Simulation of an Indoor Autonomous Drone for Assisting Police in Mass Shootings

Massachusetts Academy of Math and Science

Rumaisa Abdulhai

rabdulhai@wpi.edu

Drones and Autonomous Robotics Systems

Literature Review

I. Introduction

The field of robotics consists of computer-controlled electromechanical systems using various sensors to navigate the environment and manipulate objects around them. These robotic systems are utilized in a variety of fields, from assembly lines and medical surgery to space exploration and self-driving cars. As automation becomes more entrenched in humans' lives, the perceived potential of these robotics systems has shifted. Assembly lines, once thought to be a radical invention, are much more predictable compared to the current applications of robotics systems such as fighting fires, where they are expected to perform actions in highly uncertain and uncontrolled environments. Sensor measurements become key in such situations and must be utilized to take decisive action under uncertainty in uncontrolled environments.

With the advent of autonomous drones or autonomous unmanned aerial vehicles (UAVs), there have been various outdoor applications in fields such as agriculture, cinematography, and delivery. Drones have reduced the time for crop and soil inspection, and have made the process of inspection more consistent. One drone, namely the Agras DJI octocopter, can spray fertilizer with its 4 nozzles on up to 6000 square meters of land in 10 minutes (Puri, Nayyar, & Raja, 2017). Drones are also being used for delivering medical aid. The Zipline Company uses drones to deliver blood packages to hospitals in Rwanda. In addition to cutting normal delivery time from hours to minutes using GPS technology, the zipline drones have also saved the lives of patients who suffer from blood loss during surgery (Ackerman & Strickland, 2018). The true feat of autonomous drone technology can be seen in the new Skydio 2, which can follow an individual participating in extreme outdoor sports at high speeds while avoiding obstacles and recording high-quality footage at the same time (Skydio Inc, 2019). Autonomous drones are making an enormous impact in the agriculture and the cinematography industry and are expected to contribute to other domains including fighting urban and forest fires, law enforcement and safety inspection of national infrastructure such as roads, dams, and bridges.

While there have been many outdoor drone applications, building an indoor drone application has unique challenges: the drone must navigate through narrow and tight spaces while avoiding obstacles and fly in a GPS-denied environment (Khosiawan & Nielsen, 2016). Therefore, there must be an alternative to GPS. SLAM, or Simultaneous Localization & Mapping, is essential for indoor navigation. SLAM is the methodology of estimating the location of a robot in an environment while trying to form a map of the environment, using sensors such as lasers, monocular cameras, or binocular cameras (Li, Bi, Lan, Qin, Shan, Lin, & Chen, 2016). Once a map is established and the drone is aware of its position in the environment, a path planning algorithm is necessary for the drone to efficiently navigate autonomously from a source point to a destination point while avoiding any obstacles in its path.

This project focuses on building an autonomous indoor application to assist police during mass shootings. The project will explore the use of autonomous indoor drones in providing effective mechanisms to collect real-time information

about the precise location and perpetrator(s) of a mass shooting in order to significantly reduce the loss of life. The goal of this project is to simulate mapping, path planning, and navigation of an indoor drone to a point of emergency, collect real-time information about the location of the perpetrators involved and relay it to a trusted group of individuals such as law enforcement who then can take appropriate action to save precious human lives. The literature review will primarily cover the development of localization, mapping, and path planning methods and algorithms to date in the field of autonomous robots and drones. It will also cover the usage of various sensors used in terrestrial and aerial robots. The following provides an organization of this report. Section 2: Interfaces and Programs, provides a brief survey and overview of the software components that could potentially be used in this project. Section 3: Sensors, surveys the various sensor devices that are used in mobile robotics currently. Section 4: Probabilistic Robot Interaction, reviews the fundamental principles and filters used in robotics that is relevant to this project. Section 5 surveys the path planning algorithms used in the mobile robotics arena. Section 5 finally provides conclusions and pointers for future planned work.

II. Interface and Programs

This section gives an overview of the types of drones and the software platforms that are commonly used in research of autonomous vehicles that could potentially be used in this project.

A. Research vs. Commercial Drones

Drones can be classified into two broad categories: commercial and research drones. Commercial drones are off the shelf drones that are to be used for the express purpose for which they were built, such as aerial photography or spraying fertilizer. Commercial drones lack the processing power or the software development platform to modify them for other purposes. In addition they have sensors limited to the application for which they are built. Research drones on the other hand have large computing power, wide variety of sensors and a rich development environment that allows developers to build and test many different autonomous drone applications. The other major difference between a commercial drone and a research drone is that a research drone has an onboard computer that makes decisions during navigation in real time while most commercial drones are remote controlled. Research drones are able to process computation-heavy algorithms on board and utilize the information gained to perform complex tasks. Research drones are also customizable with the addition of various sensors such as stereo cameras, laser sensors, or ultrasonic sensors. Hence, research drones are better suited for developing indoor drone applications than commercial drones (Guermonprez, 2017). A basic research drone model from the hector_quadrotor package provided by the Robot Operating System (ROS) will be used in the first phase of this project to test the flight of an autonomous drone.

B. Gazebo Simulator

Gazebo is a well-known industrial software platform for developing realistic 3D simulations of robots in both indoor and outdoor environments. It provides support for real sensors on the market such as monocular or stereo cameras, and 2D or 3D lasers as well as a visualization environment to observe the navigation of an autonomous drone/robot (OSRF).

The gazebo simulator allows one to create virtual custom objects, robots, drones and environments in which they interact. Using the basic cylinder, sphere, and cube shapes provided, various indoor environments such as a school hallway or a hotel lobby can be constructed. One can also choose to import standard models of environments or robots provided by Gazebo rather than build everything from scratch. Some examples of these models include gas stations, walls, warehouse robots, and apartment buildings. Using the link inspector in the model editor, the visual appearance, size, and orientation of objects can be changed. The object can appear to be large but only a small part of it is real in the environment.

One can also choose how to view the world by changing the perspective of the camera. There is also a cube icon that allows one to view their world from common perspectives such as top view, front view, right view, and side view. Meshes are used to insert custom 3D objects that have complex size measurements and are not necessarily defined by a certain width, length, and height. Joints are used to connect objects in a custom model. There are various types of joints provided in the Gazebo simulator, including fixed, revolute, and screw joints. An object can turn into a real sensor using plugins. There are plugins for specific camera and laser models available online. For example, one can use a Kinect depth sensor plugin to record depth information about its surroundings (OSRF).

All of these features of Gazebo help developers to build custom indoor and outdoor environments such as an office building or an outdoor amusement park. This project will use Gazebo to build an indoor school environment to simulate and test an autonomous drone application that can help police during mass shootings.

C. Robot Operating System

The Gazebo Simulator is highly integrated with another software platform called the Robot Operating System, otherwise known as ROS. ROS, supported by the Open Source Robotics Foundation, provides the infrastructure for building autonomous robots. It is a powerful tool for autonomous robot programming as it is open source and applies to multiple programming languages including Python, Java, and C++. ROS also makes available various libraries for building and writing code. With a subscriber/publisher system, multiple parts of the robot can listen to sensor output by subscribing to topics where the information is published, and can take appropriate action based on these measurements (OSRF). This project will utilize ROS as the primary development environment to build an autonomous drone application.

III. Sensors

This section provides a survey of the most important sensor devices used in the field of autonomous robots. Sensors are devices that enable a robot or drone to figure out the state of itself and the objects around it using processes such as mapping and obstacle avoidance. In a way sensors are the eyes, ears, and the skin of the robot or drone that provide the vital capabilities of sight, hearing, and touch to discover the objects in the environment. Sensors could be as simple as a touch switch that senses whether a door is open or closed or a more complex device such as LIDARs (Light Detection and Ranging) that are used by self driving cars to know the exact shape and size of the objects on the road. Some sensors such as cameras are passive, which means they do not emit signals but rather take screenshots of the environment. Active sensors such as Sonars (Sound Navigation and Ranging) emit sound pulses and then detect its echo to locate the distance to obstacles. Sensors can be broadly classified based on the signals they use: ultrasound, electromagnetic, or vision (Matarić 2008).

A. Ultrasound sensors

Ultrasound sensors, otherwise known as sonar sensors, use sound waves in excess of human audible frequencies (above 40KHz) to detect objects that could be at a distance of about 1-100 feet using the principles of echo and reflection of sound. Sonars have found use in medical imaging (fetal imaging) as well as mobile robotics to detect obstacles. Although ultrasound sensors are relatively inexpensive, the distances calculated are not very accurate as the sound waves could suffer specular (mirror like) reflection from smooth surfaces and the emitted sound pulses may not return to the robot. This drawback is avoided by modifying the surfaces of objects in the environment and make them rough to cause proper bounce back of sound signals. The other approach called active perception in an effort to make ultrasound sensors more accurate is for the robot to move around and make several measurements at different angles and take the average of distances. Despite their disadvantages, ultrasound sensors have found successful applications in outdoor robots to map complex surroundings as they are affordable. Their applications in indoor drone applications is somewhat limited due to specular reflection (Matarić 2008).

B. Electromagnetic sensors

Electromagnetic sensors use radio, infrared or visible electromagnetic waves for object detection, navigation, and mapping. When used in the visible spectrum they are called lasers. Unlike ultrasound sensors, lasers do not suffer from the drawback of specular reflection and can map their surroundings and detect obstacles to a very high degree of accuracy down to millimeters (mm). However, lasers cannot detect very close objects as it cannot distinguish between emitted and reflected

waves due to the high speed of light. To overcome this deficiency, phase shift measurements are used to estimate the distance to the obstacle instead of using the speed of light. Lasers are relatively expensive when compared with ultrasound sensors. LIDARs which also use light waves and follow similar principles as lasers are even more expensive and could also be bulky. So LIDARs are used in high demanding applications such as terrestrial robots and to a limited extent in drones because of their very high accuracy and range of perception (several hundred meters). In the preliminary phase of this project, a model of the drone with a LIDAR sensor is used in Gazebo and ROS to map out the initial environment as it is readily available as a software package (Matarić 2008).

C. Visual sensors

Cameras are passive sensors that fall into the category of visual sensors as they do not emit a signal but take pictures of the environment from different angles to estimate the edges of obstacles in order to avoid them. Camera based sensors could be monocular (one camera) or binocular (two or more cameras). Binocular vision sensors are also called stereo cameras. While monocular camera sensors provide planar images, stereo cameras have depth and thus are able to construct 3D images. The basic principle of camera sensors is to take a number of pictures (frames) successively at different angles and reconstruct a 3D picture with edges of objects in the environment. In addition to the objects in the field of vision the images are also corrupted with usual noise and shadows. Special processing is needed to remove these corruption of images by a process of convolution. Convolution is applying filters to images to remove noise or other corruption such as shadows. Camera based sensors are relatively inexpensive similar to ultrasound sensors but the downside is they require very high signal processing as they have to process potentially hundreds or thousands of frames of images in real time. So the robots using them must be equipped with high computing power such as a special purpose GPU (Graphics Processing Unit) from either Intel or Nvidia. Although LIDAR sensors are good to use in the initial phase of this project, stereo based camera sensors are more practical to employ in drones due to their ubiquitous availability and cost (Matarić 2008).

IV. Probabilistic Robot Interaction

This section reviews the fundamental principles, notation and filters used in robotics that is relevant to this project. Sensor measurements, control actions, as well as states (for example position and pose) of the robot have unavoidable uncertainty in them and therefore the mathematics of probability plays a huge role in robotics. Sensor measurements, control actions, and states of robot are all modeled as random variables in robotics and have probability density functions that allows one to estimate the probable pose of the robot or drone.

A. State Estimation and Control

State is a collection of information about the robot as well as the environment. Dynamic state includes the parameters that change as the robot moves around. 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 state of the robot is updated at discrete time steps using an important principle known as the 'Markov property.' The Markov property states that given the current state x_t of a robot and all its previous states x_{t-1}, x_{t-2} . . . the next state x_{t+1} depends only on the current state x_t and is independent of all the previous states. This simple principle allows us to calculate the future states of a robot rather accurately using the theory of conditional probability.

The first type of interaction a robot can have with the environment is to take a sensor measurement, 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, the mere fact that time has elapsed, is considered to be a control action and the robot state has to be updated. An odometry sensor can be used to collect control data about the revolution of a robot's wheels that will also result in the state being updated. The state (x_t), control action (u_t) and sensor measurement (z_t) are the primary random variables that are used in the methods and algorithms of probabilistic robotics.

B. Bayes Filter

The autonomous robot or drone needs to estimate its position and orientation in an environment in order to execute its task successfully. The Bayes filter is a method or algorithm used to compute the position and orientation of the robot or other objects in its environment. The belief of a robot, describes the set of probabilities about its current state in an environment. The belief at time t can be noted as bel(x_t). At the beginning of time step t, the robot is in state x_{t-1}. During 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 (Thrun, Burgard, & Fox, 2010). This basic sequence of taking first an action followed by a sensor measurement at every time step is how the robot updates its own state and the state of its environment. These sequence of steps are combined in a basic algorithm known as the Bayes Filter given in Figure 1. The Bayes Filter is one of the most essential concepts that forms the basis for many other filters and localization algorithms in the field of robotics.

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 $bel(x_t)$

Figure 1: Pseudocode for Bayes Filter (Thrun, Burgard, & Fox, 2010)

The goal of the Bayes filter is to calculate the current belief of an autonomous robot/drone based on the previous belief and current measurement and control data. The steps in Figure 1 are briefly explained below for any time step *t*:

1. This step just defines inputs used by the Bayes filter. The previous belief bel(x_{t-1}), control action u_t the robot intends to take and the acquired measurement z_t are used as inputs.
2. This step initiates a for loop for all possible values of the state x_t
3. The robot takes action u_t at x_t. The input's prior belief, bel(x_t - 1) and control action u_t are used to calculate the intermediate belief.
4. After taking the sensor measurement z_t, the robot re-calculates the final belief at x_t.
5. This step ends the for loop.
6. The new calculated belief becomes the previous belief for the next time step t+1 and the algorithm continues until the robot reaches its intended goal.

The Bayes filter forms the fundamental basis for all other filter approaches and is therefore applicable broadly. The Kalman filter is an extension of the Bayes filter for continuous state spaces with the assumption that the state space is normally (Gaussian) distributed.

C. Simultaneous Localization and Mapping

SLAM is the method for building a map of the indoor environment while tracking the robot's pose in real time using sensors such as lasers or cameras. Real-time processing allows for information to be collected from camera or laser positions and estimate the pose rather accurately. The first research on SLAM was conducted in 1986 on terrestrial robots with lasers by Smith and Cheeseman (Kudan 2016). In Visual SLAM, stereo cameras or a combination of monocular cameras are used to map the environment. Visual SLAM works by tracking a set of features or points in the environment through successive camera frames and then 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. This project will use the gmapping package provided by ROS which implements SLAM.

V. Path Planning

This section reviews the various path planning algorithms used in the field of robotics. The general robotics path planning problem consists of a robot trying to find the optimal or fastest path between two points given a map of the environment and the robot's localization. Path planning has various applications in autonomous mobile robots as well as in network routing, video games, and gene sequencing. The same method is also used to find the shortest path for data packets on the internet.

For path planning algorithms to execute, the environment's map must be interpreted properly. The most common method used to build a map and interpret it is called discrete approximation. In this method, the whole environment is divided into a square grid. Each square within the grid is called a node that is connected to other squares (nodes) by edges. The resulting map known as an occupancy grid map is a discrete map where obstacles, free space, and unexplored space are marked with separate squares. The free space is designated with white squares, the obstacles space is designated with black squares, and the unexplored space is marked with gray squares.

There are two broad categories of path planning algorithms; grid/graph-based planning and sample-based planning algorithms. Algorithms such as Dijkstra's, A*, and D* fall into the grid-based category, while Rapidly Exploring Random Tree (RRT) and its variations fall into the sample-based category. Dijkstra's Algorithm is one of the earliest path planning algorithms (Correll, 2014). Invented by Edsger W. Dijkstra, it finds the shortest path between any two nodes on a graph of nodes connected by weighted edges (Yan, 2014).

A. A Star Algorithm

A* also finds the shortest path, but usually between two points on a coordinate grid. The algorithm uses two lists to keep track of visited and unvisited nodes, with the starting node in the visited list. Adjacent obstacle-free nodes are added to the unvisited list. For each adjacent or child node of the current node, the sum of the g and h values are calculated to return the f value. The g value is the distance from the starting to the current position. The h value is the heuristic, which is the estimated cost from the current to the goal position (Wenderlich, 2011). The node with the minimum f value becomes the next current node and is added to the visited list. The algorithm continues until the goal node is added to the visited list. The path is generated by referencing the parent of the current node until the starting node is reached (Swift, 2017). D* extends on A* in that it allows part of the path to be replanned around an obstacle. However, if the graph is extremely large, A* and D* take much longer due to the high calculations required. This weakness in grid-based planning is addressed in sample-based planning algorithms such as RRT (Correll, 2014).

B. RRT Algorithms

The Rapidly-exploring Random Trees (RRT) Algorithm, invented in 1998 by LaValle & Kuffner, is an online

method of building a tree to explore a region of free space (Williams & Frazzoli, 2010). The RRT Algorithm works by randomly generating points in the free space and trying to connect each point to its closest neighbor point. 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 lies in a certain radius of the goal, the algorithm finishes execution. Although the algorithm is simple to implement and takes a very short amount of time to execute, RRT does not find the shortest path. Its graph is very cubic and makes irregular paths (Chin, 2019).

The RRT* algorithm was made to fix this problem. RRT*, developed by Dr. Karaman and Dr. Frazzoli, builds on top of RRT to find a shortest path. The RRT* Algorithm has two additions to the RRT Algorithm. The first is 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 than the RRT algorithm's graph (Chin, 2019). Figure 2 below demonstrates the RRT algorithm in action in the presence of obstacles in a python simulator. For this sample implementation the original open source code, which simply generated random points in the sample space, was used and modified to work with obstacles and a node class which draws out the final path. The expanding tree of nodes is shown in white, the obstacles are shown in red, and the highlighted blue line is the path found in the presence of obstacles.

In the first phase of this project, Dijkstra's algorithm is expected to be used for path planning. In a later phase other specialized path planning algorithms like RRT or A* are expected to be used.

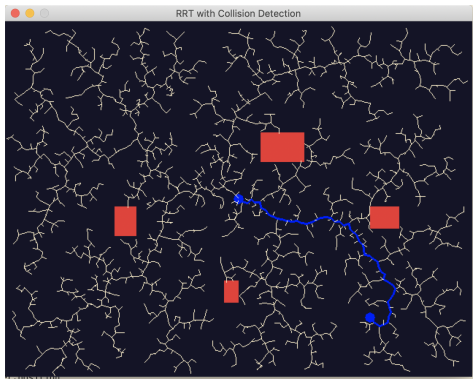

Figure 2: RRT Algorithm in action with obstacle avoidance (Abdulhai, 2019)

VI. Conclusion

This literature review has covered the software modules, sensors, localization, mapping, and path planning methods and algorithms that are used currently in autonomous robots and drones with a goal to develop a new autonomous drone application that can help police during mass shootings. Various sensors such as ultrasound, electromagnetic and

vision sensors have been surveyed along with their pros and cons. In the field of mapping the fundamental filter called the Bayes Filter has been introduced. Other filters such as Kalman filters which are extensions of Bayes filters, are also planned to be investigated for use in this project. The most common path planning algorithms such as Dijkstra, A*, and RRT have been surveyed. In the first phase of the project, an initial layout of a simple indoor environment will be designed in Gazebo. The SLAM gmapping package will be used to localize the robot and map the environment using a laser sensor. A model drone is planned to be imported into gazebo and then used to navigate from one point to another in the environment autonomously using the Dijkstra's path planning algorithm. In the second phase, the indoor environment will be enhanced to resemble a typical school hallway. Next a more complex path planning algorithm such as A* or RRT will be implemented for navigating the robot autonomously between two points in the indoor environment.

VII. References

Abdulhai, R. (2019). rumaisaabdulhai/PathPlanning. Retrieved from https://github.com/rumaisaabdulhai/PathPlanning/blob/master/rrt_modified.py.

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

Chin, T. (2019, February 26). Robotic Path Planning: RRT and RRT*. Retrieved from https://medium. com/@theclassytim/robotic-path-planning-rrt-and-rrt-212319121378.

Correll, N. (2014, October 14). Introduction to Robotics #4: Path-Planning. Retrieved from http://correll. cs. colorado.edu/?p=965.

Guermonprez, P. (2017, November 13). Autonomous Drone Engineer - A1 - Intel Aero in 5mn. Retrieved from https://www. youtube. com/watch?v=7t7l885g8dI.

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

Kudan. (2016, May 23). An Introduction to Simultaneous Localisation and Mapping. Retrieved from https://www. kudan. io/post/an-introduction-to-simultaneous-localisation-and-mapping.

Matarić Maja J. (2008). *The Robotics Primer*. Cambridge, Mass: The MIT Press.

Melissa Yan (2014). Dijkstra's Algorithm [PDF file]. Retrieved from https://math. mit. edu/~rothvoss/18. 304. 3PM/Presentations/1-Melissa. pdf

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*.

OSRF. (n. d.). Beginner: Overview. Retrieved from http://gazebosim. org/tutorials?tut=guided_b1&cat=.

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

ROS Wiki. (n. d.). Retrieved from http://wiki. ros. org/ROS/Introduction.

Skydio Inc. (2019). How does Skydio 2 work? (n. d.). Retrieved from https://support. skydio. com/hc/en-us/articles/360036116834-How-does-Skydio-2-work-.

Swift, N. (2017, March 1). Easy A* (star) Pathfinding. Retrieved from https://medium. com/@nicholas. w. swift/easy-a-star-pathfinding-7e6689c7f7b2.

Thrun, S. , Burgard, W. , & Fox, D. (2010). *Probabilistic Robotics*. Retrieved from https://docs. ufpr. br/~danielsantos/ProbabilisticRobotics. pdf

Wenderlich, R. (2011, September 29). Introduction to A* Pathfinding. Retrieved from https://www. raywenderlich. com/3016-introduction-to-a-pathfinding.

Williams, B. , & Frazzoli, E. (2010). *16. 410 Principles of Autonomy and Decision Making.* Massachusetts Institute of Technology: MIT OpenCourseWare. Retrieved from https://ocw. mit. edu.