Robotic Localization
Intro
Welcome to week four of the Estimation and Learning module in
the Robotics specialization. This week, we'll look at localization
algorithms for robotics. First of all, we'll explore how robots
represent the pose, that is where they're located and how they can estimate this
over time using a localization algorithm. Then, we'll see how, given a map,
we can use the information in the map to help us build up this
registration of the pose to the map. Finally, we'll explore a canonical
algorithm called the particle filter which robots can use to estimate their
localization information using sensors. In summary, this week you will learn how
robots can keep track of where they are in space and time using these particle filter
algorithms to represent their uncertainty.

4.1 Odometry Modeling
This week we will learn about
self-localization techniques including the particle filter. In this first lecture,
we will consider models for odometry as a first order
approximation to the robot's location. As in your car, where the odometer
records how many miles you have traveled, odometry provides a measurement
of how far the robot has moved. Odometry is just one method of finding
the robot's location in the world. If we look at a typical
application of localization, car navigation,
we see several ways to find location. Information sources include GPS, global
positioning system, cellular
networks, and Wi-Fi access points. Each of these sources, however, have certain levels
of noise
that lead to various levels of accuracy. Driverless cars, for instance, will need better than
3.5 meters
of accuracy that the GPS provides. That error is the difference between
occupying the sidewalk and the road. The previous sources represent global
knowledge of position, exact coordinates. Odometry and other sources of
information can augment the global localization sources with local knowledge. How
have they changed coordinates? These sources of information are more
precise, giving centimeter accuracy. However, integrating sources,
like encoders and gyroscopes, over time can lead to drift. This is due to the
accumulation
of errors in time. Errors from slippage of the wheels
deceive the encoder for instance. Other local sensors like
laser scanners and color and depth cameras can help

to correct these errors. We will see how this incorporation
happens later in the week. Odometry updates start
with modeling the robot. Different robots, such as humanoids or aerial vehicles,
will require different models. In our case, we will model
a skid steer four-wheeled robot. The odometry measurements come
from ticks from the encoder that measure how much the wheels
have rotated in a given timeframe. These ticks can be mapped into translation
and rotation of the body of the robot. First, let's explore the rotation
odometry calculation. With a skid steer robot, the left and right sets of wheels
are controlled independently. When turning,
these two sides form the inner and outer radii of circles that
share the same center. Coupled with the knowledge of
how wide the robot vehicle is, we can determine a change in angle
based on these encoder ticks. First, we want to translate motor ticks
into meters traveled by the inner and outer wheels along their respective arcs. This conversion requires
knowledge of the wheel sizes. Here, these measurements in
meters are denoted eo and ei. The inner and outer arcs are known, but they also share the same
angle of rotation. With knowledge of the width of the robot, we can use the difference in arcments
to calculate the shared angle data. Next, we will consider
the translation of the robot. Conveniently, the translation requires
knowledge of the rotation that we have already calculated. In measuring translation, we can form a
triangle with the known angle of rotation. We then can average
the change in position for both the inner and outer wheel sets to
find the change in the x direction. The change in the y direction
requires a similar methodology. For small movements, this is a good
approximation for the translation. Unfortunately, the encoder measurements
can be noisy due to wheel slippage. Angular estimates then propagate
errors into the translation estimates. One solution to this problem
is to utilize the gyroscope to find a more precise
measurement of angular change. For a small number of time steps,
the gyroscope can be very accurate. Thus, angular odometry is measured
solely by the rate of change observed by the gyro,
integrated over time. This measurement aids in
translation calculations as well. This simple odometry
approach to localization requires a frame of reference for
where the robot began its trip. Local measurements from the encoders and

gyroscopes still provide noisy estimates. So we want to include more measurements to correct errors. The next sections will discuss using maps to aid in localization correction, as well as ways to probabilistically define our localization state.

4.2 Map Registration

In this lecture, we will consider
correlation based matching strategies for location a robot on a map
given laser range data. This map registration process
provides a very precise complement to odometry based localization. First we should introduced
the LIDAR depth sensor. LIDAR stands for
light detection and ranging and it provides distance measurements. Often engineered in a laser scanner
to provide two dimensional data. The laser scanner we will
model in this lecture takes depth measurements
in polar coordinates, where a continuous distance reading
r is made at discrete angles theta. Here, theta encompasses 270 degrees,
not a full circle. The laser scanner can only
see 10 to 30 meters away. In this range restriction,
means that distance measurements showing here is black dots, can only
be found within the area in green. Thus, due to the rays generating from
a single point and the limited range, the robot can only see the dotted
lines and not the lines in brown. Just as in the previous lecture,
a two dimensional occupancy grid map will be used in localization,
where a light colored cell represents high probability of an obstacle and a dark
colored cell present a low probability. The cells here are meant
to replicate the laser skin shown in the previous slide when the robot
is approaching a corner in a hallway. Because the robot lives in a finite grid
world the grid must sometimes be expanded as the robot can escape the boundaries. In this case, the map representation
should increase in size as the robot turns and travels on the corridor
shown in the top left of this map or else information will be lost. In addition to mapping the laser
data discussed in week 3, we can access map data and try to find the robot
pose in the map given the laser data. The complimentary stages of mapping and
localization when performed together are known as SLAM,
simultaneous localization and mapping, which is a major

research topic in robotics. In the localization problem we have two sets of information. First, the occupancy grid map provides a grounds truth knowledge of what the robot should expect to observe in the world. Second, the set of lighter scan measurements provides information on what the robot is observing at the current time. The lighter scan measurements must be discretized according to the map representation, as discussed in week three, in order to be compared to the information from the occupancy map. With these two pieces of information the goal is to find the best robot pose on the map that explains the measured observations. Searching over all possible poses of the robot can be difficult. But based on the odometry information discussed in the last lecture, we have some tricks to make the search easier. We can constrain the search to a limited number of poses based on odometry information. Because we track the robot over time, we have the last known position of the robot and odometry information on how far the robot most likely moved. Thus, the most likely pose for the robot is now given a new set of laser data, is probably close to where the odometry predicts the robot to be. This prediction means that we can refine our search to poses near the prediction and be more confident in the validity of our search results. We measure each pose p in the search based on a map registration metric. One metric is to consider the sum of the map values m, at coordinates x and y, where the laser returns r, hit. This correlation metric can be modified to suit the application at hand. In our case, the value of our map cell will be a log odds ratio, so laser returns that are seen at a map location with high probability of occupancy will strongly increase the registration in the metric score. Laser returns with map locations known as free cells will decrease the metric score. Additionally, the correlation can be scaled where returns from far distances affect the metric calculation less than nearby the laser returns. We register the robot on the map, at the pose that maximizes the registration metric. Thus, when the odometry is calculated, it uses this pose to predict a new position of the robot, in time. In addition to considering merely the laser returns, we can consider points for the laser returns penetrated. This calculation can further corroborate our map registration. It requires considerably more computation however. To capture pose uncertainty using

a simple Gaussian on position and angle may not provide a feasible approach. In the next lecture, we will present
a pose filter that can capture bi-modal uncertainty and non-linear models and
a computationally tractable way.

4.3 Particle Filter

In this lecture, we will talk about
a probabilistic state estimation technique using a sampling-based distribution representation known as the Particle Filter. Instead of a fully defined function, the Particle Filter represents a distribution with a set of samples, referred to as particles. These particles represent the distribution. The statistics of the samples match the statistics of the distribution, such as the mean or standard deviation. However, they can be more
complicated metrics as well. In this way, there are no parameters as were seen in the mean and covariances of the Gaussian models. Instead, a full population is tracked. In essence, the particle filter population represents a mixture of Gaussian distributions that we have seen in the first week. Here, the variance will go to 0. With 0 variance, the Gaussian distributions
become Dirac Delta functions. Initially, a set of particles represent the underlying belief state. Each particle is a pair of the pose and the weight of that pose. This is similar to representing a probability function where the weight is the probability of that pose in the underlying distribution. Here, darker colors represent higher weights, and lighter colors represent lower weights. Just like the Kalman filter, a motion model will move
the underlying distribution. Here, the particles move based on odometry measurements taken from the robot. A companion uncertainty model captures the noise underlying the motion model. For instance, this could be wheel slip or friction changes. In the particle filter,
where we do not track the motion model in explicit parameters, we add sampled noise from the motion noise model. In this case, we use a Gaussian distribution to model
noise with 0 mean and non-0 covariance. Noise is uniquely added to each particle. So separate samples are made for
each particle. After the noise is added, the dispersion of the particles captures the uncertainty due to movement. Like the Kalman filter,
we can use a separate set of observations to constrain our noise and update our belief distribution. Here we will leverage

the LIDAR correlation from previous lectures
on map registration. We will update the weights of the
particles to reflect the correlation score from the map registration by utilizing
the current weights as a prior belief. The new set of particles captures
the distribution after odometry and sensor measurements. However, this may not be the optimal
set to represent the distribution. Here, you can see that only a few
particles have significant weights. Most of the particles
are lightly colored and do not give much information
about the distribution. To make the set of particles
more accurately represent the belief state distribution, we check
the number of effective particles. The number of effective particles
acts as a criterion for when to resample particles. This resampling process provides
a probabilistically motivated way to prune out lower weighted particles. With the set of large and small weights, using the cumulative
probability function can aid in sampling. With normalized weights,
the sum of the weights is 1, and can be represented as a monotonically
increasing cumulative function. We sample a number ,uniformly,
between 0 and 1 of the cumulative range and
find which weight includes that number. The particles with the indices
found in the resampling approach become the new set of particles to be
fed into the next odometry update. Particles may be duplicated, but the odometry noise will
differentiate these particles. This approach provides a good way to
approach a multi-nodal belief state distribution and
non-linear effects of your motion model.