Particle Filter SLAM: A Project Review

Hao-Yuan Tang
Department of Electrical Engineering
University of California, San Diego
h6tang@ucsd.edu

I. Introduction

In this project, particle filter simultaneous localization and mapping (SLAM) was used to localize an autonomous car given its surrounding. SLAM can be separated into two parts: localization given a surrounding map of the environment for estimating robot trajectory and mapping surrounding with robot state simultaneously. Also, texture map was calculated using two aligned camera to visualize "large depth" pixel on the image plane onto surrounding map.

Particle filters, which is a form of Bayesian filter, can effectively model a robot's predicted locations in the world as discrete random variables with Dirac delta probability mass function based on past input controls and current observations. In problem formulation session, we will discuss on essential parts of particle filters SLAM, including differential-driven kinematic model, occupancy mapping and particle filter in mathematical sense. In technical approach session, detailed implementation of particle filter SLAM was given, such as transformation of coordinate systems, car model localization using odometry and surrounding map using lidar sensors.

II. Problem Formulation

A. Differential-driven Kinematic Model

Differential-driven kinematic model can be described as a planar motion model, which consists of linear velocity within a 2D plane and rotation with single axis. Denote linear velocity $\boldsymbol{v}_t$, angular velocity ω_t and yaw angle θ_t, differential-driven kinematic model can be written as:

$$\dot{\boldsymbol{x}} = \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = f(\boldsymbol{x}, \boldsymbol{u}) \coloneqq \begin{bmatrix}v\cos\theta\\ v\sin\theta\\ \omega\end{bmatrix} \quad (1)$$

In this project, we simply assume that the car's motion is 2D plane, which might not be definite but reasonable in our case. As figure 1 shown, given $\boldsymbol{v}_t$, ω_t, θ_t, robot state $\boldsymbol{x}_t$ and time interval τ, the robot state at next timestep can be predicted:

$$\boldsymbol{x}_{t+1} = \begin{bmatrix}x_{t+1}\\ y_{t+1}\\ z_{t+1}\end{bmatrix} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \coloneqq \boldsymbol{x}_t + \tau\begin{bmatrix}v_t\cos\theta_t\\ v_t\sin\theta_t\\ \omega_t\end{bmatrix} \quad (2)$$

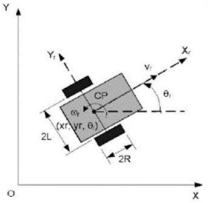

Fig. 1. Differential-driven kinematic model [1]

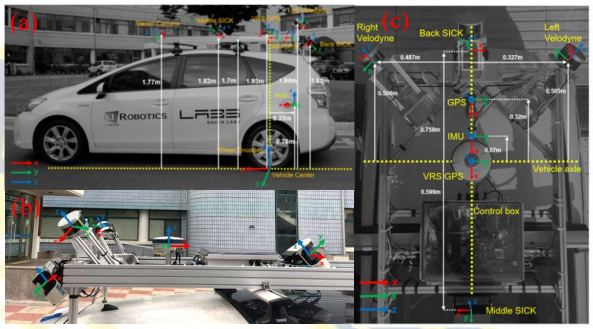

Fig. 2. Sensor layout on the autonomous vehicle. We will only use data from the wheel encoders, fiber optic gyro (FOG), front 2D LiDAR (Middle SICK), and the stereo cameras in this project. (a) Side view, (b) sensors on top, including lidar and (c) top view

In addition, to estimate uncertainty of system should be taken into account. A gaussian random noise with zero means and constant variance was added when prediction, i.e., $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\sigma_x^2, \sigma_Y^2, \sigma_\omega^2))$.

For estimated pose, observations from encoders, which gave linear velocity $\boldsymbol{v}_t$, and FOG, which gave yaw angular velocity ω_t, were necessary.

B. Occupancy Mapping

At time t, we are given observations from sensors z_t and current car estimated pose by differential-driven kinematic model. With the car's trajectory in world frame $\boldsymbol{x}_t$ and lidar reading, we can generate and maintain a log-odds grid m_t that is used to record the car's current surrounding.

Each cell i in the grid m_t was labelled as occupied (+1) or free (-1) independently of other cells. If we define $\gamma_{i,t} \coloneqq p(m_i = 1|z_{0:t}, x_{0:t})$, then:

$$m_{i,t} = \begin{cases}+1\ (occupied), & with\ prob.\ \gamma_{i,t}\\ -1\ (free), & with\ prob.\ 1-\gamma_{i,t}\end{cases} \quad (3)$$

And the odds ratio $o_{i,t} \coloneqq o(m_i|z_{0:t}, x_{0:t})$ of a cell, updated over time by the measurements is given by:

$$o_{i,t} = \frac{p(m_i = 1|z_{0:t}, x_{0:t})}{p(m_i = -1|z_{0:t}, x_{0:t})} = \frac{\gamma_{i,t}}{1-\gamma_{i,t}} = \frac{p(z_t|m_i = 1, x_t)}{p(z_t|m_i = -1, x_t)} \times \frac{\gamma_{i,t}}{1-\gamma_{i,t}} \quad (2)$$

Here, we can take the log of this expression to obtain $\lambda_{i,t} \coloneqq \lambda(m_i|z_{0:t}, x_{0:t}) \coloneqq \log\big(o(m_i|z_{0:t}, x_{0:t})\big) = \log(o_{i,t})$ to update the log-odds map:

$$\lambda_{i,t+1} = \lambda_{i,t} + \Delta\lambda_{i,t+1} \quad (3)$$

where $\Delta\lambda_{i,t+1}$ is a hyperparameter that specifies how much we trust our observation z_{t+1}. In this way, our update at time t + 1 for any cell of our log-odds grid is simply to the current value of the corresponding cell.

For the initial cell values, i.e., before we have any observations to process, it is reasonable to initialize all the cells to zero.

To generate the occupancy grid (a binary image of cells considered either occupied or free), we threshold the log-odds grid based on thresholds chosen via empirical observations, which are another hyperparameter to tune.

C. Particle Filter - Localization

The probability distribution $p(x)$ of the car's next pose $\boldsymbol{x}_{t+1}$ after the **prediction step** can be represented as a mixture of delta functions weighted by the N particle weights:

$$p_{t+1|t}(x) = \sum_{n=1}^{N} \alpha_{t+1}^{(n)} \delta(x; \mu_{t+1}^{(n)}) \quad (4)$$

Where $\mu_{t+1}^{(n)}$ is the prediction for the nth particle given the differential-driven kinematic motion model.

To complete the update step, we simply compute the correlation between each particle's next observation and our current knowledge of the world (which was encoded into occupancy grid) and weight the particle accordingly:

$$p_{t+1|t+1}(x) = \sum_{n=1}^{N} \frac{\alpha_{t+1}^{(n)} p_h(z_{t+1}|\mu_{t+1|t})}{\sum_{n=1}^{N} \alpha_{t+1}^{(n)} p_h(z_{t+1}|\mu_{t+1|t})} \quad (5)$$

Where p_h represents the observation filter model in a general Bayesian filter.

D. Texture mapping

For better understanding SLAM, texture mapping was used to visualize surroundings by projecting pixel RGB values, which were collected by two stereo cameras on the top of vehicle, onto another grid map like occupancy map. First, to transform pixel coordinates into optical frame, it's necessary to compute focal length by intrinsic parameter matrix of camera:

$$K_s = \begin{bmatrix} s_u & s_\theta & c_u \\ 0 & s_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, F_f = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, K_f = \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} s_u & s_\theta & c_u \\ 0 & s_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -f & 0 & 0 \\ 0 & -f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} f_{su} & f_{s\theta} & c_u \\ 0 & f_{sv} & c_v \\ 0 & 0 & 1 \end{bmatrix} = K \quad (6)$$

where K_s, F_f, K_f and K represents pixel scaling, image flip, focal scaling and calibration matrix respectively. With matrix computation, the focal length f can be obtained.

The next step is to compute disparity of each pixel by cv2.StereoBMcreate[5], then we can get "depth" of each pixel, i.e., the distance between objects in this pixel and the focal plane. The relationship between pixel coordinates on left image $[u_L, v_L]$ and its optical frame coordinates $[x_o, y_o, z_o]$ can be built by disparity d:

$$\begin{bmatrix} u_L \\ v_L \\ d \end{bmatrix} = \begin{bmatrix} f_{su} & 0 & c_u & 0 \\ 0 & f_{sv} & c_v & 0 \\ 0 & 0 & 0 & f_{su} b \end{bmatrix} \frac{1}{z} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \quad (7)$$

where b represents baseline between two camera and other parameters were given in camera calibration matrix.

Finally, the pixel RGB values in optical frames were transform into another grid map for visualization after optical to world frame transformation, like how we implement in occupancy grid map.

III. Technical Approach

A. Coordinate Transformation

Since readings in sensors were in sensor frame, the first task in this project was to transform them into world frame, the convert world frame coordinate to occupancy grid cells for mapping.

The transformation from sensor frame {s} to world frame {w} can be obtained by a hierarchical way by first transform to vehicle frame {b} then to world frame. Denote the transformation from {s} to {b} as ${}_{\{b\}}T_{\{s\}}$ and from {b} to {w} as ${}_{\{w\}}T_{\{b\}}$, then the transformation from {s} to {w}:

$$\begin{aligned} {}_{w}T_{s} &= {}_{w}T_{b}\,{}_{b}T_{s} = ({}_{w}T_{b})^{-1}\,{}_{w}T_{s} \\ &= \begin{bmatrix} {}_{w}R_{b}^{T} & -{}_{w}R_{b}^{T} \times {}_{w}\boldsymbol{p}_{b} \\ \boldsymbol{0}^{T} & 1 \end{bmatrix} \begin{bmatrix} {}_{w}R_{s} & {}_{w}\boldsymbol{p}_{s} \\ \boldsymbol{0}^{T} & 1 \end{bmatrix} \end{aligned} \quad (8)$$

where R and $\boldsymbol{p}$ represent rotation matrix and translation vector, respectively. With ${}_{w}T_{s}$, sensor readings can be transformed into {w} by dot product with $[s_b\ 1]^T$, where s_b represents the readings.

B. Particle Filter(Localization)

The basic setup of the vehicle's measurement in this scenario is:

1. Available lidar scans gave information about surroundings at certain time, i.e., distance between the sensor and the obstacles with 195deg FOV.
2. Noisy odometry readings, including FOG and encoders, gave information about relative movement between timesteps in {w}.

For our vehicle, the surroundings were known by lidar scans by measuring the distance between lidar sensors (origin in {s}) and obstacles. Therefore, transform lidar hits to world frame{w} and further to occupancy grid was the first task. And Bresenham's line generation algorithm[7] was used to determine the free cells, i.e., all points "before" lidar scans to {s}. Then, particle filter can be used with such map. The idea of particle filter was as following:

1. **Initialize**: generate N hypothetical positions, which were called **particles**, randomly distributed within free cells in occupancy grid.
2. **Prediction step in particle filter**: predict every particle's next pose by differential-driven kinematic model with additive noise to model the uncertainty.
3. **Update step in particle filter**: To determine the optimal choice of particle, estimate the correlation between the information obtained from the lidar scans at t + 1 to the current occupancy grid, and give each particle a weight $\alpha^{(n)}$ proportional to this correlation.
4. **Update of occupancy grid**: with the particle with highest weight, the predicted vehicle's pose was assigned, and used the lidar scans at the next time step z_{t+1} to update occupancy grid, this is the mapping step in SLAM.

5. **Resampling**: In particular case, if the most of particles have weights that make them useless, then generate new particles based upon the current distribution of weights of particles, and give them uniform weights.
   For the noise, we generated independent zero-mean Gaussians for the vehicle's relative odometry in x, y and θ.

For the noise, we generated independent zero-mean Gaussians for the vehicle's pose. The variance of $[\sigma_x^2, \sigma_Y^2, \sigma_\omega^2]$ were set [0.05, 0.05, 0.015] be empirical test. Sufficient noise improved model prediction in the trials, while there's a threshold of noise level. Different types of noise were set, including constant, linear and square root proportional to vehicle's relative odometry reading, and the constant variance outperformed the other two.

For computing correlation of a given particle given lidar scans, we simply computed the sum of cells that agreed between the latest scan and the current binary occupancy grid map. Softmax function was used to convert weights of particles into probability, and the correlation map was obtained simply by similarity function between the transformed and discretized scan and the occupancy grid:

$$\mathrm{corr}(\boldsymbol{y}, \mathrm{m}) = \sum_i \mathbb{I}\{\mathrm{y_i} = \mathrm{m_i}\} \quad (9)$$

where $\boldsymbol{y} = r(\boldsymbol{z}, \boldsymbol{x})$ is the transformation from a lidar scan $\boldsymbol{z}$ to grid cell indices $\boldsymbol{y}$ and $\mathbb{I}\{\mathrm{y_i} = \mathrm{m_i}\}$ was output of indicator function of lidar scan $\mathrm{y_i}$.

For update step in particle filter, resampling was applied if most of the particles were redundant. The threshold to determine resampling was defined as effective number of particles $\mathrm{N_{eff}}$ as following:

$$\mathrm{N_{eff}} := \frac{1}{\sum_{k=1}^{N} (\alpha_{t|t}^{(k)})^2}$$

where denominator represents sum of particles' weights $\alpha_{t|t}^{(k)}$ at time step t. Resampling was useful when number of particles were large, since representative particles be much less than useless particles is the nature of Bayesian particle filter.

IV. Results

This section showcases several test cases that the complete program was put through and the success and failures are discussed. For all validation images, the segmentation mask and images with bounding box are displayed. For images where recycling bin was not found, all the predicted bounding boxes are displayed to see whether the result was rejected because of its morphology or model performance.

A. Trajectory prediction

The trajectory of our vehicle model was shown in Fig. 3. As we can see, the predicted trajectory by particle filter and the "ground truth" by odometry was highly overlap, which indicated that particle filter SLAM was appropriate for long-range robotic motion modeling. Since the working range of lidar sensors was 80m in 195deg FOV, only obstacles within this region were shown (black points) and all the pixels "in front of" lidar hits were label as free cell (white) in the occupancy map.

Since lidar hits' relative positions were calculated to vehicle's position, which was predicted as the position of particle with highest weight, the occupancy grid would differ between each trials and such uncertainty eventually converged. Couple of factors that influenced convergence rate were observed, including number of particles, variances of differential-driven kinematic model, level of trust, and the map resolution (meters/pixel).

B. Log-odds map

As we expected, in Fig. 4., the predicted trajectory followed the "path" of free cells (dark color) in occupancy map. And the neighbor blue regions were also free cells, but with smaller log-odds. Some parameters in probabilistic occupancy grid mapping, such as the "trust" on sensors' measurement, was not discussed in this report, but such parameters would change log-odds map values and generate a different occupancy map. Since the update step in particle filter depends on occupancy grid, change these parameters might influence trajectory prediction. So, in this project we only fix these parameters to understand SLAM algorithm.

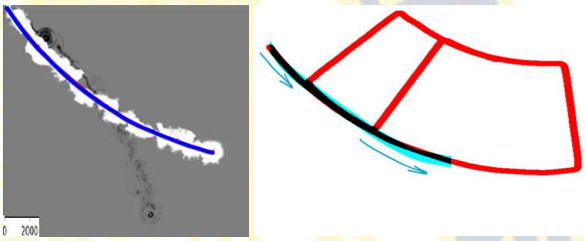

Fig. 3. Occupancy grid over 29000 lidar scans (around 300 sec). The map resolution was set 0.1 meters per pixel. Trajectory of vehicle by odometry (blue) and predicted by particle filter SLAM (red), occupied cells (black), and free cells (white) were shown. The unit of scale was 2000 pixels. And the right image shows trajectory during whole travels.

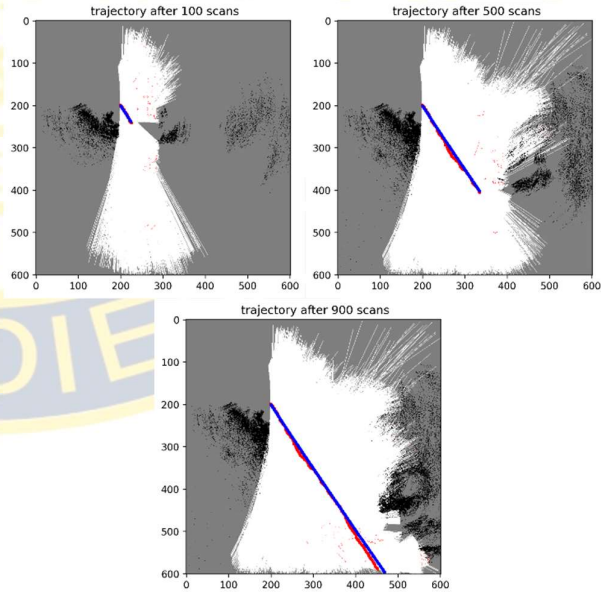

Fig. 4. Occupancy grid iteration with 50 particles (a)-(c) iteration from t = 100 to 900

C. Number of Particles

Different number of particles were experimented in this project. Intuitively, the more particles we used, the higher computation cost was generated. Due to constraint of testing hardware, the maximum number of particles we tried was 300, which gave reasonable results without too much computation expensive.

The overall prediction was not much different between different number of particles, however, the convergence rate increased when number decreased and this can be observed from Fig. 5. Large number give a sparser distribution of particles, and most of them were totally biased in random initialization. Also, during iteration, large number of particles resampled more often, which increase the computation cost in linear sense. But more particles trial still give better results over long time, this might be related to the benefits from resampling, which focused the representation power of the particles to likely regions while leaving unlikely regions with only few particles.

D. Noise Simulation

As mention in Problem formulation session, the uncertainty level in differential-driven kinematic model influenced the accuracy of prediction in particle filter. Additive noise can be seen as additional uncertainty in the whole motion system, and this would cause some errors accumulated over time. As a result, the predicted trajectory would be biased if the errors getting to large. Such phenomenon is common in motion model, such as drifting in gyroscope readings, which caused by integration of inherent imperfections and noise within the device.

Certain types of noise were simulated: constant, linear and square root. The variance of noise was described as linear or square root depends on the order of variance which relative odometry readings proportional to:

$$\text{Linear noise: } \begin{bmatrix}\sigma_x^2\\ \sigma_y^2\\ \sigma_\omega^2\end{bmatrix} \propto \begin{bmatrix}x_{t+1}\\ y_{t+1}\\ \omega_{t+1}\end{bmatrix} - \begin{bmatrix}x_t\\ y_t\\ \omega_t\end{bmatrix} \quad (10)$$

$$\text{Square root noise: } \left(\begin{bmatrix}\sigma_x^2\\ \sigma_y^2\\ \sigma_\omega^2\end{bmatrix}\right)^2 \propto \begin{bmatrix}x_{t+1}\\ y_{t+1}\\ \omega_{t+1}\end{bmatrix} - \begin{bmatrix}x_t\\ y_t\\ \omega_t\end{bmatrix} \quad (11)$$

The main consideration of these noise model is because of widely discussions on noise in differential-driven kinematic model in literatures[2-4]. A simple question to inspire this idea: does gaussian noise with the same variance and mean cause the same effect on prediction when the vehicle in low or high speed? Intuitively, the higher speed this vehicle has, the higher overall uncertainty is in the whole system, so lower variance might be a good choice to simulate in this condition. However, this is not the case: as Fig. 6. showed, the errors during this process was accumulated faster in linear and square noise than the constant one. As a result, the constant noise gave more accurate prediction in the very beginning. It's obvious that the noise model heavily influenced prediction and how the particles converge, so further study on such effects should be taken into account to improve the algorithm.

E. Texture mapping

The disparity map was calculated by OpenCV function, and the texture map was generated as Technical Approach session mentioned. The difficulty in this part is computational complexity, since we need to update a very wide range field of view in every timestep to visualize the map, here we only showed very beginning for idea presentation. As we can see, the orange line on the ground was shown in this map and the distance (or depth) was estimated via stereo model. Also, the texture map showed a circular sector-like shape of FOV, which is what researcher expected.

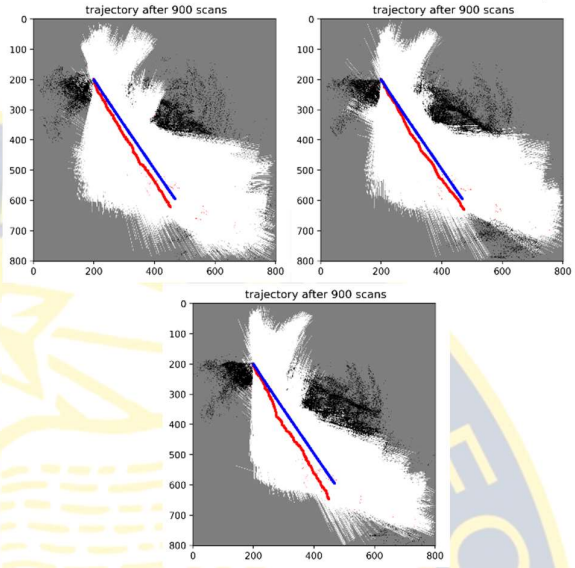

Fig. 5. Effects of number of particles on trajectory prediction (a) N = 2 (b) N = 50 (c) N = 300

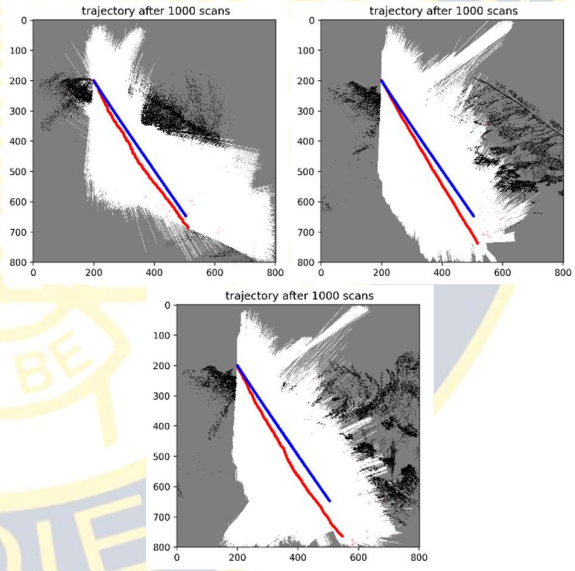

Fig. 6. Noise effect (a) constant, (b) square root and (c) linear

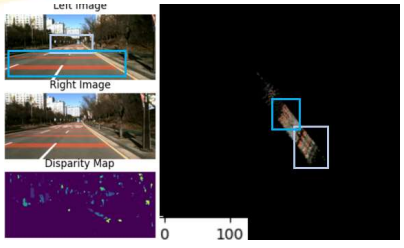

Fig. 6.(a) Image by left camera, right camera, and the corresponding disparity map (b) texture mapping

REFERENCES

[1] A. Filipescu, V. Minzu, B. Dumitrascu, A. Filipescu and E. Minca, "Trajectory-tracking and discrete-time sliding-mode control of wheeled mobile robots," 2011 IEEE International Conference on Information and Automation, 2011, pp. 27-32, doi: 10.1109/ICINFA.2011.5948958.

[2] Vatsal, V.; Bhargav, V. Design and Testing of the TeamIndus ECA Lunar Rover Navigation and Control. Preprints 2021, 2021050238 (doi: 10.20944/preprints202105.0238.v1).

[3] H. Andreasson, T. Duckett and A. J. Lilienthal, "A Minimalistic Approach to Appearance-Based Visual SLAM," in IEEE Transactions on Robotics, vol. 24, no. 5, pp. 991-1001, Oct. 2008, doi: 10.1109/TRO.2008.2004642.

[4] Robert Grepl. Balancing Wheeled Robot: Effective Modelling, Sensory Processing And Simplified Control. Engineering MECHANICS, Vol. 16, 2009, No. 2, p. 141–154

[5] https://docs.opencv.org/3.4/d9/dba/classcv_1_1StereoBM.html

[6] UCSD ECE276A: Sensing & Estimation in Robotics (Winter 2022) https://natanaso.github.io/ece276a

[7] http://www.phatcode.net/res/224/files/html/ch35/35-01.html