Project Notes

Project Title: Simulation of an Indoor Drone for Assisting Police in Mass Shootings
Name: Rumaisa Abdulhai

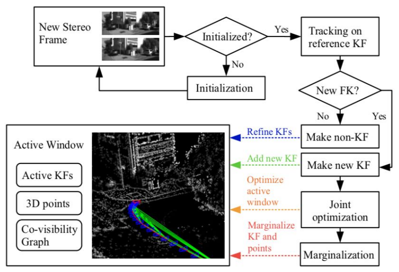
Table of Contents:

Knowledge Gaps

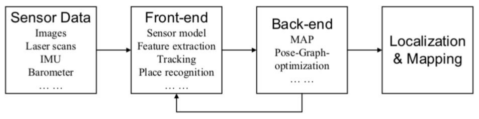
This list provides a brief overview of the major knowledge gaps for this project, how they were resolved and where to find the information.

Knowledge Gap	**Information is located**	**Date resolved**
Simultaneous Localization & Mapping	#2, #13	10/19
Gazebo Simulator with ROS	#5,#12, #18-21	11/19
Probabilistic Robotics Concepts	#22-25	12/19
Path Planning Methods	#26-28, #30, #34-36, #39	12/19
Localization Methods	#33	12/19

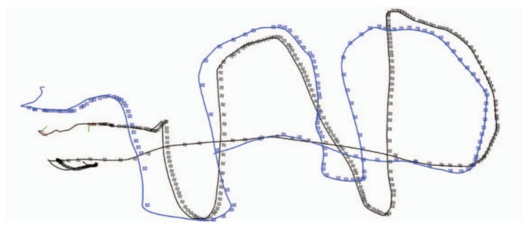
Article #1: Stereo Direct Sparse Odometry

Source Title	Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras
Source Author	Rui Wang, Martin Schwörer, Daniel Cremers
Source citation	Wang, R., Schwörer, M., & Cremers, D. (2017). *Stereo Dso: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras*. *Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras* (pp. 1–9).
Original URL	Stereo DSO: Large-Scale Direct Sparse Visual Odometry with ...https://vision.in.tum.de › _media › spezial › bib › wang2017stereodso
Source type	Paper
Keywords	Visual Odometry, SLAM
Summary of key points	The paper proposes a method for localization of a robot system in a large-scale environment. Used two popular data sets to test out their algorithm and concluded that it surpassed other algorithms such as ORB SLAM (not including the loop enclosure for fair comparison).
Important Figures	 New Stereo Frame Initialized? Yes No Initialization Tracking on reference KF New FK? No Yes Make non-KF Make new KF Joint optimization Marginalization Refine KFs Add new KF Optimize active window Marginalize KF and points Active Window Active KFs 3D points Co-visibility Graph Figure 2. System overview.
Reason for interest	Considering using computer vision to map an environment with the drone
Notes	Abstract Summary: The paper highlights a certain method used to estimate the pose (position and orientation) of quadcopters using stereo cameras on large scale environments. They claim that their method is more precise and detailed than previous attempts at 3D mapping of an environment. Introduction: Now that there has been more promise in SLAM algorithms over the years, there is a high demand for them because they are accurate and have a sensing range, and are relatively cheap to implement. This paper claims to have the most robust algorithm.
Follow up Questions	How could the algorithm be changed to accomodate for loop enclosure (fixing mistakes and connecting pieces of the map together)?

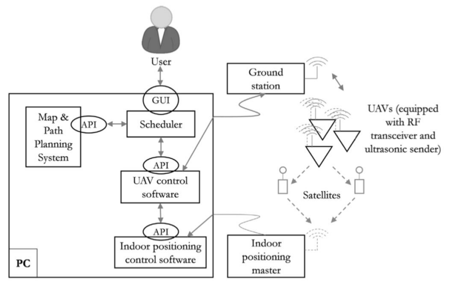
Article #2: Survey of SLAM

Source Title	Real-time Simultaneous Localization and Mapping for UAV: A Survey
Source Author	Jiaxin Li et. al
Source citation	Li, J., Bi, Y., Lan, M., Qin, H., Shan , M., Lin, F., & Chen, B. M. (2016). *Real-time Simultaneous Localization and Mapping for Uav: A Survey*. *Real-time Simultaneous Localization and Mapping for UAV: A Survey*.
Original URL	http://www.imavs.org/papers/2016/237_IMAV2016_Proceedings.pdf
Source type	Paper
Keywords	SLAM
Summary of key points	Discusses the existing SLAM algorithms and the issues with certain algorithms being implemented in drone applications, such as stability and processing time and capability
Important Figures	 Sensor Data Images Laser scans IMU Barometer … … Front-end Sensor model Feature extraction Tracking Place recognition … … Back-end MAP Pose-Graph-optimization … … Localization & Mapping Figure 1: Modern architecture of SLAM consists of front-end and back-end. Figure 1: The front end of SLAM includes computer vision and sensor data processing techniques which filter out the geometrical information and feed that information into mathematical models. The optimization, or fine tuning of these models to get the desired results is part of the back end of SLAM. The back end is where the maps and/or poses are achieved.
Reason for interest	I am considering using SLAM so I want to know what type of SLAM is the best for my application and is most feasible.
Notes	- Abstract Summary: This paper discusses the various methods for potentially achieving 3D SLAM (Simultaneous Localization and Mapping), specifically those methods which are relevant for Autonomous UAV solutions. - Introduction: GPS is fine for outdoor applications, but not indoors. UWB and Mo Cap can be used indoors but are too expensive. Onboard sensors are ideal. Visual odometry has become popular for this reason. - SLAM is like a chicken & egg problem. Localization needs mapping of the area & mapping needs localization of the object.
Follow up Questions	What is the most reliable type of SLAM used in labs today? What type of SLAM would be ideal and feasible for indoor environments?

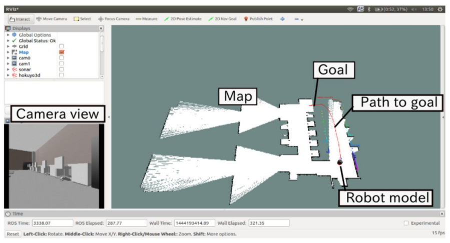
Article #3: Multi-UAV Collaborative SLAM

Source Title	Multi-UAV Collaborative Monocular SLAM
Source Author	Patrik Schmuck and Margarita Chli
Source citation	Schmuck, P., & Chli, M. (May 2017). Multi-UAV collaborative monocular SLAM. Paper presented at the 3863-3870. doi:10.1109/ICRA.2017.7989445
Original URL	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7989445&tag=1
Source type	Conference Proceeding
Keywords	Collaborative SLAM, Ground station, Keyframe
Words to Investigate	Extended Kalman Filter (EKF), Bundle Adjustment, AsTec Neos (drone)
Important Figures	 Fig. 6: Global Map with KFs from two agents (blue and black camera frusta) and position ground truth (solid lines). Absolute trajectory error (RMSE): 0.22m
Reason for interest	Another approach of mapping of an environment
Notes	They propose a centralized system where there are multiple small drones flying around an area in different directions that send information to a ground station (server) which processes the information and tries to combine all the individual maps. Multiple drones can increase the reliability of the estimated map. When the drones send information to these servers, the servers store it so the drones do not have to process as much information.

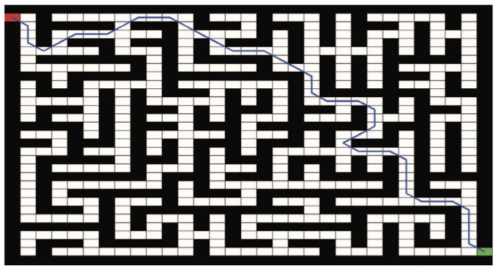
Article #4: Review of Indoor UAV Application

Source Title	A system of UAV application in indoor environment
Source Author	Yohanes Khosiawan & Izabela Nielsen
Source citation	Khosiawan, Y., & Nielsen, I. (2016). A system of UAV application in indoor environment. *Production & Manufacturing Research, 4*(1), 2-22. doi:10.1080/21693277.2016.1195304
Original URL	https://www.tandfonline.com/doi/pdf/10.1080/21693277.2016.1195304?needAccess=true
Source type	Journal
Summary of key points	High-level feasibility of an indoor application (cost, safety, accuracy) Break down how an indoor drone system works (figure below)
Important Figures	 User GUI Map & Path Planning System API Scheduler API UAV control software API Indoor positioning control software PC Ground station UAVs (equipped with RF transceiver and ultrasonic sender) Satellites Indoor positioning master **Figure 4.** Physical architecture of UAV system in 3D indoor environment.
Reason for interest	Weighing options for indoor vs. outdoor application
Notes	● Indoor UAVs are great for using "upper air space" ● Difficulties: narrow spaces, obstacles, no GPS, height constraints ● Three elements for a UAV Application ○ Task- end goal at hand ○ Environment- where will drone fly ○ Operation system- sensors, docking station ● Unexpected event may harm drone/environment ○ In a factory a machine might break down or the drone's engine fails and crashes into humans in scene ○ Propeller harm ○ Unexpected closed doors- cannot go to desired location ● Aspects of indoor uav application ○ Map and path planning ○ Localization or positioning ○ Control system- ground station ○ Scheduler- commands drone ■ Takes information and puts into action

Article #5: Terrestrial Robot Simulation

Source Title	Simulation Environment for Mobile Robots Testing Using ROS and Gazebo
Source Author	Kenta Takaya et. al
Source citation	Takaya, K., Asai, T., Kroumov, V., & Smarandache, F. (2016). Simulation environment for mobile robots testing using ROS and Gazebo. *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, 96-101.
Original URL	http://fs.unm.edu/ScArt/SimulationEnvironment.pdf
Source type	Conference Proceeding
Summary of key points	This paper discusses how they used ROS & Gazebo for autonomous navigation of terrestrial robots and mapping the environment in 3D. The code developed for the Gazebo simulator can directly be implemented in real life.
Important Figures	 Fig. 4. Map generation using Hector mapping in Rviz
Reason for interest	This paper uses the simulator I would like to use for my project (Gazebo)
Notes	- Goal of research is to develop a robot that can guide the elderly or disabled in outdoor or indoor environments - General: Many publications about robot simulations of drones, terrestrial robots, and underwater vehicles are based on ROS and Gazebo packages - URDF (Universal Robotic Description Format) is an XML file in ROS used for describing elements such as sensors, joins, and links in a robot model. URDF is converted to SDF (Simulation Description Format) using gazebo_plugins (a topic) in the file. The SDF file is compatible with Gazebo & describes the simulated world with the robot. - For their research: Used laser sensor & dead reckoning for building map for reliability indoors and outdoors. Octomap package & Hokuyo YVT-X002 LIDAR sensor used for 3D map, which uses odometry measurements. Hector mapping (SLAM Algorithm) used for 2D map
Follow up Questions	Would the uncertainty in map measurements increase with a flying robot?

Article #6: Autonomous Indoor Low-Cost UAVs

Source Title	Autonomous Indoor Navigation of Low-Cost Quadcopters
Source Author	Ahmed Hussain et. al
Source citation	Hussein, A., Al-Kaff, A., de la Escalera, A., & Armingol, J. M. (Nov 2015). Autonomous indoor navigation of low-cost quadcopters. Paper presented at the 133-138. doi:10.1109/SOLI.2015.7367607 Retrieved from https://ieeexplore.ieee.org/document/7367607
Original URL	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7367607
Source type	Conference Proceeding
Summary of key points	
Important Figures	 Fig. 1. Path planning grid map
Reason for interest	The drone system presented is similar to the product I would like to achieve in my project.
Notes	The objective of this paper was to use an affordable drone, namely the Parrot AR Drone 2.0, to find its way in an unknown environment. To accomplish this task, the authors found the possible path that could be taken by the drone with a Simulated Annealing (SA) optimization algorithm, which is a path planning algorithm. In their procedure, they used an Inertial Measurement Unit (IMU) consisting of accelerometer and gyroscope sensors. They had a total of four experiments to test the quality of their implementation of the SA algorithm.

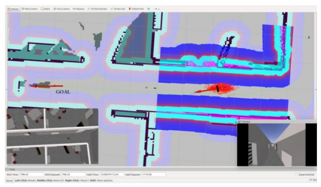
Article #7: Drone Application in Agriculture

Source Title	Agriculture drones: A modern breakthrough in precision agriculture
Source Author	Vikram Puri, Anand Nayyar & Linesh Raja
Source citation	Puri, V., Nayyar, A., & Raja, L. (2017). Agriculture drones: A modern breakthrough in precision agriculture. *Journal of Statistics and Management Systems, 20*(4), 507-518. doi:10.1080/09720510.2017.1395171
Original URL	https://www.tandfonline.com/doi/pdf/10.1080/09720510.2017.1395171?needAccess=true
Source type	Journal
Keywords	Agricultural drones
Summary of key points	This article discusses the various tasks that drones can accomplish on farms. They compare different drones used today in the agricultural field and their capabilities.
Important Figures	Figure 5 Agras MG-1-DJI This drone has 4 nozzles for spraying fertilizer in farms. It is 70 times faster than farmers spraying manually, covering up to 6000 square meters of land in 10 minutes.
Reason for interest	It is a UAV application of relevance today in an important industry
Notes	- Drones are useful in farms for inspecting soil. They can produce 3D maps of soil useful for farmers during seed ploughing. - Drones can also inspect crops efficiently and accurately. Using infrared and other sensors, drones can determine the health of crops and relay that information to farmers. - They can also spray pesticides or water to crops precisely and consistently , making sure the right amount is sprayed on each crop. This will increase the crop yield and maintain the overall quality - Using GIS (Geographic Information System) Mapping, drones can measure certain features of the field and allow farmers to manage issues with resources and cost.
Follow up Questions	Where in the United States are these drones being used? (Specific places) How much profit have drones contributed to the agricultural industry?

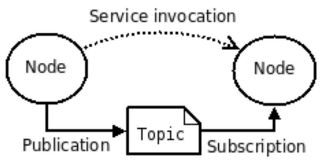
Article #8: Medical Aid Delivery Drone

Source Title	Medical Delivery Drones Take Flight in East Africa
Source Author	Evan Ackerman & Eliza Strickland
Source citation	Ackerman, E., & Strickland, E. (2018, Jan). Medical delivery drones take flight in east africa. *IEEE Spectrum, 55*, 34-35. doi:10.1109/MSPEC.2018.8241731 Retrieved from https://ieeexplore.ieee.org/document/8241731
Original URL	https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8241731
Source type	Journal Article
Keywords	Autonomous delivery drone, GPS Navigation, medical aid delivery
Summary of key points	This journal describes a company called Zipline using drones to transport blood products to hospitals in times of emergency in Rwanda
Important Figures	This is a package delivered by zipline drones. They are released with a parachute to ensure safe landing. When the package is about to be released, the drone notifies its receiver through text
Reason for interest	It is a UAV application of relevance today, serving people and saving lives
Notes	- The drone cuts normal delivery time from hours to minutes. Drones are safer as they do not rely on infrastructure on the ground which can break down and do not get slowed by rough terrain or human traffic. - The drone uses GPS navigation and air traffic control to head to its destination. - Saved the life of a 24 year old woman who lost blood giving birth through C-section. Doctors placed an emergency order and were able to get seven units of red blood cells, two units of plasma, and two units of platelets just in time. - Drone can carry 1.5 kg worth of payload. The drone itself weighs 12 kg
Follow up Questions	What type of battery does the drone use? What is the maximum flight time of the drone? Can Zipline add a safety component where the drone can facially recognize the person who ordered the package to ensure medical aid is in the right hands?

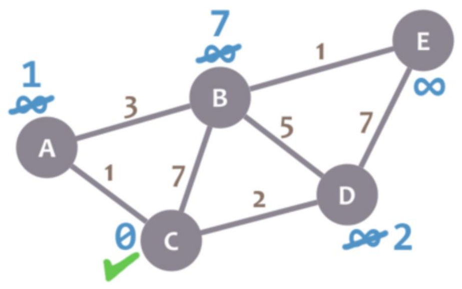
Article #9: Autonomous Mapping & Exploration

Source Title	Autonomous Mapping and Exploration with Unmanned Aerial Vehicles Using Low Cost Sensors
Source Author	Ankit A. Ravankar, Abhijeet Ravankar, Yukinori Kobayashi, & Takanori Emaru
Source citation	Ravankar, A., Ravankar, A., Kobayashi, Y., & Emaru, T. (2018). Autonomous Mapping and Exploration with Unmanned Aerial Vehicles Using Low Cost Sensors. *Proceedings*, *4*(1), 44. doi: 10.3390/ecsa-5-05753
Original URL	https://www.mdpi.com/2504-3900/4/1/44
Source type	Conference Proceeding
Keywords	Autonomous mapping and exploration, UAVs, sensor fusion
Summary of key points	This paper proposes a sensor fusion algorithm that can build a dense 3D map using low cost sensors such as a 2D LIDAR and an RGBD camera on a quadcopter. They test their method in the Gazebo simulator and use ROS
Important Figures	 **Figure 4.** Autonomous navigation using adaptive monte carlo localization (AMCL). The red arrows indicate particles. Goal is indicated by the red arrow. Cost map around the obstacles are shown in blue and cyan.
Reason for interest	Their research is very similar to what I will be doing
Notes	- Drone Model: Based on Astec Hummingbird drone. Contains IMU, barometer, Microsoft Kinect camera, & 2D LIDAR - Extended Kalman Filter: Controlled drone's velocity, position, and orientation, PID controllers stabilized drone - Simulated ArduPilot flight controller which translated thrust and torque messages into motor voltages to fly UAV - SLAM: Used GMapping to create a 2D occupancy grid map from LIDAR data. Map continuously updates as drone explores. - Octomap package: Constructs 3D occupancy map from the 3D point cloud generated from Kinect sensor using spatial alignment - A* algorithm for path planning - Monte Carlo Localization (AMCL) stack from ROS: uses particle filters to track the pose of a robot on the given map
Follow up Questions	How did they create their Gazebo environment? Did they import it?

Article #10: Robot Operating System (ROS)

Source Title	ROS Wiki
Source Author	Open Source Robotics Foundation
Source citation	ROS Wiki. (n.d.). Retrieved from http://wiki.ros.org/ROS/Introduction.
Original URL	http://wiki.ros.org/ROS/Introduction
Source type	Documentation Website
Summary of key points	Describes what is ROS and its functions
Important Figures	 Service invocation Node Node Publication Topic Subscription
Reason for interest	I will be using ROS for building code
Notes	- ROS stands for Robot Operating System - Powerful tool for autonomous robotics programming as it is open source and is applicable to many coding languages such as Python, Java, or C++ - Provides various libraries for building and writing code - It runs on Unix-based platforms including Ubuntu and Mac OS X - Package: organized collection of nodes, libraries, or config files - Master: Allows nodes to find each other and exchange messages - Node: a program that runs inside a user's pc under the ROS master. Each node has a single purpose and is executable by the user. - Topic: a stream of messages over which nodes can communicate over. Programs use this by having a node publish its information to a topic, where subscribing nodes can read information on the topic. - Message: defines what happens in a topic. A node sends a message to a topic. Other notes may subscribe to this topic to receive the message. This all happens through the master system. - Bags: a way to save ROS message data (sensor data) to reference again - Service: a request/reply system where a node offers a service under a name and the client (another node) uses the service by sending a request message and awaiting for a reply message from the service.

Article #11: Dijkstra's Algorithm

Source Title	Dijikstra's Algorithm
Source Author	Rasco
Source citation	Shortest Paths with Dijkstra's Algorithm. (2017). Retrieved from https://www.codingame.com/playgrounds/1608/shortest-paths-with-dijkstras-algorithm/dijkstras-algorithm.
Original URL	https://www.codingame.com/playgrounds/1608/shortest-paths-with-dijkstras-algorithm/dijkstras-algorithm
Source type	Website
Summary of key points	This algorithm calculates the shortest path between a selected node and every other node in a graph.
Important Figures	 A B C D E 1 7 0 2 ∞ 3 1 7 5 7 2 1
Reason for interest	This algorithm is the basis for the A* path-planning algorithm, which was used in a previous paper I read.
Notes	Summary: 1. Mark selected initial node with a current distance of 0 and every other node with ∞ as the minimum distances are not known yet 2. Select the non-visited node with the smallest current distances as the current node C. 3. For each neighbor N of the current node C, add the current distance of C with the weight of the edge connecting C to N. If that value is smaller than the current distance of N, set it as the new current distance of N 4. Mark the current node C as visited 5. If there are unvisited nodes, go to step 2. Pick the unvisited node with the smallest minimum distance.
Follow up Questions	How would the actual algorithm look like for a very large graph (in code)? Is a graph generated for the map of an environment? If so, how?

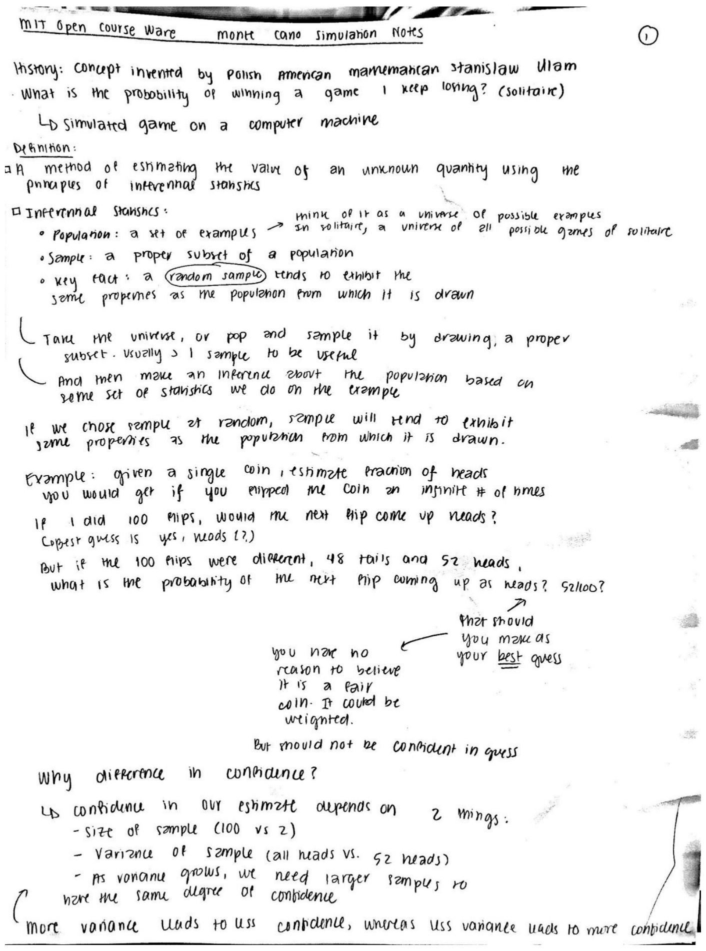
Article #12: What is Gazebo

Source Title	Beginner: Overview
Source Author	OSRF (Open Source Robotics Foundation)
Source citation	OSRF. (n.d.). Beginner: Overview. Retrieved from http://gazebosim.org/tutorials?tut=guided_b1&cat=.
Original URL	http://gazebosim.org/tutorials?tut=guided_b1&cat=
Source type	Tutorial
Reason for interest	Overview of what the Gazebo simulator is
Notes	● Dynamic 3D simulator ● Can test robotics algorithms, design robots, and do regression testing ● Provides various sensors, robot models, and environments ● Realistic Scenario ● High quality graphics ● Simulate indoor and outdoor environments

Article #13: Introduction to SLAM

Source Title	An Introduction to Simultaneous Localisation and Mapping
Source Author	Kudan
Source citation	An Introduction to Simultaneous Localisation and Mapping. (2016, May 23). Retrieved from https://www.kudan.io/post/an-introduction-to-simultaneous-localisation-and-mapping.
Original URL	https://www.kudan.io/post/an-introduction-to-simultaneous-localisation-and-mapping
Source type	Blog Post
Reason for interest	This is about a method I will likely develop in my project.
Notes	- SLAM is NOT an algorithm. Marker-based tracking, such as using AR tags, is not SLAM as the "map" is already known. 3D reconstruction with a fixed camera is not SLAM as the positions of the cameras are known. - It is the challenge of trying to find the position and orientation of a sensor with respect to the environment while simultaneously trying to form a map of that environment. Note that the method is executed in real time, which means that processing on each camera frame must be completed before the next frame comes. Unlike real time processing, offline reconstruction cannot give information about the camera positions. First research conducted in 1986 on terrestrial robots with laser sensors. - Visual SLAM is solving the problem using a camera sensor. The majority of research in this field uses stereo cameras or a combination of monocular cameras and other sensors such as a GPS or an IMU. - Visual SLAM works by tracking a set of features or points through successive camera frames and trying to find their 3D location while at the same time using that information to calculate the camera poses at those times. SLAM can be seen as an optimization problem to minimize the error between the tracked and expected locations, which is called bundle adjustment. Multi-core processor machines are fundamental to being able to run this algorithm (bundle adjustment). Using a multi-core processor, the localization part can separately run in real time on one thread while the mapping thread runs bundle adjustment on the map on the other thread. - Loop closure connects the missing pieces of the puzzle together, minimizing the error. Relocalization is needed when the camera is covered for some time. It is when the tracker, which tracks camera pose, matches a part of the previously visited map with the most current camera view.

Article #14: Monte Carlo Simulation

Source Title	Lecture 6: Monte Carlo Simulation
Source Author	John Guttag
Source citation	Eric Grimson, John Guttag, and Ana Bell. *6.0002 Introduction to Computational Thinking and Data Science.* Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.
Original URL	https://youtu.be/OgO1gpXSUzU (originally accessed lecture as a video)
Source type	Lecture
Reason for interest	Wanted to familiarize myself with this term as it is part of a localization method called called AMCL or augmented monte carlo localization, which uses a particle filter to estimate the position of a robot using a known map of an environment. I had found this term on multiple papers and videos.
Notes	 MIT Open Course Ware monte carlo simulation Notes ① History: concept invented by Polish American mathematician Stanislaw Ulam · What is the probability of winning a game I keep losing? (solitaire) ↳ simulated game on a computer machine Definition: □ A method of estimating the value of an unknown quantity using the principles of inferential statistics □ Inferential statistics: • Population: a set of examples → think of it as a universe of possible examples. In solitaire, a universe of all possible games of solitaire • Sample: a proper subset of a population • key fact: a random sample tends to exhibit the same properties as the population from which it is drawn Take the universe, or pop and sample it by drawing, a proper subset. Usually > 1 sample to be useful And then make an inference about the population based on some set of statistics we do on the example If we chose sample at random, sample will tend to exhibit same properties as the population from which it is drawn. Example: given a single coin, estimate fraction of heads you would get if you flipped the coin an infinite # of times If I did 100 flips, would the next flip come up heads? ↳ best guess is yes, heads (?) But if the 100 flips were different, 48 tails and 52 heads, what is the probability of the next flip coming up as heads? 52/100? ↗ that should you make as your best guess you have no reason to believe it is a fair coin. It could be weighted. But should not be confident in guess Why difference in confidence? ↳ confidence in our estimate depends on 2 things: - size of sample (100 vs 2) - Variance of sample (all heads vs. 52 heads) - As variance grows, we need larger samples to have the same degree of confidence more variance leads to less confidence, whereas less variance leads to more confidence

The Law of Large Numbers: Bernoulli's Law:
One of the two most important theorems in statistics

↳ In repeated independent tests with the same actual probability p of a particular outcome in each test, the chance that the fraction of times that the outcome occurs differs from p coverges to zero as the number of trials goes to infinity

Gambler's fallacy
↳ Has to even out → NO

Regression to the mean - different to gambler's fallacy

↳ Following an extreme random event, the next random event is likely to be less extreme

Quantifying Variation in Data

squaring - don't care + or - squaring gives outliers larger emphasis

$$\text{variance}(X) = \frac{\sum_{x \in X} (x - \overset{\text{average}}{\mu})^2}{|X|} \leftarrow \text{normalises by \# of numbers}$$

$$\sigma(X) = \sqrt{\frac{1}{|X|} \sum_{x \in X} (x - \mu)^2}$$

- Standard deviation simply the square root of the variance
- outliers can have a big effect
- standard deviation should always be considered <u>relative to mean</u>

Confidence levels and intervals

- Instead of estimating an unknown parameter by a single value (eg, the mean of a set of trials), a confidence ~~the~~ interval provides a range that is likely to contain the unknown value and a confidence that the unknown value lays within that range
- "The return on betting a pocket 10k times in European roulette is -3.3%, the margin of error is +/- 3.5% with a 95% level of confidence"
- What does this mean?
- If I were to conduct an infinite # of trials of 10k bets each.
- My expected average return would be -3.3%
- My return would be between -6.8% and 0.2% 95% of the time

PDFs

- Distributions defined by Probability density functions (PDFs)
- probability of a rand var lying btwn 2 values
- Defines a curve where the values on the x-axis lie btwn min and max value of the variable
- Area under curve btwn 2 points, is probability of example falling within that range

Normal distribution - equally likely errors are in each direction.
↳ peaks at mean

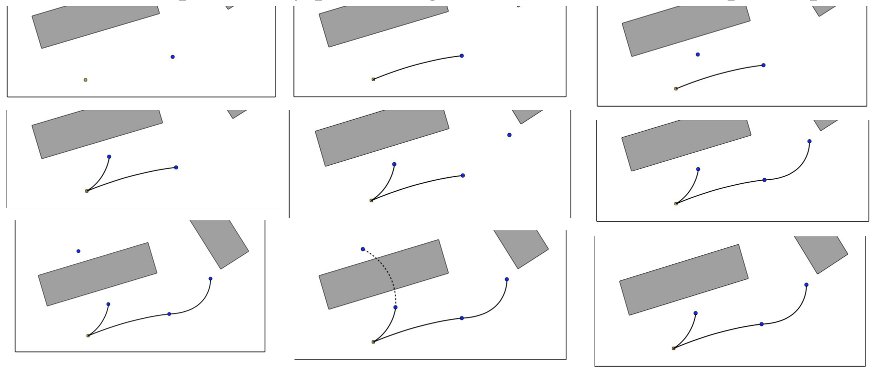
Article #15: Kalman Filter Intro

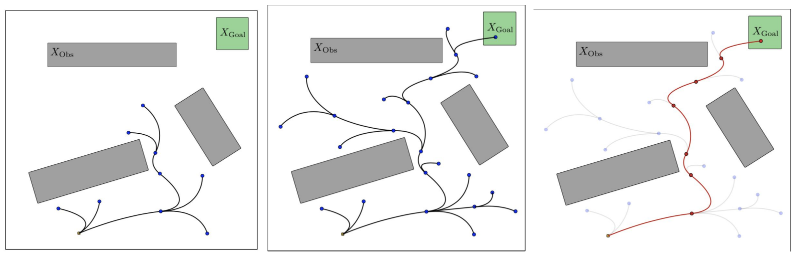
Source Title	Why You Should Use The Kalman Filter Tutorial
Source Author	Ritesh Kanjee
Source citation	Startups, A. (2016, August 19). Retrieved from https://www.youtube.com/watch?v=bm3cwEP2nUo.
Original URL	https://youtu.be/bm3cwEP2nUo
Source type	Youtube Tutorial
Reason for interest	It is used in the EKF SLAM Algorithm and is a common term in robotics
Notes	- You can use a gaussian curve to estimate a moving object's position. The mean of the curve is likely where the object is, but there is variance so the object can be a little below or above the mean. - You find the position in two ways. Using a radar, you estimate the object is 15 meters away (mean of your gaussian curve). That is your measurement estimate. Using kinematics questions (given object's velocity), you estimate the object is 16.5 meters away (mean of your gaussian curve). That is your state estimate. What will be closest to the true position of the object? Is it the state estimate or the radar estimate, or exactly in between? You cannot take the average of the two estimates and conclude that the object is 15.75 meters away, because the two estimators are not equally weighted. This means that one estimator may be more accurate than the other because one may have a higher probability than the other. The key to solve this problem is to take the mean and variance of each of the estimators and form a new gaussian curve (optimal estimate) that has a lower variance and a mean between the two curves. - The 1st step in finding the new PDF (probability density function) is to calculate the kalman gain. This is calculated by using the measurement error and state estimate error. The kalman gain is the state estimate error divided by the sum of the state estimate and measurement error. The kalman gain can have a value between 0 and 1. If the gain is 0, then the estimate is stable and the measurements are inaccurate. If the gain is 1, the measurements are accurate but the estimate is unstable. - The 2nd step is to calculate the current estimate (at t). This is found by taking the estimate from the state estimator (at t-1) and adding that to the product of the kalman gain and the difference between the measured state estimate and the state estimator estimate. - The 3rd step is to calculate the new state estimate error. That is calculated by taking 1 minus the kalman gain multiplied by the state estimate error at t-1.

Article #16: Introduction to RRT Path Planning

Source Title	Robotic Path Planning: RRT and RRT*
Source Author	Tim Chin
Source citation	Chin, T. (2019, February 26). Robotic Path Planning: RRT and RRT*. Retrieved from https://medium.com/@theclassytim/robotic-path-planning-rrt-and-rrt-212319121378.
Original URL	https://medium.com/@theclassytim/robotic-path-planning-rrt-and-rrt-212319121378
Source type	Blog Post
Reason for interest	This will provide me an introduction to a path planning algorithm I can implement for my project
Notes	The path planning problem in robotics involves a robot trying to navigate from point A to point B while avoiding obstacles (Cobs). The robot can move through Cfree in order to get to the destination point B. Most path planning algorithms do not create a graph of edges and vertices and find a shortest path. The RRT Algorithm does both, but does not find the shortest path. RRT*, developed by Dr. Karaman and Dr. Frazzoli, builds on top of RRT and works to find a shortest path. The RRT Algorithm works by randomly generating points and trying to connect each point to its closest neighbor point. The randomly generated points are made sure that they lie in Cfree. When points are chained into a tree, it is made sure that the connections do not pass through obstacles and avoid obstacles instead. When a randomly generated point reaches in a certain radius of the goal or hits a limit, the algorithm finishes execution. The upside to this algorithm is that it is very fast and simple to implement. The downside to this algorithm is that it does not find a shortest path. The tree graph is very cubic and makes irregular paths. The RRT* Algorithm is the same as the RRT algorithm but with 2 significant additions. The first being that RRT* finds the costs of the vertices, which are the distances between the points/vertices and their parent vertex. Nodes with cheaper costs replace the nearest nodes. The second addition is that RRT* rewires the tree every time a new vertex is connected to its nearest neighbor. This rewiring ensures that the cost of each node is minimized and makes the graph smoother and straighter than the RRT algorithm's graph.

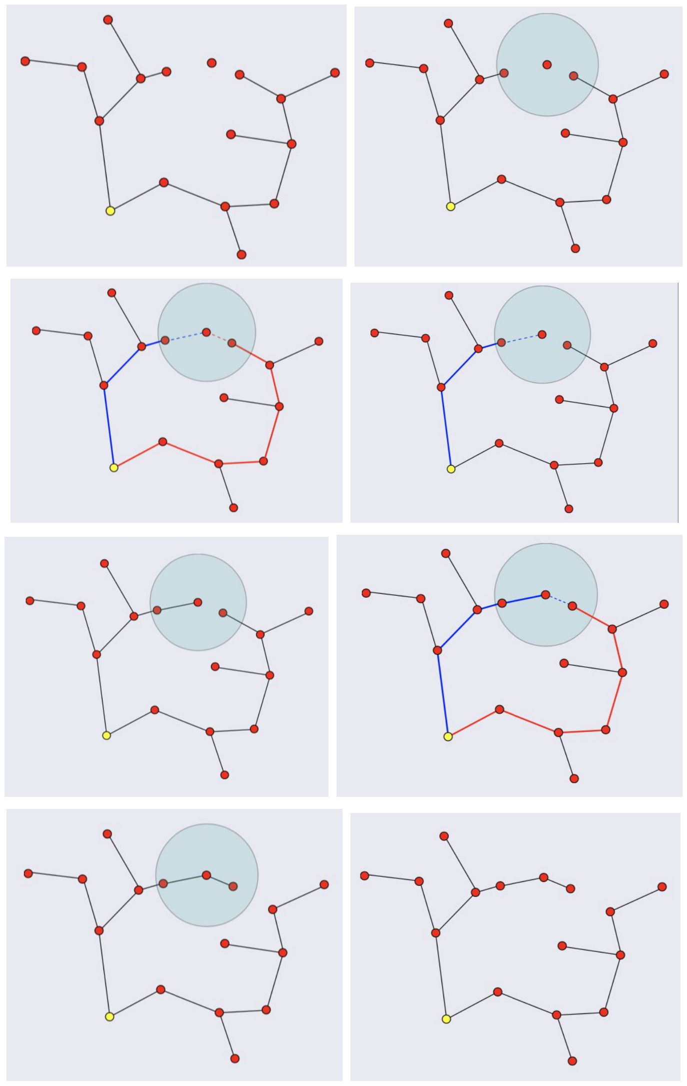
Article #17: RRT and RRT* Algorithms

Source Title	Lecture 15: Sampling Based Algorithms for Motion Planning
Source Author	Emilio Frazzoli
Source citation	Brian Williams, and Emilio Frazzoli. *16.410 Principles of Autonomy and Decision Making.* Fall 2010. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.
Original URL	https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16_410F10_lec15.pdf
Source type	Course Presentation
Reason for interest	This goes more in depth about RRT and RRT*
Notes	Basic Robotics Problem: Find control signal u (if exists) in dx/dt = f(x,u) such that x(t) is only part of the goal set X_goal and not set X_obs Motion planning methods: - Algebraic planners: using complicated algebra to find path. Impractical - Discretization + graph search: sensitive to graph size - Potential fields/navigation functions: Attractive and repulsive forces - Incremental sampling-based algorithms: based on distribution samples. Real-time, multi-resolution, applicable to general dynamical systems. Probabilistic Roadmaps (PRM): 1994, Kavraki & Latombe - Offline method of building a graph of environment - Learning: Generate n points in X_free = [0,1]^d / X_obs. Connect points using local planner that don't cut through obstacles and add edges to graph - Run: connect start and end goal to closest nodes & find path on roadmap Rapidly-exploring Random Trees (RRT): 1998, LaValle & Kuffner - Online method of building a tree to explore state space region - Sample random point from X_free and connect to closest point in tree. - Con: RRT traps itself by preventing the formation of more optimal paths 

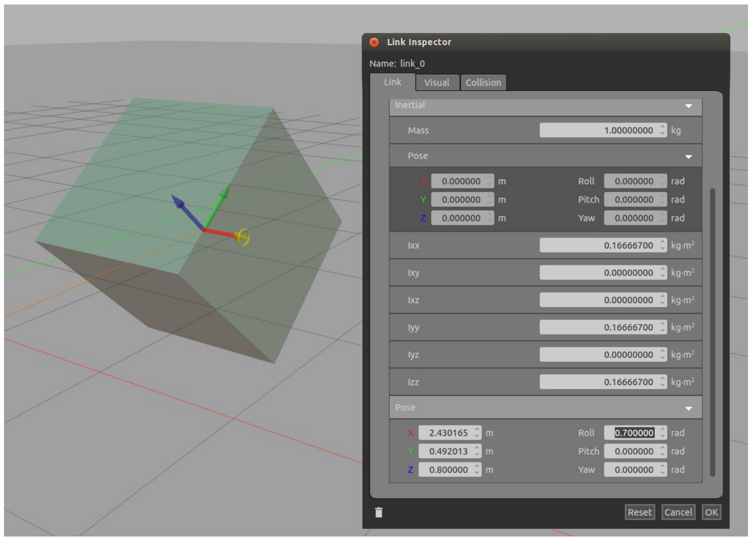


RRT*- Variant of Rapidly Exploring Random Graphs (RRG)

- Rewires entire tree as better paths are discovered by finding cost.



Article #18: Gazebo Model Editor

Source Title	Model Editor
Source Author	OSRF
Source citation	OSRF. (n.d.). Model Editor. Retrieved from http://gazebosim.org/tutorials?tut=model_editor.
Original URL	http://gazebosim.org/tutorials?tut=model_editor
Source type	Documentation Website
Reason for interest	More background on the Gazebo Interface
Notes	- Run gazebo by typing "gazebo" in terminal - Ctrl+M to open Model Editor - GUI: Palette has the Insert and Model (list of current parts in model) tabs, and 3D view allows user to preview and edit their model's properties - Using the basic cylinder, sphere, and cube shapes provided, various objects can be constructed - To add meshes: Add button under Custom Shapes. Meshes are used to insert custom 3D objects in other formats (such as stl) that have complex size measurements and are not necessarily defined by a certain width, length, and height. - Create Joints using the line segment icon in toolbar. Joints connect objects together. Different types available like fixed, revolute, and screw. Joint inspector allows joint properties to be edited. - Using the link inspector in the model editor, the visual appearance, size, and orientation of objects can be changed  Link Inspector Name: link_0 Inertial: Mass 1.00000000 kg; Pose: X 0.000000 m, Y 0.000000 m, Z 0.000000 m, Roll 0.000000 rad, Pitch 0.000000 rad, Yaw 0.000000 rad; Ixx 0.16666700 kg·m²; Ixy 0.00000000 kg·m²; Ixz 0.00000000 kg·m²; Iyy 0.16666700 kg·m²; Iyz 0.00000000 kg·m²; Izz 0.16666700 kg·m². Pose: X 2.430165 m, Y 0.492013 m, Z 0.800000 m, Roll 0.700000 rad, Pitch 0.000000 rad, Yaw 0.000000 rad

Article #19: Building Chassis Robot & Env in Gazebo

Source Title	[Tutorial] Building a Simulated Model for Gazebo and ROS from Scratch (part 1)
Source Author	Richard Wang
Source citation	Wang, R. (2016, September 20). Retrieved from http://moorerobots.com/blog/post/1
Original URL	http://moorerobots.com/blog/post/1
Source type	Blog Post
Reason for interest	I used this tutorial a while back and wanted to include in my notes
Notes	In this tutorial I learned the basic structure of a Gazebo ROS environment. There are three packages necessary and they are named in the following manner: mybot_control: enable control for joints of robot model mybot_description: describes robot model mybot_gazebo: contains world files and has a launch file for launching the robot model and environment I learned how to create the run launch files that ran a gazebo world file I also learned how to publish twist commands to make the robot move: rostopic pub /cmd_vel geometry_msgs/Twist "linear: x: 0.2 y: 0.0 z: 0.0 angular: x: 0.0 y: 0.0 z: 0.1" More details are provided in my STEM logbook

Article #20: Adding Sensors to Chassis Robot

Source Title	[Tutorial] Adding Sensors to the Gazebo Model (part 2)
Source Author	Richard Wang
Source citation	Wang, R. (2016, September 20). Retrieved from http://moorerobots.com/blog/post/2
Original URL	http://moorerobots.com/blog/post/2
Source type	Blog Post
Reason for interest	I used this tutorial a while back and wanted to include in my notes

Notes	In this tutorial I learned how to add the camera and laser sensor models and plugins to the appropriate xacro files.
	
	I learned how to run the camera node individually and also in rviz to see the camera image output:
	$ rosrun image_view image_view image:=/mybot/camera1/image_raw
	$ roslaunch mybot_description mybot_rviz.launch
	
	More details are provided in my STEM Logbook

Article #21: Using the ROS Navigation Stack

Source Title	[Tutorial] Simulating the ROS Navigation Stack (part 3)
Source Author	Richard Wang
Source citation	Wang, R. (2016, September 20). Retrieved from http://moorerobots.com/blog/post/3
Original URL	http://moorerobots.com/blog/post/3
Source type	Blog Post
Reason for interest	I used this tutorial a while back and wanted to include in my notes
Notes	In this tutorial I learned how to use the turtlebot to map an environment and autonomously be able to navigate it. I ran the commands below and controlled the turtlebot in teleop mode by pressing keys on the keyboard. I first had to run the gazebo world file. On a separate terminal tab, I had to run the gmapping launch file (provided by ROS) which runs a SLAM algorithm and localizes the robot while building a map of the environment around the robot. On a separate terminal tab, I then ran the rviz file to see the map built as I navigated in the world. On another terminal window I ran the teleop launch file which allowed me to control the robot around the environment. When I was finished with the mapping, I saved the map to a path and then closed all the terminals. I ran the world file once again as well as the saved map file with an amcl file and the rviz file. Using 2D navigation goals, I was able to specify the desired destination to the turtlebot and the robot was able to navigate without bumping into any obstacles. 1. roslaunch turtlebot_gazebo turtlebot_world.launch 2. roslaunch turtlebot_gazebo gmapping_demo.launch 3. roslaunch turtlebot_rviz_launchers view_navigation.launch 4. roslaunch turtlebot_teleop keyboard_teleop.launch 5. rosrun map_server map_saver -f ~/mybot_ws/test_map 1. roslaunch turtlebot_gazebo turtlebot_world.launch 2. roslaunch turtlebot_gazebo amcl_demo.launch map_file:=~/mybot_ws/test_map.yaml 3. roslaunch turtlebot_rviz_launchers view_navigation.launch More details are provided in my STEM logbook.

Article #22: Introduction to Robotics

Source Title	Probabilistic Robotics, Chapter 1: Introduction
Source Author	Sebastian Thrun, Wolfram Burgard, and Dieter Fox
Source citation	Thrun, S., Burgard, W., & Fox, D. (2010). *Probabilistic Robotics*. Retrieved from https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Original URL	https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Source type	Textbook
Reason for interest	This chapter talks about general characteristics of autonomous robotics
Notes	- Probabilistic Robotics describes the field of computer controlled mechanical objects using sensors to understand the environment around them and trying to navigate and manipulate other objects in the environment. These robotics systems are in practice in a wide variety of fields, from assembly lines and surgical procedures in medicine to space exploration and self-driving cars. - There is a shift in the perceived potential of these robotics systems. Assembly lines, once thought to be a radical invention, is much more predictable compared to the applications of robotics systems today. Unlike robotics systems in the past, robotics systems today utilize sensor measurements to make decisions about what actions to take. - Robotics systems today use the principles of uncertainty to make rational decisions about the future. Uncertainty can be found in an environment where the state of objects may be dynamic, through noise and limitations in sensors, through the control noise in robotic actuators, models, and finally through the algorithms that must run in real-time. - Localization of a robot is a key aspect in autonomous robotics. Localization is the method of estimating the location of a robot with respect to its environment, using sensor data and a map of the environment. Global localization entails localization from scratch. - Probabilistic robotics is essential in certain problems where the location of the robot is suddenly lost and must be recovered, or where an environment must be mapped accurately and precisely without a GPS system.

Article #23: State & Interaction in Probabilistic Robotics

Source Title	Probabilistic Robotics, Chapter 2: Recursive State Estimation
Source Author	Sebastian Thrun, Wolfram Burgard, and Dieter Fox
Source citation	Thrun, S., Burgard, W., & Fox, D. (2010). *Probabilistic Robotics*. Retrieved from https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Original URL	https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Source type	Textbook
Reason for interest	This section gives a brief overview of the states and interactions of a robot with its surroundings
Notes	Sensor measurements, controls, and states are all modeled as random variables in probabilistic robotics and have probability density functions. - State is a collection of information pertaining to the robot as well as the environment. Dynamic state is state that changes. State at a certain time *t* can be noted with the symbol x_t. The state includes everything from the robot's location and orientation, often referred to as a robot's pose, to the velocities and locations of moving objects in the surrounding environment. The markov property states that given the current state x_t and all its previous states, the only state that helps predict the future state would be the current state. - The first type of interaction a robot can have with the environment is to take sensor measurements, such as a laser scan. Measurement data at a particular time *t* can be noted with the symbol z_t. Control actions, on the other hand, involve the robot using its actuators and motors to move itself or to move other objects in the environment. Control data can be noted with the symbol u_t. Even if the robot does not move any of its motors or actuators, it is considered the robot has taken a control action. An odometry sensor can be used to collect control data about the revolution of a robot's wheels. - The belief of a robot, otherwise known as its state of knowledge, describes the set of probabilities about its current state in an environment. The belief about a state at time *t* can be noted as bel(x_t). - At time *t-1*, the robot is at its current state x_t. At time step *t*, the robot takes an action u_t and moves to x_t. The robot then takes a measurement z_t and updates its state to x_t.

Article #24: Bayes Filter

Source Title	Probabilistic Robotics, Chapter 2: Recursive State Estimation
Source Author	Sebastian Thrun, Wolfram Burgard, and Dieter Fox
Source citation	Thrun, S., Burgard, W., & Fox, D. (2010). *Probabilistic Robotics*. Retrieved from https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Original URL	https://docs.ufpr.br/~danielsantos/ProbabilisticRobotics.pdf
Source type	Textbook
Reason for interest	This filter is the basis for many algorithms in the robotics field
Notes	See below

```
1:    Algorithm Bayes_filter($bel(x_{t-1}), u_t, z_t$):
2:        for all $x_t$ do
3:            $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx$
4:            $bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$
5:        endfor
6:        return $bel(x_t)$
```

The goal of the (recursive) Bayes filter is to calculate the current state's new belief based on measurement and control data. The steps are as follows:

1. The robot takes action u_t at x_t. The control and prior belief, bel(x_t-1) are used to calculate the intermediate or posterior belief (line 3)
2. After taking the sensor measurement z_t, the robot calculates the final belief at x_t (line 4)

Example: A robot is trying to predict whether a door is open or closed. The initial belief is the set of probabilities concerning the states of the door. The only states of the door are open and closed. In the beginning, the probabilities of the door being open or closed are both 0.5 (the very first prior belief).
- As the sensors of the robot are noisy, the probabilities of the sensor detecting a certain state of the door must be known. There are two conditional probabilities for every state of the door, totalling the probabilities to four. These probabilities are called sensor or *measurement* probabilities.

$$\begin{aligned} p(Z_t = \textbf{sense\_open} \mid X_t = \textbf{is\_open}) &= 0.6 \\ p(Z_t = \textbf{sense\_closed} \mid X_t = \textbf{is\_open}) &= 0.4 \end{aligned}$$

$$\begin{aligned} p(Z_t = \textbf{sense\_open} \mid X_t = \textbf{is\_closed}) &= 0.2 \\ p(Z_t = \textbf{sense\_closed} \mid X_t = \textbf{is\_closed}) &= 0.8 \end{aligned}$$

- If the robot takes a control action u_t called *do_nothing*, the probability that the door is open given that the door is open is 1, the probability that the door is closed given that the door is open is 0, the probability that the door is open given that the door is closed is 0 and the probability that the door is closed given that the door is closed is 1. These probabilities are called *transition* probabilities.
- If the robot takes a control action u_t called *push*, the probability that the door is open given that the door is open is 1, the probability that the door is closed given that the door is open is 0, the probability that the door is open given that the door is closed is 0.8 and the probability that the door is closed given that the door is closed is 0.2.
- For this example, the robot does nothing. If the intermediate or posterior belief is calculated, the integral in line 3 becomes a finite sum (as this calculation is discrete):

$$\begin{aligned}\overline{bel}(x_1) &= \int p(x_1 \mid u_1, x_0)\; bel(x_0)\; dx_0 \\ &= \sum_{x_0} p(x_1 \mid u_1, x_0)\; bel(x_0)\end{aligned}$$

After substituting values, the parts of the posterior belief can be calculated:

$$\begin{aligned}&\overline{bel}(X_1 = \textbf{is\_open}) \\ &\quad = p(X_1 = \textbf{is\_open} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_open})\; bel(X_0 = \textbf{is\_open}) \\ &\qquad + p(X_1 = \textbf{is\_open} \mid U_1 = \textbf{do\_nothing}, X_0 = \textbf{is\_closed})\; bel(X_0 = \textbf{is\_closed}) \\ &\quad = 1 \cdot 0.5 + 0 \cdot 0.5 \;=\; 0.5 \qquad (2.46)\end{aligned}$$

The probability of the door being closed remains 0.5 as well. To calculate the final belief, a sensor measurement must be taken. From line 4, the sensor accuracies are substituted and multiplied by the corresponding posterior beliefs to get the normalizer *n*.

$$bel(x_1) = \eta\; p(Z_1 = \textbf{sense\_open} \mid x_1)\; \overline{bel}(x_1)$$

$$\begin{aligned}&bel(X_1 = \textbf{is\_open}) \\ &\quad = \eta\; p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_open})\; \overline{bel}(X_1 = \textbf{is\_open}) \\ &\quad = \eta\; 0.6 \cdot 0.5 \;=\; \eta\; 0.3\end{aligned}$$

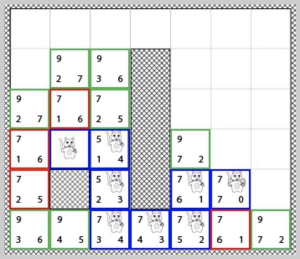
$$\begin{aligned}&bel(X_1 = \textbf{is\_closed}) \\ &\quad = \eta\; p(Z_1 = \textbf{sense\_open} \mid X_1 = \textbf{is\_closed})\; \overline{bel}(X_1 = \textbf{is\_closed}) \\ &\quad = \eta\; 0.2 \cdot 0.5 \;=\; \eta\; 0.1\end{aligned}$$

$$\eta = (0.3 + 0.1)^{-1} = 2.5 \qquad \begin{aligned}bel(X_1 = \textbf{is\_open}) &= 0.75 \\ bel(X_1 = \textbf{is\_closed}) &= 0.25\end{aligned}$$

Article #25: Simple Bayes Rule Example

Reason for interest	Learn the bayes rule and law of total probability
Notes	Problem: You go for your annual checkup and have several lab tests performed. A week later your doctor calls you and says she has good and bad news. The bad news is that you tested positive for a marker of a serious disease, and that the test is 97% accurate (i.e. the probability of testing positive given that you have the disease is 0.97, as is the probability of testing negative given that you don't have the disease). The good news is that this is a rare disease, striking only 1 in 20,000 people. What are the chances that you actually have the disease? x = event you have the disease GIVENS prior $\text{bel}(x) = \frac{1}{20{,}000} = 0.00005 \leftarrow P(x)$ (0.005%) $\text{bel}(\bar{x}) = \frac{19{,}999}{20{,}000} = 0.99995 \leftarrow P(\bar{x})$ (99.995%) y = event that you test positive $*P[y \mid x] = 0.97$ $P[\bar{y} \mid x] = 0.03$ $*P[y \mid \bar{x}] = 0.03$ $P[\bar{y} \mid \bar{x}] = 0.97$ measurement probabilities STEP 1 FORMULA DERIVATION: $P(xy) = P(x) \cdot P(y \mid x)$ $P(xy) = P(y) \cdot P(x \mid y)$ $P(x) \cdot P(y \mid x) = P(y) \cdot P(x \mid y)$ Bayes Rule ⭑IMPORTANT $\leftarrow P(x \mid y) = \frac{P(x) \cdot P(y \mid x)}{P(y)}$ STEP 2 $P(y) = \underbrace{\text{you have disease}}_{P(x)}$ and $\underbrace{\text{tested}}_{P(y \mid x)}$ + or $\underbrace{\text{you don't have disease}}_{P(\bar{x})}$ and $\underbrace{\text{tested}}_{P(y \mid \bar{x})}$ + $P(y) = P(x) \cdot P(y \mid x) + P(\bar{x}) \cdot P(y \mid \bar{x})$ $P(y) = P(y \mid x) \cdot P(x) + P(y \mid \bar{x}) \cdot P(\bar{x})$ (discrete) $P(y) = \sum_i P(y \mid X_i) \cdot P(X_i)$ (continuous) $P(y) = \int f(y \mid x) \cdot f(x)\, dx$ Law of Total Probability ⭑IMPORTANT ★ Find $P(x \mid y)$ STEP 3 $P(x \mid y) = \frac{P(x) \cdot P(y \mid x)}{P(y \mid x) \cdot P(x) + P(y \mid \bar{x}) \cdot P(\bar{x})}$ $P(x \mid y) = \frac{(0.00005)(0.97)}{(0.97)(0.00005) + (0.03)(0.99995)}$ $P(x \mid y) = 0.001614$ $\boxed{P(x \mid y) = 0.1614\%}$ Probability of the person having the disease

Article #26: A* Path Planning Algorithm Example

Source Title	Introduction to A* Pathfinding
Source Author	Ray Wenderlich
Source citation	Wenderlich, R. (2011, September 29). Introduction to A* Pathfinding. Retrieved from https://www.raywenderlich.com/3016-introduction-to-a-pathfinding.
Original URL	https://www.raywenderlich.com/3016-introduction-to-a-pathfinding
Source type	Article
Reason for interest	This is a well known path planning algorithm in the field of robotics
Notes 	- This article goes through an example of a cat trying to find a bone to explain the A* algorithm. In order to simplify the environment, the area will be turned into a two dimensional array of squares representing obstacles or free space. - There are two lists, called the open list and the closed list. The open list is a list of the squares yet to be discovered for finding the shortest path, and the closed list is a list of the squares that have been discovered. - The starting position A and is added to the closed list as it does not have to be considered. The positions of the four squares around A are added to the open list as they are currently being considered for the shortest path. The cat gives a score for each square to decide where it must go. - Each square has a G, H, and F value. The G value is the cost from the starting position to the current position, which is currently 1 (for each square adjacent to A). The H value, or heuristic, is the estimated movement cost from the current square to the goal position B. This example interpreted H as the "city-block distance." The F value is a sum of the G and H values. - The cat always chooses the square with the lowest F value. This square will be added to the closed list, and the adjacent squares of that square will be added to the open list. The adjacent squares which are also obstacles will not be added to the open list. Also, adjacent squares which are currently in the closed list will not be added to the open list again. - If all the squares in open list have the same F value, the cat will choose the most recently added square. The algorithm stops to find the final path once the goal is added to the open list.

Article #27: A* Algorithm Pseudocode

Source Title	Easy A* (star) Pathfinding
Source Author	Nicholas Swift
Source citation	Swift, N. (2017, March 1). Easy A* (star) Pathfinding. Retrieved from https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2.
Original URL	https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2
Source type	Blog Post
Reason for interest	This covers the A* algorithm with Pseudocode
Notes	The H value is the sum of the squares of the change in x and the change in y between the current and goal nodes (the Pythagorean theorem without the square root). This means that the heuristic allows for diagonal squares. The square root is unnecessary as heuristic only has to be measurable. 1. Add the starting node to the open list 2. Repeat the following until the goal is found: a. Find the current square with the lowest F value in the open list, and move it to the closed list b. For each of 8 squares (children) adjacent to the current: i. If the child square is on an obstacle or is in the closed list, ignore it. Otherwise: ii. If the child square has not already been added to the open list, add it. Set the parent of this child square as the current square. Find the F, G, and H values of the child square. iii. If the child square was already in the open list, check to see if this path to that square is better by using the G value. If the calculated G value for the child square is lower than the G value it previously had, change the parent of the child square to the current square and recalculate F and G. iv. Stop when the target square is added to the closed list which means the path has been found, or when the target square cannot be found and the open list is empty (which means there is no path) c. From the target square, find the parent of each square until the starting square is reached and save the path.

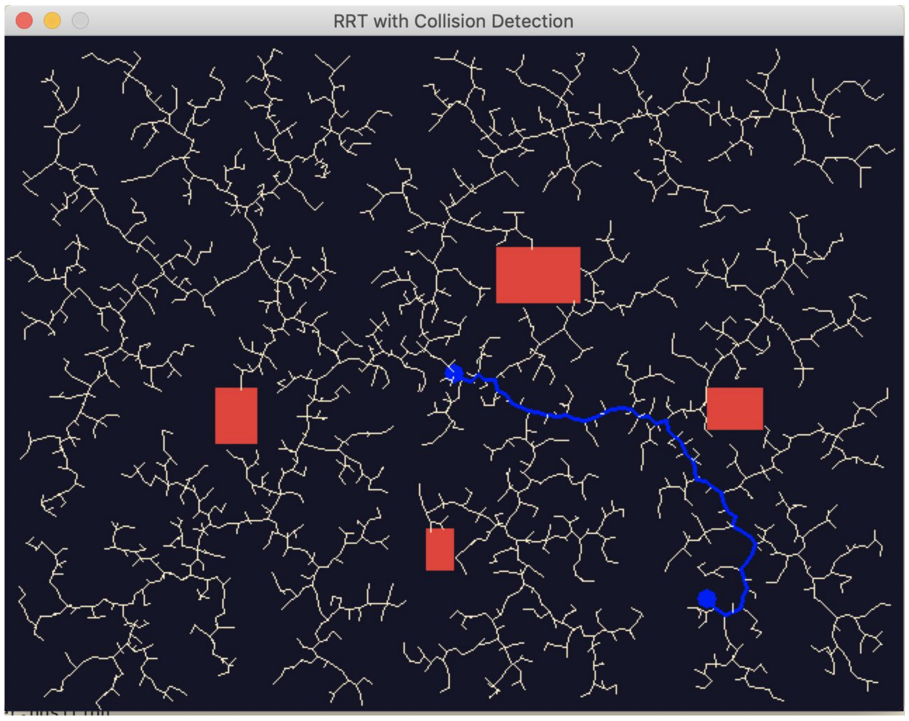
Article #28: Path Planning Overview

Source Title	Introduction to Robotics #4: Path-Planning
Source Author	Nikolaus Correll
Source citation	Correll, N. (2014, October 14). Introduction to Robotics #4: Path-Planning. Retrieved from http://correll.cs.colorado.edu/?p=965.
Original URL	http://correll.cs.colorado.edu/?p=965
Source type	Article
Reason for interest	This source gives an overview of path planning and the algorithms that can be used to solve the pathfinding problem
Notes	Path planning has various applications in autonomous mobile robots as well as in network routing, video games, and gene sequencing. It allows robots to find the optimal or fastest path between two points given the map of the environment and localization of the robot. Path planning has been used to find the shortest path of delay for packets of data. The most common method used to interpret a map to build a path is called a discrete approximation. In this method, the map is divided into chunks represented by nodes that are connected to each other by edges. An occupancy grid map is a discrete map where obstacles and free space are marked with gridded squares. In a probabilistic grid map, each square would represent the probability it is an obstacle. These maps may require large memory and take a long time to traverse given many vertices. Dijikstra's Algorithm was one of the earliest path planning algorithms. A heuristic function can speed up Dijikstra's algorithm by giving priority to nodes which are estimated to be closer to the destination. A heuristic is another word for a "rule of thumb." Heuristics can be calculated using Euclidean or Manhattan Distance. Heuristics are used in A* and D*. D* is an extension of A* which allows for part of the path to be replanned around an obstacle. However, if the graph is extremely large, A* and D* become computationally expensive. The RRT path planning addresses this problem by generating random points in the environment and connecting them to their nearest vertex. Other path planning techniques include depth first search (DFS) or breadth first search (BFS). It is important to note that path planning algorithms do not take the robot's orientation into account.

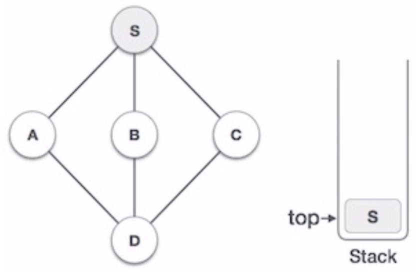
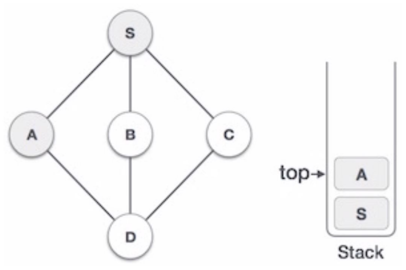
Article #29: Skydio 2 Drone Overview

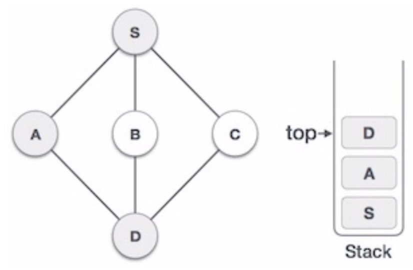
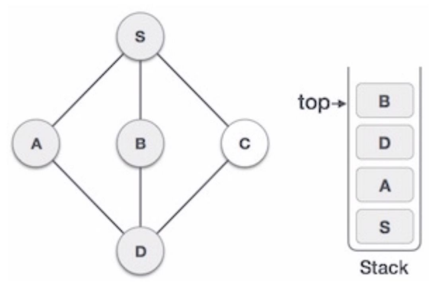
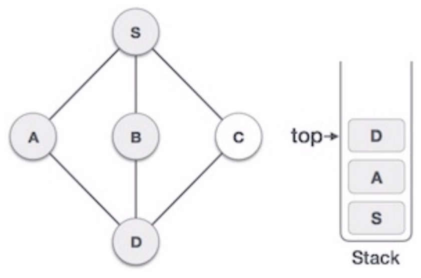
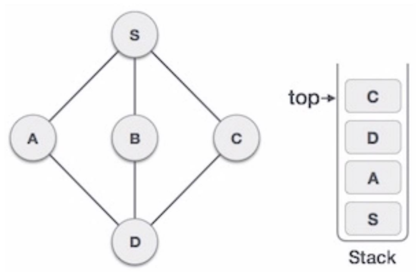
Source Title	How does Skydio 2 work?
Source Author	Skydio Inc.
Source citation	How does Skydio 2 work? (n.d.). Retrieved from https://support.skydio.com/hc/en-us/articles/360036116834-How-does-Skydio-2-work-.
Original URL	https://support.skydio.com/hc/en-us/articles/360036116834-How-does-Skydio-2-work-
Source type	Informative Article
Reason for interest	I wanted to provide some details on current drone applications used today
Notes	Skydio is an autonomous drone that can record 4K quality videos and 12 megapixel photos. It can follow a person doing an activity or a moving car.

Article #30: RRT in Python Simulator

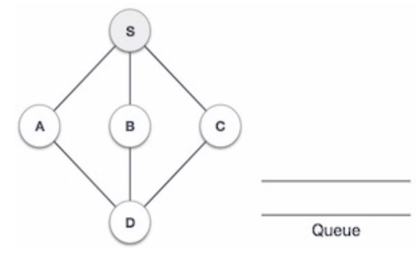
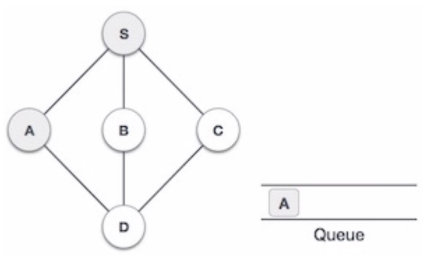
Source Title	LaValle: Motion Strategy Lab
Source Author	Steven M. LaValle
Source citation	LaValle, S. M. (2011). LaValle: Motion Strategy Lab. Retrieved from http://msl.cs.illinois.edu/~lavalle/code.html.
Original URL	http://msl.cs.illinois.edu/~lavalle/code.html
Source type	Website
Reason for interest	I wanted to visualize how RRT worked and modified Professor LaValle's code (who invented the path planning algorithm). This will be useful when I implement the RRT* algorithm on the Gazebo Simulator
Notes	The modified code is on my GitHub page: https://github.com/rumaisaabdulhai/PathPlanning/blob/master/rrt_modified.py This was the result of my modifications: 

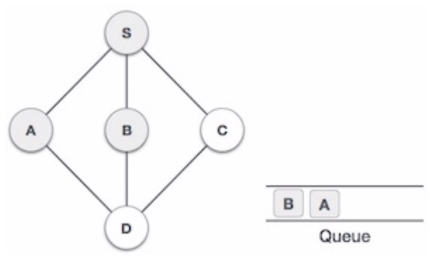
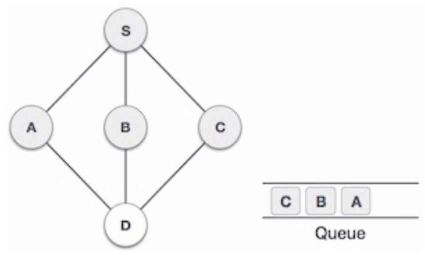
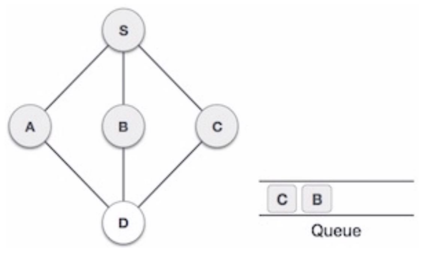
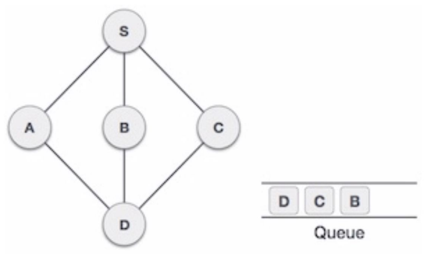
Article #31: Depth First Search (DFS)

Source Title	Data Structure - Depth First Search Traversal
Source Author	Tutorials Point
Source citation	Data Structure - Depth First Traversal. (n.d.). Retrieved from https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm.
Original URL	https://www.tutorialspoint.com/data_structures_algorithms/depth_first_traversal.htm
Source type	Tutorial
Reason for interest	This search is another path planning method
Notes	DFS is an algorithm used for traversing a graph in a depthward manner. As nodes are visited, it uses a stack to remember the next node to look for adjacent neighboring nodes that have not been searched yet. The very first step is to choose a start node. 1. Visit any adjacent unvisited node, mark it visited and place in stack. 2. If there is no adjacent node, pop a node from the stack 3. Repeat Steps 1 and 2 until the stack becomes empty  

	 S A B C D top D A S Stack  S A B C D top B D A S Stack  S A B C D top D A S Stack  S A B C D top C D A S Stack

Article #32: Breadth First Search (BFS)

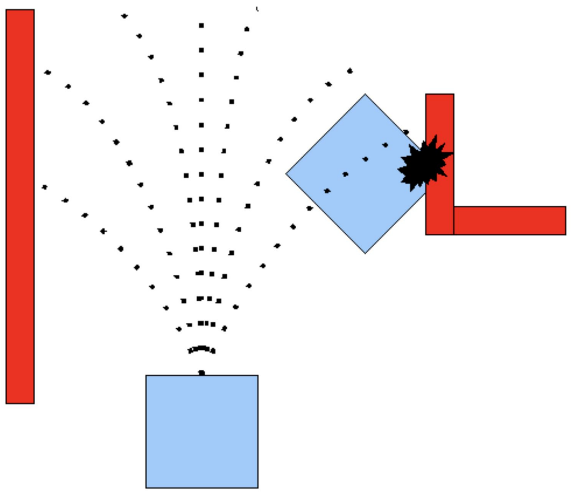
Source Title	Data Structure - Breadth First Search Traversal
Source Author	Tutorials Point
Source citation	Data Structure - Breadth First Traversal. (n.d.). Retrieved from https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm.
Original URL	https://www.tutorialspoint.com/data_structures_algorithms/breadth_first_traversal.htm
Source type	Tutorial
Reason for interest	This search is another path planning method
Notes	DFS is an algorithm used for traversing a graph in a breathward manner. As nodes are visited, it uses a queue to remember the next node to start another search. First step is to choose a start node (mark as visited and don't put in stack) 4. Visit any adjacent unvisited node, mark it visited and place in the queue. 5. If there is no adjacent node, remove a node from the stack 6. Repeat Steps 1 and 2 until the queue becomes empty  

	 S A B C D B A Queue  S A B C D C B A Queue  S A B C D C B Queue  S A B C D D C B Queue

Article #33: amcl ROS Package

Source Title	amcl - ROS Wiki
Source Author	OSRF (Open Source Robotics Foundation)
Source citation	amcl. (n.d.). Retrieved from http://wiki.ros.org/amcl?distro=melodic.
Original URL	http://wiki.ros.org/amcl?distro=melodic
Source type	Documentation Website
Reason for interest	This package is used for localization in my drone package
Notes	AMCL is a localization system used in probabilistic robotics for localizing a robot moving two-dimensionally. The amcl package is provided by ROS that can be installed onto one's local computer for use. It uses a particle filter to estimate the pose of a robot within a known map of an environment. The package only takes in laser scanned maps. The author of the package is Brian P. Gerkey. This package is part of the much larger ROS navigation package - The amcl node uses a laser scanned map, current laser scans, and transform messages and gives pose estimates - The amcl node subscribes to the scan, tf, initialpose, and map topics - It publishes to amcl_pose, particlecloud, and tf

Article #34: dwa_local_planner ROS Package

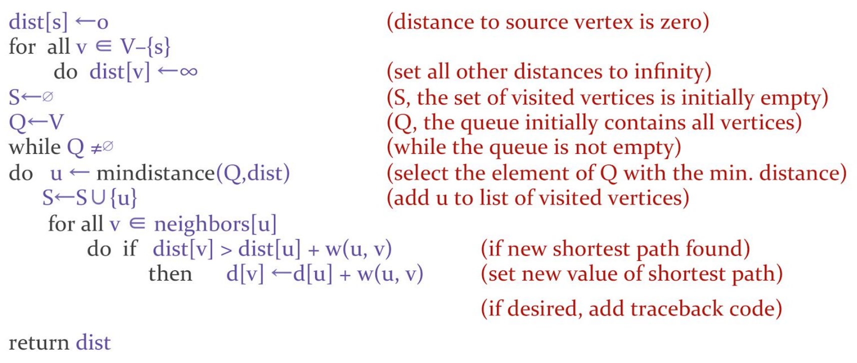
Source Title	dwa_local_planner
Source Author	OSRF (Open Source Robotics Foundation)
Source citation	dwa_local_planner (n.d.). Retrieved from http://wiki.ros.org/dwa_local_planner?distro=melodic.
Original URL	http://wiki.ros.org/dwa_local_planner?distro=melodic
Source type	Documentation Website
Reason for interest	This package is used for navigation in my drone package
Notes	The dwa_local_planner package implements the Dynamic Window Approach for navigation of an autonomous robot. The steps for the algorithm are: 1. Sample velocities in the control space 2. Predict trajectories for each sample velocity 3. Score each trajectory based on distance to goal, speed, distance to obstacles, and distance to the global path. Discard trajectories which collide with obstacles 4. Pick the highest scoring trajectory and send the velocity to the robot 5. Repeat  The figure shows the robot predicting its trajectory for each of the sampled velocities

Article #35: Dynamic Window Approach (DWA)

Source Title	The Dynamic Window Approach to Collision Avoidance
Source Author	Sebastian Thrun, Wolfram Burgard, and Dieter Fox
Source citation	Thrun, S., Burgard, W., & Fox, D. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine, 4*(1) Retrieved from https://ieeexplore-ieee-org.ezproxy.wpi.edu/stamp/stamp.jsp?tp=&arnumber=580977
Original URL	https://ieeexplore-ieee-org.ezproxy.wpi.edu/stamp/stamp.jsp?tp=&arnumber=580977
Source type	Journal Article
Reason for interest	This method is implemented in the dwa_local_planner
Notes	The dynamic window approach is a local planner used for avoiding dynamic obstacles in a mapped environment. The dynamic window is the set of velocities reachable in a short time interval. The planner only selects the velocities which allow the robot to stop safely if there is a dynamic obstacle. The optimum velocity is selected by maximizing an objective function, which considers the progress towards the goal, the velocity of the robot, and the distance from any obstacles on the trajectory. These researchers tested their robot RHINO with their dynamic window approach and found it to safely control their robot with speeds of up to 1 meter per second in different kinds of environments and avoid obstacles Indoor robots must be able to carry through with their missions and respond to sudden changes in their perception of the environment in real time. Many commercial devices today use static models of environments and so they are unable to avoid unpredictable dynamic obstacles. Dwa deals with the constraints imposed by limited velocities and accelerations.

Article #36: Dijikstra's Algorithm: Further Background

Source Title	Dijikstra's Algorithm
Source Author	Melissa Yan
Source citation	Melissa Yan (2014). Dijkstra's Algorithm [PDF file]. Retrieved from https://math. mit. edu/~rothvoss/18. 304. 3PM/Presentations/1-Melissa. pdf
Original URL	https://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf
Source type	Presentation
Reason for interest	This source goes over the background of Dijikstra's algorithm and pseudocode

Notes	● Invented by Edsger Wybe Dijkstra
	● Finds shortest paths from source vertex to all vertices in the graph
	● Works for directed and undirected graphs
	● Nonnegative weights
	● Nodes in graph connected
	
	Pseudocode:
	
	`dist[s] ←0` (distance to source vertex is zero)
	`for all v ∈ V–{s}`
	`    do dist[v] ←∞` (set all other distances to infinity)
	`S←∅` (S, the set of visited vertices is initially empty)
	`Q←V` (Q, the queue initially contains all vertices)
	`while Q ≠∅` (while the queue is not empty)
	`do  u ← mindistance(Q,dist)` (select the element of Q with the min. distance)
	`    S←S∪{u}` (add u to list of visited vertices)
	`    for all v ∈ neighbors[u]`
	`        do if dist[v] > dist[u] + w(u, v)` (if new shortest path found)
	`            then d[v] ←d[u] + w(u, v)` (set new value of shortest path)
	(if desired, add traceback code)
	`return dist`
	
	The Bellman-Ford Algorithm works with negative weights

Article #37: US Police Departments Using Drones

Source Title	Police departments are using drones to find and chase down suspects
Source Author	April Glaser
Source citation	Glaser, A. (2017, April 6). Police departments are using drones to find and chase down suspects. Retrieved from https://www.vox.com/2017/4/6/15209290/police-fire-department-acquired-drone-us-flying-robot-law-enforcement.
Original URL	https://www.vox.com/2017/4/6/15209290/police-fire-department-acquired-drone-us-flying-robot-law-enforcement
Source type	News Article
Reason for interest	I want to know how drones are being used by law enforcement currently
Notes	In 2016, at least 167 US police and fire departments purchased drones, which is more than double the amount used the previous year, according to the Center for the Study of the Drone at Bard College, NY. About 80 percent of the drones are DJI drones. Drones have helped police find suspects such as the Indiana man who ran away in a police chase in 2016. In 2017, a Maryland sheriff was able to retrieve stolen expensive construction equipment. The Oakland fire department in California piloted a drone to explore a deadly warehouse fire to look for places to use the extinguisher. The US states of Texas and California have the highest number of agencies with drones.

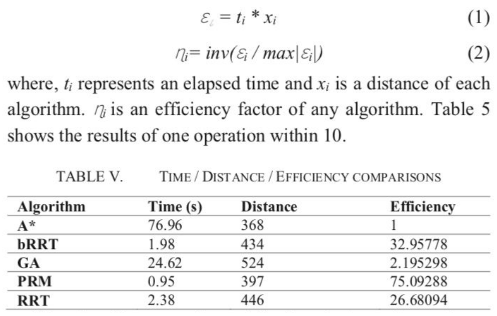
Article #38: British Police Departments Using Drones

Source Title	Most British police forces now have drones - and they're getting better at watching us. Is this the future we want?
Source Author	Cahal Milmo
Source citation	Milmo, C. (2019, September 6). Most British police forces now have drones - and they're getting better at watching us. Is this the future we want? Retrieved from https://inews.co.uk/news/uk/eye-in-the-sky-drone-capable-of-spotting-violence-in-crowds-raises-questions-about-hi-tech-policing-278914.
Original URL	https://inews.co.uk/news/uk/eye-in-the-sky-drone-capable-of-spotting-violence-in-crowds-raises-questions-about-hi-tech-policing-278914
Source type	News Article
Reason for interest	I want to know how drones are being used by law enforcement in other places
Notes	● 31 out of 45 of Britain's police forces have drones at their disposal for operations such as search and rescue ● According to Arthur Holland Michel, co-director of the New York Center for the Study of the Drone, a manned helicopter costs about $6,000 per hour to operate. But now, surveillance is available at a much lower cost and therefore can be utilized on a regular basis. ● Norfolk police in 2018 located and saved a 75 year old man who was trapped in marshland at Titchwell.

Article #39: Comparison of Path Planning Algorithms

Source Title	Comparison of optimal path planning algorithms
Source Author	Mehmet Korkmaz & Akif Durdu
Source citation	Korkmaz, M., & Durdu, A. (2018). Comparison of optimal path planning algorithms. *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*. doi: 10.1109/tcset.2018.8336197
Original URL	https://www.researchgate.net/profile/Akif_Durdu/publication/324962143_Comparison_of_optimal_path_planning_algorithms/links/5c3ed8d1299bf12be3cb43f7/Comparison-of-optimal-path-planning-algorithms.pdf
Source type	Journal Article
Reason for interest	This article compares path planning algorithms
Notes	

- A* finds the shortest path but PRM finds a path in the shortest time
- The optimal route can be found using probability or heuristics
- There are also online or offline methods
- For assigning tasks successively elapsed time matters
- Heuristic algorithm is A* which is a combination of Dijkstra's Algorithm
- The RRT is a probabilistic based algorithm
- Bidirectional RRT has two trees from the starting and goal nodes
- PRM is another probabilistic based path planning algorithm where nodes are generated randomly from the free space and then the nodes are connected to each other. This does not guarantee the shortest path.
- gmapping is a type of SLAM. Inputs are the robot odometry and sensor data and output is the map. The form of the map is an occupancy grid map
- White: free to move, black: High chance of an obstacle
- Results: PRM gives the best results in terms of shortest time and distance



$$\varepsilon_i = t_i * x_i \quad (1)$$

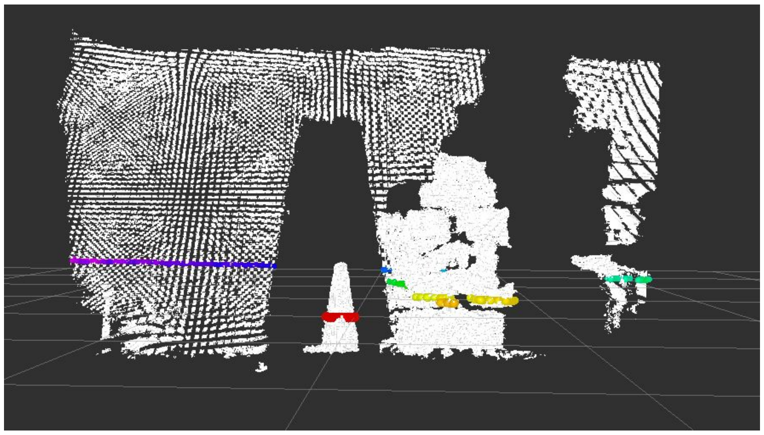
$$\eta_i = inv(\varepsilon_i / max|\varepsilon_i|) \quad (2)$$

where, t_i represents an elapsed time and x_i is a distance of each algorithm. η_i is an efficiency factor of any algorithm. Table 5 shows the results of one operation within 10.

TABLE V. TIME / DISTANCE / EFFICIENCY COMPARISONS

Algorithm	Time (s)	Distance	Efficiency
A*	76.96	368	1
bRRT	1.98	434	32.95778
GA	24.62	524	2.195298
PRM	0.95	397	75.09288
RRT	2.38	446	26.68094

Article #40: Depth Image to Laser Scan

Source Title	depthimage_to_laserscan
Source Author	OSRF (Open Source Robotics Foundation)
Source citation	depthimage_to_laserscan (n.d.). Retrieved from http://wiki.ros.org/depthimage_to_laserscan.
Original URL	http://wiki.ros.org/depthimage_to_laserscan
Source type	Documentation Website
Reason for interest	This is a package I will be using to convert depth information from a stereo camera to a laser scan to map the environment.
Notes	The package is basically a node that converts depth images from a depth camera and outputs a 2D laser scan. The package subscribes to a topic similar to the name /camera/depth/image_raw and publishes to the desired topic with the laser scan data of type LaserScan.  This image shows the point cloud data and the laser data projected on top of it. The package can work with many RGBD sensors, including the microsoft kinect and intel realsense depth cameras. The author of this package is Chad Rockey. There is another package called pointcloud_to_laserscan which also accomplishes the same task but with an input of type PointCloud2.