (corresponding to the non-empty portion of the container), relative to the area of the overall container, is then used to estimate the residual food quantity (by volume percentage). The idea of using IR images for quantity estimation enables *SmrtFridge* to estimate residual quantity for various types of containers, e.g., transparent as well as *opaque* containers.

5. *Additional Workflow Steps:* Once we have determined the food item and its remaining quantity, *SmrtFridge* can appropriately update a repository of refrigerated food contents. Similar to prior work, such changes ('bottle of product A, 30% full' inserted) may be pushed to a Web server, which can be instrumented to generate relevant alerts (e.g., "send an SMS if a container with residual quantity $\leq$20% has been sitting in the fridge without any user interaction for more than a week"). Note that such a Web back-end has not been implemented in the present *SmrtFridge* prototype.

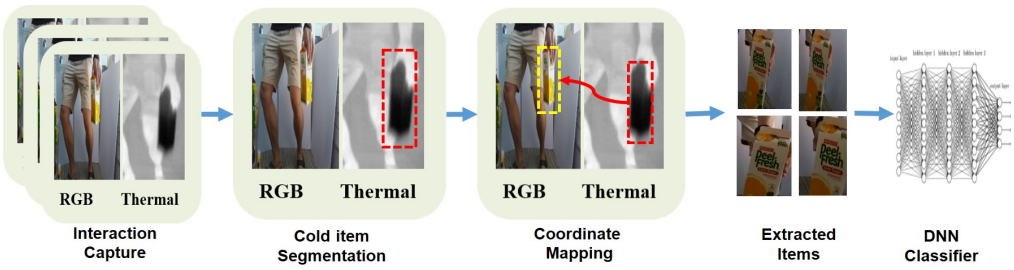


Figure 2.4: IR Driven Item Extraction Steps

This workflow is repeated for each user-item interaction therefore identifying food item labels and residual quantity for each item that the user is interacting with. The subsequent sections will describe *SmrtFridge's* both pipelines in detail.

2.4 Food Item Identification

This section provides detailed description of food item identification pipeline of *SmrtFridge* . The process for extracting food items includes two alternative pipelines (one purely using visual images vs another fusing IR and visual images), coupled with a CNN-based item classifier. Both these pipelines would assign a label to the food item the user is interacting with. While they can be triggered in parallel, I see visual only pipeline as a fallback pipeline when temperature difference necessary for effective activation of the IR-driven pipeline is absent.

2.4.1 IR-driven Image Extraction

Figure 2.4 represents IR driven item extraction pipeline. This pipeline utilizes the insight that a refrigerated item will typically be *much colder* than either the interacting human's body or the ambient temperature. Since the IR camera encodes the IR image regions by their temperature, the regions with lower temperature would look differently from regions with higher temperature. For example, the image region containing a cold item would appear relatively darker than the region containing hot objects. Based on this hypothesis, an IR camera should be able to easily isolate a

cold food item from other ambient objects, as the food item's pixels will be much darker than other ambient objects. Accordingly, *SmrtFridge* attempts to use a *pixel intensity-based segmentation* approach. In this approach, during an ongoing interaction episode, *SmrtFridge* extracts the cold item from the thermal image (a frame with timestamp t) by selecting all pixels below a threshold value Th_{temp}. Then, it computes the Cartesian coordinates of all the selected pixels, thus segmenting the cold item from the thermal image. It then estimates a bounding box (i.e., the smallest rectangular region that *contains* the entire contour area) to represent the segmented object. This bounding box represents the cold item location in IR image.

Once the item's bounding box in the IR camera's coordinates is identified, we need to find the corresponding location in RGB image (since the classifier requires RGB images only). For this, we utilize the fact that both the IR and RGB cameras *concurrently* and continuously record images/videos, albeit with different FoV (field of view), during an interaction episode. Because the two cameras are fixed, we transform the IR camera's coordinates into the RGB camera's frame of reference using an a-priori computed *transformation matrix*.

However, empirically (because our Raspberry Pi-based implementation does not support a real-time OS), we observe that the frames of the two cameras are not always synchronized and that the frame rate of the RGB camera is usually higher than that of the IR camera. Accordingly, we select all the RGB frames that have a timestamp $(t - \Delta, t + \Delta)$, where Δ represents the time offset: for each of these frames, we extract the sub-image corresponding to the (transformed) RGB bounding box coordinates. Separately, the optical flow approach (Section 2.4.2), applied to selected RGB frames ($t \in \{t - \Delta, t + \Delta\}$), provides another set of candidate images. Each of these "potential food item" images (two per frame) is then sent to the downstream item recognition DNN classifier. This process is repeated for each thermal image captured in each user-item interaction.

2.4.2 Visual-Only Image Extraction

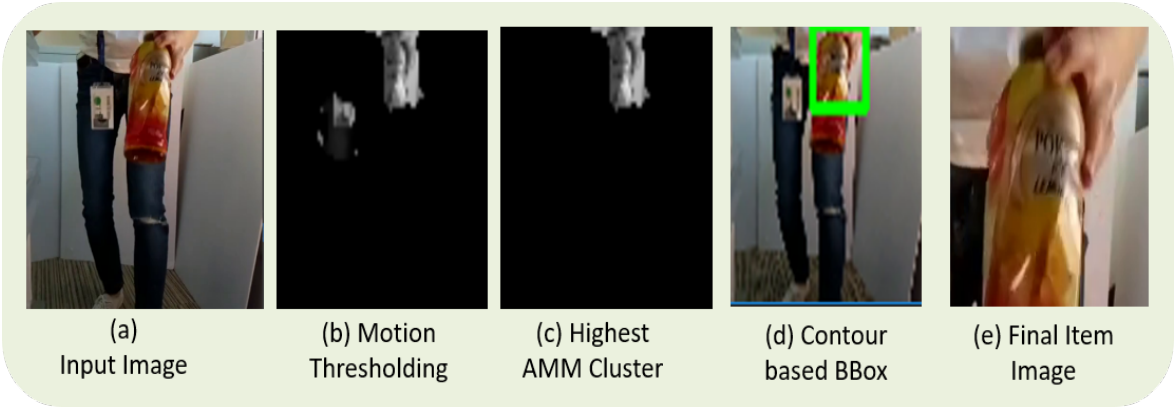


Figure 2.5: Visual Item Extraction Steps

The IR-driven approach relies on food item temperature being different than the ambient temperature. But sometimes, it might not be the case, for example, imagine a situation when a user brought juice box from supermarket and kept it outside for certain amount of time before putting it inside the refrigerator. Meanwhile, the food item might have gained heat and its temperature might match room temperature. Now, while putting the item inside, the IR camera would not be able to isolate the item from ambient objects, resulting in a failure of the baseline IR-driven approach. To provide an alternative means of food item tracking under this broader set of conditions, we utilize the fact that a user's interaction with a food item (either removing or inserting into the fridge) involves a directional motion either away from or towards a fridge-mounted camera. The approach, illustrated in Figure 2.5 first applies the principal of *optical flow* to identify the image segments that are moving (across consecutive frames), thereby eliminating the parts of the image that correspond to a static background. Such optical flow estimation identifies motion vectors (direction and displacement magnitude) for each pixel in an image.

We then identify the parts of an image with *significant movement*, i.e., with motion magnitude higher than a minimum threshold Th_M. The resulting pixels (Figure 2.5(b)) are likely to contain the food item, as well as other moving objects captured in the camera's field of view (FoV), such as the user's limbs, the moving

fridge door and even background movement (e.g., an animal moving in the background). To then isolate the food item from additional movements, we first employ spatial clustering. More precisely, we create a feature vector where each pixel's feature consists of its coordinates, as well the magnitude and direction of its motion vector, i.e., {x,y, motion-mag, motion-dir}. We employ the K-Means clustering technique to cluster the pixels into distinct, spatially disjoint, *motion clusters*, and then pick the cluster with the highest average motion magnitude (AMM) value (Figure 2.5(c)). This is based on our intuition that the food item of interest is usually *the moving object closest to the camera*, and thus extremely likely to have the largest displacement magnitude from the camera's perspective.

The resulting cluster (Figure 2.5(c)) consists of both the food item, as well as possibly additional background pixels. To better isolate the image segment corresponding to the food object, we then execute the *Canny* edge detection algorithm, followed by standard morphological operations (e.g., erosion and dilation) to help connect some of the disconnected edges. The resulting edges are then passed through a contour detection algorithm to obtain an outline of the food item, before fitting a bounding box (Figure 2.5(d)) over this contour to represent the image. As this bounding box image is from a scaled-down version of the initial RGB frame (the down-scaling was initially performed to speed up the computation), we finally scale-up the bounding box coordinates, using template matching, to select the high-resolution sub-image (Figure 2.5(e)) that represents the extracted food item. As before, each such 'food item' is sent to the downstream item recognition DNN classifier.

2.4.3 The Food Item Recognition Process

Once the food item is extracted using the pipelines described above, next step is to identify the food item and assign a label to it. Given the extracted item, we then use a well-known CNN-based deep learning classifier, the *ResNet v2 (152 layers)* [51]

to classify the food item. Note that (see Figure 2.2) that this classifier receives *multiple possible* food item images. Specifically, the IR-driven pipeline provides one coordinate-transformed image for each RGB frame with a timestamp within Δ of an IR frame, whereas the 'Pure Visual Extraction' approach provides an image for every RGB frame with a foreground cluster exceeding the motion threshold.

The item recognition process consists of the following steps:

1. Given K different classes of food items, we first train a multi-class CNN classifier that outputs $K + 1$ labels: each of the K food items + a *null* class (corresponding to a 'non-food' classification).

2. During the test phase, each interaction involves a sequence of say S image frames, provided by both the IR+MV and MV-only methods. Each frame was then individually passed through the classifier, generating a probability/confidence value for each of the K+1 labels. Let $p_i^k \ k = \{1, \ldots, K+1\}, \ i = \{1, \ldots, S\}$ represent the probability of the k^{th} class for the i^{th} frame.

3. For each such frame, if the highest likelihood class is $K + 1$ (the non-food or background class), then we discard the corresponding frame. From our empirical studies, we observed that such frame discard occurred $\sim$50% of the time.

4. For the remaining L frames, we compute the cumulative likelihood of each of the K food item classes using the (a) **FREQ-CONF** method: for each class, we compute the frequency of identification, as well as the sum of confidence values (across L frames) within the episode, and then select the most-frequent class that has the highest frequency probability/likelihood across the L frames. (b) **LOG:** In this case (corresponding to an assumption of maximizing the probability across independent frames), we compute the sum of the confidence log-values, i.e., for the k^{th} class, we compute $\sum_{j \in L} log(p_j^k)$. In practice, the two methods were found to be empirically equivalent; hence,

all results and analysis presented here are based on the simpler FREQ-CONF estimation technique.

5. Finally, we select the food item label that has the highest cumulative likelihood value across all the frames. An alternative strategy of just using the classification output from a single 'randomly-selected' frame may reduce the energy consumption but has much lower accuracy (Section 2.7.4).

2.5 Quantity Estimation

Besides identifying the food item removed or replaced, *SmrtFridge* also quantifies (at a coarse granularity) the quantity of residual food inside the identified container. Quantifying such content is vital for several possible applications, e.g., informing users if the quantity of juice in a container falls below a minimum threshold (e.g., 20%), or if a close-to-full container has been lying inside the fridge for a very long duration. For our exposition, we estimate quantity as a fraction of the container volume, e.g., if a 1000 ml juice container presently has $(\frac{3}{5})^{th}$ (600 ml) of juice remaining, the quantity should be ideally estimated to be 60%.

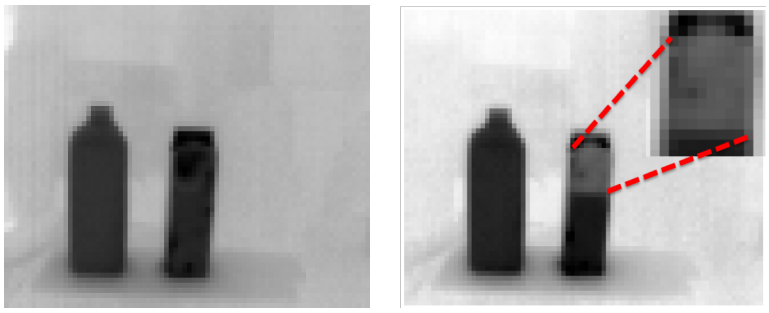
2.5.1 The IR-based Approach

SmrtFridge uses a non-intrusive quantity estimation technique that is both robust to different ambient lighting conditions and the *opaqueness* of the food container. In this technique, an inexpensive relatively low-resolution IR camera is used to record and extract the food item's *thermal profile*, when the user is re-inserting an item back inside the fridge. The technique is motivated by a fundamental observation on *differential specific heat properties* of a container and the item that it contains. In particular, whenever a currently refrigerated food item is removed and placed outside, its temperature will start to increase as it absorbs ambient heat (assuming room temperature is higher than the item temperature). For a full container, all parts

Food Item	—	Container Material	
Juice	3.4	Plastic	0.4
Milk	3.93	Glass	0.2
Water	4.18	Paper	0.33
Yogurt	3.52	Air	0.718 (C_v)

Table 2.1: Specific Heat of Substances (KJ/kg/ C)

of the container (containing the solid or liquid food item) will gain heat at a similar rate, whereas in a partially filled container, there will be a difference between the rates at which the empty & filled portions of a container warm. Table 2.1 lists the specific heat capacity of some of the common liquid/solid food items and typical container material. In general, we see that the food items have significantly higher specific heat than typical container material: intuitively, the part of the container in direct contact with the food item (liquid or solid) will share its acquired heat conductivity with the item, and thus experience a slower temperature increase than the empty portion (which will heat faster). Moreover, the *larger the specific heat of the food item*, the higher the difference between itself and the container and thus the larger the expected differential between the thermal intensity of the empty vs filled parts of the container.



Cold items at t=0 Cold item at t=20s

Figure 2.6: Thermal Intensity Differential after 20 seconds

Our hypothesis is that the thermal camera can utilize this temperature difference to estimate the remaining quantity inside the container. Such differentiation will, of course, depend on the thermal resolution of the IR camera; we found that com-

Figure 2.7: RGB & IR cameras deployed inside the fridge

modity cameras (e.g., the Raspberry PI compatible Bricklet camera[2]) typically have resolution of 0.1°C or lower. As an illustration of this hypothesis, Figure 2.6 shows two thermal images, each containing two cold containers (the left one being full and the right one partially filled). Left side image is a thermal image when both the containers were just taken out from the refrigerator ($t = 0$) whereas right side image shows the thermal image of the same containers after they were kept outside for $t = 20$ seconds. It can be seen that the thermal image of the partially filled container shows two regions of different pixel intensities, with the empty region having higher temperature values (less dark pixels) and the 'filled' region having lower temperature values (darker pixels). *SmrtFridge* leverages this difference in pixel intensities to estimate the size of the empty portion of the container and thereby derive the quantity of the food item inside the container.

2.5.2 Processing Pipeline

Figure 2.7 shows the interior of the test refrigerator. The thermal camera was installed on the left side of the refrigerator facing outwards.The red rectangular box shows the exact location of both the cameras (RGB & IR). Figure 2.8 shows two representative thermal images captured when two different food items are being

[2] `https://www.tinkerforge.com/en/shop/thermal-imaging-bricklet.html`

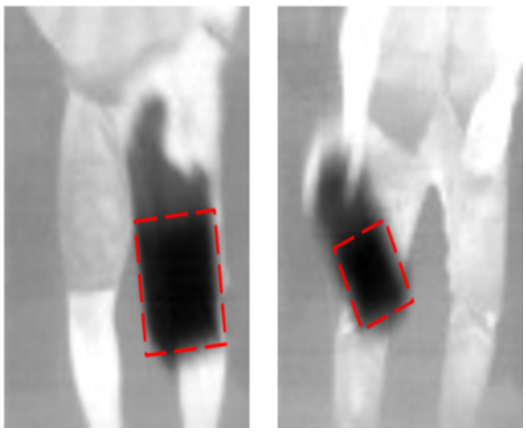


Figure 2.8: Sample thermal images taken by thermal camera

Figure 2.9: Quantity Estimation Processing Pipeline

kept inside the refrigerator. After the thermal images are captured, each image is passed through the following processing pipeline (illustrated in Figure 2.9):

- *Partial Capture Check*: Due to the continuous capture of images during the user-item interaction episode, the thermal camera will generate multiple images of the food container. Because of the underlying motion dynamics, some images will capture the item only partially, while others will obtain a larger, clearer view. Since the estimated quantity is represented as a fraction of the whole container capacity, estimating quantity on partially captured images might give us wrong estimate. So, it becomes important to ignore the partially captured images. To eliminate *partial captured* images (which can be ignored for estimating quantity), *SmrtFridge* checks to see if the container's contour intersects with the boundary of the captured image. If so, the container has likely been captured only partially; we thus discard the image.

- *Cold Item Segmentation & Noise Removal*: After filtering out partially captured images, next step is to isolate the cold item from the full frame. Given

the thermal image, we follow the pixel intensity based segmentation steps outlined in Section 2.4.1 to extract the image segment corresponding to the food item container. While the image segmented in this manner will contain all the pixels corresponding to the cold item container, it might also contain some extraneous pixels. This is due to heat leakage around the cold item container, whereby pixels that are *near* the cold container have an intermediate temperature value that is lower than the ambient temperature. To remove these neighboring intermediate pixels, we use clustering and contour detection. First, we cluster all the segmented cold points into two clusters, one containing the intermediate neighborhood pixels (and empty part of container) and another containing the "filled-part" of the container itself. Second, we find contours from both the clusters, labeling the contour with the larger perimeter value as the *outer contour* (this contains all the neighborhood intermediate cells) and the other as the *inner contour*. To selectively discard only the neighborhood pixels, we first obtain the top-most point (highest y coordinate) of the inner contour. Because the empty part of the container is always *above* the filled portion (due to gravity), we then discard those pixels from the outer contour that lie below this top-most point (i.e., have smaller y coordinates) and combine the remaining pixels (which we anticipate to correspond to the empty portion of the container) with the pixels of the inner contour to obtain the *container's contour*.

- *Occluded Pixels Removal:* Depending on the user-item interaction pattern, some part of the container can be occluded by the user's hand. This occlusion is evident (as high brightness pixels) in the thermal image, for example, the first image in Figure 2.8 shows the upper part of the container occluded by user's hand. The occlusion can cause an under-estimation of the container volume and consequently result in inaccurate quantity estimation. To overcome this occlusion, *SmrtFridge* uses an interpolation strategy, where it first

extends the detected contour to a more regular (often rectangular) shape. The occluded pixels within this extended contour are then given an *estimated* thermal value, computed as the mean/median of the neighboring non-occluded pixels.

- *Clustering*: Finally, *SmrtFridge* applies clustering on the pixel values of the extended container contour obtained from the previous step. Intuitively, if the item container is full, then there should only be a single cluster, whereas a partially filled item should be separable into two clusters. To determine whether there should be only 1 cluster or 2, *SmrtFridge* uses the Silhouette Coefficient method [98]. Silhouette coefficient quantifies the distance between the resulting clusters. If the number of preferred clusters is 2, we compute the fractional quantity of the food item by dividing the pixel count of the "food item" (lower temperature) cluster by the total pixels in both the filled and empty clusters. If number of preferred clusters is 1 (as per silhouette coefficient), then the container is most likely full (alternately, the container might be completely empty; however, in such a case, a user is unlikely to return the item to the fridge).

- *Averaging:* Since *SmrtFridge* captures the whole user-item interaction as a sequence of RGB + IR images, we might have multiple IR images for quantity estimation as well. In that case, given multiple valid images for a given interaction episode, the final quantity estimate is obtained by averaging the fractional estimates of each image using the above described pipeline.

2.6 *SmrtFridge* Prototype

To empirically demonstrate feasibility of the above described techniques for IR+visual based food item identification and IR-based quantity estimation, we have built and tested a *SmrtFridge* prototype. The prototype (costing less than USD 300)

was built using a commodity fridge (Toshiba GR-R20STE, double door with 184L capacity), with the following sensors controlled by a Raspberry Pi 3 model B:

(a) RGB Camera (b) IR Camera (c) Door Sensor

Figure 2.10: Sensors used in *SmrtFridge* prototype

- *Visible Light Camera sensor*: Raspberry Pi camera module V2 as shown in Figure 2.10(a).
- *IR/Thermal Camera sensor:* Thermal imaging bricklet as shown in Figure 2.10(b)[3]. One important property is that the IR sensor has a relatively low resolution (80 by 60 pixels). While higher-resolution IR sensors might offer better accuracy, they were significantly more expensive.
- *Door Contact sensor*: Normally open magnetic reed switch as shown in Figure 2.10(c).

2.6.1 Placement of Sensors

One of the important empirically-determined choices relates to the placement of the sensors. In particular, the IR and RGB camera sensors need to be appropriately positioned to support multiple concurrent objectives: (a) *maximize gesture coverage*, i.e., support the video based capture of user-item interactions performed in a variety of ways, across different shelves of the fridge; (b) *minimize occlusion*,

[3] `https://www.tinkerforge.com/en/shop/thermal-imaging-bricklet.html`

i.e., ensure that the food item is maximally visible within individual frames (to aid proper computation of the residual quantity); (c) *maximize visible frames*–slightly different from the above objectives, the goal here is to have the item be visible in the maximum number of possible frames (to maximize the chances of correct food item classification).

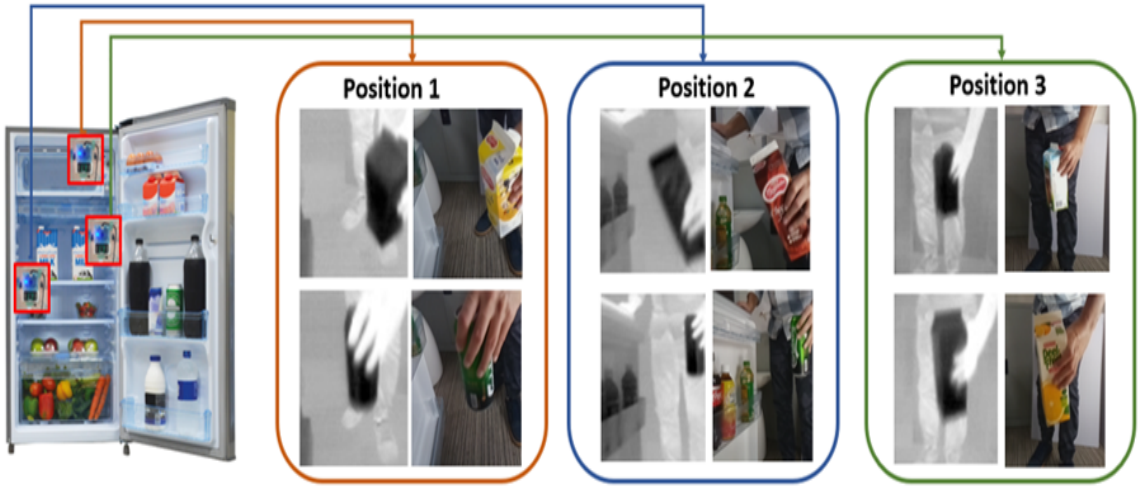


Table 2.2: Potential camera deployment positions. Position 3 is preferred due to {greater coverage, lower occlusion}.

We empirically experimented with various positions, of which the top three choices are illustrated in Figure 2.2. (The figure also shows sample images captured from each of these positions.) Note also that we explicitly chose positions where the sensors were an integrated part of the fridge frame/body–accordingly, we did not consider choices that involved placing the sensors externally. On analyzing sample video frames (obtained from our controlled studies) we observed the following characteristics:

- *Position 1*: Here the camera (IR+ RGB) sensors are installed on top of the refrigerator, thus providing a top view of the items while they are being added/removed from the fridge. Although this view is likely to capture most of the item interactions, it is often unable to capture the height of the containers properly (see Figure 2.2) especially when the containers are picked from the lower racks, leading to lower accuracy of quantity estimation.

- *Position 2*: Here the thermal and visible light camera sensors are deployed on the left side (closer to the door) of the refrigerator. In this case, the captured items often include items kept in the trays mounted on the fridge door. While such images can possibly be eliminated by optical flow techniques, the presence of such cold items is likely to increase the error of the thermal segmentation process.

- *Position 3*: This is the case when both thermal and visible light camera sensors are deployed on the right side (away from the door hinge) of the refrigerator. From our sample observations, we found that the vast majority of interactions (across a variety of 'removing' or 'inserting' gestural patterns) were visible with this placement, with the camera's field-of-view (FoV) primarily capturing the user-item interactions. Furthermore, occlusion of the food items was also very rare. Accordingly, we have used Position 3 as the preferred placement in our prototype.

We believe that, while these observational insights can benefit future fridge design, additional model-specific studies would be needed to determine optimal placement in other scenarios (e.g., single vs double door fridges). Also, the depending on the fridge size, multiple RGB+IR cameras might be required to cover the whole fridge area.

2.6.2 Object Recognition DNN

To identify the food item objects, we utilize the well-known *ResNet v2* Model (152 layers) with pre-trained ImageNet weights. The classifier is trained, using *TensorFlow* on an Intel Core i7-7700 CPU @ 3.60GHz with 64GB RAM & NVIDIA GeForce GTX 1080 Ti GPU, The classifier had 19(objects) + 1(background) class and 2,000 images per class. To generate the training set (in a commercial setting, such training data could be crowd-sourced directly from food manufacturers), we (a) used a camera to record videos of the food items under different conditions, such

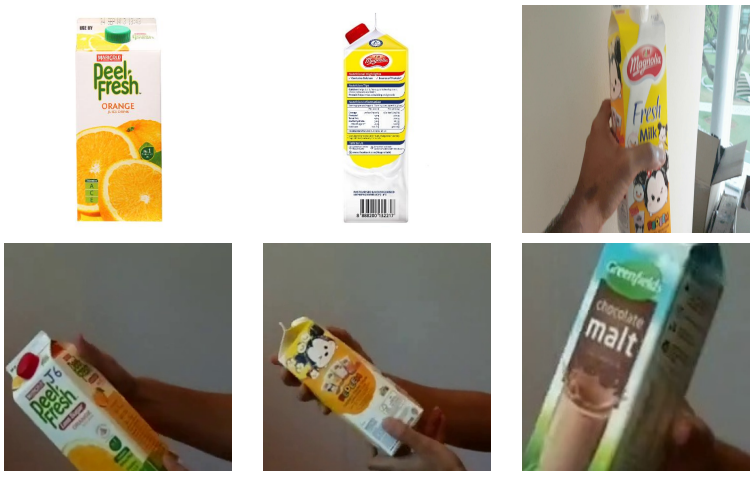


Figure 2.11: Sample images used in classifier training

as varying zoom levels, object rotation, background lighting and occlusion levels; and (b) downloaded corresponding Web images using Google's Custom Search engine. Figure 2.11 shows some of the sample images used in classifier training.

Also, for the 'null' (background) class, we shot videos of various indoor lab settings. From this dataset, we utilized 80% for training, 10% for validation and 10% testing, achieving a test accuracy of 97.6%. Our training dataset didn't include *in-fridge* videos of any item.

2.7 Performance Analysis

2.7.1 Controlled Study & Validation

We performed *preliminary controlled studies* using the *SmrtFridge* prototype (described in section 2.6) to understand the basic feasibility of our previously-described IR-based quantity estimation process. In particular, we experimented with a paper container that was filled to 60% of its capacity with 3 different liquids and initially placed inside the fridge. The container was then brought out of the fridge and placed outside for a variable duration, before being re-inserted into the fridge. The IR-based quantity estimation technique was then applied to the images captured during the user's interaction during this re-insertion phase. This analysis was

repeated at two separate deployments described below:

1. **Lab setup** : The fridge was kept inside a university laboratory with the ambient temperature regulated to 22-23°c.

2. **Home setup** : The fridge was placed inside an actual home with its ambient temperature varied from high ($\sim 25°c$) to low ($\sim 18°c$)

We studied two distinct questions:

- *How does estimation accuracy vary with different food items?* To address this question, we experimented with 3 distinct liquids {*juice, milk, water*} placed inside the container.

- *How long does a container need to be placed outside for the thermal differentiation to be discernible?* Intuitively, if this ambient exposure time is too short, the thermal difference would be too negligible to permit proper clustering; conversely, if the duration was too long, then both the empty and filled portions of the container would reach (or be close) to the ambient temperature and be indistinguishable. To address this question, each of the 3 liquids was placed outside the fridge for a duration T_a that varied between {0,5,15,30,60,90,150,200,450, 800,1100,1800} seconds.

Performance of Lab Setup: Figure 2.12 plots the estimation error (expressed as % of whole container capacity) for all 3 liquids, as a function of the ambient exposure duration T_a. We see that:

- The quantity estimation error is typically less than 15-20% for all liquids, indicating *our IR-based approach provides good* ***coarse-grained*** *quantity discrimination capability.*

- This error is relatively insensitive to the ambient duration (T_a), as long as this duration varies between 5 secs - 15 minutes. From our empirical observations across multiple households, we observed that vast majority of interactions

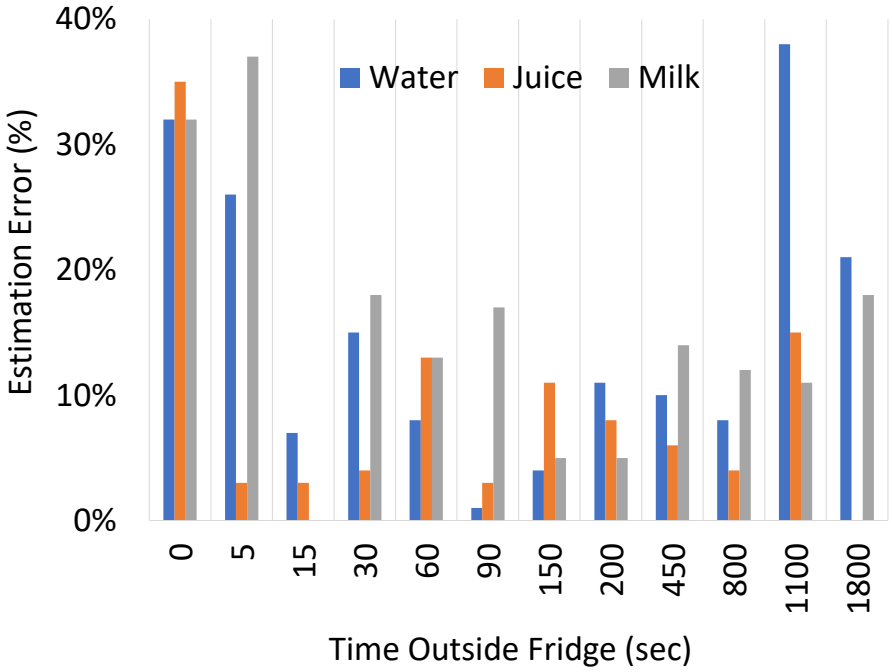


Figure 2.12: Lab Study: Estimation Error vs Time Spent Outside Fridge

with refrigerated items involved keeping the items outside for at least 5 secs, and no more than 15 minutes. Our results thus suggest that *our IR-based approach is applicable to a very wide variety of user interaction patterns, even though its accuracy would degrade if a container was left outside too briefly (<5 secs) or for too long (>20 mins).*

Performance of Home Setup: Figure 2.13 and figure 2.14 plots the estimation error for all 3 liquids, as a function of the ambient exposure duration T_a when ambient temperature is kept at (a) high temperature ($\sim 24 - 25°$) and (b) low temperature ($\sim 18°$) respectively.

We make following observations:

- For both the ambient temperatures, estimation error is high when item exposure time $< 5sec$.
- When value of $T_a > 5$ seconds and $\leq$15 minutes, the estimation error is approximately 15-20% for all 3 liquid types.
- Compared to high ambient temperature, the low ambient temperature scenario

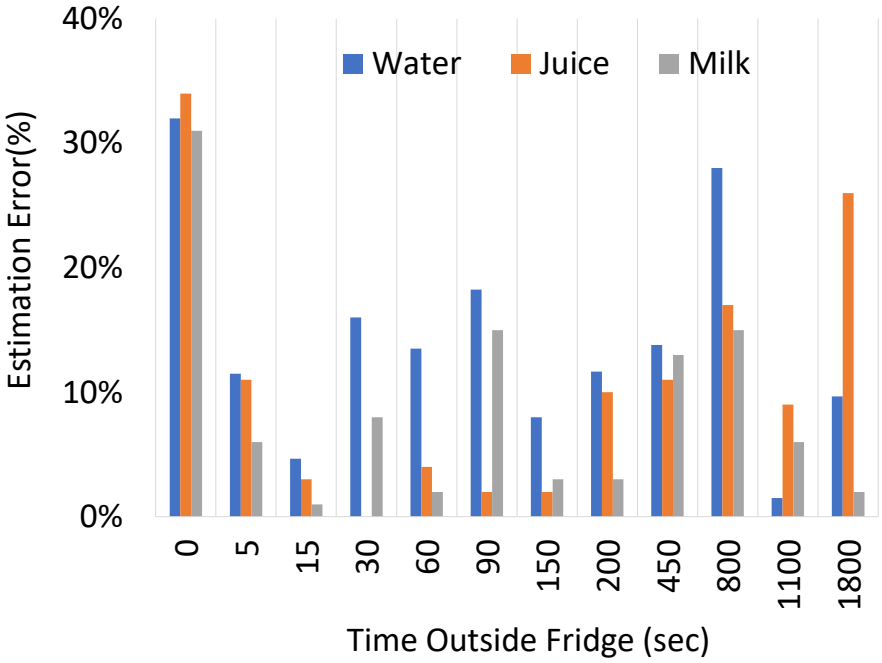


Figure 2.13: Home Study: Estimation Error vs Time Outside Fridge (High Temperature)

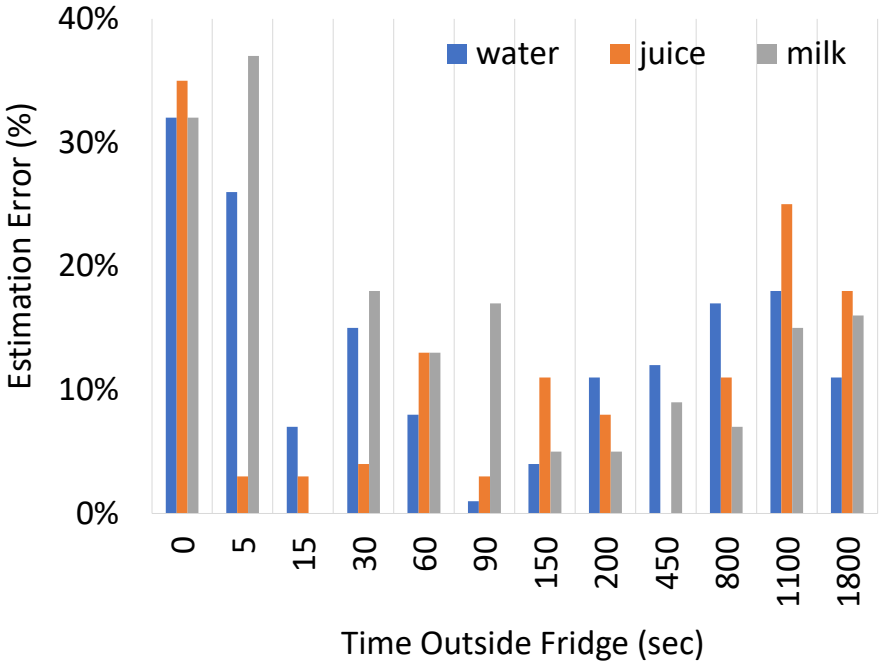


Figure 2.14: Home Study: Estimation Error vs Time Outside Fridge (Low Temperature)

results in higher estimation error when exposure time is 5 seconds. This is intuitively expected as a lower ambient temperature will result in a slower heat gain process, thereby requiring larger time for distinct heat clusters to

emerge. For our settings, however, it does seem that an exposure time of $\geq$15 seconds is enough to provide sufficient clustering accuracy.

Overall, we see that across various ambient temperature ranges, the *SmrtFridge* system typically offers an estimation error of 15-20% when exposure time is $>$ 5 seconds and $<=$15 minutes. For colder ambient environments, the items might require higher exposure times as compared to high temperature ambient environments. In order to improve accuracy for scenarios where exposure time $<$ 5 seconds and $>$ 15 minutes, we can (a) use higher precision IR cameras capable of detecting minute temperature changes (this comes at increased camera cost) or (b) add weight sensors in the fridge (details discussed in later section). While using a high precision IR camera might increase the range of acceptable out-of-fridge (exposure) duration, especially in terms of the lower bound (which determines the minimum change in temperature that is discernible), it is quite likely that even a finer-resolution sensor might not work for very low exposure durations. Accordingly, using a combination of IR sensing and weight estimation might prove out to be a more robust and cost-effective solution in such environments.

2.7.2 Data Collection & User Studies

Our results are based on following user studies:

- **Naturalistic User Study:** In this study, users performed natural fridge-based interactions with N different & common food items, for example, chocolate milk, orange juice, guava juice, etc. Users were asked to naturally insert and remove common food items from the fridge multiple times, without any restriction on how long the item could remain outside. This user study is used principally to study the efficacy of the *item identification* process. It was conducted at two different deployment locations of *SmrtFridge* system namely (a) a university laboratory and (b) an actual home.

- *In-Lab Study:* In this study, conducted in a university lab with an explicit institutional IRB approval[4], 12 different users (members of the general public) initially performed natural fridge-based interactions with 15 different & common food items. In a subsequent phase, 7 new users participated in an expanded study, which included 4 additional fruit & vegetable items (oranges, broccoli, green peppers, eggplant).
- *At-Home Study:* This study, conducted with the *SmrtFridge* system deployed in a fridge kept at an actual home, involved 2 different users who performed natural fridge-based interactions with 15 food items (same items as used in lab study). The home setting also allows us to evaluate performance when the captured images contain a different "residential" background.

- **Quasi-Controlled Micro Study:** The goal of this lab-based separate study was to ascertain the accuracy of *item quantity estimation*, under varying quantity levels, different vertical angles and for different liquid food items. Table 2.3 shows the details of various parameters of this study, e.g., we chose 3 commonly available liquid types (and 3 semi-liquid products in plastic containers as discussed in section 2.7.5) and 3 evenly separated quantity values. Since, many of the products included in this study had paper-based containers, so we arbitrarily chose paper as our container material for this study.

  In this study, 7 users performed *natural-like* interactions with different items, but with explicit instructions on (a) the items to be kept inside or removed from the fridge and (b) how long the items were kept outside (the ambient exposure time).

[4]IRB approval number: IRB-18-134-A112(1118)

S.No	Parameter	Values
1	Liquid Types	Juice, Milk, Water
2	Content Quantity	100%, 60%, 30%
3	Container Material	Paper
4	Time Outside Refrigerator	20 Seconds

Table 2.3: Quasi-Controlled Study Specs (Quantity Estimation)

2.7.3 Item Extraction Performance

We first evaluate the performance of *SmrtFridge's* item extraction pipelines on the data collected during the naturalistic user study. We use two principal metrics:

(a) Intersection Over Union (IoU), which evaluates the relative overlap between the (manually annotated) ground-truth bounding box of the item (BB_{GT}) and the bounding box (BB_{Est}) computed by the automated *SmrtFridge* pipeline. It is computed as $\frac{BB_{GT} \cap BB_{Est}}{BB_{GT} \cup BB_{Est}}$.

(b) Item Coverage $ICov$ ($= \frac{BB_{GT} \cap BB_{Est}}{BB_{GT}}$), which computes the ratio of the intersection area of the ground-truth and computed bounding boxes to the ground-truth bounding box.

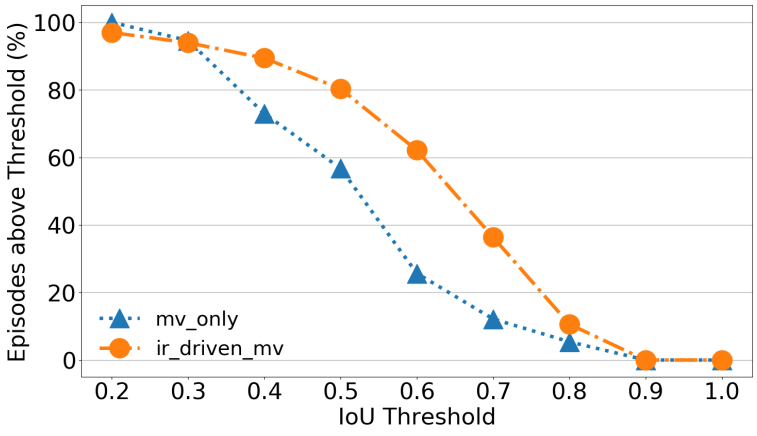


Figure 2.15: IoU scores vs % Episodes above it (Lab Study)

Performance based on Lab Study: Figure 2.15 plots the fraction of extracted images (across 80+ randomly selected user-interaction episodes from the user study)

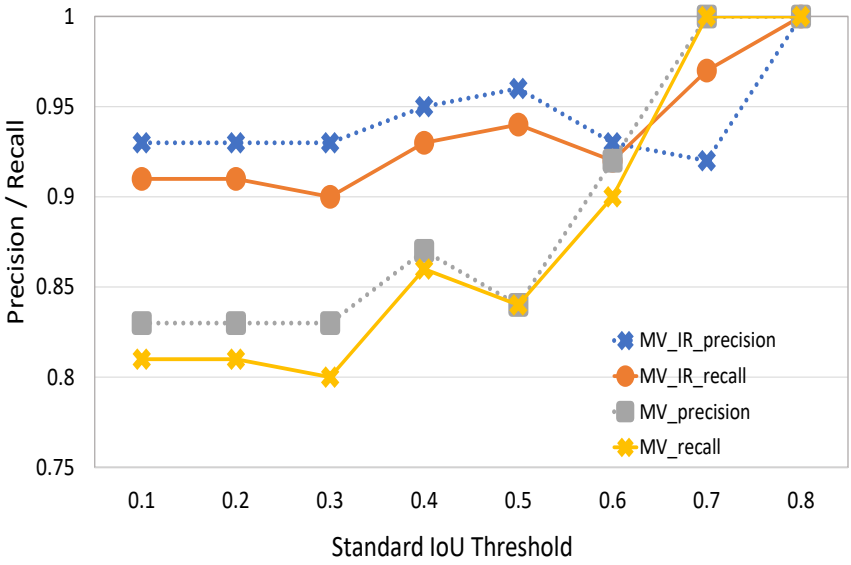


Figure 2.16: IoU score vs image identification precision recall

whose IoU exceeds the specified threshold. We compute the IoU scores separately when using (a) just the RGB motion vector pipeline (mv_only) and (b) just the IR-driven motion-vector based mapping to RGB coordinates (ir_driven_mv). It can be seen that the ir_driven_mv approach provides the best extraction performance: over 60% of images have IoU values $\geq$0.6 (object detection frameworks typically require IoU values higher than 0.45-0.5). In contrast, the pure RGB motion vector-based approach performs the poorest, achieving IoU values $\geq$0.6 in $\leq$ 20% of the images. During analysis, it was found that the food items extracted by the ir_driven_mv approach are relatively more precise, i.e., contain less *non-item* content; whereas the mv_only pipeline tends to extract item images with more non-item background area, therefore resulting in a poor IoU score.

To further understand the importance of high IoU values (i.e., ensuring that the extracted image faithfully captures the food item), Figure 2.16 plots the precision/recall values for *DNN-based item identification* for those images whose IoU value exceeds the corresponding x-axis value. We observe that the item identification accuracy increases with IoU, reaching 95%+ when the IoU value exceeds 0.7.

Figure 2.17 plots the distribution of $ICov$ values, for both the ir_driven_mv ap-

Pipeline	$ICov$ >=**95%**	$ICov$ >=**75%**
MV only	82.4	97.3
IR-driven MV	83.3	97

Table 2.4: Percentage of Episodes vs $ICov$ (Lab Study)

proach and the mv_only methods. We see that the ir_driven_mv technique achieves $ICov$ values $\geq$0.7 in 80% of the interaction episodes. The higher $ICov$ values observed for the mv_only occur because this approach typically extracts a larger fraction of the image but also includes a disproportionately larger 'background' component (hence, the lower IoU score). As we shall show in Section 2.7.4, the presence of a larger background leads to poorer performance of the DNN-based item identifier. To further illustrate the preciseness of *SmrtFridge's* item extraction process, Table 2.4 quantifies the number of episodes (out of a randomly selected 20% of the total episodes) that contain at least 1 extracted item image with $ICov$ values higher than $\{75\%,95\%\}$.

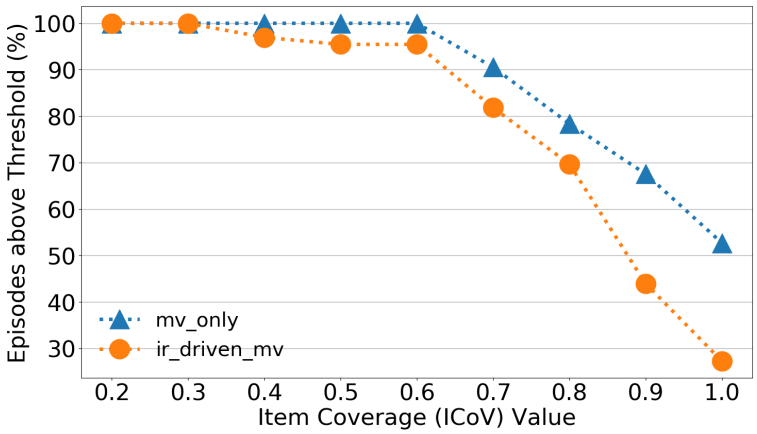


Figure 2.17: $ICov$ score and % of Episodes above it (Lab Study)

Performance based on Home Study: Figure 2.18 plots the fraction of extracted images (across 40+ randomly selected user-interaction episodes from the home study) whose IoU exceeds the specified threshold. It can be seen that mv_only approach performs relatively better, for example, for the mv_only approach over 74% of images have IoU $\geq$ 0.6. In contrast, for the ir_driven_mv approach, 64%

of images have IoU values ≥ 0.6.

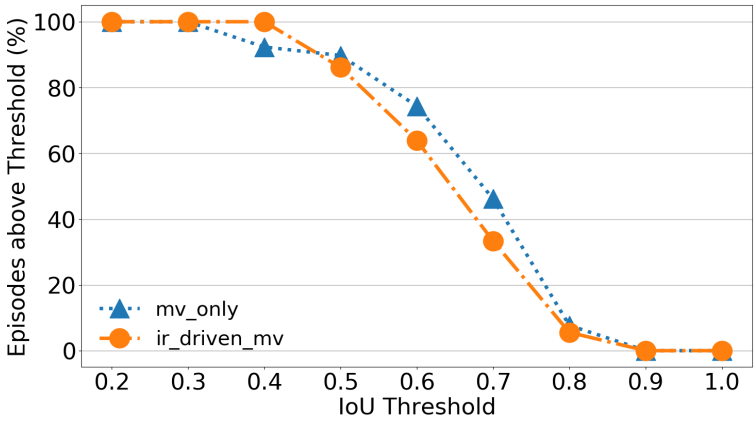


Figure 2.18: IoU scores vs % Episodes above it (Home Study)

Figure 2.19 plots the distribution of $ICov$ values, for both the ir_driven_mv approach and the mv_only methods. We see that the ir_driven_mv technique achieves $ICov$ values $\geq$ 0.7 in 80% of the interaction episodes, whereas the mv_only approach has $ICov$ values $\geq$0.7 in 73% of the interaction episodes.

Comparing the performance of item extraction pipelines across lab & home studies, I observed that the performance of IR-driven MV pipeline decreased for home study as compared to lab study. As per my understanding, this can happen for 2 reasons: (a) smaller participant dataset (2 different users participated in home study compared to 12 for lab study) and (b) fog issues inside the fridge at home (discussed in detail in sec 2.9) that might have affected the IR sensor and therefore causing the drop in overall IR-driven extraction accuracy.

Table 2.5 quantifies the number of episodes (out of a randomly selected 50% of the total episodes) that contain at least 1 extracted item image with $ICov$ values higher than $\{95\%, 75\%\}$. It can be seen that compared to the lab setup, the item extraction performance for $ICov \geq 95\%$ drops for both extraction pipelines, whereas for $ICov \geq 75\%$, we don't see any major accuracy drop. This drop can be attributed to the fact that while lab-setup was a semi-controlled setup (with fewer objects in background and proper lighting arrangements), the home-setup was a

completely uncontrolled setup, thus making it more challenging to extract items with $ICov \geq 95\%$. But, even in such an uncontrolled setup, it can be seen that both the extraction pipelines maintained $ICov \geq 75\%$ for $92\%+$ of episodes and looking at Figure 2.16, we can take a hint that achieving an extraction score $\geq 75\%$ can potentially result in item identification accuracy of 90%+.

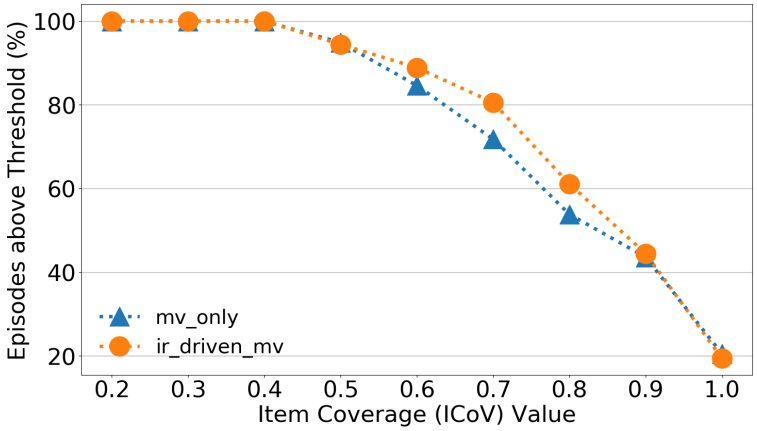


Figure 2.19: $ICov$ score and % of Episodes above it (Home Study)

Pipeline	$ICov$ >=**95%**	$ICov$ >=**75%**
MV only	56.5	92.3
IR-driven MV	51.2	97.4

Table 2.5: Percentage of Episodes vs $ICov$ for Home study

Key Takeaways: Overall, it can be seen that *SmrtFridge's* item extraction pipelines were able to effectively isolate the food items from their background when *SmrtFridge* system is deployed at two different locations. The setup in university lab resulted in comparatively better overall item segmentation performance.

2.7.4 Item Identification Performance

This section describes the item identification accuracy, based on the extracted images (from an initial study with a 15-item classifier & 12 individuals, followed by a 19-item classifier with additional fruit & vegetable items & 7 users), achieved by the ResNet-based DNN.

Performance based on Lab Study: Table 2.6 plots the item classification results (for episodes involving the original 12 users who interacted with the original 15 food item classes), for both the 15-class classifier and the subsequent 19-class classifier. It can be seen that the combined pipeline results in the highest and identical precision/recall values (of $\sim$0.84). Moreover, the results are fairly stable over the 15-class and 19-class classifiers. As a point of comparison, the food item precision/recall is 74% and 72% respectively, for the episodes involving the 7 new users, who interacted solely with the 4 new fruits & vegetable items.

Approach	15 class Classifier		19 class Classifier	
	Precision	**Recall**	**Precision**	**Recall**
Motion Vector (MV) Only	0.82	0.79	0.83	0.81
IR driven MV	0.80	0.78	0.81	0.79
MV+IR Merged Pipelines	0.83	0.83	0.84	0.84

Table 2.6: DNN-based Item Identification accuracy (per Episode)

Following observations can be made:

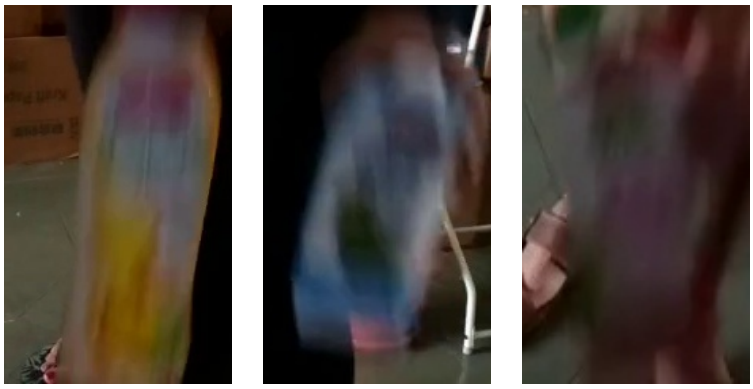
- The overall item recognition accuracy is high but not as high as the 97%+ accuracy reported on the externally curated training data. In large part, this is due to the lack of sufficient *relevant* training data for our classifiers. In particular, the training corpus consists entirely of images of items extracted from the Web or shot in close proximity by a video camera. These training images are quite distinct from the partial views of items captured by the *SmrtFridge* RGB+IR sensors. The anticipation is that the accuracy will improve as the corpus is continuously expanded in the real world (similar to approaches used by consumer ML-based devices such as Amazon's Alexa™) to include more such *in-the-wild* images.

- The accuracy is lower for the newer episodes that involved the 4 new food items. This was principally due to the lack of sufficient *appropriate* training images—unlike canned items, fruits and vegetables have greater diversity in shape and color, and thus require more diverse training data.

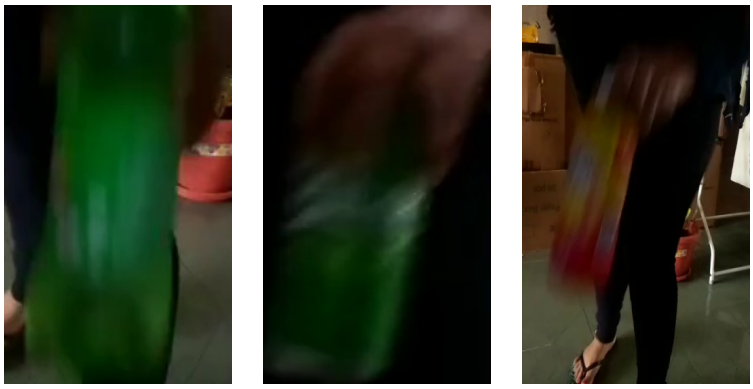
Performance based on Home Study: Executing the item identification pipeline on data collected from home study resulted in $95+\%$ of extracted items being classified as background. This was caused by the extracted items being quite blurry in nature (see figure 2.20). Unlike the lab setup, no additional light sources were installed at home, therefore resulting in comparatively poorer lighting conditions at home. So, in order to capture interactions, I reduced the shutter speed (to allow more light into the lens) of the Rpi camera. The Rpi camera and user being in vicinity of each other (distance $\leq 2m$ during item interaction), a reduced shutter speed led to blurry captures.

In my opinion, there are two ways to handle such problems: (a) by improving the ambient lighting conditions near the fridge area and (b) by using a fixed-focus (fixed at $\sim$ 2m rather than infinity) camera with high ISO gain capabilities. As a preliminary evidence of performance improvement in better lighting conditions, achieved by using an additional light source, I collected data for 8 different episodes and ran item identification pipelines on it. I found that the precision values, for classification (using 19 class classifier) of extracted food items, was found to be 0.73, 0.70 & 0.72 for MV only, IR-driven MV & MV+IR merged pipelines respectively. This observation provides a pathway to overcome the limitations imposed by the poor lighting conditions in a typical home deployment.

Alternative Classification Strategies: To further underline the importance of accurate sub-image extraction, we computed the accuracy of a baseline where the DNN classifier operated on full-HD images (containing both the food item and miscellaneous background content). The DNN classifier then performed very poorly, achieving precision and recall values of only 0.53 and 0.20. Similarly, if the classification is performed only on 1 extracted image (as opposed using the highest cumulative likelihood across all frames), the item identification accuracy drops to 0.48.



Smoothie Coconut Water Guava Juice



7 Up Beer Can Ice Tea

Figure 2.20: Examples of extracted images from home study

2.7.5 Quantity Estimation Performance

I also performed a quasi-controlled study to evaluate *SmrtFridge's* coarse-grained quantity estimation technique. Figure 2.21 plots the estimated quantity for 3 different liquids {juice, milk, water}, and 3 different fractional quantities {30%,60%,100%}. The plot shows that these 3 levels are distinguishable (distinct median values, with low overlap between 1^{st} and 4^{th} quartiles). However, the estimates are significantly more noisy for juice when the container is only 30% full). Studies with additional semi-solid items {yogurt, ketchup, peanut butter} show that the estimation error remains within 10-20%, indicating the robustness of our technique.

Coarser Estimation/Classification: While fine-grained quantity estimation is challenging for certain (liquid, container) combinations, coarser-grained estimates

are acceptable for many applications. For example, an application that generates alerts (when the food quantity becomes very low) may just need to know when the quantity drops below, say, 20%. Accordingly, we now study the accuracy of the coarser-grained classifier that assigns the captured IR image into one of 3 bins/classes: 30—60—100%. For this ternary classification problem, we achieve a classification precision of 75% and recall of 71%. Overall, our results suggest that IR-based technique may be useful for obtaining coarse-grained quantity estimates (average error of $\sim$15%).

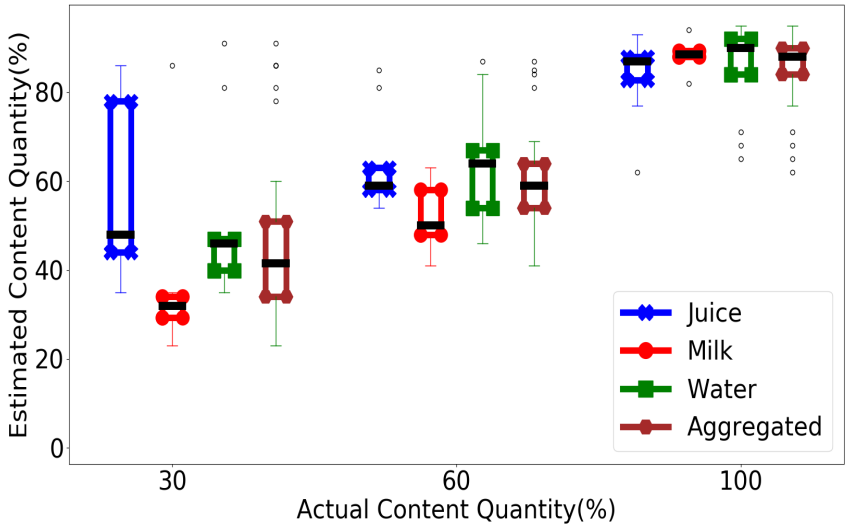


Figure 2.21: Median Accuracy of Quantity Estimation

Item Insertion Angle vs Accuracy: While the item is being re-inserted into the fridge, the user might hold it in many ways. One of the variables is the inclination angle, i.e., the angle at which an item is with respect to horizontal plane, e.g., floor. This section describes the study on whether the quantity estimation accuracy depends on the inclination angle of the container. Figure 2.22 shows mean quantity estimation error, as % of whole container, when a juice container was put inside at 7 different horizontal angles (via a controlled study) ranging from $\theta = 0$-180° (container is vertical at $\theta = 90^\circ$). It can be seen that the estimation error is usually within 10-25% (and thus sufficient for coarse-grained resolution), unless the container is horizontal $\theta = \{0, 180\}^\circ$. As an intuitive explanation, note that most food contain-

ers are taller and narrower. The same residual quantity thus results in a larger empty *height* when the container is vertical $\theta = 90°$, and a much smaller empty height when horizontal. Moreover, we observed that even modest hand movements during the interaction can cause the liquid to splash vertically around inside the container and 'contaminate' the empty portions "above". Given the relatively low spatial resolution of our IR camera, the clustering error (illustrated in Figure 2.23) is thus much larger when the container is horizontal, than when vertical.

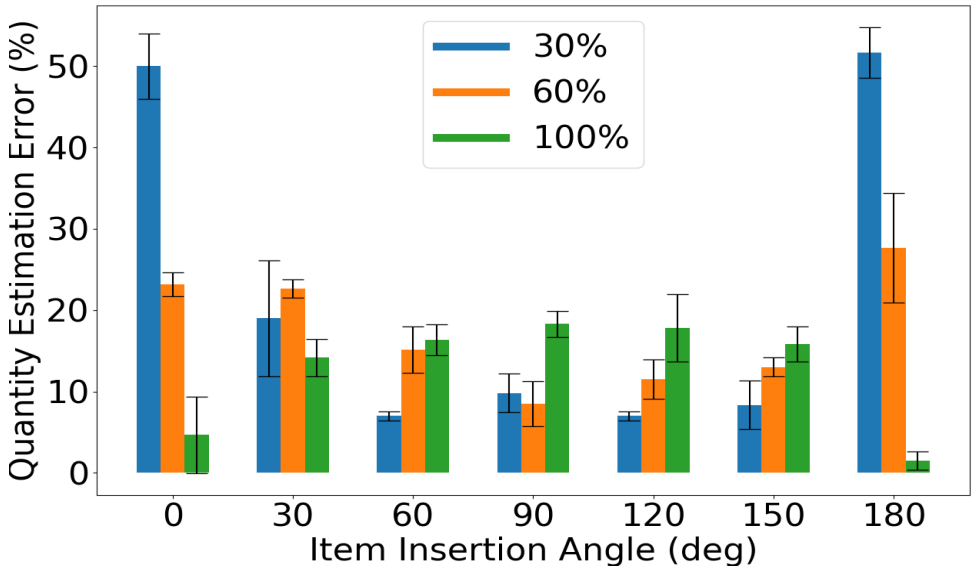


Figure 2.22: Quantity Estimation Error vs Item Angle

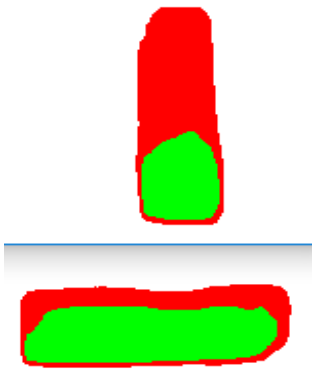


Figure 2.23: Thermal Clusters for $\theta = 90°, 0°(content = 30\%)$

2.7.6 Additional Performance Characteristics

Supporting Multiple Items: While the *SmrtFridge* pipeline supports the user's concurrent interaction with multiple items, we observed that such interaction (e.g., retrieving a milk carton and a yogurt container together) is very unusual (never occurred in our Naturalistic study). To understand the performance of *SmrtFridge* under such possible multi-item interactions, we collected data for 8 episodes, where 2 users were explicitly instructed to retrieve 2 items concurrently. In this admittedly small sample, *SmrtFridge's* clustering technique reliably identified 2 distinct items and extracted them with IoU values between 0.63-0.71. However, more detailed studies are needed, as such concurrent retrieval may give rise to other non-obvious usage artifacts (e.g., occlusion of one items).

Energy Consumption: As per surveys [5], households tend to interact with their fridge about 15-25 times per day. If *SmrtFridge* system is triggered at this frequency on a daily basis, the amount of energy consumption becomes an important factor. This section talks about the observed energy consumption of *SmrtFridge* . Via measurements with 40+ distinct episodes, it was found that average energy consumption is 7.5mWh/episode for the quantity estimation pipeline, and 90mWh/episode for the item extraction & recognition pipeline. In contrast, the yearly average energy consumption of a typical fridge (e.g., Toshiba GR-R20STE 185L), is 566 kWh. Accordingly, *SmrtFridge* is expected to impose an additional overhead of only **0.15**% on a fridge's energy consumption.

2.8 Discussion

2.8.1 Privacy Concerns

SmrtFridge's use of an outward-facing camera can raise privacy concerns: consumers may be wary of devices that not only capture images of food items but also,

[5] `https://www.housebeautiful.com/uk/lifestyle/storage/news/a2110/fridge-food-cupboard-habits/`

potentially, that of the individual and the residence's background, *even if all image processing is performed locally on the fridge*. We believe that commercial products can address this issue via appropriate design and placement of cameras, while utilizing *SmrtFridge's* interaction-driven paradigm. In particular, instead of the outward-facing camera setup, we can deploy multiple narrow-FoV (field-of-view) cameras on the rim of the fridge, such that they are capable of only taking 'sideways' images of the fridge. However, such a setup can increase occlusion (at least on one side). Accordingly, we may need to modify the image extraction pipeline to accommodate multiple simultaneous images (of varying occlusion) from multiple cameras.

2.8.2 Extending to Other Food Types

The experimental results presented here focused primarily on discrete container-enclosed items, as discrete food items (e.g., oranges & eggplant). However, the quantity estimation of such discrete items is currently unexplored and will require newer approaches, e.g., thermal segmentation is unlikely to be able to distinguish between 1, 2 or 3 bananas.

2.8.3 Additional Sensors for Finer-grained Sensing

SmrtFridge's current visual recognition pipeline recognizes only food item *types/ brands*, and not instances. For example, if a fridge has 2 Coke cans (both 50% full), the system cannot distinguish between them if one of them is removed and returned (with 30% residual content). Additional sensor types may help overcome such limitations. For example, explicit weight sensors (load cells), can help provide fine-grained estimates of changes in the fridge's weight, which can then be used to discriminate between multiple identical items. A single 100 lb ($\approx$ 45 kg) Futek LSB200 sensor[6] can detect load changes as small as 10 grams. Other novel sensors may enable additional functionality, such as detection of expired food items. For

[6] `http://www.futek.com/files/pdf/Product%20Drawings/FSH00091.pdf`

example, Goel et al. [49] have applied hyper-spectral imaging to infer the aging of food items such as fruits.

2.8.4 Out of Scope Functions

As consumers can certainly desire additional capabilities from a smart fridge, it is important to explicitly state the functions that *SmrtFridge does not* currently support. Very specifically:

- *No Product Expiration: SmrtFridge* does not have any notion of detecting the possible expiration dates of individual food items. While *SmrtFridge* can provide the duration for which an item has been residing in the fridge (and a rule-based back-end may trigger alerts when a specified period has been exceeded), more precise expiration tracking will require coupling our mechanisms with alternative approaches (e.g., OCR-based parsing of expiration dates on containers).

- *No Support for Unlabeled Food Items: SmrtFridge's* operational logic is based on extracting visual images of a food container or discrete food items (e.g., fruits), and then performing DNN-based recognition of the product. Accordingly, *SmrtFridge* cannot presently support recognition of unlabeled food items (e.g., home cooked foods such as salads or curries), although future versions can integrate ongoing deep learning work on recognition of cooked foods (e.g., FoodAI [3]).

- *Approximate Support for Quantity Estimation: SmrtFridge's* IR sensing techniques help to provide coarse-level estimates of residual food quantity in containers (e.g., less than 25% remaining). However, *SmrtFridge* does not aim to measure such food quantity precisely (e.g., 30 mg of juice). It is likely that the addition of high-resolution weight sensors might enable more precise quantity estimation of food items.

- *No Tracking of Specific Item Instances: SmrtFridge's* visual sensing effectively recognizes specific food types or brands (e.g., a can of *Coke*), rather than specific individual item instances.

2.9 Reflections and Lessons Learnt

This section highlights some of the lessons learnt from the design & performance analysis of *SmrtFridge* system.

2.9.1 Lighting Conditions

Being a camera based system, ambient lighting conditions surrounding the fridge might impact the overall performance of *SmrtFridge* system. These impacts can be seen in our lab and home deployments – the accuracy of *SmrtFridge* in the home setting was slightly inferior compared to the lab setup. In addition, the number of test users in the home setting was quite low (just 2 users), implying that those results may be more susceptible to unique individual-specific behavioral artifacts exhibited by the users. To tackle the light issues, one might reduce the camera shutter speed, which in-turn might lead to blurry images (as seen in section 2.7.4). So, a trade-off must be established between the camera shutter speed and the ambient light intensity. Also, it might be useful to explore if a NoIR camera (no IR filter)[7] coupled with an IR flash light might help alleviate this issue.

2.9.2 Location of Cameras

During the experiments, we observed that sometimes users might put items in front of the cameras, therefore blocking their view. So, it might be useful to install cameras such that there is no place to put items in-front of them. Based on our insights about possible camera positions (section 2.6.1), we can say that if we move cameras

[7] `https://www.raspberrypi.org/products/pi-noir-camera-v2/`

from position 3 to nearer to the door hinges, then it might help resolve this issue.

2.9.3 Fridge Steam on Camera Lens

Depending on the factors like fridge type, temperature and so on, it is possible that steam from the fridge might accumulate on the lens of RGB camera, therefore creating a blurry effect. This will impact the performance of item extraction and item identification pipelines. To tackle this, an anti-fog coating[8] can be applied on the camera lens.

[8] https://www.weetect.com/anti-fog-solution-guide/

Chapter 3

Collaborative Energy-Efficient Multi-Camera Sensing

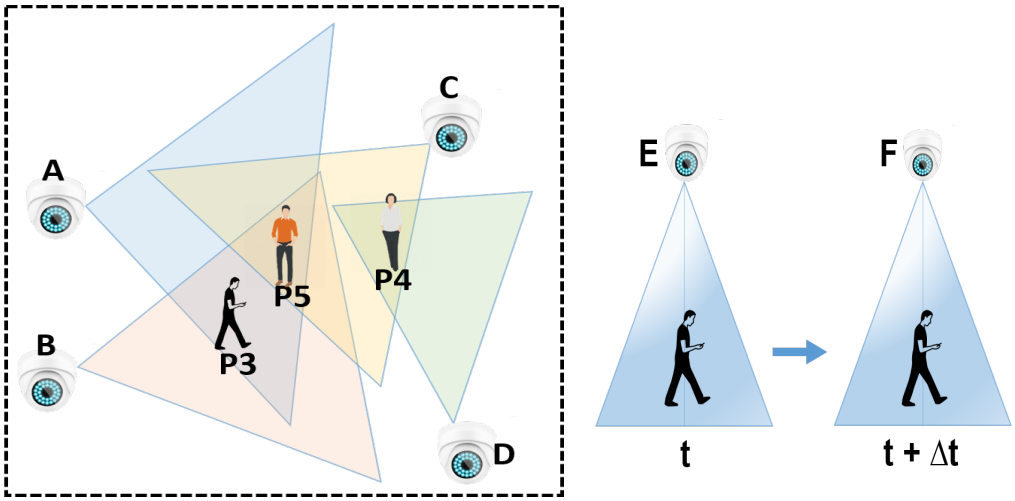
3.1 Introduction

Vision-based sensing has seen dramatic advances in both image sensing hardware (4K resolution image sensors are now priced below $25) and Deep Neural Network (DNN) based models (e.g., YoLov3 [94]). With rapid progress in miniaturization technologies and consequent drop in prices, IoT devices such as cameras are witnessing a rapid increase in deployment rates in public spaces, such as college campuses, subway stations and parks (e.g., Singapore's multi-camera lampposts project ($95,000+$ lampposts) [103], Beijing & London each having 800,000 & 628,000 cameras [36] respectively). By executing a variety of vision-based processing pipelines on such camera-generated video feeds, we can enable a host of advanced IoT-centric applications such as real-time occupancy estimation [29, 24], crowd flow monitoring [32] and even tracking of individual/group movements [68, 31, 96] (the last one is particularly relevant for supporting currently-popular functions such as contact tracing).

Despite many applications, the energy cost of vision sensing continues to be a formidable challenge in the deployment of continuous vision sensing-based ap-

plications in public spaces, such as university campuses and outdoor parks. The overall energy consumption of a typical vision sensing node is composed of (a) capture/sensing energy: consumed during image/video capture (b) storage energy: consumed while storing/retrieving sensor data to/from memory and (c) transmission energy: consumed when sensor data is transmitted over the network for further processing. To facilitate the large-scale deployments of vision-sensors, it is necessary to reduce the energy overheads of vision sensing, ideally to the point where such sensing can utilize battery-less, energy-harvesting based sensors. While recent work on *energy proportional* vision systems [71] has shown that decreasing the image resolution or frame rate can reduce the sensing energy overheads [109], they come with an *energy-vs-accuracy tradeoff*: lowering the resolution might save energy but it can also diminish the accuracy of visual tasks, such as object detection [94].

In this chapter, I shall demonstrate that this energy-accuracy tradeoff can be tackled by using the concept of *collaborative intelligence* [21], whereby the inference pipelines of different sensors work cooperatively to provide performance enhancements that are not realizable via independent operation. It has been observed that many large-scale deployments of networked vision sensors are characterized by *partially overlapping spatial coverage*, i.e., a particular region might be covered by more than one cameras at a time. For such overlapping regions, the availability of multiple simultaneous views, usually from differing perspectives, offers an opportunity to preserve the task accuracy while lowering the captured image resolution, therefore resulting in reduction of *sensing (image capture), storage & communication* energy overheads. This chapter describes one such approach to energy reduction–namely, the intelligent and selective reduction of the imaging resolution of such vision sensors.



(a) Spatial Overlap (b) Temporal Correlation

Figure 3.1: Illustration of spatial overlap and temporal correlation among multiple cameras

3.2 Multi-Sensor Collaboration

In a multi-sensor deployment scenario, sensor collaborations can be useful in various ways, e.g., to increase overall sensing coverage, reduce overall system energy consumption or improve the ability to detect less perceptible objects. While a sensor collaboration can involve multiple heterogeneous sensors, e.g., a Passive Infrared (PIR) sensor can enable a camera to capture images as and when motion is detected, this work focuses on *multi-camera* collaboration where multiple RGB "visual" cameras, deployed at the 'edge' of a networked sensing infrastructure, collaborate with each other to reduce overall system energy consumption. A large-scale camera deployment often leads to a partial spacial-overlap among cameras as shown in figure 3.1(a). Similarly, as shown in figure 3.1(b), cameras in these deployments might also have a temporal correlation among them, i.e., a person visible in camera E at time t might also be visible in camera F after $t + \Delta t$ time. While both these properties can be exploited for collaboration, this dissertation focuses only on the spatial overlap based collaboration.

3.3 *CollabCam* Overview

In this chapter, I discuss the design and performance analysis of *CollabCam*, a framework for multi-camera collaboration for energy efficient sensing. *CollabCam* utilizes selective resolution reduction combined with collaborative inference to achieve its goal of reducing overall energy consumption of a vision sensor while keeping the object detection accuracy intact. However, each vision sensor typically has only *partial* spatial overlap, i.e., only a portion of the region within its field of view (FoV) is likely to be shared with the FoVs of other cameras. Given this characteristic, the camera resolution cannot simply be reduced uniformly for the entire image: while a reduced resolution might offer acceptable task accuracy in the *overlapping area*, it would still suffer performance loss in the non-overlapping, exclusive part of its FoV. *CollabCam* therefore introduces a novel paradigm that enables it to overcome this conundrum: **Mixed Resolution Frame Sensing** (MRFS). MRFS differs from prior work on dynamic camera resolution adaptation (e.g., [57]), which implicitly assumes that any single frame is acquired at a *constant* spatial resolution. As illustrated in Figure 3.2 (where 3 different regions of the image frame are sampled at 3 distinct resolutions), MRFS breaks away from this assumption by acquiring *different portions of a single image frame at different resolutions*. Intuitively, this mixed-resolution operation allows *CollabCam* to obtain the best of both worlds, using a lower resolution (with lower energy consumption) for the overlapping areas, while preserving the use of higher resolution for the non-overlapping areas.

From deployment perspective, *CollabCam* is composed of multiple MRFS-enabled and spatially-overlapping cameras, where each camera's mixed-resolution image frames are optionally stored in memory and then transmitted wirelessly (see Figure 3.3) to a common analytics node (e.g., an edge computing platform). This node can collectively apply the relevant visual processing pipeline, without requiring any deployment-specific DNN training, to the composite set of captured images.

Figure 3.2: MRFS: Mixed-resolution Image with 3 sub-regions

3.3.1 Key Challenges

In order to realize the goals of *CollabCam*, following key challenges must be addressed:

- **Autonomously & Accurately Determine Overlapping Areas:** To take advantage of the MRFS paradigm and reduce the sensor resolution over shared regions in an *autonomous* manner, the foremost step is to establish the overlapping regions between cameras. So, each camera first needs to know the different parts of its FoV that are shared with other peer cameras. In order to avoid the need of manual intervention, which can become a prohibitive overhead in large-scale deployments, the overlap estimation should be done in an autonomous manner, i.e., without requiring any explicit human involvement. Secondly, the amount of energy savings also depends on the size of overlapping region, i.e., more overlap can lead to more savings, so, the overlap region estimation should be done accurately.

- **Optimal Resolution Estimation for Overlapping Areas:** The total amount of energy savings from multi-camera collaboration depends on the extent of spatial overlap between cameras and the respective imaging resolutions of the overlapping regions. So, after identifying the overlapping areas, the next chal-

lenge is to estimate the optimal resolution for them. It is important to select optimal sensing resolutions for the overlapping areas because while a ”lower than optimal” resolution might cause unwanted drop in end-task accuracy, the ”higher than optimal” resolution wouldn't save the maximum possible amount of energy. The choice of optimal resolution might depend on various application/deployment specific factors such as (i) target object type, e.g., vehicles might be easier to detect at lower resolution than pedestrian users (ii) object distance from camera, e.g., closer objects might appear bigger and therefore can be detected even at lower resolutions than objects that are far and appear smaller in camera. A collaboration system must be able to maximize energy savings while keeping the end-task accuracy drop as minimum as possible.

- **Maintaining Task Accuracy with Low-Resolution Images:** The images created using MRFS will have lower resolution in overlapping regions therefore can result in reduced task accuracy if an off-the-shelf object detector is used for inference. For example, as we shall see in Section 3.7.2, reducing the overlapping area resolution from 512x512 to 70x70 results in 48-61% drop in mAP (Mean Average Precision) score for our benchmark datasets. *CollabCam* should be able to maintain this object detection accuracy while capturing images at lower resolution.

3.3.2 Operational Concept

Figure 3.3 illustrates a high-level operational overview of *CollabCam*, where multiple vision sensors collectively monitor a larger area (such as a university campus or a city neighborhood), with varying spatial overlap in the regions covered by each individual camera. *CollabCam* employs a hub-and-spoke architecture where the vision sensors capture and transfer images to a common gateway or edge device, denoted as E. The device E receives all the camera image streams and executes

the relevant DNN-based inference tasks over the entire area, e.g., performing object detection over each individual sensor stream. In this conceptual diagram, we observe that: (i) cameras A & B share an overlapping area of coverage (C1), and thus will be able to simultaneously observe (from different perspectives) the person P1 located in this overlapping area (ii) likewise, cameras C & D share an overlapping coverage area (C2), and thus simultaneously observe individual P2 and (iii) cameras A, B & C together have a common overlapping area of coverage (C3) and simultaneously observe individual P3. The vision here is that the *CollabCam* system will take advantage of these overlaps to reduce overall energy consumption by selectively reducing the camera resolution for those portions of the camera-specific frames that correspond to such overlapping areas. For this work, I am assuming that the cameras, and thus the overlapping areas, are static and do not change over time. The underlying hypothesis (validated in Section 3.9) is that collaborative inference, by fusing the inputs from the multiple cameras that 'cover' a common overlapping area, will help preserve the task accuracy (e.g., for object detection) in spite of the loss in image quality due to the reduced camera resolution. Note that *CollabCam's* eventual performance gains are *deployment-specific*, being dependent on the amount of actual spatial overlap.

3.3.3 Resolution vs Energy Characteristics of a Vision Sensing System

A typical networked vision sensing system has multiple sensing nodes, where each node can capture images, (optionally) store them in memory and transmit them over a network. To justify *CollabCam's* approach of reducing image resolution to reduce energy overheads, it is useful to understand the internal architecture of a typical vision sensing node. As shown in figure 3.4 (partially reproduced from [71]), such a node contains one or more of following components (a) image sensor(s) to capture images (b) processing & memory units to process and temporarily store the

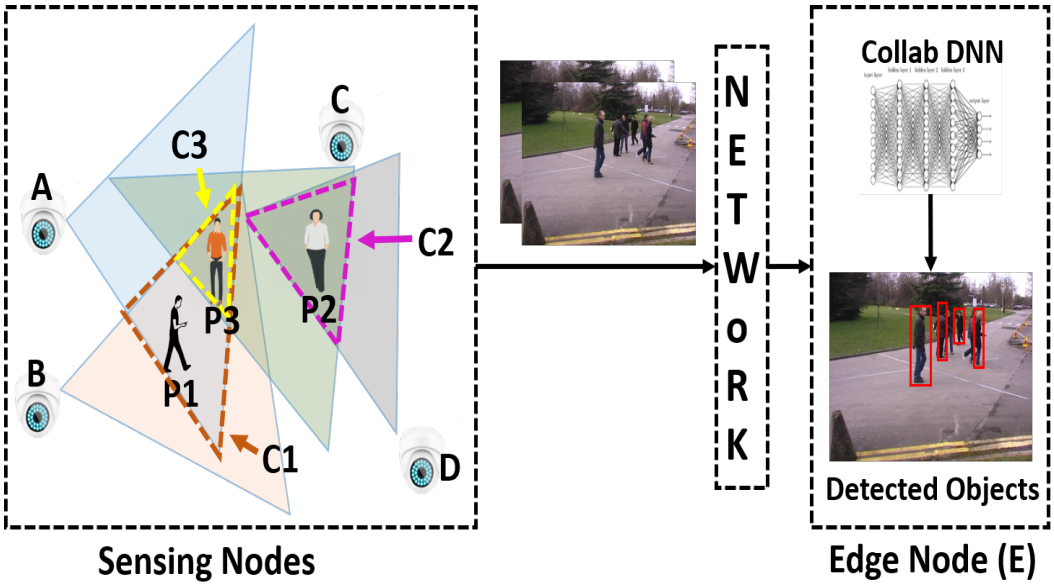


Figure 3.3: Illustration of Operational Concept of *CollabCam*

captured images and (c) a network module to transfer these images. The energy consumption of each component (which cumulatively define the sensor node's total energy overhead) is effectively proportional to the number of captured pixels.

A typical CMOS sensor consists of an array of MxN pixels, where the light captured by each pixel's photo-detector is converted and stored in a capacitor. Next, the Analog Signal Chain component (which typically imposes $\sim 70-85\%$ of the total power consumption [71]) employs several amplifiers and Analog-to-Digital (ADC) circuits to convert this capacitive voltage to a digital output. The sensor's image processor then performs basic digital image processing tasks, e.g., white-balancing, denoising etc. By reducing the sensing resolution, we effectively reduce the amount of analog-to-digital (pixel content) conversions. For example, in a truly energy-proportional configuration, reducing the resolution by half along both $x\&y$ coordinates should result in a 4-fold decrease in the ADC energy consumption. Moreover, such a reduction in the volume of digitally-captured pixels should also reduce the energy overheads of the subsequent image processing steps, such as denoising, storage & the wireless transfer of the captured data to an external processing unit (e.g., to an edge device).

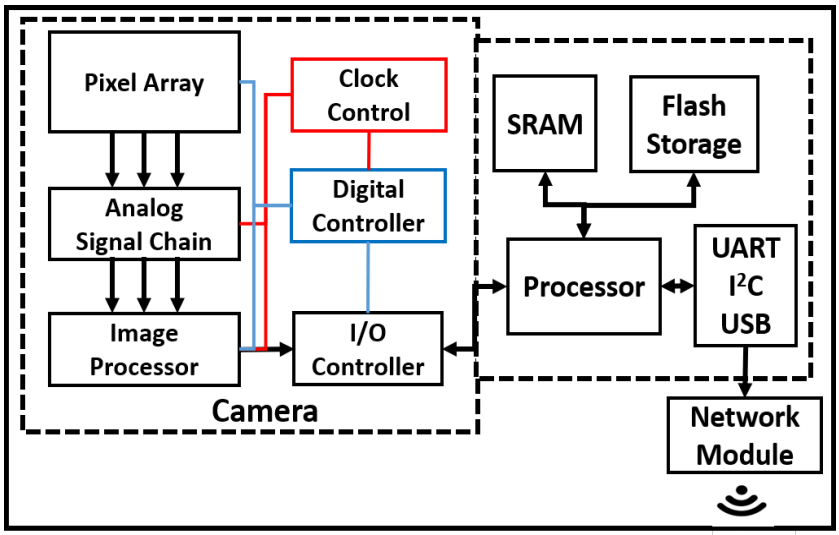


Figure 3.4: Components of a vision sensing node

3.3.4 System Architecture

Figure 3.5 outlines the various functional components of *CollabCam*, some of which execute on each individual vision sensor with the other components executing on the edge device. This section provides a brief overview of these components, outlining how they work together to achieve the vision outlined above:

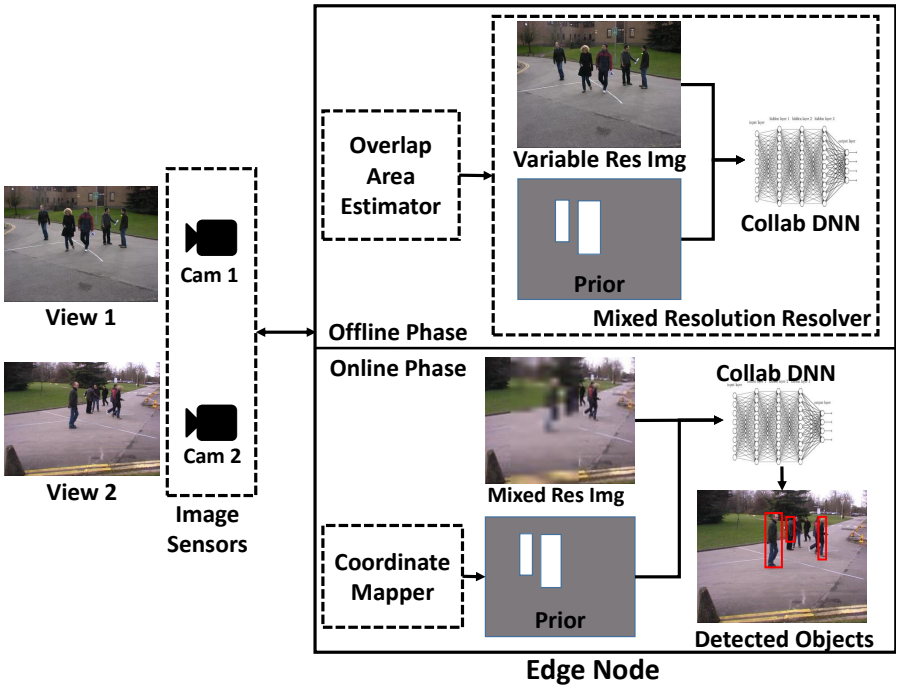


Figure 3.5: *CollabCam* Components (Sensor & Edge)

1. *Overlap Area Estimator: CollabCam* first automatically and autonomously (i.e., without any explicit human configuration) estimates the spatially overlapping areas, if any, between a set of cameras. For this, *CollabCam* employs an *object matching (ReID)* based approach, i.e., if the same object is visible in two simultaneously captured images from a camera pair, the corresponding locations of that object, in each camera's reference system, are part of their mutually overlapping area. By repeating this process over a large-enough set of frames and aggregating the locations of all ReID-ed objects, we can estimate the overlap area between each camera pair. The Overlap Area Estimator component (described in detail in Section 3.5) executes on the edge device, where it has access to multiple images from each sensor. Since the cameras are assumed to be static, this estimation is performed offline, prior to the actual runtime MRFS-based adaptive sensing.

2. *Mixed Resolution Resolver:* Given the knowledge of the areas of overlap between multiple cameras, *CollabCam's* operation requires each vision sensor to subsequently determine the *minimum* required resolution for the corresponding overlap area. This is done by the Mixed Resolution Resolver component of *CollabCam* . To determine this resolution, I shall use studies on benchmark datasets (Section 3.9) to first empirically establish the relationship between the resolution used and the accuracy of the corresponding vision task for such overlapped regions, and thereby identify the *optimal* resolution for each (vision sensor, overlapping area) tuple. The resolver can communicate these optimal choices, along with the corresponding region specifications, to each individual camera.

3. *Cross-Sensor Coordinate Mapper:* For fusing the perspectives of multiple collaborating vision sensors, it is necessary to perform coordinate mapping, i.e., transform an object's location from one camera's coordinate system to another camera's coordinate system. While well-established techniques, such

as Homography [86], triangulation or projective transformation [69] exist to perform such mapping, they often assume that details of each camera's parameters (e.g., height, location) and orientation are known a-priori. While developing new coordinate mapping techniques is not part of my goal or contribution, Section 3.6 will describe a simple and reasonably effective linear regression based mapping technique, that operates autonomously without requiring explicit knowledge of camera parameters.

4. *Collab-DNN (SSD):* To provide competitive accuracy even from lower resolution images, *CollabCam* requires an enhanced DNN inference model that performs the intended visual task by ingesting multiple camera perspectives, of mutually shared regions, suitably mapped to a common coordinate space. In Section 3.7, I shall describe the enhanced object detector model (applicable for our canonical 'object detection' task), called Collab-SSD, which does not require deployment-specific training. I shall demonstrate that it offers a superior accuracy-vs-resolution trade-off than the baseline SSD (Single Shot Detector) model.

5. *Mixed Resolution Image Sensor:* To achieve the targeted energy savings, *CollabCam* requires each vision sensor to be MRFS-capable. In *CollabCam* , once the overlapping sub-region(s) and their sampling resolutions have been determined, these parameters are transmitted from the edge device to each individual pervasive vision sensor, which then configures itself to generate and transmit back the corresponding mixed-resolution image streams. Section 3.8 describes a first-of-a-kind prototype implementation of such an MRFS-capable image sensor and characterize its energy vs resolution performance.

3.4 Benchmark Datasets

CollabCam uses two publicly available, multi-camera datasets, i.e., PETS [11, 12, 43] and WILDTRACK [33] for performance analysis.

PETS Dataset

PETS is an open dataset publicly released in 2009 as part of a challenge. PETS is a multi-camera, variable crowd-density dataset containing human beings. It has total of 8 cameras, some of which capture near views and others capture far view of a specific geographical area. PETS cameras are deployed such that many of them have spatially overlapping regions with other camera(s). Table 3.1 summarizes overall details of PETS dataset. The high level deployment locations of all the cameras is as shown in figure 3.6. PETS is a low-medium density crowd dataset created for several pedestrian activities, e.g., walking, running.

Total Cameras	Resolution	Frames/Cam	Frame Rate	Area Covered	Participants
8	720x576	795	7	100m x 30m	40

Table 3.1: Details of PETS Dataset

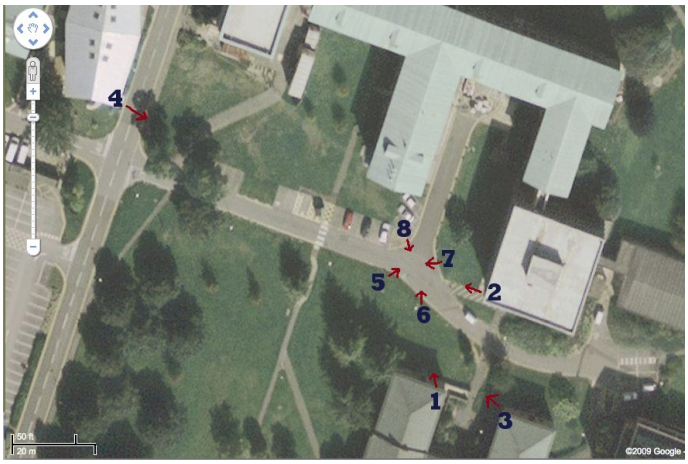


Figure 3.6: High level view of camera deployment in PETS

For this dissertation, I use cameras 5, 6, 7, 8 because of requirement of spatially overlapping cameras. Out of many sets of data (each aimed at a specific type of

task), I use $S2\_L1$ (Dataset S2: People tracking, difficulty : Level 1, sparse crowd). Figure 3.7 shows some sample images from PETS dataset.

(a) View 5 (b) View 6

(c) View 7 (d) View 8

Figure 3.7: Sample images from PETS dataset

WILDTRACK dataset

WILDTRACK is a large-scale, multi-camera public dataset created with the aim of testing object detection, tracking algorithms in the wild settings of high crowd-density. The data collection setup consists of 7 HD cameras with partially overlapping FoVs deployed outdoor in a University campus. Table 3.2 summarizes some of the details of WILDTRACK dataset. WILDTRACK dataset is a high crowd-density dataset and captured in a non-actor setup, i.e., participants are not given any kind of instructions and are free to move the way they want. Though the captured image resolution is kept at 1920x1080, the dataset also provides full-HD video recordings. For labels, it provides 400 annotated frames from each camera, i.e., total of 2,800 frames, covering over 56,000 bounding boxes. These frames are synchronized ($\sim 50ms$ accuracy) across cameras. It covers a total of over 300 people walking and

Total Cameras	Resolution	Frames/Cam	Frame Rate	Bounding Boxes	Participants
7	1920x1080	400	60	56,000	313

Table 3.2: Details of WILDTRACK Dataset

standing. Figure 3.8 shows some of the sample images from this dataset. For this dissertation, I have utilized all 7 views for training/testing the various components of *CollabCam* .

(a) View 1

(b) View 3

(c) View 4

(d) View 5

Figure 3.8: Sample images from WILDTRACK dataset

3.5 Autonomous Spatial Overlap Area Estimator

CollabCam's approach for automatic spatial area estimation relies on the use of state-of-the-art techniques for two functions: (a) *object detection*, which is applied independently on each camera's image stream and provides the bounding box coordinates of objects visible in each image, followed by (b) *object re-identification* (ReID), which identifies/associates common objects that are simultaneously observed by two or more vision sensors. The basic idea is to identify the same object

across two different camera views simultaneously. Once identified, we can infer that the corresponding object coordinates, in each camera, are part of the overlapping region shared by the camera pair. While object detection models are fairly standard (e.g., SSD [74], YOLO [92]), object ReID algorithms typically look for common visual or motion features (e.g., [68]) to establish correspondence and often require custom enhancements to tackle deployment-specific artifacts.

For the specific case of *CollabCam* involving the detection and tracking of *humans*, object matching is equivalent to a person Re-Identification (ReID) problem. Instead of developing new ReID techniques, I utilize the machine learning based approach for pedestrian recognition proposed in [52], where each human object is converted to a 128-dimensional vector, with two images declared to be the same human object if the Euclidean distance between their vectors is less than an empirically-determined threshold T.

I shall now discuss two different approaches employed by *CollabCam* for person matching. The first approach, called Baseline, is an initial approach used to establish the baseline capability of *CollabCam* and simply uses an off-the-shelf idea of person matching. The second approach, called weighted approach, improvises on Baseline and modifies the matching logic to better satisfy the *CollabCam's* requirements.

3.5.1 Baseline Approach for Area Estimation:

This approach operates on each pair of cameras independently, and assumes that each camera provides a series of time-synchronized "training" images. The approach takes a pair of simultaneous images $Image_1$ and $Image_2$, from camera C_1 and C_2, and executes the following algorithm:

1. Let the initial overlap areas (for C_1 & C_2, respectively) be represented by $A_1 = \{\}$ and $A_2 = \{\}$.

2. Determine O_1 and O_2, the sets of bounding boxes (locations of human objects) in $Image_1$ & $Image_2$ respectively.

3. For each object $o_1^i \in O_1$ and $o_2^j \in O_2$, extract its image (using their bounding boxes) and compare them for equivalence, i.e., use [52] to verify if $o_1^i = o_2^j$.

4. For each matching object pair (i,j), i.e., if $o_1^i = o_2^j$, get their corresponding bounding boxes $BBox_1^i$ and $BBox_2^j$ respectively.

5. Perform a set-theoretic addition of $BBox_1^i$ and $BBox_2^j$ to $A_1 \& A_2$ i.e., $A_1 \leftarrow A_1 \cup BBox_1^i$, and $A_2 \leftarrow A_2 \cup BBox_2^j$.

6. Repeat the above process for the entire set of image pairs.

7. Finally, fit *minimum bounding rectangles* to the final overlap areas A_1 and A_2.

Figure 3.9 shows an example of steps involved in spatial area estimation; the bottom half of each image denotes the current estimation of the overlapping area. Here, (1) reflects the scenario when first object in camera 1 is successfully matched (re-ID) with another object in camera 2, whereby their locations, denoted by the red bounding boxes in their corresponding image frames, become part of estimated overlap area. As more objects are matched between the two images, the estimated overlap area grows as seen in (2), (3) & (4). Note that, as the image sensor eventually performs resolution adaptation for rectangular-shaped sub-regions, the estimator eventually provides a rectangular approximation of the common area.

3.5.2 Weighted Overlap Estimation

The baseline approach described above can lead to significant over-estimates: as *every matched* object causes the estimated overlap area to grow, any false positives in matching result in a spurious increase in the estimated overlap area. Such over-estimation can eventually lead to a use of lower-than-warranted imaging resolution and a consequent loss in object detection accuracy. The key issue with baseline approach is that, once a region of pixels is added to the overlap area, *it cannot be undone*.

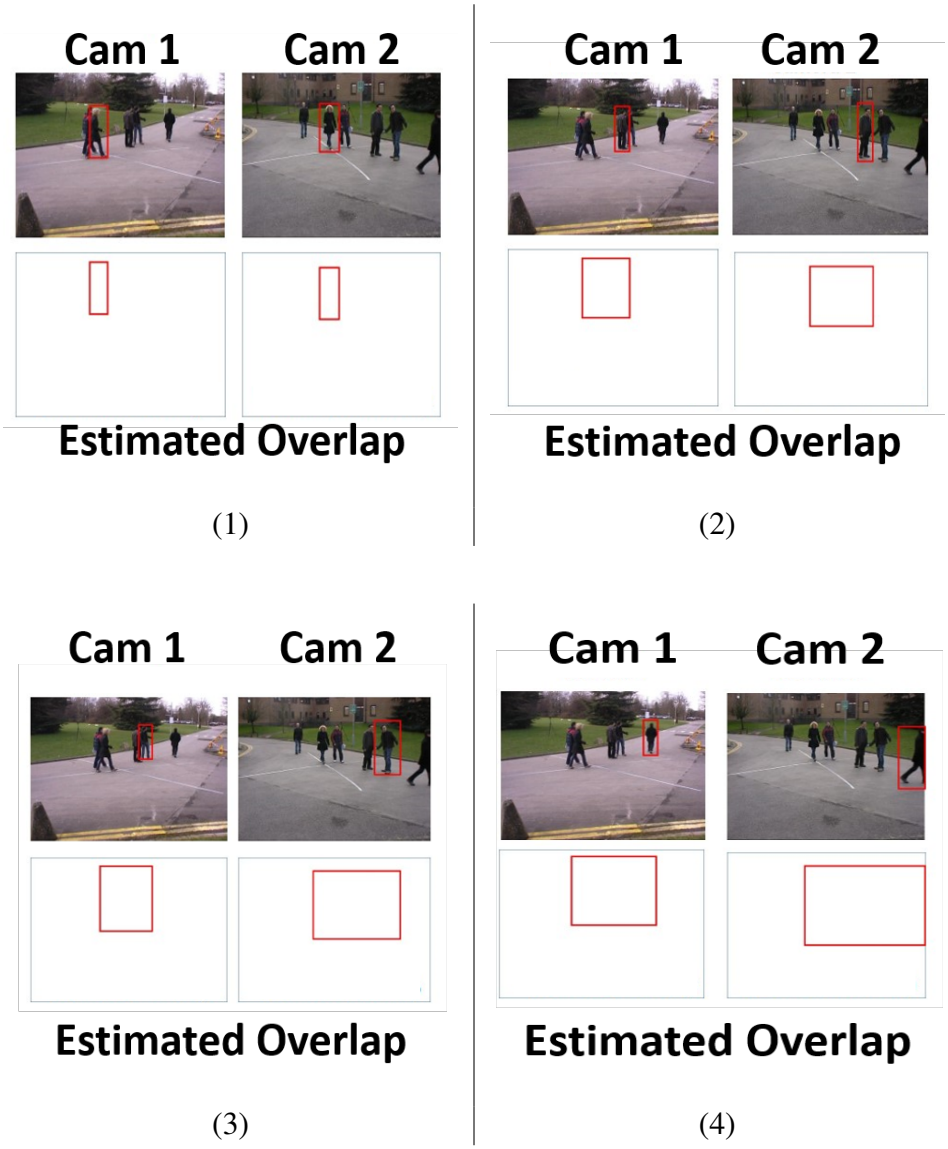


Figure 3.9: Steps in Spatial Overlap Area Estimation

To tackle this problem, *CollabCam* moves away from the binary use of 'overlapping' vs 'non-overlapping' labels to a weight-based softer segmentation strategy. In particular, for any pair of cameras, each pixel in a camera's FoV has an associated *weight*, initialized to 255 (maximum pixel intensity). For every object match/mismatch, the weights of pixels associated with the object decrease/increase. For a pair of simultaneous images $Image_1$ and $Image_2$, from camera C_1 and C_2, the weighted approach operates as following:

1. Let the initial overlap areas (for C_1 & C_2, respectively) be represented by $A_1 = \{\}$ and $A_2 = \{\}$. Let the initial weight for each pixel in FoV of C_1 and C_2 be

represented by matrixes $W^1_{m,n}$ and $W^2_{m,n}$ respectively, such that $W^1_{m..n}$ = $\{255\}$ and $W^2_{m..n}$ = $\{255\}$ initially.

2. Determine O_1 and O_2, the sets of bounding boxes (locations of human objects) in $Image_1$ & $Image_2$ respectively.

3. For each object $o^i_1 \in O_1$ and $o^j_2 \in O_2$, extract its image (using their bounding boxes) and compare them for equivalence, i.e., use [52] to verify if $o^i_1 = o^j_2$.

4. For each matching object pair (i,j), i.e., if $o^i_1 = o^j_2$, get their corresponding bounding boxes $BBox^i_1$ and $BBox^j_2$ respectively.

5. *Decrease* weight of all pixels in $BBox^i_1$ and $BBox^j_2$ by δ such that $W^1_{BBox^i_1}$ = $W^1_{BBox^i_1} - \delta$ and $W^2_{BBox^j_2}$ = $W^2_{BBox^j_2} - \delta$

6. For each non-matching object pair (i,j), i.e., if $o^i_1 \neq o^j_2$, get their corresponding bounding boxes $BBox^i_1$ and $BBox^j_2$ respectively.

7. *Increase* weight of all pixels in $BBox^i_1$ and $BBox^j_2$ by δ such that $W^1_{BBox^i_1}$ = $W^1_{BBox^i_1} + \delta$ and $W^2_{BBox^j_2}$ = $W^2_{BBox^j_2} + \delta$

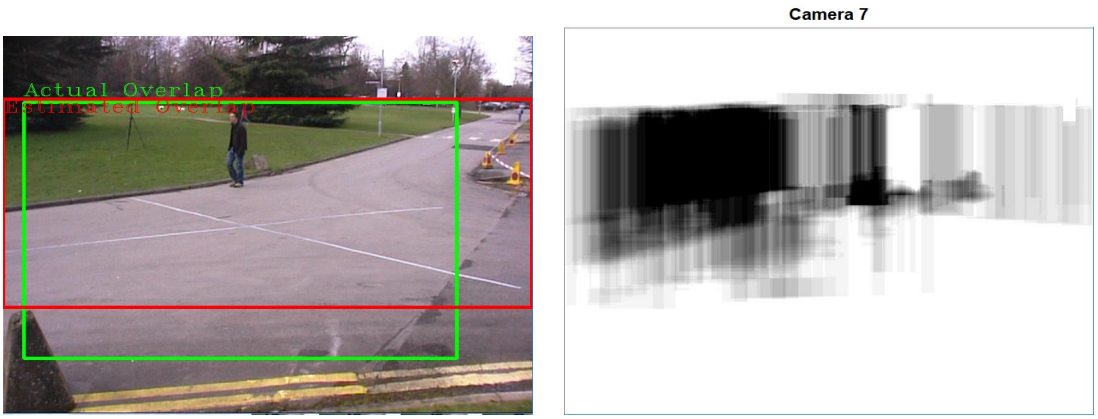
8. Repeat the above process for the entire set of image pairs.

9. Finally, select a maximum pixel weight threshold T_w and filter each pixel (a,b) from C_1 and C_2 such that $W^1_{[a,b]} \leq T_w$ and $W^2_{[a,b]} \leq T_w$.

10. Fit a *minimum bounding rectangle* to all the selected pixels to get the final overlap areas A_1 and A_2.

3.5.3 Performance of Overlap Estimator

I shall now evaluate the performance of two approaches, the baseline unweighted approach and the advanced weighted approach, using the benchmark PETS and WILDTRACK datasets. The goodness of overlap estimation is evaluated using the



(a) Non-weighted Approach (b) Weighted Approach

Figure 3.10: *PETS: Estimated vs Actual* overlap (camera 7)

Intersection over Union (IoU) score between the estimated overlap area and actual (ground truth) overlap areas, based on the formula

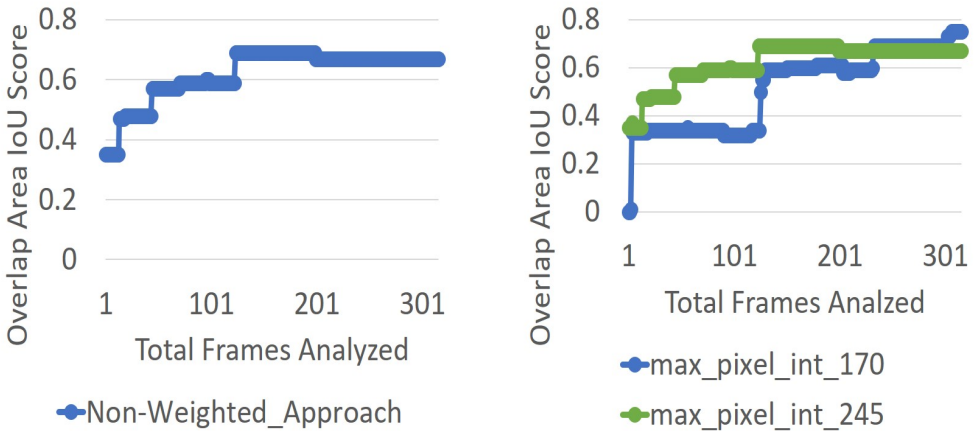
$$IoU = \frac{R_e \cap R_g}{R_e \cup R_g}$$

where R_e and R_g denote the estimated and ground truth regions respectively.

Performance on PETS:

This section presents results on arbitrarily chosen camera pairs (7, 8) and (5, 7) from PETS dataset. For both pairs, I assume former and latter cameras as the reference and collaborating cameras respectively. Figures 3.10(a) and 3.10(b) show the estimated overlap area (red box) vs the actual overlap area (green box), for the reference camera 7 (from cam pair (7, 8)), using the baseline (non-weighted) and weighted approaches, respectively. It can be seen that the weighted approach results in a closer match between the darker area and the actual overlapping region (the green box), unlike the non-weighted approach which includes a significant portion of *spurious* pixels.

IoU vs Number of Frames: For both methods, the resulting IoU depends on the number of frames used in 'training'; for the weighted approach, the IoU also depends on the threshold T_W used. Figures 3.11(a) and (b) show variation of IoU score vs total number of frames used for analysis, for non-weighted and weighted



(a) Non-Weighted (b) Weighted

Figure 3.11: PETS: Overlap Area IoU vs Total Analyzed Frames

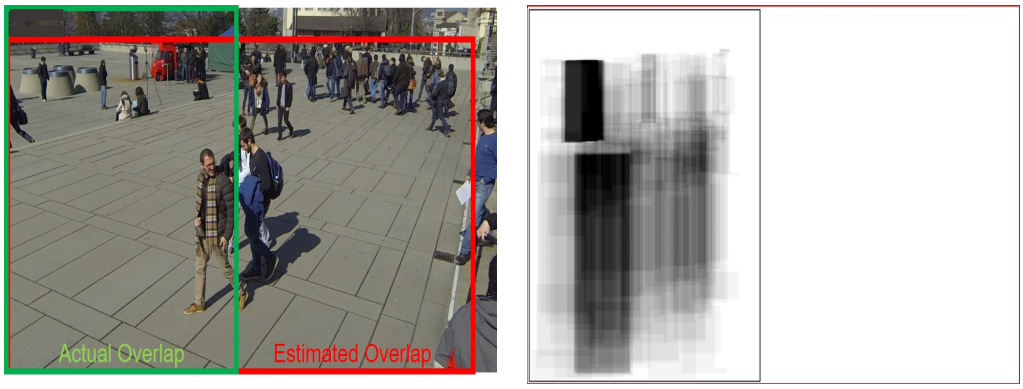
Camera Pair	Saturated IoU Score	
	Non-Weighted	Weighted
7, 8	0.67	0.76
5, 7	0.71	0.73
5, 8	0.64	0.68
5, 6	0.69	0.69
6, 8	0.57	0.56

Table 3.3: IoU variation with Total Frames Analyzed (PETS)

approaches respectively on camera pair (7, 8). It can be seen that the IoU score typically improves as the training set is increased till $\sim$ 200 frames, after which it saturates. In general, a denser dataset results in a larger number of potentially matching objects per frame and thus a more rapid convergence to the final estimated overlap area. Overall, the weighted approach results in a higher IoU (0.76 when $T_W = 170$), compared to an IoU=0.67 achieved by the non-weighted approach. Table 3.3 shows performance of both approaches on other camera pairs. It can be seen that the weighted approach performs better than the non-weighted approach for most of the camera pairs.

Performance on WILDTRACK:

This section discusses the performance of overlap estimator on two set of arbitrarily chosen camera pairs, (1, 4) and (4, 7), from WILDTRACK dataset. Figure 3.12

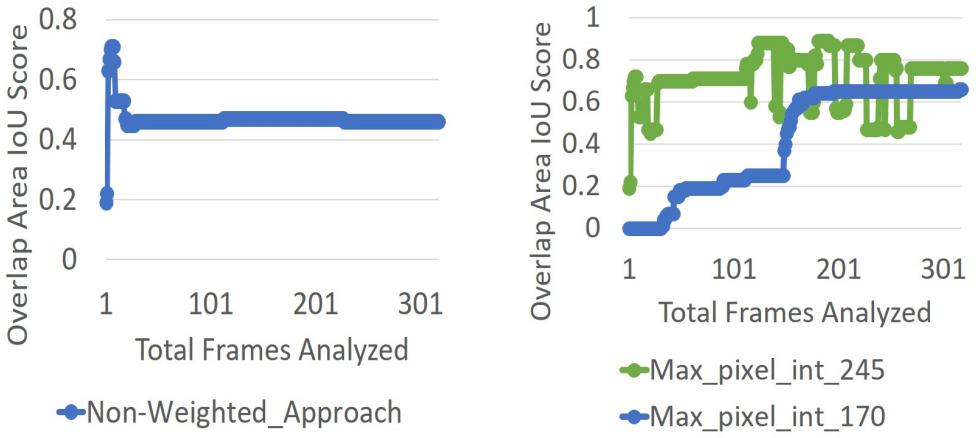


(a) Non-Weighted Approach (b) Weighted Approach

Figure 3.12: *WILDTRACK: Estimated vs Actual* overlap (cam 1)

shows the resulting estimated-overlap areas using both non-weighted & weighted approaches on reference camera 1 (from camera pair (1, 4)). After analyzing ~ 300 frames each from camera 1 & 4, *CollabCam* achieved *maximum* IoU score of 0.71 using non-weighted approach and 0.89 using weighted approach. It can be seen in Figure 3.12 that the weighted approach outperforms the non-weighted approach where the former *significantly* over-estimates the overlapping region. Compared to PETS, the over-estimation problem is more severe in the WILDTRACK dataset, where the higher object density and the higher fraction of distant humans generates a higher fraction of false positive matches.

IoU vs Number of Frames: Figure 3.13 shows variation of resulting IoU score vs the number of 'training' frames for camera pair (1, 4). Figure 3.13(a) shows the baseline non-weighted approach where the IoU score initially increases with an increase in the number of analyzed frames (reaching a peak value=0.71), but then begins to decrease, eventually saturating at ~ 0.46, due to the false positives which dramatically inflate the estimated overlap area. From Figure 3.13(b), it can be seen that, for the weighted approach, $T_W = 245$ (denoted as $Max\_pixel\_int\_245$) achieves maximum IoU score of 0.89 and saturates at 0.76 whereas $T_W = 170$ ($Max\_pixel\_int\_245$) saturates at score of 0.66. Table 3.4 shows performance of both proposed approaches on few other camera pairs. We observed that for (a) camera pairs (1, 4) & (4, 7), the weighted approach outperforms the non-weighted



(a) Non-Weighted Approach (b) Weighted Approach

Figure 3.13: WILDTRACK: Overlap Area IoU vs Total Analyzed Frames

Camera Pair	**Saturated IoU Score**	
	Non-Weighted	Weighted
1, 4	0.46	0.76
4, 7	0.25	0.5
1, 6	0.70	0.72
5, 1	0.6	0.64

Table 3.4: IoU variation with Total Frames Analyzed (WILDTRACK)

approach and (b) camera pairs (1, 6) & (5, 1), both approaches performed almost the same.

Key Takeaway: Overall, across both benchmark datasets, it can be seen that the weighted approach is (a) more accurate, achieving up to 13% and 3-100% higher IoU values, for PETS and WILDTRACK respectively and (b) more robust, being able to more closely approximate the true overlapping regions across a wider variety of operating conditions.

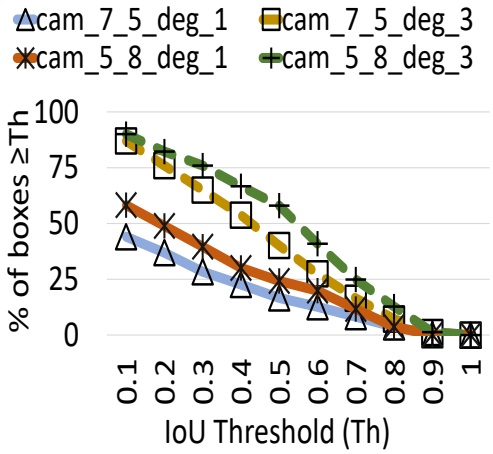
3.6 Cross-Sensor Coordinate Mapper

After the *mutually shared* region between a pair of cameras has been identified (Section 3.5), *CollabCam* will provide the locations of objects, detected within this region by a collaborating camera, as *priors* to the DNN performing object detection on the image provided by the reference camera. As each camera has a different

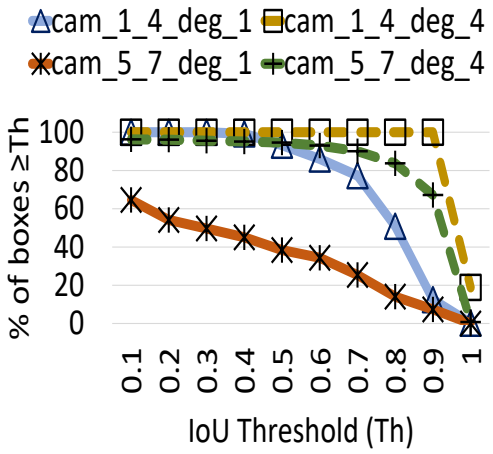
orientation and frame of reference, the provided coordinates of such objects must be transformed via a *mapping function* to the pixel coordinating space of the reference camera.

To provide such a mapping, *CollabCam* autonomously builds a regression model, which (a) takes {*bottom-mid point, height & width*} of each object's bounding box, relative to one camera coordinate system, as input and (b) estimates the corresponding {*bottom-mid point, height & width*} values in the other camera's camera coordinate system. The regression model itself is trained autonomously by: (a) taking the set of ReID-ed objects, simultaneously observed in both the camera frames and creating polynomial feature vectors from them, and (b) fitting regression models of various polynomial degrees and selecting the best fit model.

I use IoU score (and a minimal acceptance threshold Th), between the transformed & actual object coordinates, to experimentally determine the quality of different regression models. Figure 3.14 shows the resulting IoU scores for different camera pairs from our benchmark datasets. Higher order polynomials, which can accomodate non-linear mappings, perform better for both the datasets. For example, in PETS dataset, choosing an IoU threshold $Th = 0.5$, the degree=1 regression model produces only 17-24% bounding boxes with IoU$\geq Th$, whereas the degree=3 models produces 40-58% of such bounding boxes. Similarly, in WILDTRACK dataset, for $Th = 0.7$, the degree=1 regression model produce only 25-77% bounding boxes with IoU$\geq Th$, whereas degree=4 models produce 90-100% of such bounding boxes. In general, the WILDTRACK regressor is more accurate than PETS: this is likely due to the higher density of humans in WILDTRACK, which results in a larger set of training data points. Also, as discussed in [33, 87, 46, 34], the observed area in PETS has a pronounced slope, which introduces greater non-linearity in the mapping process.



(a) PETS



(b) WILDTRACK

Figure 3.14: Regression based Coordinate Mapping

3.7 Collab DNN & Mixed-Resolution Resolver

This section describes the collaborative object detection model, called *Collab-SSD* which is used at runtime to perform vision-based inferencing (at the Edge) using additional information from a collaborating peer camera. I also present micro-studies on benchmark datasets that help establish the superiority of *Collab-SSD* and that inform the resolution choices suggested by the Mixed-Resolution Resolver.

3.7.1 Collab-SSD

In general, it is well known that reducing the image resolution reduces the information content, thereby reducing the accuracy of DNN-based vision models used for

tasks such as object detection. To overcome this accuracy loss, *CollabCam* employs a collaborative object detection technique where the DNN has access to the additional information about the location of objects, within mutually shared regions, detected from images supplied by the collaborative camera(s). The underlying hypothesis is that this additional information, provided as *priors*, enables such a DNN to overcome the accuracy drop caused the lower resolution content within the shared regions of the image.

More specifically, as an exemplar, *CollabCam* employs a collaborative object detector DNN that enhances the popular, state-of-the-art Single Shot Detector (SSD [74]) model often used on pervasive devices. In this approach, the *prior* information is represented by a grayscale image (illustrated in Figure 3.5), supplied by the collaborating camera/DNN after suitable spatial transformation (Section 3.6), where the pixels representing an object are highlighted as White while background regions are colored Gray. The enhanced *Collab-SSD* utilizes a 4-channel input, where 3 channels correspond to the primary RGB image, while the 4^{th} channel corresponds to the gray-scale image provided by the collaborator.

Model Training: To create the augmented training data ($train_{aug}$) for Collab-SSD, I (a) selected a subset of training images ($train_{org}$) from both of our benchmark datasets, (b) for each input image, created random shared region of various sizes, aspect ratios and resolutions, and (c) created priors by randomizing ground truth object locations (within such shared regions). By introducing randomization in ground truth labels, I simulated the scenario where priors are provided by the collaborating camera(s) and are subject to real-world coordinate mapping errors. For training the Collab-SSD model, I took a Keras-based implementation of SSD512 and re-trained it on $train_{aug}$. I trained the Collab DNN model by modifying the off-the-shelf Keras-based implementation of SSD512[17] and re-training it on $train_{aug}$, using a standard server-class machine (8 core i7-7700 CPU@3.60GHz, 64 GBR RAM and 2 Nvidia GTX 1080Ti GPUs with 12GB memory).

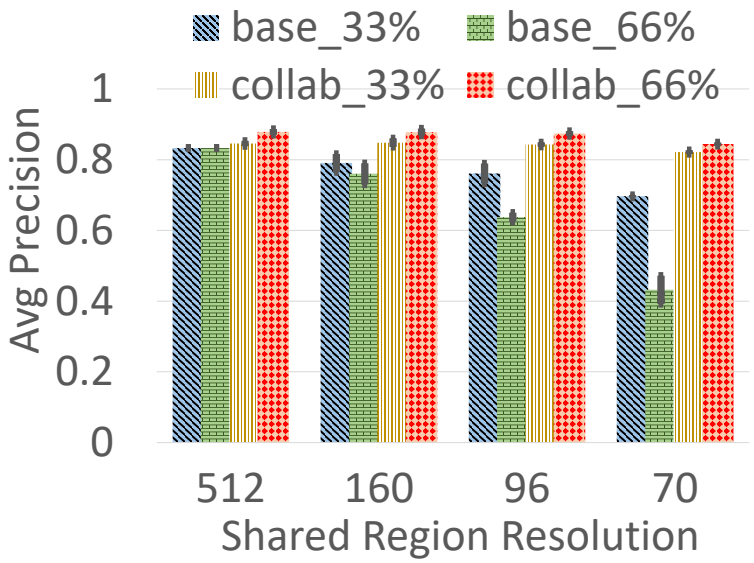
3.7.2 Performance of *Collab-SSD*

This section describes micro-studies to evaluate the performance benefits of Collab-SSD, compared to a baseline SSD that operates solely on the reference camera's image. To perform such studies, I create a hypothetical, randomly generated set of shared regions of different $\{$*size, location, resolution*$\}$. For one arbitrarily selected camera each from PETS & WILDTRACK, I varied the location (L_{sr}), size (S_{sr}) & resolution (R_{sr}) of the shared region such that:

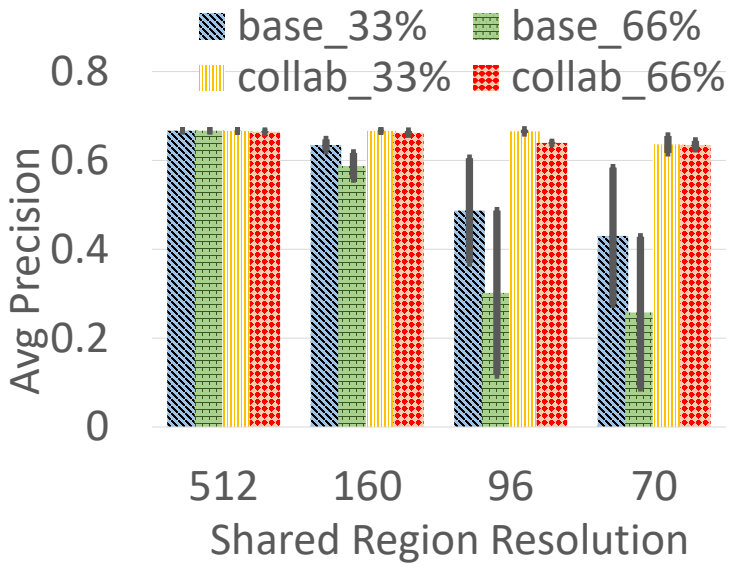
$$S_{sr} \in \{33\%, 66\%\},$$
$$P_{sr} \in \{leftmost, rightmost\} \text{ of image,}$$
$$R_{sr} \in \{512\text{x}512, 160\text{x}160, 96\text{x}96, 70\text{x}70\}$$

For each test image, the prior was initially created using its ground truth labels (only within the shared region), and then perturbed as follows: for each combination of overlapping region percentage and image resolution (S_{sr},R_{sr}), I (a) iterated through each value of P_{sr} and computed the resulting object detection accuracy, and then (b) computed avg. and std. dev. of such accuracy values. To quantify performance of the baseline DNN (SSD), I utilized the above created mixed-resolution images but without the additional (4^{th} channel), i.e., without the corresponding priors generated by perturbations of the ground truth.

Performance on PETS: Figure 3.15a shows the resulting performance on PETS dataset. It can be seen that the average precision score (mAP) of the baseline (non-collaborative) SSD model reduces rapidly as the pixel resolution of the shared region decreases. For example, when resolution is reduced from 512x512 to 70x70, the accuracy of baseline model for $S_{sr} = \{33\%, 66\%\}$, denoted by base_33% & base_66%, drops to 0.70 (16% reduction) and 0.43 (48% reduction) respectively. Clearly, the larger the shared region, the greater the impact of the resolution reduction, leading to a larger drop in object detection accuracy. In contrast, collab-SSD maintains high accuracy over a wider range of resolution reduction and increased S_{sr}– as the image resolution reduces from 512x512 to 70x70, the accuracy



(a) PETS



(b) WILDTRACK

Figure 3.15: mAP vs resolution for Collab-SSD & SSD

of Collab-SSD for $S_{sr} = \{33\%, 66\%\}$, denoted by collab_33% & collab_66% shows only a modest loss in the mAP score, to 0.82 (2.7% reduction) and 0.84 (4.5% reduction) respectively.

Performance on WILDTRACK: Figure 3.15b shows the corresponding results on the WILDTRACK dataset. Similar to PETS, Collab-SSD outperforms the non-collaborative SSD baseline in all cases. For example, when resolution is reduced from 512x512 to 70x70, the accuracy of SSD, for 33% & 66% shared regions,

drops to 0.43 (36% reduction) and 0.258 (61% reduction) respectively. In contrast, the reduction in accuracy (mAP score) for Collab-SSD is less drastic, drops to 0.64 (4.4% reduction) and 0.63 (4.5% reduction) for S_{sr}= 33% & 66% shared regions, respectively.

Key Takeaway: Overall, it can be seen that, by taking advantage of priors provided by a collaborating camera, the proposed collaborative DNN model (Collab-SSD) is able to handle the loss in pixel resolution much better than classical non-collaborative DNNs. The performance degradation does depend on the object density of the monitored environment: interestingly, a lower resolution causes a lower mAP score for WILDTRACK (higher human density) than PETS (e.g., 0.63 vs 0.84 for (R_{sr} =70x70, $S_{sr} = 66\%$)). These findings open up the possibility of significant energy savings for *CollabCam*, as discussed in following sections.

3.7.3 Mixed-Resolution Resolver

The Mixed-Resolution Resolver determines the optimal acceptable sensing resolution, for mutually shared regions among a pair of cameras, that best balances energy savings and object detection accuracy. While this resolution choice is DNN model-dependent, this is computed empirically, for a *given DNN model*, by (a) observing the DNN accuracy over the *shared region* of an image at varying levels of pixel resolution, and (b) selecting the smallest value where Collab-SSD's performance is reasonably close to that of SSD (at highest resolution). Once pre-computed, this set of resolution choices is stored in the Edge device and communicated to the set of associated *CollabCam* cameras.

3.8 *CollabCam* System Prototype[1]

While *Collab-SSD* shows how to sustain high DNN accuracy with lower-resolution images of overlapping areas, *CollabCam* also needs appropriate sensing hardware to realize the resulting energy advantages. To demonstrate such capabilities, we developed two different platforms for MRFS-based vision sensing:

- We initially utilized an open source camera hardware CMUCam-5 (Pixy2 platform) [27], specifically designed for low-power, pervasive sensing. The sensing board includes an Aptina MT9M114 CMOS Image Sensor, with native resolution of 1296x976, integrated with an image flow processor controlling its operation. The Pixy2 platform also includes an NXP LPC4330 dual-core ARM processor clocked at 204MHz, 264 KB SRAM for storing dynamic variables/buffers and 2MB Flash memory for code storage, as well as multiple data communication interfaces.

- As an additional vision sensor, we used Raspberry Pi (RPi) camera module v2 mounted on Pi 4B board [44]. The module v2 consists of a Sony IMX219 CMOS Image sensor with a native resolution of 3280 × 2464 pixels. The Pi 4B board has BCM2711, a quad-core 64-bit ARM-Cortex processor A72 from Broadcom clocked at 1.5GHz with 4GB of LPDDR4 SDRAM. Pi 4B supports CSI interface for connecting the camera sensor, while wireless connectivity is provided by on-board 802.11ac radio and Bluetooth 5.0 chips.

3.8.1 Pixy2 MRFS Camera

To support MRFS-like functionality, we modified both the open-source firmware (running on the ARM processor) and the software on the host machine (which logs the images transferred from the Pixy2 board). By default, the ARM processor on

[1]Developed as a joint effort, led primarily by Vithurson Subasharan & Dr. Manoj Gulati. My contribution is in designing the overall experiment plan and help write power-data collection scripts. This setup is included in my dissertation for end-to-end description of *CollabCam*.

Pixy-2 is clocked with a 12MHz external crystal oscillator, which is frequency-scaled 17x times by an internal PLL to generate a "System Clock" of 204 MHz that drives the ARM processor. The ARM processor firmware configures the CMOS sensor's image flow processor, which internally captures an individual frame, which is stored in the internal SRAM before being fetched by the host machine.

MRFS requires the specification of different regions and distinct per-region sampling resolutions, within a *single* image frame. However, the CMOS image flow processor's physical registers permit concurrent specification of only a single windowing (sub-region) and binning (sub-sampling) operation. Moreover, the SRAM's extremely limited capacity (264 KB) allows us to capture a maximum of only 300x214 pixels in one frame acquisition cycle. Accordingly, we modified the firmware to support a "virtual MRFS" as follows: assuming our desire to acquire an image with K sub-regions, we make K consecutive calls to the sensor, specifying the resolution and area of the i^{th} ($i = \{1, \ldots, K\}$) sub-region, via modifications to the internal registers to control the height/width of the camera output and the region-of-interest (ROI). In effect, we take K distinct images, each corresponding to one region, and then "stitch" them together to create a composite $K - region$ mixed-resolution image. As a result: (a) the overall frame rate now is only $\frac{1}{K}^{th}$ of the maximum sensor frame rate–given our focus on demonstrating *CollabCam's* energy-efficiency, this is not a major limitation; (b) the overall sensing process now incurs the energy and latency overhead of K distinct sensor reconfiguration steps per MRFS image. Indeed, the overall Pixy2 image acquisition process involves 4 distinct delays:

(a) *Reconfiguration Delay*: the time spent on resetting the internal registers on the image sensor

(b) *Capture Start Delay*: the transient period needed by the sensor to activate the current register configurations and acquire a sub-frame

(c) *Frame Capture Delay (or Frame Time)*: the time taken to fetch the entire

image from the image flow processor and write it to the ARM processor's internal SRAM

(d) *Communication Delay*: the time needed to transmit the SRAM's image content to a remote host

By separately quantifying these individual delay and energy overheads, we shall shortly show how we can compute the total energy overhead of an *idealized* MRFS sensing process, which consists of only steps (b)-(d). In addition, we shall see later that non-real time vision applications (e.g., when used with low-power periodic sensing MAC protocols [115]) may require such captured images to also be temporarily stored on the Pixy2 board's *external storage*.

3.8.2 Raspberry Pi (RPi) MRFS Camera

Although the official PiCamera[2] library provides the ability to capture images at various resolutions and sizes, it does not support *mixed-resolution* capture of a single frame. Moreover, the default Sony IMX219 driver provides only a small set of binning options. As a result, we cannot reduce the sensing resolution of an image below 640x480; any lower resolution is achieved in software, via downsampling by the CPU.

To mimic MRFS functionality on RPi, we modified the driver code[62] for the RPi camera and used our custom code for interactions with the modified camera driver. More specifically, we did following enhancements (a) modified the internal registers of driver to enable capture of *native* images of any arbitrary size & location within the original FoV of the camera – this allows creation of a mixed-resolution image with N sub-regions by capturing & stitching N images of different sizes & locations and (b) added more *camera-modes* to enable 8x8 binning (maximum supported by the camera sensor) by combining binning & row/column skipping, allowing a maximum 64x decrease in the resolution of each sub-region.

[2] `https://picamera.readthedocs.io/en/release-1.13/`

3.8.3 Power Measurement Setup

To examine drop in power and corresponding energy consumption for both Pixy2 & RPi, we used Monsoon Power Monitor (FTA22D) sampling at 5kHz and current sensitivity of 60uA[3]. Monsoon monitor can provide a regulated output voltage in the range of 2.1-4.55V for intrusive power measurement using a built-in, fixed value current resistor of 0.056Ω. Furthermore, it allows USB pass-through power measurements for USB enabled devices. We customized data cables to interrupt the VCC and GND wires and monitored power consumed by both Pixy2 and RPi both.

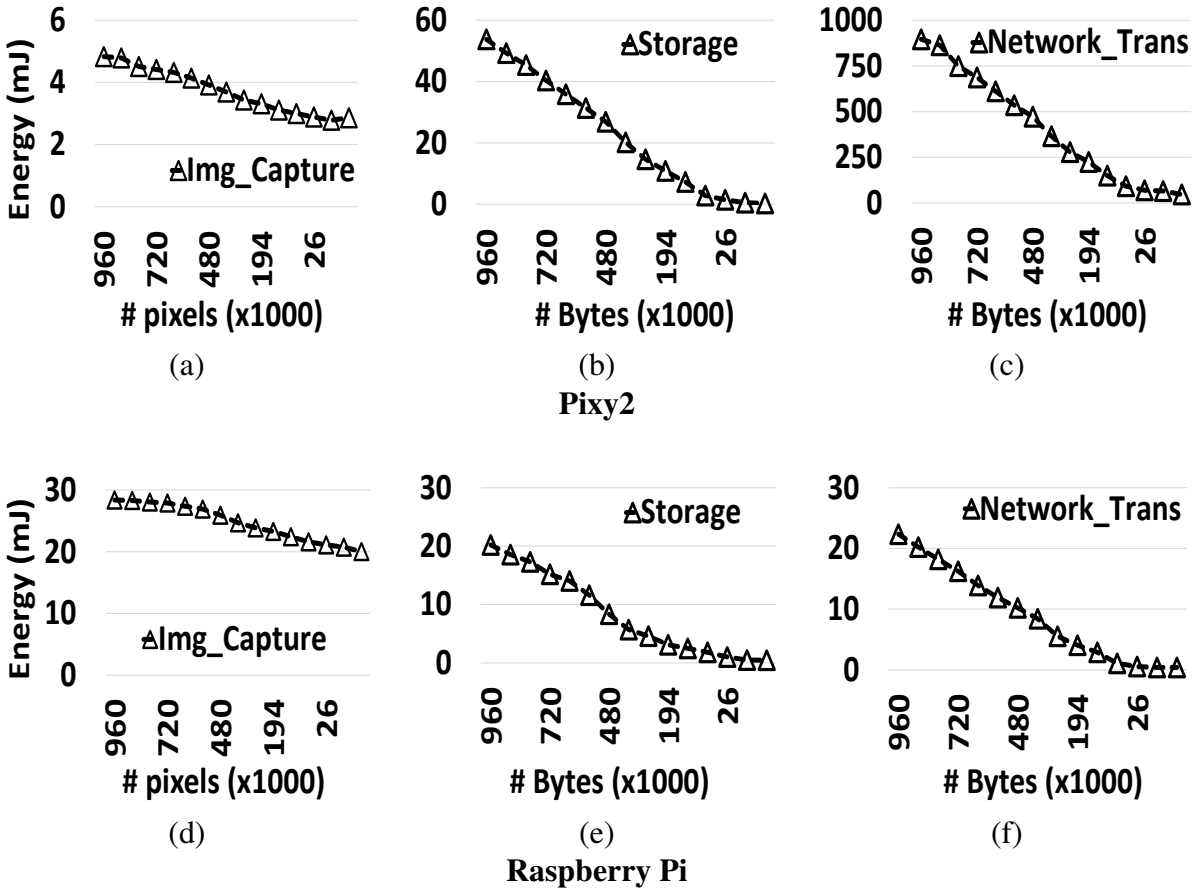


Figure 3.16: Energy Consumption vs Amount of Data Processed

[3]http://msoon.github.io/powermonitor/PowerTool/doc/LVPM%20Manual.pdf

3.8.4 Energy Consumption Profile

We separately monitored the energy consumed by each camera sensor during three distinct steps: (i) image capture; (ii) image storage and (iii) network transmission. To accurately measure the relationship between image resolution (pixel count) and the corresponding energy overhead, we (a) quantify the average idle-state (no images captured) energy consumption (E_I) of Pixy/RPi platform, (b) for each resolution value $r \in R$, we use ≥ 50 images to measure the average *active-energy* consumption (E_A^r); (c) compute the difference $E_T^r = E_A^r - E_I$ to obtain the true energy overhead E_T^r, and (d) fit a regression model to the observed values of E_T^r. Size of shared region and its sensing resolution can vary from one camera pair to another, therefore varying the total number of pixels processed *eventually*. So, I used a regression model to map the *eventual* pixel count to its corresponding energy consumption rather than empirically measuring the energy consumption for each pixel count. Figure 3.16 visualizes the observed relation between the different components of energy consumption and the total number of pixels captured.

Image Capture Energy

The energy for image capture refers purely to the process of reading sensor pixel values followed by their A-D conversion, and excludes any subsequent steps involving data storage or transmission. Figures 3.16(a), 3.16(d) plot the observed relationship as the captured image resolution is varied (ranging from 1200x800 to 70x70); detail power plots, which reveal the transient spikes in the underlying hardware, are omitted due to space limits. We note that, while the capture energy decreases, as expected, with reduced pixel count, the decrease is not *energy-proportional* (for reasons detailed in [71]). We utilize this data to create linear (degree=1) regression models (5-fold R^2=0.987 for Pixy2, R^2=0.937 for RPi) mapping the number of pixels captured to the energy consumption of the camera sensor.

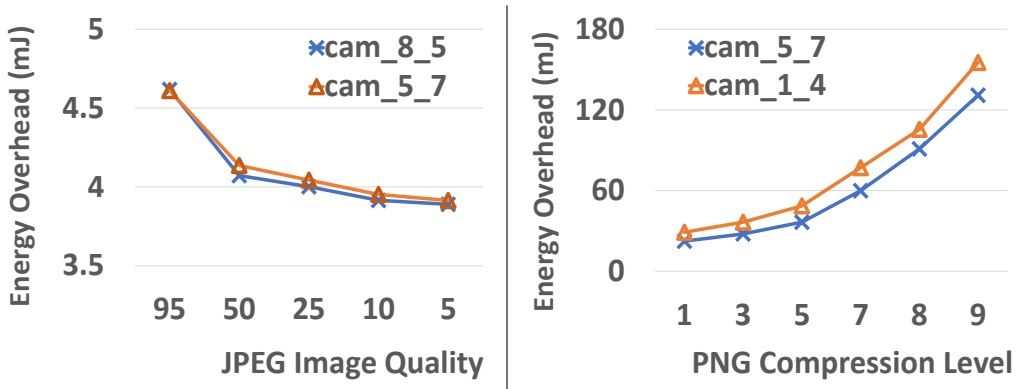
Image Storage Energy

In several vision sensing systems, the captured images may need to be temporarily stored (on external storage) on the camera sensor, prior to transmission to the Edge device. Such a need may arise, for example, in a networked camera setup, where low-power MAC protocols (e.g., [115]) may be used to schedule each camera's transmission slot. To quantify such overheads, we compute the energy consumption when the *already captured* image data is written to (a) the external Flash storage for the Pixy2 platform, and (b) SD card for Raspberry Pi platform. Figures 3.16(b), 3.16(e) plot relationship between energy consumption in storage and number of bytes written–it is important to observe that *storage energy overhead is significant and, in fact, can be ∼3-10 times higher than the image capture overhead.* For Pixy2, we found that image data is written in "chunks" of 256 bytes, with each chuck consuming $\sim$ 15μJ when supplied with a voltage of 4.55v. We use this relationship to directly compute energy consumption for any amount of data to be written in storage. For RPi, we found that data writes are quicker and consume less power compared to Pixy2. We use such data to create a linear regression model (degree=1, 5-fold R^2=0.980) for both prototypes, mapping number of bytes written in memory to the corresponding storage energy consumption.

Network Transmission Energy

To provide wireless connectivity to Pixy2, we can connect an ESP32 [4] board on its UART port, and differentially measure the energy consumed by the ESP32 alone while wirelessly transmitting data (using sockets) of various sizes to a server running on another machine. For RPi, we utilized the in-built 802.11ac WiFi chip for transmitting data to an MQTT server running on a separate machine, and measured the differential energy consumption of whole RPi board while transmitting data of various sizes. Figures 3.16(c), 3.16(f) plot our measured results–we see

[4] `https://www.espressif.com/en/products/devkits/esp32-devkitc/overview` last accessed: 25/10/2020



(a) PETS (b) WILDTRACK

Figure 3.17: Compression Level vs Energy Overhead for benchmark datasets

that network transmission energy is decreasing with a decrease in number of bytes transmitted over the network. We use this data to create linear regression models (degree=1, 5-fold R^2=0.993 for Pixy2, R^2=0.992 for RPi) for mapping number of bytes transmitted to corresponding energy consumption.

Image Compression Energy

Before transmitting data over wireless network, it might be useful to compress the data first and therefore save some of the network transmission energy. The captured mixed-resolution image can be compressed using various JPEG/PNG encoding schemes and each such scheme will impose additional energy overhead for running the compression algorithm. To measure the energy overhead I (a) arbitrarily selected two camera pairs from each dataset and replaced their shared-regions with white-padding (b) compressed their corresponding *non-shared* regions with different compression levels (c) measured computation energy overhead, averaged over 100+ iterations, for each compression level. Due to resource constraints, the compression schemes were only tested on Raspberry Pi platform and not Pixy2.

- *Compression on PETS Dataset:* Since PETS images are jpg, we used JPEG encoding (lossy) for compression. We varied the image quality from 95 (lowest compression) to 5 (highest compression) and measured the energy overhead for each quality value. Figure 3.17(a) shows the resulting energy

overhead of compressing an image from PETS dataset. It can be seen that maintaining higher quality image requires more energy whereas lower quality images can be produced more efficiently (because of lossy nature of JPEG encoding).

- *Compression on WILDTRACK Dataset:* This dataset has png images, so we used PNG encoding (loseless) for compression. The compression level is varied from 1 (lowest compression) to 9 (highest compression) and average energy overhead is measured for each level. Figure 3.17(b) shows the resulting energy overhead. It can be seen that increasing the compression level also increases the energy overhead. This is because PNG encoding is loseless, so higher compression levels attempt to encode information more compactly, therefore resulting in high computation costs.

We use this data, together with energy consumption data for transmitting compressed images over the network, to derive total reduction in network transmission energy when images from our benchmark datasets are compressed and transmitted over the network.

Overall, these results show that *CollabCam's* DNN models, which permit lower resolution sensing of overlapping areas, enable substantial overall energy savings/frame.

3.9 *CollabCam* Performance in Real-World Deployments

Several aspects of *CollabCam's* performance (e.g., the Overlap Area Estimation accuracy, Collab-SSD mAP) are dependent on deployment-specific features such as (i) the actual overlap between cameras and (ii) object density in various regions of a camera's FoV. I now utilize the PETS & WILDTRACK deployments to provide

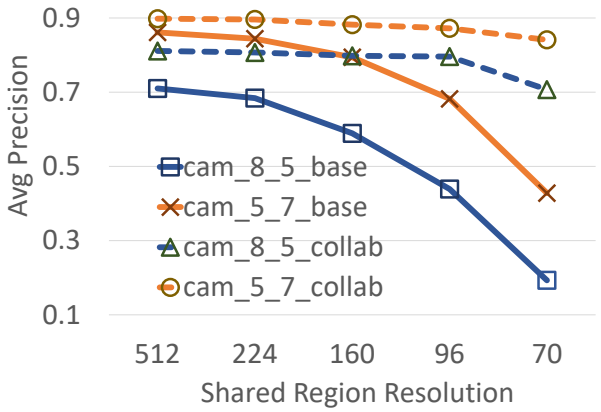
an *emulation-based* (i.e., assuming the use of *CollabCam*) quantification of *CollabCam's* overall performance.

3.9.1 Study S_1: Idealized Collaboration

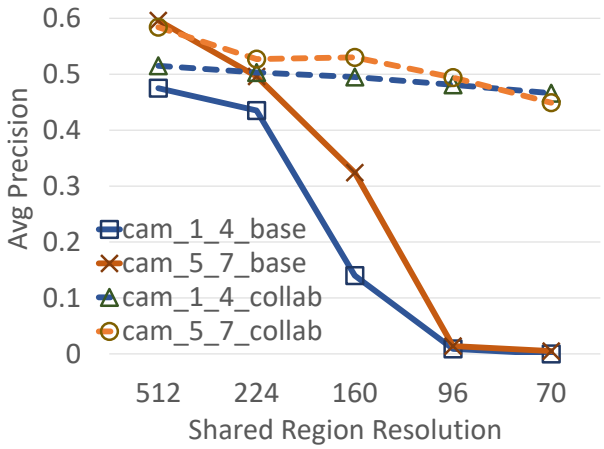
This section explores the deployment-specific benefits of *CollabCam*, in terms of improved accuracy vs resolution tradeoff, over the mutually-shared region, assuming an *idealized collaboration model*. Such a model assumes the availability of perfect priors (no errors in overlap estimation, the bounding boxes of objects detected by the collaborator or the subsequent coordinate mapper) and provides an *upper bound* to the benefits of *CollabCam*. For a set of N test images for a given (ref, collab) camera pair, I modify the reference camera images by (a) extracting the portion of the reference camera's image corresponding to the overlapping region, (b) re-sampling the extracted image to the required resolution value (R_{sr}), and (c) applying black-padding to re-scale this re-sampled image into a 512x512 SSD-compliant image. We then provide this re-scaled image, together with the ground truth coordinates of objects as a prior (4^{th} channel), to the *Collab-SSD* model.

Performance on PETS: Figure 3.18a shows the resulting performance on two arbitrarily selected camera pairs 8, 5 (49% overlap) and 5,7 (53% overlap). Clearly, the performance of the baseline, non-collaborative, SSD model drops dramatically with lower resolution (50-73% lower mAP score when the resolution reduces from 512x512 to 70x70). *CollabCam*, however, is able to maintain significantly higher accuracy at lower resolutions, suffering only a 6-13% drop in mAP scores for an identical reduction in resolution.

Performance on WILDTRACK: Figure 3.18b shows the resulting performance on two arbitrarily selected camera pairs 1,4 (47% overlap) and 5,7 (63% overlap). Once again, *CollabCam* outperforms the baseline model, e.g., reducing the resolution of both cameras from 512x512 to 70x70 results in $\sim 99\%$ drop in accuracy for the baseline model, but only 9-23% drop in accuracy for *CollabCam* .



(a) PETS



(b) WILDTRACK

Figure 3.18: Study S_1: Idealized Collab-SSD Performance

Overall, it can be seen that within the shared regions of images across *both* deployments, *Collab-SSD* is able to tolerate a loss of resolution far better than the baseline, non-collaborative model.

3.9.2 Study S_2: Real-world "Noisy" Collaboration

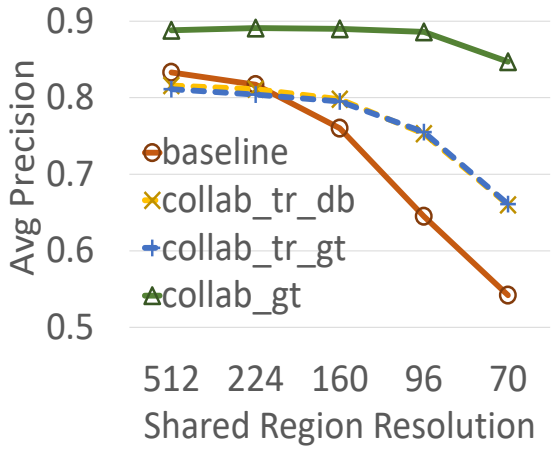
Given the reality of inevitable errors in the intermediate steps of object detection and cross-camera coordinate mapping, I now describe *CollabCam's* performance under various degrees of errors in the underlying priors (an input to *Collab-SSD*. More specifically, for a given (ref, collab) camera pair, I created priors in 3 different ways:

1. *collab_gt*: using ground truth labels corresponding to each reference camera image.

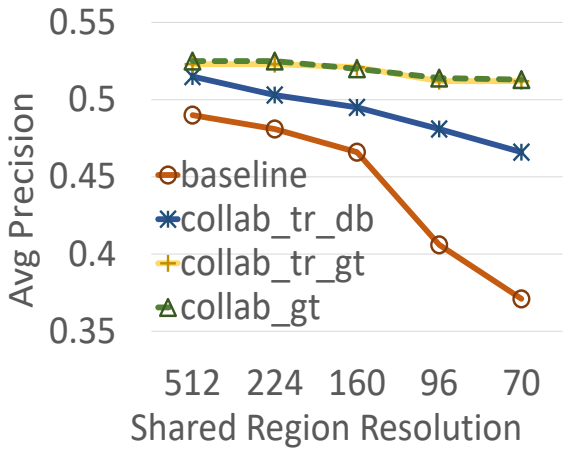
2. *collab_tr_gt*: applying regression-based coordinate mapping (Section 3.6) over the ground truth labels of the collaborator's captured image.

3. *collab_tr_db*: applying regression-based coordinate mapping over the actual bounding boxes detected by applying SSD on the collaborator's image.

While *collab_tr_gt* is subject to spatial mapping errors, *collab_tr_db* mimics a real-world operation where each intermediate step is potentially noisy.



(a) PETS (Cam 8,5)



(b) WILDTRACK (Cam 1,4)

Figure 3.19: Study S_2: Collab-SSD Performance with Noise

Performance on PETS: Figure 3.19a shows the resulting performance for an ar-

bitrarily chosen camera pair 8,5 (overlap 49%). The accuracy (mAP score) of baseline, non-collaborative SSD reduces to 0.542 (35% drop) as the image resolution is reduced from 512x512 to 70x70. However, with *collab-SSD*, the accuracy drops very modestly to 0.85 (4.5% drop) for *collab_gt*, to 0.66 (18% drop) with *collab_tr_gt* and to 0.66 (19% drop) with *collab_tr_db*. Also, the almost-identical performance of *collab_tr_gt* & *collab_tr_db* shows that the loss of accuracy with *Collab-SSD* is primarily *due to the errors in coordinate mapping* and can be overcome via the use of higher-precision, advanced coordinate remapping mechanisms(e.g., Tsai Calibration[40]).

Performance on WILDTRACK: Figure 3.19b shows the resulting performance for an arbitrarily chosen camera pair 1,4(overlap 47%). All three variants of *Collab-SSD* outperform the non-collaborative baseline: decreasing resolution from 512x512 to 70x70 reduces the baseline model accuracy to 0.371 (23% drop), compared to 0.513 (2.2% drop), 0.512 (2.3% drop) and 0.466 (9.5% drop) for *collab_gt*, *collab_tr_gt* & *collab_tr_db*, respectively. *collab_gt* and *collab_tr_gt* perform similarly, due to the higher coordinate mapping accuracy on WILDTRACK dataset (Figure 3.14).

3.9.3 Study S_3: Compressed Images vs Object Detection Accuracy

In real-world deployments of energy constrained vision sensors, it might be useful to compress images before transmitting them over the network. But, the compression might result in reduced object detection accuracy of resulting images. So, I now describe *CollabCam's* performance on images with various degrees of compression. For test images selected from two arbitrarily chosen camera pairs, I estimated object detection accuracy by (a) replacing the shared region of the images with white-padding (not compressing shared region because it is already being captured at lowest possible resolution) (b) encoding the resulting image using JPEG/PNG

encoding scheme of various compression levels (c) decoding the compressed image and measuring object detection accuracy of base-SSD and Collab-SSD models on the decoded image. For all these experiments, the shared region is sampled at 70x70 resolution.

Performance on PETS: I used JPEG encoding, a lossy algorithm, for PETS dataset images. Figure 3.21 shows the % drop in object detection accuracy when compressed images of varying quality (higher quality means lower compression and vice-versa) are used for object detection. It can be seen that higher compression levels lead to more drop in accuracy. For example, when image quality is reduced from 95 to 5, the baseline and collaborative models for (a) camera pair (8, 5), exhibit 33% & 12% accuracy drop respectively and (b) camera pair (5, 7) exhibit 19% & 0.7% drop respectively.

To explain this accuracy drop, figure 3.20 shows images of various qualities (achieved by varying compression ratio) and their corresponding objects, as detected by our base-SSD model. It can be seen that fewer objects are detected in lower quality (high compression ratio) images as compared to higher quality (low compression ratio) images. This is because of JPEG encoding being lossy algorithm which results in increased noise and greater loss of high-frequency spatial features when image compression ratio is increased. We observed that the drop in accuracy is primarily because of false-negative cases i.e., when base-SSD model failed to detect an object present in the image (therefore dropping recall score). These false-negatives were majorly observed for objects that are either far from the camera (thus look small in size) or are occluded in some manner.

Overall, the Collab-SSD model outperforms the base-SSD model for both the camera pairs. We can say that *CollabCam* allows us to compress non-shared region of an image without significantly dropping the object detection accuracy. More specifically, for (a) camera pair (8, 5), the image quality of non-shared region can be dropped to 50 with a minimal 0.5% drop in accuracy (b) camera pair (5, 7), the image quality can be dropped all the way to 5 with a minimal 0.7% drop in accuracy.

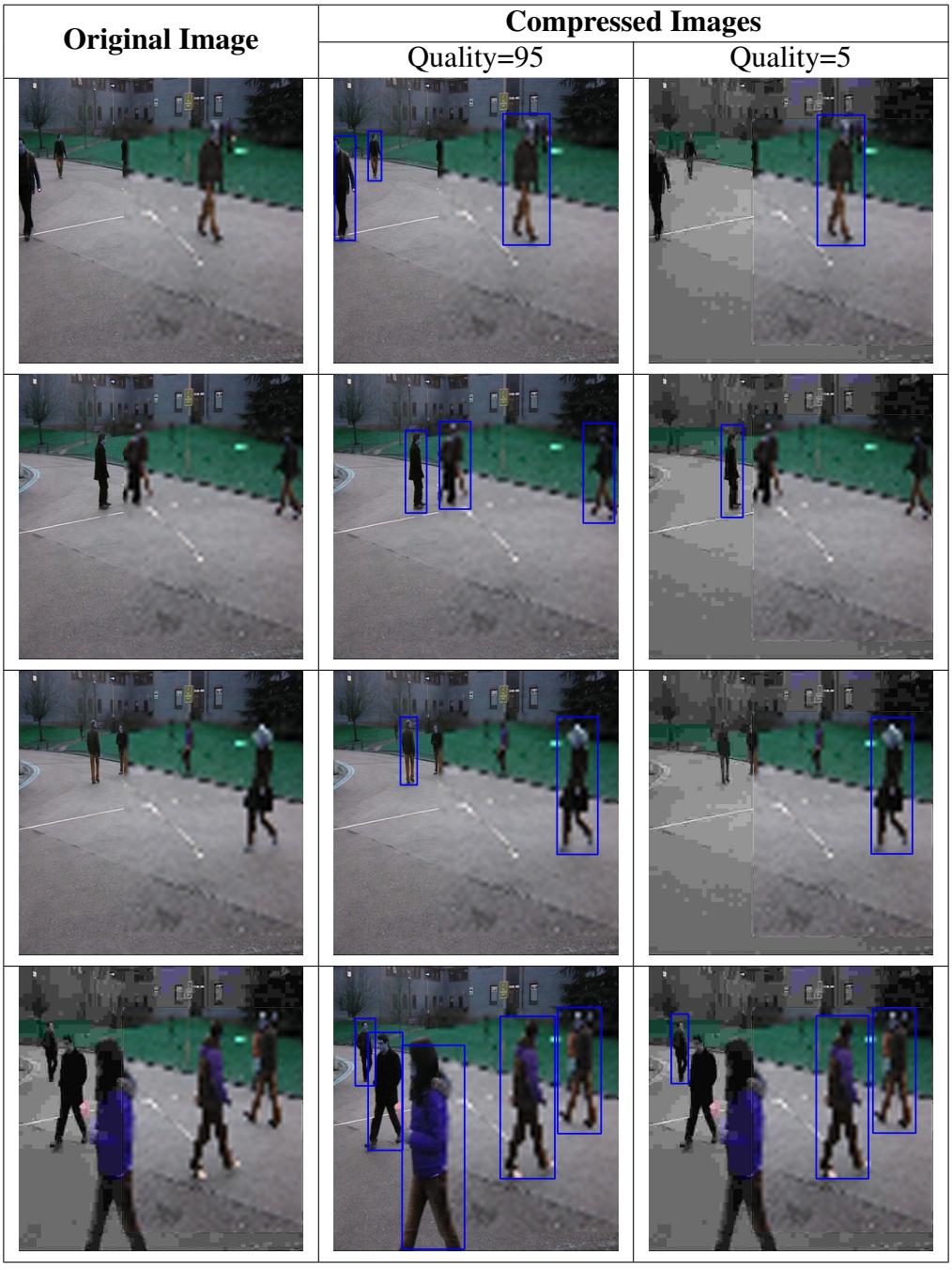


Figure 3.20: Object detection performance vs image quality (compression ratio)

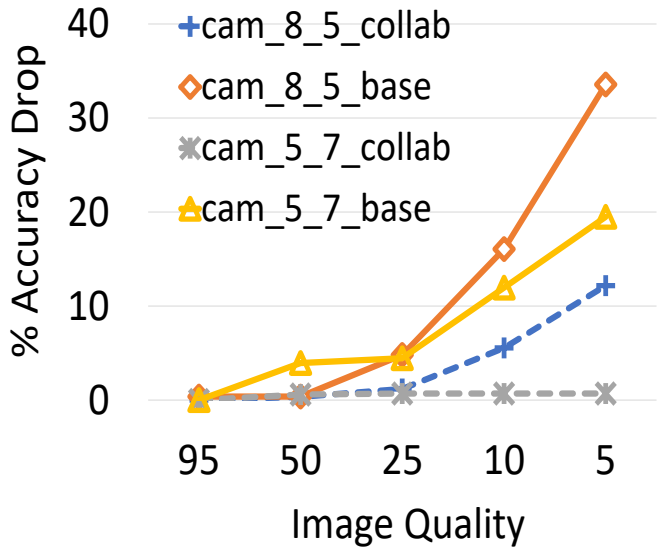


Figure 3.21: *CollabCam* Performance on Compressed Images

The camera pair (5, 7) is such that the shared region covers most of the area where people are present and therefore person detection accuracy is not much impacted by the resolution drop in non-shared regions.

Performance on WILDTRACK: Since WILDTRACK has png images, so I used PNG compression scheme to further compress the images in this dataset. I varied the compression level from 1 (lowest compression) to 9 (highest compression) and measure energy overhead for each level. Since PNG encoding is lossless, i.e., no information is lost during compression-decompression phase, I found that the accuracy of *both the baseline and Collab-SSD models didn't decrease at all*, i.e., 0% drop in accuracy for both the camera pairs (5, 7) and (1, 4) when compression level is increased from 1 to 9.

Overall, it can be seen that the proposed collaborative model offers significantly better resolution-accuracy trade-offs by employing MRFS approach and image compression. Both MRFS and compression are complementary to each other, i.e., while MRFS saves overall energy by resolution reduction for shared regions, the image compression is aimed at compressing *non-shared regions* of an already captured mixed-resolution image to further reduce network transmission energy. When only MRFS is used, then in presence of a high accuracy object detector and

a robust coordinate mapping scheme, MRFS can reduce the data to be processed by *67-95%* with only *2-5%* drop in accuracy. On the other hand, when MRFS is combined with image compression (of non-shared regions), the drop in data to be processed can be further reduced to *95-98%* with $\leq 1\%$ further drop in accuracy.

3.10 Energy Savings

The reduction in sensing resolution, over a spatially overlapping "shared region", allows *CollabCam* to reduce sensing, storage and communication energy. Using the prior results on resolution (non-shared= 512x512, the highest input resolution for SSD512; shared= 70x70), I now quantify the *emulated* energy savings that can be achieved using the benchmark PETS & WILDTRACK deployments. For simplicity, I provide a detailed breakdown for the Pixy2 camera, while summarizing the equivalent results for the RPi camera.

3.10.1 PETS Dataset

CollabCam enables the different components of the sense-making pipeline to be optimized as follows:

- *Image Capture*: Using the Pixy2 energy estimation model (Section 3.8.4), each PETS camera operating at its original 720x576 resolution consumes 3.74 mJ of energy per image. With *CollabCam*, lower 70X70 resolution in the shared region reduces the number of total pixels processed from 414,720 to (a) 124,728 (by $\sim 70\%$) for camera pair (5, 7) which have 53% overlap, and (b) 136,257 (by $\sim 67\%$) for pair (8,5) which have 49% overlap, consequently resulting in 16.5% & 15.8% energy savings, respectively.

- *Image Storage:* Writing one image of size 720x576 takes 1620 writing cycles (1 cycle writes 256B) on Pixy2, thereby consuming 23.3 mJ of energy. Adopting *CollabCam's* approach reduces the writing cycles to (a) 488 for camera pair (5,7),

a 69.9% reduction in storage energy overhead, and (b) 533 for camera pair (8,5), achieving 67.1% energy savings.

- *Network Transmission*: Using the energy estimation model (section 3.8.4), I estimate that transmitting a PETS image at original resolution (720x576) consumes significant (416mJ) energy. On the other hand, the $\sim 70\%$ and $\sim 67\%$ reduction in image size, respectively, for pairs (5,7) and (8,5), translates into a corresponding 62% and 60% reduction in transmission energy/frame.

3.10.2 WILDTRACK Dataset

CollabCam's reduction in energy overheads can be broken down as follows:

- *Image Capture*: The base resolution of 1920x1080 incurs an energy of ∼7.27 mJ/frame. *CollabCam's* ability to operate with a reduced 70x70 resolution on the shared/overlapping region reduces (a) the total number of pixels processed by camera pair (1,4) with 47% overlap to 142,411 (by $\sim 93\%$), resulting in 56.5% energy savings, and (b) the total number of pixels processed by camera pair (5,7) with 63% overlap to 99,218 (by $\sim 95\%$), resulting in 57.8% energy savings, respectively.

- *Image Storage:* Each 1920x1080 image requires 8100 write cycles on Pixy2, thereby consuming ∼116 mJ of energy. *CollabCam* reduces the number of writing cycles to (a) 557 for camera pair 1,4), reducing the energy of Flash writes by ∼93%, and (b) 388 for camera pair (5,7), resulting in ∼95% energy savings.

- *Network Transmission*: Transmission of an original 1920x1080 resolution ($\sim$ 2MB) consumes ∼1.9J of energy on the Pixy2. *CollabCam* reduces the total number of transmitted bytes for camera pairs (1,4) and (5,7) by $\sim 93\%$ ($\sim$ 141KB) and $\sim 95\%$ ($\sim$ 100KB), respectively. This translates into transmission energy savings of 91% & 93%, respectively.

3.10.3 Total Energy Savings

This section discusses the total energy savings, potentially achievable by *CollabCam* system, in absence/presence of image compression functionality.

No Image Compression

Tables 3.5 & 3.6 represent the total potential energy savings for both of the benchmark datasets for Pixy2 & RPi setups respectively. Overall, the proposed collaborative sensing paradigm can reduce the total energy consumption of a camera sensor by $\sim 50\%$ for PETS and $\sim 87\%$ for WILDTRACK. For a more-limiting real-time streaming approach (where one can ignore the substantial storage overheads), the resulting savings are a slightly more modest at 41% and 82%. Overall, it can be seen that *CollabCam* can realize a **2 - 7.7** times increase in the operational lifetime of a typical networked camera sensor.

		% Savings			
Dataset	**Cams**	**Energy**$_{Cap}$	**Energy**$_{Mem}$	**Energy**$_{NW}$	**E**$_{Total}$
PETS	5, 7	16.5	69.9	62.1	62.2
PETS	8, 5	15.8	67.1	59.7	59.7
WT	1, 4	56.5	93.1	90.8	90.9
WT	5, 7	57.8	95.2	92.9	92.9

Table 3.5: *CollabCam's* Total Energy Savings(%) from Pixy2

		% Savings			
Dataset	**Cams**	**Energy**$_{Cap}$	**Energy**$_{Mem}$	**Energy**$_{NW}$	**E**$_{Total}$
PETS	5, 7	10.7	70.7	70.3	51.7
PETS	8, 5	10.3	67.9	67.5	49.6
WT	1, 4	44.2	93.3	93.2	87
WT	5, 7	45.1	95.4	95.3	88.9

Table 3.6: *CollabCam's* Total Energy Savings(%) from RPi

Image Compression Enabled

When image compression is enabled, *CollabCam* selects the maximum possible compression level, for each camera pair, such that the {*compression overhead, ac-*

curacy drop} (section 3.8.4, section 3.9.3) are minimized. This results in selection of (i) image quality = 5 & 50 for PETS camera pairs (5, 7) & (8, 5) respectively and (ii) compression level = 1 for WILDTRACK camera pairs (1, 4) & (5, 7). I observe that though higher compression levels (> 1) for WILDTRACK camera pairs do contribute to reducing network transmission energy, but their computation overheads grow more rapidly (figure 3.17) and shadow the gains in network transmission energy.

Table 3.7 shows the resulting total energy savings on RPi setup, when image compression is enabled. While image compression enables more energy savings, compared to no-compression, for PETS camera pairs (highlighted in green), the WILDTRACK camera pairs (highlighted in red) actually end up consuming more energy compared to no-compression scenario. This happens primarily because of high computation overhead of PNG encoding scheme (lossless) used for WILDTRACK images.

		% Savings			
Dataset	**Cams**	**Energy**$_{Cap}$	**Energy**$_{Mem}$	**Energy**$_{NW}$	**E**$_{Total}$
PETS	5, 7	10.7	70.7	85.6	57.2
PETS	8, 5	10.3	67.9	84.3	55.7
WT	1, 4	44.2	93.3	75.4	78.9
WT	5, 7	45.1	95.4	81.7	82.7

Table 3.7: *CollabCam's* Total Energy Savings(%) from RPi with Image Compression Enabled

3.11 Discussion

3.11.1 Additional Collaborators

An individual camera may share an overlapping area with $N \geq 2$ cameras (if the deployment has greater redundancy). Above presented experimental results consider only one collaborator ($N = 1$) for *CollabCam*, begetting the question of whether greater collaboration can lead to higher performance improvement. To test this

hypothesis, I conducted tests where I varied N between $\{1, \ldots, 5\}$ by creating synthetic additional priors by randomly perturbing the ground truth labels in the shared region. Table 3.8 shows the resulting mAP scores for both datasets. While $N = 1$ offers a sharp increase relative to $N = 0$, choices of $N \geq 2$ offer almost no significant additional improvement, indicating that *CollabCam* typically requires *only one collaborating camera prior, for any specific overlapping region*.

N	**0**	**1**	≥ 2
mAP (PETS)	0.76	0.89	0.893
mAP (WT)	0.466	0.52	0.522

Table 3.8: *CollabCam* Accuracy vs N: (shared res=160x160)

3.11.2 Alternate Approaches of Resolution Adjustment

CollabCam empirically estimates the optimal resolution of a region shared among a set of cameras. So, the exact value of this resolution can be influenced by various factors, e.g., type of objects being detected, distance of camera from the target objects and quality of object inference model. Having discussed the influence of a DNN on the resolution reduction, this section focuses on other two factors for resolution reduction.

Depending on the type of objects being detected, value of the optimal resolution might vary, e.g., automobile objects might be easier to detect as compared to pedestrian users, therefore automobile objects can be captured at relatively lower resolution. In presence of multiple objects of various types, *CollabCam* can utilize the empirically created lookup table containing $\{object\_type, resolution\}$ mapping and adapt its sensing resolution to the best suited resolution for the type of object most likely to be present in the frame at that time.

Secondly, for a given type of object, the distance of object from the camera can also impact the optimal sensing resolution. Objects closer to camera would appear bigger, compared to far objects, and therefore would be represented by more pixels. This can enable the possibility of further reducing the sensing resolution for

these objects. In such cases, the *CollabCam* can use an empirically created lookup table containing $\{object\_type, distance, resolution\}$ mapping. Such objects can be captured at relatively lower resolution while still maintaining the object detection accuracy.

Both of these functionalities require some a-prior information (without running heavy-weight inference on the frame) about the possible type of objects and their likely distance from the camera, in the current frame. This can be achieved by using, for example, a light-weight object detector on the camera module itself or using some preliminary feature analysis approach to derive estimated probabilities for the type of objects present in the frame.

3.12 Reflections and Lessons Learnt

3.12.1 Overlap Estimator Tightly Coupled with Threshold

For a pair of cameras, the eventual energy savings of *CollabCam* depends on the extent of overlap between cameras. In this context, we found that the performance of weighted approach for overlap estimation (sec 3.5.3) is tightly coupled with the value of maximum pixel value threshold (T_w). While selecting a higher value of T_w will result in overestimation of overlap area, the lower T_w value will cause under-estimation overlap area. Therefore, selecting an optimal value of T_w is crucial for overall energy savings of *CollabCam* .

3.12.2 Coordinate Mapping vs Collab-DNN Performance

Absence of accurate coordinate mapping scheme impacts the minimum acceptable resolution (sec 3.9.2) for overlapping region, consequently, affecting the eventual energy savings. We found that coordinate mapping accuracy highly depends on the deployment specific attributes of camera systems and one specific algorithm might not fit all camera pairs (I used regression models of different degrees for different

camera pairs).

3.12.3 Importance of Appropriate Sensor Platform Selection

From our experience with prototyping on two different camera sensors, we found that camera specifications (both hardware configurations and software capabilities) impact the feasibility of implementing mixed-resolution sensing paradigm and overall energy savings of a multi-camera system. While RPi offers more support in terms of better computation and networking capabilities, the Pixy2 platform provides a comparatively *lower-level* programming control of the vision sensor that might be crucial in enabling mixed-resolution image sensing.

Chapter 4

Literature Review

4.1 *SmrtFridge*

The related work in this field consists of both Smart Fridge-specific prototypes and systems, as well as broader work in the area of IR/visual camera-based sensing. The desired capabilities of a smart fridge are often motivated by studies on food wastage [45, 30] which found that 48% of food wastage is due to items that have passed their expiration date, with 36.5% of the cases arising from food left untended (without the user's awareness) inside a fridge.

4.1.1 Smart Fridge Prototypes

A widely adopted approach for tracking the contents of a fridge involves the use of RFID tags attached to individual food objects. Noutchet [85] and Gu & Wang [50] propose attaching RFID tags to each product, with an RFID reader scanning each tag whenever an item is placed in or removed from the fridge. While this approach may be useful once all packaged items are universally tagged, its use at present would require extensive manual effort in labelling each object before inserting in the fridge and removing such tags before eventual discard. The Pervasive Fridge system [97] also envisions a system that tracks food items and their expiry dates (a feature that has been reported [37] to be extremely desired by users) via a manual

process that uses multiple modalities (barcode scanning, audio input or text input) to explicitly add items and their attributes (e.g., expiry dates).

The CloudFridge prototype [100] is one of the first systems to build and evaluate a sensor-based prototype to track user-fridge interactions in real-time and retrieve current and historical states of food items. Similar to our approach, CloudFridge applies video-based recognition to identify individual food items and uses multiple additional proximity (IR) sensors to keep track of each item's location inside the fridge. Evaluations performed using full-frontal images of the objects achieve precision values of $\approx 70\%$. However, CloudFridge does not directly address the real-world problems of extracting food item sub-images from videos of real-world human interaction or of estimating the residual food amount in a container. Along similar lines, the *PerFridge* system [82] augments a refrigerator with various sensors such as proximity (IR) and magnetic sensors to track various forms of 'wasteful' behavior, such as leaving a fridge door open for an excessively long duration or stacking multiple items in one corner (resulting in improper air flow). *PerFridge* does not, however, automatically identify food items or their residual quantity, relying instead on a touch-screen interface for explicit human input.

Several commercial smart fridge products have also recently been announced. An example is the *Liebherr* smart fridge [8], which uses an interior-mounted camera to classify food items inside the fridge and a voice recognition system to process voice commands (such as ordering food items). At present, we are, however, unable to quantify the performance of such commercial systems under densely-packed, occluded scenarios.

4.1.2 Analysis of Food Item Type

This section describes a line of work that has explored the use of automated techniques to identify food items based on image analysis. Nutrinet [78] applied a CNN-based approach to recognize 520 commonplace food and drink items, typi-

cally captured by a smartphone camera, with an accuracy of $\approx 87\%$, whereas Kagaya et al. [65] previously demonstrated how CNNs provided better food recognition accuracy (using a public food blogging dataset) than shallow classifiers, such as SVMs. More recently, Wang et al. [106] has shown how phase/RSSI information from RFID tags mounted on containers can help distinguish between different liquids in containers with accuracy as high as 94%. Lastly, the Annapurna system [101] addressed the problem of identifying and extracting images of plated food items captured by a smartwatch-embedded camera. Most such approaches are based on close-up photos of food content that is assumed to constitute the foreground. In *SmrtFridge*, we explicitly tackle this challenge of extracting out the food object from images of natural human interaction, captured by a fridge-embedded camera.

4.1.3 Food Content Quantity Estimation

There has been a variety of innovative work, employing different sensing modalities (e.g., visual, weight and RF), to infer various attributes of container-based food items. In the most recent and relevant work, Jiang et al. [61] employ a CNN (Convolutional Neural Network) based approach to estimate four discrete levels of content inside a glass or *transparent* bottle. Although the CNN is trained with various coloured plastic/glass bottles, a purely visual sensing approach does not work for non-transparent containers, e.g., paper cartons. A while back, Chi et al. [35] had demonstrated a method for estimating the type of food ingredients and their quantity using a combination of weight sensors and camera-based identification of ingredients (on a specially instrumented countertop). We believe that our use of an IR sensor to identify the quantity of liquids/semi-solids inside a container is a novel approach that exploits the temperature differential between a fridge's interior and its ambient surroundings. IR-based thermal tracking has been proposed in [88] to monitor the quality of vacuum-packed food containers. This approach, however,

monitors the *whole* container and does not attempt to use thermal variations for residual quantity estimation.

4.2 *CollabCam* System

The related work in this section broadly covers various multi-camera energy optimization techniques as well as research on multi-camera collaboration systems.

4.2.1 Camera Energy Consumption Optimizations

Classical work has utilized either local on-board image processing (e.g., Cyclops [90]) or multiple low & high resolution cameras (e.g., SensEye [66]) to reduce the energy overhead of vision-related tasks. The EECS framework [38] uses camera collaboration to deliver highly accurate object detection in an energy efficient way. Each camera is profiled for its detection accuracy and cost of executing some specific video processing algorithms on it. Then, for a given object detection task and the desired detection accuracy, EECS selects a set of optimal camera sensors and processing algorithms to minimize the overall energy consumption of the system. Unlike EECS, *CollabCam* system exploits the idea of mixed-resolution images to save overall sensing energy. Similarly, Zam et.al., [112] proposed an energy aware approach, for collaborative object detection, that activates specific sensor nodes based on the object's location in a camera and residual energy of each battery-operated camera node. This approach utilizes in-network collaboration among cameras for a cooperative object detection.

Captured image resolution can be an important factor for saving camera energy. Banner [57] saves camera energy by reducing overall image resolution based on the distance of object from the camera. By changing memory allocation scheme in OS, Banner achieves upto 70% reduction in camera energy. But, Banner is a single camera system and reduces the *overall* image resolution rather than capturing a mixed-resolution image. The Elf prototype [109] supports low-power estimation of

aggregate object counts by a energy-constrained camera by dynamically adjusting the number of distinct frames sampled over a user-defined time window.

4.2.2 Multi-Camera Collaboration Systems

Multiple cameras with overlapping field of view (FoV) have been used for cross-view person tracking and surveillance. Xu et al. [111] proposed a spatio-temporal Attributed Parse Graph (ST-APG) which combines semantic attributes (e.g., facing direction, action etc.) of a person with individual person trajectory for cross-view person tracking. Semantic attributes carry plenty of information and can help differentiate one user from another in cross-view scenario. Along similar lines, authors in [110] proposed a hierarchical composition approach for tracking people in a multi-camera setup. The proposed approach selects the optimal attributes, out of visual similarity, geometric properties, motion attributes etc. Aghajan & Wu [23] show how features extracted from multiple overlapping cameras can be fused to improve the accuracy of passive tracking of human gestures. More recently, the Caesar system [75] utilized both rule-based and DNN-based detection techniques to detect complex activities across multiple cameras.

SceneCam [91] proposes an Augmented-Reality (AR) based approach for optimal camera selection for a particular physical task spanning potentially across multiple cameras with overlapping FoVs. In a multi-camera setup, each camera might capture a particular perspective of a physical task, and it might be difficult for a remote user to select optimal camera for monitoring the task. SceneCam utilizes the AR based visualizations to aid the remote user to select the best possible camera. Multiple perspectives captured by cameras in a multi-camera setup can provide additional information that can be helpful in cases of object occlusion, motion-gesture sensing etc. Authors in [108] proposed a multi-camera collaboration based passive gesture recognition system. Human gestures can sometimes be difficult to recognize due to large range of possible gestures. Different low-level features, captured

by different cameras, are adaptively aggregated to estimate more abstract higher level features.

Multi-camera collaborations can also be used for increasing object detection accuracy (by utilizing different perspectives captured by different cameras), reducing communication bandwidth and reducing latency. Machine intelligence at the edge, as described in [79], talks about a visionary system where video cameras can utilize collaboration among themselves to reduce latency and increase object detection accuracy. Similarly, authors in [107] shows a system where multi-camera collaboration can increase object detection accuracy. Collaborating cameras can share the objects, detected in their frame, with other cameras and this extra information can be utilized by DNN layers to enhance the object detection accuracy. Unlike this system, the *CollabCam* system uses knowledge about the location of objects to create a mixed-resolution image to save sensing energy.

Chapter 5

Conclusion and Future Directions

In this chapter, I conclude this dissertation by summarizing its main contributions and also discuss some potential future directions of this work.

5.1 Summary of Contributions

In this dissertation, I propose two novel vision-analytics techniques to tackle two of the various potential challenges that might arise when machine-learning based IoT applications are deployed in real-world. As example applications of these techniques, I describe the design and performance analysis of two sensor systems, i.e., *SmrtFridge* & *CollabCam*. Each of these systems makes following novel contributions.

5.1.1 *SmrtFridge* System

In chapter 2, I described *SmrtFridge* – a system for user-interaction driven food item identification and residual quantity sensing using embedded IoT sensors. *SmrtFridge* uses a combination of visual and IR sensors to capture the user-interactions (add/remove items) with the fridge and pass the captured interactions to the processing pipelines which employ an IR driven or motion vector based food item segmentation techniques. The segmented food items are passed to an image classifier which

assigns a label to the food item. At the same time, the IR data from the captured interactions is used for estimation of residual quantity of food inside item containers as and when they are being re-inserted in the fridge. Overall, *SmrtFridge* makes following key contributions:

- **Dual Mode Visual Extraction of Individual Food Items:** I demonstrate a novel segmentation technique that reliably isolates the portion of an image frame, containing a food item object, from other unwanted objects in the frame, for example, human user, background objects and so on. The segmentation technique combines two approaches: (a) a combined *IR+ visual* approach, which allows easy visual isolation of the cold part of the image (very likely corresponding to a refrigerated item); and (b) a pure visual *optical flow-based* approach, which identifies foreground food item content even when it is at the ambient (room) temperature. Real-world user studies show that, in over 97% of interaction episodes, *SmrtFridge* can extract the food item with a bounding box that contains at least 75% of the item's pixels, and achieve a median Intersection Over Union (IoU) value of 0.68 (which is higher than the 0.45-0.5 threshold required for state-of-the-art object detection frameworks).

- **Accurate, Robust Object Recognition:** Based on the conducted user studies, I observed that a single user-item interaction episode typically lasts for $\sim$5-10 seconds, with the food item being visible in $\sim$5-10 images captured by a 30fps commodity camera. With these images, I utilize a DNN-based image classifier to reliably identify the specific food item that is either being inserted or extracted. Experimental studies, conducted with 12 users and 19 common food items, demonstrate that *SmrtFridge* can identify the food item brand/type with 84+% precision & recall, whereas the same DNNs achieve a baseline precision & recall of only 53% & 20% respectively, when supplied with the entire *un-cropped* image.

- **Accurate, Robust Quantity Estimation:** *SmrtFridge* uses the IR data to

estimate residual quantity inside container-based (opaque/transparent) food items. Quantity estimation is based on the property that differential heat-gain rate is different for container and its content. Using a controlled study conducted with 5 different items and 3 different quantities in paper containers, I show that the resulting thermal differences captured by a commodity IR sensor are discernible when the food item is placed outside the fridge for a period varying between 15 seconds to 15 minutes. By applying appropriate quantization and clustering techniques on such thermal images, I show that we can estimate the residual quantity of food items with a median and mean errors of 11% and 14% respectively (of the overall container capacity) and achieve 75% accuracy in classifying the residual quantity into three broad levels.

- **Practical *SmrtFridge* Prototype:** Using commodity visual and IR sensors and embedded platform (e.g., Raspberry PI), I build a prototype of *SmrtFridge*, comprising 1 IR sensor, 1 RGB camera and 1 door-contact sensor installed on an actual fridge. I also empirically determine the appropriate placement of these sensors, such that they provide both good item visibility and high spatial coverage under diverse, natural, user-item interaction patterns.

5.1.2 *CollabCam* System

In chapter 3, I described *CollabCam* – a multi-camera collaboration system for energy-efficient sensing. *CollabCam* utilizes collaboration among cameras to selectively drop resolution of shared regions of an image while keeping the non-shared regions at their original sensing resolution. The resulting potential accuracy drop due to reduced imaging resolution is tackled by utilizing the different perspective of the overlapping region as shared by collaborating cameras. *CollabCam* demonstrates that this reduced resolution can save overall camera energy consumed in

sensing, storage and network transmission of images. *CollabCam* makes following key contributions.

- **Accurate and Autonomous Estimation of Overlapping Areas:** *CollabCam* proposes an object-matching based approach to autonomously (i.e., no human intervention required) estimate the overlapping region between a pair of camera sensors. Resolution reduction within these shared regions is the key idea of *CollabCam* to enable energy-efficient sensing. The reduced imaging resolution enables the energy savings in image capture, their optional storage and finally network transmission. I show that the proposed weighted approach, for overlap area estimation, is able to estimate the overlapping region significantly more accurately ($IoU \geq 0.76$, with ground truth on benchmark datasets), compared to the non-weighted baseline approach which achieves IoU in the range $0.46 - 0.67$.

- **Technique for Coordinate Mappings between Sensor Pairs:** In order to enable cameras to share object location information with each other, it is necessary to transform object coordinates from one camera view to another camera view. *CollabCam* proposes a regression based approach mapping $\{X_1, Y_1, height_1, width_1\}$ of an object in one camera view to corresponding $\{X_2, Y_2, height_2, width_2\}$ of same object in another camera view. The regression-based model provides high accuracy coordinate translation of objects across different cameras, e.g., $\sim$60% & 100% of objects have IoU ≥ 0.5 in our benchmark datasets PETS & WILDTRACK respectively.

- **Collaborative DNN Model for Object Detection using Low-Resolution Images:** The mixed-resolution images can result in reduced task accuracy if an off-the-shelf object detector is used for inference, e.g., reducing the overlapping area resolution from 512x512 to 70x70 results in 48-61% drop in mAP score for our benchmark datasets. So, to maintain high task accuracy while using mixed-resolution images, *CollabCam* proposes a collaborative

object detection DNN that exploits the additional object location information shared by other collaborating cameras. The proposed collaborative model significantly improves the resolution-vs-accuracy tradeoff: across the PETS and WildTrack datasets, it ensures that the mAP score drops by no more than 5-6% and 2-5%, respectively, when the sensor's image resolution drops from 512x512 to 70x70 for various camera pairs. This represents a ∼46-56% and ∼38-105% improvement in mAP, for PETS and WildTrack respectively, over the non-cooperative baseline where each sensor's image is analyzed in isolation.

- **Prototype of an MRFS Vision Sensor:** We modified two vision sensor prototypes, the Pixy2 camera and the Raspberry Pi camera, with suitable firmware modifications such as variable row/column skipping across different portions of the same frame to support the capture of individual frames with mixed-resolution sub-regions. By carefully isolating and measuring the energy costs of multi-image capture, storage & network transmission for these vision sensors, we substantiate the substantial energy savings afforded by mixed-resolution sensing: 60-93% for Pixy2 & 35-80% for RPi camera.

5.2 Future Directions

In this section, I propose some potential directions for future work based on this dissertation.

5.2.1 *SmrtFridge* System

I believe that functionality of *SmrtFridge* can be further extended in following manners:

- **Multiple RGB and IR Cameras:** In order to provide coverage for bigger and multi-door fridges, it would be necessary to install multiple RGB and IR

cameras. In such cases, each of these camera should be able to collaborate with other camera in order to effectively execute the item identification and quantity estimation pipelines.

- **Fine-grained Quantity Estimation:** The granularity of quantity estimation depends on the resolution of thermal images captured by the IR camera. I believe that if we add weight sensors to the fridge, then the IR camera and weight sensors can collectively perform fine-grained quantity estimations. However, to minimize the number of weight sensors required, it is necessary that these sensors are sensitive enough to detect smaller quantities as well as support heavy weights (a typical fridge might weigh $\sim 100kg$).

- **Night Vision Functionality:** Currently, the RGB camera used by *SmrtFridge* might not work effectively during night time and thereby affecting the item identification accuracy. I believe that using no-IR filter RGB cameras combined with IR flood light(s) can help resolve the night visibility issue.

- **Context-Aware Item Recognition** In *SmrtFridge's* current implementation, extracted item images from an interaction episode are sent to the DNN for classification and then a majority voting technique is applied on the item labels to get the final label. On the other hand, a context-aware implementation of *SmrtFridge*, wherein the details like list of all items currently inside the fridge, location (inside fridge) and neighbouring items of the food item being added/removed are available, can be used to further improve the item identification accuracy. This additional information can help decide the final item label, from a set of labels returned by the DNN, by (a) ruling out items that are not present in the fridge and (b) ruling out items whose usual location is quite far from that of the item being added/removed. Realizing this functionality would require either instrumentation of fridge shelves with pressure sensors or inclusion of other internal cameras to track the motion of objects.

5.2.2 *CollabCam* System

Functionality of *CollabCam* can be further extended in following ways:

- **Adaptive Clock-Frequency:** Since shared regions of a mixed-resolution image are sampled at a lower rate, so while capturing these regions, it might be useful to decrease clock frequency supplied to the camera to further reduce energy consumption. The amount of clock frequency reduction is subjective to the size of shared region and the minimum acceptable resolution of these regions.

- **Adaptive Resolution in Shared Regions:** Among other factors governing the minimum acceptable resolution within a shared region, one factor is the size of objects as seen by the camera. Since objects that are closer to the camera would appear relatively bigger, so it might be possible to capture those objects at even lower resolution and therefore save more energy. Realizing this functionality might require a light-weight object detector and tracker either on the camera itself or at the edge node.

- **Multiple Low-Resolution Cameras Fusion** It would be interesting to explore if the *CollabCam* approach can be extended further such that low-resolution images from multiple cameras, placed at different orientations, angles, etc., can be (a) collectively used to achieve same end-task accuracy as with a single high resolution camera and (b) combined together to extract very fine-grained information (e.g. cracks in food items) that might not be possible from a single image taken from a specific point-of-view. Such a functionality could be quite useful in a food quality assessment system, employed for example, by an online grocery shopping company.

Bibliography

[1] Apple unveils new ipad pro with breakthrough lidar scanner and brings trackpad support to ipados. https://www.apple.com/sg/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in

[2] The best camera phone in 2020 — xiomi mi note 10. https://www.digitalcameraworld.com/buying-guides/best-camera-phone.

[3] Deep learning based system to recognize cooked food. https://foodai.org/. [Online; Last accessed 10-April-2019].

[4] Drones to survey dengue hotspots. https://www.smartnation.gov.sg/what-is-smart-nation/initiatives/Urban-Living/drones-to-survey-dengue-hotspots-1.

[5] Gaming applications and toys. https://www.bosch-sensortec.com/applications-solutions/gaming-toys/.

[6] Infrared sensors. https://sg.rs-online.com/web/c/automation-control-gear/sensors-transducers/infrared-sensors/.

[7] The internet of things 2020: Decision maker's view. https://www.businessinsider.com/internet-of-things-report?IR=T.

[8] Introducing the liebherr smart refrigerator in cooperation with microsoft. https://blog.liebherr.com/appliances/sg/liebherr-smart-refrigerator-microsoft/. [Online; Last accessed 15-Feb-2019].

[9] Motion sense — new features coming to google pixel 4. https://www.blog.google/products/pixel/new-features-pixel4/.

[10] Nvidia jetson nano. https://www.nvidia.com/en-sg/autonomous-machines/embedded-systems/jetson-nano/.

[11] Pets 2009 benchmark data. http://www.cvg.reading.ac.uk/PETS2009/a.html.

[12] Pets 2009 benchmark data. http://cs.binghamton.edu/~mrldata/pets2009.

[13] Pixy - pixycam. https://pixycam.com/pixy-cmucam5/.

[14] Raspberry pi camera module v2. https://www.raspberrypi.org/products/camera-module-v2/.

[15] Smart london — smart city initiatives. https://www.london.gov.uk/what-we-do/business-and-economy/supporting-londons-sectors/smart-london.

[16] Smart singapore — smart nation initiatives. https://www.smartnation.gov.sg,.

[17] Ssd: Single-shot multibox detector implementation in keras. https://github.com/pierluigiferrari/ssd_keras.

[18] Ultra short range radar — texas instruments. http://www.ti.com/solution/automotive-ultra-short-range-radar.

[19] The working principle and key applications of infrared sensors. https://www.azosensors.com/article.aspx?ArticleID=339.

[20] Home automation using edge computing and internet of things. In *2017 IEEE International Symposium on Consumer Electronics (ISCE)*, pages 47–49, 2017.

[21] T. Abdelzaher, Y. Hao, K. Jayarajah, A. Misra, P. Skarin, S. Yao, D. Weerakoon, and K.-E. Årzén. Five challenges in cloud-enabled intelligence and control. *ACM Trans. Internet Technol.*, 20(1), Feb. 2020.

[22] P. Aditya, R. Sen, P. Druschel, S. Joon Oh, R. Benenson, M. Fritz, B. Schiele, B. Bhattacharjee, and T. T. Wu. I-pic: A platform for privacy-compliant image capture. In *Proceedings of the 14th annual international conference on mobile systems, applications, and services*, pages 235–248, 2016.

[23] H. Aghajan and C. Wu. Layered and collaborative gesture analysis in multi-camera networks. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 2007.

[24] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri. Iot-based occupancy monitoring techniques for energy-efficient smart buildings. In *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 58–63, 2015.

[25] I. Ali, S. Sabir, and Z. Ullah. Internet of things security, device authentication and access control: a review. *arXiv preprint arXiv:1901.07309*, 2019.

[26] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. A low power, fully event-based gesture recognition system. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[27] S. R. Anthony Rowe, Rich LeGrand. http://www.cmucam.org/projects/cmucam5.

[28] S. B. Baker, W. Xiang, and I. Atkinson. Internet of things for smart healthcare: Technologies, challenges, and opportunities. *IEEE Access*, 5:26521–26544, 2017.

[29] L. Baroffio, L. Bondi, M. Cesana, A. E. Redondi, and M. Tagliasacchi. A visual sensor network for parking lot occupancy detection in smart cities. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 745–750, 2015.

[30] M. Bonaccorsi, S. Betti, G. Rateni, D. Esposito, A. Brischetto, M. Marseglia, P. Dario, and F. Cavallo. 'HighChest': An Augmented Freezer Designed for Smart Food Management and Promotion of Eco-Efficient Behaviour. *Sensors*, 17(6):1357, June 2017.

[31] I. Bouchrika, J. N. Carter, and M. S. Nixon. Towards automated visual surveillance using gait for identity recognition and tracking across multiple non-intersecting cameras. *Multimedia Tools and Applications*, 75(2):1201–1221, 2016.

[32] P. Bour, E. Cribelier, and V. Argyriou. Crowd behavior analysis from fixed and moving cameras. In *Multimodal Behavior Analysis in the Wild*, pages 289–322. Elsevier, 2019.

[33] T. Chavdarova, P. Baqué, A. Maksai, S. Bouquet, C. Jose, L. Lettry, F. Fleuret, P. Fua, and L. V. Gool. Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection. *2018 Ieee/Cvf Conference On Computer Vision And Pattern Recognition (Cvpr)*, pages 5030–5039, 2018.

[34] T. Chavdarova and F. Fleuret. Deep multi-camera people detection. *CoRR*, abs/1702.04593, 2017.

[35] P.-Y. P. Chi, J.-H. Chen, H.-H. Chu, and J.-L. Lo. Enabling Calorie-Aware Cooking in a Smart Kitchen. In H. Oinas-Kukkonen, P. Hasle, M. Harjumaa, K. Segerst\a ahl, and P. Øhrstrøm, editors, *Persuasive Technology*, pages 116–127, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[36] Comparitech. Surveillance camera statistics: which cities have the most cctv cameras? `https://www.comparitech.com/vpn-privacy/the-worlds-most-surveilled-cities/`.

[37] A. Cusack, A. Fox, A. Hiscock, M. VanOirschot, E. Oguejiofor, P. Eng, and M. P. Dioron. Refrigeration revolution project proposal. 2012.

[38] T. Dao, K. Khalil, A. K. Roy-Chowdhury, S. V. Krishnamurthy, and L. Kaplan. Energy efficient object detection in camera sensor networks. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1208–1218, 2017.

[39] A. Davis, M. Rubinstein, N. Wadhwa, G. J. Mysore, F. Durand, and W. T. Freeman. The visual microphone: Passive recovery of sound from video. 2014.

[40] P. Dias. Tsai camera calibration. *IEETA/Universidade de Aveiro, Portugal*, 2003.

[41] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[42] S. Fan, H. Shin, and R. R. Choudhury. Injecting life into toys. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, pages 1–6, 2014.

[43] J. Ferryman and A. Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 1–6, Dec 2009.

[44] R. P. FOUNDATION. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/?resellerType=home.

[45] S. Gaiani. *Lo spreco alimentare domestico in Italia: stime, cause ed impatti*. PhD thesis, alma, 2013.

[46] W. Ge and R. T. Collins. Crowd detection with a multiview sampler. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 324–337, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[47] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[48] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[49] M. Goel, E. Whitmire, A. Mariakakis, T. S. Saponas, N. Joshi, D. Morris, B. Guenter, M. Gavriliu, G. Borriello, and S. N. Patel. Hypercam: hyperspectral imaging for ubiquitous computing applications. In *In Proc. of UbiComp*. ACM, 2015.

[50] H. Gu and D. Wang. A content-aware fridge based on rfid in smart home for home-healthcare. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 2, pages 987–990. IEEE, 2009.

[51] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[52] A. Hermans*, L. Beyer*, and B. Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*, 2017.

[53] J. Hester and J. Sorber. Flicker: Rapid prototyping for the batteryless internet-of-things. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–13, 2017.

[54] J. Hester and J. Sorber. The future of sensing is batteryless, intermittent, and awesome. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–6, 2017.

[55] J. Hester, K. Storer, and J. Sorber. Timely execution on intermittently powered batteryless sensors. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pages 1–13, 2017.

[56] S. L. Hoe. Defining a smart nation: the case of singapore. *Journal of Information, Communication and Ethics in Society*, 2016.

[57] J. Hu, A. Shearer, S. Rajagopalan, and R. LiKamWa. Banner – an image sensor reconfiguration framework for seamless resolution-based tradeoffs (video). In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '19, page 705–706, New York, NY, USA, 2019. Association for Computing Machinery.

[58] L. N. Huynh, R. K. Balan, and Y. Lee. Deepmon: Building mobile gpu deep learning models for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 186–186, 2017.

[59] A. Inc. What is apple face id and how does it work. https://www.pocket-lint.com/phones/news/apple/142207-what-is-apple-face-id-and-how-does-it-work, 2019.

[60] A. Itzkovitch. The internet of things and the mythical smart fridge, 2013-09-18. https://uxmag.com/articles/the-internet-of-things-and-the-mythical-smart-fridge.

[61] Y. Jiang, E. Schenck, S. Kranz, S. Banerjee, and N. K. Banerjee. CNN-Based Non-contact Detection of Food Level in Bottles from RGB Images. In I. Kompatsiaris, B. Huet, V. Mezaris, C. Gurrin, W.-H. Cheng, and S. Vrochidis, editors, *MultiMedia Modeling*, pages 202–213, Cham, 2019. Springer International Publishing.

[62] jmondi. Based on sony imx258 camera driver. *https://github.com/jmondi*, 2018.

[63] Ju, L. Yun, Z. Wang, Hui, and Chang. The application of improved yolo v3 in multi-scale target detection. *Applied Sciences*, 9:3775, 09 2019.

[64] J. Jörissen, C. Priefer, and K.-R. Bräutigam. Food waste generation at household level: Results of a survey among employees of two european research centers in italy and germany. *Sustainability*, 7(3):2695–2715, 2015.

[65] H. Kagaya, K. Aizawa, and M. Ogawa. Food detection and recognition using convolutional neural network. In *ACM Multimedia Conference*, 11 2014.

[66] P. Kulkarni, D. Ganesan, P. Shenoy, and Q. Lu. Senseye: A multi-tier camera sensor network. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05. ACM, 2005.

[67] N. D. Lane, P. Georgiev, and L. Qendro. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 283–294, 2015.

[68] L. Lee and W. E. L. Grimson. Gait analysis for recognition and classification. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 155–162, 2002.

[69] S. Levy. Projective transformations. *30th Edition of the CRC Standard Mathematical Tables and Formulas (CRC Press)*, 1995.

[70] J. Lezama, Q. Qiu, and G. Sapiro. Not afraid of the dark: Nir-vis face recognition via cross-spectral hallucination and low-rank embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6628–6637, 2017.

[71] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, page 69–82, New York, NY, USA, 2013. Association for Computing Machinery.

[72] R. LiKamWa and L. Zhong. Starfish: Efficient concurrency support for computer vision applications. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '15, page 213–226, New York, NY, USA, 2015. Association for Computing Machinery.

[73] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2014.

[74] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[75] X. Liu, P. Ghosh, O. Ulutan, B. Manjunath, K. Chan, and R. Govindan. Caesar: cross-camera complex activity recognition. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pages 232–244, 2019.

[76] S. Luo, J. Jin, J. Li, et al. A smart fridge with an ability to enhance health and enable better nutrition. *International Journal of Multimedia and Ubiquitous Engineering*, 4(2):69–80, 2009.

[77] M. Mateen, J. Wen, S. Song, Z. Huang, et al. Fundus image classification using vgg-19 architecture with pca and svd. *Symmetry*, 11(1):1, 2019.

[78] S. Mezgec and B. K. Seljak. Nutrinet: A deep learning food and drink image recognition system for dietary assessment. In *Nutrients*, 2017.

[79] A. Misra, K. Jayarajah, D. Weerakoon, R. Tandriansyah, S. Yao, and T. Abdelzaher. Dependable machine intelligence at the tactical edge. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, page 1100608. International Society for Optics and Photonics, 2019.

[80] A. Misra, D. K. W. WEERAKOON MUDIYANSELAGE, and K. Jayarajah. The challenge of collaborative iot-based inferencing in adversarial settings. 2019.

[81] H. Mohsen, E.-S. A. El-Dahshan, E.-S. M. El-Horbaty, and A.-B. M. Salem. Classification using deep learning neural networks for brain tumors. *Future Computing and Informatics Journal*, 3(1):68 – 71, 2018.

[82] S. Murata, S. Kagatsume, H. Taguchi, and K. Fujinami. Perfridge: An augmented refrigerator that detects and presents wasteful usage for eco-persuasion. In *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pages 367–374. IEEE.

[83] G. Murthy and R. Jadon. A review of vision based hand gestures recognition. *International Journal of Information Technology and Knowledge Management*, 2(2):405–410, 2009.

[84] S. Naderiparizi, P. Zhang, M. Philipose, B. Priyantha, J. Liu, and D. Ganesan. Glimpse: A programmable early-discard camera architecture for continuous mobile vision. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 292–305, 2017.

[85] A. D. Noutchet. Novel User Centric RFID Fridge Design. *Computer and Information Science*, 6(2), Apr. 2013.

[86] OpenCV. Basic concepts of the homography explained with code. *Web Page*.

[87] P. Peng, Y. Tian, Y. Wang, J. Li, and T. Huang. Robust multiple cameras pedestrian detection with multi-view bayesian network. *Pattern Recognition*, 48(5):1760 – 1772, 2015.

[88] A. Popa, M. Hnatiuc, M. Paun, O. Geman, D. J. Hemanth, D. Dorcea, L. H. Son, and S. Ghita. An intelligent iot-based food quality monitoring approach using low-cost sensors. *Symmetry*, 11:374, 2019.

[89] H. Qiu, X. Liu, S. Rallapalli, A. J. Bency, K. Chan, R. Urgaonkar, B. S. Manjunath, and R. Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. In *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 48–59, 2018.

[90] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, SenSys '05. ACM, 2005.

[91] T. A. Rasmussen and W. Huang. Scenecam: Using ar to improve multi-camera remote collaboration. In *SIGGRAPH Asia 2019 XR*, SA '19, page 36–37, New York, NY, USA, 2019. Association for Computing Machinery.

[92] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[93] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[94] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[95] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[96] E. Ristani and C. Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6036–6046, 2018.

[97] J. Rouillard. The Pervasive Fridge. A smart computer system against uneaten food loss. page 7.

[98] P. J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[99] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[100] T. Sandholm, D. Lee, B. Tegelund, S. Han, B. Shin, and B. Kim. CloudFridge: A Testbed for Smart Fridge Interactions. *arXiv:1401.0585 [cs]*, Jan. 2014. arXiv: 1401.0585.

[101] S. Sen, V. Subbaraju, A. Misra, R. Balan, and Y. Lee. Annapurna: Building a real-world smartwatch-based automated food journal. In *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2018.

[102] J. Shu, R. Zheng, and P. Hui. Cardea: context-aware visual privacy protection for photo taking and sharing. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 304–315, 2018.

[103] G. Singapore. Singapore lamppost as a platform. `https://www.tech.gov.sg/scewc2019/laap`.

[104] G. Singh, S. Miao, S. Shi, and P. Chiang. Fotonnet: A hw-efficient object detection system using 3d-depth segmentation and 2d-dnn classifier. *arXiv preprint arXiv:1811.07493*, 2018.

[105] X. Toh, H.-X. Tan, H. Liang, and H.-P. Tan. Elderly medication adherence monitoring with the internet of things. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6. IEEE, 2016.

[106] J. Wang, J. Xiong, X. Chen, H. Jiang, R. K. Balan, and D. Fang. Tagscan: Simultaneous target imaging and material identification with commodity RFID devices. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom 2017*, 2017.

[107] W. M. D. K. WEERAKOON, K. JAYARAJAH, R. T. DARATAN, and A. MISRA. Resilient collaborative intelligence for adversarial iot environments. 2019.

[108] C. Wu and H. Aghajan. Collaborative gesture analysis in multi-camera networks. In *ACM SenSys Workshop on DSC*. Citeseer, 2006.

[109] M. Xu, X. Zhang, Y. Liu, G. Huang, X. Liu, and F. X. Lin. Approximate query service on autonomous iot cameras. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, page 191–205. Association for Computing Machinery, 2020.

[110] Y. Xu, X. Liu, Y. Liu, and S.-C. Zhu. Multi-view people tracking via hierarchical trajectory composition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4256–4265, 2016.

[111] Y. Xu, X. Liu, L. Qin, and S.-C. Zhu. Cross-View People Tracking by Scene-Centered Spatio-Temporal Parsing. In *AAAI*, 2017.

[112] A. Zam, M. R. Khayyambashi, and A. Bohlooli. Energy-aware strategy for collaborative target-detection in wireless multimedia sensor network. *Multimedia Tools and Applications*, 78(13):18921–18941, 2019.

[113] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun. Alignedreid: Surpassing human-level performance in person re-identification. *CoRR*, abs/1711.08184, 2017.

[114] S. Zhu, C. Zhang, and X. Zhang. Lishield: Privacy protection of physical environment against photographing. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 522–524, 2017.

[115] M. Zlmmerling, F. Ferrari, L. Mottola, T. Voigt, and L. Thiele. ptunes: Runtime parameter adaptation for low-power mac protocols. In *2012 ACM/IEEE 11th International Conference on Information Processing in Sensor Networks (IPSN)*, 2012.

[116] H. Zou, J. Yang, H. Prasanna Das, H. Liu, Y. Zhou, and C. J. Spanos. Wifi and vision multimodal learning for accurate and robust device-free human activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.