System Requirements Specification Documents

Automated Spinal Segmentation and Remote Monitor Calibration for Surgical Assessment

Project for EN.601.456 Computer Integrated Surgery II

Group 8:
Damiano Marsilli, Arijit Nukala, Jonathan Young

Introduction

Purpose

This document is intended to outline the design of a calibration method for IMUs placed on a patient's back along their spine. Calibration is necessary for our mentor's team, CurveAsssure, as they intend to measure the behavior of the patient's spine over the course of 48 hours. To ensure that there is minimal error data collected between the IMUs, calibration must take place periodically. Therefore, our team develops a calibration procedure that uses a monocular video to locate the IMUs and utilizes a transfer function that aligns the IMUs' readings with the vision readings. Lastly, this process must take place on the cloud.

Intended Audience

Our project is merely a prototype of a pipeline for uploading, processing, and outputting the results of the video recording and IMU data of the patient's movement. However, our pipeline is not meant for widespread clinical use. Until there has been further testing of the robustness of our system, this pipeline should only be used as a prototype. With this in mind, our team aims to utilize this document to inform users of the prototype's design.

Intended Use

Current methods of analyzing the feasibility of spinal surgeries can often be inaccurate due to their static nature. Therefore, CurveAssure aims to develop a product that builds an accurate model of the patient's spine. To assist in their product, our team's pipeline will enable the IMUs to be calibrated at the patient's home utilizing the only camera and the sensor data provided. All the computationally heavy tasks will be done via the cloud to enable ease of use at home.

Overall Description

Product Perspective

From a high-level perspective, our design encompasses the following:

1. **Spinal Keypoint Labeling**
   A machine learning model that is able to locate points on the spine of a patient
2. **Transfer Function**

A function that is able to map orientation values between computer vision-generated coordinates and data from the IMU.

3. **Cloud Functionality**
   A pipeline that is able to host the above-mentioned features while also creating a systematic way of uploading and processing data.

Product Features

Primary stakeholders of this system are clinicians, who will use the system's output to make surgical decisions, and patients, who will use provided IMUs to capture spinal angle data for clinicians to analyze. Patients must be able to:

- Configure a mobile device to function as the video input for the system
- Attach the provided spinal IMUs to approximate locations on the back
- Upload data for cloud processing by the system

Clinicians must be able to:

- View spinal angle statistics and visualizations processed from IMU input data

This system should be simple for both stakeholders to use as to maximize the impact it can provide for the patient.

Operating Environment

AWS - Amazon Web Services, is the platform in which all processes will take place. The environment in which both the transfer function and the keypoint processing will take place is through Py-Torch-oriented Ubunutu 22.10.04 LTS. All lambda functions and scripts in the EC2 instance utilize boto3.

Assumptions and Dependencies

The project assumes the availability of monocular RGB video of the patient, as well as the assumption that the patient is able to adhere to a standardized data-collection procedure from home. Another assumption is that the spine of the patient is at most only marginally deformed. Whilst it is challenging to quantify this assumption, we expect the spinal keypoint estimation models' performance to deteriorate as patient spines deviate from standard curvature.

Dependencies to complete the project include keypoint-annotated videos and IMU data required for training, as well as the AWS credits necessary to implement the pipeline and train our models.

System Features and Requirements

Functional Requirements

Spinal Keypoint

The spinal keypoint estimation model must be able to estimate spinal keypoints on a patient from monocular RGB video. It must be able to estimate keypoints under novel conditions, including but not limited to, different cameras, different lighting conditions, different patient gender and ethnicities, and different spinal curvatures. The model must also be largely robust to occlusion, and we expect an estimation of spinal keypoints even in orientations when the point is not otherwise visible. Lastly, the model must be lightweight, such that it can perform amortized real-time inference for use on the cloud.

Transfer Function

The transfer function's primary requirement is to convert IMU quaternion data into patient angles. The function must be able to use the resulting IMU derived patient spinal angles to output minimal and maximal angles as well as the proportion of patient movement within specific ranges of angles. IMU's inherently carry drift, a phenomenon in which the IMU's angle output loses accuracy over time. Drift is primarily caused by random noise. Due to the extended time period for which the IMU will be installed, the transfer function must be able to handle drift error and minimize it over the period of use. The transfer function is expected to return angles with a degree of accuracy slightly below that of gold standard methods such as infrared.

Cloud Pipeline

High-level-wise, the cloud pipeline should be able to adequately store data for a given patient, process that information, and provide some form of a method for the user to visualize that data, either through the cloud itself or through downloading data and visualizing the results. It must be also capable of hosting both the spinal keypoint task and transfer function on the cloud and link the two systems together while also feeding the data into those systems. When it comes to the runtime of this processing, the system needs to process the data adequately to prevent a backlog of tasks that need to be completed. Adjustments to the scaling capacity of the program will help rectify this. The system must also be able to adequately keep track of which data goes to which patient and isolate patient results, thereby ensuring that no other user can access data that they shouldn't. Lastly, an easy-to-use endpoint system that allows developers, in the front end, to link up with these systems to develop an easy-to-use webpage for customers.

External Interface Requirements

The interface is purely software driven. The user must have a functioning computer that is capable of linking with the internet to upload and download large portions of data. Recommended internet speeds are at least 20 Mbps for upload and 100 Mbps.

Nonfunctional Requirements

Security Requirements

Since this project handles patient data, it is imperative that the entire AWS pipeline is HIPPA compliant. Such requirements include authentication for users, securing data when stored on buckets, and that data is deleted as soon as possible from buckets if the data has no subsequent use. Our team also has to ensure compartmentalization on our part, ensuring that only developers with the proper access can retrieve information from sources. The data also has to be safeguarded against authorized use, such as using it for other purposes as well. Furthermore, our team needs to constantly evolve the pipeline as well and ensure that new methods of security are implemented. This is especially important when the field of cloud computing is constantly evolving. Any subsequent plugins and services have to be vetted for HIPPA compliance as well.

As of the creation of this document, on the back end side, there are no known vulnerabilities that would warrant concern. However, on the front end side, using SSO (Single Sign On) or MFA (Multi-factor Authentication) might be appropriate for this use case.

Software Quality Requirements

All software generated for this system, including the services being used from AWS, should be readable, organized, and well-documented in accordance with good coding practices. This entails creating wikis, version numbers for which libraries are being used, and detailed documentation about the functions and their functionality. Furthermore, software testing should use unit tests, which also should be well documented both in the testing management plan and wiki.