Vision System for MediBot

Shon Cortes
Sameer Pusegaonkar

Quad Chart

Vision System for MediBot
Sameer Pusegaonkar, Shon Cortes

OVERVIEW

MediBot is a product by Acme Robotics that is used in hospitals to deliver medicines.

Due to the rise of the number of patients, doctors, nurses in hospitals medibot needs a software module which can detect and track humans inorder to avoid them

This software module will be a computer vision program that will run on MediBot's microprocessor.

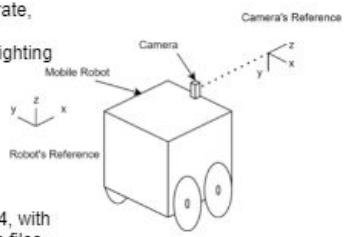
HOW?

Assumptions about the camera:

- A monocular camera
- Known Field of View, focal length, frame rate, position wrt the robot frame
- Camera will not be affected by unknown lighting conditions

Assumptions about the robot:

- Robot will always travel in a hospital environment in hallways
- Known physical dimensions
- Robot microprocessor uses Ubuntu 18.04, with all OpenCV, DLib installed & configuration files for MobileNet



APPROACH

We will be utilizing MobileNet for detecting humans & then relying on the co-relation filter from the DLib library. Git will be used for version control.
GTest will be used for testing.

This approach is significantly better than other perception techniques as it relies on a camera instead of lidar. To add, lidar sensors are expensive as compared to cameras.

This technique is more accurate as it relies on a pre-trained model which we can directly incorporate & use it on MediBot.

WHEN?

A well-documented & fully tested software module will be received by Acme Robotics in the span of 3 weeks that can detect and track humans while returning their position in the robot frame.

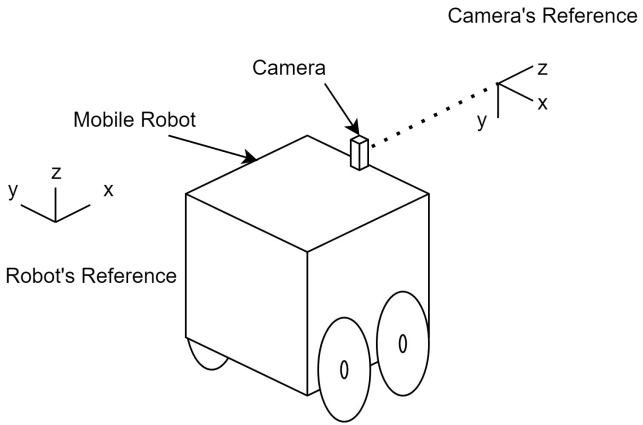
This includes:
Source Code
Documentation - Doxygen documentation generation
UML Diagrams - Class & Activity Diagrams
README with proof of code coverage(90+%) and unit test passing

Project Overview

- Assist in the development of the **vision system** for the MediBot platform.

- Purpose is to **navigate hospital setting** and **deliver medical supplies** without being obstructed by any humans.

- Vision system needs to **detect and track humans** then **return their position** in the robot's coordinate frame in real time.



Deliverables

- A well-documented & fully tested software module that can detect and track humans while returning their position in the robot frame.
- This includes:

  Source Code

  Documentation - Doxygen documentation generation

  UML Diagrams - Class & Activity Diagrams

  README with proof of code coverage(90+%) and unit test passing

Assumptions and Constraints

Assumptions about the camera:

- A monocular camera
- Known distortion parameters, Field of View, focal length, frame rate, position wrt the robot frame
- Camera will not be affected by unknown lighting conditions

Assumptions about the robot:

- Robot will always travel in a hospital environment in hallways
- Known physical dimensions
- Robot microprocessor uses Ubuntu 18.04, with all OpenCV, DLib installed, MobileNets Neural Network, C++, x64 Architecture

Project Organization

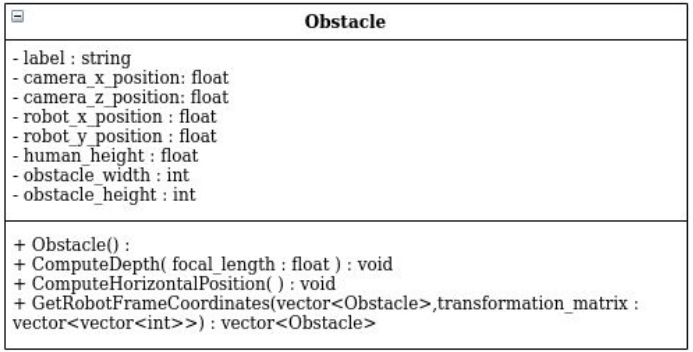
- UML diagrams : **flow of program, structure of the entire software, provide initial product backlog**

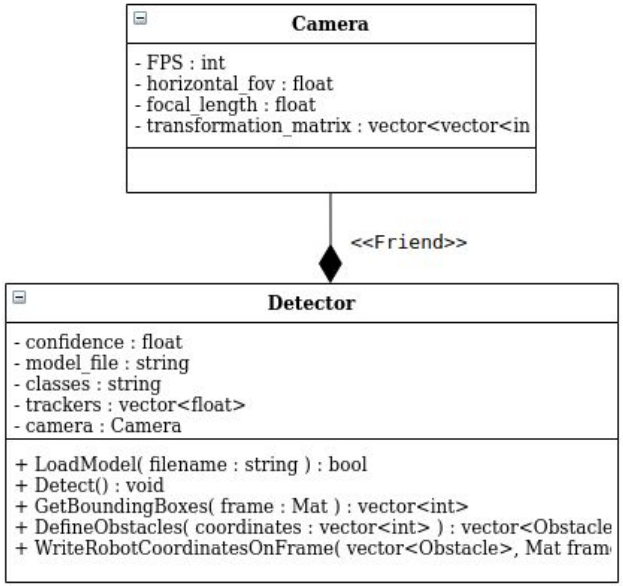
- Agile:
  - **3x One week sprints** - daily scrum meetings and an iteration review
  - Azure Devops - Product Backlog, Sprint Progress, Bug Fixes, and Test Plans.
  - Testing - V Model

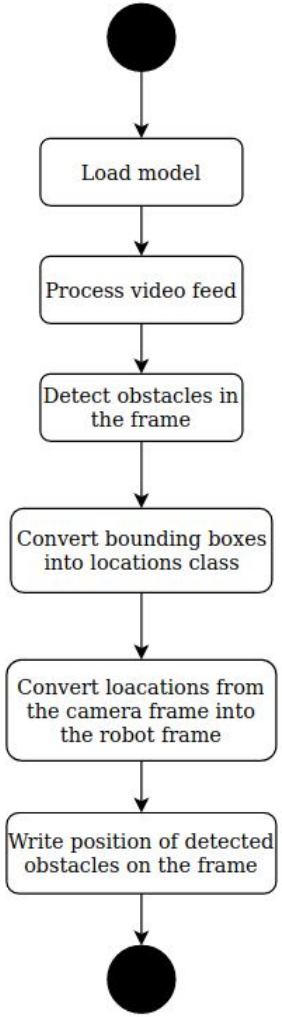
- Work will be divided by both team members, **scrum meetings** will be an opportunity to address sticking points.
  - Pair programming will be done as needed.

- Time budget is 10 hours a week per team member.

UML Diagrams







Initial Product Backlog

- Initial time estimate of approximately 60 work hours

ID	Title	Work Item Type	State	Original Estimate	Value Area	Iteration Path	Tags
33	Detection Model Testing	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
34	Obtain testing data for human detection.	Task	New	3		ENPM808X - Midterm\Sprint 1	
35	Clean testing data.	Task	New	4		ENPM808X - Midterm\Sprint 1	
27	Unit Testing	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
28	Define tests for the Obstacle class method for converting positions into the robot frame.	Task	New	2		ENPM808X - Midterm\Sprint 1	
29	Define tests for the Obstacle class method for converting from bounding boxes to positions in the camera frame.	Task	New	2		ENPM808X - Midterm\Sprint 1	
30	Write test for the Detector Video feed method.	Task	New	2		ENPM808X - Midterm\Sprint 1	
31	Write tests for the Detector class Get bounding box method	Task	New	2		ENPM808X - Midterm\Sprint 1	
32	Write tests to confirm accuracy of human detection model.	Task	New	2		ENPM808X - Midterm\Sprint 1	
25	Repository	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
26	Write README.md	Task	New	4		ENPM808X - Midterm\Sprint 1	
36	Set up Travis and coveralls unit testing	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
37	Generate Deoxygen files	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
12	Camera Class	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
18	Define Camera attributes.	Task	New	2		ENPM808X - Midterm\Sprint 1	
19	Get initialization values for all camera attributes.	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
20	Initialize all camera attributes with actual camera parameters.	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
11	Detector Class	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
21	Define Detector attributes.	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
22	Write method to load model for object detection.	Task	New	3		ENPM808X - Midterm\Sprint 1	
23	Write method for processing video files.	Task	New	4		ENPM808X - Midterm\Sprint 1	
24	Write method for detecting the bounding boxes using dlib.	Task	New	3		ENPM808X - Midterm\Sprint 1	
10	Obstacle Class	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
13	Define Obstacle attributes	Task	New	0.5		ENPM808X - Midterm\Sprint 1	
17	Define position of camera with respect to robot's cordinate frame.	Task	New	1		ENPM808X - Midterm\Sprint 1	
14	Define camera to robot transformation matrix.	Task	New	2		ENPM808X - Midterm\Sprint 1	
16	Create method to convert positions form camera frame to robot frame.	Task	New	4		ENPM808X - Midterm\Sprint 1	
15	Create method to convert from bounding boxes to camera frame.	Task	New	4		ENPM808X - Midterm\Sprint 1	
7	Proposal	User Story	New		Business	ENPM808X - Midterm\Sprint 1	
2	Film Midterm Proposal Video	Task	New	2		ENPM808X - Midterm\Sprint 1	
3	Midterm Proposal Paper	Task	New	8		ENPM808X - Midterm\Sprint 1	
8	Bugs	User Story	New		Business	ENPM808X - Midterm\Sprint 2	

Risk Management

- Low Accuracy of Model- Move to a YOLO which has a higher accuracy or switch to a color tracker/template matching algorithm.
- Hardware Limitations - Switch to a smaller model with fewer parameters
- Avoid work on outdated code - Git for Software version control
- Improper implantation of a critical code module - Unit testing

Methods, Tools, Techniques

- MobileNet, Correlation filter, Mapping, Transformation Matrix

- Google Test - Unit tests
- Cpplint - Google Style Sheet
- Code coverage - Coveralls.io

- Doxygen - Documentation

- OpenCV - Support Function
- Dlib - Support Function

What we are using: Reference Materials

[1] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications - https://arxiv.org/abs/1704.04861

[2] C++: https://isocpp.org/

[3] x64 Architecture: https://en.wikipedia.org/wiki/X86-64

[4] OpenCV: https://github.com/opencv/opencv

[5] DLib: http://dlib.net/imaging.html

Definitions and Acronyms

- MediBot: A 4 wheeled medical mobile robot developed by Acme Robotics.
- FOV: Field of view of the camera in degrees
- UML: Unified Modeling Language
- MobileNet: A machine learning model used to get the bounding box for a detected object.
- OpenCV: An open source computer vision library.
- DLib: An open source, general purpose library used for image processing, machine learning, linear algebra, and more.

Monitoring and Controlling Mechanisms

- Unit testing - Google Test
- Git - Software version control
- Azure devops - Agile Project Management