Christopher LaJon Morgan

CS 470: GridFilter Hw

Oct 30, 2013

**Discuss how the observation model relates to the likelihood in Bayes rule.**

They are one in the same. The observation model is used to model the observations obtained by a nosy sensor. Therefore, it models, given an object is there, how often the sensor reports the object. It also models, given an object is not there, how often the sensor reports an object being there.

The likelihood in Bayes rule represents this exact information. Given a particular state (object there, object not there), what is probability our sensor will report it. In other words, given an object is there, how often the sensor would report the object. And given an object is not there, how often the sensor reports an object being there.

**Combinations**

Original: 98%, 60%

|  |  |  |  |
| --- | --- | --- | --- |
| # | True Hit Rate | False Alarm Rate | Description |
| 1 | 98 | 60 | The first reading contained a few dark gray marks, with very few black marks. It was able to have a pretty good guess after what appeared two observations. In other words, it left a trail of mostly white. It also appeared that only about one pass was required to recognize an obstacle, they pretty well just stayed black. |
| 2 | 95 | 60 | Diff from 1: More spotty on none obstacle read, left a trail that was more spotty. |
| 3 | 85 | 60 | Diff from 2: Much more spotty, didn’t leave much of a trail of white. |
| 4 | 70 | 60 | Very shoddy. Hard time discerning anything, can’t tell when an obstacle is passed. |
| 5 | 98 | 50 | Diff from 1: Trail becomes cleaner. |
| 6 | 98 | 40 | “ |
| 7 | 98 | 30 | “ |

**What happens as the TrueHit value is lowered and why?**

As one decreases the TrueHit value the ability of the algorithm to effectively clear free ground and appropriately mark occupied ground becomes more and more muddled, to the point that one cannot tell if an obstacle has been found or not, even with multiple passes.

This occurs for some reason that I cannot explain. This is the part where there is something counter intuitive going on. Realistically, I would say something is wrong.

However, if I had to **throw** **something** together:

This occurs because as ones senor becomes more and more probable of a missed detection (1 – TrueHit) it degrades the probability that there is free ground when the sensor says there is free ground and the probability that it actually hit something when the sensor says it hit something. Therefore a decrease in the TrueHit value will cause the sensor to be less effective at reporting a true miss and a TrueHit. This is what causes the probability graph to stay muddy longer with a very low TrueHit rate.

**What happens when the FalseAlarm parameter is lowered and why?**

As one decreases the FalseAlarm rate the algorithm becomes more and more effective at clearing free ground. In other words, the algorithm clearly marks free ground more easily and faster as the FalseAlarm rate is decreased.

This occurs because as one decreases the false alarm rate with a high true hit rate, the sensor in essence becomes more and more precise at detecting none occupied ground. In essence, the lower the FalseAlarm rate the more sure one is that when the sensor says there is nothing there, one can conclude there is nothing there. That is why as it decrease free ground becomes white so quickly.

**What happens if FalseAlarm=0 and TrueHit=1, and why?**

The algorithm perfectly clears any free area and solidly fills in any obstacles.

This occurs because, by making False Alarm Rate zero and True Hit Rate one, one in theory has a perfect sensor. It never fires when it shouldn’t and always fires when it should. Therefore, everything is perfectly classified as it passes under the sensor.