Sensor Fusion for Mobile Devices

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS
in
Computer Science

by

Kumar Vishal
201250925
kumar.vishal@research.iiit.ac.in

Center for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2015

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Sensor Fusion for Mobile Devices" by Kumar Vishal, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Dr. C. V. Jawahar

To my family

Acknowledgments

First of all, I would like to show my deepest gratitude towards my adviser, Dr. C. V. Jawahar, for his guidance, support, motivation and for keeping me from going off track. I would also like to wholeheartedly thank my family, my parents and my wife, for their love and unwavering support in all my ventures. I am thankful to my lab mates and friends at IIIT for all the fruitful(as well as not so fruitful) discussions Anand, Devendra, Udit, Praveen, Yashaswi, Nataraj, Mohak, Rajvi. A special thanks to all the faculty members at CVIT, my adviser Dr. Jawahar as well as Dr. P. J. Narayanan, Dr. Anoop Namboodiri, Dr. Jayanthi Sivaswamy for creating a wonderful environment in CVIT to do research, and CVIT administration Satya, Phani, Rajan and Nandini for helping me on numerous occasions.

Abstract

Sensor fusion is a process by which data from several different sensors are "fused" to compute something more than could be determined by any one sensor alone. Simple embedded devices like smart phones, smart watches encompass wide variety of sensors like camera, Gps , accelerometer etc. In this thesis, we have shown how user experience of these devices can be greatly enhanced by providing them capability to process and analysis these data sensor data automatically. Fusing Gps and vision sensor we have designed an algorithm for localization in 3D and showed how these sensor complementary for each other. Moving ahead in a different applications we showed how analyzing and fusing the different sensor data along with machine learning algorithms can boost the performance of these devices. We tried to optimize the power consumption of batteries in smartphones by using the sensor data from accelerometer, touch screen and cpu usage.

Consider a wearable device for localization in 3D which answers the question "Where Am I ?" for a given environment. Localization in 3D is an important problem with wide ranging applications from autonomous navigation in robotics to location specific services on wearable and mobile devices. GPS sensors are a commercially viable option for localization, and are ubiquitous in their use, especially in portable devices. With the proliferation of mobile cameras, however, maturing localization algorithms based on computer vision are emerging as a viable alternative. Although both vision and Gps based localization algorithms have many limitations and inaccuracies, there are some interesting complementarities in their success/failure scenarios that justify an investigation into their joint utilization. Such investigations are further justified considering that many of the modern wearable and mobile computing devices come with sensors for both Gps and vision. In this work, we investigate approaches to reinforce Gps localization with vision algorithms and vice versa. Specifically, we show how noisy Gps signals can be rectified by vision based localization of images captured in the vicinity. Alternatively, we also show how Gps readouts might be used to disambiguate images when they are visually similar looking but belong to different places. Finally, we empirically validate our solutions to show that fusing both these approaches can result in a more accurate and reliable localization of videos captured with a Contour action camera, over a 600 meter long path, over 10 different days.

There is a rapid growth in memory and processing power of mobile devices unfortunately the battery life is still limited in terms of size and capacity. This implies that managing the battery power is paramount in such devices. As long as the battery technology continues its slow pace of improvement, the only viable approach is to reduce the amount of energy required to provide specific services. Two

of the most power consuming services on a smartphone are network and wireless data. Though internet connectivity is important, the nature of data transfer does not require uninterrupted service. We take advantage of this fact to cut off power to these modules when it is not required. The challenge is to predict, when the users require these services and when they do not. We log the sensor data and fuse them to make one single feature and using machine learning approaches to intelligently schedule Wi-Fi according to a users activity level. Several higher order features from the raw sensor data stream are used to classify the activity level. Two aspects of the problem are considered, namely the accuracy of estimating the activity level as well as the power required in sensing and estimation. Experimental results on Android based smartphones demonstrate that an active user can get up to 37% increase in battery life without significant effect on the user experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sensor data fusion is an emerging research field with the aim to combine information from multiple and diverse sources (e.g. different sensors such as thermal and visible cameras, laser, Gps , accelerometer etc.) to achieve inferences that cannot be obtained from a single sensor or source, or whose quality exceeds that of an inference drawn from any single source [4,5]. Sensor data fusion is a multi-disciplinary subject that overlaps with areas like statistical estimation, signal processing, computer vision machine learning etc. In computer vision familiarly image sensor is fused with other sensors in order to develop an intelligent system. Now days cameras are cheap small and ubiquitous, and with the emergence of energy-efficient and powerful processors, it has become possible to incorporate practical computer vision and machine learning capabilities into embedded systems, mobile devices etc.

Localization is a fundamental problem associated with autonomous navigation. One of the simplest solution to the localization problem is with the help of Global Positioning Systems(Gps). In literature, computer vision methods have also been successfully used for predicting the location. A wide variety of modern gadgets (e.g. smartphones) have both Gps and vision sensors, and such devices are becoming increasingly affordable for low cost robotic systems. Each sensor has its own unique characteristics and its challenges. Unfortunately, the accuracy of the popular Gps tracking devices are limited to only 10$\sim$20 meters, which is insufficient for many robotic tasks[6]. So noise in the Gps signal becomes a significant issue if we want a reliable localization performance. Where as vision based localization methods suffers problems like occlusion, perceptual aliasing etc. Hence we can deduce for reliable localization fusing multiple sensors which are complimentary in nature will lead us to more accurate and robust localization. The ambiguities in visual localization can be reduced with the help of Gps . Similarly, noise in the Gps signal can be reduced by looking at the visual consistency across multiple sessions. We empirically validate our solution and show that fusing of Gps and camera sensors which are having complementary advantages and disadvantage can result in a more accurate and reliable localization.

Smartphones have become essential part of our daily life as they continue to evolve and introduce newer features. However battery capacity of smart phones have failed to keep up the pace at which smartphones have evolved in recent years. It was observed that the network services like Wi-Fi, EDGE and 3G are most power consuming process on the smartphones [7]. Hence, reducing the power con-

sumption by these processes could help in power savings. Using the significant data from different sensors like accelerometer, touch and system usage we propose a novel approach to schedule services like Wi-Fi and 3G on smartphones. Different sensors data were fused to model and monitor the user activity level to decide when the wireless data module should be turned off for maximal energy saving. Taking Wi-Fi as an example, we show that intelligent scheduling of the Wi-Fi service based on a users activity level leads to lower power consumption without adversely affecting the user experience.

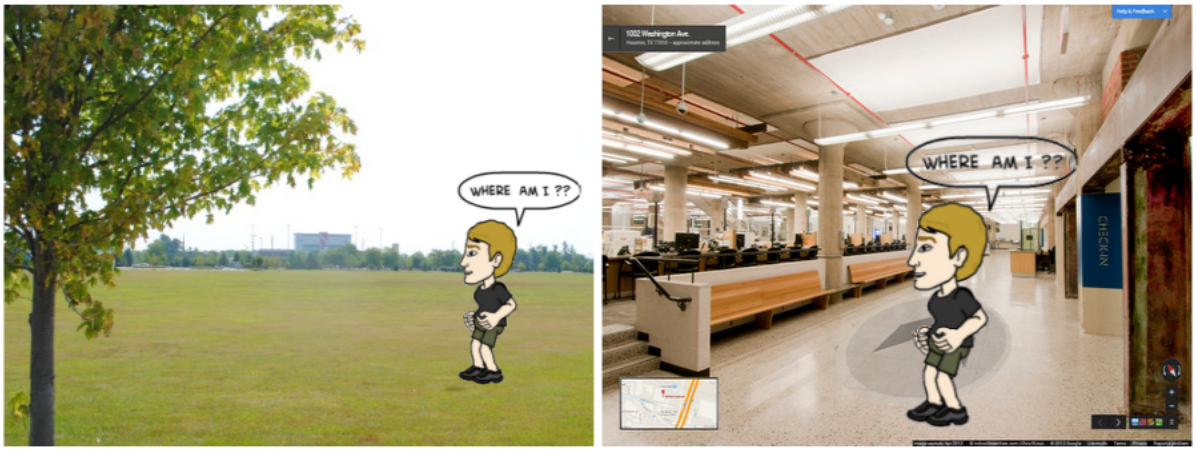


Figure 1.1: Localization is a process that determine the position of a robot/human pedestrians in a given environment.

Computer vision task is notably demanding, for a variety of reasons. First, simply put, there are massive amounts of data to consider. Frame resolutions higher than 1 Mpixel need to be streamed uncompressed at 30 to 60 frames per second, and every pixel of each frame requires processing attention. Typical vision processing algorithms for classification or tracking, for example, also combine a mix of vector and scalar processing steps, which demand intensive external memory access patterns with conventional processing architectures. Shuttling intermediate processing results back and forth between the CPU and external memory drives up power consumption and increases overall processing latency. In recent years have we seen vision capabilities show up in mobile applications. Running these algorithms on mobile device comes up with new set of challenges like optimizing the existing algorithms, good understanding of system in order to handle memory and computational requirement.

We describe the background for the computer vision and machine learning tasks tackled in this thesis in Section 1.1. Problem statement and the technical contributions of this thesis have been presented in Section 1.2.

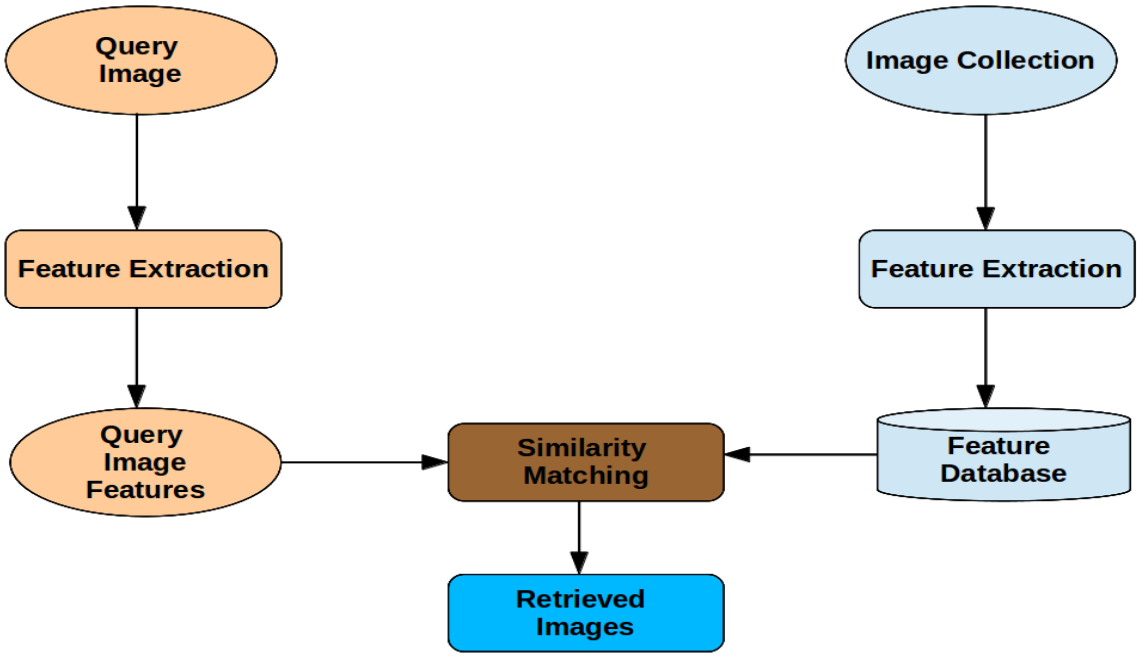


Figure 1.2: Figure shows a generic model of Content-based image retrieval system. User can fire the query in form of images in result system return the similar images from the database.

1.1 Computer Vision and Machine Learning Concepts

1.1.1 Content based image retrieval

Content-based image retrieval (CBIR), also known as query by image content (QBIC) is an application of computer vision techniques to the image retrieval problem, that is, the problem of searching for digital images in large databases [8]. A generic and simple model of a CBIR is shown in Figure 1.2. "Content-based" means that the search analyzes the contents of the image rather than the metadata such as keywords, tags, or descriptions associated with the image. Given a query image, its features are extracted and compared with the stored database features and a ranked list of images relevant to the query is retrieved.

The CBIR technology has numerous applications like vision based localization, fingerprint identification, biodiversity information systems, digital libraries, crime prevention, medicine, historical research etc [9]. In this thesis visual localization is formulates as CBIR based problem. A query image is provided by an image sensor given this single image, we retrieve visually similar images from the database. A voting mechanism is applied on all the retrieved images in order to choose best match and estimates the location.

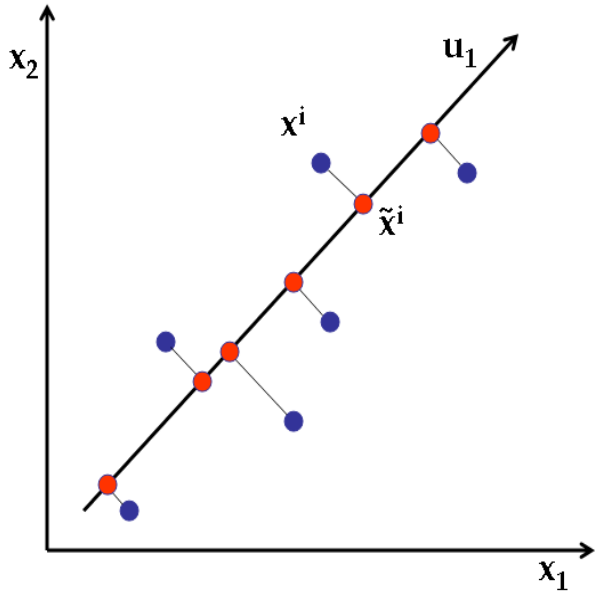


Figure 1.3: Two dimensional data points(blue) are projected along the vector u_1(red) in order to reduce the dimensionality.

1.1.2 Classification

In machine learning, classification is the problem of labeling to which category a new observation belongs to, using the training set of data containing observations whose labels are known to us. A popular example can be for a new incoming e-mail we have to label it as ”spam” or ”non-spam”. In machine learning, classification is taken as instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. An algorithm that implements classification at concrete implementation level is known as a classifier. The term ”classifier” sometimes also refers to the mathematical function, implemented by a classification algorithm, that labels the input data. Support Vector Machine(SVM), Decision Tree, Logistic Regression and Naive Base are few famous classification algorithm which are used frequently to perform the classification task.

1.1.3 Dimensionality Reduction

In machine learning dimensionality reduction is the way to reduce the number of random variables under consideration and can be divided into feature selection and feature extraction. Dimensionality reduction is usually performed on higher dimensionality data set in order to avoid curse of dimensionality. Principal component analysis (PCA), linear discriminant analysis (LDA), or canonical correlation analysis (CCA) are the techniques which can be used to perform the dimensionality reduction. Figure 1.3 shows a demo example of dimensionality reduction.

1.2 Problem Statement and Contribution

The focus of this thesis is on multi-sensor data fusion. We have shown empirically by fusing data from multiple sources increases the robustness and efficiency of a system. We aim at using the sensors like GPS, camera, accerelometer and touch sensors and demonstrated how they compliment each other. Taking these into consideration, we would like to specifically address the following sub-problems:

1. **Improving Localization by Fusing Camera and GPS Sensors:** We addresses the problem of localization by fusing Gps and camera sensor using vision algorithms. Peculiarly we demonstrate how a noisy Gps signal can be amend by vision based localization of images in an environment, alternatively we also show that Gps information can be used as a tie breaker for visually similar looking images but belong to different places. We focus on the challenges like occlusion, change in view point and noise in Gps data and showed by using multiple sensor data we can handle these problems more efficiently as compare to single sensor.

2. **Reducing Power Consumption by Fusing Sensors data on Smartphones:** Smartphone are powered from batteries which is limited in their capacity. Services like Wi-Fi and 3G are one of the most power consuming processes on smartphone but one good thing with these services are they are not required in a continuous fashion. To reduce the power consumption by these services we predict the user activity level and make them available only when they are needed. We model the user activity level with respect to smartphone device by fusing different sensors data like accelerometer, touch screen, cpu and memory usage. Based on these levels we schedule the services like Wi-Fi and 3G to reduce the power consumption of the smartphones.

1.3 Thesis Outline

In $Chapter2$, we briefly discuss the technical background needed for the later chapters in the thesis. In $Chapter3$ the problem of localization by fusing the Gps and image sensors has been tackled. In this chapter we demonstrate how Gps and image sensor can be fused together to overcome the issues like occlusion, perceptual aliasing etc.. $Chapter4$ shows how sensor data fusion with machine learning algorithm can make the smart devices more smarter and efficient. Using the data from accelerometer, touch sensors, CPU and RAM usage we have tried to predict the user engagement with phone and schedule the Wi-Fi accordingly in order to save the battery power. Conclusions, and future directions of our work have been given in $Chapter5$.

Chapter 2

Background

In this chapter, we briefly discuss the machine learning and computer vision concepts that have been used in the thesis.

2.1 Machine Learning Concepts

2.1.1 Principal Component Analysis

Principal Component Analysis(PCA) is a very popular dimensionality reduction approach proposed by Hotelling [10]. Apart from dimensionality reduction, PCA is also used for data visualization, feature extraction and lossy data compression [11]. PCA is the projection of data onto a orthogonal set of basis vectors such that the variance of the data after the projection is maximized. These basis vectors are obtained by solving an eigenvalue problem involving the covariance matrix of the data samples.

Consider a dataset consisting of n observations $x_1, x_2, ...x_n$ such that $x_i \in R^d$. Let the first principal component be obtained by projecting the data along u_1, i.e. the direction of maximum variance of the data. The variance of the data along the direction u_1 can be represented as

$$u_1^T S u_1 \tag{2.1}$$

where S is the covariance matrix of the observations given by

$$S = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T. \tag{2.2}$$

Here the column vectors x_i have been centered around the dataset mean. In order to maximize the variance given in Equation 2.1, we need to maximize the objective with respect to u_1. However, unconstrained maximization of this objective will result in u_1 having very large magnitude. In order to keep the l_2 norm of u_1 under check, a unit norm constraint is added to objective. This constrained objective can be converted to an unconstrained one using the method of Lagrange multipliers . Using this approach, the unconstrained objective can be rewritten as

$$u_1^T S u_1 \quad + \lambda_1 (1 - u_1^T u_1), \tag{2.3}$$

where λ_1 is the Lagrange multiplier. This objective can be maximized by setting its derivative with respect to u_1 as zero. This results in the following eigenvalue problem

$$S u_1 = \lambda_1 u_1. \tag{2.4}$$

Hence u_1 is the eigenvector of S which corresponds to its largest eigenvalue. The other eigenvectors can be found similarly with added constraint that the eigenvectors must be orthogonal to one another. We have described the maximum variance formulation for PCA as in Bishop [11].

2.1.2 Support Vector Machine

Consider a binary classification problem where any feature vector $x \in R^d$ and corresponding class labels can be $y \in \{1, -1\}$. Support vector machine(SVM) constructs a decision surface in the form of a separating hyperplane:

$$w^T x + b = 0 \tag{2.5}$$

so that the positive and the negative examples are separated by the hyperplane. There can be an infinite number of hyperplanes which separates the positive & the negative examples(in case the data is linearly separable). Out of all such hyperplanes, SVM picks the hyperplane which maximizes the margin between the examples in the positive and the negative class. Such a hyperplane must satisfy:

$$w^T x_i + b \geq 1 \ \text{ for } \ y_i = 1, \tag{2.6}$$

$$w^T x_i + b < -1 \ \text{ for } \ y_i = -1 \tag{2.7}$$

The margin of separation ρ, between the positive and the negative examples is given by

$$\rho = \frac{2}{|w|} \tag{2.8}$$

The margin maximizing hyperplane can be found by solving the following constrained optimization problem

$$\min_{w,b} \frac{1}{2} w^T w \quad \text{subject to} \quad y_i (w^T x_i + b) \geq 1. \tag{2.9}$$

The above optimization problem is convex and hence, a unique hyperplane exists which is a global solution of the above constrained optimization problem. However, it turns out that many real world datasets are not linearly separable. Hence, in such a case, any feasible solution for the above objective does not exist. A *soft margin* version of SVM exists which can deal with such a scenario.

2.1.2.1 Kernelized Approaches

Sometimes the patterns inherent in the data are not apparent from the original feature representation. In such scenarios, it is a common practice to do a linear or a non-linear mapping/transformation of the data so that after the transformation, the patterns in the data become more obvious. Kernel based approaches are a non-linear feature mapping approach which map the data points in the original space to its mapping in a Reproducing Kernel Hilbert Space(RKHS).

Consider feature vector $x \in R^d$. A kernel function k is defined as

$$\begin{aligned} k : X \times X \mapsto \mathbb{R}, \\ < x_i, x_j > \mapsto k(x_i, x_j), \end{aligned} \tag{2.10}$$

where x_i and x_j objects from X. The kernel function k is said to be positive definite if it satisfies the following properties

- k is symmetric, i.e. $k(x_i, x_j) = k(x_j, x_i)$
- $\sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0$, where $\alpha_i, \alpha_j \in \mathbb{R}$ and $n \in \mathbb{N}$

If the kernel function is positive definite, then there exists some mapping function ϕ such that

$$< \phi(x_i), \phi(x_j) >= k(x_i, x_j), \tag{2.11}$$

here ϕ is a mapping function which maps data points from original space to a Reproducing Kernel Hilbert Space(RKHS).

2.1.2.2 Multiclass SVM

Multiclass SVM assigns the label to an instance where number of labels are more than two. The dominant approach for doing so is to reduce the multiclass problem into multiple binary classification problems. Building binary classifiers which distinguish between (i) one of the labels and the rest (one-versus-all) or (ii) between every pair of classes (one-versus-one). Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.

2.1.3 Artificial Neural Networks(ANN)

Artificial Neural Network sometimes also abbreviated as Neural Networks is a computational model inspired from biological neural networks. *Dr. Robert Hecht-Nielsen* inventor of first neuro computer

defines neural network as -
"*...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*".

2.1.3.1 Structure of Artificial Neural Networks

An ANN model Figure 2.1 encompass three different layers input layer, hidden layer and output layer.

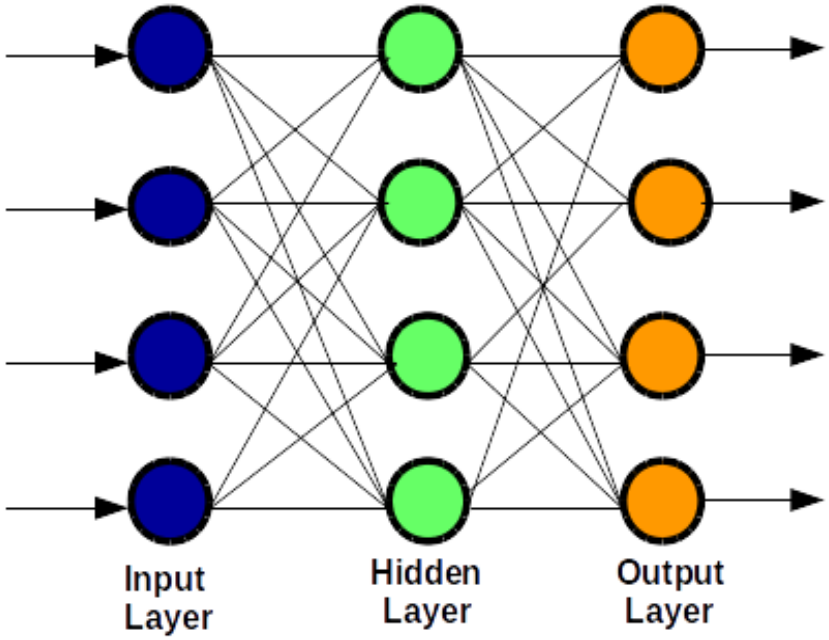


Figure 2.1: Figure is decepting a general ANN model and different layers.

Input Layer: All the inputs are fed into the model through this layer. The Input layer is a communication medium with the external environment that feed a pattern to the ANN.

Hidden Layer: Hidden layer is a group of neurons combined with the activation function[12] is an intermediate layer between the input and output layer. The job of this layer is process the input from its previous layer and pass the extracted data to next layer. Many researches has been conducted for evaluating the number of neurons in the hidden layer but still none of them was successful in finding the accurate result. Also there can be multiple hidden layers in an ANN model. Number of hidden layer depend upon the complexity of the input data. If we have a data which can be separated linearly, then there is no need of hidden layer as the activation function can be implemented to input layer which can solve the problem. But in case of data is non-linear and problems deals with complex decisions, we can use multiple hidden layers based on the degree of complexity of the problem. Increasing the number of

layers will not always result into high accuracy. A stage comes when the accuracy becomes constant or falls with an extra added layer. Also, we should calculate the optimal number of neurons for each network. If the number of neurons are less as compared to the complexity of the problem data then there will be very few neurons in the hidden layers to adequately detect the signals in a complicated data set. Whereas a greater number of neurons present in ANN may lead to overfitting of the model. In-order to tune the number of hidden layers and neurons there are some empirically-derived rules-of-thumb, of these, the most commonly relied on is, *"the optimal size of the hidden layer is usually between the size of the input and size of the output layers" - Jeff Heaton*[13].

Output Layer: The output layer of the neural network is the set of results generated by the previous layer. The number of neurons in output layer should be directly related to the type of work that the neural network was performing. To determine the number of neurons in the output layer, first consider the intended use of the neural network.

Working of ANN: In the Figure 2.1 each connection between two neurons indicates the flow of the information. Each connection has there correspondig weights which control the signal intensity between two neurons. If the network generates desired result then there is no need of weight adjustment else weight adjustment is carried out by using Back Propagation Algorithm.

Back Propagation Algorithm: It is the training algorithm used to efficiently train ANN. It learn by example and adjust weight of neurons so that it can produce desired output from the trained model.

2.1.4 Nearest Neighbors

Nearest neighbors are one of the most classifiers, possibly because it does not involve any training. This technique simply consists of storing all the labeled training examples and given a test images, the label of closest training sample(or majority label of k-neighbors) is assigned to the test image. Nearest neighbors can be kernelized or coupled with metric learning [14].

For classification techniques such as SVM, only a single weight vector needs to be stored per class(in a one-vs-rest setting), however, a nearest neighbor classification strategy typically consists of storing all the training samples. This is one of the major demerits of nearest neighbors.

2.1.5 Random Walks on Graphs

Let G be a graph or digraph with the additional assumption that if G is a directed graph, then $deg^+(v) > 0$ for every vertex v. Now consider an object placed at vertex v_j . At each stage the object must move to an adjacent vertex. The probability that it moves to the vertex v_i is $\frac{1}{deg(v_j)}$ or $\frac{1}{deg^+(v_j)}$ if (v_j, v_i) is an edge on G and G is a graph or directed graph, respectively. Otherwise the probability is 0. Therefore if we define as

$$m_{ij} = \begin{cases} \frac{1}{deg(v_j)} & \text{left if } (v_j, v_i) \text{ is an edge in the graph } G \\ \frac{1}{deg^+(v_j)} & \text{left if } (v_j, v_i) \text{ is an edge in the directed graph } G \\ 0 & \text{otherwise} \end{cases}$$

Then $M = (m_{ij})$ is a Markov matrix. Note that the roles of i and j are reversed as we need the columns of M to sum to 1. As each stage occurs, a sequence of adjacent vertices is produced. This sequence represents the position of the object at a given stage. Moreover this sequence is a walk in the graph. We call such a walk a random walk on the graph or digraph G. Using the Markov matrix, we see that the i, j entry of M^k represents the probability that a random walk of length k starting at vertex v_j , ends at the vertex v_i. The steady-state vector will correspond to the probability of being at a given vertex after a sufficient number of random walks. The above random walks on graph definition is cited from [15].

2.1.6 Evaluation Measures

k-fold cross-validation: We used k-fold cross validation technique to do the analysis of our experiments. In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data. The cross-validation process is then repeated k times(the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged(or otherwise combined) to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.
The above definition of k-fold cross validation is taken from Wiki source [16] .

2.2 Computer Vision Concepts

2.2.1 Bag Of Words

The Bag-of-Words (BoW) methodology was first proposed in the text retrieval domain problem for text document analysis, and it was further adapted for computer vision applications [17]. For image analysis, a visual analogue of a word is used in the BoW model, which is based on the vector quantization process by clustering low-level visual features of local regions or points, such as color, texture, and so forth. Below is the details description of the Bow methodology by Tsai [18].

To extract the BoW feature from images involves the following steps: (i) automatically detect regions/points of interest, (ii) compute local descriptors over those regions/points, (iii) quantize the descriptors into words to form the visual vocabulary, and (iv) find the occurrences in the image of each

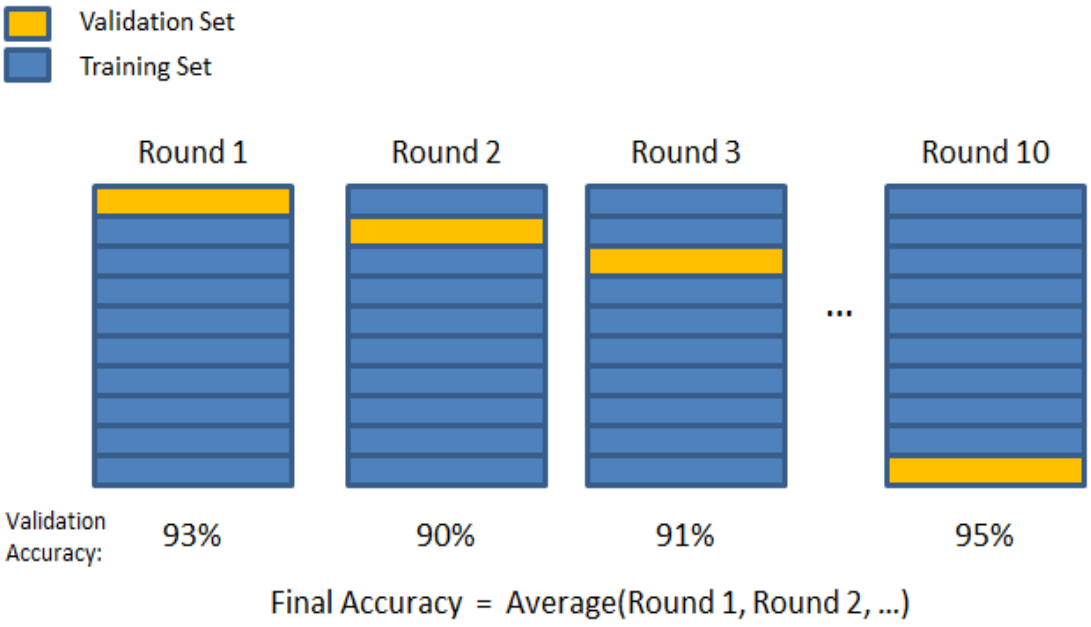


Figure 2.2: The training set is split into k smaller sets and a model is trained using $k-1$ of the folds as training data. The resulting model is validated on the remaining part of the data. The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop.

specific word in the vocabulary for constructing the BoW feature (or a histogram of word frequencies). Figure 2.3(courtesy [18]) describes these four steps to extract the BoW feature from images.

The BoW model can be defined as follows. Given a training dataset D containing n images represented by $D = d_1, d_2, ..d_n$, where d is the extracted visual features, a specific unsupervised learning algorithm, such as $k - means$, is used to group D based on a fixed number of visual words W (or categories) represented by $W = w_1, w_2, ..w_v$, where V is the cluster number. Then, we can summarize the data in a $V \times N$ co-occurrence table of counts $N_{ij} = n(w_i, d_j)$, where $n(w_i, d_j)$ denotes how often the word w_i occurred in an image d_j.

2.2.1.1 Interest Point Detection

The first step of the BoW methodology is to detect local interest regions or points. For feature extraction of interest points (or keypoints), they are computed at predefined locations and scales. Several well-known region detectors that have been described in the literature are discussed below [19, 20].

1. Harris-Laplace regions are detected by the scale-adapted Harris function and selected in scale-space by the Laplacian-of-Gaussian operator. Harris-Laplace detects corner-like structures.

2. DoG regions are localized at local scale-space maxima of the difference-of-Gaussian. This detector is suitable for finding blob-like structures. In addition, the DoG point detector has previously

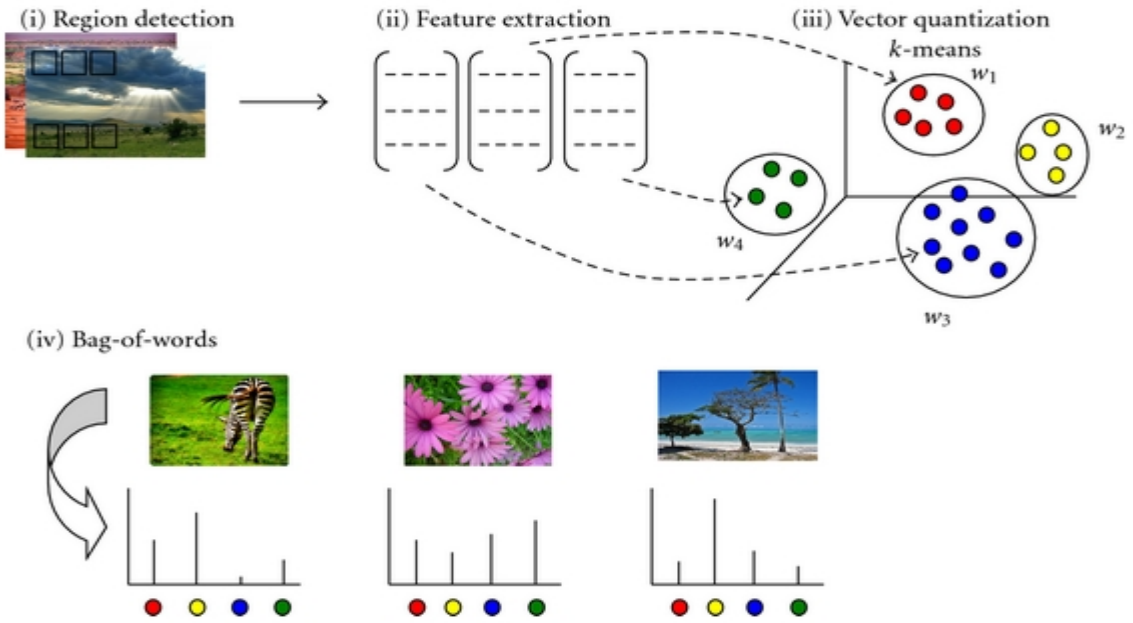


Figure 2.3: Steps for constructing the bag-of-words for image representation. In first and second step finding the interest region and feature extractions are performed. While in third and fourth step clustering of features and histogram representation of the images are done.

been shown to perform well, and it is also faster and more compact (less feature points per image) than other detectors.

3. Hessian-Laplace regions are localized in space at the local maxima of the Hessian determinant and in scale at the local maxima of the Laplacian-of-Gaussian.

4. Salient regions are detected in scale-space at local maxima of the entropy. The entropy of pixel intensity histograms is measured for circular regions of various sizes at each image position.

5. Maximally stable extremal regions (MSERs) are components of connected pixels in a thresholded image. A watershed-like segmentation algorithm is applied to image intensities and segment boundaries which are stable over a wide range of thresholds that define the region.

2.2.1.2 Local Descriptors

In most studies, some single local descriptors are extracted, in which the Scale Invariant Feature Transform(SIFT) descriptor is the most widely extracted [21]. It combines a scale invariant region detector and a descriptor based on the gradient distribution in the detected regions. The descriptor is represented by a 3D histogram of gradient locations and orientations. The dimensionality of the SIFT

descriptor is 128. SIFT was found to work best [19, 22–24]. SURF, ORB and BRISK few more widely used descriptors.

2.2.1.3 Visual Word Generation/Vector Clustering

When the keypoints are detected and their features are extracted, such as with the SIFT descriptor, the final step of extracting the BoW feature from images is based on vector quantization. In general, the $k-means$ or $hierarchical\ k-means$ clustering algorithm is used for this task, and the number of visual words generated is based on the number of clusters(i.e.,). Jiang et al. [25] conducted a comprehensive study on the representation choices of BoW, including vocabulary size, weighting scheme, such as binary, term frequency(TF) and term frequency-inverse document frequency (TF-IDF), stop word removal, feature selection, and so forth for video and image annotation.

2.2.1.4 Building Inverted Index

To perform large-scale image retrieval, Bag of Features methods require efficient indexing strategies and the ability to handle large vocabularies. An inverted index method for indexing the gallery term vectors is one where each term records the images in which it appears, along with the weight for each image. If each image yields approximately 2,000 interest points and the vocabulary size is $10K$ terms, then the resulting term vector (assuming no multiple/soft assignments) will have at most 20% of its entries as non-zero. In a larger system with a vocabulary of $100K$ to $1M$ terms, the term vectors will be even sparser.

2.2.1.5 Query Handling

A simple approach to ranking query results is to compute the distance between the query vector and the term vectors of each gallery image, requiring an $O(N)$ computation in term of the number of gallery images. The majority of those images are likely to have very few terms in common due to the sparsity of the vectors, however. Instead, one can look at each non-zero term in the query vector and get the list of images in which that term appears via the inverted index. The normalized L_p distance to each image discovered in the inverted index can be computed incrementally as the inverted index is traversed, such as demonstrated in [26]. While the worst case complexity remains $O(N)$, there is a huge speed improvement for the average case. Inverted indexes are used in all contemporary BoF-based image retrieval methods that were surveyed in the above report.

2.2.2 Trifocal Tensor

Trifocal Geometry: Finding the correspondence among the image primitives (points or lines) over multiple views is recognized as a fundamental problem in computer vision. Assume we have three images captured by three calibrated cameras with the known extrinsic calibration(camera positions in

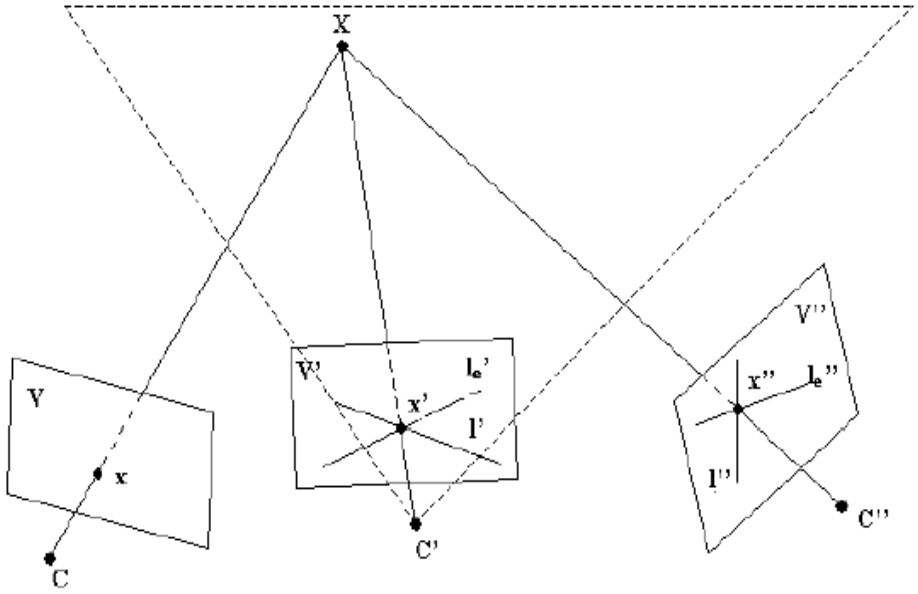


Figure 2.4: Trifocal geometry of three views. X is a $3D$ point viewed from three camera having optical center C, C' and C''.

Euclidean space) and the intrinsic parameters (camera calibration information). Figure 2.4 shows the geometry of three cameras. Their optical centers are denoted by C, C' and C''. A $3D$ point X in Euclidean space has its unique image on the image plane of each camera at where the ray linking the point to the camera optical center intersects the plane. However the opposite is not true since the image point can be the projection of any $3D$ point on that ray. The nonzero scale factor λ in the equation 2.13 explains this ambiguity. The Euclidean camera is defined by a 3×4 camera matrix $P = K[R|t]$.

$$P_i^P = H_j^P \tag{2.12}$$

The 3×3 camera calibration matrix K contains the intrinsic parameters of the specific camera. The camera's pose with respect to the world coordinate system is represented by 3×3 rotation R and $3D$ vector translation t. The projection from $3D$ point $(X; Y; Z)$ to image point $(x; y)$ is up to a scale. The scale ambiguity can be overcome by having correspondences for a feature in two views. The $3D$ point X can then be reconstructed by means of triangulation using the point correspondence pair x and x' on view V and V'. Then this point can be re-projected into the third view V'' to find out where the image coordinate of this $3D$ point should appear on that view. This process x, $x \rightarrow x''$ is called transfer, and it can be done with at least one calibrated camera and correspondences across two views. For un-calibrated cameras no transformation between an image point and a ray in Euclidean space is

provided because their intrinsic and extrinsic parameters are unknown.

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.13}$$

In this case the cameras are defined to be projective cameras($K = I$) and any one can be replaced by another. The $3D$-to-$2D$ projection P^p becomes camera independent. The projective spaces where such projections take place are all equivalent since they can align to each other only by a $3D$ projective transformation (4×4 matrix H)[27].

$$P_i^P = H_j^P \tag{2.14}$$

It is the relative camera position that determines the transformation between two projective spaces. Usually when more than one camera are considered, a normalized camera coordinate system is used and one of cameras is $P = [I|0]$. Consequently the positions of other cameras in this system, and the extrinsic camera parameters are all that is necessary to define the camera projection matrices. Therefore even when the three cameras mentioned above are not calibrated, image correspondences can still be transferred from two views to the third. As will be discussed in the next section, it is possible to define the projective geometry of the three cameras by a trifocal tensor. When the tensor is computed the transfer can be done directly by it without computing an intermediate $3D$ point. Since this removes the need of camera calibration, the trifocal tensor has been recognized an important tool for processing three views.

It needs to be pointed out that corresponding image points on three views are also related by epipolar geometry. The ray projected from one image point will be viewed as a line in another view(called the epipolar line). The relation between image correspondences of a view pair is described by a 3×3 fundamental matrix, *i.e* $x'^T F_{12} x = 0$ of view i and j. The three views can be registered together in a pair-wise manner. Again given a pair of image correspondence(x, x'), intersecting their epipolar lines on the third view should show the image position of the $3D$ point. However, if the point X is in the trifocal plane defined by the optical centers C, C' and C''. or if the centers are aligned, it is impossible to determine if three image points belong to a single $3D$ point by epipolar geometry. Such ambiguity can be avoided by using a trifocal tensor since it provides a more accurate and stable description of three views' geometry than the fundamental matrices between each pair of views.

Trifocal Tensor: The trifocal tensor is expressed by a set of three 3×3 matrices,$[T_i]$, $i = 1, 2, 3$. It describes the projective geometric relations of image triplets taken from three cameras. Considering a view triplet, if the camera matrix of the first view is in canonical form, $P_1 = [I|0]$, and the camera matrices of the other two views are expressed as $P_2 = [A|e']$, $P_3 = [B|e'']$, where A and B are 3×3 matrices, and e' and e'' are the epipoles corresponding to the image of the center of the first camera on

the image plane of the second and third cameras respectively, then the $3 \times 3 \times 3$ trifocal tensor can be denoted as $T = [T_1, T_2, T_3]^T$, with:

$$T_i = a_i e''^T - e' b_i^T \tag{2.15}$$

The equation 2.15 presents a straightforward way to construct the trifocal tensor from the camera matrices.

Chapter 3

Accurate Localization by Fusing Images and GPS Signals

Localization in 3D is an important problem with wide ranging applications from autonomous navigation in robotics to location specific services on mobile devices. Gps sensors are a commercially viable option for localization, and are ubiquitous in their use, especially in portable devices. With the proliferation of mobile cameras however, maturing localization algorithms based on computer vision are emerging as a viable alternative. Although both vision and Gps based localization algorithms have many limitations and inaccuracies, there are some interesting complementarities in their success/failure scenarios that justify an investigation into their joint utilization. Such investigations are further justified considering that many of the modern wearable and mobile computing devices come with sensors for both Gps and vision.

In this work, we investigate approaches to reinforce Gps localization with vision algorithms and vice versa. Specifically, we show how noisy Gps signals can be rectified by vision based localization of images captured in the vicinity. Alternatively, we also show how Gps readouts might be used to disambiguate images when they are visually similar looking but belong to different places. Finally, we empirically validate our solutions to show that fusing both these approaches can result in a more accurate and reliable localization of videos captured with a Contour action camera, over a 600 meter long path, over 10 different days.

3.1 Introduction

Localization refers to the idea of "locating" the position of an object within its environment. It has numerous applications in wearable computing, robotics, entertainment devices and consumer electronics. Most popular localization approaches are designed to represent object location in 3D coordinate systems either using the lat/long format like Global Position System (Gps) sensors, or by using metric distances like vision based localization methods. Gps based methods provide global/absolute information about the location of an object with the help of special purpose sensors, and satellite communication. Vision based approaches usually provide localization relative to a reference image, and are not global in nature. Visual localization is achieved by matching images using interest points like SIFT,

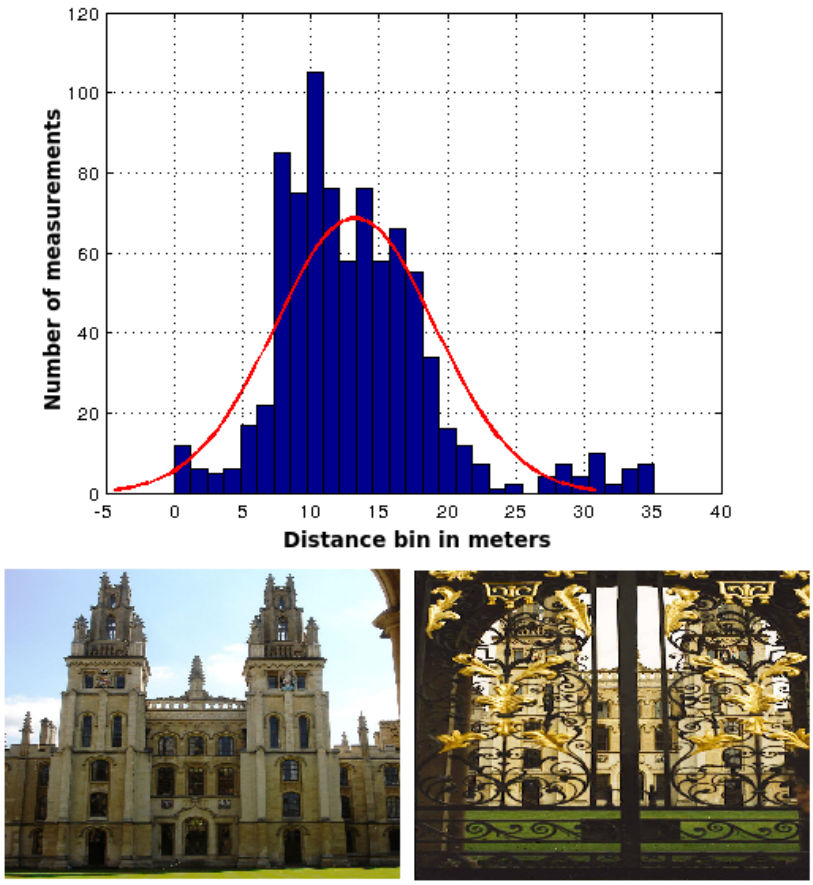


Figure 3.1: Histogram of Gps localization error (top row) of a *stationary* Gps sensor, showing how inaccurate they can be. Bottom row is an example of two images belonging to the approximately same pose. Visual localization is inaccurate here since in one image the object is occluded, while Gps sensors give accurate localization.

and estimating relative positions by computing and decomposing multiview geometric quantities like the Fundamental/Essential matrix. There are several advantages and disadvantages of using Gps based localization vis-a-vis visual localization approaches. Commercial viability of Gps sensors make them *cheap* to obtain, thus explaining their ubiquitousness. Such sensors are generally useful for obtaining coarse localization of objects in a *global* coordinate system. They are also not usually affected by the visual quality of an object's surroundings, *i.e.* Gps sensors localize with similar accuracy irrespective of whether they are used on a beach (no unique interest points) or near a popular monument (uniquely identifiable structures), and they give *unambiguous* localization to visually similar but differently located places. However, Gps sensors are *inaccurate* beyond a certain point, as illustrated in Figure 3.1, and can fail in many environments due to reasons such as *sporadic unavailability* of the satellite signal [28]. Thus, *cheap, global, unambiguous, inaccurate, sporadic unavailability* are keywords that characterize Gps sensors.

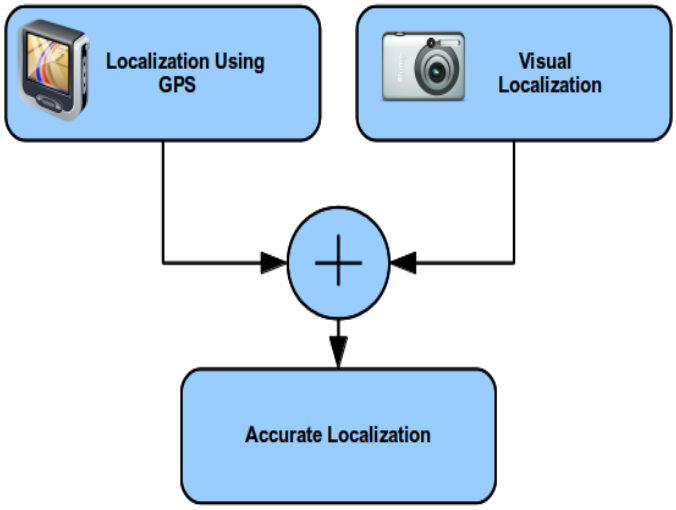
With the sudden increase of consumer cameras found on portable devices, vision based localization approaches are also now *cheaply* available. Such approaches are generally useful for fine localization of objects in a *local* coordinate system relative to a reference frame. Compared to Gps sensors, vision based localization systems also provide reasonable *accurate* estimates of the object's location. However, chances of *ambiguities* are higher in vision based localization methods since visual similarity of two images of far apart places can lead to erroneous localization estimates. However, since vision based localization methods are not dependent on satellite connectivity, such approaches are readily *available* for utilization. Thus vision based localization approaches can be characterized to be *cheap, local, ambiguous, accurate and available*.

Notice that the two sensors have complimentary advantages and disadvantages. Thus, it is natural to ask *why not combine the advantages of both to improve their accuracy and reliability (Figure 3.2a)*. With recent portable devices carrying both Gps and vision based sensors, we answer this increasingly important question in this paper. In section 3.1.1, we discuss related work. We then describe an approach to improve visual localization using Gps sensory output in section 3.2. Then we elaborate on an approach to improve Gps output using visual information in section 3.3, before describing an experiment that complimentarily fuses both the improved estimates to do sequential localization in section 3.4. Finally we performed the relevant experiments on collected dataset to demonstrate the results of our approach in section 3.5, and conclude in section 3.6.

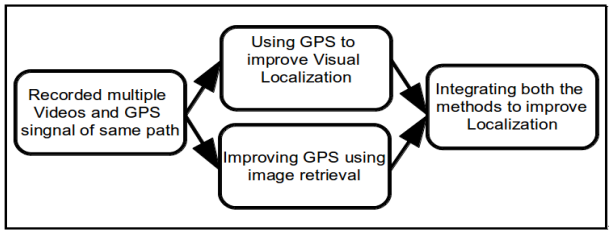
3.1.1 Related Work

Unreliability in Gps tags critically affects many computer vision tasks like 3D reconstruction and localization for shorter range [29, 30]. This unreliability in Gps has been addressed in several previous works[6]. This includes the use of additional cues such as wifi strength [31], additional special purpose hardware [32] or algorithms that learn error patterns [33]. Vision based methods have been used for Gps tag refinement. Most of the approaches often require a reference point (e.g. Street View) or dataset with pre-assumed correct Gps tag in case of vision based refinement and multi sensor input in case of Kalman filter algorithms [28]. Zamir *et. al.* [34] propose a self-refinement process, that has an internal noise reduction and robustness mechanism which effectively uses initial noisy Gps tags of the images to give refined values. Accurate localization is critical to many robotic applications [35, 36].

Visual localization is the problem where the location of a query image is identified by comparison with location-tagged images in a database. Its a challenging problem because images of even the most common scenes like urban environments show wide diversity in appearance. They can vary on different parameters e.g. different viewpoint, scale, occlusion, illumination etc. than the prior images in the database. For this paper we refer occlusion as blocking of the camera view because of non permanent objects. An occlusion in foreground (refer Figure 3.1) or similar looking images for example, could degrade visual localization. Performance evaluation in visual localization is measured as the Euclidean distance between the Gps tags of query image and retrieved images [35]. Due to inaccuracies in Gps



(a)



(b)

Figure 3.2: (a) Problem Statement: We want to simultaneously use the noisy Gps signals and erroneous visual localization to generate accurate localization. (b) The block diagram of the proposed method.

devices people integrate other sensor data like IMU, wheel odometry, and LIDAR sensors [32,37] to get an accurate localization.

3.1.2 Contributions

In this work we propose a method which localizes with an accuracy of 7.5m by fusing vision and Gps together. In this regard, we address 3 main challenges in visual and Gps localization: (i) *Perceptual aliasing*, when similar-looking images have very different Gps locations, (ii) *camera occlusion* (Figure 3.1), when dissimilar images are co-located, and (iii) noisy Gps data. To do this, we present an approach to learn the useful feature [38, 39] to improve the localization performance, along with an approach to correct noisy Gps outputs using visual localization [34].

Dataset: To do experiments in this paper, we collect 10 video datasets using a Contour action camera, while walking along a 600m path repeatedly over 10 days. We extract images from these videos at `10fps` or `1fps` depending on the requirement. We then extract SIFT features and store them for each

frame. While processing one video, images from the other 9 are used to build the visual bag of words vocabulary for image retrieval.

3.2 Use of GPS for Better Visual Localization and Extracting Useful Features

Visual localization problem is often formulated as an image retrieval problem. To achieve this visual features are extracted and clustered to form a visual vocabulary. Bag-of-Word model represents the image database as unordered set of visual words in the form of "inverted index". Inverted index is represented as a $(key, value)$ pair where key is the visual word index, and value is list of the images in which it appeared with their corresponding reliability weights. We identify the visual words in a query image using standard techniques [26]. With the help of visual words and inverted index score of n^{th} retrieved image is computed as:

$$Score(img_n) = \sum_{z_k \in Z_q} W_k^n \tag{3.1}$$

where Z_q is the set of feature descriptors in query image and W_k^n is the reliability weight of visual word corresponding to the z_k feature descriptors in n^{th} image. The image with highest score is chosen as the best matching image corresponding to the query image. Due to occlusion while extracting the features from an image many noisy features are extracted. This includes features generated around unstable interest points. Noisy feature rejection is motivated by the fact that occlusion and unstable object features will be likely to exist in a single image, while useful features are likely to be found in multiple images of the same object or location. Identifying the features which are robust to change in view can be determined by tracking which features exist in multiple views and are geometrically consistent with one another. This requires a minimum of two views, assuming that the object or location exists in the database prior to the useful feature extraction stage.

Extracting Useful Features and Role of GPS: To determine the useful image features Bag-of-words of image database containing the full feature set is constructed. Each image in the database is used as a query and the top k images are retrieved. All the images in the database have Gps tags associated with them. In order to avoid the perceptual aliasing and camera occlusion we considered only those images which lie in the radius of distance d from the query image. Inliers are then used to estimate epipolar geometry and only features which are geometrically consistent labeled as useful features. We perform experiments for d = 10,20 and 30 meters and show how it effects visual localization. For details of these experiments please refer sub-section 3.5.1. A sample result of useful feature extraction is shown in Figure 3.3. Note that the image has more than 70% non-distinctive area with occlusion. Our approach has filtered out almost all the noisy features generated around transient object and non-distinctive area. Figure 3.4 describes the flow diagram of the proposed method.

(a) (b)

Figure 3.3: Original image features (a) vs those features which could be considered useful features (b). Transient objects, occlusions in the foreground and non-distinctive areas of the scenes are found to be without useful features.

3.3 Improving GPS Signals through Image Retrieval

Zamir *et. al.* [34] proposed a self refinement method to refine user-specified Gps tags of images. We extend their work and apply it to discrete noisy Gps signals in order to get more accurate and consistence Gps signals, which we term refined Gps signals.

3.3.1 Details of Algorithm

We have a set S = $\{(V_1, G_1), (V_2, G_2), ..(V_n, G_n)\}$ where V_i and G_i is the i^{th} video and it's corresponding Gps signal. Each Gps signal G_i has a noise attached with them $G_i = \hat{G}_i + \eta$, where $\hat{G}_i$ is the refined signal and η is the noise attached to G_i. Our goal is to extract out $\hat{G}_i$ in a self-refinement manner without using any external reference point.

We sample each V_i at `1fps` and use each frame as query image $\mathcal{I}$ against the rest of the frames from the set $\{$S-$V_i\}$, and retrieve the top μ matches $\{m_1, m_2, ..m_\mu\}$ for each $\mathcal{I}$. We use SIFT Bag-of-Words with vocabulary size of 1 million for the purpose of image retrieval. We then form $\binom{\mu}{2}$ triplets from each query image & each pair of database images and estimate the relative location of the triplet by using Bundler for camera localization [40]. For each triplet $\{\mathcal{I}, m_i, m_j\}$ we get $\{l_\mathcal{I}, l_i, l_j\}$ which are camera locations of $\mathcal{I}, m_i$, and m_j in the SfM local co-ordinate system. However, we note that since in most images the relative height of the camera w.r.t the ground is the same, we could transform these

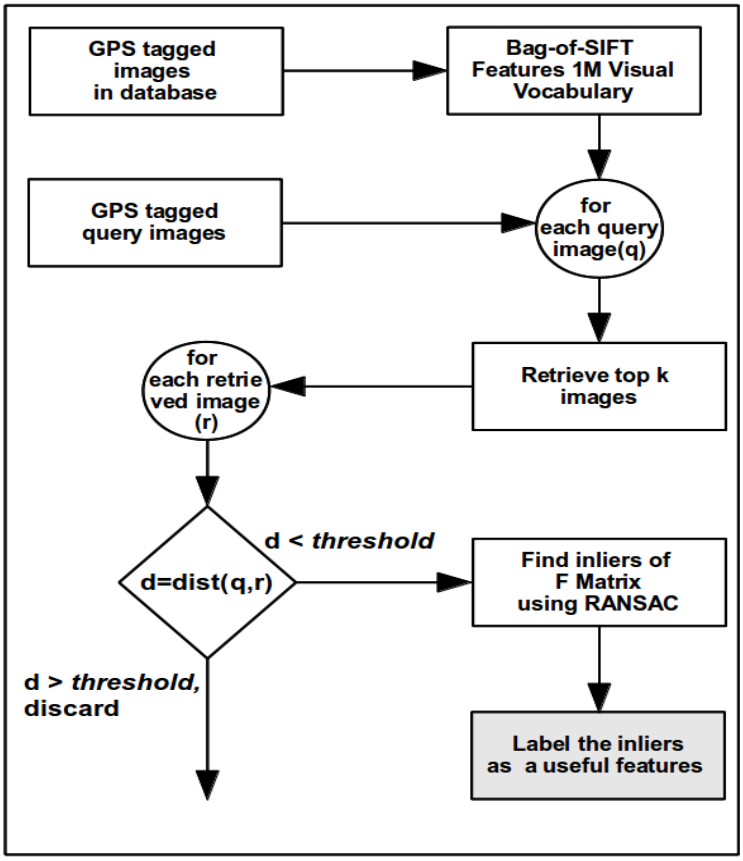


Figure 3.4: Flow chart to extract the useful features. For each query image we retrieve top k images using SIFT BoW. If the geographical distance between the query image and the retrieve image is less then a threshold then we find the inliers between tag them as a useful feature.

3D vectors into 2D vectors. Thus, applying an assumption that video tracks were recorded roughly on a planar surface we can reduce the dimensionality of $l_\jmath, l_i$ and l_j to two (e.g. using PCA).
Our aim is to calculate the Gps of $\jmath$ using the image triplet $\{l_\jmath, l_i, l_j\}$. To do this, the locations $\{l_\jmath, l_i, l_j\}$ should be mapped from SfM local co-ordinate system to the global Gps co-ordinate system. These two Cartesian co-ordinates are related through a similarity transformation matrix RST.

$$\begin{bmatrix} g \\ 1 \end{bmatrix} = (\mathbf{RST}) \begin{bmatrix} l \\ 1 \end{bmatrix} \tag{3.2}$$

where l is a point in the SfM coordinate system and g is it's corresponding point in global Gps co-ordinate system, $\left[\begin{smallmatrix} g \\ 1 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} l \\ 1 \end{smallmatrix}\right]$ are homogeneous co-ordinates of g and l. RST is denoted by 3×3 matrix. We need at least two pairs of $g \leftrightarrow l$ correspondence in-order to calculate the RST matrix from Equation 3.2. In each triplet m_i and m_j are Gps tagged, we use their Gps tags and their locations l_i and l_j to compute RST of the triplet. Now this transformation is used for finding the location of $\jmath$ in global

Gps co-ordinate system. Since we have $\binom{\mu}{2}$ possible triplets, we will get $\binom{\mu}{2}$ possible Gps estimates for a query image.

3.3.1.1 Robust Estimation through Random Walks

The estimated Gps locations of $\mathcal{I}$ yielded by the triplets is accurate only if the Gps tag of reference images m_i and m_j is accurate. We use Random Walks on estimated triplets to discover the reliable subset of estimations. We define a graph $\mathcal{G}$ = (N,E) where N and E represent the set of node and edges. Each node represents one estimation, i.e. N = $\{g_1, g_2, ...g_\lambda\}$, and there is and edge between each pair of nodes, E = $\{(g_i, g_j), i \neq j\}$. We include the original Gps tag of $\mathcal{I}$, for the estimation of its correct Gps -location, in set N. The transition probability from node i to node j according to their Gps distance is calculated using Equation 3.3 given by Zamir *et. al.* [34]:

$$p(i,j) = \frac{e^{-\sigma||g_i - g_j||_2}}{\sum_{k=1}^{\lambda} e^{-\sigma||g_i - g_k||_2}} \tag{3.3}$$

where $||.||_2$ denotes the l_2 norm, λ is the number of node in graph $\mathcal{G}$ and σ is call insensitive parameter.

Image Geo Density and Initial Node Score: The purpose of image geo density is to handle the phenomenon of non uniform distribution of images across popular sites like flickr, facebook, etc ... In our case, where we have multiple videos of the same path, we can assume uniform distribution of images. The initial score of the nodes is not going to be effected by geo-density, hence initial score of the n^{th} node $v(n)$ can be calculated as:

$$v(n) = \frac{1}{\lambda} \tag{3.4}$$

3.3.1.2 Adaptive Damping Factor

For a each query image we got a graph $\mathcal{G}$ having λ nodes where each node is initialized with score $v(n)$. The Random Walks algorithm will update the score of one node at every iteration using the transition probability from other nodes to it. Equation 3.5 is the basic Random Walks formula

$$x_{(k+1)}(j) = \sum_{k=1}^{\lambda} \overbrace{\alpha}^{①} x_k(i) p(i,j) + \overbrace{(1-\alpha)}^{②} v(j) \tag{3.5}$$

where x_k(i) is the relevance score of i^{th} node at k^{th} iteration. The use of the damping term in Random Walks is to use the prior knowledge about the relevance of nodes and to ensure irreducibility of the transition probabilities matrix which is a convergence condition for Random Walks [41]. The summation of term ① and term ② in Equation 3.5 should be 1 because the relevance score at any iteration must sum to one, *i.e.*, $\sum_{k=1}^{\lambda} x_k(i) = 1$. Zamir et al. [34] further propose an adaptive damping factor which adaptively changes according to the consistency of each node w.r.t others. They formulate the damping

term of a node as a function of its relevance score at each iteration:

$$x_{k+1}(j) = \frac{1}{\eta}\Big(\sum_{i=1}^{\lambda} \overbrace{\big(1-(1-\alpha)x_k(j)\big)}^{①} x_k(i)p(i,j) + \overbrace{(1-\alpha)x_k(j)}^{②} v(j)\Big) \tag{3.6}$$

The normalization constant η given by Equation 3.7 forces the sum of all relevance scores to be one.

$$\eta = \sum_{j=1}^{\lambda}\Big(\sum_{i=1}^{\lambda}\big(1-(1-\alpha)x_k(j)\big)x_k(i)p(i,j) + (1-\alpha)x_k(j)v(j)\Big) \tag{3.7}$$

Estimation of Final GPS-Tag using the Relevance Scores: The estimations which are badly effected by noise are expected to have relevance score of ≈ 0, other nodes should gain the score based on their transition probability and initial score. Finally we compute the refined Gps -tags of the query $\mathcal{I}$, utilizing a weighted mean using the relevance scores x_π.

$$\hat{g} = \sum_{i=1}^{\lambda} g_i x_\pi(i) \tag{3.8}$$

where $\hat{g}$ is refined Gps -tags.

3.4 Improving the Localization

Integrating Visual Localization and GPS Refinement: In section 3.2 we have shown how with the help of Gps we can overcome issues like occlusion and perceptual aliasing in order to improve visual localization using image retrieval, whereas section 3.3 describe how to reduce the error in Gps signal with the help of image retrieval and random walks. As mentioned above, we integrate both modules to improve localization in challenging scenarios like over short distances where Gps signals tend to fail. First we improve Gps signals and label all the images in the database with refined Gps tags. Using these refined Gps tags we filter out the useful features for building a vocabulary and inverted index.

Sequential Localization: Without any loss of generality, we can assume that the motion of any portable device with Gps is a smooth motion. So we can assume such motion satisfies the Markov assumption *i.e.* that is the current pose X_t is dependent only on the previous pose X_{t-1}. Our sequential visual localization method consists two phases: initial localization phase and query retrieval phase. In initial localization phase we keep on retrieving relevant images to the query image until estimation of the pose is known to us with minimum probability P^*. P^* is calculated using Algorithm 1. Once the initial pose X_t is fixed we use temporal sequence property to fix the pose of X_{t+1} in query retrieval phase. In

section 5 we have demonstrated the effect of useful features, refined signal as well as combined effect of both on localization by conducting different set of experiments. We also performed the sequential localization after integration.

Algorithm 1 Estimation of initial pose

Input: Bag-of-Word frame work, Q: Queue
Output: Initial pose with minimum probability P^*.
for i=1 $\rightarrow N$ **do**
    $r_i \leftarrow$ Retrieved image using query image q_i.
    if Q is NULL **then**
        $Q \leftarrow r_i$
    else
        for j=1 $\rightarrow$ Number of element in Q **do**
            $dist$ = calEculideanDistance(r_i,Q_j)
            if $dist < thresholdDistance$ **then**
                Score(Q_j) = Score(Q_j) + 1
            else
                $Q \leftarrow r_i$
            end if
        end for
    end if
end for
P^* = arg max(Score(p_k) / N ; where $p_k \in$ Q
Return the pose corresponding to p_k

3.5 Experiments and Results

In this section, we demonstrate the utility of our approach with quantitative experiments. We capture the videos using a Contour action camera [42] with resolution 1920×1080 at $30 fps$. The device also has an inbuilt Gps sensor which recorded the corresponding Gps signal at $1Hz$. This gives us two different loosely aligned signals Gps and videos. Since we are also interested in studying the utility of multiple runs captured over time, we collect the videos on ten different days by walking on the same $600m$ long path. Data is captured in the evening at peak traffic hours to ensure approximately same crowded urban environment setting. This process yields ten videos of the same path with labelled Gps tags.

For vision based localization, we use bag of words representation with 1 M visual words built on top of dense SIFT descriptors. During all the experiments, we have ensured that the test data is not used for the vocabulary construction. Wherever, multiple runs are used, one run is used for the evaluation in the leave one out manner.

As can be seen in Figure 3.5, our sensor fusion scheme is yielding superior estimates of the localization. One may notice the noisy (zig zag) path of the original localizations and the more or less smooth

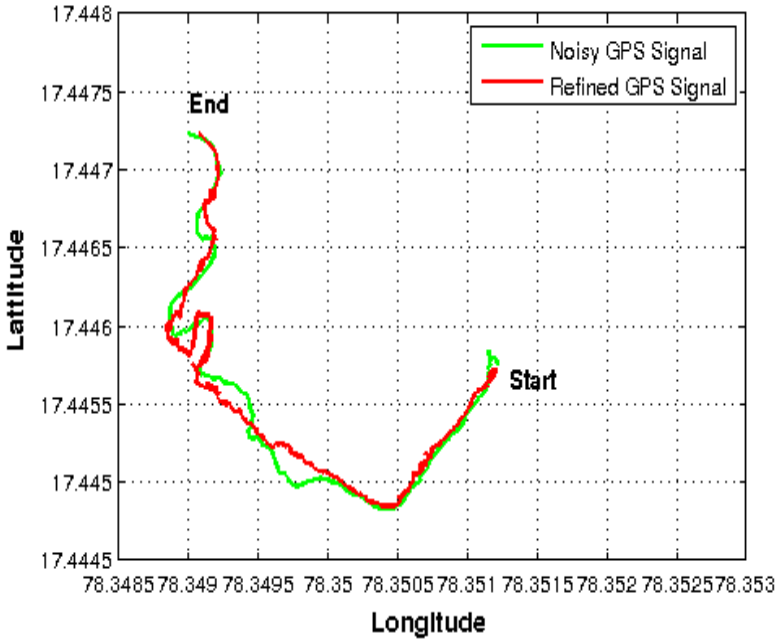


Figure 3.5: Plot demonstrating the noisy and refined Gps signals. One can observe refined Gps signal (red) is less random and in zig-zag shape as compare to noisy Gps signal (green).

path estimated after the multi sensor localization as we discussed in the previous section. In the rest of this section, we demonstrate the quantitative results of various experiments.

3.5.1 Effect of Useful Features on Visual Localization

First, we demonstrate the utility of feature selection in improving the accuracy of localization. Effect of useful features on visual localization performance was evaluated using K-fold cross validation with number of folds set to 10. For each of the query image we retrieved an image from Bag-of-Words model build by using useful features only. If the Euclidean distance between retrieved image and query image is greater than δ_d distance it is consider as "invalid localization". The percentage error in the visual localization is define as:

$$P_e = \left(\frac{TotalNo.OfInvalidLocalization}{No.OfQueryFrames}\right) \times 100 \tag{3.9}$$

In this experiment we have set the $\delta_d = 10m$. Table 3.1 shows the comparative percentage error in visual localization between two approaches. In the table experiment E_i means i^{th} video is used as the source for query images and rest all are use in building visual vocabulary. One can observe significant drop in error while using useful features for visual localization. The other observation that can be made is, the drop in percentage error P_e is less for increasing values of d. It is because of number of inliers are almost constant with varying d. Hence we set $d = 10m$ for further experiments to expedite the process without compromising accuracy a lot.

Table 3.1: Effect of useful features on visual localization. There is a significant drop in percentage error in visual localization with useful features. As d increases the increase in number of inliers are very less. Therefore useful features are almost constant for different values of d. Hence the drop in P_e error is less. NA=Not Applicable, OF=Original Features, UF=Useful Features as shown in Figure 3.3.

Exp. No.	P_e, *d=NA* *OF*	P_e, *d=10* *UF*	P_e, *d=20* *UF*	P_e, *d=30* *UF*
E_1	31.63	23.56	22.69	22.57
E_2	31.70	23.47	22.65	22.59
E_3	29.49	22.21	21.49	21.52
E_4	32.06	23.93	23.03	22.98
E_5	30.54	22.89	22.15	22.15
E_6	31.98	23.07	22.26	22.23
E_7	31.04	22.87	22.15	22.09
E_8	30.68	22.72	22.13	22.07
E_9	31.75	23.47	22.70	22.58
E_{10}	32.30	23.72	22.63	22.56

3.5.2 Refined vs Noisy GPS Signals RMSE Score

The recorded GPS signal has noise associated with it. To motivate, if one log the Gps signal while keeping the device stationary, one can see large variation in Gps values up to 35m (as shown in Figure 3.1). The term noisy Gps signals refer to the Gps signals which were recorded using Gps receiver. By improving the Gps signals using images (section 3.3) we get refined Gps signals.

To refine a Gps signal the corresponding video run was used as the query video in the vision based localization. We do this for each of the videos separately. Using all these attempts, we refine the Gps tags and finally we integrate all the GPS tags to get the refined Gps signal.

To quantitatively compare, we used Root Mean Square Estimate(RMSE) values. We assumed one of the Gps signal as a ground truth and calculated the RMSE for other signals with respect to assumed ground truth. RMSE for the G_n Gps signal is given as:

$$RMSE(G_n) = \frac{\sqrt[2]{\sum_{i=0}^{N} \Big(\textit{Dist}(G_{gt}(i), G_n(i))\Big)^2}}{N} \tag{3.10}$$

where $N = \min\big(length(G_{gt}), length(G_n)\big)$ as all the signals are approximately same length, so discarding few values will not effect the RMSE much, $G_{gt}(i)$ and $G_n(i)$ is i^{th} Gps reading of assumed ground truth signal and n^{th} Gps signal respectively. *Dist* calculates the Euclidean distance between two Gps measurements. Mean RMSE of the n^{th} Gps signal is the mean of all the RMSE values taking n^{th} signal as a ground truth. We calculated RMSE values for noisy Gps signals and refined Gps signals separately. Table 3.2 shows the comparison between Mean RMSE values of noisy and refined Gps signals for each video. S_n indicates that n^{th} signal was taken as ground truth to calculate the Mean RMSE. The Mean RMSE for the noisy Gps signal is ≈10m whereas for refined Gps signals it comes down to

≈6-7m. This demonstrates that the sensor fusion is resulting in a more consistent localization results than using a single sensor alone.

Table 3.2: Mean RMSE comparison of noisy and refined Gps signal. For refined Gps signal RMSE values ≈7m.

Run No.	*RMSE Noisy Signal*	*RMSE Refined Signal*	*% drop in RMSE Error*
S_1	9.81	6.76	**31.10**
S_2	10.55	7.36	**30.23**
S_3	10.49	7.01	**33.17**
S_4	9.76	6.73	**31.05**
S_5	9.54	6.79	**28.82**
S_6	10.31	6.98	**32.30**
S_7	10.21	6.92	**32.22**
S_8	10.01	6.87	**31.36**
S_9	10.65	7.01	**34.17**
S_{10}	9.96	6.95	**30.22**

3.5.3 Comparison of Denoising using Synthetic Noise

In another experiment to test the method in various extreme circumstances we added random Gaussian noise as an input error with mean values 100, 500, 1000, 2000, 3000 and 4000 meters to 5, 10, 20, 33 and 50 percent of the 24648 images in our dataset and the standard deviation was set to the half the mean to replicate the Gps device error with $mean = 12.23m$ and $standard\ deviation$ = 5.98 (Figure 3.1). We improvised the synthetic error on the top of the already existing noise. Hence, the additional synthetic noise determines the lower bound of noise since the exact amount of error in the dataset is unknown. We also made sure that in this experiment, the query images were among the ones with contaminated Gps tags to ensure the evaluation fair and challenging. We got similar results like Zamir *et. al.* [34](Figure 3.6). We observed for the contamination percentages less than 33%, the method almost completely eliminates the error regardless of the mean of the contamination in the input. Once the input error increase beyond the 33% and 50% the error in output is no more avoidable yet still less than the error in input.

3.5.4 Effect of Refined GPS Signal On Localization:

We also performed the experiments to study the effect of refined Gps signal on localization. The ten videos sampled at $10fps$ and tagged each frame with their corresponding noisy and refined Gps tags. Bag-of-Word framework was build using SIFT features from the nine runs with visual vocabulary set to 1 million words. We performed visual localization using one of the video run as a query dataset. Percentage error in visual localization for distance was calculated using Equation 3.9. Table 3.3 shows

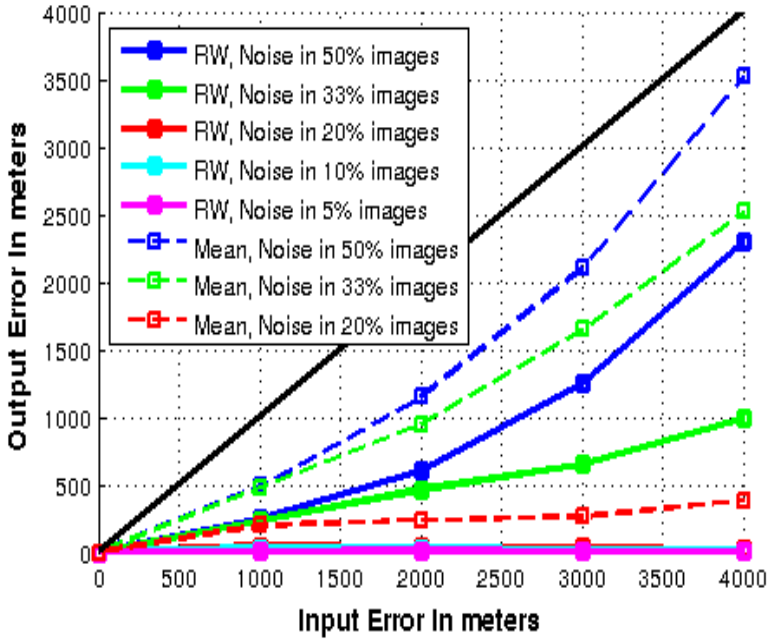


Figure 3.6: Plot is demonstrating the synthetic noise reduction process and comparison between Random walks and simply taking the mean. It is clear from the graph as contamination percentage increase Mean is not that much efficient comparison to Random walks in order to remove the added noise.

the comparison in visual localization performance between refined and noisy Gps signal for different δ_d distances. An experiment E_i means i^{th} run was used as a query run(different query run contains 4000-4193 frames) and rest all other for vocabulary building. From the Table 3.3 one can also infer that for δ_d = 7.5m the difference in the P_e error for noisy Gps signal and refined signal is huge, but as δ_d increases from 10m and beyond P_e is approximately same for both kind of signals. Variance in the P_e for refined Gps signal less than noisy Gps signal as standard deviation are 9.67 and 6.70 respectively for the noisy and refined signal.

3.5.5 Multi Sensor Localization

We integrated both the modules *i) Use of* Gps *for Better Visual Localization* and *ii)Improving the* Gps *Signal Through Image Retrieval* to form a pipeline for more accurate localization. First we refined the Gps signals using images and adjusted the Gps tags therein to the correct locations. Then we improved the visual localization by labeling the useful features with the help of refined Gps . The pipeline was tested on ten video runs, where each runs contains 4000-4193 frames. Nine runs were used to build the Bag-of-Word frame work for image retrieval having vocabulary size of 1 million visual words. All the frames from a video run were used as query dataset. Percentage error in visual localization was calculated by the Equation 3.9. We localize for the distance δ_d = 7.5m. We also performed the sequential localization by fixing the initial position with Algorithm 1. In the Algorithm 1 we set N =

Table 3.3: Effect of refined signal on percentage error. With small value of δ_d there is significant difference in the performance. As δ_d increases the performance gap is narrowed down.NS = Noisy Signal, RS = Refined Signal.

Exp.No.	P_e, δ_d=7.5m		P_e, δ_d=10m		P_e, δ_d=15m	
	NS	RS	NS	RS	NS	RS
E_1	45.98	29.68	31.63	23.11	16.05	15.33
E_2	40.00	33.68	31.73	30.73	18.53	20.00
E_3	32.00	27.85	24.09	24.52	16.19	14.28
E_4	33.00	24.50	25.24	21.28	13.61	10.00
E_5	37.12	29.09	29.43	22.19	16.30	14.80
E_6	39.00	28.67	28.84	25.05	17.46	15.09
E_7	38.85	28.72	28.19	24.23	16.57	14.91
E_8	37.18	28.99	27.79	25.45	16.47	15.00
E_9	40.49	29.81	30.35	23.52	15.82	14.73
E_{10}	36.72	30.01	28.80	24.74	16.14	14.94

10($i.e$ re-sampling frequency of the videos) and $thresholdDistance$ = δ_d and P^*=0.7. Table 3.4 and Figure 3.7 shows significant drop in P_e with multi sensor localization method. Our proposed method is useful to perform localization for a shorter distance($\approx 7.5m$) by fusing images and noisy Gps tags obtained by using any commercial Gps receiver.

Table 3.4: Comparison of percentage error P_e in localization with δ_d =7.5m while using the methods i)Bag-of-Words frame work ii)Bag-of-Words frame work + Integration of modules iii)Bag-of-Words frame work + Integration of modules + Sequential Localization.

Exp. No.	*Simple BOW*	*BOW Integration*	*BOW Seq. Localization*
E_1	31.63	8.09	2.97
E_2	31.70	8.03	2.73
E_3	29.49	7.10	2.14
E_4	32.06	10.03	3.30
E_5	30.54	7.60	2.71
E_6	31.98	9.07	2.85
E_7	31.04	8.10	2.56
E_8	30.68	7.71	2.60
E_9	31.80	8.29	2.12
E_{10}	31.44	8.56	2.82

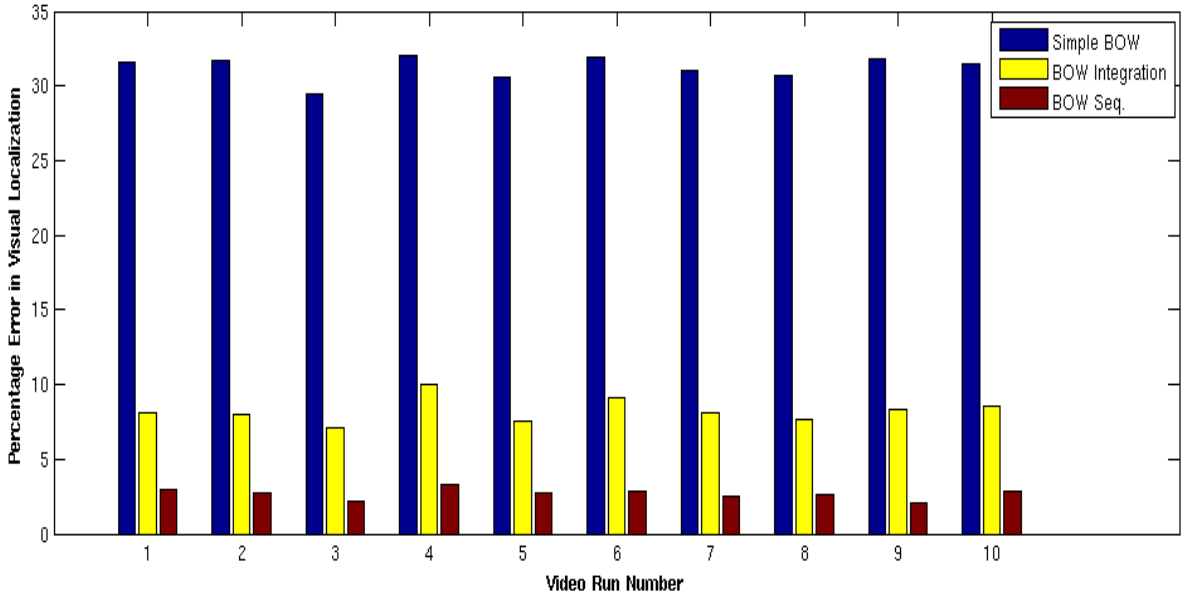


Figure 3.7: Bar graph visualization of percentage error in localization using different methodologies.

3.6 Summary

This work proposed a multi sensor based localization scheme that fuse two popular localization schemes to result in a more accurate localization strategy. The objective of our work has been to fuse Gps signals and images together in order to improve the localization. We propose methods that use noisy Gps to improve the vision based localization. We also propose methods that use vision based localization to improve the Gps signals. Finally we use these two steps in an iterative manner to get further improvement. We also presented a set of experiments to validate the fusion schemes and thereby the improvements in localization by fusing image and Gps .

Chapter 4

Modeling User Activity for Conserving Power on Smartphones

Mobile consumer-electronics devices, especially phones, are powered from batteries that are limited in size and therefore capacity. This implies that managing the battery power is paramount in such devices. As long as the battery technology continues its slow pace of improvement, the only viable approach is to reduce the amount of energy required to provide specific services. Two of the most power consuming services on a smartphone are network and wireless data. Though internet connectivity is important, the nature of data transfer does not require uninterrupted service. We take advantage of this fact to cut off power to these modules when it is not required. The challenge is to predict, when the users require these services and when they do not. Existing solutions usually schedule the services based on fixed hand-coded rules that do not work for all users or usage scenarios. In this paper, we propose an activity-based model that uses machine learning approaches to intelligently schedule Wi-Fi according to a user's activity level. Several higher order features from the raw sensor data stream are used to classify the activity level. Two aspects of the problem are considered, namely the accuracy of estimating the activity level as well as the power required in sensing and estimation. Experimental results on Android based smartphones demonstrate that an active user can get up to 37% increase in battery life without significant effect on the user experience.

4.1 Introduction

A smartphone is a mobile phone built on a mobile operating system, with higher computing capability and connectivity options compared to a feature phone. Inclusion of features such as accelerometers, GPS, compass, radio receivers, cameras and audio processors have made smartphones the primary device for personal assistance including communication, navigation, entertainment, imaging and information access. By far the most significant problem with most smartphones is their highly limited battery life. In recent times, hardware manufacturers have worked extensively to solve this problem, resulting in highly energy efficient processors and some promising directions in power-efficient displays. However, the wireless communication modules continue to consume significantly more energy in comparison and is a potential source for energy savings.

Today's internet is the mobile internet. People are moving towards smart-phones for general browsing, social networking and emails etc. Figure 4.2 shows the percentage of phone owners who used internet or email on their phone. 56% of smartphone users use their phones, not their PCs or tablets, as their main way to access the internet. Thus reducing the energy consumed by the network processes like Wi-Fi on smartphone devices could help in huge energy savings.

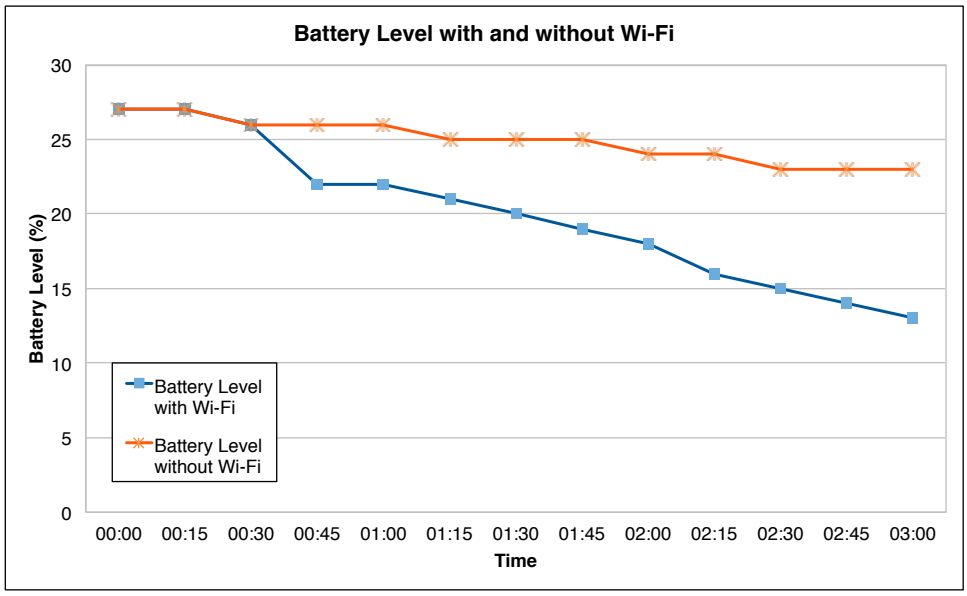


Figure 4.1: Battery level variation on a smartphone with and without Wi-Fi enabled (Data transfer: 2KB at 5 minute intervals).

An interesting nature of wireless data communication arises the connectionless protocols used for data communication. The data is buffered at different points in the communication path and is delivered whenever a connection to the device is available. Moreover, most usage patterns involve intermittent periods of activity and inactivity. We take advantage of this fact to switch off the wireless data radio, whenever the usage is not significant. This would result in significant savings of power is used appropriately. However, there are two challenges that need to be addressed to achieve this. We need to be able to predict periods of activity and inactivity of the specific user to shut down the wireless appropriately. Inaccuracies in this prediction will result in a poor user experience or a lost opportunity for power saving. On the other hand the data collection for determining the user state and the classification process should be simple enough to avoid excessive energy usage for its purpose, nullifying any savings resulting from powering off the wireless.

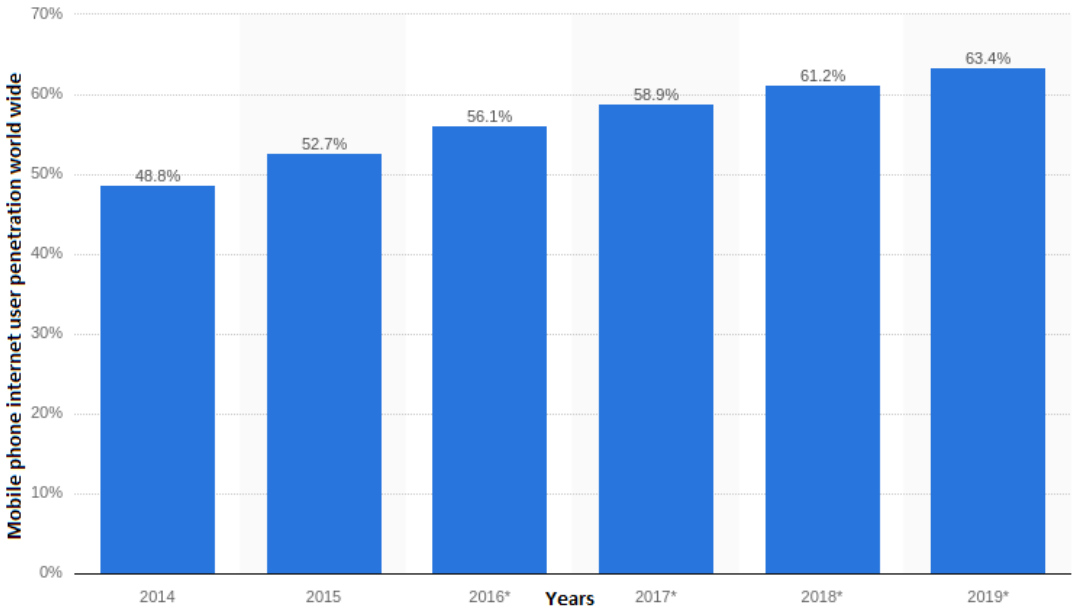


Figure 4.2: World wide percentage of mobile phone users and prediction, who use internet or email on their phone in $2014 - 2019$[1].

4.1.1 Contributions

In this work, we look at modeling and monitoring the user activity level(via machine learning) to decide when the wireless data modules should be turned off for maximal energy saving without compromising the user experience. The Wi-Fi scheduling is formulated as a classification problem, where Wi-Fi is scheduled based on the predicted usage of the user in the next time window. The specific problems that we deal with in this work may be summarized as:

- *Activity Modeling on Smartphone:* We propose a power-efficient approach to modeling the user's activity level on the smartphone device using various sensors and parameters that are easily available on most smartphones.

- *Energy Reduction by continuous Activity Scheduling:* We proposed an approach to schedule continuously running processes like Wi-Fi for better energy management without hampering the user experience.

We first take a look at the existing works in the area of activity classification and energy efficient mobile computing to understand the popular approaches in this area, which are then leveraged in coming up with an efficient way of activity modeling.

4.1.2 Related Work

Mobile phone based pattern recognition and data analysis has been a hot research field in the last decade. Specifically, the area of modeling user behavior to provide location, time and need based services has seen significant attention. Most smartphones are filled with a variety of applications that use sensors to measure movement and understand the world around us. They can be programmed to teach itself to work better, based on how you use it, the applications you run, and how you use it to communicate with others.

A lot of work has also been done on activity and gesture recognition with the help of accelerometer sensor data. Sun *et al.* [43] proposed a mechanism to classify physical activities of the bearer of the device using accelerometer data with varying positions and orientations. The embedded tri-axial accelerometer in a mobile phone can continuously log the experienced forces at each sampling interval and produce 3-D acceleration reading A = (a_x,a_y,a_z) along three orthogonal axes. In addition to the raw sensor data, several features derived from the accelerometer sensor including mean, variance, correlation, energy, frequency-domain entropy, cepstral coefficients, and log-FFT frequency bands [44–51]. Further, Longstaff *et al.* [52] proposed different health related applications via semi-supervised activity modeling on smartphone. Realizing the effectiveness of accelerometer data in determining various usage parameters, mobile phone manufacturers have started embedding dedicated low-power motion sensor processors in addition to the main CPU on smartphones and tablets. In this work, we use accelerometer based features that were proved good by most of these methods for modeling the user activity. Along with accelerometer data, we also took other important indicators that suggest the type of user activity such as user interaction, and CPU and RAM usage on the device. A detailed description of the specific features that are used is given in Section 4.4.1.

Activity traces can improve the user interface mechanisms across a range of applications. For example, they can be used as automated reminder or triggered for an action or input at a convenient moment [53]. Recently, researchers have started applying various machine learning approaches to activity recognition on smartphones. Yan *et al.* [54] proposed an efficient continuous activity recognition system. They tried to model continuous user activity in a Markovian model based on the currently recognized activity. Zhuang *et al.* [55] looked at the problem of energy-efficient location sensing on a smartphone. They tried to find the location using substitution, suppression, piggybacking and adaptation methods, thus avoiding the use of power intensive sensors like GPS for getting the location. Unlike the above works that tries to do a specific task in an efficient manner, we use a combination of activity recognition and intelligent scheduling to save power dissipation by the most power hungry processes on a typical smartphone. Towards this, we present a more detailed analysis of the aspects of the phone that consumes the most amount of power (in Section 4.2). This is followed by a description of our approach and the activity classification model. In Section 4.5, we provide the results of various experiments comparing our approach with the base case of regular usage profiles.

4.2 What consumes Power?

Different applications have different power requirements. Figure 4.3 shows the relative power consumed by various services and applications on a smartphone device. It can be clearly observed that network services such as Wi-Fi, EDGE and 3G are most power consuming processes on the device, and of the three, Wi-Fi is the worst offender. However, in idle mode, Wi-Fi consumes very less power.

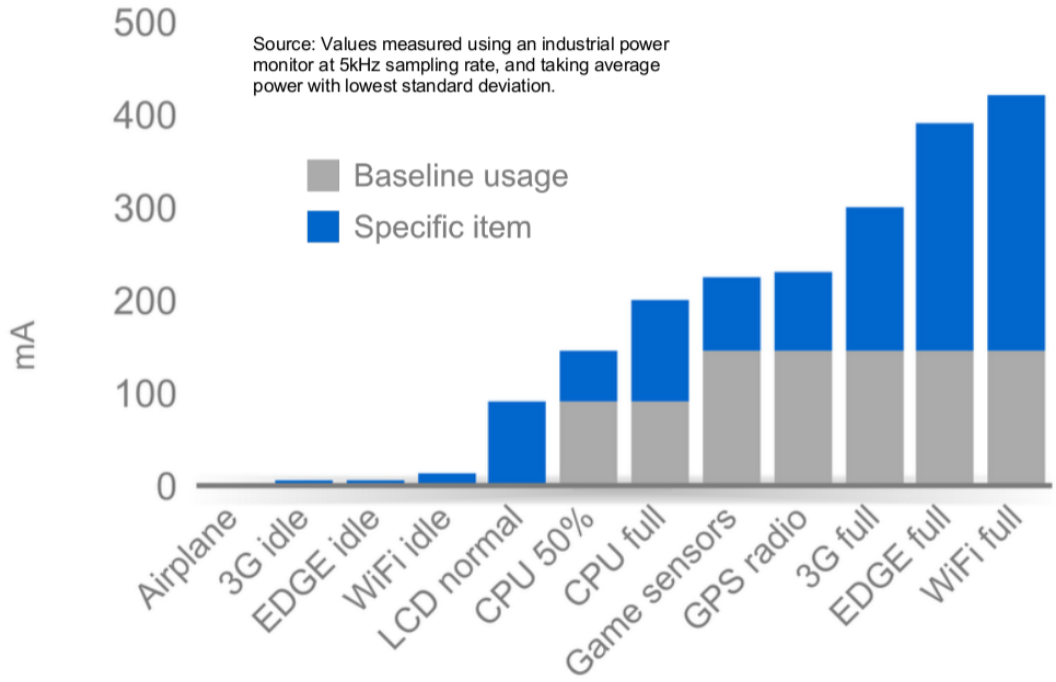


Figure 4.3: Component-wise energy consumption on Android (Jeff Sharkey [2]).

The power consumed by the network services also varies with the duration between data transfer operations. Figure 4.4 shows the power consumed by the service when bundled transfer mode is used, compared to unbundled data transfer mode. In our experiments, the energy consumed by unbundled transfer was up to 3 times that consumed by bundled transfer.

The best practices for the network applications is to fetch as much data as possible in one go. Google recommends pre-fetching data that will initiate in next 2 to 5 minutes, and in order of 1-5 MB. So in order to save the battery, batch mode should be used rather than on-demand mode for sending and receiving the data. We tried to use this practice to improve the energy efficiency in the mobile device. The data is scheduled in batch mode by intelligently switching between the Wi-Fi states. Data is fetched in bulk only when it is required by the user and the Wi-Fi is turned off when user is sleeping or inactive on the mobile device.

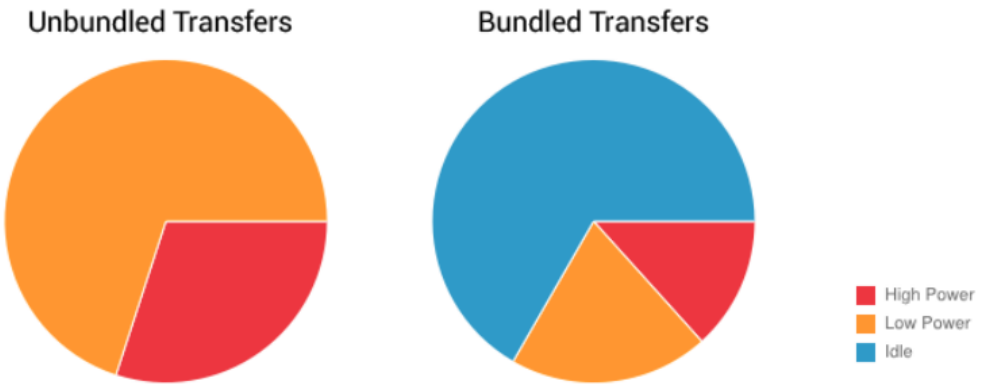


Figure 4.4: Duration of time spend under different states (high power, low power and idle) by applications under Bundled and Unbundled packets transfer mode (Kumar Rangarajana [3])

4.3 Wi-Fi scheduling as Multi class Classification

Internet services are high power consuming processes. So they need to be scheduled in a personalized way in order to save battery. We model users behavior to schedule the Wi-Fi state over time. The user current activity level is predicted and the model is scheduled for the next half hour. If the user is highly active on the device, the Wi-Fi is turned on. Conversely, if the phone is idle the Wi-Fi is turned off to save the battery. This way the power is conserved without much hampering the users experience. There are three main modules. The feature extraction module, the classification module and the scheduling module. Figure 4.5 shows a flow diagram of our approach. The feature extraction module creates the feature vector using the accelerometer sensors, screen on duration and CPU and RAM usage by the device. The classification module learns these features and tries to predict the Wi-Fi usage by the user in next half an hour. The classification module interprets the user's activity state as one of the following classes:

1. *Very Low* (l_0) : Corresponds to no or very low activity by the user in the last half an hour.
2. *Low* (l_1) : Corresponds to low activity by the user in last half an hour.
3. *Medium* (l_2) : Corresponds to moderate activity usage by the user in last half an hour.
4. *High* (l_3) : Corresponds to intense activity by the user.

Based on the user state prediction, the Wi-Fi in the device is scheduled in the following manner.

- Label l_0 turn off the Wi-Fi for next half an hour.
- Label l_1 turns off the Wi-Fi for next 25 minutes. The Wi-Fi is then turned on for the next 5 minutes.

- Label l_2 turns off the Wi-Fi for the next 20 minutes. The Wi-Fi is then turned on for next 10 minutes.
- Label l_3 turns on the Wi-Fi for the next half an hour.

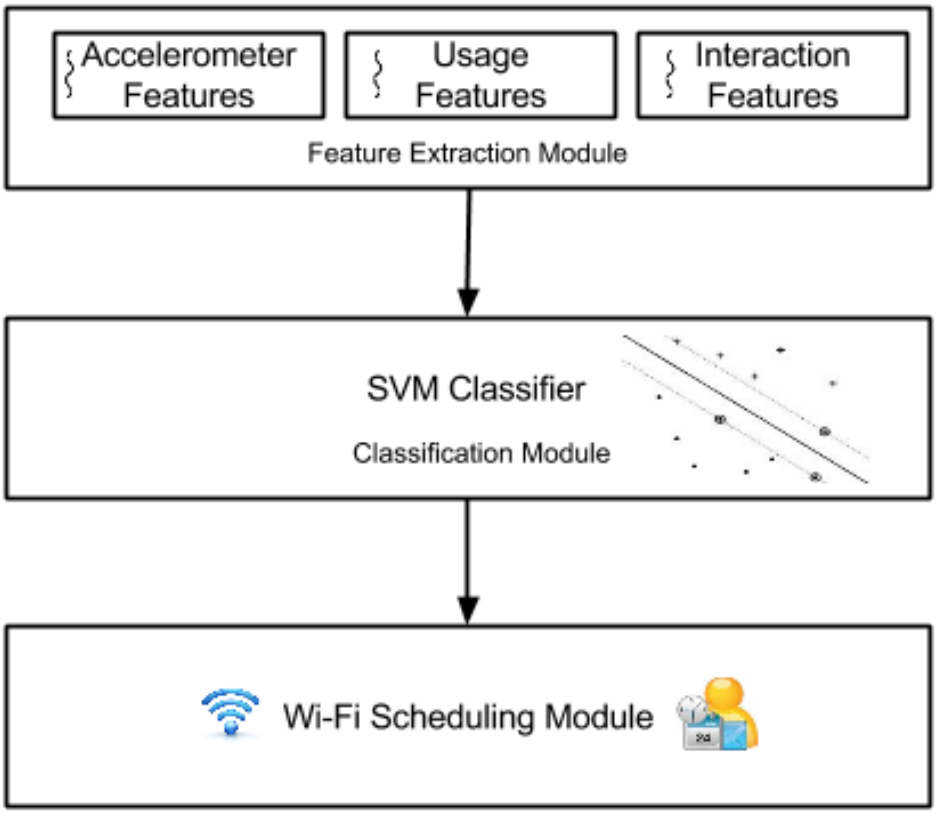


Figure 4.5: Primary steps in the algorithm with data flow. Feature extraction module extracts the Accelerometer based, usage based and interaction based features. Classification module predicts the user activity level using SVM Classifier and Wi-Fi Scheduling Module schedules the Wi-Fi state based on the predicted output.

4.4 Classification Frame Work

4.4.1 Feature Extraction

To extract the data from the smartphone, we created an Android application that records the required data. Data was collected over the period of half an hour to predict the users behavior for next cycle.

After each half an hour, the application starts a service to collect the data. The service runs for the duration of 30 seconds and executes the following processes.

1. Partial wake-lock is acquired to prevent smartphone from going to sleep during the process execution.
2. The required data is collected for the duration of 30 seconds.
3. The user activity label is predicted by the classification module from the collected data.
4. The Wi-Fi is scheduled for next half an hour based on the predicted label.
5. Wake-lock is released and phone is allowed to go to sleep.

We divides the features into three categories- accelerometer based, system usage based and interaction based features. These features are explained in details below.

4.4.1.1 Accelerometer Based Features

Accelerometer based features are indication of the physical activity of the user like stationary, walking, running etc. The assumption behind is if the user is running or walking the probability of using his phone is much less as compare to stationary state.

Accelerometer continuously samples the acceleration at the specified sampling interval and produces 3-D acceleration readings S = (S_X, S_Y, S_Z) along X, Y and Z direction. As the acceleration in a direction depends on the orientation of the phone, the overall acceleration S_ϕ = $||S_X, S_Y, S_Z||$ is also considered as feature. The sensor reading thus becomes a 4-D vector. Each device is calibrated differently. So we neglected the acceleration due to gravity while calculating the acceleration of the smartphone.

The accelerometer data is tracked for the user with the sampling frequency of $5Hz$. The accelerometer is tracked for 30 seconds and the following features are extracted.
Average mean along S_X, S_Y, S_Z and the S_ϕ vector.

$$\mu_d = \frac{\sum_{i=1}^{n} S_{d,i}}{n}; d \in \{X, Y, Z, \phi\}$$

Standard deviation along S_X,S_Y,S_Z and the S_ϕ vector. Standard deviation indicates the dispersion in the acceleration along different axis.

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^{n}(S_{d,i} - \mu_d)^2}{n-1}}; d \in \{X, Y, Z, \phi\}$$

Correlation among each pair of S_X, S_Y, S_Z and the S_ϕ vector. Correlation implies the strength between two corresponding axis signals.

$$r_{d,d'} = \frac{\sum_{i=1}^{n}(S_{d,i} - \mu_d) \times (S_{d',i} - \mu_{d'})}{\sqrt{\sum_{i=1}^{n}(S_{d,i} - \mu_d)^2 \times \sum_{i=1}^{n}(S_{d',i} - \mu_{d'})^2}};$$

$$d, d' \in \{X, Y, Z, \phi\}, d \neq d'$$

Fourier energy for S_X, S_Y, S_Z and the S_ϕ vector. The energy feature is the sum of all the squared DFT component magnitudes except the DC component of the signal as the DC component (mean) has been used as an individual feature. The magnitudes are divided by the number of components for normalization.

$$\varepsilon_d = \frac{1}{n} \sum_{i=1}^{n} ||\mathcal{F}(S_d)_i||; d \in \{X, Y, Z, \phi\}$$

Fourier entropy for S_X, S_Y, S_Z and the S_ϕ vector. Fourier entropy is the normalized information present in the DFT of the signal, excluding the DC component.

$$\delta_d = \sum_{i=1}^{n} ||\mathcal{F}(S_d)_i|| \times log(\frac{1}{||\mathcal{F}(S_d)_i||})$$

The intuition behind using accelerometer based features is that Wi-Fi requirements are different during different activities of the user.

4.4.1.2 System Usage Based Features

System Usage based features tells how extensively the phone is being utilized. Features will be higher if the phone is used for heavy applications like voice calling, network browsing vs. when the phone is used for viewing SMS, setting alarm etc.

The system usage feature of the smartphone is computed based on the following entity.

- The average CPU utilization of the smartphone.
- The average memory used by the applications in the smartphone.

The CPU utilization tells about state of running applications and the memory used tells about the state of installed applications and background services.

4.4.1.3 Interaction Based Features

Interaction based features tells how frequently user is using his phone. More frequent the user is on the device, more likely he will requiring the network connectivity.
Interaction based features consist of the following features.

- *Screen On Time*: In half an hour duration, total time in milliseconds for which screen was on.
- *Frequency Of Interactions*: In half an hour duration, total number of interaction (that is number of time the screen has been turned off and turned on) by the user.

The features are extracted in parallel so as to get the consistent data and reduce the battery consumption due to wake-lock acquisition by the application. Finally we concatenate all the features to form a final feature vector for the classification.

4.4.2 Training and Classification

The user's activity is modeled based on his behavior. The features extracted by the feature extraction module is fed to the classification module for prediction. We classified the user's activity level into four different classes, labeled by l_0, l_1, l_2 and l_3.

4.4.2.1 Training

For training purposes, we created an application that logs the accelerometer, screen on duration, internet, CPU and RAM usage of the device at every half an hour interval. We scaled the internet usage based on the screen on duration in half an hour as well as and the data traffic. The assumption is if the screen is off and the Wi-Fi is still on then without any data traffic this should be handled in intelligent way. We have categorized user into four categories no usage, low usage, medium usage and high usage. Class label of a feature vector created during current time window is decided based on the internet usage by the user in next time window. We run the application for 7 working days which resulted into 310 feature vector. For preserving the usability of the application, the classes are given different penalty scores. Table 4.1 shows the weights assigned to different classes while training. Weights are calculated in such a way so that(Number of features of a class $\times$ Weight of the corresponding class) should be constant across the classes.

Table 4.1: Class weights assigned to different activity level while training.

Class Label	Very Low (l_0)	Low (l_1)	Medium (l_2)	High (l_3)
Weight	1.0	2.71	8.92	4.92

4.4.2.2 Classification

For classification, the trained SVM model along with the feature extraction module is deployed on the smartphone. The features are collected for half an hour interval. After half an hour, the prediction (based on the features from feature extraction module) is done. The SVM model created during training is used for predicting the labels. The predicted label is then send to the Wi-Fi scheduling module and the Wi-Fi is scheduled appropriately.

4.5 Experimentation and Results

We evaluated our approach based on two metrics, the power saved and the user happiness. Power saved metric measures the overall power saved by our method. Although the power saving is important,

it should not hamper the user's overall experience on the smartphone. So user happiness testing is performed to test the user's satisfaction with the software.

For measuring the power saved by our software, we subjected the phones to real-world behaviors (via calls, messages and emails) under two different settings, when our proposed method is installed and when it is not. The percentage drop in the power is computed in both the settings to find the energy efficiency achieved by our method. For finding the user happiness, we installed the application on smartphones of two different users and recorded the number of times they switched the Wi-Fi state manually. Each manual switch is considered as a penalty on user happiness. Section 4.5.1 explains the implementation of the system. Section 4.5.2 shows the power saved by the application and Section 4.5.3 shows the user happiness scores obtained when the application was installed on user's smartphone.

4.5.1 Implementation

An android application is created for capturing the features, predicting the activity and scheduling the Wi-Fi. The application schedules a prediction service at intervals of half an hour. Along with the service, it also registers a broadcast receiver that listens to screen state change event and logs the changes. After every half an hour, the service wakes the android system and 3D accelerometer readings are recorded for 30 seconds. Based on the readings, the accelerometer signal's mean, standard deviation, correlation and Fourier transform is computed. Along with these features, the screen on time and frequency of interactions are also evaluated using the logs created by the broadcast receiver. Android OS logs the internet, CPU and memory usage of the system. The average internet, CPU and memory usage is calculated based on the logs created by the OS. To maintain consistency among different features, all the features are extracted at the same time. Features are concatenated into a single feature vector. SVM classifier is used to predict the usage in next half an hour (using a pre-trained model) utilizing these features. Based on the predicted output by the classifier (high, medium, low or very low), the Wi-Fi is turned off and an alarm manager is scheduled to turn it on after suitable interval of time. All the decisions made by the classifier are logged for the purpose of evaluation. For evaluating user happiness, all Wi-Fi state changes are also logged. The number of manual state change events are extracted and considered as a measure for calculating user happiness. The activity prediction is shown to be achievable using light-weight computations without consuming much power or memory. Our application took only 5MB of RAM and 101KB external memory, which is similar to the average requirements for an application on smartphones. Also, our application took 1% of battery in 24 hours[1].

4.5.2 Power saved by the software

User have different activities scheduled at different times. For example, number/durations of calls in one hour might be very different from the number of calls in the next hour. Similarly the power consumed in other tasks like messages, games etc. might be different in the two hours. So, calculating

[1]The power consumed by the application is recorded through battery info manager on android OS.

the battery drop in the two corresponding hours (once with the scheduler and once without the scheduler) might not be an accurate measure to evaluate the power saved by our software.

To evaluate the power saving by our software we took two device from both the categories (refer Table 4.4 for specification) having same set of application, battery state, memory usage etc., and performed below test plan on all four devices, two having our software installed (one from each category) and two without our software. Table 4.2 lists the activities carried out on the smartphone. All the activities are performed at almost same time and for same duration. For example, the calls on all the devices are performed simultaneously for 2 minutes duration. Similarly, all the messages and emails are also send simultaneously so that the loss incurred by other processes remains same. The controlled experiment was performed for 4 hours in similar settings and the battery dissipated on each device is recorded. Figure 4.6 shows the relative percentage drop in battery on the devices after each hour.

Table 4.2: Energy Efficiency Testing: Percentage battery dropped in Samsung Galaxy S (Dev.1) and HTC Explorer (Dev.2) during controlled tests.

	Messages Received	**Number of Calls**	**E-Mails Received and Replied**	**Drop in Battery (Dev.1)**	**Drop in Battery (Dev.2)**
Without Scheduler	25	15	25	38%	52%
With Scheduler	25	15	25	28%	15%

Table 4.3: User Happiness: Count of Number of times Wi-Fi was manually turned on.

	Wi-Fi turn-on count (Usr 1)		% Battery Drop (Usr 1)		Wi-Fi turn-on count (Usr 2)		% Battery Drop (Usr 2)	
	Working Hrs.	Non-Working Hrs.	Working Hrs.	Non-Working Hrs.	Working Hrs.	Non-Working Hrs.	Working Hrs.	Non-Working Hrs.
Without Scheduler	NA	NA	38	26	NA	NA	33	8
With Scheduler	5	2	29	19	3	0	13	7

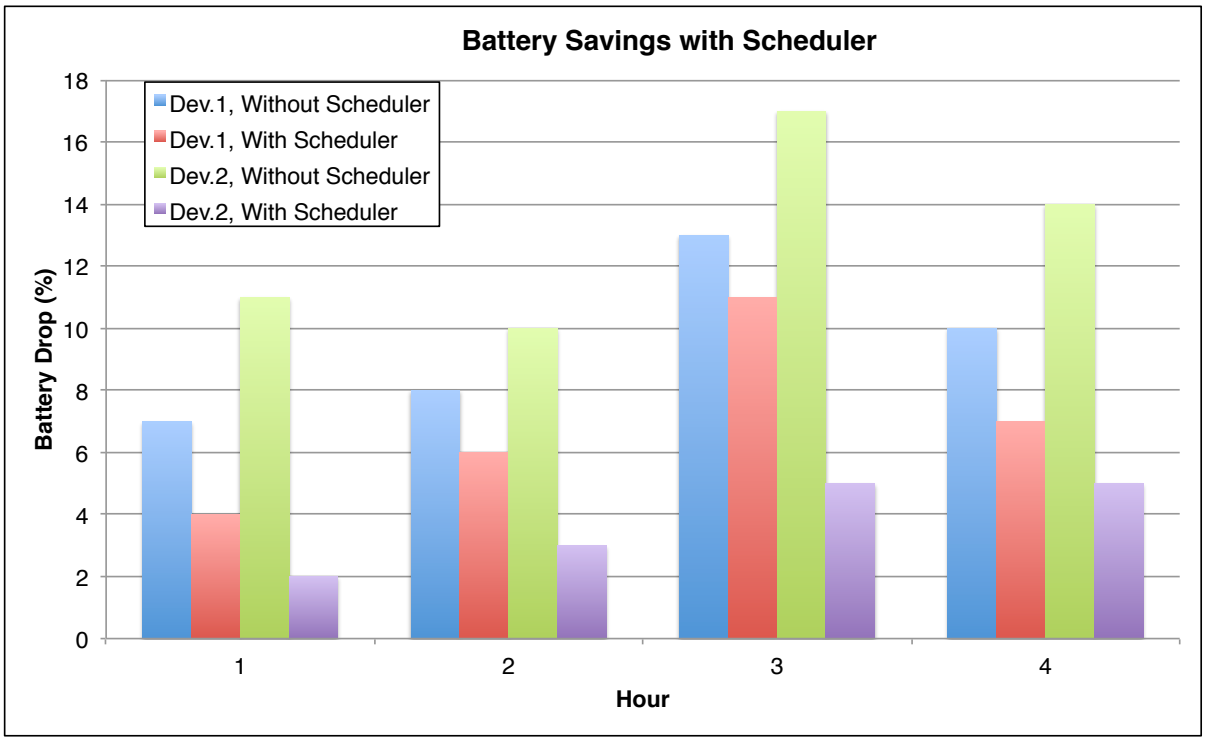


Figure 4.6: Percentage drop in battery with time under different settings (Dev.1 and Dev.2).

4.5.3 Measuring User Happiness

Power saving is important but at the same time we have to take care that user experience should not be hampered a lot. To verify this we conducted an experiment in which we have given our software to users and logged there battery usage in two categories first when the user is very active second when the user is less active. Along with battery usage, the number of times the Wi-Fi state is changed manually is also recorded. Each Wi-Fi switch by the user is considered as misclassification by the algorithm. The higher the number of switches by the user, the more unhappy the user is with the application. Table 4.3 shows 6 hrs analysis when the user is highly active (working hours) and shows the 6 hrs analysis when the user is not much active (non-working hours) on the smartphone device. Our application took just 1% of battery in 24 hours and saved 10 to 37% of battery in 4 hours without much affecting user convenience. The battery usage vs. amount of battery saved confirms that the application can be used non-stop and can save significant amount of power, especially for heavy internet users.

4.6 Experiment Extended

We further extend the experiments and benchmark the performance of different machine learning algorithms with fresh dataset using same software as Section 4.5 by porting it on a ASUS low cost smart phone Table 4.4. We did the comparitive study of the three algorithms *kNN, SVM and Neural*

Table 4.4: Specifications of the phones on which testing was conducted.

	Feature	Dev.1: Samsung	Dev.2: HTC	Dev.3: Asus
General	2G, 3G Freq	GSM, HSDPA, Mini	GSM, HSDPA, Mini	as1
Display	Type, Size	AMOLED capacitive touch, 4.0"	TFT capacitive touch, 3.2"	as1
Memory	Internal	512 MB RAM, 512 MB ROM	512 MB RAM, 512 MB ROM	as1
Data	[GPRS], [EDGE], [Speed], [WLAN], [Bluetooth], [USB]	[Class 12 (4+1/3+2/2+3/1+4 slots), 32 - 48 kbps], [Class 12], [HSDPA, 14.4 Mbps; HSUPA, 5.76 Mbps], [Wi-Fi 802.11 b/g/n, DLNA, Wi-Fi hotspot], [Yes, v3.0 with A2DP], [Yes, microUSB v2.0]	[Up to 80 kbps], [Up to 236.8 kbps], [HSDPA, 14.4 Mbps; HSUPA, 5.76 Mbps], [Wi-Fi 802.11 b/g/n, DLNA, Wi-Fi hotspot], [Yes, v3.0 with A2DP,EDR], [Yes, microUSB v2.0]	as1
Features	[OS], [Chipset], [CPU], [GPU], [Sensors], [Messaging]	[Android OS, v2.3 (Gingerbread)], [Qualcomm MSM8255T Snapdragon], [1.4 GHz Scorpion], [Adreno 205], [Accelerometer, proximity, compass],[SMS(threaded view), MMS, Email, Push Mail, IM, RSS]	[Android OS, v2.3 (Gingerbread)], [Qualcomm MSM8255T Snapdragon], [600 MHz Cortex A5], [Adreno 200], [Accelerometer, proximity],[SMS(threaded view), MMS, Email, Push Mail, IM, RSS]	as1

Networks by replicating the same set of experiments like power saved and user happiness as perfomed in previous Section 4.5.

4.6.1 Dataset Collection and Hyperparameter Tuning

We followed exactly the same process for the dataset generation as done in previous Section 4.5. After every *30 mins*, the software service wakes up the android system and log all the necessary information like 3D accelerometer readings, screen on time, frequency of interaction, CPU and memory usage of the system are also logged using broadcast receiver and Android OS logs. By following these steps we run the application for 20 working days which resulted into 956 feature vectors. Parameter tuning is carried by spliting the dataset in train and validation in 80-20 ratio.

4.6.1.1 *k*NN Paramaeter Tuning

In a *k*NN algorithm number of neighbors(*k*) is a hyper parameter that need to be tune for model building. No optimal number of neighbors suits all kind of data sets, individual dataset has it's own requirements. The small value of *k* leads to noise will have a higher influence on the result where as large value of *k* make it computationally expensive. Figure 4.7 mean error rate for different values of *k* ranging from 1 to 30. For our model selection we have chosen ***k = 7*** as it seems to be good balance between the noise and the cost of computation.

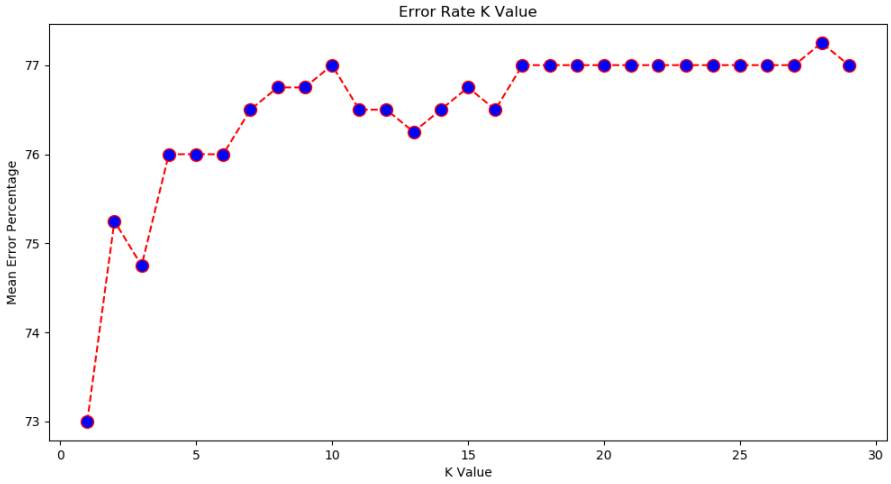


Figure 4.7: Tuning number of neigbours(*k*). We vary *k* from 1 to 30 and report the mean error on the validation set. We can see after *k* = 16 the error rate is almost constant which indiactes absence of noise in the model.

4.6.1.2 SVM Parameter Tuning

C and Gamma are the parameters for a nonlinear support vector machine(SVM) with a Gaussian radial basis function kernel. We ran grid search algorithm to find optimal C and Gamma pair for our SVM model(Figure 4.8). We build our model *C=8* and *Gamma=0.3125* and achieved 84.7% accuracy with 10-fold cross validation process.

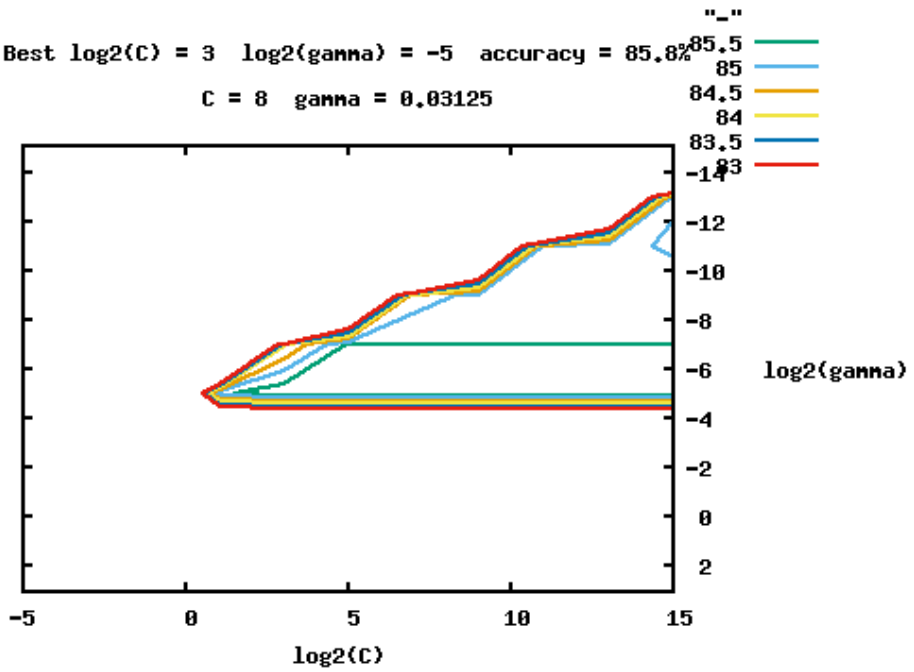


Figure 4.8: Grid search on the dataset to find the optimal C and Gamma pair for the SVM model.

4.6.1.3 Neural Networks Paramaeter Tuning

While training a neural network we have to make many decisions regarding the following hyperparameter those are (i) *Solver* (ii) *Number of hidden layers* and *Number of hidden units for each hidden layer* (iii) *Learning rate* . We have used scit-learn framework [56] to tune these hyperparameters.

Solver: The default solver *adam* for the scikit-learn works good with relatively large dataset with thousands of training samples. For small dataset *lbfgs* converge faster and perform better [56]. In all our experiments we have used *lbfgs* solver.

Number of Hidden Layers and Hidden Units: We have two decesions to made (i) *How many hidden layers* (ii) *How many units in each hideen layer*. We will first fix the number of hidden layer to use in neural network. From *Introduction to Neural Networks for Java, second edition* by *Jeff Heaton* summarizes the capabilities of neural network architectures([57]) with various hidden layers like Table 4.5. In our experiments we have considered upto to 2 hidden layers for our model. Deciding number of hidden layers is only a small part we shold also fix the the number of neurons in each hidden layer. Using too few neurons in the hidden layers will lead underfitt where as too many neurons can overfitt the model. In order to find the right number of neurons in hidden layer cross-validation accuracy mechanism were used.

Learning rate:

No Hidden Layer	Result
0	Only capable of representing linear separable functions or decisions.
1	Can approximate any function that contains a continuous mapping from one finite space to another.
2	Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.
>2	Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers.

Table 4.5: Hidden layer units capability to capture data distribution.

4.7 Summary

Power consumption and the consequent low battery life is one of the primary limiting factors for modern mobile devices. In this work, we proposed an effective way to nullify some of this problem by predicting user activity levels and scheduling the power hungry wireless modules accordingly. The encouraging results of the experiments lead us to affirm that a step forward has been taken to personalize the equipment so that it can function well while improving power efficiency.

The effectiveness of the proposed approach can be improved and extended in many ways. Application time interval can be scheduled dynamically, making it adaptive to user's activity level, which should give better result in terms of user happiness as well as battery saving. As the application is monitoring and modeling the user's activity, it can also perform additional tasks such as silencing the phone between meeting/sleep hours, or put the phone in silent mode when the user is driving.

Chapter 5

Conclusions and future work

In this thesis we have presented how the different sensor data can be fused in a complementary fashion to overcome the each other drawbacks. Data fusion techniques have been extensively employed on multi-sensor environments with the aim of fusing and aggregating data from different sensor. In a given context data fusion leads to less expensive, higher quality and more relevant information. We consider a computer vision and machine learning based task to show case the sensor fusion methodologies. We have addressed localization problem by fusing image and Gps sensors and also demonstrated how machine learning and multi-sensor data fusion techniques are critical to the system success.

Accurate Localization by Fusing Images and GPS Signals: Localization by fusing the image and Gps sensor has out perform the localization process by using image or Gps sensor alone. We improved the visual localization using the Gps and overcome the problems like occlusion and perceptual aliasing. Then we used image retrieval to reduce the noise in Gps signal.

Problem of visual localization was framed as image retrieval problem and by fusing Gps with vision we filtered out the useful set of features for each training image. Noise in commercial Gps receivers is another problem for short distance visual localization. Noise in Gps impact the labeling of the images. To overcome this problem we demonstrated a vision based solution which gives use more robust and consistent Gps signal as end result. Finally we integrated both of them and performed the experiments to show the various aspects of method and improvement in localization.

Future Work: Localization is one of the very critical problem in computer vision. One can build numerous solution over it. In future work we will be focusing on a computer vision based solution for heritage sites *i.e* "Storytelling for Heritage Sites on Smart Phones". We want to replicate virtual tourist guide functionality in a mobile phone application which adapts the story according to every user movement and location on the heritage site. Various other sensors like accelerometer, gyroscope scope can be used to predict the user current location with high precision. Integrating social network data is other interesting extension that could add to the story telling application. Social footprint of people could be stored in form of friendship graphs; choices of every friend visiting the same site in past, the time he

spent at each site etc. could assist application to suggest near by sites to current user.

Modeling User Activity for Conserving Power on Smartphones: We have presented a novel approach to schedule Wi-Fi(can be extend to 3G or any other service on smartphones). Using Wi-Fi as an example, we show that intelligent scheduling based on a users activity level leads to lower power consumption without adversely affecting the user experience. Data from various sensors is used to model and predict a users activity, which is then used to schedule the wireless services. Running the machine learning algorithms on the embedded devices is a challenging job. Implementation should be very efficient in terms of CPU power consumption and speed to provide an smooth user experience.

Future Work: One can expand this work to come up with an intelligent and automatic profile switcher. With the help of this intelligent profiler smartphone will be able to handle the situations like:

1. Putting the mobile in silent mode as in your enter in a meeting.
2. Respond to calls when your are driving.
3. Automatic switching between Wi-Fi and data network.

Above we have stated few use cases of our work but the possibilities are endless. Using machine learning we can give ability to machine perceive the world as humans.

Related Publications

1. Kumar Vishal, C. V. Jawahar and Visesh Chari: Accurate Localization by Fusing Images and GPS Signals, IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015

2. Kumar Vishal, Romil Bansal, Anoop M. Namboodiri and C. V. Jawahar: Providing Services on Demand By User Action Modeling on Smart Phones, ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014.

Bibliography

[1] Mobile phone internet user penetration worldwide from 2014 to 2019. *http://www.statista.com/statistics/284202/mobile-phone-internet-user-penetration-worldwide/*.

[2] Jeffrey Sharkey. Coding for life battery life, that is, 2009. http://dl.google.com/io/2009/pres/W%5F0300%5FCodingforLife-BatteryLifeThatIs.pdf.

[3] Kumar Rangarajana. Android battery consumption - the what and how, 2012. http://www.slideshare.net/littleeye/power-optimization-for-android-developers.

[4] Philippe Bonnifait, Pascal Bouron, Paul Crubillé, and Dominique Meizel. Data fusion of four abs sensors and gps for an enhanced localization of car-like vehicles. In *Robotics and Automation, 2001*. IEEE.

[5] Ujwal Koneru, Sangram Redkar, and Anshuman Razdan. Fuzzy logic based sensor fusion for accurate tracking. In *International Symposium on Visual Computing, 2011*. Springer.

[6] James Hays and Alexei A Efros. Im2gps: Estimating geographic information from a single image. In *CVPR*. IEEE, 2008.

[7] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *USENIX annual technical conference*. Boston, MA.

[8] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on pattern analysis and machine intelligence*.

[9] Venkat N Gudivada and Vijay V Raghavan. Content based image retrieval systems.

[10] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 1933.

[11] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[12] Activation function. https://en.wikipedia.org/wiki/Activation_function/.

[13] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, 2008.

[14] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, 2005.

[15] Random-walks-graphs. https://people.math.osu.edu/husen.1/teaching/571/random_walks.pdf/.

[16] Cross-validation. https://en.wikipedia.org/wiki/Cross-validation_(statistics).

[17] Anna Bosch, Xavier Muñoz, and Robert Martí. Which is the best way to organize/classify images by content? *Image and vision computing*.

[18] Chih-Fong Tsai. Bag-of-words representation in image annotation: A review. *ISRN Artificial Intelligence*.

[19] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local features for object class recognition. In *Computer Vision. ICCV 2005. Tenth IEEE International Conference on*. IEEE.

[20] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*.

[22] Pedro Quelhas, Florent Monay, Jean-Marc Odobez, Daniel Gatica-Perez, and Tinne Tuytelaars. A thousand words in a scene. *Pattern Analysis and Machine Intelligence, IEEE*.

[23] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

[24] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*.

[25] Yu-Gang Jiang, Jun Yang, Chong-Wah Ngo, and Alexander G Hauptmann. Representations of keypoint-based semantic concept detection: A comprehensive study. *Multimedia, IEEE*.

[26] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference*.

[27] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*.

[28] Daniel Maier and Alexander Kleiner. Improved gps sensor model for mobile robots in urban terrain. In *ICRA*. IEEE, 2010.

[29] Tsung-Yi Lin, Serge Belongie, and James Hays. Cross-view image geolocalization. In *CVPR*. IEEE, 2013.

[30] AH Hafez, Manpreet Singh, K Madhava Krishna, and CV Jawahar. Visual localization in highly crowded urban environments. In *IROS*. IEEE, 2013.

[31] Eladio Martin, Oriol Vinyals, Gerald Friedland, and Ruzena Bajcsy. Precise indoor localization using smart phones. In *Proceedings of the international conference on Multimedia*. ACM, 2010.

[32] Lijun Wei, Cindy Cappelle, Yassine Ruichek, and Frédérick Zann. Intelligent vehicle localization in urban environments using ekf-based visual odometry and gps fusion. In *World Congress*, 2011.

[33] Mark Cummins and Paul Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 2008.

[34] Amir Roshan Zamir, Shervin Ardeshir, and Mubarak Shah. Gps-tag refinement using random walks with an adaptive damping factor. In *CVPR*, 2014.

[35] Michael J Milford and Gordon Fraser Wyeth. Seqslam: Visual route-based navigation for sunny summer days and dtormy winter nights. In *ICRA*. IEEE, 2012.

[36] Mark Cummins and Paul Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 2011.

[37] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*. Citeseer, 2007.

[38] Panu Turcot and David G Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshops*. IEEE, 2009.

[39] Jan Knopp, Josef Sivic, and Tomas Pajdla. Avoiding confusing features in place recognition. In *ECCV*. Springer, 2010.

[40] Changchang Wu, Sameer Agarwal, Brian Curless, and Steven M Seitz. Multicore bundle adjustment. In *CVPR*. IEEE, 2011.

[41] Yushi Jing and Shumeet Baluja. Visualrank: Applying pagerank to large-scale image search. *Pattern Analysis and Machine Intelligence*, 2008.

[42] Contour+2 action camera. `http://contour.com/cameras/contour2/`.

[43] Lin Sun, Daqing Zhang, Bin Li, Bin Guo, and Shijian Li. Activity recognition on an accelerometer embedded mobile phone with varying positions and orientations. In *Proceedings of the 7th International Conference on Ubiquitous Intelligence and Computing*.

[44] L. Bao and S. S. Intille. Activity Recognition from User-Annotated Acceleration Data.

[45] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*.

[46] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*.

[47] Uwe Maurer, Asim Smailagic, Daniel P. Siewiorek, and Michael Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*.

[48] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *Proceedings of the 4th International Conference on Pervasive Computing*.

[49] Jiahui Wu, Gang Pan, Daqing Zhang, Guande Qi, and Shijian Li. Gesture recognition with a 3-d accelerometer. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*.

[50] Angshu Rai, Zhixian Yan, Dipanjan Chakraborty, Tri Kurniawan Wijaya, and Karl Aberer. Mining complex activities in the wild via a single smartphone accelerometer. In *Proceedings of the Sixth International Workshop on Knowledge Discovery from Sensor Data*.

[51] Gary Weiss and Jeffrey Lockhart. The impact of personalization on smartphone-based activity recognition. In *Activity Context Representation: Techniques and Languages Workshop at Association for the Advancement of Artificial Intelligence*.

[52] Brent Longstaff, Sasank Reddy, and Deborah Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *Proceedings of Pervasive Health*.

[53] A. Sixsmith and N. Johnson. A Smart Sensor to Detect the Falls of the Elderly. *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*.

[54] Zhixian Yan, Vigneshwaran Subbaraju, Dipanjan Chakraborty, Archan Misra, and Karl Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Proceedings of 16th Annual International Symposium on Wearable Computers (ISWC)*.

[55] Zhenyun Zhuang, Kyu-Han Kim, and Jatinder Pal Singh. Improving energy efficiency of location sensing on smartphones. In *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*.

[56] Scit-learn. *https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html*.

[57] Introduction to neural networks with java. *https://www.heatonresearch.com/2017/06/01/hidden-layers.html*.