A SOFTWARE FRAMEWORK FOR IMAGE RETRIEVAL AND VISUAL UNDERSTANDING IN DYNAMIC AND SENSOR RICH ENVIRONMENTS

THESIS

Noah C. Lesch, Captain, USAF

AFIT-ENG-MS-17-M-045

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-17-M-045

A SOFTWARE FRAMEWORK FOR IMAGE RETRIEVAL AND VISUAL UNDERSTANDING IN DYNAMIC AND SENSOR RICH ENVIRONMENTS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Noah C. Lesch, BS

Captain, USAF

March 2017

AFIT-ENG-MS-17-M-045

A SOFTWARE FRAMEWORK FOR IMAGE RETRIEVAL AND VISUAL UNDERSTANDING IN DYNAMIC AND SENSOR RICH ENVIRONMENTS

Noah C. Lesch, BS

Captain, USAF

Committee Membership:

Lt Col John M. Pecarina, PhD
Chair

Douglas D. Hodson, PhD
Member

Kenneth M. Hopkinson, PhD
Member

AFIT-ENG-MS-17-M-045

Abstract

Performing search and retrieval operations with massive amounts of visual and environmental sensor information is problematic in time sensitive and mission critical situations, such as emergency management and disaster response. Distinct sensor readings can be fused to create a compact multimodal representation of a location. Efficient search and retrieval would allow a system to scale to process larger amounts of information. Content Based Image Retrieval (CBIR) systems inherently rely on the search and retrieve operations to support timely and accurate responses. However, there is currently no adequate software framework for multimodal CBIR to support situational awareness in dynamic and sensor rich environments. In this thesis, an extensible framework for CBIR is proposed to support an understanding of a sensor rich environment through the automated search and retrieval of relevant images and the context of their capture. This constitutes assisted CBIR as embodied in the proposed framework for multi-sensor assisted CBIR system (MSACS). The framework of MSACS is proposed with a software design document and partially prototyped to implement the core CBIR system using the state of the art Bag of Visual Words paradigm. The system is evaluated using functional analysis against existing CBIR systems, prototype system, and a notional localization application. The extensibility of the framework allows a standalone application to interface with it. A localization application is created as a proof-of-concept to provide a result in latitude and longitude in relation to known sensors. Assisted CBIR could lead to increased confidence in autonomous systems and support vision-based understanding of an environment to execute tractable search and retrieve operations in a timely manner.

Acknowledgments

I'd like to thank my academic advisor, Lt Col John Pecarina, for showing me how to approach research in a common sense manner and guide me through the inevitable research hurdles.

Noah C. Lesch

Table of Contents

Page

List of Figures

Page

List of Tables

Page

A SOFTWARE FRAMEWORK FOR IMAGE RETRIEVAL AND VISUAL UNDERSTANDING IN DYNAMIC AND SENSOR RICH ENVIRONMENTS

I. Introduction

To operate within the modern battlespace, the need for good situational awareness is desired, but massive amounts of visual and sensor data threaten to overload manual and automated processes to use them effectively. To answer to this problem, warfighters need better tools for finding relevant data, especially for imagery, where information processing demands are particularly high. If modern methods for image retrieval are adapted to the modern battlespace, timely and relevant queries may provide quality situational awareness to decision makers.

Image retrieval is an open problem in the realm of computer vision and deals with techniques, methods, and implementations of timely and precise search results. One method of image retrieval utilizes a content based image retrieval (CBIR) system. A CBIR system returns a set of ranked digital images to a user, based on a query. The query can be combinations of an image and any other form of data which are accepted by the CBIR system. The system can be used to parse, query, filter and catalog large amounts of images or video to feed computer vision applications, bridging the gap between human and computer understanding of dynamic environments.

A high level depiction of CBIR is shown in Figure 1. A robust CBIR system would enable vision-based understanding of an environment to ease the burdens of information overload and increase human confidence in autonomous systems. A robust CBIR system should perform search

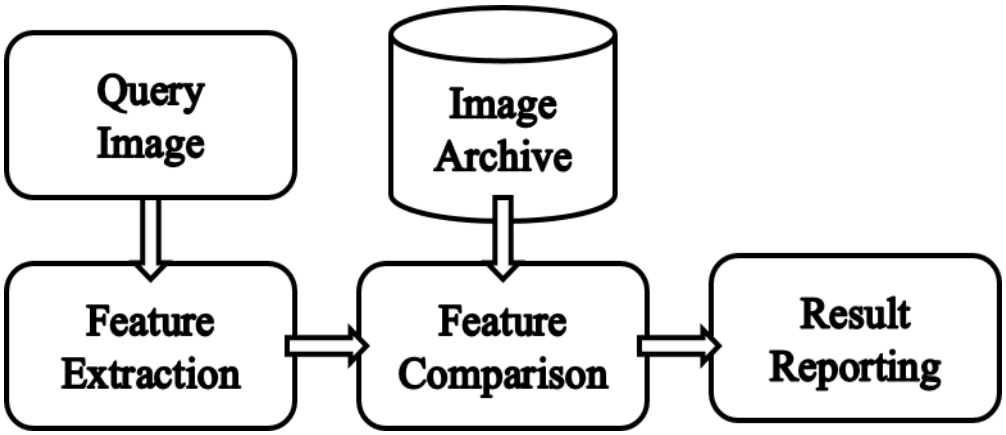

Figure 1. High Level CBIR System Flow

and retrieval operations in a tractable amount to time to maintain usefulness. Feature extraction and comparison can be automated while results reporting would require a user for final decisions.

1.1. Motivation

The Department of Defense (DoD) makes tactical decisions from images and video data obtained from intelligence, surveillance, and reconnaissance (ISR) sources. This data is so ubiquitous that it is virtually assumed when utilizing airborne sources. There have been efforts to fuse video and image data with metadata from other onboard sensor modules to generate a clearer picture of a battlespace [1]. This fused data can then be archived for later search and retrieval operations.

Searching through video archives with thousands of hours of video data is computationally cumbersome and not feasible for a human operator without prescient knowledge of the specific date, time, and correlated metadata associated with a specified search. The National Geospatial-Intelligence Agency (NGA) has turned towards open-source solutions to make the problem of video and image retrieval more computationally efficient [2]. The Air Force has stated they believe

that small UAS will be the cornerstone of Air Force ISR in the next 20 years. Specifically, swarms of UAS will work together to accomplish military objectives while providing no single point of failure due to their collaborative, distributed configuration [3]. In order for these UAS swarms to adapt and reconfigure they need to react and orient to multimodal sensor data derived from the battlespace. Thus a command and control cell requires the ability to direct UAS swarms with only a small number of people while interpreting and responding to a potential plethora of sensor data [4].

In the commercial industry, CBIR is used in many search engines such as Google, the Chinese search engine Baidu, and TinEye. These websites make use of CBIR by providing a reverse image lookup capability. This allows a user to drop an image into a search bar and see return images that are visually identical or similar to the one provided by the user. Google's image search extracts several features from the image such as shapes, lines, proportions, and colors to create mathematical models. These features then match against images already in Google's image index [5]. If the image contains metadata this additional information will be used to help narrow down a description. These algorithms from Google are proprietary and not made available for public use.

CBIR has been hypothesized for situational awareness scenarios to aid, for example, emergency services or forensic inquires. In [6] and [7] the authors suggest moving away from a text-based image retrieval system for hospitals and police stations because the text annotations, or metadata, are subjective to every person's interpretation. Instead they propose using CBIR to quantize color and texture data to link together mugshots with visual evidence, or by proposing a patient's diagnosis based on past diagnoses using x-rays, scans, etc. Similarly to [6] and [7], in [8] the author proposes a CBIR system for matching visual evidence with past police cases to

determine patterns and possible matches. This system also uses relevance feedback which takes in user inputs to assist in narrowing down CBIR results. In [9] the authors use color and texture features to create a CBIR system to detect wildfires within images. These results are annotated with social media tags from Flickr and were used for information dissemination during the Colorado wildfires of 2012. The results showed that CBIR along with social media serve as an alternative source of information dissemination and possible emergency services mobilization. The authors of [10] describe using CBIR along with machine learning techniques to inspect and classify solar image data and related solar events.

The ability to search large digital archives and provide timely, accurate, and relevant results to a user drives this research. A CBIR system can identify similarity between images to provide relevant results from an image corpus to a user query. If the query image is a landmark, the CBIR system can return other images of the landmark to allow the user to gain understanding of its location in the environment. If the query image is a potential target, the CBIR system can return results that help to match and identify the target. The focused development of CBIR for these two use cases allows optimization from the generic CBIR system approach. In addition, other sources of information, (i.e. magnetic, Wi-Fi, etc.) can help to improve the accuracy of the CBIR system.

1.2. Problem Statement

The opportunity to exploit images from a dynamic, sensor rich environment calls for assisted CBIR, which integrates search and retrieval mechanisms for image content with other environmentally derived context data. There is currently no software framework for assisted CBIR. Thus a framework is proposed for assisted CBIR that is extensible, modular, and scalable. This framework will solve the problem of multimodal CBIR by using standardized interfaces to accept

input from many sensor types, maintain data persistence, and respond to user queries. The proposed framework will use these multimodal inputs to filter results obtained from the search and retrieval operations.

1.3. Approach

A research approach to assisted CBIR must answer fundamental questions about the design and intended operation of the target system. Therefore, the research questions this thesis seeks to answer are as follows:

1. What are the requirements to create an extensible software framework which implements assisted CBIR?
2. How could an assisted CBIR system drive a wide array of applications and data needs?
3. Can a CBIR system support specific applications such as localization?

The research approach lays the groundwork for future implementations of prototypes and experimental systems. A proof-of-concept prototype was created and tested to determine viability which is detailed in Section V. This prototype used traditional image-only CBIR to perform search and retrieval operations. The results of the CBIR system were used to drive an application focused on situational awareness. The situational awareness prototype performs a naïve localization function to determine positional information in latitude and longitude.

1.3.1. Software Design Document

A software design document (SDD) for the extensible assisted CBIR framework is presented. The SDD will be created using IEEE standard 1016 as a reference [11]. The proposed

SDD contains more information than a traditional SDD by also detailing requirements in addition to the typical material from IEEE 1016. Thus the SDD will lay the foundation for future research and programming activities in this topic area and will detail the envisioned composition of the framework through diagrams, descriptions, and requirements. The multi-sensor assisted content-based image retrieval system for situational awareness system is named MSACS.

1.3.2. CBIR Programming

Within the proposed framework will be a CBIR system. The CBIR system will use the Bag of Visual Words (BoVW) method for image search and retrieval [12]. This CBIR system is presented in Section V and an experiment is conducted as a proof-of-concept. The proposed CBIR system presented in the SDD will extend the traditional BoVW search and retrieval operation by adding multimodal sensor inputs into the search operation. The use of the multimodal sensor in conjunction with image data for image search and retrieval is henceforth called assisted-CBIR.

1.3.3. Localization Prototype

A situational awareness prototype will be a localization application which relies on the results of the CBIR system. This application will respond to a user's query to the CBIR system with a best guess for latitude and longitude using the query information to match with known information in a database. The results of the multimodal CBIR operation could then be used to support some yet-to-be developed application supporting mission objectives.

1.4. Thesis Overview

Figure 2 depicts the coverage of this thesis. This thesis investigates the prototype system developed as a proof-of-concept exercise and also creates an SDD with the intent of driving future development iterations. The summation of the prototype and the SDD guide the way for a modular, extensible software framework.

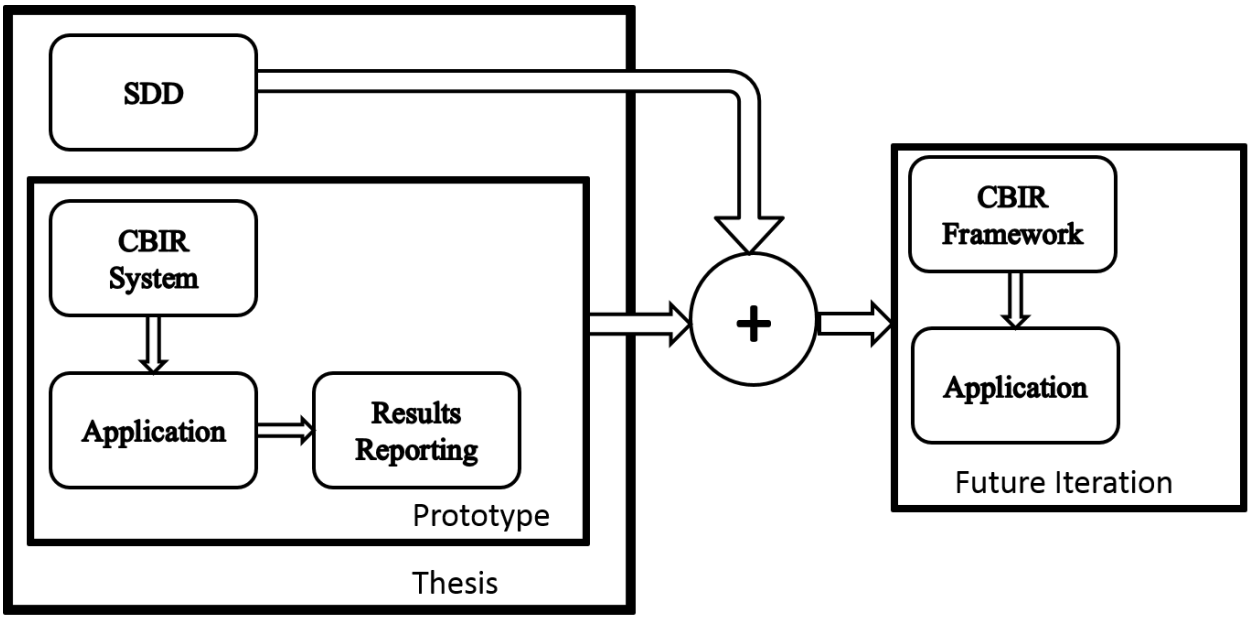

Figure 2. High Level Thesis Approach

1.4.1. Expected Contributions

This thesis claims three contributions extended from research questions listed in Section 1.3. These contributions are evaluated with a prototype detailed in Section V. The research activities seek to create a software design document, a prototype based on the software design document, an evaluation of the prototype's performance, and applicability of an application using the prototype.

By answering these questions this thesis seeks to makes the following contributions:

1. A software design document for an extensible software framework which implements assisted CBIR. A prototype that implements a subset of the design ideas is created to prove the usefulness and accuracy of the software design document.
2. A prototype assisted CBIR approach to image search and retrieval. This prototype is evaluated by using prevalent performance metrics related to CBIR operations from similar systems.
3. A localization application driven by the assisted CBIR results. This application is evaluated by investigating practical uses for a localization application that relies on Wi-Fi access points.

These contributions help realize a system which implements a CBIR system that is sensor agnostic and extensible for future data needs.

1.4.2. Structure

This thesis is organized as follows: section II discusses the preliminaries and related works for image retrieval and image comparison research areas, section III discusses the high-level requirements, design, and prototyping by building upon the preliminary information, section IV details the SDD and presents requirements and diagrams of the intended system implementation, section V discusses the prototype and localization application by implementing portions of the SDD, and section VI concludes the thesis and describes future work.

II. Preliminaries and Related Works

Humans search their surroundings for an object of interest with their eyes and typically have prescient knowledge of the object they are searching for. Once the object has been located a human will know, usually instantly, that the object is the one they were searching for. Similarly a computer requires prescient knowledge of the object it needs to search for in a scene while also understanding the scene itself. In order to understand a scene a computer needs a method to discriminate between objects, boundaries, and perspectives. These methods of discrimination are computer vision algorithms which quantify the visual content such as objects, within an image. This quantification results in a mathematical representation of an object from a scene. These mathematical representations are able to be implemented in tandem with a CBIR system to allow a user to query and obtain a response based on the content of the image.

CBIR has been around for decades and is still an open problem under the topic of Computer Vision. CBIR searches a digital library or database of images using the visual content extracted from a query image to match with the previously stored content of the archived images. This visual content can be color, texture, shape, or any salient data extracted from a query image. To extract content from an image requires the use of open-source or proprietary algorithms. Parsing an image with these extraction algorithms will quantize the visual content, or features, within the image. Once these features are quantized they can be stored in a database for future search and compare operations.

2.1. Image Comparison Methods

Image comparison is the fundamental backbone of CBIR. Some description of an image is made, often a histogram. A similarity measure is used to distinguish between the closeness of images. This section briefly describes methods of relevance in this thesis.

2.1.1. Histogram Construction

A naïve method of CBIR would compare color or greyscale histograms of all image pixels and corresponding colors to determine the similar measure of the histograms. However histograms are rarely used in CBIR because images are susceptible to noise in the form of light. The changes of illumination on an object as a result of the time of day, ambient lighting, or weather changes will change the pixel colors and thus the histogram will be different. Histograms are also redundantly detailed, as it is sufficient to only capture prominent image features of a histogram and suppress insignificant details [13].

Early methods of image comparison relied on color histograms. Color histograms can be represented in the red, green, and blue (RGB) color space. The color histogram is a compact summary of an image by representing the number of pixels of color in the image as a vector for each RGB color channel. For a given image a user can query a database to determine the closest match. The histograms are compared using a similarity measure to quantify their similarity. Color histograms tend to be coarse representations of images. While color histograms are generally insensitive to small changes in camera viewpoint they are susceptible false positives during search and retrieve operations. The false positives occur when two images with dissimilar content produce similar histograms due to the color distributions in the images [14].

Grayscale histograms are an even more compact representations of an image when compared with RGB histograms. Grayscale histograms encode the shade, or intensity, of a pixel instead of the individual color contributions from the RGB color channels. Important features within the image such as edges, regions, etc. are not lost by using a grayscale histogram [15-16].

Image feature detection algorithms seek the edges, regions, etc. in an image to produce a quantized feature descriptor. Another method of image comparison creates a histogram of these features within an image. The SIFT method of feature detection is presented in 2.1.4. Once a feature is detected in an image it needs to be described. Feature descriptors are quantized representations of the detected image features. Thus histograms of the extracted features are created. Images are then queried for these features and provide a more fine-grained method of search and retrieval when compared to RGB and grayscale histograms [17].

2.1.2. Similarity Measures

A similarity measure is the result of comparing data, in this case a query, against some other already existing data in order to determine a score of their similar content. This similarity measure can be applied over an entire corpus and then used to return a result set from the original query. Assessing the similarity between data can be calculated in several methods. One method is the Euclidean distance between a query image's feature vector and a stored feature vector in a database or an exhaustive feature vector computation of an entire dataset [18]. The Euclidean distance is the magnitude of the differences of the distances between each element in the feature vectors.

The cosine distance is another similarity measure. For example the cosine distance could be calculated between a query image and each image in a dataset or stored feature information in a database. The cosine distance is calculated by:

$$Similarity = \cos\theta = \frac{\vec{Q}\cdot\vec{D}}{|\vec{Q}|\cdot|\vec{D}|} \quad (1)$$

The similarity returned from the operation in equation 1 is the measure of the similarity [19] between two vectors. In this case $\vec{Q}$ is the query image vector and $\vec{D}$ is the database image vector. The value of the cosine distance ranges from 0.0 to 1.0. A cosine similarity value of 1.0 means that the two vectors are identical.

2.1.3. BoVW

BoVW is the visual approach to the Bag of Words (BoW) histogram associated with text documents and is illustrated in Figure 3. In BoW a histogram of word frequencies in a document is computed. In BoVW a histogram of the visual image patches, or words, is computed for each image patch in vocabulary.

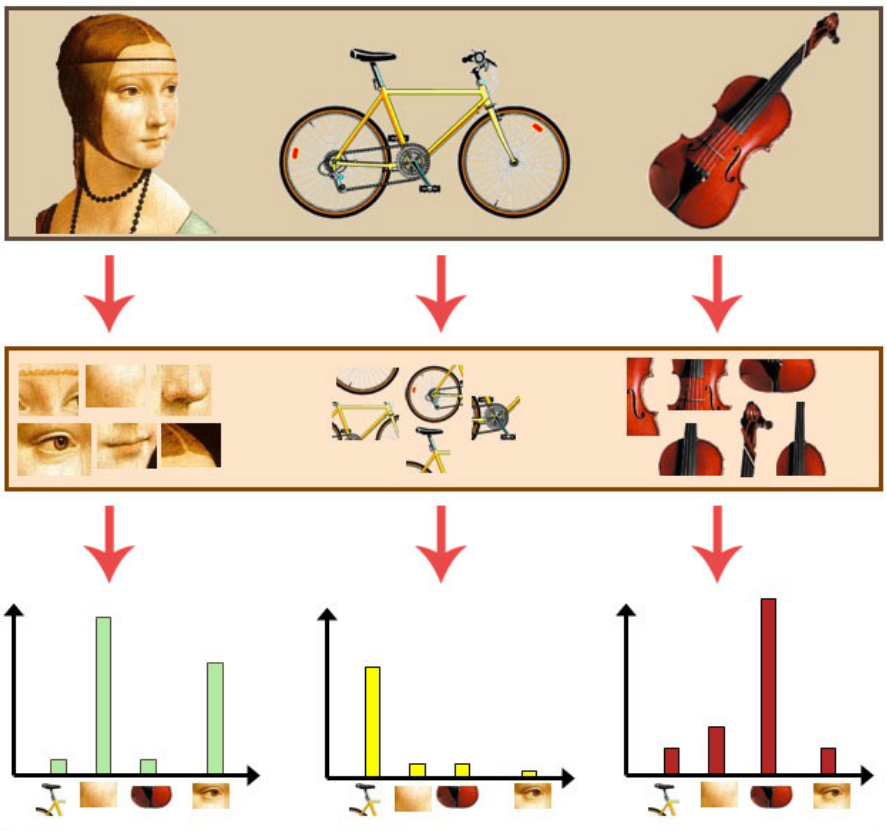

Figure 3. BoVW Example [20]

This BoW histogram can then be used for information retrieval problems. An inverted index can be created from the BoW histogram to give greater search weight to terms that occur less frequently in a document. This will improve a document's chance of retrieval from within a database.

In BoVW a similar approach is taken. Building on the feature extraction algorithm used in CBIR a codebook is generated with the quantized image features, which become the words. The codebook is generated with a clustering algorithm, typically k-means, in order to determine the k-unique cluster centers. To calculate the k cluster centers the quantized features are first graphed. Then an iterative process begins which calculates the cluster centers by producing approximately

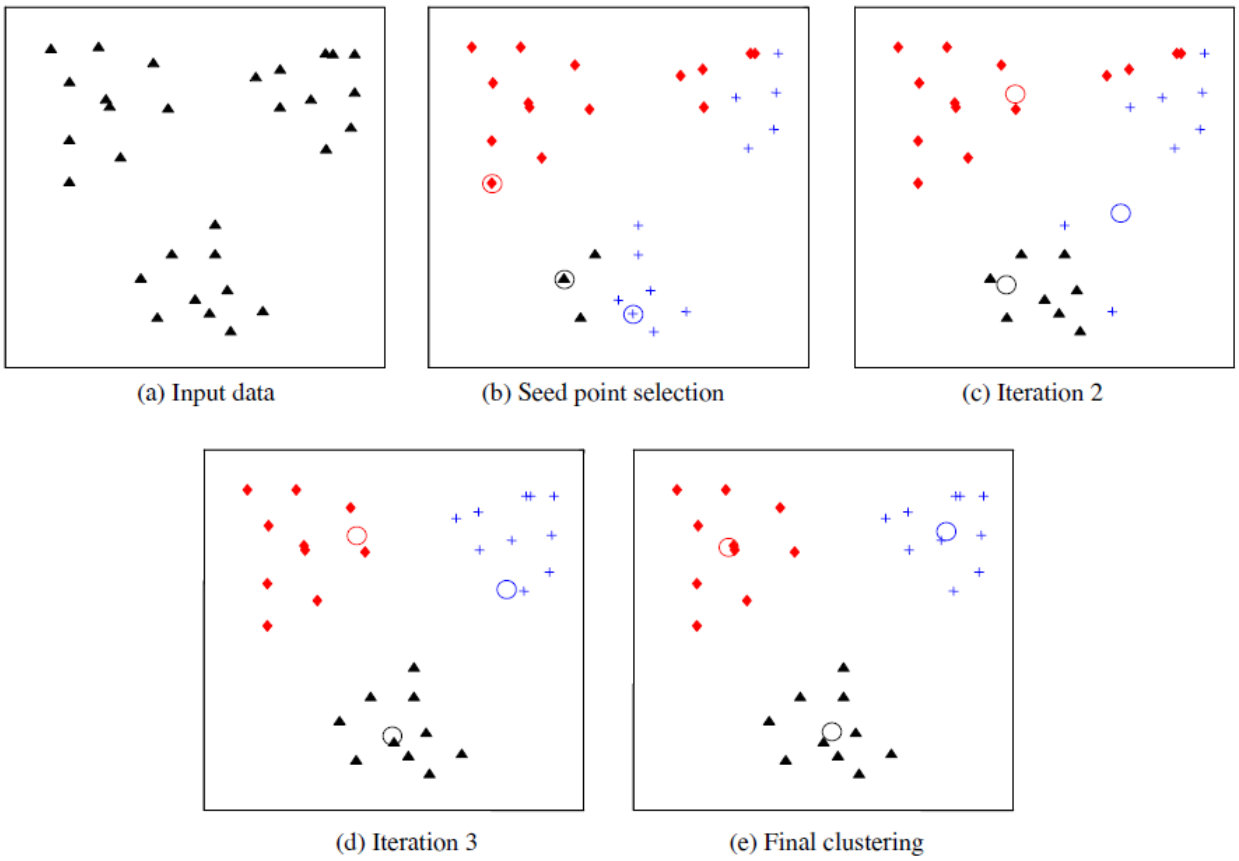

Figure 4. K-means Clustering Visualization [21]

k equal sized clusters. These k cluster centers become the codebook which can be used against future query images. Figure 4 depicts the k-means clustering algorithm from start to finish

Each cluster center in the codebook represents one word from the BoVW model. When a query image is parsed and the visual features are quantized a BoVW histogram based on the codebook can be computed for the image. This BoVW histogram can now be compared to other BoVW histograms from a dataset or corpus to determine a similarity distance metric. This method is faster than parsing an entire corpus every time a query is submitted. The BoVW codebook can be recalculated if there is a need to recalculate based on new images in a corpus or creating an entirely new corpus. Words from a BoVW codebook can also be graphed to discern data patterns between words. This is graphing and subsequent grouping is referred to as clustering.

The goal of data clustering is to discover natural grouping(s) of a set of patterns, points, or objects [1]. Clustering algorithms will discover k-groupings of n data points by using some similarity measurement. The similarity measure of data within a single group will be high, while the similarity measure of data across two groups will have a low similarity measurement. One of the most used and well known clustering algorithms is k-means clustering. The k-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. The goal of k-means is to minimize the sum of the squared error over all k clusters. The k-means algorithm will always converge to a local minimum which is not necessarily the global minimum. Hence k-means is not deterministic.

K-means clustering is used to create a codebook based on the SIFT feature descriptors. The codebook serves as a reference guide to discriminate between vectors by determining their nearest cluster centers. The closer a vector is to a cluster center the more similarity there is between the vector and the center. The number of clusters is equivalent to the codebook size.

2.1.4. SIFT

One method for creating the BoVW codebook mentioned in section 2.1.3 is to use the scale-invariant feature transform (SIFT). SIFT is a local feature extraction algorithm which is invariant to translation, scale, rotation, image noise, and partially invariant to illumination changes [22]. The SIFT algorithm has four major stages: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint descriptor representation.

In the scale-space extrema detection stage the image undergoes four rounds of Gaussian smoothing. The difference of the Gaussian (DoG) of each round is calculated and returns four DoG representations. A pixel in one DoG representation is checked to see if it is a local maxima or

minima with its eight neighbors. Then the same pixel is compared one scale higher and one scale lower with its nine neighbors to see if it is still a local maxima or minima. If the pixel is a local maxima or minima then it becomes a keypoint candidate.

The keypoint localization stage begins by interpolating nearby detected keypoints. The Taylor series expansion up to the x^2 term is computed for every detected extrema from the previous step. If the offset is greater than 0.5 if any dimension the extrema is discarded because the point will not be an extremum. Once each extremum has been identified the algorithm will discard each extremum with low contrast. The last operation of the keypoint localization stage will eliminate keypoints that are poorly located but appear to be an edge/boundary within an image.

The orientation assignment stage will take the remaining keypoints from the previous stage and assign them a direction based on their surrounding nearest neighbor pixels. A 36-bin histogram is formed, each covering 10 degrees for a total of 360 degree range of orientations. This histogram is populated from all scales. If the orientation corresponding to the scale is within 80% of the highest value then this orientation becomes to dominant gradient of the keypoint. Figure 5 shows SIFT keypoints that have been identified by the SIFT algorithm.

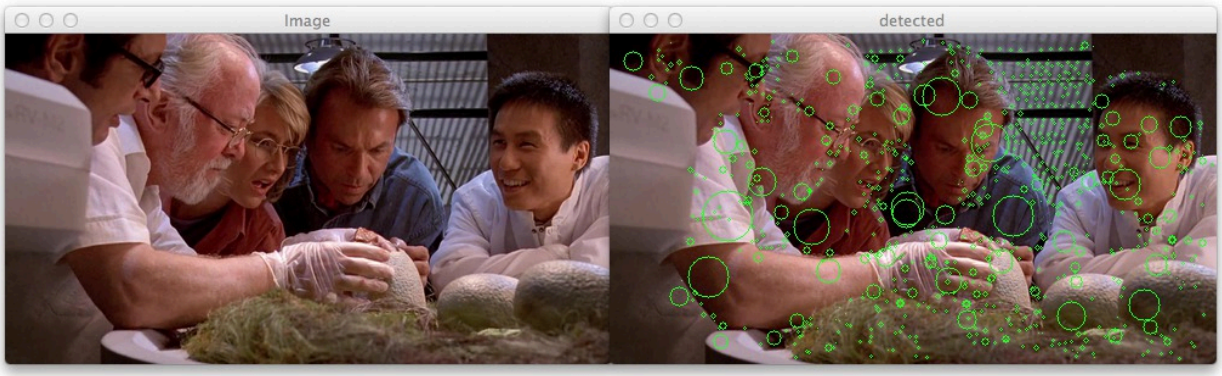

Figure 5. SIFT Keypoints [23]

The keypoint descriptor stage takes the dominant gradient found from the previous stage and creates a descriptor vector. Each keypoint is decomposed into a 16x16 grid, which is then decomposed again into 16 4x4 grids. An orientation histogram is computed for each 4x4 region. Each orientation histogram has a total of eight directions. Thus a final keypoint descriptor is composed of 16 of these 8-bin histograms. This keypoint descriptor is represented as a 128-dimensional vector.

Additionally, SIFT has been further improved with the RootSIFT algorithm. RootSIFT performs and element-wise normalization of SIFT features and then takes the square root. This brings the SIFT algorithm up to the performance obtained by the more recent Hellinger or Chi-square histogram comparison methods [24]. RootSIFT is superior to SIFT in all cases and can be easily implemented without incurring additional computing or storage costs. This framework makes use of RootSIFT as the default image feature extractor.

2.2. Image Retrieval

The image comparison methods drive image search and retrieval operations in digital archives by returning identical or similar images to an original query. The comparison methods can be applied independently or in stages to achieve desired results.

2.2.1. CBIR

CBIR fuses the disciplines of computer vision, machine learning, information retrieval, human-computer interaction, database systems, statistics, and data mining. These disciplines are used to identify image content, quantize content, and implement one or more similarity metrics to drive CBIR operations. Figure 6 depicts the complexity that is a CBIR system.

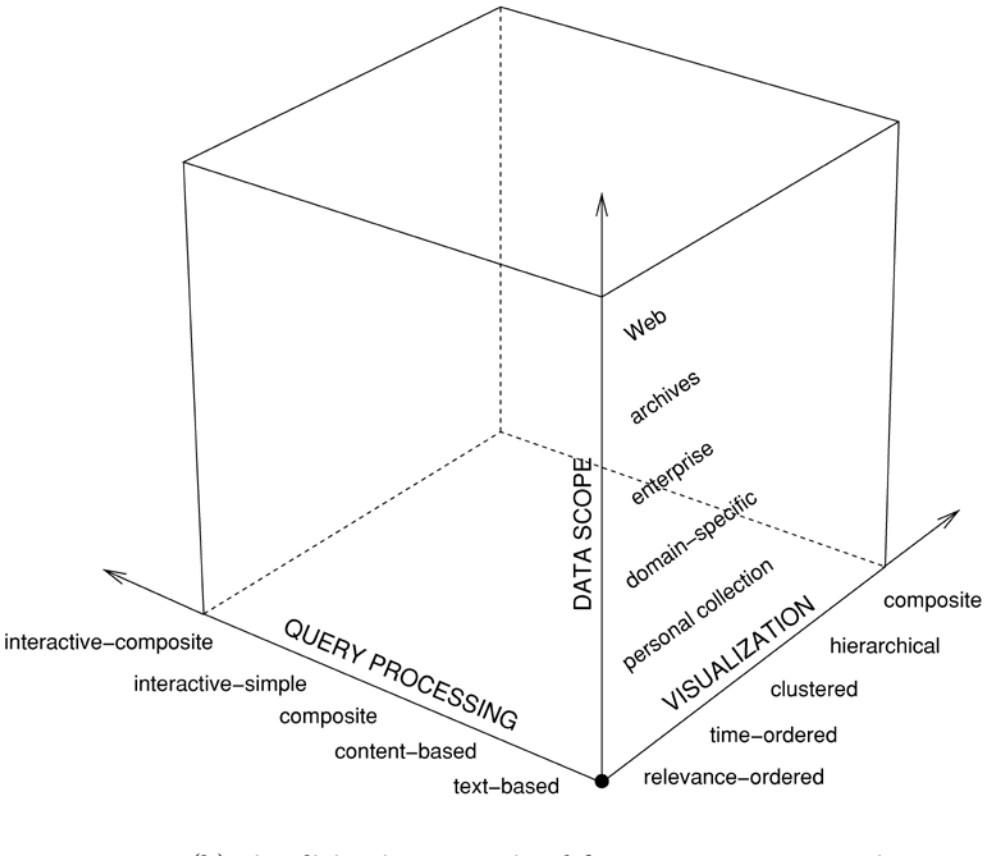

Figure 6. CBIR System Visualization [18]

Another implementation of CBIR would calculate the similarity measure comparison by comparing text-based tags generated automatically or manually by a user [18]. The popular photo sharing website Flickr performs CBIR by allowing users to manually annotate images created and uploaded by other users on the network [18], [25]. Tags can also be assigned automatically by extracting features and comparing them to known quantized features obtained through computer machine and or machine learning algorithms [26].

CBIR systems are built to implement one or more factors from each dimension depicted in Figure 6. Once a CBIR system's coding and integration is completed it cannot be modified to implement other factors from Figure 6 without significant software or hardware changes. A CBIR

framework can provide a degree of flexibility when it comes to designing and implementing CBIR functionality.

2.2.2. CBIR Frameworks

A framework offers modular options for a user to accomplish a task within a specified set of constraints. Frameworks for CBIR allow a user to extract and quantize image features, store the quantized data in a database, and return similarity measure results based on a user query to the CBIR system. The authors created feature vectors by concatenating color, texture, and shape information into a single vector. A similarity measurement between query and database images is calculated using the Euclidean distance and each vector can be weighted. The authors also state and cite that most CBIR systems use a shape descriptor in one form or another.

In [27] the authors use multiple shape features to perform image search and retrieval. The first, second, third, and fourth shape moments are extracted which creates a global shape descriptor. They also implement probabilistic classification clustering to classify a global shape descriptor as a known class. This probabilistic classification is called data mining by the authors. Once data mining is complete and a class is known a comparison of the query image to the known images of that class is conducted.

The authors of [28] implement a multimodal CBIR framework using speeded up robust features (SURF), contour and edge detection, and text search. Thus their CBIR framework makes use of not only image content, but also non-image feature metadata data as well. The contour and edge detection portion of the framework allow a user to draw an image on a tablet which then extracts the contours and edges to determine if a similar image already exists in the database along with the extracted SURF features and text search to create a multimodal CBIR system.

The frameworks of [27-29] all make use of multimodal image features. However [28] uses textual data not associated with extracted image features, in the form of image tags, or metadata, which are already annotated on images to further refine the CBIR search and retrieval operation. The proposed framework will make use of image features along with metadata data as in [28]. However the metadata will be collected from sensors which collect environmental information at the time of image capture. This sensor data will be used in the proposed modular framework to refine search data in order to improve search retrieval and accuracy. The use of extracted image features and correlated metadata is known in this thesis as assisted CBIR. Assisted CBIR extends previous approaches since the CBIR system is using not only content from within the image but also outside data to assist in searches.

In [29] the authors create a framework in which the CBIR system relies on multiple types of image features, which are called multimodal features. The CBIR system extracts color, texture, and shape features from an image. Multimodal features are used in CBIR systems because [30] demonstrated that CBIR systems which use only one type of image feature, i.e. only color, do not perform as well as those which use multiple features. Thus the framework presented aims to make use of more than just a single feature.

2.2.3. Assisted CBIR

Assisted CBIR has been implemented and used in the medical field for a decade, however it has not been named as such. CBIR can be used to extract, search, and compare images from one patient's medical photo to another patient's medical photo to assist in diagnosis [31-32]. In [31] the authors implement a CBIR system to analyze image features and metadata individually and

also combined (assisted CBIR) to compare the results. The assisted CBIR performed better than the metadata retrieval and image feature retrieval separately.

In [32] the authors survey the state of CBIR in the medical field. They investigate multimodal and non-image data (assisted CBIR) image retrieval. The authors come to the conclusion based on their survey-like approach that combining the non-image data with the multimodal data improves the CBIR system performance.

CBIR is used to support applications as in [33] the authors use latitude and longitude data to automatically annotate new images captured and uploaded to Flickr. Thus these new images within a similar geographical location will be tagged with labels from other already tagged images that currently exist in the Flickr database. The system replicates this work to demonstrate its usefulness and adds Wi-Fi access point lookups through the Wigle API [34].

2.3. Summary

CBIR has been implemented successfully in large deployments such as Google and other mainstream search engines. CBIR has also been used for medical purposes to aid patient diagnoses and has been discussed for use in law enforcement activities. These implementations are static and are unable to change or adapt based on the needs of an organization or user. This gap in adaptability, or hence force called extensibility, is examined in the next section.

This extensibility gap in CBIR systems also opens up the possibility for a CBIR system to drive an independent application. An independent application is one that completes some function using the output of the CBIR as its input. This is discussed in section IV and section V.

III. Framework and Approach

Assisted CBIR will extend the ability of autonomous entities to understand the environment by linking the content of images with environmentally sensed inputs. In this section, the high level design process for the Multi Sensor Assisted CBIR System (MSACS) is discussed by developing high level requirements, describing the system and its key components, providing high level software design documentation information, and prototyping an application to demonstrate the utility of a framework.

3.1. High Level Framework Requirements

The framework will need to ingest, parse, and maintain large amounts of text data related to image and sensor features from multiple input sources. This drives four requirements. First, the data obtained from the sensor modules must be associated with an image and incorporated in search and retrieval. Thus at a minimum an image is required to perform the most basic form of CBIR which works solely on extracted image features. With this, the core functionality of image retrieval is necessary and the goal is to support simple or complex processing algorithms to do this, such as state of the art for CBIR, the BoVW paradigm.

Second, the framework has to be extensible. The extensible sensor modules can be Python files that contain, at a minimum, a heartbeat function, a data reading function, and a database populate function. A configuration file associated with the framework will be used to point these modules to the framework at the time of operation. The heartbeat function lets the framework know that the module is intended for the framework.

Third, data must be scalable and portable. To keep the size of the data as small and as portable as possible the framework will work with common data formats such as JSON and XML.

JSON will be the default data format in the system as it is more lightweight than XML. As a result of this all XML data in the system is converted to JSON as it is parsed. A scalable, distributed database will allow this data to propagate efficiently if used in a network. The database function tells the framework how to populate the database with the incoming module information.

Fourth, raw and processed data needs to be exposed to support applications. In this use case a user queries the MSACS application programming interface (API) with data from a camera, Wi-Fi sensor, and magnetometer. For example, a user may perform self-localization in an unfamiliar environment. The localization application can access data through the API. The following summarizes the high level requirements for MSACS:

1. The framework shall support image-based CBIR using simple or complex processing, such as BoVW, and incorporate these additional inputs to improve search and retrieval. (Assisted CBIR).
2. The framework shall be extensible, supporting multiple sensor inputs and processing of these inputs. (Extensible)
3. The framework shall use a standardized data format, and be supported by a scalable, common database across platforms. (Scalable and Portable)
4. The framework shall support post processing to enable image annotation and application interfaces. (API)

These requirements serve as the basic design concepts for the prototype of MSACS. They address the basic high level design decisions in the system, detailed next.

The current trends in CBIR systems and frameworks are moving towards multimodal feature extraction and detection. The authors of [29] use multiple features that are extracted from within the image to perform search and retrieval operations. These features are color, texture, and

shape. In [27] the authors combine machine learning techniques such as classification in addition to the shape features. In [28] the authors use external inputs such as user-created sketches and keyword searching in addition to the traditional CBIR image operations. All of these frameworks are rigid and do not allow the extensibility of feature types. The proposed framework will introduce modularity to enable new information to be collected about an image at the time of capture. This new information will be environmental data collected by additional sensors located at or around the same location as a camera.

Assisted CBIR could be considered as an offshoot of traditional CBIR. In Table 3 from [18] the authors document a qualitative analysis of CBIR offshoots and applications. This table is extended to include the use of sensors and potential applications and is reflected in Table 1. The user feedback column has been omitted because it is optional in the qualitative analysis. The remaining categories are either desirable or essential. These qualitative design requirements drive the assumptions and requirements choices for the framework.

Table 1. Assisted CBIR Qualitative Requirements Analysis

Applications & Offshoots	Similarity Measure	Machine Learning	Visualization	Scalability	Sensors	Applications
Assisted CBIR	Essential	Essential	Desirable	Essential	Essential	Desirable

3.1.1. Assumptions

The following assumptions are made for MSACS:

1. All data generated by sensors are in the form of XML or JSON, except the camera/video data, which can assume multiple image formats (.gif, .jpg, .png). These data formats are widely used, extensible, and interoperate with several different programming languages.

2. For the purposes of the prototype a single Cassandra database is assumed to support proof-of-concept operations.

3. For the purposes of the prototype a small scale deployment of the system is assumed, i.e. one camera which connects to the database from the 2nd assumption. Additional sensors are not implemented in the prototype to highlight the proof-of-concept operations.

3.2. High Level Framework Description

The concept of multi-sensor assisted CBIR is the integration of sensed information in the environment at the moment of image capture within the indexing, search, and retrieval process. This concept can be easily distributed in its operation, allowing for populate/update operations to be distinct from query/response ones. In this way known information from the populate/update stage can be linked to real time queries. The results from the queries can be used to propel applications that gain situational awareness of the environment through image repositories.

This concept is illustrated in Figure 7. Cameras and sensors in the bottom half of the figure may publish images annotated with sensed information within a populate/update use case. Continual updates are vital to maintain the world model consistency with reality. The top half of the figure represents use cases that support queries/responses from autonomous entities and users.

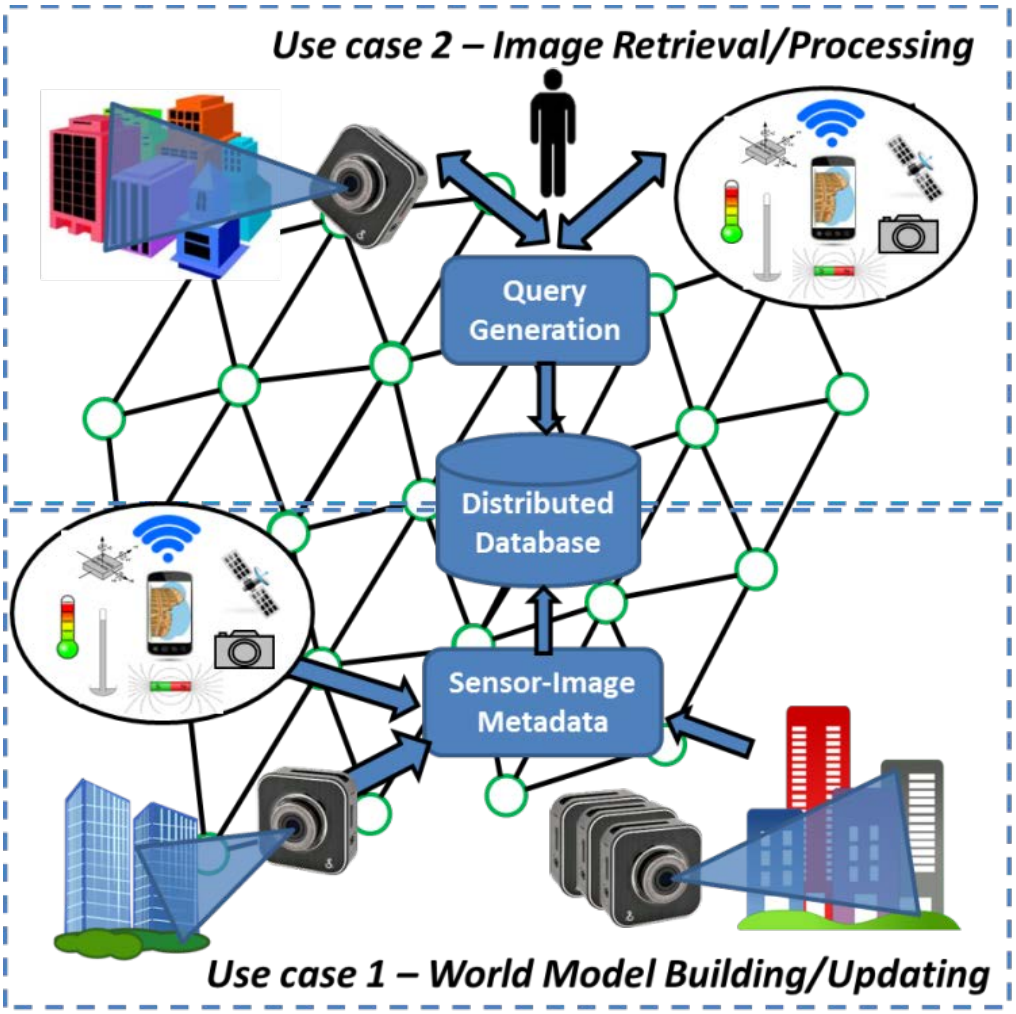

Figure 7. High Level System Connectivity

The concept aims to support image queries to receive similar images as returned values. Non-image queries such as Wi-Fi, magnetometer, and other potential modules can also retrieve images of relevance. These are built upon a distributed network and distributed database to support scalable world model building and querying across a region of interest.

The core of the MSACS is displayed in Figure 8 and Figure 9. As shown in Figure 8, MSACS incorporates data obtained from additional sensors. As data is captured it is stored in file-based folder structures enabling it to be assigned a time stamp (t_0, t_1, …, t_n) for first-in first-out

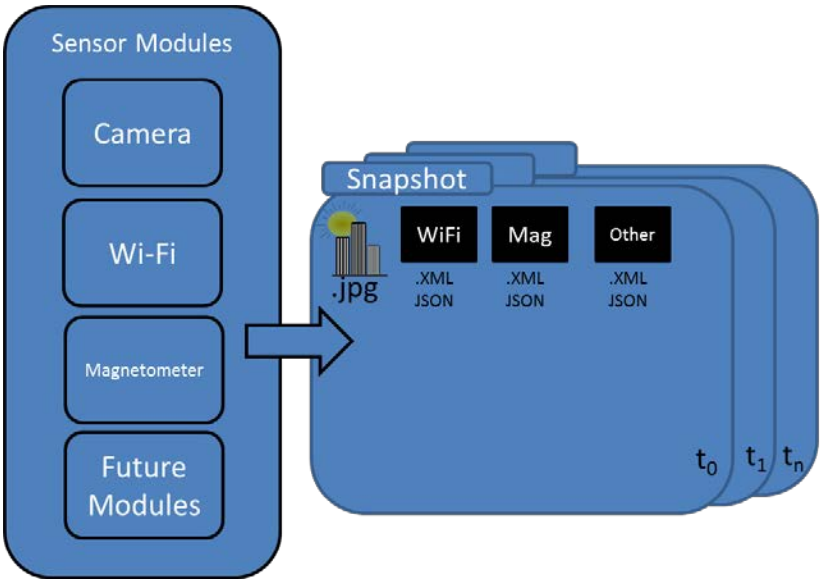

Figure 8. Proposed Assisted CBIR Modules

processing operations. This way all correlated data remains together. The types of captured data processed depend on the modules currently loaded and incorporated into the framework.

All data is read into the framework as XML or JSON, with the exception of the image capture. Each sensor has a specific framework module which parses sensor-related data to extract and then quantize information and store it in a database. For illustration purposes the data presented in Figure 9 is shown as XML. After quantization the individual feature vectors are appended to the image as annotations and archived for later retrieval. The database will be available for search and retrieval operations once it contains data. This data supports a localization application tied to the first use case.

The framework is not intended to be limited to a single database. However, for the purposes of this prototype, only a single database is used. Cassandra databases have the ability to shard and exchange information depending on their configuration. Cassandra is used for applications that require high availability and have the potential to scale. Thus, Cassandra is selected as the implemented database due to the potential scaling needs.

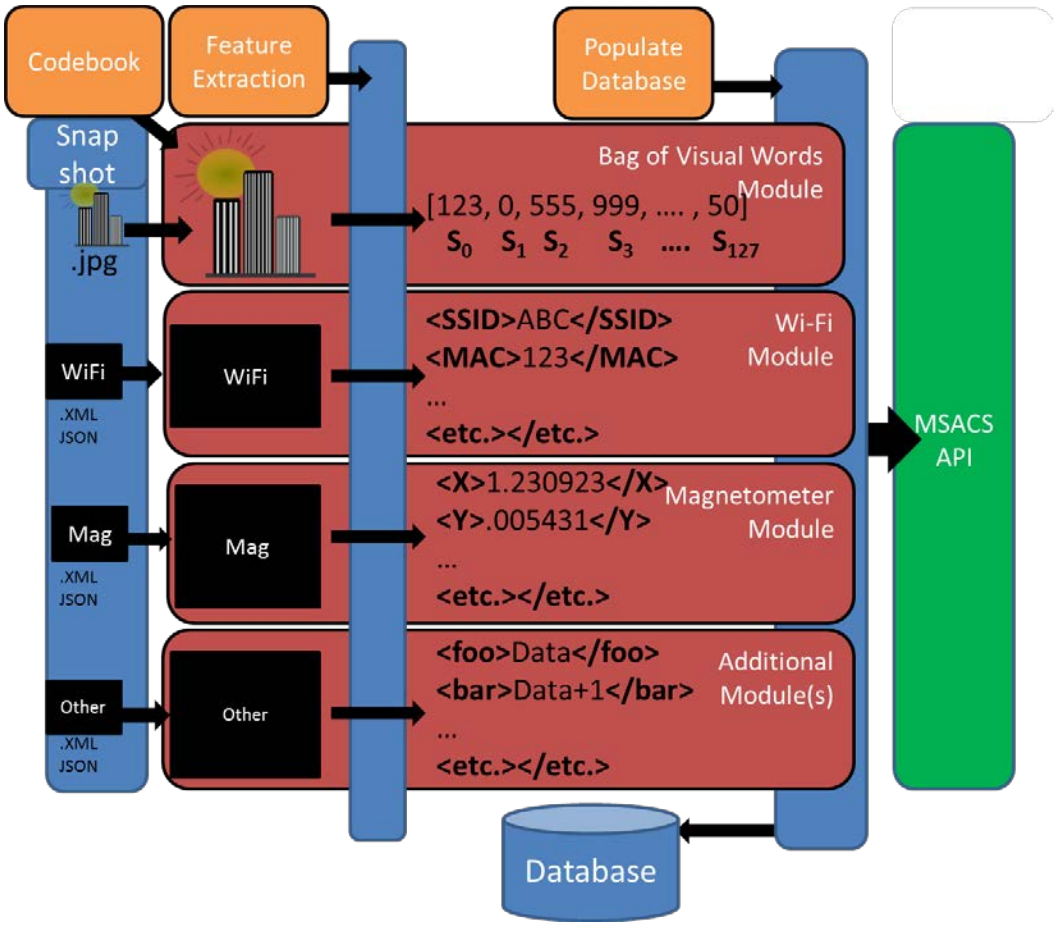

Figure 9. Proposed MSACS Framework

On the right of Figure 9, applications can access the MSACS API to make use of the assisted CBIR data. For example, a localization application can utilize the known latitude and longitude information in the database and provide an inference of the latitude and longitude for the query image. This is accomplished by averaging the latitude and longitude of the images in the return set. The cardinality of the return set is user defined in a framework configuration file. The performance of the localization application is affected by the return set size. The result of the localization application is a pair of latitude and longitude coordinates returned to the user.

3.3. Development Approach

The development approach will create software documents which describe the system from the ground up. A software design document (SDD) detailed in Section IV will use IEEE standard 1016. IEEE 1016 describes the standards for system and software design descriptions. The SDD is a representation of a software design to be used for recording design information and communicating that design information to the key design stakeholders. The SDD is typically prepared to support the development of a software item to solve a problem, where the problem has been expressed in terms of a set of requirements [11].

3.3.1. Requirements Specification

The SDD detailed in Section IV will use IEEE standard 1016 and ISO 29148-2011 as the basis for design. ISO 29148-2011 describes standards for requirements engineering for systems and software engineering projects [35]. The requirements are typically specified in a separate document from the SDD. However this thesis details the requirements of the MSACS framework as a subsection (4.3 Software Requirements Specification) within the SDD.

3.3.2. Modeling

The SDD also presents several figures to further detail and explain the desired functionality and aid in the interpretation of the created requirements. As such, these figures are intended to guide development and are not concrete depictions of operation.

3.4. Evaluation Approach

A prototype of the proposed system is evaluated. This prototype implements subsets of the requirements presented in Section IV in order to create and execute a CBIR system that performs basic search and retrieve operations. This prototype also provides input to a prototyped localization application.

3.4.1. Prototyping

A prototype has been created which validates the utility of the framework. This is covered in-depth in Section V. The prototype is an implementation of a CBIR system using BoVW. Different vocabulary sizes are used to generate search metrics in the form of precision and recall. The results of the BoVW-only approach are promising. Implementing the assisted CBIR functionality within the framework will theoretically improve search and retrieval results to result in more accurate return sets. This will be accomplished by either adding weights to the different search parameters or filtering the results based on the existing sensors.

3.4.2. Application

The CBIR prototype described above will drive a situational awareness application. This application will use the results of the CBIR prototype operation to determine a location in latitude and longitude coordinates. The application will interface with the MSACS API in order to obtain positional data. The application can be user-defined and is not limited to what is presented in this thesis. The intent of the application is to utilize the results of the assisted CBIR operation to drive user-defined application.

3.5. Conclusion

The MSACS framework is intended to be a distributed network of sensors and databases which store sensor data and respond to queries from a user. The response results can then be fed through the API to drive some user-defined application. The proposed SDD is based on IEEE 1016 and also implements requirements per the ISO 29148-2011 standard. Multimodal, or assisted CBIR systems have been implemented and evaluated before. However none have taken a modular, extensible approach to implementing individual modalities in the system. The development approach will implement an extensible module-based framework based on a prototype developed for this thesis.

IV. Software Design Document

Assisted CBIR will extend the ability of autonomous entities to understand the environment by linking the content of images with sensed inputs. In this section the design process for the Multi Sensor Assisted CBIR System (MSACS) is exposed by developing requirements, describing the system, and detailing its key components.

The framework design approach is from the bottom up with a focus on extensibility and modularity. Because of this the composition viewpoint from the IEEE 1016 standard is used. As described in section III, this standard lays the foundation for designing software systems.

4.1. Purpose

The purpose of this SDD is to describe the implementation of the MSACS software framework described in Section III. The MSACS framework is designed to be an extensible, modular system which performs assisted CBIR to support user-defined applications. The modularity aspect of the framework will utilize one or more sensor modules to collect data from the surrounding environment. This data will be correlated with an image file and saved in a database for future search and retrieval operations. This SDD contains more information than a traditional SDD does to emphasize requirements for the various subsystems that comprise the framework.

4.1.1. Scope

This SDD describes the implementation details of the MSACS framework. The framework will enable assisted CBIR using one or more sensor inputs to drive search and retrieval operations in a situational awareness scenario. The results of the search and retrieval operations are then fed

into an application. This SDD also covers the requirements for the MSACS framework and associated the subsystems.

4.1.2. Definitions

Many terms in this SDD are defined here to clear up any confusion related to the components, subsystems, and pieces of the MSACS framework.

Application – A program which serves to complete a function.

Assisted CBIR – Performing CBIR using multimodal information obtained from sensors in addition to extracted image features.

CBIR – Content-based image retrieval uses the features contained within digital images to match to other images.

Multimodal – Using more than one type, or mode, of data.

Sensor – A device which collects data from the surrounding environment.

4.2. Design Overview

4.2.1. Description of Problem

There is currently no framework which supports extensible assisted CBIR operations. A framework is ideal to create an extensible, modular approach to assisted CBIR. This modular approach allows a user to create a sensor module file to interface with the framework which is fed real world data. The framework will then parse, extract, and store information related to that sensor module and associate all extracted data with the corresponding image from the time of capture.

4.2.2. Technologies Used

The MSACS framework will communicate with one or more Cassandra databases. The only required device for minimum functionality is a computer capable of running and executing programming languages and Cassandra databases. The computer can be Windows or Linux due to the open nature of Cassandra.

4.2.3. System Flow Diagrams

Figure 10 through Figure 12 depict high level operational flow of the framework. Figure 10 shows a user interacting with the framework through an API. The rest of the framework operations occur in the backend.

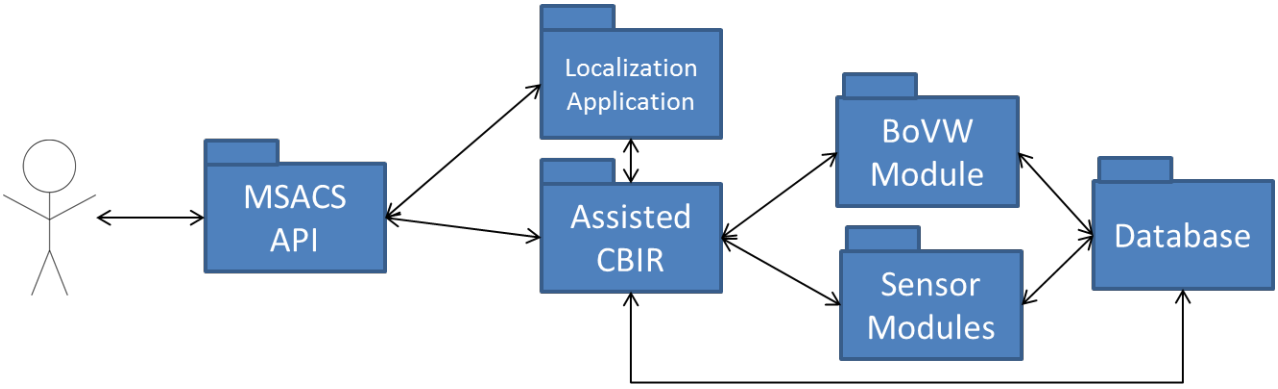

Figure 10. System Overview

Figure 11 shows the high level operational flow of image annotation process. When a sensor captures information it is ingested into the framework so the features can be extracted. At a minimum there needs to be at least an image captured at a sensor's location. Once an image has been captured and the multimodal data, if it exists, if ingested too. Then this information is extracted and stored in a database.

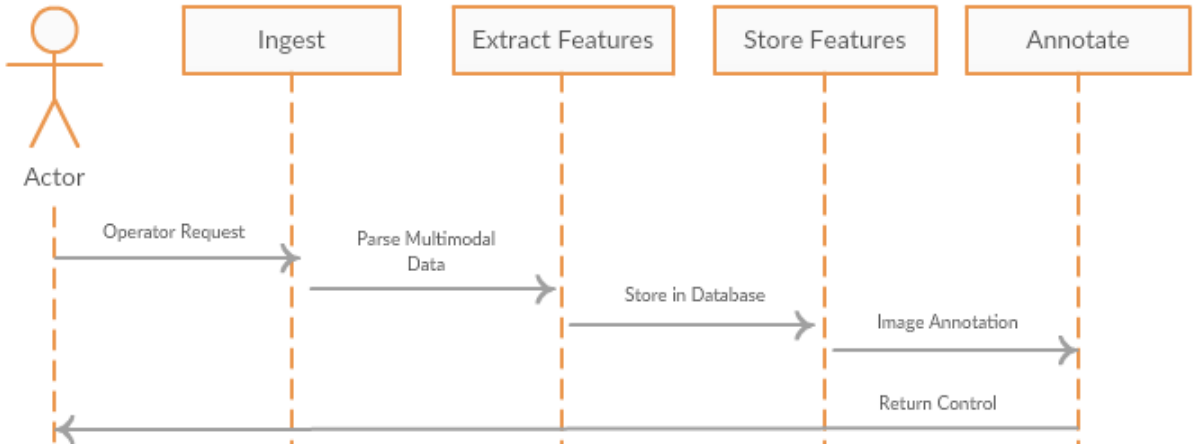

Figure 11. Image Extraction

Once the information is stored in a database the image is then annotated with all of the supporting sensor data and archived.

Figure 12 depicts a user querying the framework to obtain a result. First the user uploads an image and the image is parsed for features. These features are then checked against the stored features in a database and then filtered based on the user's weighting, if any, of the sensor importance. Once filtered the return set of images is displayed to the user.

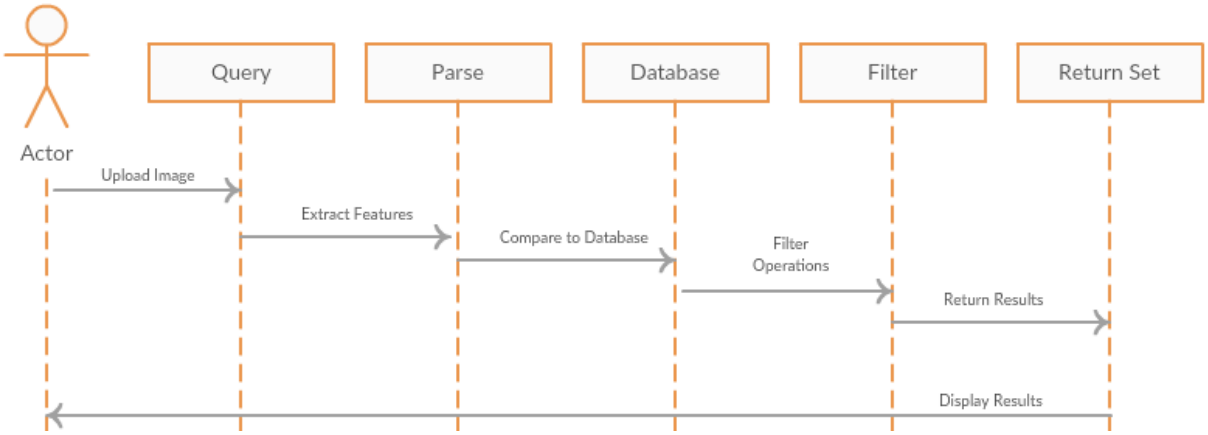

Figure 12. Query Procedure

4.3. Software Requirements Specification

4.3.1. Framework

The framework requirements describe the major operations performed by the system. This includes describing CBIR, extensibility, common data formats, and how a user interacts with the framework. The user will interact with the API and the framework will handle the other operations behind the scenes.

The requirements related to the MSACS framework are labeled as MSACS-F-##, where ## is the numerical designator of the requirement (i.e. 00, 01, etc.).

Table 2. Framework Requirements

Requirement:	**Requirement Description:**
MSACS-F-00	The framework shall implement CBIR for images using the BoVW paradigm.
MSACS-F-01	The framework shall be extensible by accepting sensor (module) files. Each module file will contain functions needed to interface with the framework. This proposed relationship is illustrated in Figure 13.
MSACS-F-02	The framework shall use a standardized data format, and store information within a scalable, common database across platforms.
MSACS-F-03	The framework shall implement post-processing image annotation by appending sensed information to an image. This is illustrated in Figure 14.
MSACS-F-04	The framework shall interface with one or more users by presenting an API.

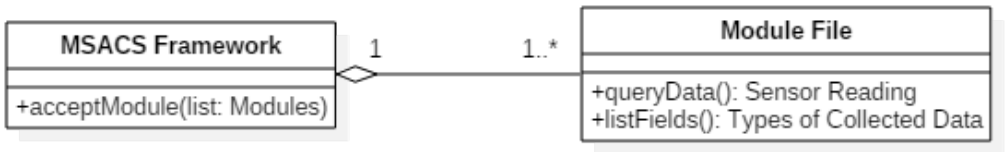

Figure 13. Proposed Member Functions.

The framework's extensibility will allow it to ingest new custom module files. These module files contain functions as depicted in Figure 13 which collect data and also provide the framework with a list of the types of data collected. This list will be used to create new columns in the database tables.

After an image has been collected it will be annotated with all associated multimodal sensor data. Figure 14 depicts a potential implementation of image annotation. When an image is captured and processed the data associated with it is appended to the image. The figure depicts this with an XML-like implementation for visualization purposes. The image annotation function is intended to be used for archiving images.

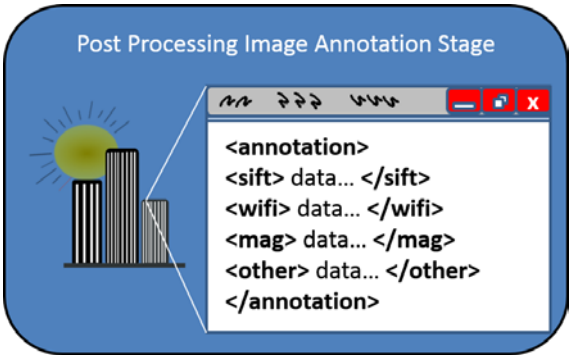

Figure 14. Proposed Annotation Scheme.

4.3.2. Assisted CBIR

Assisted CBIR uses multiple sensor inputs in addition to traditional image-based CBIR to refine search results. The sensor inputs are described in the sensor module fields and are incorporated into the framework. Additionally the different sensor inputs can be weighted based on user preference to aid in searching. Once a return set is computed the user will be able to filter the results based on the sensor type.

The requirements related to the MSACS assisted CBIR are labeled as MSACS-AC-##, where ## is the numerical designator of the requirement (i.e. 01, 02, etc.).

Table 3. Assisted CBIR Requirements

Requirement:	**Requirement Description:**
MSACS-AC-00	The framework shall execute assisted CBIR using not only the extracted image features but also the available multimodal sensor data associated with each image.
MSACS-AC-01	The assisted CBIR subsystem shall support weighting of the multimodal search parameters from 0.0 to 1.0. Thus all search parameters added together shall equal 1.0.
MSACS-AC-02	The assisted CBIR subsystem shall filter results to obtain return sets (i.e. apply filter with sensor X to see results with X, then filter with sensor Y to obtain a result set in which X and Y intersect).
MSACS-AC-03	The assisted CBIR subsystem shall present the results of the search and retrieve operation to the user in the form of images returned by the search query.

4.3.3. Modules

The framework module files contain information on the type of data they collect and are tailored for each type of sensor they support. For example a Wi-Fi module may collect information on SSID, MAC addresses, etc. to support Wi-Fi collection. This means every time the Wi-Fi module collects data all of this supporting data related to Wi-Fi are collected, if available. Each module is intended to provide one reading from the environment for each image that is captured by the system. The module files are programmed by the user and are constrained to use either the XML or JSON data types.

The requirements related to the extensible modules are labeled as MSACS-M-##, where ## is the numerical designator of the requirement (i.e. 01, 02, etc.).

Table 4. Module Requirements

Requirement:	Requirement Description:
MSACS-M-00	A module file shall obtain data from exactly one sensor to support assisted CBIR using sensor data acquired: 1. Real-time 2. From a saved file
MSACS-M-01	Data obtained from a module shall correctly to exactly one image.
MSACS-M-02	Each module file shall provide the framework with a list of the data identifiers (i.e. if a module collects Wi-Fi data then the module shall pass a list of all the data the sensor collects such as SSID, MAC Address, etc.) associated with the type of data the sensor will collect.

4.3.4. Application

The results of the assisted CBIR operation can feed into a custom application. The applications are meant to be customizable and generated by a user. A localization application example can take the results of image search and retrieval to estimate a location.

The requirements related to the MSACS framework application are labeled as MSACS-A-##, where ## is the numerical designator of the requirement (i.e. 01, 02, etc.).

Table 5. Application Requirements

Requirement:	Requirement Description:
MSACS-A-00	The framework shall support an application that uses the results of the assisted CBIR operation to perform localization.
MSACS-A-01	The localization application shall report results to a user in latitude and longitude format

4.3.5. CBIR

CBIR forms the basic functionality of the framework operations. The CBIR system is intended to extract image features, perform feature clustering, and create codebooks to aid in search and retrieval operations. At a minimum the CBIR system will need only an image to execute search operations.

The requirements related to the MSACS CBIR operation are labeled as MSACS-C-##, where ## is the numerical designator of the requirement (i.e. 01, 02, etc.).

Table 6. CBIR Requirements

Requirement:	**Requirement Description:**
MSACS-C-00	The CBIR operation of the MSACS framework shall implement a BoVW approach to image search and retrieval.
MSACS-C-01	The CBIR system shall recalculate a codebook using a corpus of images.
MSACS-C-02	The CBIR operation shall be extensible to allow for multiple feature extraction algorithms. At a minimum this includes: 1. SIFT 2. RootSIFT
MSACS-C-03	The CBIR operation shall be extensible to allow for multiple clustering algorithms. At a minimum this currently includes: 1. K-means
MSACS-C-04	The CBIR operation shall provide the user with precision metrics based on Equation (2).
MSACS-C-05	The CBIR operation shall provide the user with recall metrics based on Equation (3).
MSACS-C-06	The CBIR operation shall allow the user to specify the number of images in the return set from at least 1 image up to N images where N is the cardinality of the corpus.

$$Precision = \frac{|\{Relevant\ Images\} \cap \{Retrieved\ Images\}|}{|\{Retrieved\ Images\}|} \quad (2)$$

$$Recall = \frac{|\{Relevant\ Images\} \cap \{Retrieved\ Images\}|}{|\{Relevant\ Images\}|} \quad (3)$$

4.3.6. Image Annotation

Image annotation will take all multimodal sensor data and append it to the corresponding image. Annotated images will be archived for later use in search and retrieve operations.

The requirements related to image annotation are labeled as MSACS-AN-##, where ## is the numerical designator of the requirement (i.e. 01, 02, etc.).

Table 7. Annotation Requirements

Requirement:	Requirement Description:
MSACS-AN-00	The image annotation operation shall use all available sensor data and append it to the image capture as an image annotation.
MSACS-AN-01	If no sensor data other than the captured image exists the image shall be appended with an identifier indicating that nothing was available to append to the image.
MSACS-AN-02	The image annotation operation shall archive the annotated image for future use in search and retrieval operations.

4.4. System Diagrams

Several diagrams are illustrated in this section to further detail system design and implementation visions.

4.4.1. System UML Diagram

Figure 15 presents a high-level overview of the MSACS system design. The MSACS framework relies on the distributed nodes to capture and store information. Each distributed node contains at least 1 camera (sensor) and as many additional sensors for Wi-Fi, magnetic field, etc. as required. The distributed nodes are envisioned as creating coverage over a corpus area such as a city or town. The number of required sensors has not been defined in this SDD. This node may also have a local database to store sensor-related data. The MSACS framework implements an assisted CBIR system, which is supported by a traditional CBIR system, one or more applications, and one or more module files. The module files enable the framework to interpret data from the sensors. The CBIR system provides the traditional image feature matching functionality to the assisted CBIR system. Lastly the assisted CBIR system uses the data obtained from the sensors and the image feature extraction to perform the updated assisted CBIR operation. This is where the system executes the search and retrieval operation by using all sensor-related data and image

feature extracted data to obtain a result. This result can be filtered across the different sensor data or weighted to obtain a result.

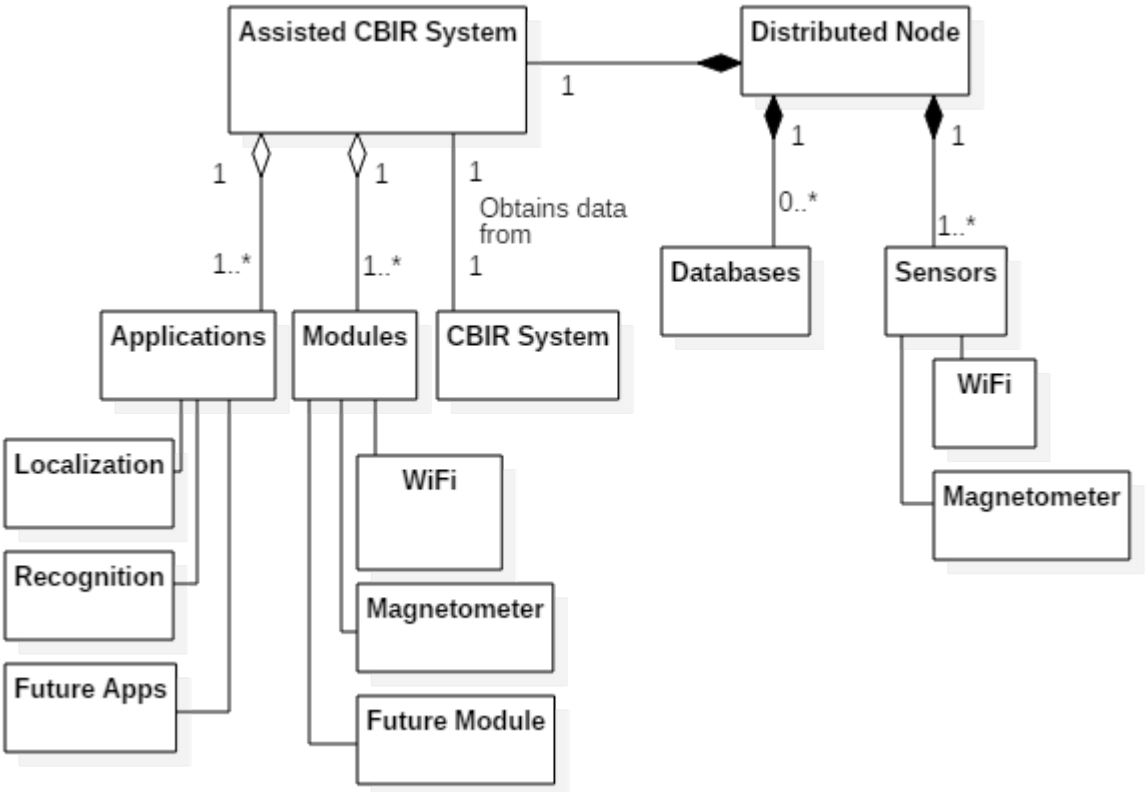

Figure 15. MSACS High Level UML View

The intended use of this system would allow an operator to open the framework, import modules as needed, and then perform assisted CBIR operations to drive some user-defined applications to produce a salient result.

4.4.2. Framework Object Instance

The object diagrams depicted in Figure 16 and Figure 17 show a potential instance of the framework based on Figure 15. The distributed node object is identified by an IP address. The database and sensor objects are identified by port numbers. Thus the database, if one exists at the node, and the sensors can be referenced by their IP addresses and port number. The sensor object

can then be referenced as 213.100.100.1:10000. The sensor is depicted as collecting the SSID and the signal strength in dBm. The CBIR system object provides a BoVW histogram of the returned image. The module object shows the framework is currently using the Wi-Fi module. The fields slot is a list of the data fields of the sensor. In this case the data fields would be SSID and dBm.

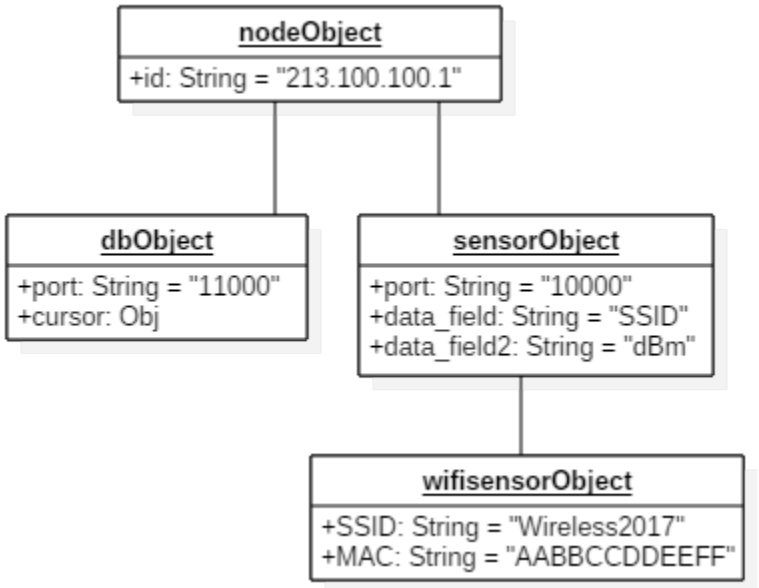

Figure 16. Node Instance Diagram

The application uses the result of the assisted CBIR object to make an inference as to the location of the query image. At this point the framework would then return control to the user and be ready to perform another assisted CBIR operation. Per the description for Figure 15 it is possible to have many sensors and many module files. These additional sensors will feed into and provide fine-grained results for the assisted CBIR operations. The diagram presented for the framework instance is just one of many possible.

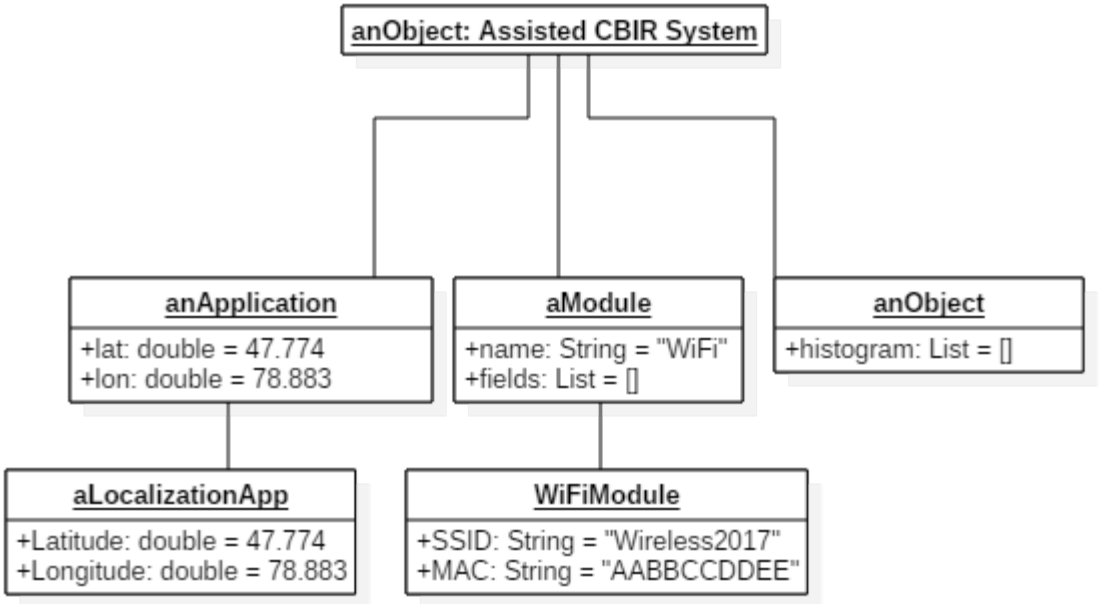

Figure 17. Assisted CBIR Instance Diagram

4.4.3. MSACS Database

The MSACS framework will use databases for information persistence. These databases will report to the framework at the time of query. Per the UML diagram in Figure 15 a node does not necessarily require a database. This means that one database can potentially hold the information of more than one node. Table 8 depicts a potential node configuration with one camera and multiple sensors. This node, Node X, has three sensors. A camera, a Wi-Fi sensor, and a magnetometer. This node also has three entries in its database. Each entry has the extracted sensor features listed. These extracted sensor features are used in search and retrieve operations. The data stored in the database should be formatted as a string.

Table 8. Sample Database Entry

Local Sensors	Image	Histogram	WiFi	Magnetometer
Camera	A.jpg	[12,10,…,45]	SSID: “Wireless2017”	X: 1.277 Y: 2.334 Z: 3.443
Wi-Fi	B.jpg	[100,78,…,12]	SSID: “Wireless2017-2”	X: 1.277 Y: 2.334 Z: 3.443
Magnetometer	C.jpg	[20,21,…,89]	SSID: “Wireless2017-3”	X: 1.277 Y: 2.334 Z: 3.443

Per requirements with the MSACS-C identifier the framework will have a CBIR system that uses extracted image features. The envisioned database format would have the name of the image being the primary key. The name of an image could be a concatenation of the node name and the timestamp from image capture, to uniquely identify an image and tie the image to its origin location and time of day from which it was captured. Since the extracted features form the backbone of the CBIR system a histogram column will exist in the database too. The image’s histogram could be represented as a string of values.

The module specific data identifiers would be located in the module file. These data identifiers contain the names of the data to be collected. For example, if a Wi-Fi module is used one data field could be SSID, another could be MAC address, etc. When the framework ingests a module file it will check to see if the module’s data fields already exist in the database. If the fields do not exist they will be added to the table. When an image is captured from a node these new fields will be populated during data collection and processing.

During a query operation the framework will parse an annotated image for the image features along with the multimodal data. This data will then be compared with the data existing in the databases to determine the best match.

4.4.4. Framework Initialization

The framework initialization state machine is depicted in Figure 18. When the framework is initialized it will attempt to connect to the network that contains the databases and distributed nodes. If this connection is not available the initialization will end without connecting. If the connection is successful and there are no new module files to import into the framework then the ready state is entered. If the connection is successful and there are new modules to import into the framework then each module will be imported until there are no more left to import. Then from the check module state the framework will move to the ready state.

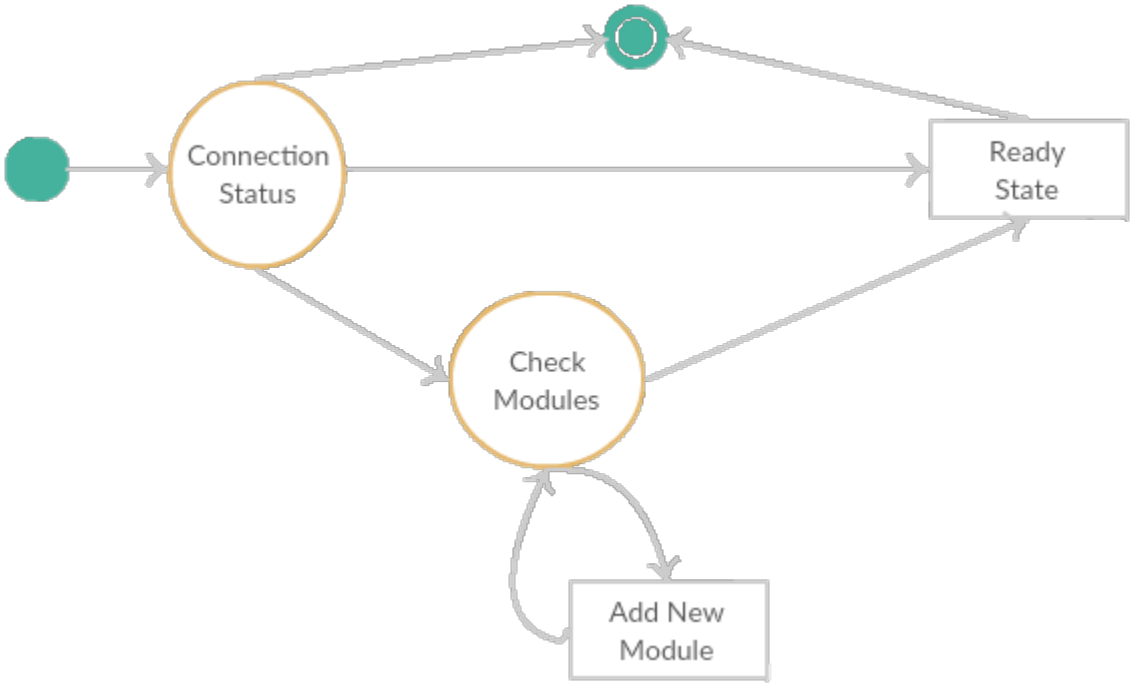

Figure 18. Framework Initialization Sequence [37]

The ready state is when the framework is idle and waiting for a user to query the framework with an input for a result. This is depicted in Figure 19.

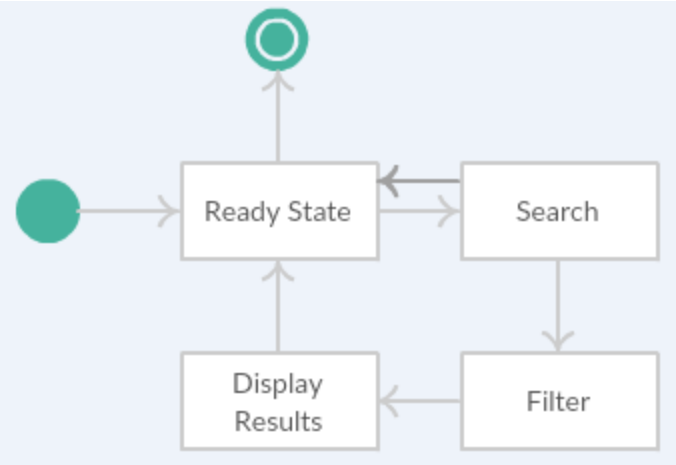

Figure 19. Search Sequence [37]

In the ready state a user can upload an image for search and retrieval. Once an image has been uploaded the assisted CBIR system will search the information repositories for a match. If no match exists then the system will return to the ready state. The user is then able to filter the results based on the types of current modules attached to the system. Once the results have been filtered the results of the operation are then returned to the user. When the user has completed viewing the results the system will move back to the ready state to wait for additional input.

4.4.5. Image Capture and Sensor Fusion

When an image and all available associated sensor data has been captured it will be annotated as shown in Figure 14. Each image, at the minimum, will have the image data. Thus each image, at the maximum, will have data from every available sensor which will be used for image annotation.

4.4.6. Image Annotation

The image annotation process will fuse together the sensor inputs from the time of capture as depicted in Figure 20. Images are annotated for archival purposes and could serve as a backup if a database issue arises.

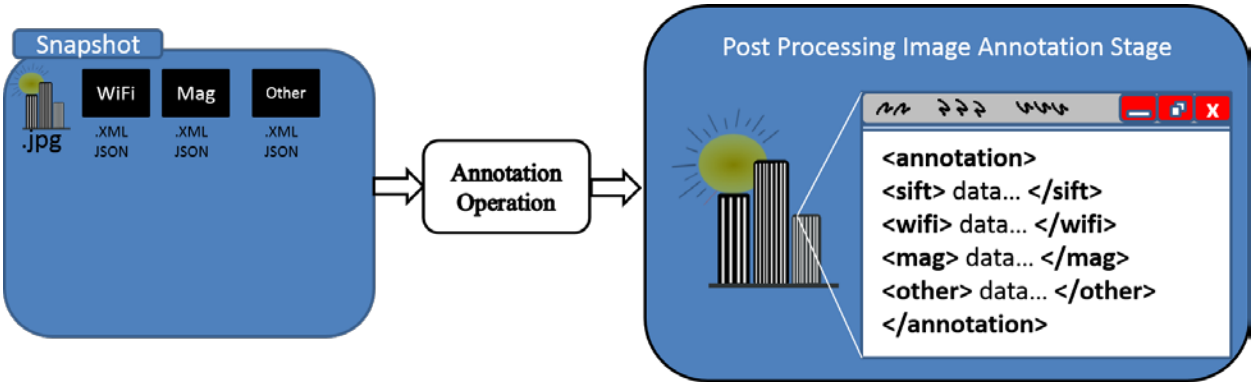

Figure 20. Annotation Operation

4.5. Interfaces

Interfaces are implemented to aid in the extensibility and modularity of the overall system. The user will communicate with the framework by accessing the MSACS API. An example of the framework's interfaces are depicted in Figure 21. This figure depicts the flow of data through the different interfaces implemented in the overall system. The entire framework is wrapped in the MSACS API. This API is used by the end user to interact with the framework. The API is then interfaced with the framework and passes data in a bidirectional manner to the user. The framework interfaces with the modules, the applications, and the CBIR system. This enables the user to update modules within the framework, update applications associated with the framework, and perform search and retrieval operations. This remainder of this section describes and depicts the data traversing these interfaces by illustrating figures with parameters and attributes across each interface.

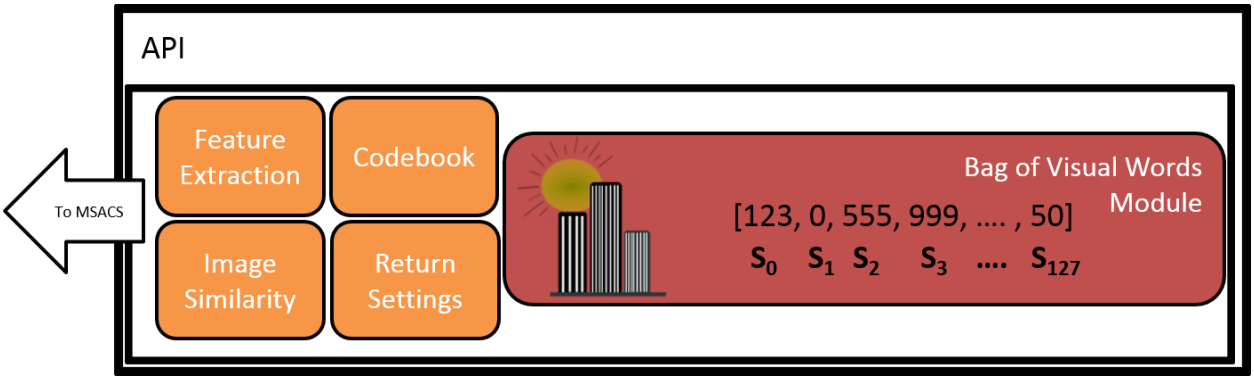

Figure 21. Interface Diagram

4.5.1. BoVW Interface

The API's BoVW interface functions of Table 9 represent how a user communicates with the framework to select BoVW settings. This table details the API functions, their return type, and parameters between the user and the API. The printAlgorithms() and setAlgorithm() functions will print a list of the feature extraction algorithms available and selectable in the framework. This could mean changing from SIFT to RootSIFT, Histogram of Gradients, etc. The codebook size can be printed in addition to creating a new codebook. Since clustering is used for codebook generation there are options to print and set the number of iterations of the k-means clustering algorithm. The number of images in the return set of the CBIR operation can be viewed and selected as well. The sensitivity of the similarity comparison between images can be selected as well from the API.

4.5.2. Wi-Fi Interface

Table 10 details the interface between the framework and the Wi-Fi module depicted in Figure 22. The parameter depicted in the figure refer to the information about a Wi-Fi reading

such as the SSID, MAC, etc. The list of parameters can be changed depending on the application they are needed for. For example if a user wants to only have SSIDs populate the database then this option will be selected. Since multiple Wi-Fi access points can intersect at a sensor's location there needs to be a way to prioritize them. This defaults to the access point with the strongest signal. However if only a single access point is used this may cause issues in the future in this access point is removed or the name changes. Thus the verbosity will let the user select the amount of access points with the highest signal.

Table 9. BoVW API Functions

User Interface Functions	
Return Type	***Name and Parameters***
List	printAlgorithms()
Void	setAlgorithm(Algorithm Name)
Int	printCodebook()
Void	createCodebook(K-means iterations, Number of Clusters)
Int	printKmeans()
Void	setKmeans()
Int	printReturnset()
Void	setReturnset()
Int	printSimilarity()
Void	setSimilarity()

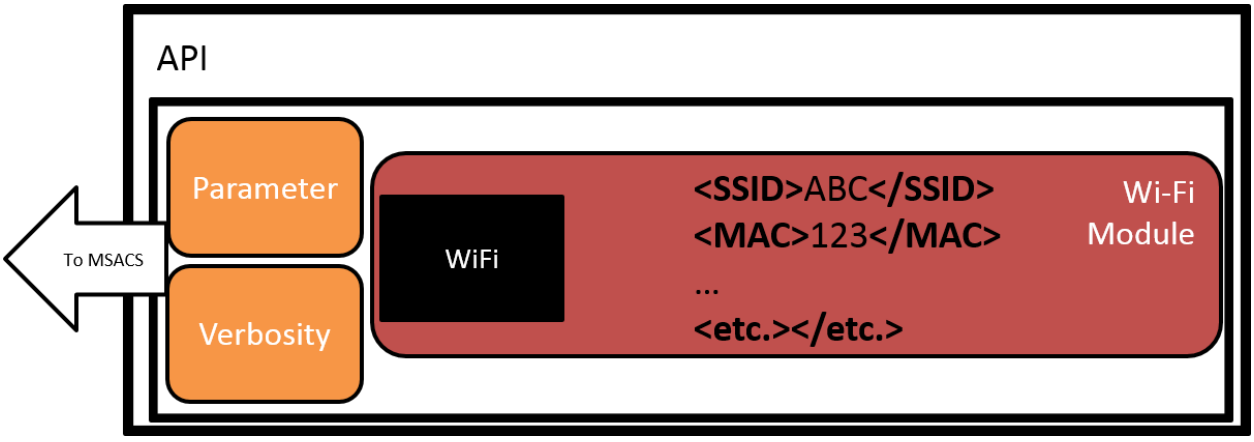

Figure 22. Wi-Fi Interface

Table 10. Module Interface Functions

Framework and Module Interface	
Return Type	***Function and Parameters***
List	getParams()
Void	setParams(List of Parameters)
List	printVerbosity()
Void	setVerbosity(Number of nodes to save)

4.5.3. Magnetometer Interface

Table 11 details the interface between the framework and the magnetometer module depicted in Figure 23. The amount of information stored by the magnetometer module is selectable by the user. Since the magnetometer module can collect X, Y, and Z axis data there are a total of six combinations of these three variables. The user can select which combination to use through the API. For example, the user may only want data from the X axis only. Or a user may want data from only the Y and Z axes. These six options will be listed through a print option and these options are then selectable through the API.

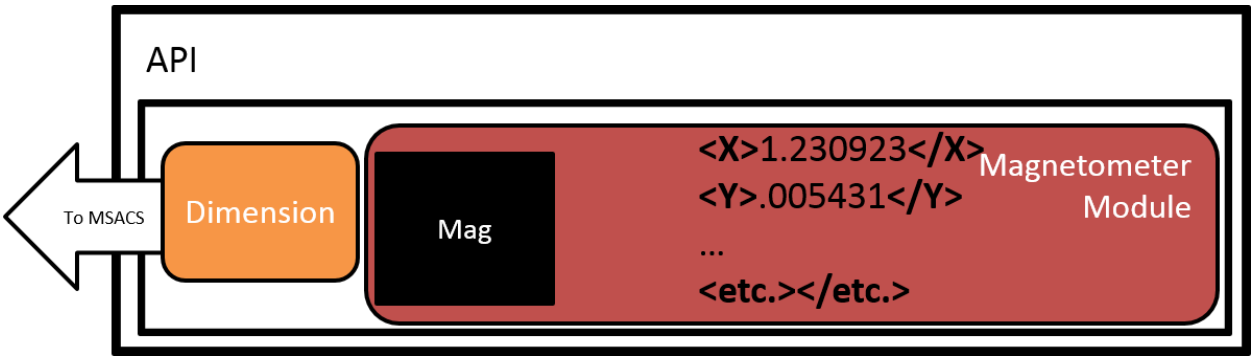

Figure 23. Magnetometer Module

Table 11. Application Interface Functions

Framework and Application Interface	
Return Type	***Function and Parameters***
List	printOptions()
void	setOptions(Option number)

4.5.4. Other Interfaces

The interface between the framework and the future modules is depicted in Figure 24. Each type of module that interfaces with the framework should have some selectable parameters relating to the information collected and processed from the module. In order for this data to be extensible the user will need to access it through the interface.

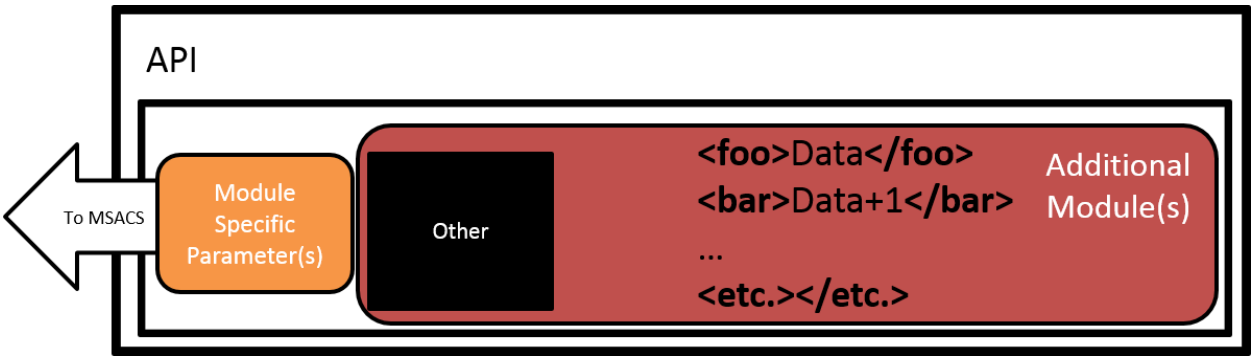

Figure 24. Future Modules

4.6. Conclusion

The MSACS framework is an extensible modular approach to implement an assisted CBIR system. The results of the assisted CBIR system can be fed into one or more user-defined applications to complete some task. The requirements, instances, uses, and interfaces of this system are given to aid in future development with the SDD as a guide.

V. Prototyping and Application

5.1. Core CBIR Implementation

A prototype of the MSACS framework CBIR system and application were implemented in Python. The prototype and application used several open source libraries including OpenCV, SciPy, NumPy, GeoPy, Basemap, and the Cassandra database Python API module. The prototype was implemented on a virtual machine running the Ubuntu 16.04 operating system. An image corpus for the prototype's evaluation was created using the Google Street View API. The API allows the creation of street-level images for a specified evaluation zone based on user-defined routes. This corpus of images was used for prototype and application evaluation.

The minimal CBIR implementation module within the MSACS framework is responsible for extracting point features from images, calculating and creating a visual vocabulary, and expressing the content of images in the form of a BoVW histogram. The BoVW implementation for this experimental evaluation used the OpenCV, Numpy, and SciPy open source libraries to complete these tasks.

Initially the images were converted from RGB to Grayscale. Then SIFT features were identified and the descriptors were extracted. These point feature descriptors were then recalculated as RootSIFT descriptors based on knowledge obtained from [24]. Then descriptors were aggregated into a list containing the corpus' image descriptors so that centroids can be identified using the k-means algorithm. Once centroids have been identified, the centroid vectors formed codewords, otherwise known as a visual vocabulary or codebook. The codebook is stored for future use by serializing the centroid data. This serialization allows the codebook to be recalled and used later to codify images in terms of their visual words in relation to the codebook.

Next each image in the corpus underwent vector quantization and histogram generation. This means BoVW signatures were calculated for each image in the dataset. This process begins with the quantization of feature descriptors using the codebook produced previously. The result of this quantization process is a list of visual words, each representing a centroid from the visual vocabulary, or codebook, which summarized the content of each image. The number of occurrences of each of the k visual words found in an image are tallied. This tally is then used to build an image histogram. An example histogram for a k=100 codebook is depicted in Figure 25.

Once histograms are calculated they are stored in the Cassandra database and are accessed as needed to facilitate search and comparison operations. When MSACS conducts a query of an image a histogram is computed for the query. Then the cosine distance is calculated against the corpus, and the corpus images are ranked by the cosine distance against the query.

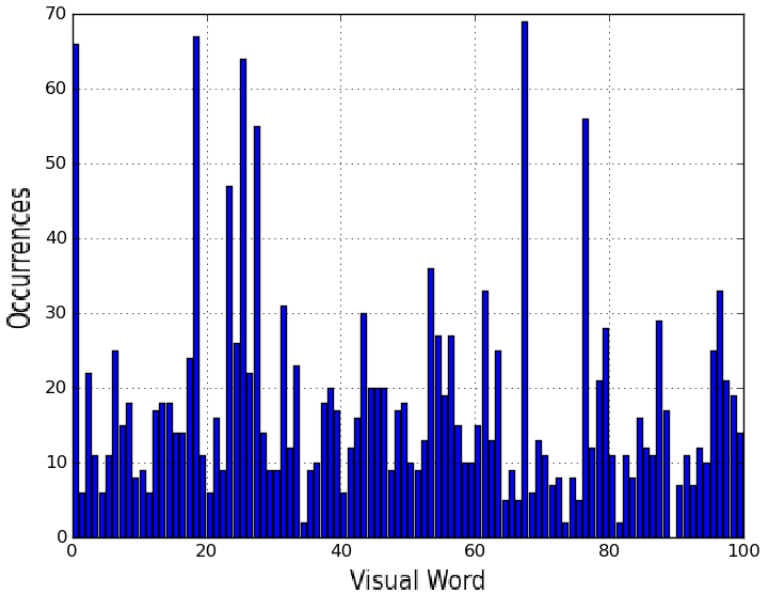

Figure 25. Codebook Histogram

During the course of prototyping the CBIR system it was discovered that the same image features from a dataset corpus may be detected and quantified in subsequent codebook calculations. However, the resulting cluster centers will never be the same mathematically. This

was first tested by creating two codebooks of size k=10. After manual inspection of both codebooks there was a discernable one-to-one match between cluster centers. However, the representation of corresponding cluster centers had a minimum variation of -.0106 and a maximum variation of .01816. Variation was calculated as the percent change from the first codebook to the second codebook. Figure 26 shows the variations of a word, or cluster, between two codebooks. Thus each matching word from both codebooks was compared and an element-wise operation using Python calculated the min and max variation obtained across all k=10 clusters and their 128-element vectors.

These intra-cluster variations could eventually affect the accuracy of the CBIR system results at execution time. After the k=10 codebook comparison two new codebooks of size k=20 were calculated. After manual inspection there were 13 cluster matches between the first and second codebook. The remaining 7 clusters did not appear in both the first and second codebooks. So these 7 clusters were assumed to be different words. This is evidence that the selection of SIFT descriptors is not deterministic either. The intra-cluster variation observed from the k=10 codebooks was present in both k=20 codebooks too. The variation for the case when k=20 was higher than that obtained when k=10. The minimum variation was -.199 and the maximum variation was .136. Again these intra-cluster variations could eventually affect the system results during operation.

```
[ 0.87652808  1.01082134  1.04864168  1.2195152   1.36332726  1.61220145
  1.13688493  0.90809005  1.85252333  1.7200228   1.10533059  1.0289669
  1.13558006  1.23971784  0.89644367  1.15350425  2.50727797  2.11919451
  0.67756379  0.47739443  0.63894159  0.70231307  0.72183168  1.55608881
  1.63248944  1.08907449  0.62312371  0.83828115  1.18072832  1.18424332
  0.88325     1.20215571  0.93389767  0.8753711   0.72043407  0.88675386
  1.32753277  2.17414474  2.17580032  1.4020251   2.4980731   1.77485847
  0.69870865  0.75773996  1.36562717  2.37560129  2.17237234  1.60533667
  3.49382782  2.33530188  0.54905659  0.47849381  0.67637014  0.75854057
  0.94655007  2.41548491  2.01042533  1.2991817   0.74157625  0.9586463
  1.31241095  1.17375672  0.81867158  1.21645343  1.03134525  0.99895322
  0.79394192  0.6686126   0.94152975  1.97400343  2.38121986  1.84531832
  1.54424918  0.72480482  0.56564981  0.557073    1.04006743  2.41000247
  3.11557603  2.90531659  3.23262501  1.48817635  0.55204296  0.47003761
  0.64529783  1.35061538  2.67558312  3.33678007  1.7128942   1.1879797
  0.86890244  1.03482127  1.21167791  1.15127563  0.95380419  1.3689487
  1.10398507  1.28381622  1.17102337  0.83109349  0.69656008  1.22577012
  1.64748669  1.51548553  0.94290841  1.14284742  1.16594589  0.91464806
  0.76912248  1.4903779   2.06429267  1.61886477  1.16817689  1.13097322
  1.00945759  0.93070662  0.8346296   1.21758115  1.77061713  1.80722368
  1.03612125  1.00852334  0.93137711  1.0372107   1.03440607  1.14534104
  0.99021757  1.0382272 ]

[ 0.88849384  1.03475034  1.09030807  1.23235226  1.29036033  1.53288496
  1.15436888  0.9535234   1.81289935  1.71336293  1.14300466  1.02982461
  1.06107748  1.19043195  0.94470924  1.21131372  2.45549774  2.08157229
  0.70326811  0.48620826  0.62463051  0.72522217  0.81247598  1.63708758
  1.61371779  1.07871008  0.63356555  0.84431821  1.16607344  1.20698071
  0.96875453  1.2830075   0.95488554  0.89533055  0.75121665  0.89677572
  1.25846863  2.08471298  2.1990087   1.48250651  2.47273493  1.81656468
  0.75105125  0.76936072  1.2918092   2.29592729  2.20726347  1.6416012
  3.47658634  2.35615611  0.58128667  0.4871054   0.65081197  0.74500835
  0.97937822  2.47525954  1.99504495  1.29242992  0.75505036  0.98598236
  1.31097317  1.19314587  0.87766153  1.27324152  1.08008909  1.01275647
  0.80149537  0.66720569  0.89846307  1.89920628  2.3908298   1.94623971
  1.54323363  0.73726517  0.58722758  0.56871355  0.99333441  2.34403443
  3.12857676  2.96033621  3.21855998  1.50874138  0.57639664  0.50060576
  0.64389336  1.31337142  2.66374636  3.34218526  1.70908833  1.20216334
  0.89931166  1.088642    1.22024083  1.13845623  0.94084972  1.3667531
  1.16266632  1.31280684  1.19123352  0.84005529  0.67943674  1.17876732
  1.63343549  1.580374    0.97796947  1.1783433   1.2134676   0.95564216
  0.76533824  1.43056405  2.02676749  1.65270531  1.17186069  1.14925647
  1.05791318  1.00391197  0.86079043  1.18592513  1.71703863  1.79602611
  1.04379368  1.02185094  0.96570671  1.10127115  1.05061781  1.11995387
  0.95131785  1.03419495]
```

Figure 26. Word Comparison

5.2. Annotated Image Repository

Corpus creation was automated using Google's Street View Image API. This API downloaded images and their geolocation (latitude and longitude) data. A dataset consisting of 1,080 images was collected from New York's Upper East Side in an area of approximately 50 square city blocks. This area and a route through the area is shown in Figure 27.

The corpus was generated by repeated route planning in the Google Maps API. Source and destination points were entered to obtain a URL by the API. This URL was fed into an online GPS visualizer to obtain an XML-based .gpx file. The .gpx file contained the source and destination points along the user-defined route. There were several intermediate GPS points represented by latitude and longitude values between the start and end points. For each latitude and longitude pair in the .gpx file four images were collected from the street-view API at random viewing angles; one from 0-90°, one from 90-180°, one from 180-270°, and one from 270-360°. The images were 500x500 pixels in resolution and suffixed with a .jpg extension. Then the query set was created by taking one random image from 0-360° at each latitude longitude pair in the .gpx file. The depiction of the route specified is shown in Fig. 5. A Cassandra database was then used to store information about each image. This information included the image name, its latitude, longitude, and field of view (i.e. the degree orientation) from which the image was generated.

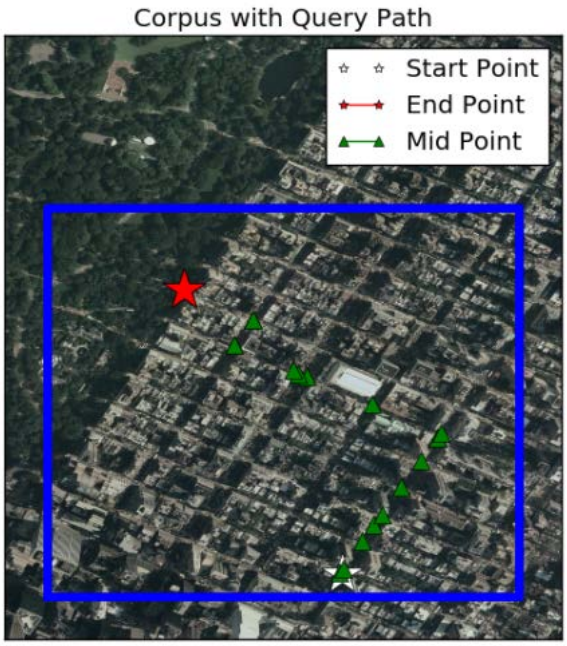

Figure 27. Corpus Area

Ground truth was computed by comparing every query image against the corpus to determine if there was a sufficient number of validated feature matches. This matching established the existence of relevant matching visual content between the image pairs. For an image to be considered relevant it must have a minimum user-defined number of matches. Applying a constraint of at least 50 matches across a set of query images and dataset images was demonstrated to establish relevance with no false positives for a manual evaluation performed using a subset of the overall data. A ground truth table was constructed of size n-by-m where n is the number of query images and m is the number of corpus images. For example if the nth query image and mth corpus image are equal to or greater than the user-defined feature matching limit then position (n,m)=1 in the ground truth table. Otherwise if the match constraint is not met then position (n,m)=0.

5.3. Image Retrieval Evaluation

The MSACS CBIR prototype was evaluated using precision and recall, the standardized way to evaluate a CBIR system [18]. Precision and recall are calculated as follows:

In the experimental evaluation the relevant results were calculated from the ground truth table as images that shared at least 50 features with the query image. To be useful for search, most relevant items should appear within a tractable amount of retrieved results. This tractability ensures that a system can accomplish its task in a timely manner. As seen in Figure 28, 75% of relevant images are returned with 10% of the total corpus size. Tests were conducted with multiple codebook sizes of k as 70, 170, 500, and 1000, and it appeared that k=170 was the best at ordering the first half of returned images. The results seem to demonstrate that the system overly generalizes at k=70, enabling features which are not instances of one another to be codified into the same

visual word. This means the k=70 codebook was too coarse-grained for adequate search and retrieve operations. For k=1000, the system seems to perform best when the return set is in the first 10% of the image repository size. This can more clearly be seen on a logarithmic scaled version of precision and recall, shown in Figure 29.

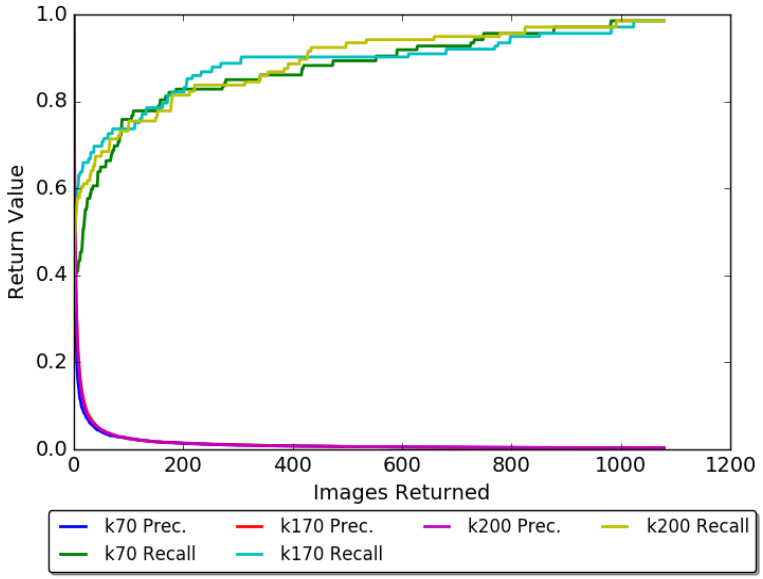

Figure 28. Precision and Recall Curves

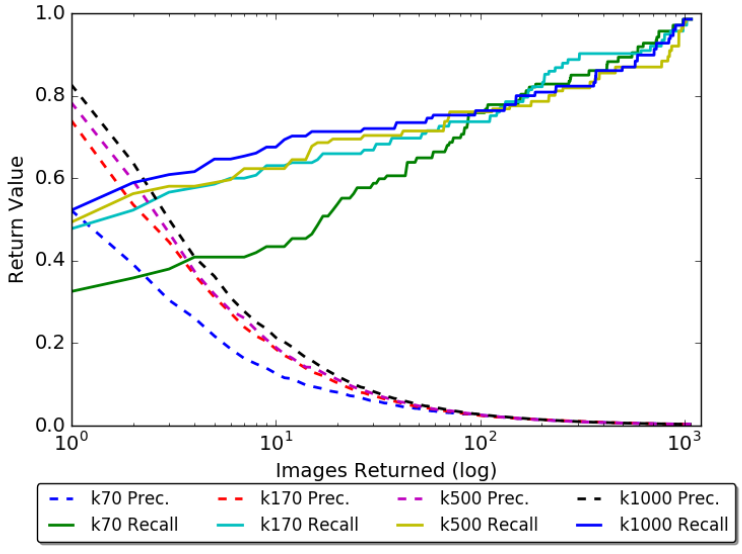

Figure 29. Precision and Recall Curves, Logarithmic

5.4. Application

To enable localization functionality, MSACS can capture images and populate a database with their latitude and longitude coordinates as in use case 1 depicted in Figure 7. The database could potentially be fed by many cameras and other sensors. There could also exist many databases as well due to Cassandra's ability to scale. Searching the database with a query image will lead the system to begin the localization application routine as shown in use case 2.

A simple approach to show this capability is simply to estimate the locations of the top ranked images for a latitude and longitude estimate for the query image. For each of the top returned corpus images the cosine distances of the BoVW histogram with respect to the query image's BoVW histogram are calculated. If the cosine distance between the query and any dataset image was greater than .99, then the location of the query image is assumed to be that of the query image. If the cosine distance of an image is less than .99 then the corpus image's latitude and longitude are summed and averaged with every corpus image's latitude and longitude values from the return set. The user was then presented with an actual location based on the existing knowledge of latitude and longitude values, a guess location based on the summing and averaging operation, and physical distance in meters between the actual and guess locations. The distance between the two was counted as distance error. The results of the localization routine through this coarse averaging among the returned set is shown in Figure 30 for multiple codebooks. The graph further reinforces that the discriminating power of the larger k values produced a more accurate location guess.

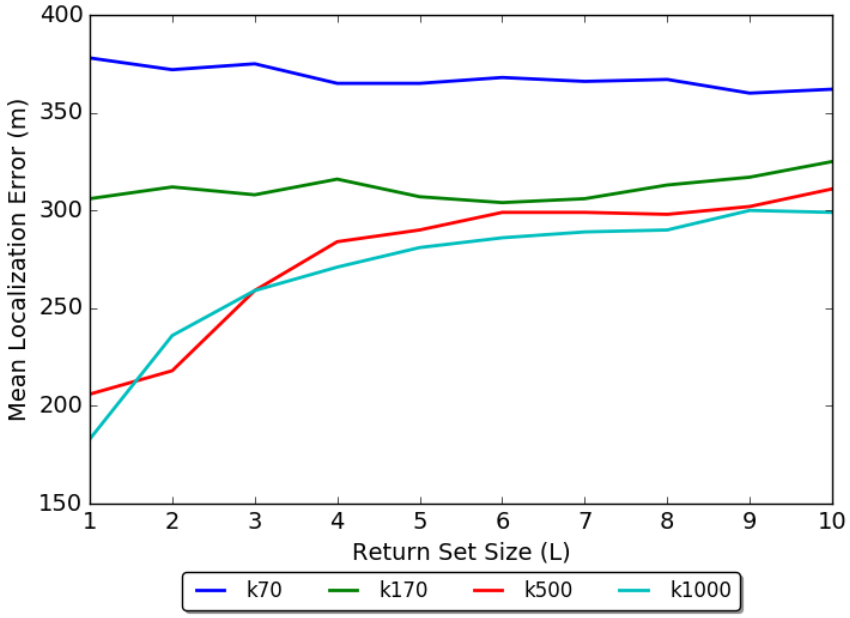

Figure 30. Codebook Size and Localization Error

These localization values were averaged guesses over a path. In an actual localization application, more sophisticated techniques to recognize landmarks and surveyed positions within images would increase the certainty of a guess. Investigation improve localization performance is reserved for future work.

In a final demonstration of the framework the ability to capture easily obtainable information about Wi-Fi access points from the images in the sensed environment is demonstrated. Location guesses were automated to search in the Wigle database [34] from a python script. This automated search displayed the results of anonymized Wi-Fi access points for a particular image search. These reuslts are shown in Table 12.

Table 12. Wi-Fi Access Point Lookup

Wi-Fi Access Point	Latitude	Longitude	Chanel
00:09:5B:XX:XX:XX	40.765308	-73.967735	1
02:22:C2:XX:XX:XX	40.765308	-73.967735	10
20:AA:4B:XX:XX:XX	40.765308	-73.967735	11
20:C9:D0:XX:XX:XX	40.765308	-73.967735	6
58:6D:8F:XX:XX:XX	40.765308	-73.967735	11
68:94:23:XX:XX:XX	40.765308	-73.967735	11
90:84:0D:XX:XX:XX	40.765308	-73.967735	3
B8:C7:5D:XX:XX:XX	40.765308	-73.967735	5

The returned results demonstrated the potential to learn about the Wi-Fi profile of an environment through an image lookup. This also hints at the inverse operation of this lookup in which a list of Wi-Fi access points could return a set of images.

5.5. Performance

The CBIR implementation required a minimum of .286 seconds and a maximum of .464 seconds per query image to perform feature extraction and then construct a histogram based on a k=70 codebook. The operation of searching with a query image required a minimum of .037 seconds and a maximum of .155 seconds. In order to cut down on the search time the entire database is loaded at the begging of program execution into a multidimensional array. This eliminates the need for every query operation to require database access. The database contained three tables which were used to store data.

Table 13. Database Information

Table	Entries	Columns
Locations	1080	4
Queries	21	4
Features	1080	2

The locations table stored four pieces of information for each image in the corpus set: filename, field of view, latitude, and longitude. The queries table stored the same four pieces of information about each query image that was created. The features table stored the name of the image and the number of extracted features present in each image. The number of images in the corpus depend on the length of the routes from the .gpx files. The routes used in the prototype ended up having 1080 images in the corpus. Because the route through a corpus is much smaller than the corpus itself the query route only had 21 images.

5.6. Functional Analysis

As the prototype was a baseline implementation of a CBIR system it did not fully implement the requirements from Section IV. A summary of the implemented requirements is below:

Table 14. Prototype Requirement Coverage

Requirement	Implemented
MSACS-F-##	
00	Yes
01	No
02	No
03	No
04	No
MSACS-AC-##	
00	No
01	No
02	No
03	No
MSACS-M-##	
00	No
01	No
02	No
MSACS-A-##	
00	Yes
01	Yes
MSACS-C-##	
00	Yes
01	Yes
02	Yes
03	Yes
04	Yes
05	Yes
06	Yes
MSACS-AN-##	
00	No
01	No
02	No

5.7. Conclusion

The prototype CBIR system was relatively successful for image retrieval operations. The codebook size affected the results obtained. The larger codebook size seemed to provide better discrimination in image matching operations. These CBIR operations fed into the localization application to provide results based on the query images. Again the larger codebook size appeared to minimize the error obtained with the localization application. An image corpus was

created by a user and then a BoVW codebook was generated using extracted SIFT features and k-means clustering. The user then planned and created a path through the corpus and used the resulting images for prototype evaluation. The prototype used a subset of requirements from those presented in the SDD.

VI. Conclusion and Future Works

Vision plays an important role in the human decision making process, but modern technology has caused information overload. Making a quick decision can be the difference between mission success and failure in a time sensitive and mission critical situations. The MSACS framework, supported by an assisted CBIR system, provides a way to search images quickly and bridge the sensor gap.

6.1. Discussion of Results

The proposed MSACS framework is compared against similar existing multimodal CBIR systems currently in use and the results obtained from the prototype are extended with a look at future work. Each contribution is evaluated and discussed based on the findings of this thesis.

6.1.1. SDD

The SDD was designed to act as an extension to the CBIR prototype presented in this thesis. The CBIR prototype performs the basic functionality of the desired framework. The SDD detailed how this prototype can be extended to create an extensible, modular approach to CBIR and incorporate inputs from many sensors to conduct search and retrieval operations. The SDD used IEEE 1016 as the guiding document and also implemented information from ISO IEEE 29148 for creating requirements related to the framework.

6.1.2. CBIR Prototype

The CBIR prototype presented an approach to image search and retrieval using the BoVW method. This prototype was limited in the sense that it did not implement the assisted

CBIR functionality discussed in the SDD. The prototype however, was able to perform search and retrieval operations. The results of these operations were then fed into the localization application that was designed for the prototype.

6.1.3. Localization Application

The localization application used the results of the CBIR prototype to calculate and return a location to the user in latitude and longitude coordinates. This application was limited in the sense that it performed naïve calculations with regards to localization by averaging coordinates of returned images to calculate current position. This application does prove that the results of the CBIR operation can be fed into an independent application and implemented for some user-defined function.

6.2. Analysis of Alternatives

Several CBIR systems and frameworks exist in the nebula of literature related to image search and retrieval. This current prototype only uses SIFT features as a means of search and retrieval. The trends in the literature for CBIR are tending towards multimodal search and retrieve operations.

The results of the basic CBIR implementation drove the localization application and proved useful for future avenues of research. The precision and recall of the CBIR system did not perform as well comparable systems in the literature. The performance of the CBIR system was not the focus of this thesis. The overall system prototype performance is expected to improve with the addition of the assisted CBIR. The addition of the extensible sensor modules could also give the system greater utility.

Table 15. Analysis of Alternatives

CBIR System	Extensible	Modes
ElAlami [27]	No	Image Classification 1^{st}, 2^{nd}, 3^{rd}, 4^{th} shape moments
Al Kabary [28]	No	Query-by-example Query-by-sketch Keyword Search
Kang [29]	No	Color Texture Shape
MSACS	Eventually	SIFT Features Future user modules

6.3. Future Work

Future iterations of this work will aim to implement the core of the framework by focusing on the modularity and extensibility. Increasing the number of search terms per image in the form of sensors will allow fine-grained search and retrieval. This future work is broken down into three different research thrust areas.

The MSACS framework is intended to be the main thrust of this research. The extensible, modular nature of the framework is expected to be the most difficult aspect in terms of coding and implementation.

The prototype will need to implement additional modes of sensed information while also storing and performing search and retrieve operations. Adding more than a single node with sensors would be the next step in prototype implementation.

The last thrust would be additional application development. The existing localization application would benefit from greater accuracy. Additional application development is also possible to expand the uses of the framework.

6.4. Conclusion

The framework may potentially benefit DoD operations which rely on many sources of data to make a tactical decision. For a UAS mission this means fusing together raw environmental data to influence ISR operations and react to new stimuli while a command and control cell supervises. This framework could decrease time for target acquisition, identification, and order issuance.

Our contribution of the modular framework provided the user with an extensible way to support image retrieval in a dynamic and sensor rich environment. A framework was designed and implemented for assisted CBIR system, incorporating state of the art core functionality using the Bag of Visual Words model. The evaluation of the prototype showed potential for image retrieval and cross sensor applications. Future studies will expand on the development of user interfaces and mechanisms for assisted CBIR across a wide array of potential applications.

In future work the aim will be to improve search results and expand image repositories. Expanding MSACS with mechanisms to improve search through the multi sensor inputs on a Raspberry Pi prototype would be the next logical step. Panning live test experiments using real world imagery, magnetometers, RF receivers and other sensors to perform similar experiments, iteratively improving the core functionality of MSACS for assisted CBIR is another research avenue.

6.4.1. Research Questions

The prototype and the SDD provided a means to answer the research questions posed in Section 1.3. At a minimum an implementation of a CBIR system requires the functionality to query, compare, and report against images which already exist in an archive. Extending this

traditional CBIR implementation to an assisted CBIR paradigm would require at least one additional source of data to be used during the query operation. This source of data can be extracted from the content of an image itself or it can be collected from environment in which the image was captured. Collecting information from an environment requires a sensor to read and interpret the raw data. This sensor data is then associated with the original image capture and used as an additional method of discrimination in search and retrieve operations.

The first question from Section 1.3 can be satisfied by expanding on the minimum CBIR implementation. In order to create an extensible software framework implementing the assisted CBIR functionality described above a modular approach needs to be taken. The modularity of the sensor modules in a framework will enable the desired extensibility. Thus the requirements needed to create an extensible software framework are:

1. Accepting modules which implement a sensor's data and the metadata of the sensor
2. An assisted CBIR system capable of performing search and retrieve operations with more than one type of data

The second question relating to the applicability of assisted CBIR can be answered by inspecting the preliminaries and background information presented in Section II and the prototype presented in Section V. As shown in [31-32] there are benefits to using assisted CBIR in the medical field to speed up and improve the accuracy of medical diagnoses. In [33] the authors use assisted CBIR for real-world disaster response situations. In [36] the authors discuss using assisted CBIR for law enforcement to compare several types of data simultaneously to aid in solving cases. Additionally the prototype presented in Section V implements image-based CBIR can be extended to include Wi-Fi data in search and retrieve operations. These wide arrays of applications all rely on data in order to function properly. The big data needs of hospital and law enforcement agencies

can be driven by the assisted CBIR system. Hospitals can inspect records from each other if they are in the same network. Law enforcement agencies use the same systems for record keeping and maintaining. The implementation of assisted CBIR will further drive big data requirements.

The third question can be answered by looking at the prototype from Section V. The CBIR system supports a localization application that estimates latitude and longitude values based on the content of an image. Using a CBIR system, and by extension an assisted CBIR system, this information can be fed into large information repositories such as Wigle to pass latitude and longitude coordinates or information related to a Wi-Fi access point such as SSID to further refine positional accuracy.

6.4.2. Contributions

The contributions of this thesis were threefold: create an SDD which outlines an extensible software framework for assisted CBIR, prototype an assisted CBIR system, and a localization application driven by the assisted CBIR results.

The SDD builds upon an existing CBIR system and provides requirements, descriptions, and diagrams with the intent of guiding future development. This SDD is intended for future thesis work and will guide development going forward.

The prototype of the assisted CBIR system uses extracted image features in conjunction with latitude and longitude information to perform search and retrieval operations.

The localization application uses the extracted image features along with the latitude and longitude information of the images in the return set drive a localization application. The localization routine uses the results of the return set to estimate a query images location within a corpus. When this localization application completes execution the estimates latitude and

longitude results are return to the user. The prototype evaluation revealed that this application can be accurate but that is not always the case.

6.5. Findings

The assisted CBIR implementations from [27-29] provide improved performance by using multimodal data for search and retrieve operations. The prototype presented benefits from the multimodal information fusion. This improved performance is expected to increase even more with the additional sensors. The additional sensors should improve system performance as seen from existing assisted CBIR implementations.

6.6. Achievements

This work formed the core of a faculty research council grant at AFIT. All work towards this thesis will be forwarded to the council for their future use. In addition, a portion of this work was accepted at the International Conference on Cognitive and Computational Aspects of Situational Awareness and Decision Support [38], to be presented in late March 2017.

Appendix A: Environment Setup

The prototyping and experiment conducted in Section V used a virtual machine with the Ubuntu version 16.04 64-bit operating system. The environment was configured to contain 4 cores each with 1 processor, 8 GB of RAM and 50GB of hard drive space. This appendix lists instructions to setup and configure the prototyping environment once the operating system has been installed. First, to install OpenCV the following commands need to be issued from the terminal:

1. sudo apt-get update
2. sudo apt-get upgrade
3. sudo apt-get install build-essential
4. sudo apt-get install python-tk
5. sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
6. sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
7. sudo apt-get install qt5-default
8. cd ~/path/to/install/directory
9. git clone https://github.com/opencv/opencv.git
10. git clone https://github.com/opencv/opencv_contrib.git
11. cd opencv
12. mkdir build
13. cd build

14. cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=OFF -D WITH_QT=ON -D WITH_OPENGL=ON -D ENABLE_FAST_MATH=1 ..
15. sudo make install

Now that OpenCV is installed there are module files required to implement the functions to enable CBIR. These module files can be installed Python's pip function. To install the modules the following commands need to be issued:

1. From the command line to install pip; sudo apt-get install python-pip
2. pip install --upgrade pip
3. sudo pip install cassandra-driver
4. sudo pip install scipy
5. sudo pip install geopy
6. Install Basemap from http://matplotlib.org/basemap/users/installing.html (for map visualization, if desired)
7. sudo pip install xmljson (module converts xml to json using multiple formats)
8. sudo pip install matplotlib
9. sudo pip install sklearn

Once the modules have installed the Cassandra database software can be downloaded by issuing by doing the following:

1. http://cassandra.apache.org/download/ and download 3.7. Or go to archive.apache.org/dist/Cassandra/3.7/ and download apache-cassandra-3.7-bin.tar.gz
2. Extract to desired directory
3. Java 8 is needed for Cassandra operation; sudo apt-get install default-jdk
4. To begin Cassandra cd to the Cassandra/bin directory and execute ./cassandra

Next, in order to generate .gpx files a route will need to be planned on Google Maps. The .gpx files are XML-like data representations of a planned route from Google Maps. To use these the following should be done:

1. Plan a route on google maps with a starting and destination point
2. Copy the URL of the route and go to the GPX website http://www.gpsvisualizer.com/convert_input
3. Choose GPX as the output format
4. Everything else can be ignored, just paste in the URL and click CONVERT
5. Download the GPX file and use as desired

Bibliography

[1] Peck, Michael. "How full-motion video is changing ISR." C4ISRNET, 23 Mar. 2016. Web. Accessed on 23 Jan. 2017. <http://www.c4isrnet.com/story/military-tech/isr/2016/03/23/how-full-motion-video-changing-isr/82124848/>.

[2] Peck, Michael. "Nga-open-source." C4ISRNET. N.p., 26 Apr. 2016. Web. Accessed on 23 Jan. 2017. <http://www.c4isrnet.com/story/military-tech/geoint/2016/04/26/nga-open-source/83546086/>.

[3] Raghuvanshi, Vivek, and Aitoro, Jill. "Air Force Seeking Out 'Ender's Game' Technology to Enable Drone Swarms." Defense News, 1 Nov. 2016. Web. Accessed on 23 Jan. 2017. <http://www.defensenews.com/articles/air-force-diux-seeking-out-enders-game-technology-to-enable-drone-swarms>.

[4] Aitoro, Jill, Kington, Tom and Tran, Pierre. "How Swarming Drones Could Change the Face of Air Warfare." Defense News, 17 May 2016. Web. Accessed on 23 Jan. 2017. <http://www.defensenews.com/story/defense/air-space/air-force/2016/05/17/drone-air-force-swarm-mini-uas/84496780/>.

[5] "Search by text, voice, or image." Inside Search, 14 June 2011. Web. Accessed on 22 Jan. 2017. <https://search.googleblog.com/2011/06/search-by-text-voice-or-image.html>.

[6] Singh, Jagpal, Jashanbir Singh Kaleka, and Reecha Sharma. "Different approaches of CBIR techniques." *Int. J. Comput. Distributed Syst* 1 (2012): 76-78.

[7] Mansourvar, Marjan, et al. "A computer-based system to support intelligent forensic study." *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation*. IEEE, 2012.

[8] Pinjarkar, Latika, Manisha Sharma, and Kamal Mehta. "Comparative evaluation of image retrieval algorithms using relevance feedback and its applications." *International Journal of Computer Applications. India* (2012).

[9] Panteras, George, et al. "Accuracy Of User-Contributed Image Tagging In Flickr: A Natural Disaster Case Study." *Proceedings of the 7th 2016 International Conference on Social Media & Society*. ACM, 2016.

[10] Banda, Juan M., et al. "Big data new frontiers: Mining, search and management of massive repositories of solar image data and solar events." *New Trends in Databases and Information Systems*. Springer International Publishing, 2014. 151-158.

[11] IEEE Standard for Information Technology, Systems Design, Software Design Protocols, IEEE Standard 1016, 2009

[12] Yang, Jun, et al. "Evaluating bag-of-visual-words representations in scene classification." *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM, 2007.

[13] Höschl IV, Cyril, and Jan Flusser. "Robust histogram-based image retrieval." *Pattern Recognition Letters* 69 (2016): 72-81.

[14] Pass, Greg, and Zabih, Ramin. "Histogram refinement for content-based image retrieval." Applications of Computer Vision, 1996. WACV'96, *Proceedings 3rd IEEE Workshop on. IEEE*, 1996.

[15] Kumar, Kamlesh, Jian-Ping Li, and Riaz Ahmed Shaikh. "Content Based Image Retrieval Using Gray Scale Weighted Average Method." *International Journal of Advanced Computer Science & Applications* 1.7: 1-6.

[16] Solomon, Chris, and Breckon, Toby. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. John Wiley & Sons, 2011.

[17] Choras, Ryszard S. "Image feature extraction techniques and their applications for CBIR and biometrics systems." *International Journal of Biology and Biomedical Engineering* 1.1 (2007): 6-16.

[18] Datta, Ritendra, et al. "Image retrieval: Ideas, influences, and trends of the new age." *ACM Computing Surveys (CSUR)* 40.2 (2008): 5.

[19] Qian, Gang, et al. "Similarity between Euclidean and cosine angle distance for nearest neighbor queries." *Proceedings of the 2004 ACM Symposium on Applied Computing*. ACM, 2004.

[20] Rosebrock, Adrian. "The bag of (visual) words model." PyImageSearch Gurus, 2015. Web. Accessed on 22 Jan. 2017. <https://gurus.pyimagesearch.com/the-bag-of-visual-words-model/>.

[21] Jain, Anil K. "Data clustering: 50 years beyond K-means." *Pattern Recognition Letters* 31.8 (2010): 651-666.

[22] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International Journal of Computer Vision* 60.2 (2004): 91-110.

[23] Rosebrock, Adrian. "Implementing RootSIFT in Python and OpenCV." PyImageSearch, 13 Apr. 2015. Web. Accessed on 22 Jan. 2017. <http://www.pyimagesearch.com/2015/04/13/implementing-rootsift-in-python-and-opencv/>.

[24] Arandjelović, Relja, and Andrew Zisserman. "Three things everyone should know to improve object retrieval." *Computer Vision and Pattern Recognition (CVPR)*, 2012 IEEE Conference on. IEEE, 2012.

[25] Sigurbjörnsson, Börkur, and Van Zwol, Roelof. "Flickr tag recommendation based on collective knowledge." *Proceedings of the 17th International Conference on World Wide Web*. ACM, 2008.

[26] Hare, Jonathon S., and Paul H. Lewis. "Automatically annotating the mir flickr dataset: Experimental protocols, openly available data and semantic spaces." *Proceedings of the International Conference on Multimedia Information Retrieval*. ACM, 2010.

[27] ElAlami, M. Esmel. "Unsupervised image retrieval framework based on rule base system." *Expert Systems with Applications* 38.4 (2011): 3539-3549.

[28] Al Kabary, Ihab, et al. "QUEST: Towards a Multi-Modal CBIR Framework Combining Query-by-Example, Query-by-Sketch, and Text Search." *Multimedia (ISM), 2013 IEEE International Symposium on*. IEEE, 2013.

[29] Kang, Jiayin, and Wenjuan Zhang. "A framework for image retrieval with hybrid features." *2012 24th Chinese Control and Decision Conference (CCDC)*. IEEE, 2012.

[30] Wang, Xiang-Yang, Yong-Jian Yu, and Hong-Ying Yang. "An effective image retrieval scheme using color, texture and shape features." *Computer Standards & Interfaces* 33.1 (2011): 59-68.

[31] Névéol, Aurélie, et al. "Natural language processing versus content-based image analysis for medical document retrieval." *Journal of the American Society for Information Science and Technology* 60.1 (2009): 123-134.

[32] Kumar, Ashnil, et al. "Content-based medical image retrieval: a survey of applications to multidimensional and multimodality data." *Journal of Digital Imaging* 26.6 (2013): 1025-1039.

[33] Sergieh, Hatem Mousselly, et al. "Geo-based automatic image annotation." *Proceedings of the 2nd ACM Int'l Conference on Multimedia Retrieval*. ACM, 2012.

[34] "WiGLE: Wireless Network Mapping." WiGLE: Wireless Network Mapping. Web. Accessed on 06 Dec. 2016. <https://wigle.net/>.

[35] ISO Systems and Software Engineering, Life Cycle Processes, Requirements Engineering, ISO Standard 29148, 2011

[36] Pinjarkar, Latika, Sharma, Manisha and Mehta, Kamal. "Comparative evaluation of image retrieval algorithms using relevance feedback and its applications." *International Journal of Computer Applications* 48.18 (2012): 12-16.

[37] "How do I export a diagram from Creately?" Creately: Support Community. Web. Accessed on Dec. 2016. <https://support.creately.com/hc/en-us/articles/221410507-How-do-I-export-a-diagram-from-Creately->.

[38] "Sponsors and Support." *2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management*. Web. Accessed on Sept. 2016. <http://cogsima2017.ieee-cogsima.org/>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
23-03-2017	Master's Thesis	October 2015 – March 2017

TITLE AND SUBTITLE

A Software Framework for Image Retrieval and Visual Understanding in Dynamic and Sensor Rich Environments

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Lesch, Noah C., Captain, USAF

5d. PROJECT NUMBER
16G130

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/ENY)
2950 Hobson Way, Building 640
WPAFB OH 45433-8865

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-17-M-045

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL Information Directorate
ATTN: Maj Hiren Patel
AFRL/RIGD, Rome, NY 13441
315-330-4315 hiren.patel@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RIGD

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Performing search and retrieval operations with massive amounts of visual and environmental sensor information is problematic in time sensitive and mission critical situations such as emergency management and disaster response. Distinct sensor readings can be fused together to create a compact multimodal representation of a location. Efficient search and retrieval can be an answer to the problem of scale. Content Based Image Retrieval (CBIR) systems inherently rely on the search and retrieve operations to support timely and accurate responses. However, there is currently no adequate software framework system for multimodal CBIR to support situational awareness in dynamic and sensor rich environments. In this thesis, an extensible framework for CBIR is proposed to support an understanding of a sensor rich environment through the automated search and retrieval of relevant images and the context of their capture This constitutes assisted CBIR as embodied in the proposed framework for multi-sensor assisted CBIR system (MSACS).

15. SUBJECT TERMS
Content-based Image Retrieval, Bag of Visual Words, Software Framework, Sensor Fusion

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Lt Col John Pecarina, AFIT/ENG
U	U	U	UU	92	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636, ext 3368 john.pecarina@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18