Lab3 Lecture Notes

Sensors and Filtering

Challenges: a sensor does not give us continuous measurements, when you want measurement you have to make it. When do we make it? Things happen in the world at arbitrary times that computer isn’t aware of. We have to make measurement at right time so that we can see a particular measurement taking place. The measurement a sensor takes isn’t actually part of a property of the world.

Lab 3 where is the robot relative to the world that it lives in. We have 1 sensor that is able to figure out the distance to surfaces in its field of view and 2 another that looks over small area and measures the intensity of the underlying surface. The ultrasonic sensor will be used to figure out how far we’re from walls, which gives us an idea of how we’re oriented with respect to plainfield. Color sensor will be used to look for lines on the floor.

When to make measurement. How do we treat the signal.

Lecture: how do we detect a line. Find the black line on a white background. We see a number in 0 to 1K as we are moving along white. As we are moving, we get nominal value, when we cross line the sensor will see reduction in intensity b/c reflectance of black line changes.

It’s not as simple as keep reading until you see the number drop: 2 problems:

1 ) when

2) how do we treat this particular signal

Assume we program light sensor to take a reading: don’t make sampling so large that you miss line. But if you keep increasing sampling rate, at some particular point we’ll sample so fast we won’t miss the line. We need an analytical method to get us to an appropriate rate. We can choose the rate but does the speed of robot change anything (yes). If we go slow enuf we’ll find line:

1. relationship b/ween speed of travel
2. sampling rate at which to make sampling rate

BLACK LINE IS 6mm. Go into proto file and get parameters associated with this black line. Find out what the width of the line is w.

How fast do we sample to guarantee getting the guy? We have to have at least one sample in black and white region. We need a minimum of 1 sample every W distance units or 2 samples per cycle. This rule is the absolute minimum. Here we talk about the notion of a special frequency. One cycle is a black and white line. We’re watching how our signal changes relative to where we are in space.

The time needed to travel w units is = velocity x time

where the velocity is 2Pi x Omega (rotation velocity of the wheels) x radius of wheel / 360

the time it takes to cover distance w is w/v

sampling frequency is 1/t. So we make sure that if we’re going at a specific velocity I need to make sure my sampling frequency is 1/t. Or Pi x Omega x RadiusWheel / 180 x W (unit is in Hz).

So now we know that at a certain width W for a specific moving velocity we can calculate our sampling frequency to make sure we can meet our minimum criterion of getting one sample in dark and one sample in light.

It is highly unlikely that the minimum sampling rate is not enuf and you’ll need more than one sample. So, either you up the frequency or you slow down you motion to get more than one sample.

But, if you up the sampling rate by a factor 3 or 4 you would get your adequate sample, but you would also severely load the processor.

We could also slow down the travel velocity or you could do a combination of upping the sampling rate and reducing the velocity.

Your hard has a limit in terms of how many operations it can perform per cycle. The machine has to go through a whole data acquisition cycle. And there’s many other threads as well so there’s a limit on the computation resources you have as well. But if we slow down speed, we aren’t moving fast enuf. So we have to find a strategy. The odometer tells you approx. where you happen to be. So what you can do is use the odometer knowledge to give you context for how to do the measurements. At some point we should be +- cm away from the point we want to be at. So to figure out where line we slow down a bit to up chances of sampling the line. then we speed up. 1) Do we have to detect every single line? Well we’ve done measurements on odometer and we know that as its going to travel that it’s going to drift by so much. Consequently, in terms of correcting myself by detecting line, maybe we don’t need to detect every single line, maybe it’s every nth line and the number n is determined by the experiments that we perform with navigation code of the odometer to basically calculate the drift.

What we can do in simulator: predict what the drift is going to be.

That was part A which addressed the question of when.

Part B, how do we know that we’ve crossed a line?

We have a time varying signal, as robot moves, we see a change in the intensity that we record. That pattern that we measure, how do we verify if that pattern matches the pattern that we should see if we’re crossing a line.

Ambient light in the background shifts the dc signals up and down. The cardboard skirts around the sensors are needed to minimize the effect of the ambient light.

The light issue with real robots: if lights are turned off the average intensity drops. The Ambient light in the background shifts the dc signals up/down. If we wanted to use the most intuitive strategy possible you might say, well if you had only performed experiment w/ lights off, the avg intensity is 48 if it drops b/fore a 44 we can detect line. there’s a fallacy and you change the background illumination then your threshold value no longer works (it is an absolutely wrong thing to do). Think through what you’re doing and not just adopt the simplest thing.

Next

We don’t have a continuous signal, we have a discrete signal, so that assumes that we’ve figured out what our appropriate sampling rate is. So that what we actually measure is a reasonable facsimile of what exists in the world. First thing to do is come up with an idealized model of what the signal should be because if we can come up w/ a model we can compare what we measure to we the model to make sure that we’re seeing what we should be expecting.

Modeled black line as a step edge: apply an operator which gets really big where the signal changes. We can use a derivative to extenuate where the signal is changing most. The derivative is great w/ locating where the continuous signal undergoes a particular change.

We don’t have a continuous signal; we have discrete set of samples so we apply approx. to our derivative. We are sweeping continuity under the stair right now.

Discrete case delta F = f(n+1) -f(n) / delta t; where the delta t is 1/(sampling frequency). When there’s a start/end of line we want to return the centre point. When we are going from dark to light the derivative becomes positive; light to dark the derivative goes negative. We get the zero crossing which serves as an approximation for the location of the line.

If black on white then the peaks shift.

We want to filter the noise. there’s lots and lots of noise.

Whatever your conditions are , the zero crossing is always going to be the same. The derivative magnifies the noise. So which peaks correspond to the real signal. We assume that the magnit. of signal that comes from feature is so big it swamps the noise. Well in that case we can look at a threshold, so define a threshold where anything beyond that value is a real signal. In some cases this may not be the most appropriate solution.

If we don’t know the thickness of line or had no knowledge of environment it makes things really hard to perform the navigation. WE know patterns so conduct set of operations to detect those patterns.

Robot is randomly placed inside a square and we have to orientate yourself. Figure out which way is north. use ultrasonic to spin around in circle to get a pattern that looks like a wave. If you get to when you get to troughs you will see that they correspond to specific direction.

The second part involves more precise knowledge of your position. Start looking for grid lines. If you’re facing north start moving until you intersect the first grid line.

The algorithm will fail dramatically if you are unable detect a line.

How do we take raw data and filter it to improve our probability to detect particular features? The first thing we need to know is how do my instruments perform. We need to know what the signals look like when we try to make certain kinds of measurements.

\

DOn’t do:

Don’t just sit and aim a sensor at a wall for 100 samples and the average is x and this is how it performs. Again you should consider the structure of the environment and the ultrasonic sensor has a particular response w/ respect to its field of view. Take into account when you’re handling it Often very handy to plot the data and see how it what it looks like. Use light sensor to measure the returned intensity.

Second SLIDESHOW

In this slide show that we can actually get a measurement of the surface that we’re looking at. How do we determine if the signal that we are measuring corresponds to the feature of interest. Model: if we present this to our sensor, what type of signal is expected.

Filtering:

We again need a model. The simplest model says that the signal we measure is = ideal signal that is corrupted by noise (noise additive). One thing that we can do is come up with a general enough model which gets to a first order notion what the noise looks like. One is normal distribution. Independently and identically distributed (gaussian iid). Central limit theorem which says that as you have a signal which is a superposition of many other signals regardless of their particular distributions when you add them all up together they tend to have the characteristics of the normal distribution. We rely on the central limit theorem a lot to justify this particular model. It’s a simplification though, b/c in reality there’s sources which are non linear.

Combination of the nice simple model (CLT) with reality non linear elements is a reasonable model we can use for engineering purposes. We characterize simple model with mean and standard deviation.

If noise can be modelled w/ gaussian IID then mean can serve as the good estimate of the uncorrupted signal. from our intuition we can tell that the noise is distributed about the mean. The net conclusion is that the average is a reasonable approach of the signal. How do we test to make sure that our data conforms to our assumptions . THIS IS SUPER IMPORTANT TO TEST OUR ASSUMPTIONS.

How do we validate our hypothesis from the data (that the noise is additive and has an IID gaussian characteristic): we create a histogram to store the frequency of each data type. The yellow line corresponds to the mean value we printed b/fore. Signal is distirubted above the mean. The green is standard deviation. Our assumption is that our signal is distributed about the 2 S.D. Take data, perform regression on it, since func is characterized by two parameters, we can estimate these two params from data , generate corresponding func and see how well does the model account for our data.

When model doesn’t conform our expectations we have to explain or account for any particular biases.

For outliers: anyone that’s to a certain point off from a signal line should be thrown away. If magnitude is equal to mean + x S.D then throw it away and then recompute mean and S.D again. This becomes hard when our particular model doesn’t characterize the model.

Video Hints:

The goal is to get to x = 1 y = 1 and theta = 0