**System Design Document**

**For**

**Machine Learning Algorithm for Rhinoplasty (M-LAR)**

Team members:

* Chris Graziano
* Victoria Jordan
* Anton Kiselev
* Jacob Preseau
* Chintan Thakrar

|  |  |
| --- | --- |
| Version/Author | Date |
| 1.0/Team 1 | 9/29/20 |
| 1.1/Team 1 | 10/1/2020 |
| 1.2/Anton Kiselev | 10/27/2020 |
| 1.3/Anton Kiselev | 10/28/2020 |

**TABLE OF CONTENT**

[1](#_30j0zll) INTRODUCTION 3

[1.1](#_1fob9te) Purpose and Scope 3

[1.2](#_3znysh7) Project Executive Summary 3

[1.2.1](#_2et92p0) System Overview 3

[1.2.2](#_tyjcwt) Design Constraints 3

[1.2.3](#_3dy6vkm) Future Contingencies 3

[1.3](#_1t3h5sf) Document Organization 3

[1.4](#_4d34og8) Project References 4

[1.5](#_2s8eyo1) Glossary 4

[2](#_17dp8vu) SYSTEM ARCHITECTURE 4

[2.1](#_3rdcrjn) System Hardware Architecture 4

[2.2](#_26in1rg) System Software Architecture 4

[2.3](#_lnxbz9) Internal Communications Architecture 4

[3](#_35nkun2) HUMAN-MACHINE INTERFACE 4

[3.1](#_1ksv4uv) Inputs 5

[3.2](#_44sinio) Outputs 5

[4](#_2jxsxqh) DETAILED DESIGN 5

[4.1](#_z337ya) Hardware Detailed Design 6

[4.2](#_3j2qqm3) Software Detailed Design 6

[4.3](#_1y810tw) Internal Communications Detailed Design 7

[5](#_4i7ojhp) EXTERNAL INTERFACES 7

[5.1](#_2xcytpi) Interface Architecture 7

[5.2](#_1ci93xb) Interface Detailed Design 8

[6](#_3whwml4) SYSTEM INTEGRITY CONTROLS 8

7 APPENDIX 9

**SYSTEM DESIGN DOCUMENT**

# **INTRODUCTION**

## **Purpose and Scope**

This document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, and external interfaces for the Machine Learning Algorithm for Rhinoplasty (M-LAR) system.

## **Project Executive Summary**

This section provides a description of the M-LAR system from a management perspective and an overview of the framework within which the conceptual system design was prepared.

### System Overview

The system will provide a patient with representative similar faces with different nose types that align with the measurements of their face, therefore allowing the patient to select which nose is preferred for their rhinoplasty procedure. Below is the high-level architecture and context diagram of the system’s overall design.

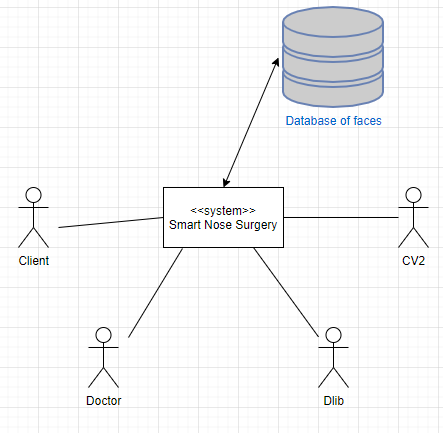


Figure 1. Context view of the system.

### Design Constraints

* The current version of the project was made with the assumption that it will work only with 2D photos, such as a face looking straight forward into a camera with aligned sides.
* The person in the photo should remove their glasses if they wear them, as they disturb the gathering of the face coordinates.

### Future Contingencies

* The current version of the Python application displays windows via the Tkinter library [3]. This could be changed if another library that could better suit our needs is discovered.
* This project, like any other machine learning project, is dependent on the initial data inputs used for training, categorization and prediction. At the current stage, it is assumed that the current selected dataset is sufficient in terms of quality and quantity to successfully complete the study.

## **Document Organization**

This document is designed to describe the structure of the Machine Learning Algorithm for Rhinoplasty (M-LAR) system. The following sections will provide information on what the product does, limitations, interactions, interfaces, hardware and software designs, and security.

## **Project References**

* + 1. System Requirements Specification

## **Glossary**

* KNN - "*k*-nearest neighbors", a machine learning algorithm that, when given a test point represented by a point on the coordinate plane, will find the *k* points in the data set, also in the coordinate plane, that are nearest to the test point
* M-LAR - Machine Learning Algorithm for Rhinoplasty (M-LAR) system
* "Patient", “Client” and "User" are used as interchangeable terms.

# **SYSTEM ARCHITECTURE**

This section describes an overview of the hardware and software architecture for the M-LAR system and subsystems.

## **System Hardware Architecture**

For the current version, the system doesn’t have any hardware components.

## **System Software Architecture**

Figure 2: Data-flow diagram. Level 0

As shown in Figure 2, the system contains 3 main components: user, main process and doctor.

## **Internal Communications Architecture**

As shown in Figure 4, the system communicates via passing results of previous processes.

# **HUMAN-MACHINE INTERFACE**

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user.

## **Inputs**

The main input for the current version is a 2D frontal photo of a patient face, in which the patient holds his/her head upright without tilts and turns, and looks straight into the camera. The inputs for the external processes are coordinates of facial features. With the current status and scope of the project, it is assumed that the input will always be a face; this implies that exceptions are currently not being handled.

## **Outputs**

The main output of the current version of the system is the required coordinate changes on the patient’s nose.

# **DETAILED DESIGN**

This section provides the information needed for a system development team to build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product.

## **Hardware Detailed Design**

As mentioned before, for the current version of the system, there are no hardware components and requirements.

## **Software Detailed Design**

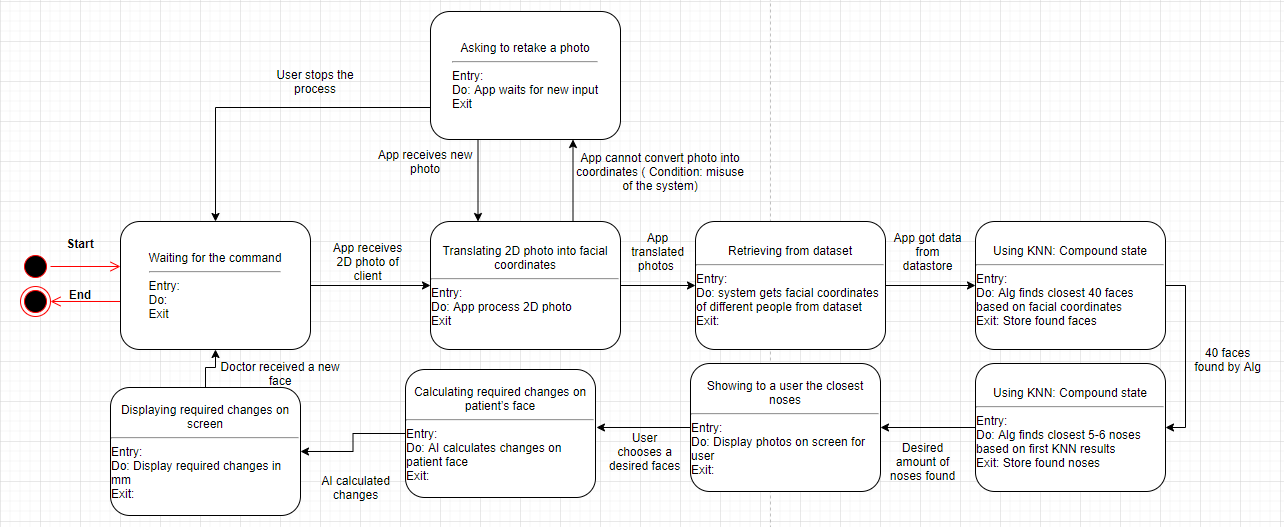


Figure 3: State Chart Diagram

Figure 3 describes what states and conditions will be cycled through the system.

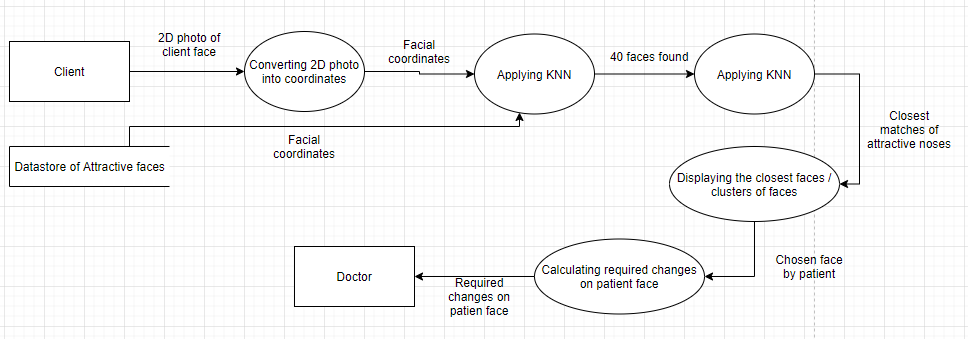


Figure 4: Data-flow diagram. Level 1

As shown in figure 4, the system will process 2D face into facial coordinates using the Python Dlib[2] and OpenCV modules[1]. The coordinates are computed with respect to the origin, which is the top-left corner of the screen. It is inferred that the final matrix that is to be fed to the ML model will be independent of the resolution of the input image, because it will consist of proportions or ratios between coordinates. The use of ratios is reliable as they are automatically adjusted to provide a consistent matrix with respect to each face. After that, the system will pass the coordinates to the KNN algorithm, which will use the coordinates to calculate the Euclidean distance relative to the user's face coordinates, and after that the KNN algorithm will be applied again to find the closest noses.

## **Internal Communications Detailed Design**

The main instance of internal communication occurs when the system reads and converts the dataset of face images into a respective set of coordinate points to send as input to the KNN algorithm. The dataset sends a set of images, currently stored as JPG files, the face recognition component converts these images into sets of coordinates, then the KNN algorithm receives the set of coordinates.

# **EXTERNAL INTERFACES**

The software model implemented uses multiple predesigned, developed, and trained models to assist with the development of the system. This decision was considered to complete the specified requirements within the allocated times. Modules include but are not limited to:

1. Dlib
2. OpenCV

These modules help the system get the facial coordinates from an aligned image and then change the image to display those points, respectively. It is assumed that the final system is expected to be used by the patient, who takes a photo of their face and then picks from some presented noses.

## **Interface Architecture**

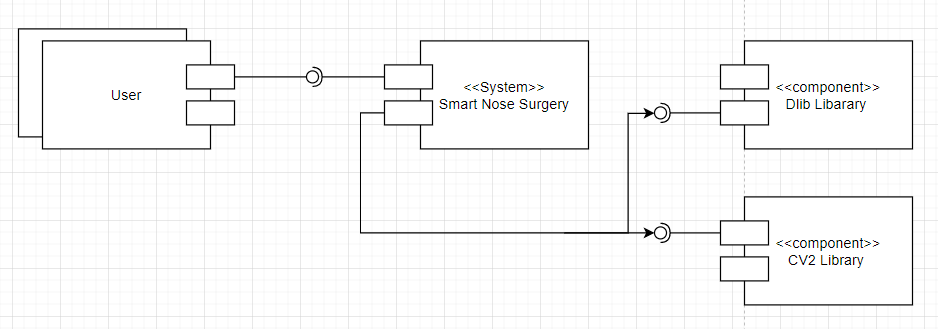


Figure 5. Interface architecture.

* + 1. Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software. Dlib is responsible for assigning the facial coordinates using a given image, essential for scanning the user’s face

* + 1. CV2

CV2 is a subcomponent of OpenCV library. OpenCV-Python is a library of Python bindings designed to solve computer vision problems. CV2 uses the data generated by Dlib to alter the given image by overlaying the points on it, This is also an essential process to scanning the users face

## **Interface Detailed Design**

* When arrays are made by the dlib library, they need to be shaped into a matrix.
* Errors in the system will mostly come from a failure of dlib to assign all coordinates, due to a substandard photo.

# **SYSTEM INTEGRITY CONTROL**S

The Python application is expected to have a security level of login input. The login will consist of a user-name and password. This will add a security layer to the system. The users i.e. the patient and doctor will have different levels of access more specifically the patients will not be able to view the computed difference. The log-in details of each user will be stored on an external SQL server.

The final submission of this project is expected to have an exception handler to make sure that the user submits a picture of the face and nothing else. The system will be able to detect and disregard any picture that does not contain a face that can be used for the processing section.

**7 APPENDIX**

References:

[1] Dlib C++ Library. (n.d.). Retrieved October 1, 2020, from <http://dlib.net/>

[2] Introduction to OpenCV-Python Tutorials. (n.d.). Retrieved October 1, 2020, from https://docs.opencv.org/master/d0/de3/tutorial\_py\_intro.html

[3] Tkinter - Python interface to Tcl/Tk. (n.d.). Retrieved October 1, 2020, from https://docs.python.org/3/library/tkinter.html