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

- Studied deep Learning principles and TensorFlow
- Gathering relevant images for dataset
- Studied the Robot API - Unable to achieve in Sprint 1

Sprint 2 - Developing Prototype Components

- Selected InceptionV3 for classification DNN
- Sliding Window Classifier using OpenCV to do object detection
- “Where’s the Bear” approach to create large amounts of “fake” images
- Robot API - wrote tests for movement, zooming, centering on given points.

Sprint 3 - Developing prototype

- Two modules, classifier (given image, provides bounding boxes) and fake robot (and draws bounding boxes using info given by classifier)
- Dealing with only still images for now, and detection is limited to people

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. Identifying name or position of people - 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 employees).

Background

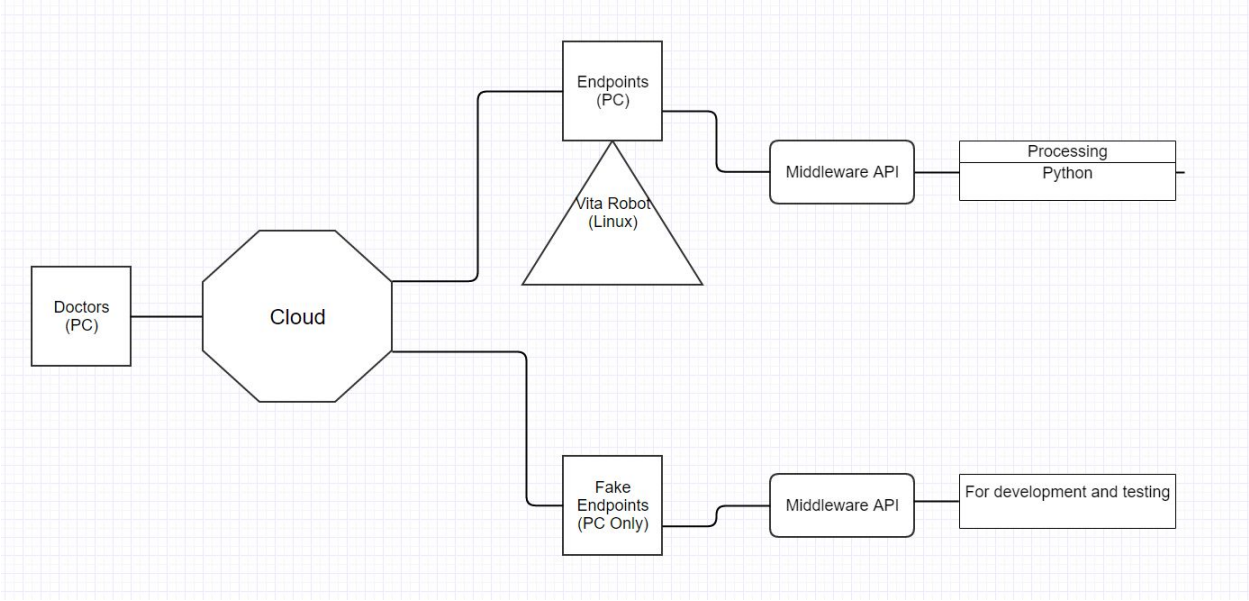
InTouch already offers a working remote care service via video stream to its customers, but we can expand upon the service by creating an augmented reality of the video stream. We will implement the A.R. interface using object detection in order to allow doctors to more easily access information and 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

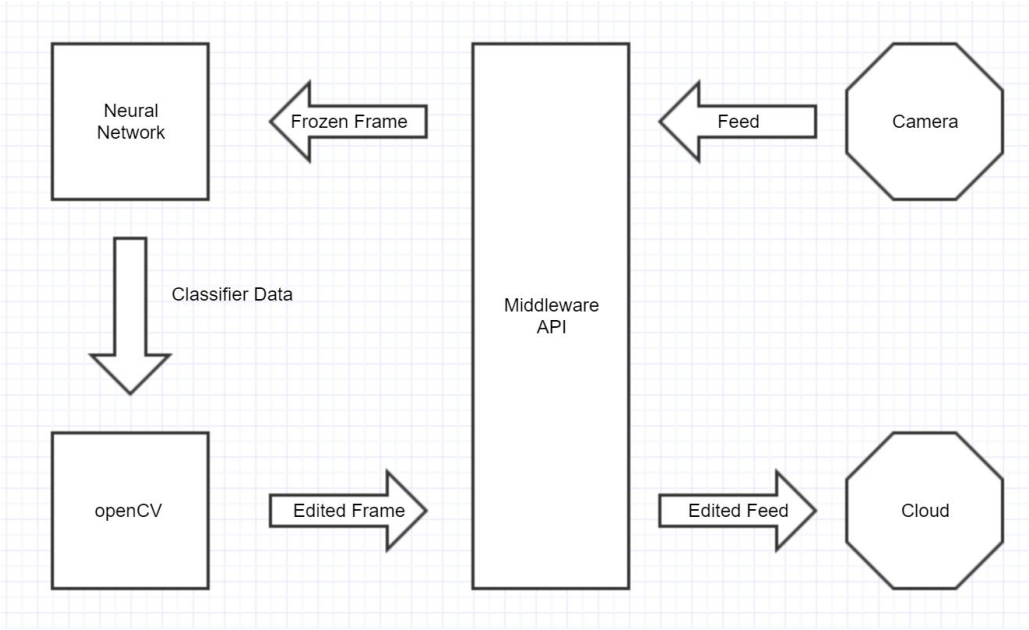
- We are given a medical robot (the Vita) with consistent internet connectivity. We are not responsible for a loss of wifi signal that would cause the “call” to drop.
- We are not responsible for the direct control of the robot’s movement: all commands will be handled through at least one layer of api calls.
- We will write a robot application that runs directly on the Vita’s Windows subsystem, and all object recognition on InTouch’s powerful desktop in order to ensure a minimal delay between object recognition, and augmented reality overlay.
- InTouch will implement an “AR mode” in their current client software to allow our classified/overlayed frame and click handling go through to the doctor.

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).

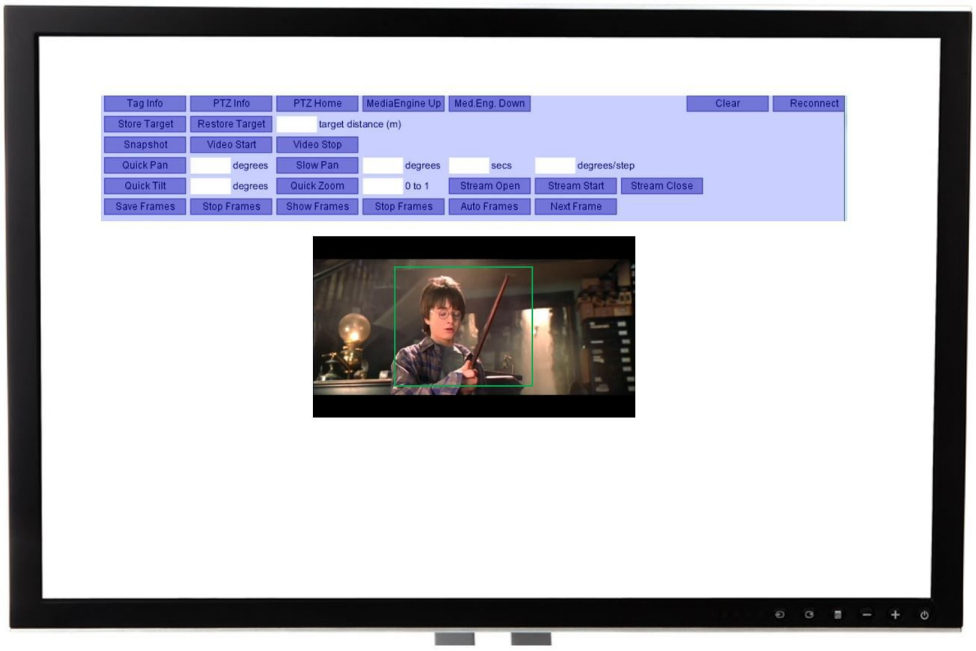


Within an endpoint, we will use our neural net in conjunction with Robot API and OpenCV to edit images (recognize objects, highlight them, U.I. buttons for context sensitive actions).

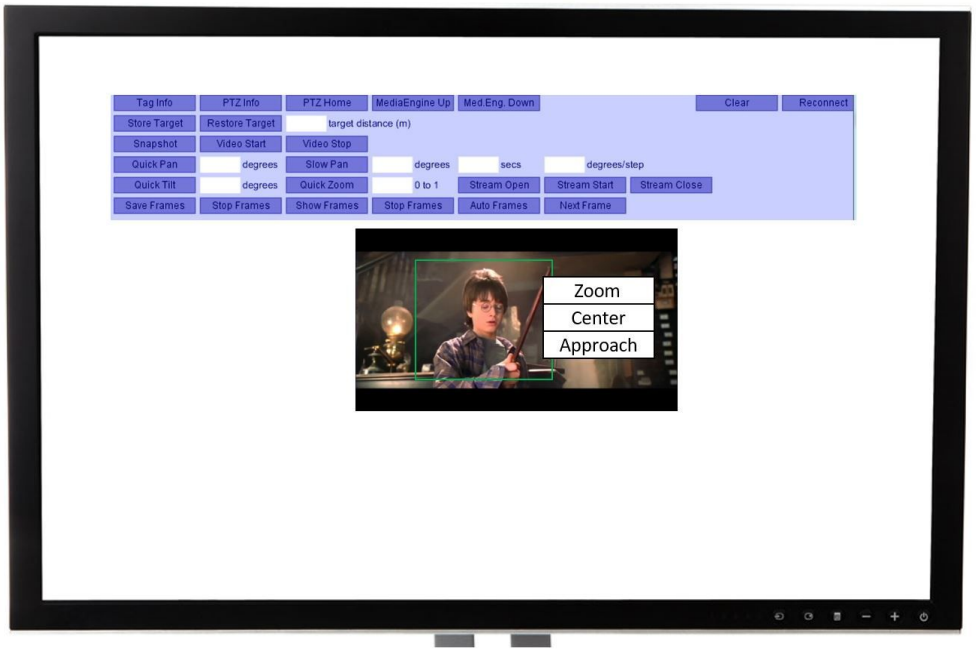


User Interaction

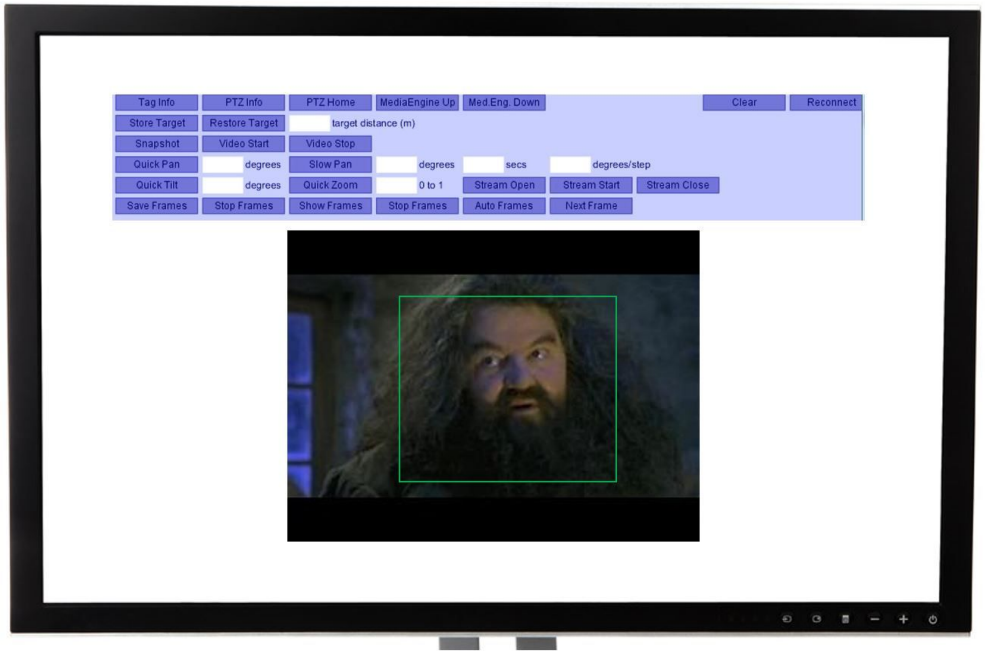
Our user interaction is limited to clicks upon objects and being able to display context sensitive actions. Here is a workflow for the prototype, where we augmented a Harry Potter trailer.
The trailer has been broken into frames and bounding box info is provided by the classifier. After loading up robot controller app, you will see first frame.



Now the user may choose to click on the bounding box and produce a set of buttons. For the prototype these actions will do nothing.



This process will continue, with each new frame image being classified and displayed. In the final iteration, we will extend our classification to a continuous video stream rather than still images, allowing live object detection. The action buttons will also be live.



Requirements

User stories

As a developer, I can create a training set for detecting faces only.
Acceptance criteria: Collected images of 200+ human faces
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/master/prototype/ClassificationModule/NeuralNetwork/trainingSet

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.).
Acceptance criteria: Given a set of base images and object images, W.T.B. processing yields a set of training images where the object images are overlaid onto the base images
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/master/createTrainingSet
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/master/resources

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.
Acceptance criteria: There is a sufficiently large set of training and testing images (use testing program on training directory) and verify that retraining provides 90% plus accuracy (program is written to print this when training is complete, currently 94% for faces only)
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/RobotDev/samples/inceptionv3/retrain.py
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/prototype/ClassificationModule/TrainingDirectoryTests.py

As a developer, I can load retrained neural network to classify new images
Acceptance criteria: Output graph can be loaded, and given an input image, classification label + confidence rating is printed
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/samples/inceptionv3/testingit.py

As a developer, I can take a relative position in an image and draw a bounding box about an object.
Acceptance criteria: Check that bounding box coordinates are given, and do not exceed image size. Use OpenCV to draw bounding box and display output for testing accuracy.
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/prototype/ClassificationModule/drawBox.py

As a developer, I can use robot api to snap images and save into a shared directory
Acceptance criteria: Still images from “robot” stored in a shared folder for classification.
Full Robot Module code
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/RobotDev/prototype/RobotModule

As a developer, I can detect objects using sliding window classification.

Acceptance criteria: Given an image, take subsets (window) to produce localized classification and approximate detection.
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/samples/objectDetection/slidingWindowClassifier.py

As a developer, I can read images from shared directory into classification module
Acceptance criteria: Images are read into a stack for classification and tagged to ensure they are not read multiple times.
Full classification module code:
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/RobotDev/prototype/ClassificationModule/classificationModule.py

As a developer, I can optimize sliding window by constraining the window size based on input image constraints (we will only have faces, where the face is close enough to be about ⅔ of image - bigger object is easier to detect)
Acceptance criteria: Sliding window is run on a 640 x 480 size image and bounding box is produced around the face. Multiple boxes, or oversized/undersized boxes are not produced.

As a developer, I can write bounding box info from sliding window classifier into a text file.
Acceptance criteria: Bounding box info (classification label and coordinates) stored using JSON format in a shared directory for robot module.
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/prototype/ClassificationModule/boxesToJson.py

As a developer, I can read bounding box info into a local data structure for click handling and for drawing boxes on output images
Acceptance criteria: JSON files read from shared directory and used to update data structure
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/RobotDev/prototype/RobotModule/parseJSONFunc

As a developer, I can implement a thread-safe data structure that stores labels and bounding boxes
Acceptance criteria: D.S. is updated on each new file read and can be searched when a button click is received or the output image is about to draw new bounding boxes
https://github.ucsb.edu/nathaniel-pincus/cARe/tree/RobotDev/prototype/RobotModule/boundingBoxCollectionFile

As a developer, I can implement a “zoom” feature, that performs a zoom of a given a zoom amount and coordinates.
Acceptance criteria: Simply print a message that shows we’ve called zoom on a given object (API cannot actually zoom on a fake endpoint, real vita will zoom in on object)
Code for robot actions

https://github.ucsb.edu/nathaniel-pincus/cARe/blob/RobotDev/prototype/RobotModule/RobotActions.pde

As a developer, I can implement a “move” feature, that moves a specified amount in a specified direction.
Acceptance criteria: Simply print a message that shows we’ve called move on a given object (real vita will move towards specified object in 1m chunks until it senses something)

As a developer, I can implement a “center” feature, that moves the camera to a specified bounding box.
Acceptance criteria: Simply print a message that shows us we’ve called center on a given set of coordinates (real vita will tilt towards)

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.
Acceptance criteria: Clicking within a bounding box produces action buttons
All GUI code
https://github.ucsb.edu/nathaniel-pincus/cARe/blob/RobotDev/prototype/RobotModule/GUI_Control.pde

As a user, I can click on the “zoom” button so that I can see the object of interest more clearly.
Acceptance criteria: When zoom button is pressed, zoom in on specified object (prototype will do nothing, only console message)

As a user, I can click on a “move” button that when called, triggers the robot to move towards an object of interest.
Acceptance criteria: When move button is pressed, move towards specified object (prototype will do nothing, only console message)

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.
Acceptance criteria: When center button is pressed, center in on specified object (prototype will do nothing, only console message)

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.
Acceptance criteria: When button is pressed, if the face is within database, results will be displayed (not implemented for prototype)

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
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/commit/6575b1a111e8daeb0caab460c9954c079995398f

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.
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/blob/RobotDev/samples/inceptionv3/retrain.py

Use Case:	Recognize and highlight objects (sliding window classifier)
Actors	Vita, Robot API, openCV
Precondition	Have a trained neural net to identify objects under different conditions and backgrounds.
Flow of Events	1) Neural network is given an image 2) Image slice is classified 3) If objects of interest are detected, save that window 4) All windows and labels are written into text file
Postcondition	Objects of interest are recognized and their bounding boxes are saved
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/samples/objectDetection/slidingWindowClassifier.py

Use Case:	Create context sensitive actions menu
Actors	User, openCV
Precondition	Bounding boxes have been drawn, object classified
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
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/pull/20/commits/262353a4b70af86c14f93ae2a43cf72004682195 https://github.ucsb.edu/nathaniel-pincus/cARe/pull/20/commits/0eb046e52cae7977e360b3201233f3a2b3c11d98

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
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/pull/20/commits/262353a4b70af86c14f93ae2a43cf72004682195

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
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/commit/bd628af8eb59d01c75af979e42529c5c4c5d4446

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
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/blob/master/samples/objectDetection/slidingWindowClassifier.py

Use Case:	Draw bounding box
Actors	Vita, OpenCV
Precondition	Video stream has started, objects have been detected and the bounding box coordinates have been saved to a text file and stored in a data structure
Flow of Events	1) A check is done for bounding box info in data structure 2) The provided info is used to give constraints for bounding box 3) OpenCV draws box on corresponding image 5) New image is displayed
Postcondition	A new image with bounding box drawn on is displayed
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/commit/1e53aa16341cd7f7ee8855eb0e4f978d5dcf02e6

Use Case:	Save frames periodically
Actors	Vita, RobotAPI
Precondition	Stream has been started
Flow of Events	1) A rate is specified 2) Vita takes image at specified rate 2) Image is saved to shared folder for classification
Postcondition	An image has been saved to the shared folder
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/pull/20/commits/512446552d04dba0b85b3288a57aaa0b33208dd2

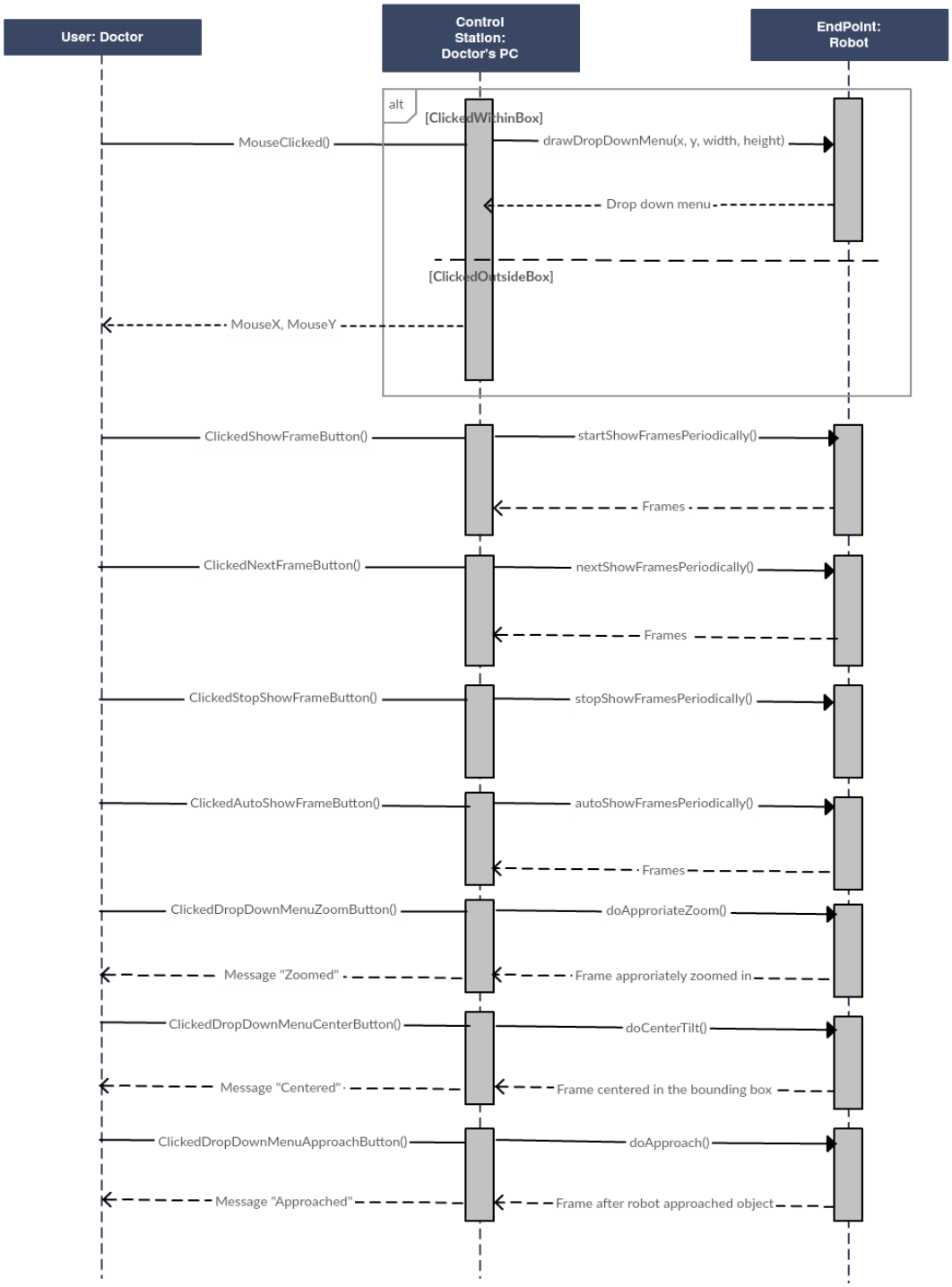
Use Case:	Store bounding box info
Actors	Vita
Precondition	Bounding box info exists in a text file
Flow of Events	1) A check is done for bounding box info in a text file 2) The information for the picture is read and stored in a data structure 3) Folder cleanup is initiated 4) The newly filled structure is returned
Postcondition	Data structure with all bounding boxes for a particular image is saved
Github commits	https://github.ucsb.edu/nathaniel-pincus/cARe/pull/20/commits/55b44586c3122589e8633782d1afd022de468aca https://github.ucsb.edu/nathaniel-pincus/cARe/commit/1e53aa16341cd7f7ee8855eb0e4f978d5dcf02e6

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

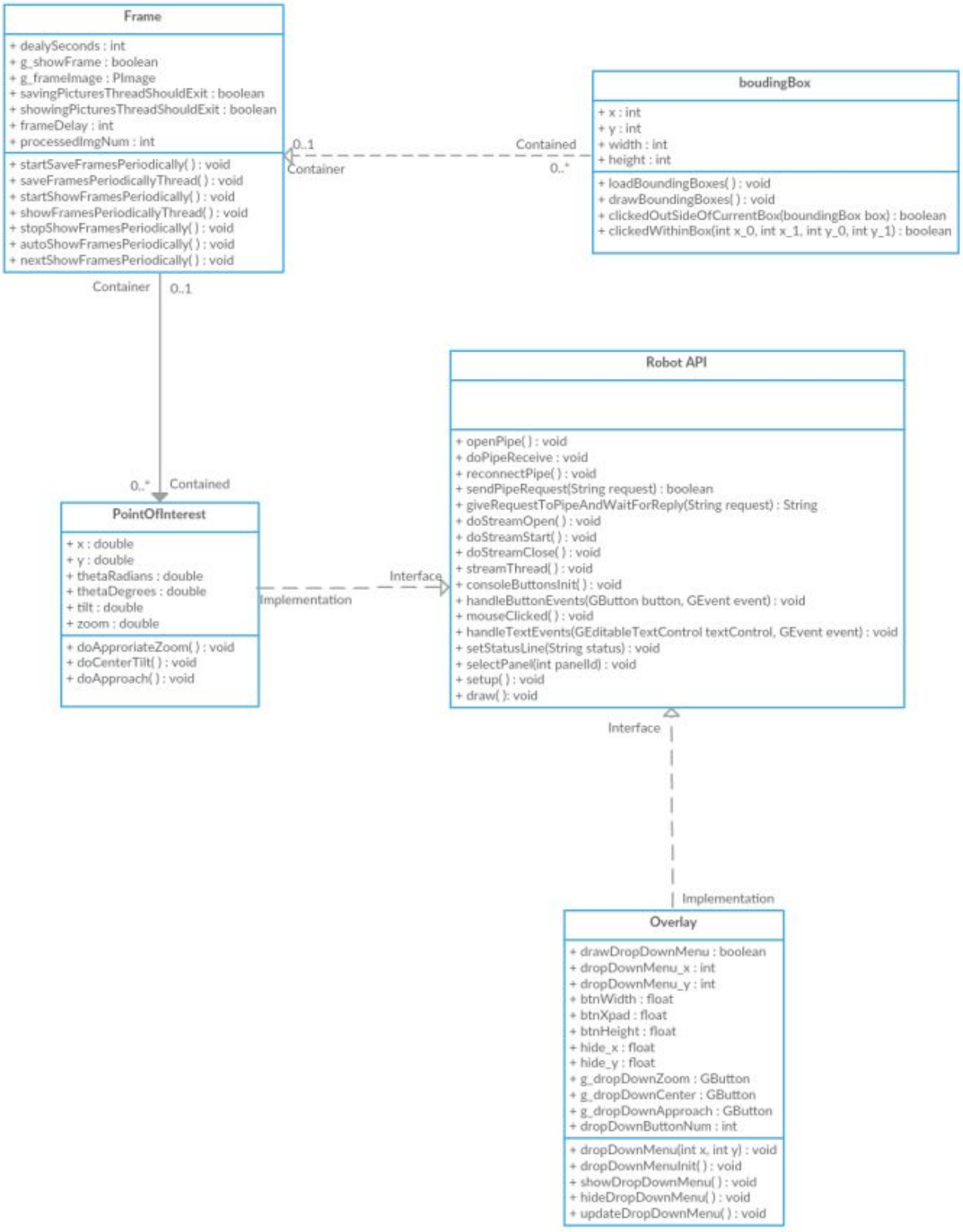
Github commits	void doCenter(){
	int midpoint = g_streamImageWidth;
	float degrees = mouseX / midpoint;
	giveRequestToPipeAndWaitForReply("<ptz> <tilt>" + degrees + "<delta> </ptz>");
	setStatusLine("");
	}

System Models

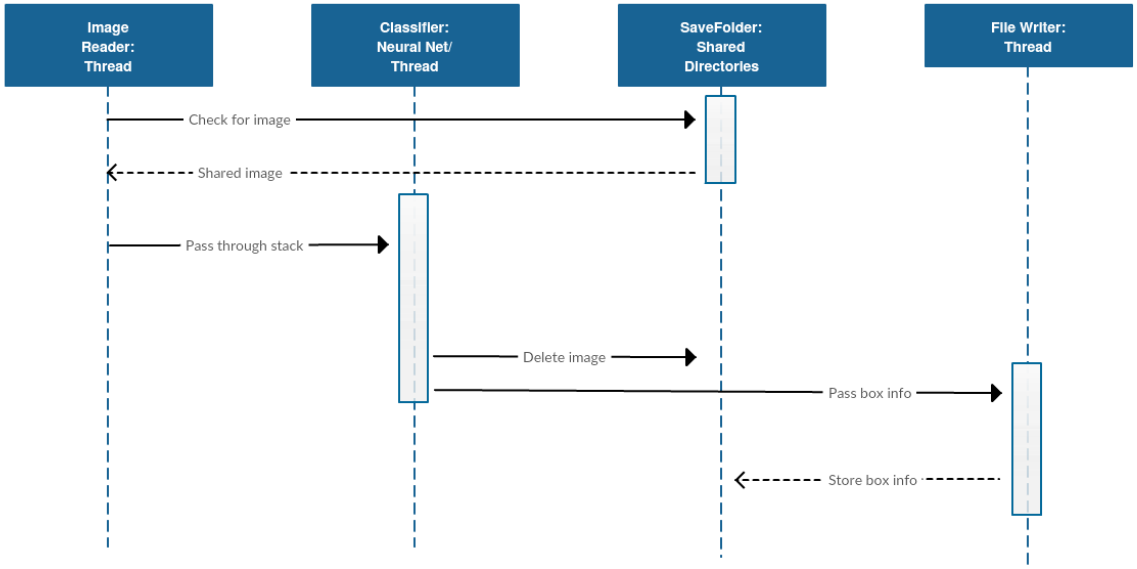
Robot application sequencing diagram:



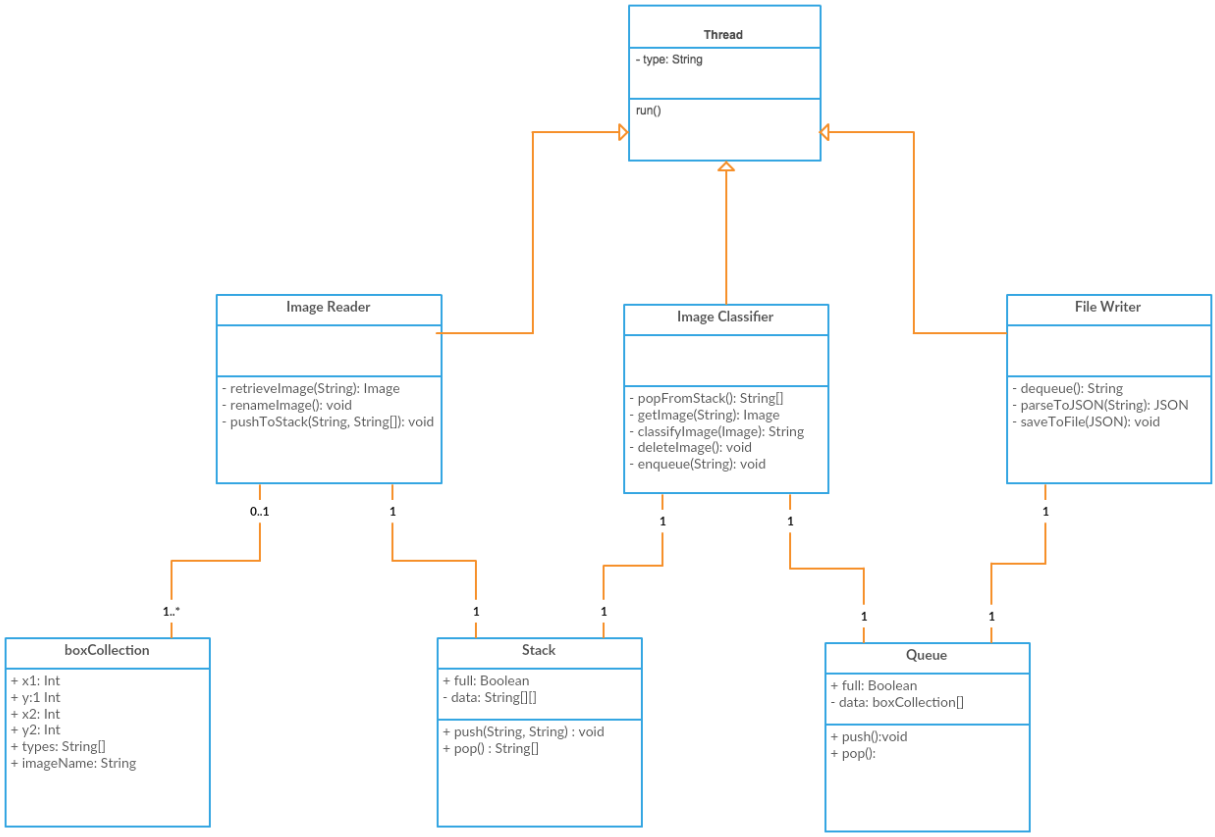
Robot application class diagram:



Sequencing Diagram for Classification Module



Class Diagram for Classification Module



Appendices:

- TensorFlow - Google's machine learning platform
- 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
  OpenCV - open source computer vision library