Augmented Reality in Robotic Telepresence
Team cARe

Team members
Sabal Malhan sabalmalhan@gmail.com (Team Leader)
Ethan Wang ethanyuwang@gmail.com
Hanna Vigil hannavigil@gmail.com (Scribe)
Nate Pincus ntpincus@gmail.com
Dongyang Li donnie.ldyang@gmail.com

Introduction

Project Overview:

We will be implementing an interactive augmented reality interface for the RP-Vita robot. Currently this robot provides a video feed, can be moved by dragging a direction arrow, and the camera zoom can be manipulated. In order to expand upon the utility of the robot, we will be using deep learning and image detection techniques to identify common medical equipment and personnel and create context-sensitive actions for the identified objects.

Specifics:

Sprint 1, Research and Planning (complete)

- Study deep Learning principles and TensorFlow
- Start gathering relevant images for dataset
- Study the Robot API - Unable to achieve in Sprint 1

Sprint 2, Begin Developing Prototype Components (in progress)

- Select a specific neural net model to modify and retrain on our dataset
- Developing dataset - using “Where’s the Bear” approach to create large amounts of “fake” images, as real hospital images are difficult to come by
- Robot API - write tests for movement, zooming, centering on given points.
- OpenCV image processing toolbox - write tests for drawing boxes and creating U.I. elements that can later be linked to actions

Target Release:

- Prototype: End of Fall Quarter (December 2016)
- Final: End of Spring Quarter (March 2017)

Team Objectives:

Goals

1. Use the robot’s camera to identify objects of interest. This currently includes monitors (vitals and X Ray), foley bags, hospital beds, and people. This will be done through creating a neural network trained on dataset of images containing these objects in hospital settings.
2. We would must verify that the neural network is accurate (is it correctly tagging objects), as accurate detection in a medical setting is crucial.

3. Use Robot API in conjunction with the neural network to detect these objects from the video stream and draw bounding box around them
4. Context sensitive actions (currently zoom, center, and move towards) appear when the highlighted objects are clicked

Non-goals

1. Context-sensitive actions upon doctors or other personnel. Actions such as “follow the doctor” (have to handle multiple doctors) or “this is the cardiologist Steve” **
2. Identification of personnel or patients - not all doctors wear lab coats, not all patients wear gowns. **
3. Handle or deliver the identified objects. We will not attempt to physically interact with the detected objects. Our problem space is confined to the video stream.
4. Body temperature, weight and height detections. The robot does not currently have the resources it needs for these actions. The company is looking into thermal cameras for future models, but it is still out of scope for now.

**We may be able to achieve some form of these as stretch goals by constraining the domain (require personnel to wear I.D.s or access the hospital D.B. of employee images or voice recordings).

Background

InTouch already offers a working remote care service via video stream to its customers, but we can expand upon the service by creating an augmented reality of the video stream. We will implement the A.R. interface using object detection in order to allow doctors to more easily access information and interact with the remote location. This will lead to an increase in the efficiency and utility of remote care and contributes to the company's vision. In order to implement an augmented reality interface that benefits the users of the product, we will need to train a neural network, which we will accomplish with the help of Google’s TensorFlow.

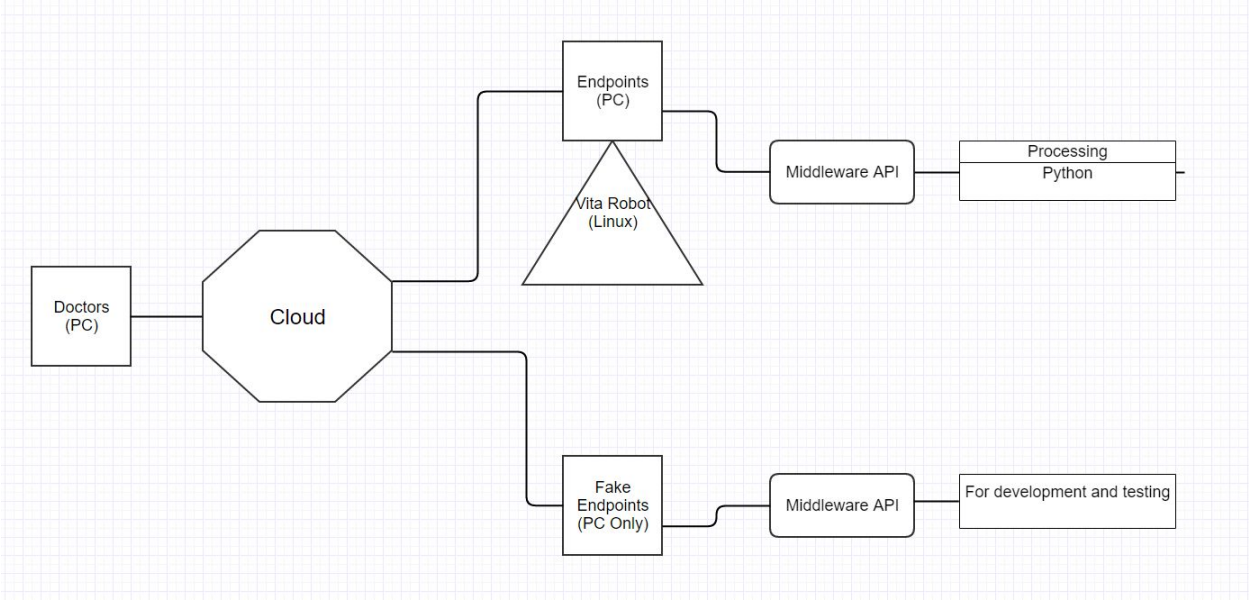
Assumptions

We are given a medical robot (the Vita) with consistent internet connectivity. We are not responsible for a loss of wifi signal that would cause the “call” to drop.

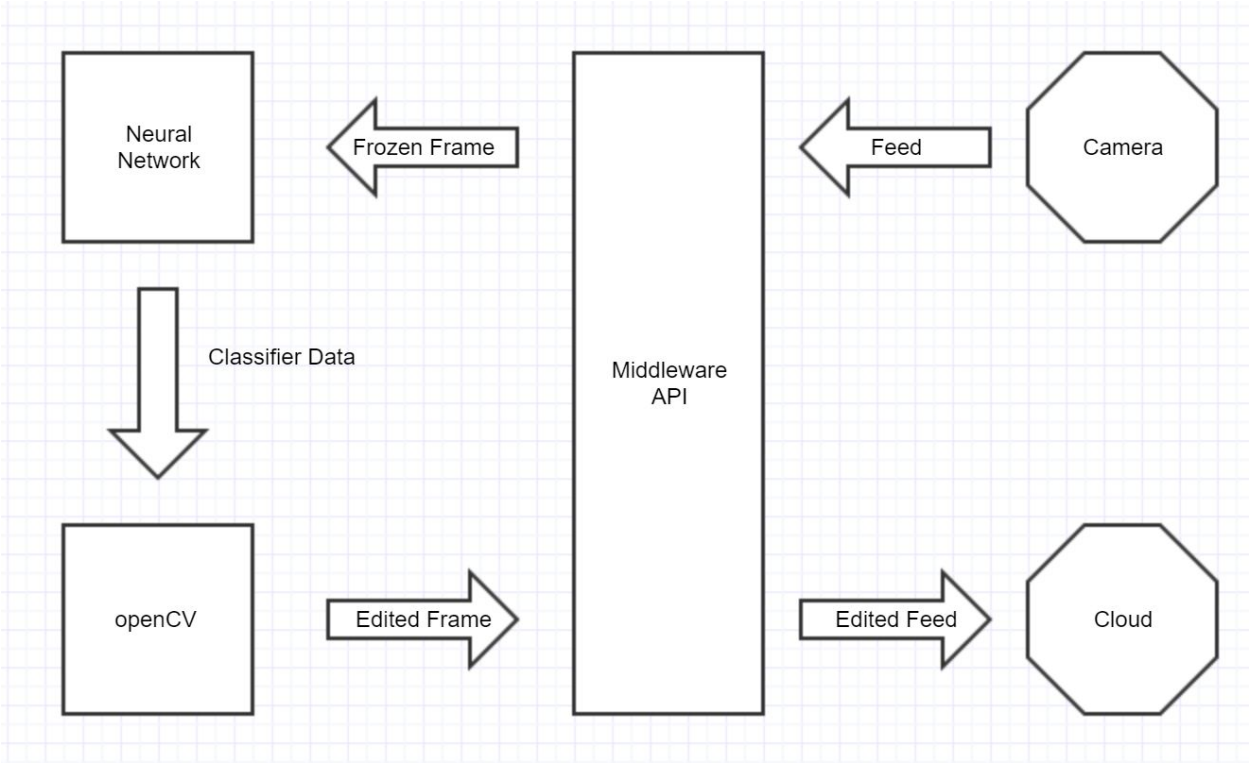
- We are not responsible for the direct control of the robot’s movement: all commands will be handled through at least one layer of api calls.
- We will write code that runs directly on the Vita’s Windows subsystem, in order to ensure a minimal delay between object recognition, augmented reality overlay, and displaying the video feed to the user. Running the object recognition and visual overlay on the device rather than on the cloud server or at the user’s end will hopefully prevent image quality loss (due to compression/decompression on the cloud) as well.
- ~~We assume that we will have data sets from hospital environments on which to train: including images of patients, doctors, nurses, devices, and any relevant audio.~~ Will have to use W.T.B. approach to generate adequately large sets.

System architecture and overview

Two possible endpoints (Vita Robot or a “fake” Robot PC which has the Robot API installed and can simulate robot’s functions).



Within an endpoint, we will use our neural net in conjunction with Robot API and OpenCV to edit images (recognize objects, highlight them, U.I. buttons for context sensitive actions).



Requirements (functional and nonfunctional):

User stories:

1. As a developer, I can create a training set from a set of base images and object images so that I can synthesize a large sample of relevant data (W.T.B.).
2. As a developer, I can use a set of images to retrain the inception V3 neural net so that it can classify objects of medical interest.
3. As a developer, I can take a relative position in an image and draw a bounding box about an object.
4. As a developer, I can implement a “zoom” feature, that performs a zoom of a given a zoom amount and coordinates.
5. As a developer, I can implement a “move” feature, that moves a specified amount in a specified direction.
6. As a developer, I can implement a “center” feature, that moves the camera to a specified bounding box.
7. As a user, I can click the bounding box about an object of interest to see a menu of buttons linked to context sensitive options so that I can interact with the object.
8. As a user, I can click on the “zoom” button so that I can see the object of interest more clearly.
9. As a user, I can click on a “move” button that when called, triggers the robot to move towards an object of interest.
10. As a user, I can click on a “center” button that when called triggers the robot to center a given point so that I can look at an object straight on.
11. As a user, I can click on a “get information” button next to a person that when called displays a person’s name and job title so that I know who I am talking to.

Use Cases:

Use Case:	Create Data set
Actors	Computer CPU
Precondition	Have a base set of images (just backgrounds) and object images with white background
Flow of Events	1) Changes object images to have a transparent background and crops them to the appropriate size. 2) The computer, using openCV will then place the new object images on top of the base images in different positions 3) Adjusts resolution and brightness of the image to resemble the photos that Vita will have access to 4) Stores the new data set in a directory
Postcondition	A set of 1,000 (initially, will expand for final version) testing images is returned with different objects that will need to be classified

Use Case:	Train neural net
Actors	Computer CPU, Tensorflow (inception V3)
Precondition	Have adequately large training data set and test data set
Flow of Events	Basic Path: 1) Inception V3 model is retrained using training set $ retrain.py --image_dir <images_directory> 2) This will store our retrained neural net inside a specified directory 3) Use test set to test and measure the accuracy of the of the retrained model 4) If accuracy is too low -> evaluate if it was due to configuration parameters or small dataset.
Postcondition	Given a test set, neural net can achieve a 60% accuracy (want 95% ultimately) in the classification process.

Use Case:	Recognize and highlight objects
Actors	Vita, Robot API, openCV
Precondition	Have a trained neural net to identify objects under different conditions and backgrounds.
Flow of Events	1) Robot API used to freeze a frame from Vita's camera 2) Neural network is given frozen frame 3) If objects of interest are detected, openCV draws bounding box about those objects 4) Return the edited image to be sent to client's feed through the Robot API
Postcondition	Objects of interest are recognized and highlighted (bounding boxes) in output image

Use Case:	Create context sensitive actions menu
Actors	User, openCV
Precondition	Bounding boxes have been drawn, object classifief
Flow of Events	1) User clicks within a bounding box 2) Draw a gui - dropdown of context sensitive buttons - zoom, move, center 3) If object is a person, also include a “get information” button
Postcondition	Context sensitive actions menu created

Use Case:	Select action
Actors	User, Vita
Precondition	Object is selected, action menu is displayed
Flow of Events	Basic path: 1) User presses one of the actions 2) Menu is removed from screen 3) Vita is notified which action was selected and for which object and initiates that use case Alternative path: 1) User does not click on any of the displayed actions and presses somewhere else on the screen 2) Menu is removed
Postcondition	Action selected and menu hidden OR no action selected, menu hidden

Use Case:	Center object
Actors	Vita
Precondition	Objects are identified and highlighted with a bounding box, center option has been selected from dropdown menu, the object is not already centered
Flow of Events	1) System takes bounding box coordinates and uses those to determine how much the vita should turn its head 2) Robot API is used to tell Vita to rotate a specified angle
Postcondition	The object is centered on the display

Use Case:	Move towards object
Actors	Vita
Precondition	Objects are identified and highlighted with a bounding box, move option has been selected from dropdown menu
Flow of Events	1) The MoveTo feature handles moving the vita via Robot API calls. We give it a relative move distance and direction into the function as input. 2) If the vita is incapable of finding a path at any point during its movement, an error is thrown and the user is notified.
Postcondition	The vita has either moved to its destination or thrown an error.

Use Case:	Zoom in on identified object
Actors	Vita
Precondition	Objects are identified and highlighted with a bounding box, zoom option has been selected from dropdown menu, the camera is not already at max zoom
Flow of Events	1) System takes bounding box coordinates and uses those to determine how much zoom is appropriate (if relative 2) Robot API is used to tell Vita to zoom in on coordinates given
Postcondition	Robot has now zoomed in on the object

Use Case:	Recognize faces
Actors	people, Vita, openCV
Precondition	There are people around in the room and being highlighted with bounding boxes. Option to get more information selected.
Flow of Events	1) OpenCV runs a facial recognition inside the bounding box. 2) If there's matching face in database, an information tag containing name and job title shows on the bounding box (start with a fake database of our own faces) 3) If no matching is found, an unknown tag is displayed on the bounding box
Postcondition	Name and job title shows on the bounding box

Appendices:

- TensorFlow - Google's machine learning platform
- RP-Vita Robot API - InTouch Proprietary Robot interaction software
- OpenCV - open source computer vision library
- W.T.B. - Academic article with a procedure of creating large datasets from base images