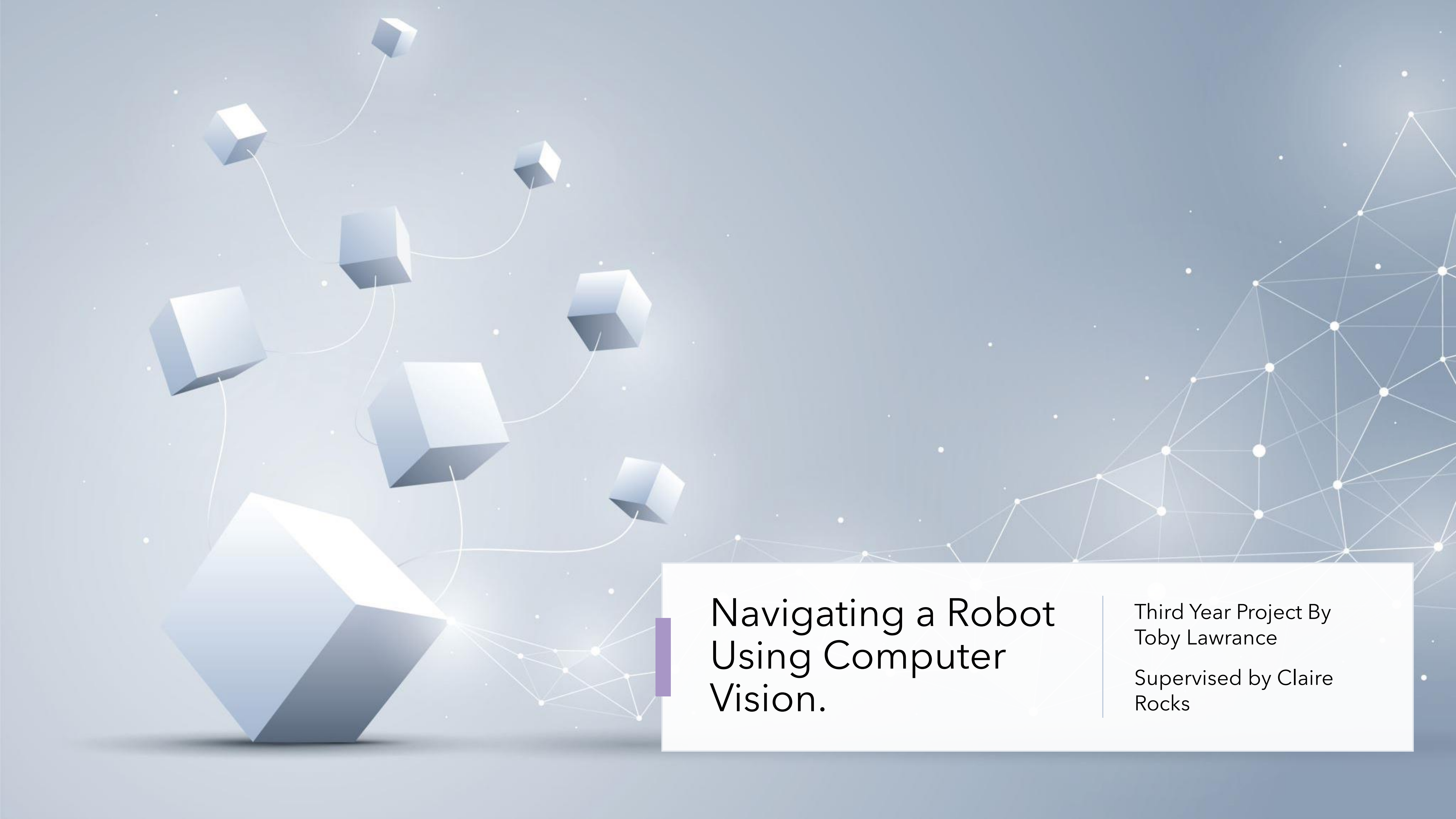


Navigating a Robot Using Computer Vision.

Third Year Project By Toby Lawrance

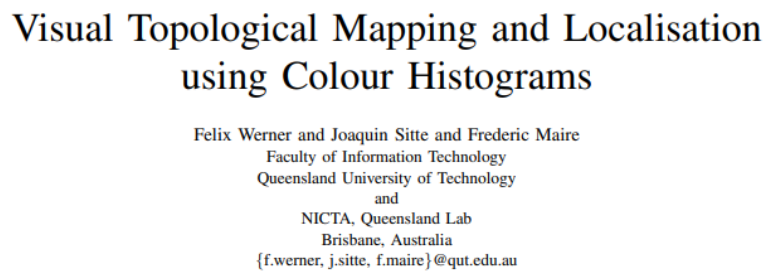
Supervised by Claire Rocks

Introduction – The problem

- Robot navigation typically relies on a broad array of sensors.
- Obstacle detection usually requires rangefinders
  - High quality rangefinders are often very expensive, e.g. LiDAR
- Cameras are typically a significantly cheaper sensor, and are often mounted on mobile robots for logging purposes.
- To navigate, the robot needs to identify obstacles

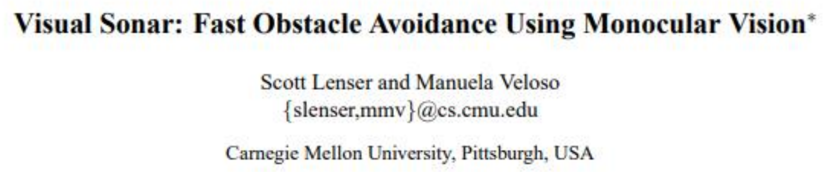
Academic context

- The papers found typically relied on colour segmentation of the image.
- Colour histograms interact poorly with noisy backgrounds
- Other approaches often used stereo cameras to achieve depth perception
- Having attempted colour segmentation, histograms could be used in a different way to identify regions.



Visual Topological Mapping and Localisation using Colour Histograms

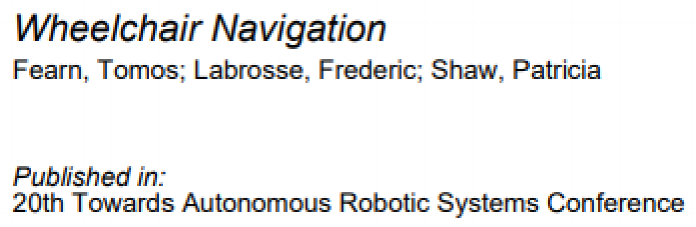
Felix Werner and Joaquin Sitte and Frederic Maire
Faculty of Information Technology
Queensland University of Technology
and
NICTA, Queensland Lab
Brisbane, Australia
{f.werner, j.sitte, f.maire}@qut.edu.au



Visual Sonar: Fast Obstacle Avoidance Using Monocular Vision*

Scott Lenser and Manuela Veloso
{slenser,mmv}@cs.cmu.edu

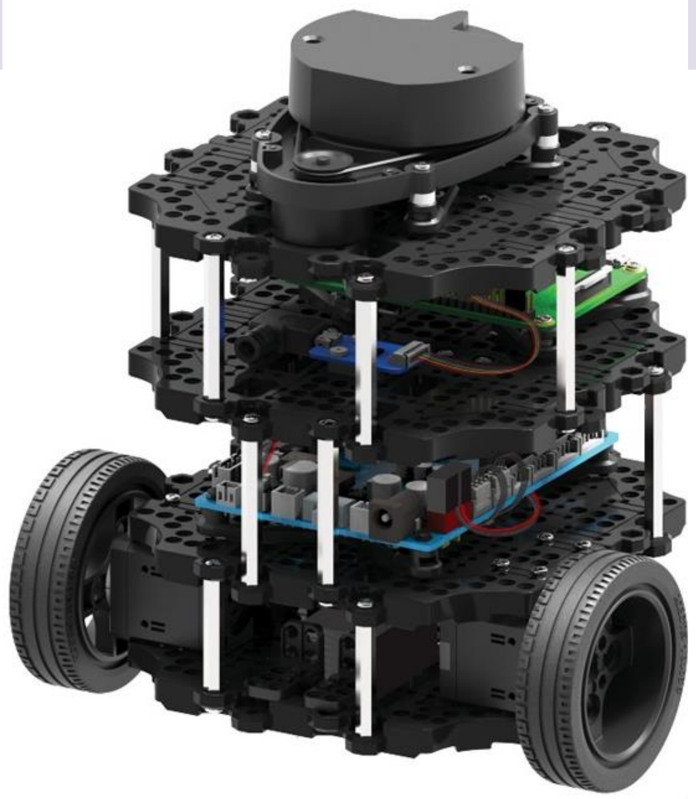
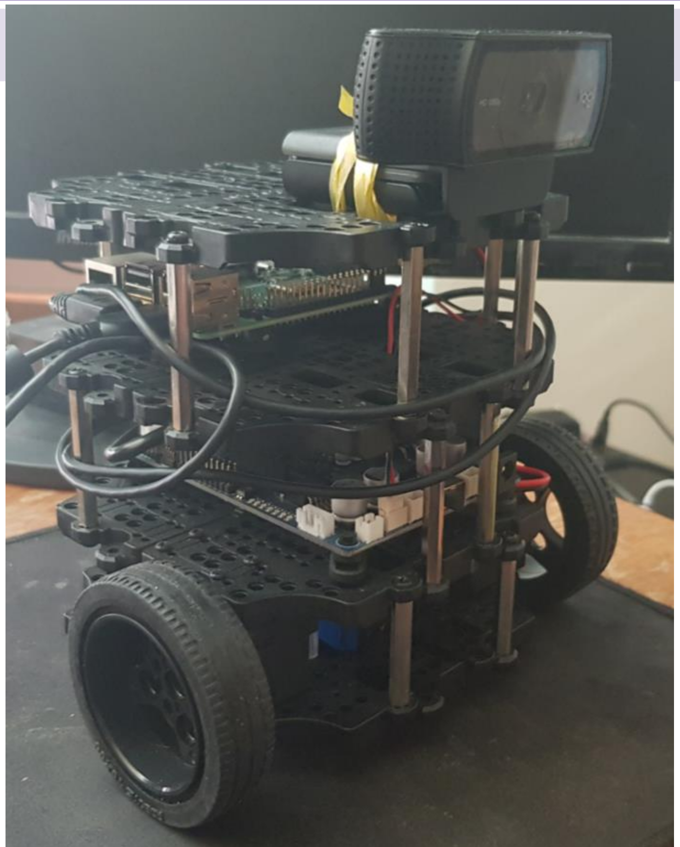
Carnegie Mellon University, Pittsburgh, USA



Wheelchair Navigation
Fearn, Tomos; Labrosse, Frederic; Shaw, Patricia

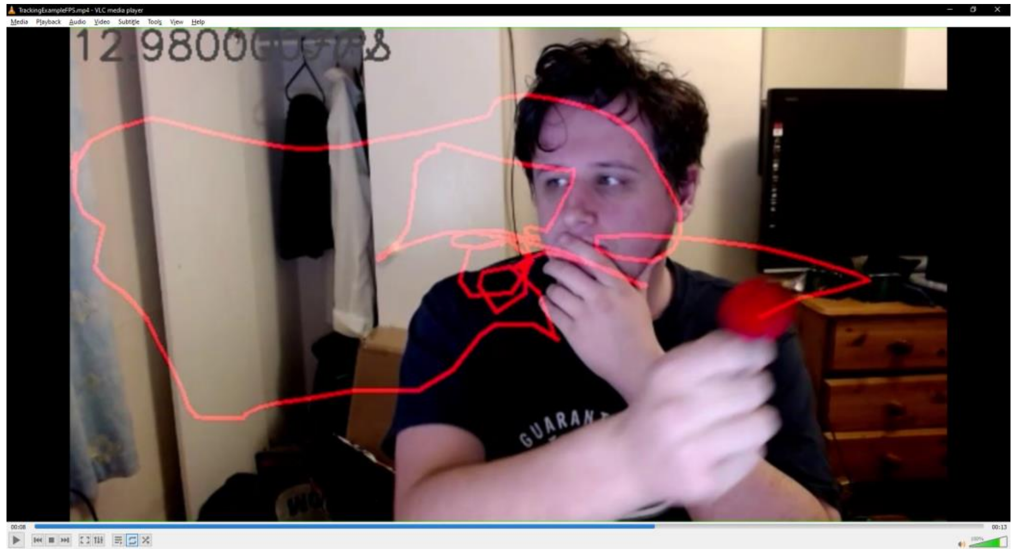
Published in:
20th Towards Autonomous Robotic Systems Conference

The hardware

Turtlebot3 by Robotis	Logitech C920 Webcam	Robot + Webcam + Ribbon
		

Early Approaches

- Colour Segmentation of Image
  - Suffered heavily from noise, Assumptions required of obstacle colour
- Feature detectors
  - Obstacle corners not “interesting” enough.
- Optical Flow
  - Similar issue to Feature detectors
- Corner Detection
  - Found the obstacle, but also everything else. Too much noise to filter.



ROS – Robot Operating System

- The Turtlebot3 Runs on ROS, ROS2 is the newest setup.
- I spent a long time getting ROS to work, it has some very specific requirements.
- I ended up using ROS2, the Turtlebot3 has an Ubuntu 18.04 Server Image on it and the paired Laptop is running Ubuntu 18.04.
- Doing so through a virtual machine has proved very tricky with no success thus far.

How ROS 2 works

- Executable programs in ROS 2 are called Nodes, nodes belong to packages and a single package may have multiple nodes
- Nodes communicate with each other by way of message passing.
- Nodes can communicate intra-computer or inter-computer with no visible change in mechanism
- The messages are passed using channels called “Topics” with publishers and subscribers.
- The Turtlebot3 subscribes to many topics, most importantly /cmd_vel which is the primary way of enacting movement. In the form of m/s velocity, linear and angular.

Methodology

- Instead of colour segmenting, we use channel filtering to produce a grayscale image where strongly coloured pixels are bright, whilst plainly coloured pixels are not.
- From this we produce horizontal and vertical histograms of the image. Calculating average pixel value
- From this we calculate points of high change, attempting to section an area of the image as interesting.
- Exponential Filtering on the histogram reduces noise

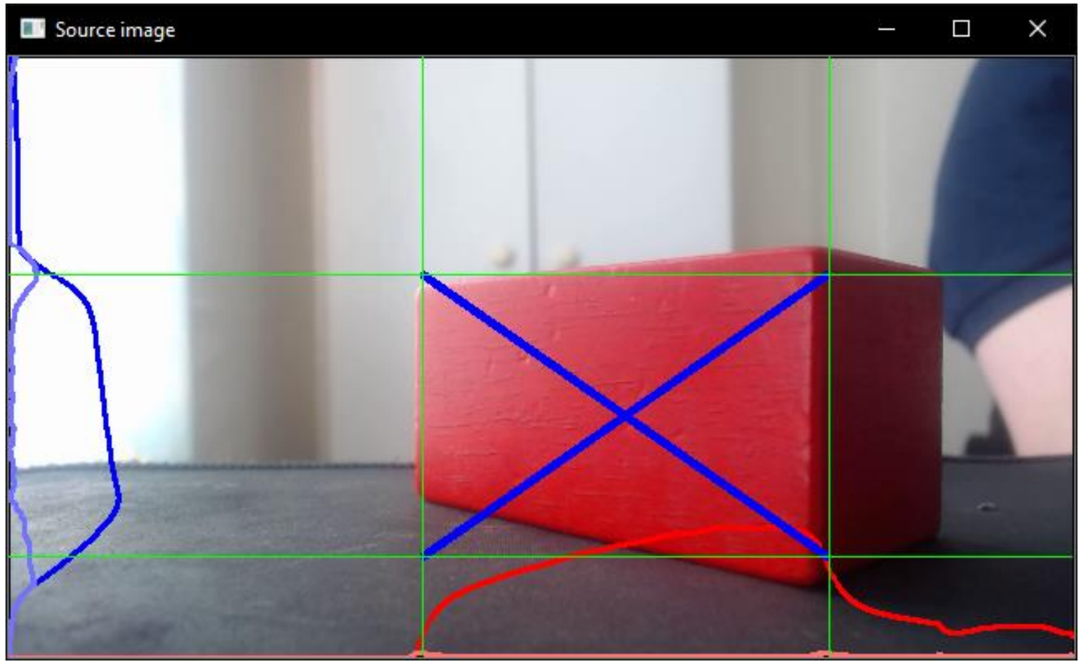


Image Processing – Preparation

For each frame, a number of steps are gone through:

1. Add a small border to enable edge clipping obstacle detection
2. Apply channel filter. (Max of R,G and B for each pixel, each minus their other two colours)
3. Threshold the values then mask the channel filtered image.
4. Use MeansDenoising to reduce artifacts.

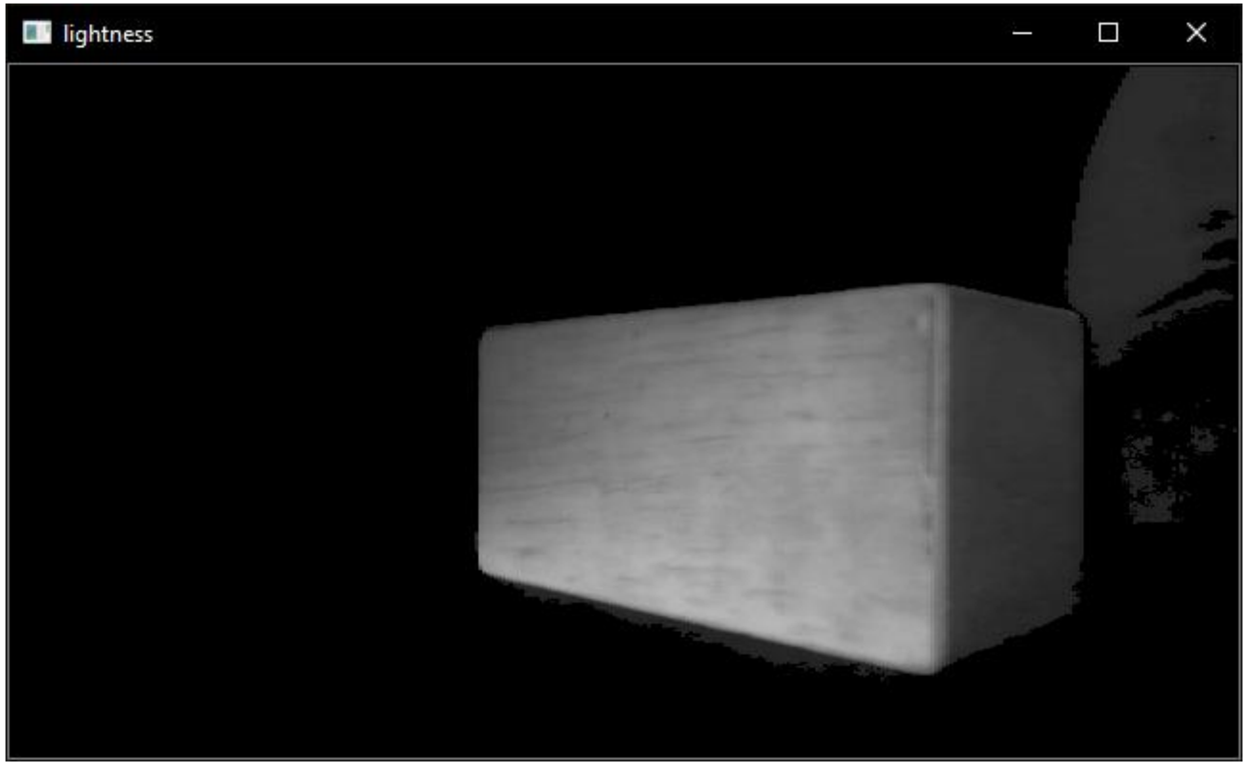


Image Processing – Possible Objects

1. Calculate the Row/Column Histograms
2. Apply exponential filter to each histogram
3. Calculate change in values
4. Calculate the threshold value
5. Identify trigger points
6. Remove adjacency in trigger points to reduce to a single point
7. Calculate row/column intersects and identify all possible rectangles above a certain size.
8. If the number of rectangles is too high, go to step 1 with a higher threshold.

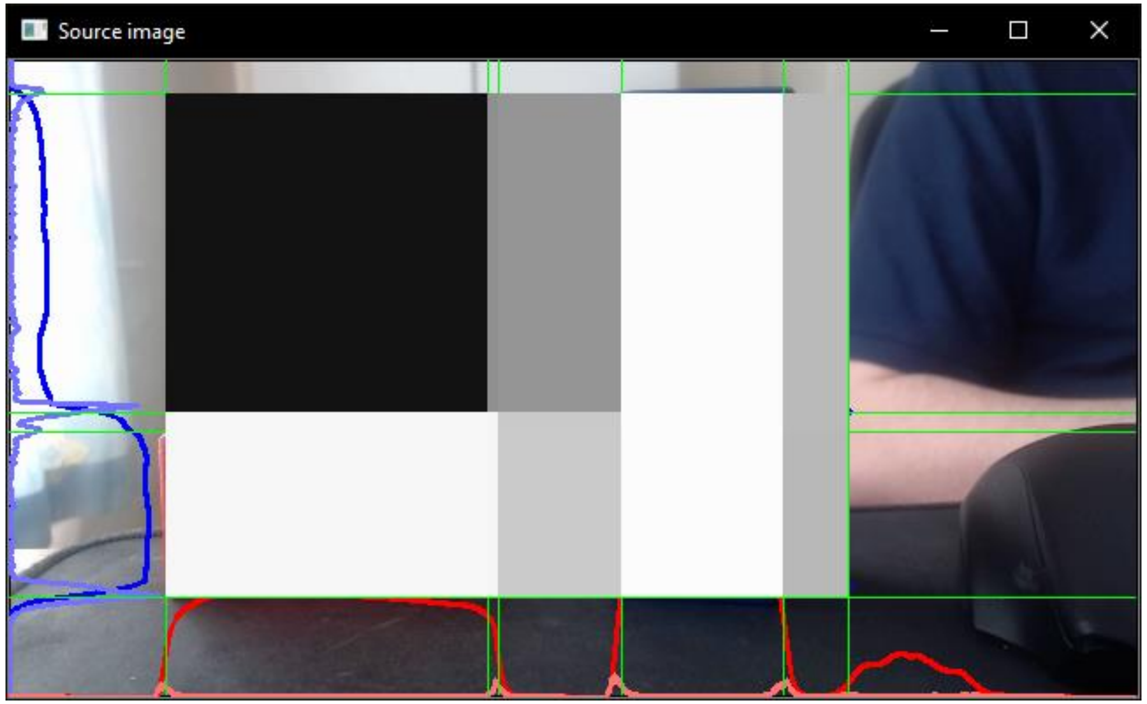
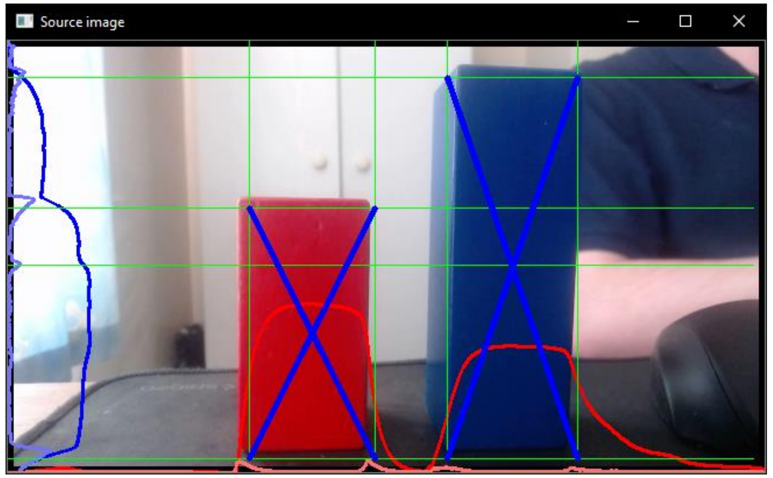
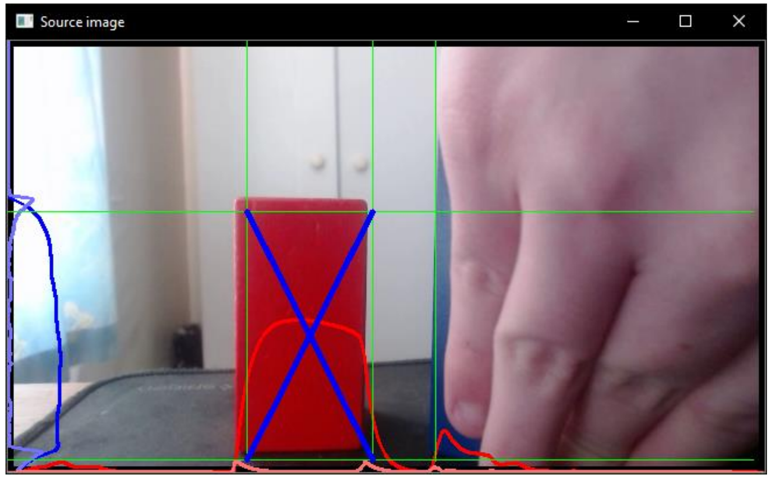


Image Processing – Object detection

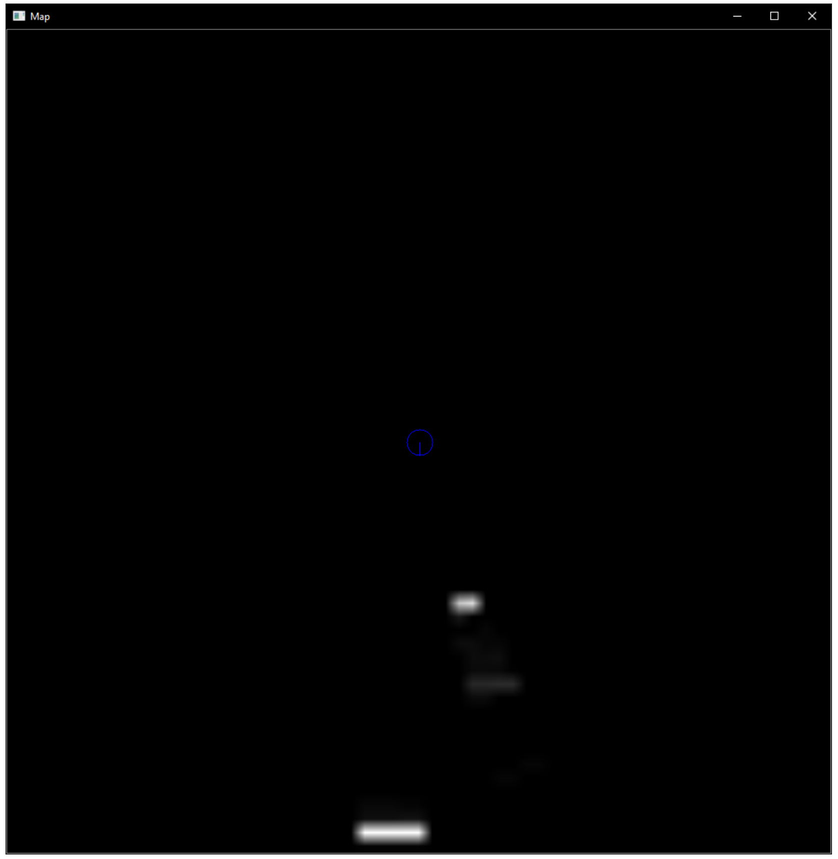
1. Sort the objects by size
2. Compute average pixel values for each rectangle, filtering out values below 50% of max value.
3. Sort objects in place by average value
4. Remove overlaps in the rectangles, favouring brighter and larger rectangles
5. Use known proportionate size of objects to estimate depth
6. Add measurement to the map





Mapping

- The map is represented as a matrix, with each value representing 1cm^2. The value in the pixel represents a belief in the representative grid square containing an obstacle.
- The rate of increasing belief is non-linear.
  - First detection: 1, Second: 10, Third: 100, Fourth 255 (Cap)
- These are drawn onto the map using Bresenham's line algorithm as reimplemented by me.
- Only the front face of the object is drawn, no assumptions are made about the depth of the object.
- The object is assumed to be 10% wider and thus slightly closer, than it is measured to be. This aids in reducing clipping when pathing.



Navigation

- Movement for the robot is given in the form of Linear and Angular velocity.
- The odometry on the robot is very good. Accurate and prone to minimal drift.
- Several navigators have been implemented to demonstrate functionality.
  1. Random explorer: Avoids obstacles but picks new directions randomly.
  2. Simple Mapper: Avoids obstacles, attempts to frequently improve its understanding of the environment
  3. Simple Pather: Given a goal point, attempts to path to that point, updating the path during travel to avoid collisions.
- The pathing has been done by re-implementing the A* Algorithm to work with my mapping system, then reducing the path to a set of waypoints where the direction changes.

Results

Play the View video, Map video and overhead

Issues

- Due to the channel filtering, the image processing is absolutely terrible at recognising walls or other typical background bounds.
- The image processing is a resource intensive process as implemented. 1FPS is the current max framerate when running on the robot itself.
- The mapping system is relatively crude and could be improved upon.
- Currently depth estimation relies on an understanding of the detected objects size. This could be done using odometric techniques, but requires estimation of tracking objects between frames. Time has not allowed for that foray.
- My navigation setup is relatively minimal, mostly just proof of concept

Project Management

- I made a set of objectives
  - The objectives to achieve shifted over time
  - As the problem got more clearly defined and the area of research became more clear, some of those objectives became less useful
- Regular meetings helped steer the direction of research
- Focus shifted over time towards the image recognition algorithm
- Navigating the robot from the information provided isn't novel and as such lost significance in the project

Conclusions

- Cameras are an effective sensor tool that can be leveraged to aid in navigation.
- Due to the volume of data involved in image analysis, the approach can be quite resource intensive.
- Distinguishing factors of obstacles can be an unreasonable requirement when used in real-world systems.
- The system however works under a set of assumptions.