Polytechnic University of Puerto Rico
Hato Rey, Puerto Rico
Department of Electrical and Computer Engineering and Computer Sciences

# Software Requirements Specifications
# Smart Eyesight
# Version 1.0

Emanuel Rivera Castro 53502
Joaquin Pockels Balaguer 54012
Yanilette Lopez Duprey 53990

**October 31, 2012**
COE 5002, FA-12
Prof. Luis Ortiz Ortiz

| Date | Version | Description | Author |
|---|---|---|---|
| first draft in progress | | | |

Table 1: Revision Table

# Contents

# List of Tables

# List of Figures

# 1 Introduction

The present document is the Software Requirements Specifications (SRS) for a region-based image retrieval (RBIR), feature extraction and probabilistic neural network (PNN) classifier software application named Smart Eyesight. The application is a salient part of a mayor project proposal directed to the Defense Advanced Research Projects Agency. Smart Eyesight serves as a structural model for future extensions and provide the basic functionality for the required image processing and classification.

## 1.1 Purpose

This Software Requirements Specifications document, establishes the requirements, functionality, capability, behavior and design of the software application. The document serves as a description of what the tool should do. This SRS is to be reviewed by the project's developers, users and client for functionality verification and acceptance purposes.

## 1.2 Intended Audience

This SRS is directed to:

1. Clients - DARPA and Prof. Arturo Geigel
2. Users - United States Armed Forces
3. Developers - AIIP/S and people responsible for maintenance and making updates to the system:
    - Joaquin Pockels
    - Emanuel Rivera
    - Yanilette Lopez

## 1.3 Project Scope

The region-based image retrieval, features extraction and PNN classifier tool named Smart Eyesight is mainly designed to efficiently and effectively extract specific features from a stream of images and classify them through a trained neural network. Smart Eyesight is a salient process of the DARPA's proposed system designed by Arturo Geigel named: "Human Computing and Machine

Learning Assisted Decentralized UAV Control". This major system involves the transmission of a stream of images via unmanned vehicles, a tagged image database, a computing infrastructure to process and identify received images, and human computer interaction between operators and unmanned systems. Such system does not include any form of image processing and classification, but provides the necessary objects to do so. The large system depends on the Smart Eyesight subsystem in order to complete the process of image identification.

The objective of the application are:

- Extracts relevant features from an incoming stream of images by using traditional features extraction algorithms and statistical methods
- Train a probabilistic neural network using the relevant features obtained from UAV incoming images
- Identify provided images based on its neural network training
- Provide feedback to mayor project's interconnected inverse file query system.
- Designed as an extensible software for CUDA code conversion
- Designed as an independent system or module
- Interconnection structure of the Smart Eyesight application with the main proposed system is to be established and validated with the consent of developers of the dependent sub-systems and the client himself

[1][2][3][4][5]

## 1.4 Definitions and Acronyms

This subsection shall define, or provide references to the definition of all terms and acronyms required to properly interpret the SRS.

### 1.4.1 Definitions:

- *Artificial Neural Network:* a neural network composed of interconnecting artificial neurons.
- *Artificial Neurons:* programming constructs that mimic the properties of biological neurons.
- *Feature Vector:* in pattern recognition and machine learning, it is an n-dimensional vector of numerical features that represent some object.

- *Independent module/system:* a fully working system which has no dependencies.
- *Smart Eyesight:* developed software application for image processing and classification.
- *Segmentation:* is the process of partitioning a digital image into multiple segments, with the goal of simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.
- *Extensible Software:* is a software designed with future growth in consideration.
- *YUV:* luminance and chromiance color space used as part of a color image pipeline.

### 1.4.2 Acronyms:

| **Acronym** | **Expansion** |
|---|---|
| API | Application Programming Interface |
| CBIR | Content Based Image Retrieval |
| BWF | Biorthogonal Wavelet Frame |
| DARPA | Defense Advanced Research Projects Agency |
| FRIP | Finding Region In the Pictures |
| GPU | Graphics Processing Unit |
| HDD | Hard Disk Drive |
| MRS | Modified Radius-based Signature |
| SSD | Solid-state Drive |
| PNN | Probabilistic Neural Network |
| RGB | Red Green and Blue color model |
| RBIR | Region Based Image Retrieval |
| SRS | Software Requirements Specifications |
| SPMP | Software Project Management Plan |
| TBD | To Be Described |
| TCP/IP | Transmission Control Protocol and Internet Protocol |
| UAV | Unmanned Aerial Vehicle |

Table 2: Acronyms

## 1.5 References

[1] D. Phillips, *Image Programming in C*. R & D Publications, 2000.

[2] B. Ko and H. Byun, "Probabilistic neural networks supporting multi-class relevance feedback in region-based image retrieval," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 4, 2002, pp. 138 – 141 vol.4.

[3] A. Kam and W. Fitzgerald, "A general method for unsupervised segmentation of images using a multiscale approach," in *Computer Vision ECCV 2000*, ser. Lecture Notes in Computer Science, D. Vernon, Ed. Springer Berlin / Heidelberg, 2000, vol. 1843, pp. 69–84, 10.1007/3-540-45053-X_5. [Online]. Available: http://dx.doi.org/10.1007/3-540-45053-X_5

[4] B. Ko, H.-S. Lee, and H. Byun, "Region-based image retrieval system using efficient feature description," in *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, vol. 4, 2000, pp. 283 –286 vol.4.

[5] B. Ko, J. Peng, and H. Byun, "Region-based image retrieval using probabilistic feature relevance learning," *Pattern Analysis & Applications*, vol. 4, pp. 174–184, 2001, 10.1007/s100440170015. [Online]. Available: http://dx.doi.org/10.1007/s100440170015

## 1.6 Overview

The rest of this document presents the full description of the RBIR and PNN classifier tool. Section 2 is directed to the client. It shows the general description of the application, including constrains, main functions, user characteristics, assumptions and dependencies. Section 3 and onwards are directed to developers. They outline, in detail, system features, specific requirements, performance, behavior, interfaces and others.

# 2 General Description

This section presents the overall description of the application. It explains factors that affect the application as well as its requirements. Starting with the product perspective and ending with assumptions and dependencies.

## 2.1 Product Perspective

Smart Eyesight serves as a multipurpose CUDA extensible software, developed and written in C programming language that runs on a Linux-based operating system. Smart Eyesight is an

integrated sub-system of a DARPA project proposal named: "Human Computing and Machine Learning Assisted Decentralized UAV Control". It also serves as a model structure for future mayor extensions. The system is part of the image processing and identification infrastructure and is resident at a Linux-based cluster as seen in the block diagram of figure 1.

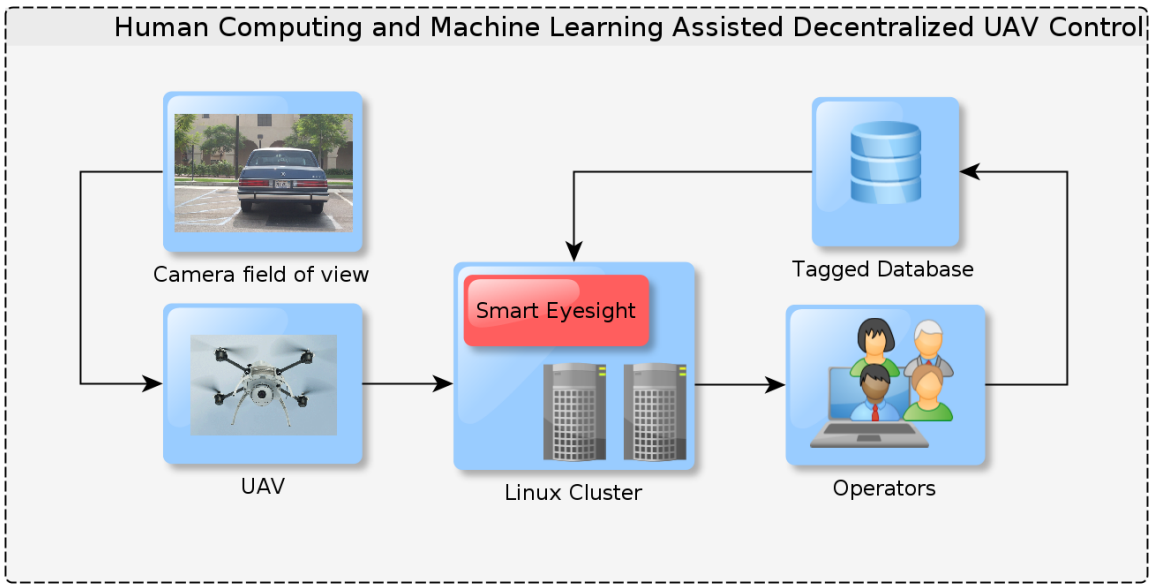


Figure 1: Smart Eyesight overall relation

Essentially, the application's purpose is to identify or classify a stream of images incoming from a tagged database and a UAV video stream, and serve feedback to the dependant sub-systems. This is done via two main processes and one shared process:

***Main processes:***

1. **Neural Network Training:** this process is dedicated to train a probabilistic neural network using extracted features of regions of images from a database of tagged images.

2. **Image Classification:** this process is based on the PNN training, Smart Eyesight will identify each image from the extracted features of the UAV input stream and provide its feedback to the interconnected back-end sub-system.

***Shared process:***

1. **Image Processing** shared process between the Neural Network Training and the Image Classification tasks. This process is in charge of the image to segmentation and feature extraction conversion through a Content Based Image Retrieval process.

## 2.2 Product Functionality

As mentioned, Smart Eyesight has two mayor processes and one process. In this section each are briefly explained in the form of functions. For more details refer to the section 4

**General Process:** First, the probabilistic neural network learns the characteristics of each image stored for training purpose. These images have an identifier in the form of class index. Both image and class gets processed in the image processing function returning a normalized feature vector. This vector is used in the PNN to create each image region pattern weights. Each pattern is related to their respective class. When the external system ask for the classification of UAV images, these images get processed the same way by using the image processing function, thus obtaining a normalized feature vector. Finally this vector is used in the PNN for pattern activation and summation, leading to the selection of the corresponding class index used by the inverse file query external system. The whole structure is visually observable at the communication and data flow diagram of figure 3.

### 2.2.1 Neural Network Training

This function use a Probabilistic Neural Network to "learn" the characteristics and properties (features) of each image by creating and re-weighting pattern nodes per each region. It uses normalized feature vectors produced from extracted features of image's regions and a class per image index. The images and class index are native of the tagged database and their conversion to feature vectors is handled by the shared *Process Image* function call. Each vector pertain to a specific class and creates a set of connected weights between the input layer and a new pattern in the pattern layer also called hidden layer. These weights will latter be responsible for "activating" each pattern neuron in the image classification function, thus obtaining a resultant index. The latter will be used for the inverted file query. For more details on Probabilistic Neural Network architecture and functionality see appendix section 5.4 page 19.

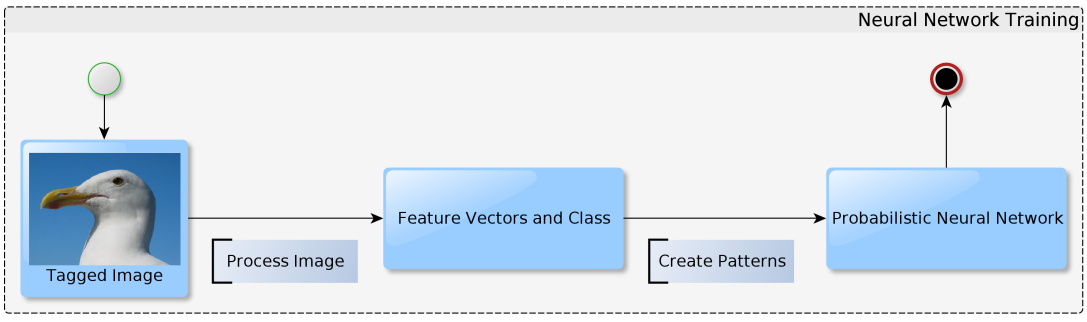


Figure 2: Smart Eyesight's Neural Network Training state diagram

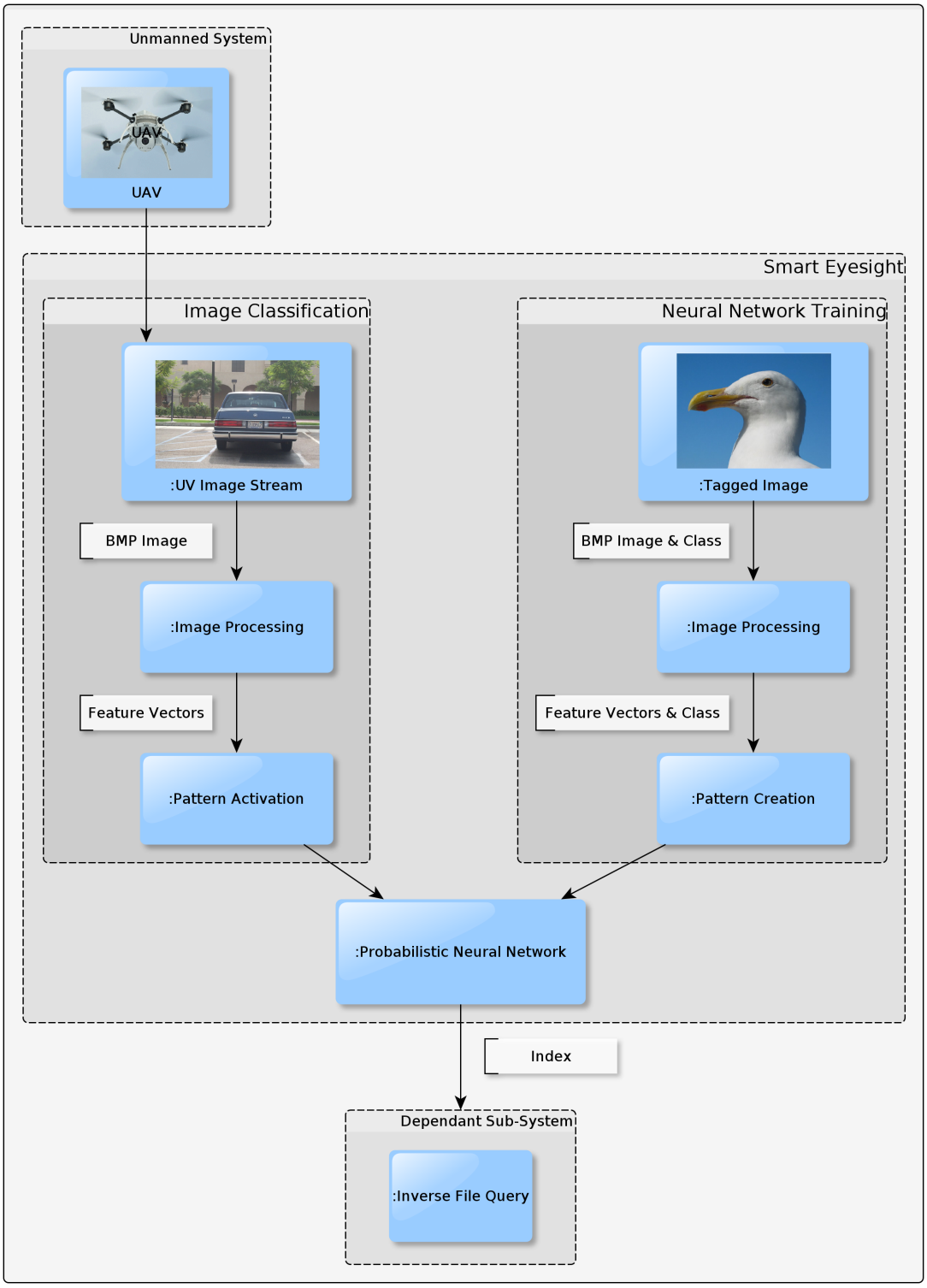


Figure 3: Smart Eyesight communication and data flow diagram

### 2.2.2 Image Classification

This task use a trained Probabilistic Neural Network to "classify" or "identify" the characteristics and properties (features) of each image by creating and re-weighting pattern nodes per each region. It uses normalized feature vectors produced from extracted features of image's regions. The images used for classification come from the UAV system and their conversion to feature vectors is handled by the shared *Process Image* function call. The function uses the feature vector to "activate" its pattern nodes and proceed with a summation of their output. The result with the greatest value will determine the class of the image in question. The function check and compare the current feature vector with every pattern weights and causes activation of patterns. A summation of region patter The result of the function or final PNN layer output is an index that will be used by the inverted file query system of the mayor system. For more details on Probabilistic Neural Network architecture and functionality see appendix section 5.4 page 19.

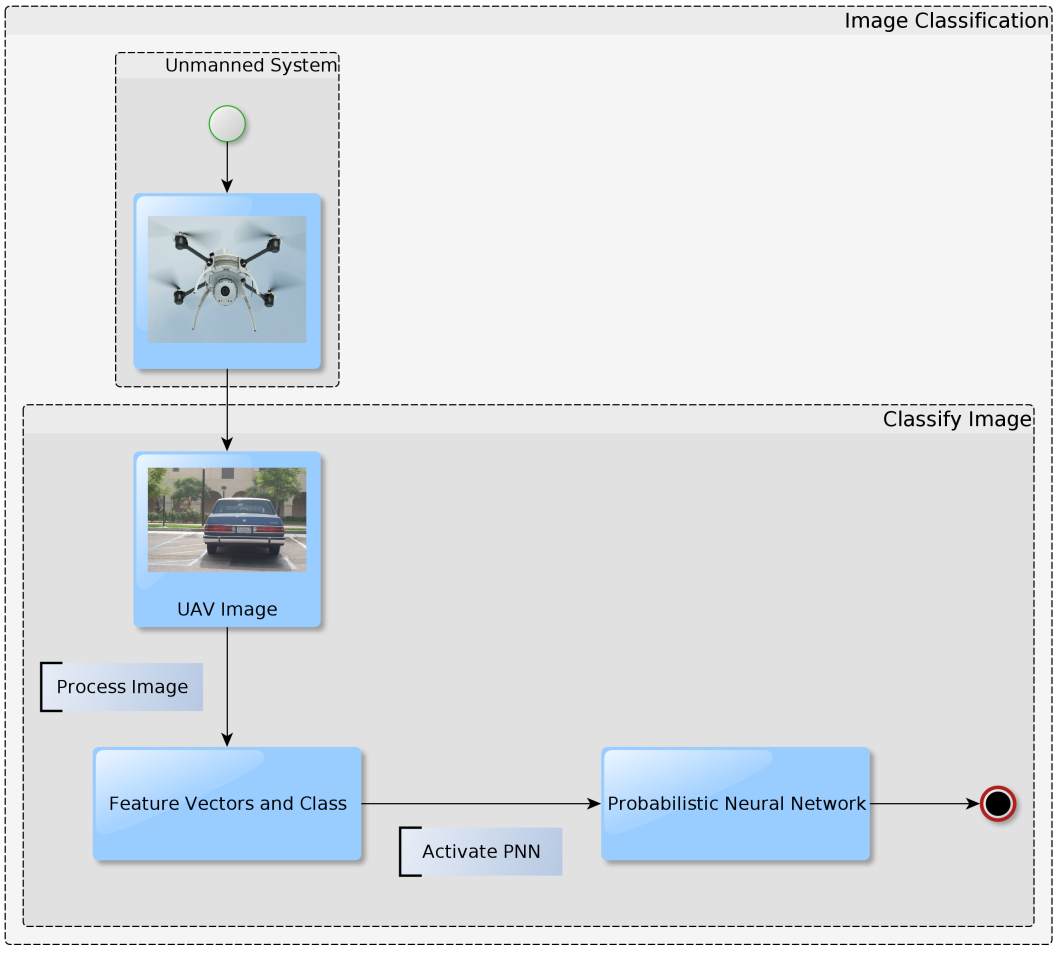


Figure 4: Smart Eyesight's Image Classification state diagram

### 2.2.3 Image Processing

Smart Eyesight's main Image Processing function is to describe the content of the image by segmenting it and then extracting features of each segmented region. That is, the semantics of image should be extracted and stored as index. This task is possible through four sub-functions (more details at section 4.1).

1. **Read Image**
2. **Segment Image**
3. **Extract Features**

Each received image (UAV and database input) is segmented through a retrieval system based on region called Finding Region In the Pictures (FRIP). From each segment, five or more specific features are extracted in order to describe effectively the image. The features to be extracted are *Color, Texture, Scale, Location and Shape*. The extracted features are converted and normalized to a feature vector which is used in the input layer of the PNN of both classification and training phases. The feature vector create and weight vectors corresponding to a class in the training face, or serve as activation data for the testing face. Furthermore, this function entail efficiency because of heavy use by the mentioned two processes and vast amount of sub-functions are required to complete the task.

## 2.3 Operating Environment

The software operates in a linux-based cluster platform with resent Ubuntu/Fedora Operating Systems. The platform is connected with the Unmanned System through wireless access points. The training process requires and additional cluster for continuous and efficient image processing and training. The components of the overall system and subsystem interconnections can be observed at figure 1.

## 2.4 Design and Implementation Constraints

In this section the design and implementation constraints of the Smart Eyesight software application are presented. Specifically and in more detailed the image format, image processing, probabilistic neural network and the interconnected sub-system structure.

### 2.4.1 Image Constraints

Constraints related with the incoming images that are processed with the FRIP method and then used in the PNN for training and classification:

| Object | Constraints |
|---|---|
| Imported Image | BMP format |
| | Header is converted to BMP format as described by the Image Processing in C book if necessary during image processing |
| | Content is converted to the RGB/YUV format if required |
| Processed Images | are stored as volatile data and are "liberated" from dynamic memory after use. |

Table 3: Image Constraints Table

### 2.4.2 Image Processing Constraints

Constraints related with the image processing functionality:

- Finding Regions in Picture is used as a Region Based Image Retrieval method
- A circular filter, must be used to reduce noise and other undesirable factors of the image
- Segmentation must be as efficient as possible
- Extracted features should be described briefly and effectively
- Five specific features are extracted as described in section 2.2.3
- A maximum of a hundred regions are segmented per image

### 2.4.3 Neural Network Constraints

Constraints related with the Neural Network to be used for the training and classification processes:

- A Probabilistic Neural Network structure is used for the training and classification tasks
- For the training function, the PNN use feature vectors of regions of images extracted from the tagged image database

- For the classification function, the PNN use feature vectors of regions of images extracted from the unmanned system's image stream

### 2.4.4 Design Constraints

Constraints related with the overall design of the whole system including Operating System, programming language, API's and other environments:

- Feedback to the interconnected dependant sub-system is in the form of a classification index, which format is in accordance to the established inverse file query structural design implemented by the developers in charge of the task
- Software's code is written in C programming language
- Linux-based Operating Systems are used to develop the application. Linux OS include but are not limited to Ubuntu and Fedora
- Smart Eyesight software is not responsible for the application programming interface present between interconnected sub-system. Only formats will be considered as explained in this section.

### 2.4.5 Used Tools and Technologies

The following table presents specific tools and technologies that are used in the making of Smart Eyesight:

| Tool/Technology | Usage |
|---|---|
| C programming language | C is used as the main programming language. It is also used as an enabler of future extensions and modifications such as CUDA code conversion and parallelization. |
| Linux-based Operating System | Linux is used to avoid violations of copyrighted software's terms of use and extended costs. The system only uses open source shared libraries and software tools. |
| Book: Image Processing in C | Some libraries, designs and image processing code from the book are used during the development. This is an open source text book that contains open source code. |
| Personal Desktop computer and laptops | These computers are used for the development and design and testing of the software application and software engineering documents as explained in the SPMP section x. |
| TCP/IP and FTP | These protocols are used to obtain the UAV incoming images through wireless access points. Tagged incoming images are stored in a non-volatile device such as HDD and SSD. |

Table 4: Used Tools and Technologies

### 2.4.6 Regulatory Policies

- At the moment, the Clients and developers have shared proprietary rights over Smart Eyesight. This information is discussed in-depth in the Smart Eyesight's SPMP document, section x.

## 2.5 Assumptions and Dependencies

Smart Eyesight has one dependency and one assumption which could affect the requirements stated in this document. The application could be affected if these assumptions and dependencies aren't met or incorrect:

- **Dependency:** Smart Eyesight depends on the images incoming from the Unmanned System and tagged database.

- **Assumption:** Images received from ether source are not corrupted. Meaning image headers and containing visual data have a fully readable header format and are considered digital images.

# 3 External Interface Requirements

This section describes the logical characteristics and communication of each interface between the software, hardware and users.

## 3.1 User Interfaces

Not applicable

## 3.2 Hardware Interfaces

Smart Eyesight's original hardware infrastructure is composed of Linux-based head nodes. Through the use of switches, these nodes are connected to access points, relay servers and image processing dedicated clusters which compose to the full system. However, since the application serves as a model of the CUDA extended infrastructure, most of the process is carried in a single linux OS computer and tasks such as the PNN training in a separated cluster for a reasonable task resolution time. Smart Eyesight obtain each UAV incoming image through wireless connections between access points. No further hardware requirements are needed besides the mentioned hardware constraints mentioned in the section 2.4.5.

## 3.3 Software Interfaces

Smart Eyesight runs on recent linux-based operating systems such as Ubuntu 1x.xx and Fedora. Dependencies are obtained through wireless and wired TPC/IP and FTP. The software is extensible through the use of C programming and variants such as CUDA C. The index obtained from the classification phase is send to the dependent subsystem which uses this information for the inverse file query task.

## 3.4 Communications Interfaces

A custom TCP/IP with FTP program is used for obtaining the dependencies. The image processing function uses these dependencies to create and obtain the normalized feature vectors used in the next two phases. The training process receive feature vectors obtained from the tagged image database. The Classification process receive feature vectors obtained from the UAV image stream.

# 4 System Features

This section list the functional requirements for Smart Eyesight by system features.

## 4.1 IP-REQ-0 Process Image

### 4.1.1 Description

This function is responsible for the efficient and effective retrieval and description of the content of the image by using the Finding Regions in Pictures method. As mentioned in section 2.2.3 the image processing function is divided into four sub-function:

1. **Read Image**
2. **Segment Image**
3. **Extract Features**

### 4.1.2 Stimulus/Response Sequence

The *Process Image* function is called by the *Neural Network Function* and *Image Classification* functions. Each provide an image to be processed. The data flow of the function is as follow:

**Basic Data Flow:**

1. *Process Image* function is called by Neural Network Training or Image Classification function
2. An Image file is provided by the calling function

3. *Read image* function reads the image header, validates it and extracts necessary values such as image height and width

4. *Segment Image* function segments the provided image through the FRIP method and stores the result in volatile memory by using the image header data previously obtained

5. *Extract Features* function extracts all features from the newly created segmented image and creates a normalized feature vector that can be used in the PNN

6. The feature vector is returned to the calling function

**Alternative Data Flow:** Not applicable

### 4.1.3 Read Image Function

This process will read, validate and store the information located at the header of the image. As mentioned in section 2.4.1, images are of BMP format. The format of the image is described by it's file header. This format's specific file header is simplistic. Important header fields such as file size, image width, image height, horizontal and vertical resolutions, colors and others are stored in memory for in order to segment the image. Information such as image width and image height are used for the segmentation filters as functional parameters. The matrix of pixels information is stored in memory together with the obtained header information in order to be used by the next task (image segmentation). For more details of the BMP image format refer to the appendix section 5.4 page 20.

### 4.1.4 Segment Image Function

In this process, image segmentation refers to partitioning an image into different regions that are homogeneous or similar in image characteristics. To do this new space is opened in dynamic memory to store the segmented image by using the extracted header information obtained form the *Read Image* sub-function. The segmentation function use traditional algorithms and statistical methods to segment each frame. In this case we use Finding Regions in Picture, which segments an image into regions using scaled and shifted color, CIE L*a*b* color and edge distribution of image.

In details, the FRIP process include the use of an adaptive circular filter, which coarsely quantize the image's color, and a region iterative merging and labeling. As an extension to this system the segmentation will be parallelized and clustered for optimization purposes. In figure 5 we can observe an example of segmented images by using FRIP.

**Finding Region in Picture (FRIP) process**

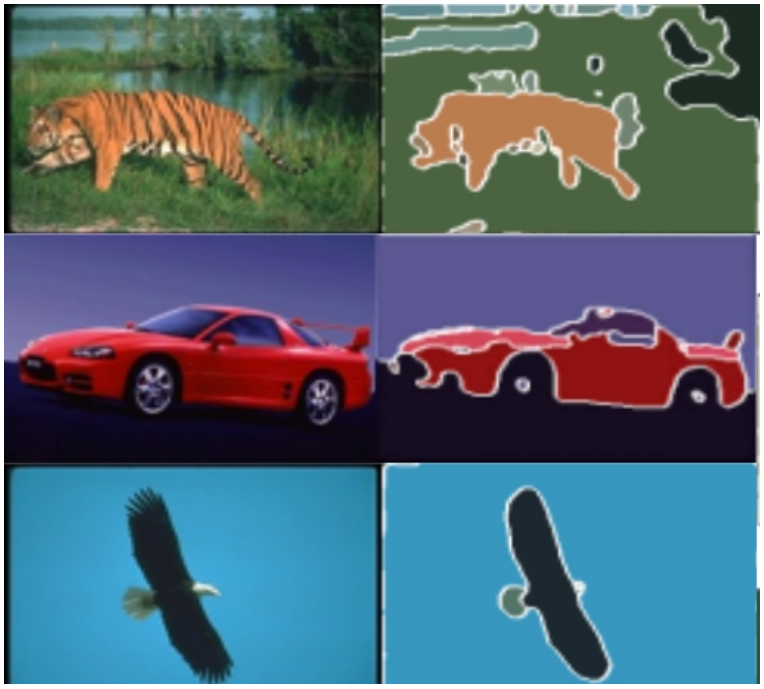

Figure 5: Segmented images example

1. *Adaptive Circular Filter:* CIE L*a*b* and RGB color space is used to represent color features. CIE L*a*b* color space is generated by linearly transforming the RGB color space to the XYZ color space by a nonlinear transformation. After the transformation, images are smoothed to remove noise using a median filter. Then the adaptive (7x7, 11x11, 15x15 pixels) circular filter is applied.

2. *Iterative Level Using Region Labeling and Iterative Region Merging:* After the first-level segmentation, some regions may be declared to be similar if they have similar color properties, even though they are in fact semantically different regions. Therefore, each region is labeled as different regions using the connected-component algorithm. In the step, different region numbers are assigned to each region.

   After region labeling, if the number of regions is over 30, we repeat the circular filtering starting with the smallest filter (7x7) in order to merge small patches into adjacent regions that are not merged at the first level, and then perform region merging with a modifiable threshold T. We restrict the maximum number of regions to TBD.

The whole process is observable at figure 6

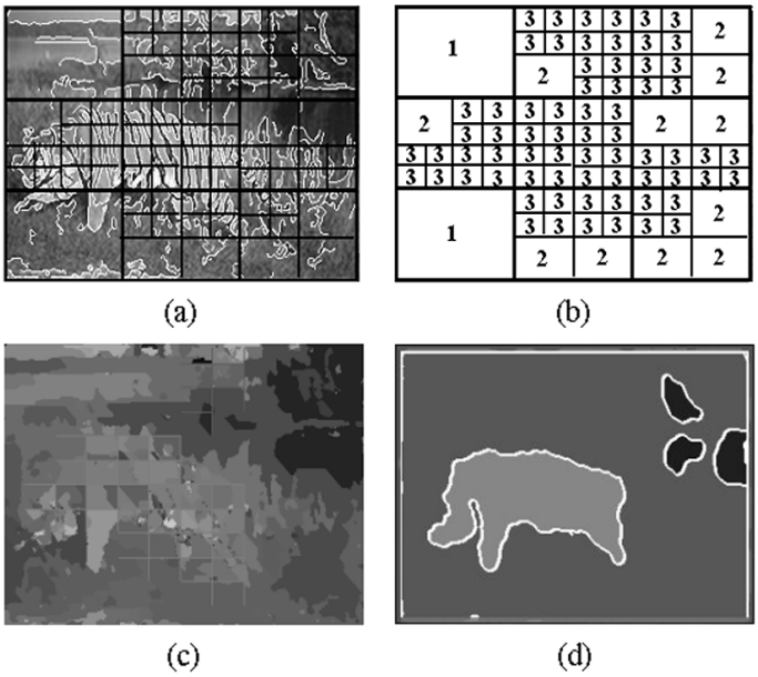


Figure 6: Image segmentation using adaptive circular filter and region labeling/merging

#### 4.1.5 Extract Features Function

From the segmented image, features are extracted from each region. Features extracted from each region are normalized and converted to a feature vector. We extract five specific features from each regions:

| **Feature** | **Details** |
|---|---|
| Color | The average color of RGB(Ar, Ag, Ab) and CIE L*a*b*(Al, Aa, Ab) color space |
| Texture | The Biorthogonal Wavelet Frame(BWF) |
| Scale/NArea | Size of the region in NArea form. That is number of pixels of a region divided by the image size |
| Location | Centroid of the region |
| Shape | Eccentricity and Modified Radius-based Signature |

Table 5: Region specific extracted features

All extracted features are then normalized to 0 - 1. Afterwards all features from a region are converted to a feature vector for latter PNN use. This function concludes the *Image Processing*

presented feature.

#### 4.1.6 Functional Requirements

- IP-REQ-1: Image segmentation Region-Based Image Retrieval application shall be the Finding Regions in Picture
- IP-REQ-2: Five specific features found at table 5 shall be extracted from the segmented image's regions
- IP-REQ-3: Maximum number of regions shall be 100

### 4.2 T-REQ-0 Train PNN

#### 4.2.1 Description

The requirements for this task are as follow:

- T-REQ-1: Use a Probabilistic Neural Network for training
- T-REQ-2: The PNN use normalized feature vectors as input
- T-REQ-3: The feature vectors used in the PNN are extracted from the *image processing* shared process
- T-REQ-4: Images and class per image index used in this function are provided by the tagged image database

### 4.3 C-REQ-0 Classify Image

#### 4.3.1 Description

The requirements for this task are as follow:

- C-REQ-1: Use a Probabilistic Neural Network for classifying images
- C-REQ-2: The PNN use normalized feature vectors as input
- C-REQ-3: The feature vectors used in the PNN are extracted from the *Image Processing* shared process

- C-REQ-4: Images used in this function are provided by the UAV

# 5 Other Nonfunctional Requirements

## 5.1 Performance Requirements

## 5.2 Software Quality Attributes

## 5.3 Other Requirements

## 5.4 Appendices

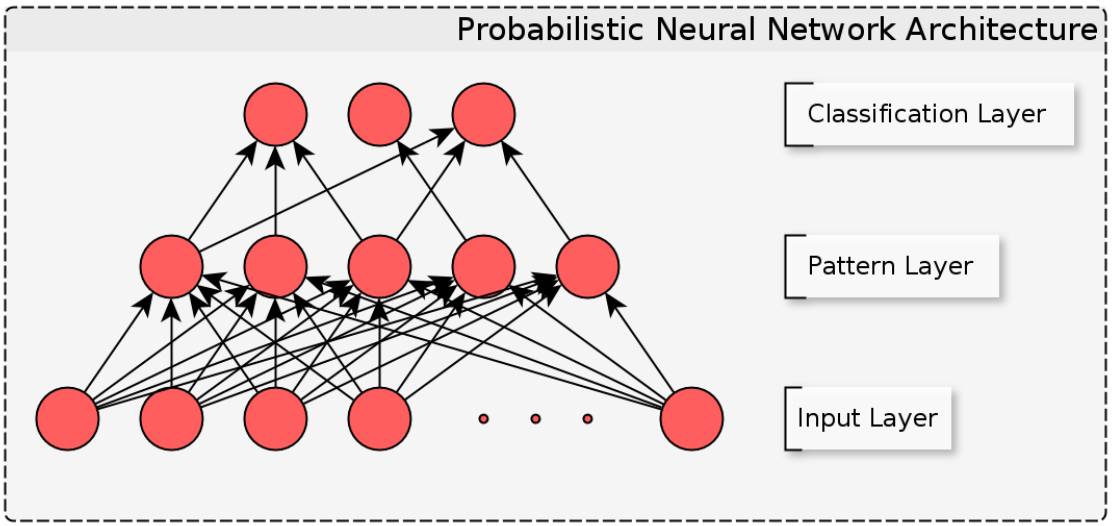


Figure 7: PNN architecture

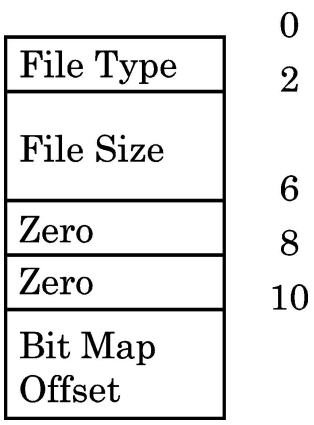


Figure 8: BMP header architecture

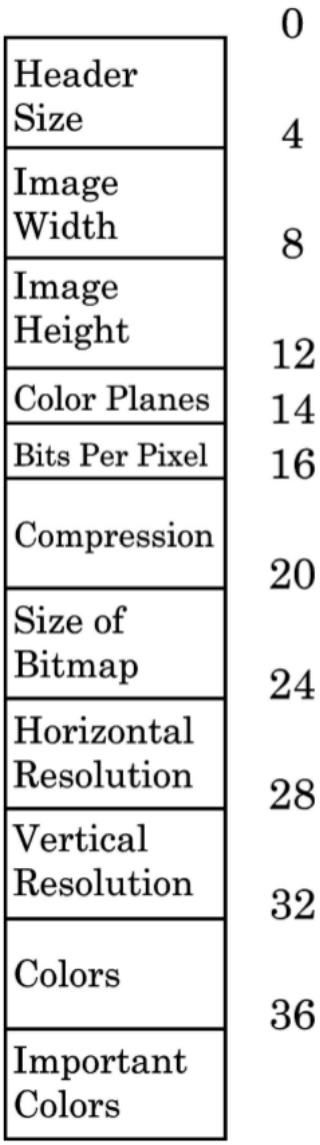


Figure 9: BMP header BitMap architecture