TED University

Department of Computer Engineering

Senior Design Project

High-Level Design Report

Group Members:

- Umay ŞAMLI [52255193822]
- Rümeysa ÖZ [10216424224]
- İhsan Melih ŞİŞMAN [56725508390]
- Fatih Mehmet YİĞİTEL [38651154902]

Supervisor:

- Yücel ÇİMTAY

Jury Members:

- Gökçe Nur YILMAZ
- Saiful ISLAM

High-Level Design Report
January 03, 2023

The High-Level Design document graduation project course is offered under CMPE491

Contents

1 Introduction

1.1 Purpose of the system

Fog, dust, smoke, and haze, which occur naturally or artificially through our environment, reduce our ability to see. Those same effects, which impair vision, are significant issues that make people's lives difficult. To act quickly and effectively, it is essential to understand the external situation as well as any dangers in the environment. When the vision is slightly or entirely reduced, the decrease in recognition causes issues with security and perception. Terrorist attacks on the ground of defense in partially cloudy and cloudy meteorological conditions are among the most prominent examples of this issue. Not only that but also affects our daily life, this problem also causes a lot of problems in transportation, for example, the haze where the plane will land. About this topic, when we look at recent times (2008-2022), technological devices and various developments in various fields (such as artificial intelligence, machine learning, and image processing) can display images even when fog exists and, at the very least, enable us to distinguish objects under fog and haze with a certain success rate. The technology under discussion is intended to be a system that successfully purifies hazy pictures. Some systems now do this with a certain success rate. We will deploy some systems such as Capsule Networks, and Dark Channel Prior (DCP). These systems were developed to address this issue by using deep learning to restore the details and contrast in hazy images, resulting in clearer and more visually appealing images in this system, the purpose is to purify the images and increase the success rate of purifying. In this context, it can be said that this project aims to successfully defog and dehaze some particles in the air like haze and fog. Briefly, we apply a process of removing or reducing the amount of haze present in an image because haze and fog are common problems in outdoor images and can significantly reduce visibility and the overall quality of the image and comparisons can be seen by the user. By using these algorithms, user can dehaze their images and compare these systems with each other and can see the success rate.

1.2 Design goals

This system aims to increase the clarity of camera images in misty and foggy weather. Several features have been added to our software design to achieve this goal. These features allow the software to be more easily understood and developed.

1.2.1 Usability

According to the system we have pre-designed, everything is presented to the client very easily. The original version of the photo is uploaded to the system with a simple button and the system dehazes and places it where necessary. Furthermore, the user can compare the algorithms with each other. After the comparison is made, the success rate is specified.

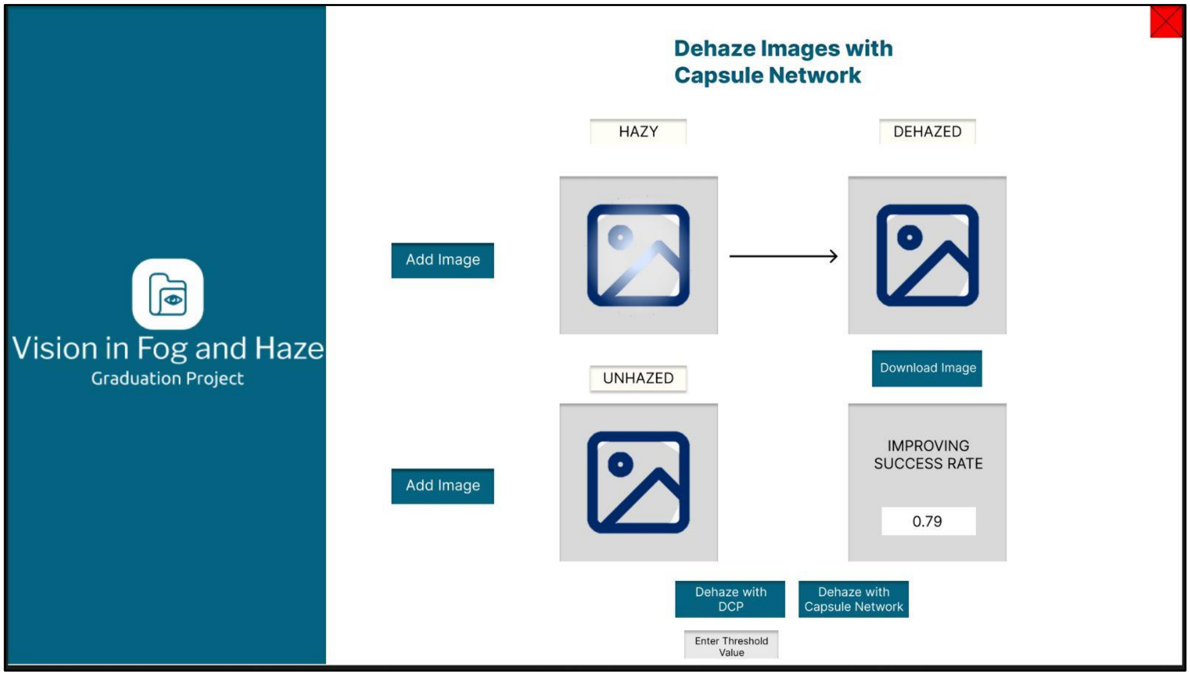

Figure 1: General Design of the Program

1.2.2 Reusability

The system aims to do this for every uploaded photo or video. Therefore, it can be used endlessly. If a photo suitable for this purpose is not uploaded to the system, the system will not make any changes.

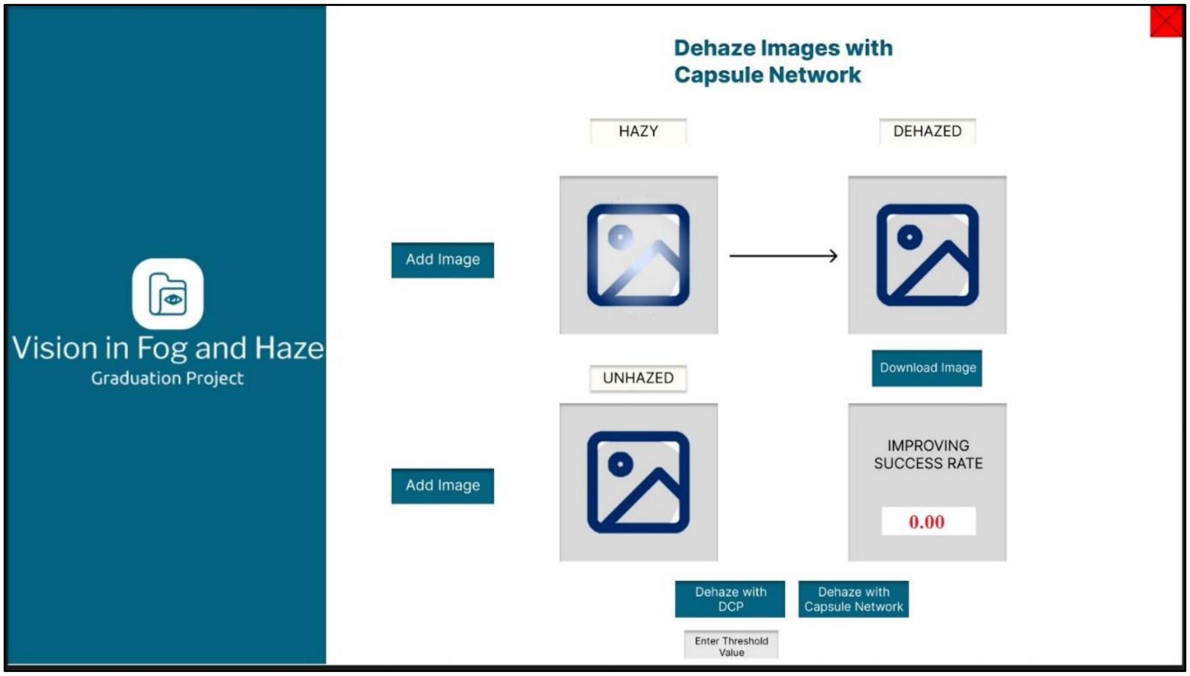

Figure 2: Picture That Has Not Changed

1.2.3 Loosely Coupled

By minimizing the dependencies between modules in our code design, each module, each class, and each function is in a structure that is responsible for its task. With the loosely coupled code design, we will be able to make changes in our codes easily. In this way, when we want to amend the code, this will be done easily.

1.2.4 High Cohesion

The unity of the elements in our modules and the focus of the module will be at a prominent level and will be placed in line with their purposes. In this sense, it can be said that our system will be within the scope of high cohesion.

1.2.5 Maintainable Code

Since this coding design is highly cohesion and loosely coupled, which we mentioned above, its maintainability will be simple and will not require difficulty because a modification or problem in one subsystem should not have an impact on the others. This is critical for maintaining service quality when additional modules are added.

1.2.6 Performance

While this program is running, not much will be done. Since it has a structure that can be summarized as loading the picture and running the code, it will have a prominent level of efficiency in terms of performance efficiency.

1.2.7 Availability

Since all these processes will be done through a program on the computer, the first version will be ready for use whenever you turn on your computer. Improvements will be made in the part of the integration of this system into systems for the cycle.

1.3 Definitions, Acronyms, and Abbreviations

- **Capsule Network:** A capsule network is a type of deep learning model that was introduced as an alternative to traditional convolutional neural networks (CNNs). The main purpose of a capsule network is to improve the ability of a model to recognize and classify objects in images, particularly when the objects are partially occluded or presented from different viewpoints.
- **Dark Channel Prior:** Dark channel prior (DCP) is a technique used to estimate the atmospheric light in an image, which is a key step in image dehazing. The atmospheric light represents the amount of light that is scattered by the atmosphere.
- **Client:** A desktop computer or workstation that can access information and programs from a server on a network.
- **Dehazing:** This is a distinct phenomenon in which extremely minute, dry particles are suspended in the air rather than water droplets. These particles are undetectable to the human eye, yet they contribute to the opalescent look of the air. Dehazing is the process of removing this haze from the video or camera.
- **Defogging:** The term fog refers to the resultant visibility of less than 1 km. However, in public predictions, this often refers to visibility less than 180 m. Defogging is the process of removing this fog from the video or camera.
- **Atmospheric Light Scattering:** Scattering happens when particles or big gas molecules in the atmosphere contact with electromagnetic radiation, causing them to be deflected from their intended course.

1.4 Overview

In general, this system aims to reduce fog and haze in cameras and videos. In this way, it will be able to give much more images. After this stage, it is possible that this project within this scope can be integrated into any system. For example, when there is heavy fog and haze during aircraft take-offs and landings, the integration of soldiers' helmets will provide much convenience when there is heavy fog and haze.

Our application consists of a desktop application in its first stage, you load the foggy image, and it corrects and returns that image to you. It does this with the aforementioned algorithms running in the background and their development. If you upload an irrelevant image, it returns the image to you without making any changes. Let's take a look at it to step by step: First, the user who opens the application uploads the image and then takes the printout to the dehaze button and looks at the comparison screen. In this way, he understands the success rate and the difference. On top of that, the user can compare each algorithm’s success rate with each other.

It gains superiority by neutralizing this fog and haze phenomenon, which affects people's daily lives, in a sense, and when it is used in the field of the defense industry, terrorists are prevented from taking advantage of the fog and haze. The benefits brought by this project provide incalculable benefits for the person using it in this context. The realization of this project will make life easier for even normal people.

2 Current Software Architecture

At this stage, we do not have any current system.

3 Proposed Software Architecture

In our project, we just have the client and back-end part.

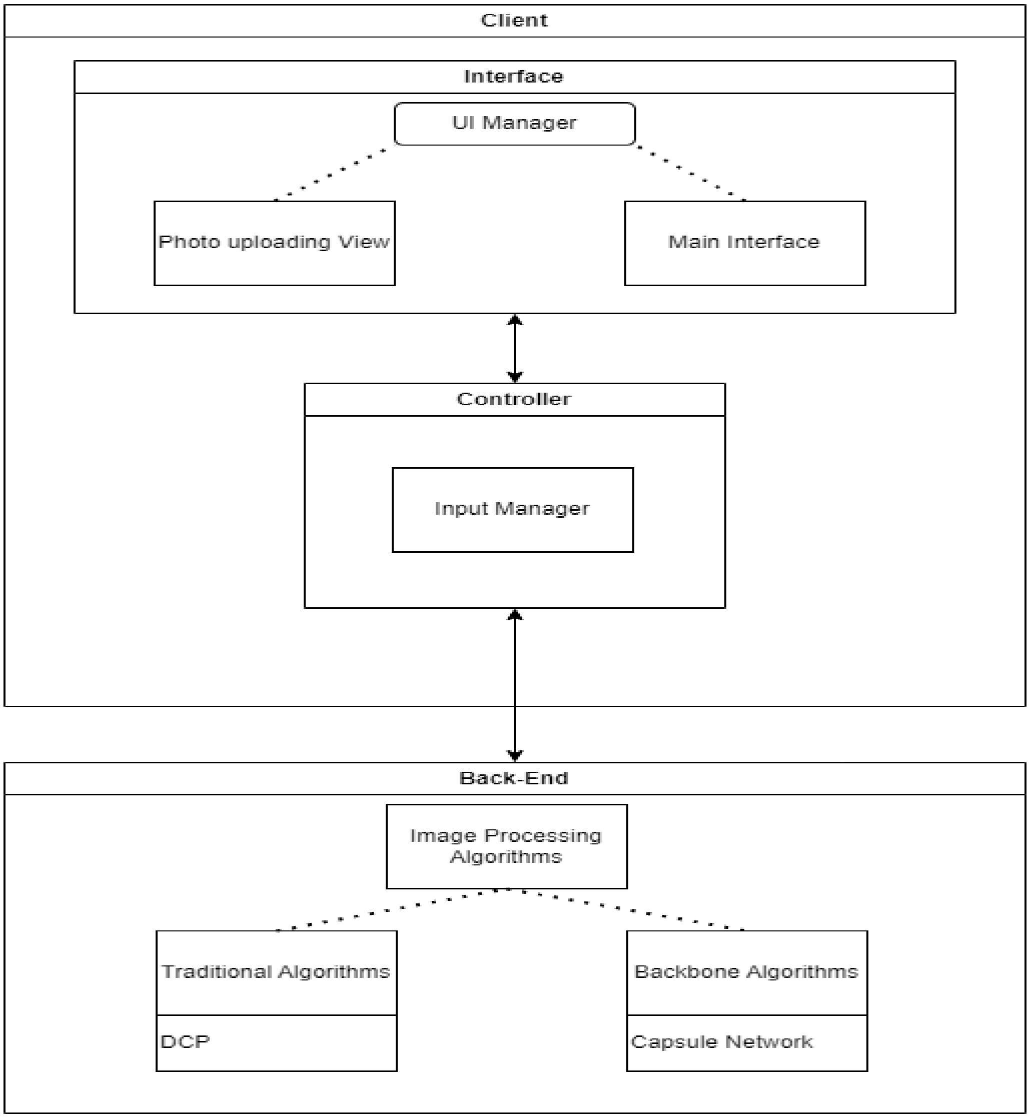

Figure 3: General Look Inside the System

3.1 Overview

3.1.1 Client

In the client part, we are having the photo from the user and convert the input data to a convertible image file (if it is possible). Otherwise, we gave an error about importing a document into the system.

- **Interface**

- **Photo Uploading View:** The class-specific class will give a client the ability to choose the specific jpeg/PNG file that he wants to upload.
- **Main Interface:** The specific class will be responsible for the main view of the program.

Controller

The part where we organize the inputs that come from the client. We look at whether is uploaded file is acceptable. If it is not, it will give an error to the user. The controller will also be responsible for having the output from the back-end part.

- **Input Handler:** It looks at the type of file that is uploaded. If it is a proper input, then we send it to the back-end part to process the image. The input handler will be responsible to manage the outputs of the back-end part.

3.1.2 Back-End

At back, end part, we will dehaze the algorithm by the chosen algorithm. Firstly, we will choose the algorithm type that we will use. Then we will choose the specific algorithm from the pool of chosen algorithm types.

- **Traditional Algorithms:** the pool of Classic dehazing algorithms that we can find commonly in use.
- **Backbone Algorithms:** Pool of new Dehazing algorithms that we implement.

3.2 Subsystem Decomposition

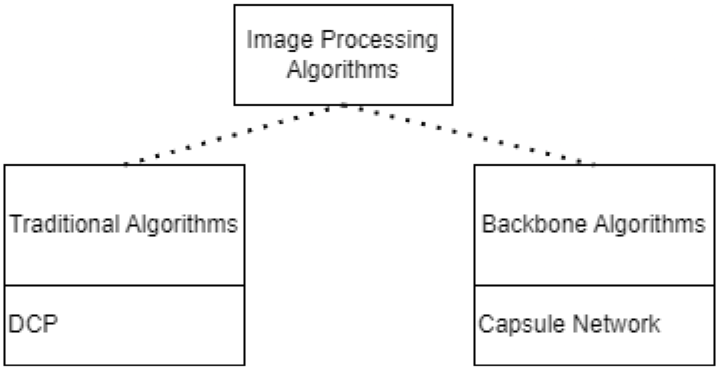

Figure 4 : Subsystem Diagram

3.3 Hardware / Software Mapping

-

3.4 Persistent Data Management

We are not holding the data in the system. That's why we are not managing the data.

3.5 Access Control and Security

In our system, the client will freely use the program without any authentication or security process. That's why we don't have an access control security plan.

3.6 Global Software Control

-

3.7 Boundary Conditions

Users can upload only image files such as:

- JPEG
- PNG
- JFIF
- PIPEG
- PJP
- TIFF

- ✓ Non-image files are restricted.
- ✓ Besides this boundary, we don’t have any boundaries to use the system.

4 Subsystem Services

The project has two sub-systems that will bring the project to the fore. One of them is DCP, which stands for dark channel prior, which is the traditional method for clearing foggy, haze-free images. The other is the capsule network structure that enables our machine to learn.

4.1 Dark Channel Prior (DCP):

Dark Channel Prior (DCP) is an algorithm that estimates the transmission map of a scene in a single image by using a dark channel. The dark channel of an image is defined as the minimum intensity value of the image across all color channels at each pixel. The transmission map is a measure of how much light is attenuated by the scene at each pixel.

To implement DCP, you will need to do the following:

1. Compute the dark channel of the input image. To do this, you will need to find the minimum intensity value at each pixel across all color channels.

2. Estimate the transmission map of the scene using the dark channel. You can do this by taking the minimum intensity value of the dark channel at each pixel and dividing it by the maximum intensity value of the dark channel across the entire image.

3. This method first takes the image from the user. Secondly, a threshold value is determined. Thirdly, it compares the uploaded image with the threshold

value and returns the processed image as output by the algorithms. Lastly, output sends to the controller.

4. Dehaze the input image using the estimated transmission map. To do this, you can use the following formula:

Dehazed image = (Input image - Atmosphere) / Transmission map + Atmosphere

where "Atmosphere" is the average intensity of the input image in the dark channel, and "Transmission map" is the estimated transmission map of the scene.

4.2 Capsule Network:

Capsule Neural Network (Caps Net) is a useful machine learning system. It is type of artificial neural network (ANN). It can be usefull to model better hierachical relationships. To implement a capsule network some libraries in phyton will be used.

In-capsule network:

- ✓ **Prepare your data:** This may include image resizing, cropping, and normalization.
- ✓ Divide your data into training and testing sets as follows: After you've trained your model, you'll need a separate set of data to evaluate it on.
- ✓ **Build the capsule network architecture:** This involves defining the layers and their parameters. A capsule network consists of multiple layers of capsules, each of which contains multiple "neurons" that are responsible for detecting distinctive features in the input data.
- ✓ **Train the model:** Use the training data to train the model by minimizing the loss function.
- ✓ **Evaluate the model:** Use the test data to evaluate the performance of the trained model

5 Glossary

- **DehazeNet:** It is an end-to-end system to remove haze in the images.

- **CNN:** CNN is a network architecture for deep learning algorithms. It is especially used for tasks that involve the processing of pixel data and recognition of images.

- **Capsule Network:** The network is trained by minimizing the Euclidean distance between the image and the output of a CNN that reconstructs the input from the output of the terminal capsules.

6 References

- https://arxiv.org/abs/1710.09829
- https://ieeexplore.ieee.org/document/5567108
- https://www.mdpi.com/1424-8220/20/22/6207
- https://ieeexplore.ieee.org/document/6816200
- https://ieeexplore.ieee.org/document/4472246
- https://ieeexplore.ieee.org/document/4442376
- https://medium.com/analytics-vidhya/introduction-to-capsule-networks-66ebcdde4837
- http://mmlab.ie.cuhk.edu.hk/archive/2011/Haze.pdf
- https://ltb.itc.utwente.nl/509/concept/89052#:~:text=Atmospheric%20scattering%20occurs%20when%20particles,redirected%20from%20its%20original%20path