Vision-based Navigation Exercise 1

Kerem Yildirir

April 2021

1 Part 1

- First line appends the path of the folder cmake_modules to a variable called "CMAKE_MODULE_PATH"
- Second line of commands set the C++ standard to C++14, and enforce it for compilation. "Required" statement ensures that if C++14 is not installed, project won't compile. Also the flag C++ extensions is set to off.
- Third line of commands define C++ compilation flags for different build configurations (Debug, Release , Relwithdebinfo). Basically, it sets the flags we would give to g++ if we were to use it alone.
- Final commands defines the executable and its source code, then links the necessary libraries to that executable

2 Part 2

Let $t = |w| = \theta$ and $v = \frac{w}{|w|}$ then $\hat{w} = \hat{v} * t$ and $\hat{v}^2 = vv^T - I, \hat{v}^3 = -\hat{v}, \dots$ (From the slide 40 in lecture slides)

After expanding the series $\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\hat{w})^n$, the output can be grouped similar to the example in slide 40:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\hat{w})^n = I + \left(\frac{t}{2!} - \frac{t^3}{4!} + \frac{t^5}{6!} - \dots\right) \hat{v} + \left(\frac{t^2}{3!} - \frac{t^4}{5!} + \frac{t^6}{7!} - \dots\right) \hat{v}^2$$

Multiplying and dividing both terms with t:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\hat{w})^n = I + \underbrace{\frac{\left(\frac{t^2}{2!} - \frac{t^4}{4!} + \frac{t^6}{6!} - \dots\right)}{t}}_{\frac{1-\cos(t)}{t}} \hat{v} + \underbrace{\frac{\left(\frac{t^3}{3!} - \frac{t^5}{5!} + \frac{t^7}{7!} - \dots\right)}{t}}_{\frac{t-\sin(t)}{t}} \hat{v}^2$$

Inserting $t = \theta$ and $\hat{v} = \frac{\hat{w}}{t}$:

$$\sum_{n=0}^{\infty} \frac{1}{(n+1)!} (\hat{w})^n = I + \frac{1-cos(\theta)}{\theta^2} \hat{w} + \frac{\theta - \sin(\theta)}{\theta^3} \hat{w}^2$$

3 Part 3

- **Why would a SLAM system need a map?**

  Having a map with some distinguished landmarks ensures that the system can recover when it starts making bad predictions by resetting its state and eliminating the erroneous predictions. It also helps in tasks like path planning and visualization.

- **How can we apply SLAM technology into real-world applications?**

  We can apply SLAM to the applications which require a globally consistent map generation. Typically, we employ an agent with sensors to perceive the world, and it observes its surroundings while running the SLAM algorithm suited for its sensors and creates a map of its observations while tracking its own location in this map.

- **Describe the history of SLAM.**

  SLAM history consists of 3 big time periods :

  - Classical age (1986 - 2004), where the main probabilistic formulations for SLAM came to surface. Approaches based on Extented Kalman Filters, Rao-Blackwellised Particle Filters and maximum likelihood estimation are introduced. In this era, basic challenges with efficient and robust data association are also studied.
  - Algorithmic analysis age (2004 - 2015), where the fundamental properties of SLAM, such as observability, convergence and consistency, are studied, Importance of sparsity for efficient computation is realized, and main open-source SLAM libraries were developed.
  - Robust perception age (2015 - Present), which is the current era we are in as the paper claims. This era focuses on robust performance (system ability to perform well under various environments for an extended amount of time), extending sytems's understanding beyond basic the geometry reconstruction to semantics and physics of the environment and resource awareness and task-driven perception, which are the attributes of the system which enables it to self adjust the computation load given the available resources and the the task at hand.