Search-Free Gridding and Temporal Local Matching Based Observation for High Frame Rate and Ultra-Low Delay SLAM System

Yuchen Yang, Songlin Du, Takeshi Ikenaga

Graduate School of Information, Production and Systems, Waseda University,
Kitakyushu 808-0135, Japan

Corresponding Author: yycdx@toki.waseda.jp

Abstract

Tracking of camera pose in simultaneous localization and mapping (SLAM) using particle filter has high potential of parallelism and is promising in high frame rate and ultra-low delay applications. However, heavy calculation and hardware-unfriendly algorithm in projection and matching which take 88.6% time consumption of particle filter become problems in acceleration. This paper proposes search-free gridding based redundant projection removal, non-iteration linear function based division, and temporal 2D local matching in particle filter based SLAM on FPGA. Search-free gridding based redundant projection removal removes dense map points and reduce projection times. Non-iteration linear function based division considers the particularly of projection problem in SLAM and achieve low delay partial division in projection. Temporal 2D local matching uses characteristic of high frame rate video and considers hardware implementation to realize a low delay local matching. The algorithm is tested on 762fps VGA (640×480) indoor sequences. Software result shows that this algorithm has 1.20° error in rotation and 1.90 cm error in translation compared with ORB-SLAM. Hardware result shows that matching and projection have 0.457 ms latency processing 1000 feature points.

Keywords: Particle filter, Simultaneous localization and mapping(SLAM), hardware-friendly algorithm.

1. Introduction

Visual simultaneous localization and mapping (visual SLAM) is known to estimate camera poses as well as sparse environment reconstruction at the same time. Current main framework of SLAM mainly consists of tracking and

mapping⁽¹⁾. In mapping, an environment map is created according to some selected frames, which are also known as keyframes need to be processed. In tracking however, camera poses are estimated using images and position of map points. In real-time applications such as drone flight control and autopilot, high frame rate and ultra-low delay estimation of camera poses is needed. There is currently high frame rate and ultra-low delay matching system on FPGA⁽²⁾. But no high frame rate and ultra-low delay SLAM system has been proposed.

Mainstream tracking algorithms in SLAM are bundle adjustment⁽³⁾ based method or filter based method^(4,5). In ORB-SLAM⁽⁶⁾, bundle adjustment provides accurate estimation but it is an optimization based method which needs several iterations. In filter method, there are extended Kalman filter (EKF)⁽⁷⁾ and particle filter^(4,8) based method. Particle filter has higher potential of parallelism and is more suitable for high frame rate and ultra-low delay applications. However, Kalman filter is used in particle filter based SLAM to estimate the position of landmarks and it has larger error in large scale sequences compared with bundle adjustment. So, in this paper, mapping part uses bundle adjustment to realize local and global map point optimization. In tracking part, particle filter is employed as tracking algorithm. For the features, we use 256-bit ORB feature⁽⁹⁾ because it is a binary feature which is simple enough to be realized on hardware. It also has less extraction time compared with other features such as SIFT⁽¹⁰⁾ and SURF⁽¹¹⁾. To achieve more parallelism and ultra-low delay, FPGA is chosen instead of GPU⁽¹²⁾.

According to software simulation, projection and matching in particle filter are the most time-consuming parts which take 76.6% and 12% separately. In projection, different particles have different poses. Since a reprojection error based likelihood is used in particle filter, all map points'

2D projections in each particle's camera plane need to be calculated. This leads to large computation. And unavoidable division in projection also leads to high latency in hardware implementation. Complex patterns in an image have more feature points crowded together and after triangulation in mapping, there are more corresponding map points crowded together in 3D space. Conventional work⁽⁶⁾ uses a gridding based method to reduce dense feature points but it needs search in each level of grids. It divides the image into several layers of grids, which is also known as an octree. In each layer it searches for regions which still have dense features. After that, it makes smaller grids in this region. Finally it only keep one feature point in each grid to achieve a balanced distribution. This algorithm is suitable for software. But if it is implemented on hardware, in worst case, it leads to large latency because each feature needs to be processed in each layer. Search-free gridding based redundancy removal is proposed to achieve a hardware-friendly algorithm which also reduces map points in complex pattern areas. It uses a pipelined design to process each projection in the last frame in the same way. This reduces time consumption compared with conventional work. In division however, conventional works usually consider a fully functioning division. It is an iteration based method that calculates the result bit by bit and causes high latency. But in projection of SLAM problem, a fully functioning division is not needed. So, a partially functioning non-iteration division is proposed to decrease latency.

Matching part matches ORB features and map points according to their descriptors. Some conventional work uses tracker⁽¹³⁾ to track corner features but it is only a visual odometry without mapping. Matching is the only part that uses image information and needs to be considered in hardware dataflow. Brute force matching is simple and has been realized in high frame rate and ultra-low delay system. But it leads to large resource cost or high latency when a feature point is compared with hundreds of map points in SLAM problem. To ensure accuracy and avoid unnecessary descriptor comparison, conventional work^(5,6) uses camera movement and map points' position to constrain the map points that a feature point is compared with. However, this method still compares feature points with each map point. Considering characteristic of high frame rate video, difference between two frames is small even if the camera moves very fast. Also, considering hardware implementation, temporal 2D local matching is proposed to reduce redundant matchings and keep accuracy.

This paper is arranged as follows: section 2 introduces

Fig. 1. Particle filter based SLAM framework.

particle filter based SLAM framework and three proposals. Section 3 shows the hardware dataflow and experiment result. Section 4 is the conclusion.

2. Proposals

2.1 Algorithm framework

Particle filter based SLAM framework is shown in Fig.1. The inputs are ORB feature points and map points. Map points are provided by mapping part in ORB-SLAM. Also, initialization part and loop closing part are also the same with ORB-SLAM⁽⁶⁾. The output of particle filter is the camera pose of each frame. This particle filter has prediction, observation and estimation. Each particle represents a different camera pose, which is an element of Lie group of 3-dimensional Euclidean transformation, $SE(3)$. Scale is not considered here. So, each particle has 3 degrees of freedom in rotation and translation separately in Lie algebra $se(3)$. Rotation is represented as Euler angles ignoring gimble lock. State vector ζ_k which represents camera pose and velocity in frame k is given by

$$\zeta_k = \begin{bmatrix} S_k \\ V_k \end{bmatrix}, k \in N, \quad (1)$$

$$S_k = [x_k \ y_k \ z_k \ \alpha_k \ \beta_k \ \gamma_k]^T, \quad (2)$$

$$V_k = [v_{xk} \ v_{yk} \ v_{zk} \ v_{\alpha k} \ v_{\beta k} \ v_{\gamma k}]^T. \quad (3)$$

In prediction, we assume that each particle has a constant velocity with a zero-mean gaussian noise in each element of $se(3)$:

$$\zeta_k = \begin{bmatrix} I_6 & TI_6 \\ 0 & I_6 \end{bmatrix} \zeta_{k-1} + \begin{bmatrix} \frac{T^2}{2} I_6 & 0 \\ 0 & TI_6 \end{bmatrix} \begin{bmatrix} W_k \\ W_k \end{bmatrix}, \quad (4)$$

$$W_k = [w_t \ w_t \ w_t \ w_r \ w_r \ w_r]^T, \quad (5)$$

$$w_t \sim N(0, \sigma_t^2), w_r \sim N(0, \sigma_r^2), \quad (6)$$

where T is the time of one frame. I_6 represents a 6×6 identity matrix. Gaussian distribution's variance is different in translation and rotation.

Then in observation, matching between 2D feature points and 3D map points is firstly done. Secondly, we do projection of each map point in each camera pose and calculate the reprojection error of each particle at moment k . Reprojection error is the difference of feature point $p_{k,j}$ and the projection of matched map point X_j in this particle's camera plane as shown in equation (7). The superscript means particle i . Equation (8) is the observation model of one map point X_j in one particle pose i , which is one part of observation in particle filter. It is also known as the camera projection model. The reciprocal of the sum of reprojection error of each particle is the likelihood. To emphasis poses with smaller reprojection error and discard ones with higher reprojection error, we take square of reciprocal as likelihood:

$$e_{k,j}^{(i)} = z_{k,j}^{(i)} - p_{k,j}, \quad (7)$$

$$z_{k,j}^{(i)} = h(X_j, S_k^{(i)}), \quad (8)$$

$$L(z_k^{(i)}) = \left(\frac{1}{\sum_{j=0}^N \|z_{k,j}^{(i)} - p_{k,j}\|^2} \right)^2. \quad (9)$$

In equation (9) the smaller reprojection error a particle has, the higher likelihood it has. For the estimation, we firstly do resampling according to the likelihood. Then the weighted average pose in $se(3)$ is calculated and taken as the output.

In the initialization of SLAM, we use an automatic initialization⁽⁶⁾ which considers homography matrix for planar scenes and fundamental matrix for non-planar scenes. After the initialization of SLAM, initialization of particle filter is based on the initial camera pose. Initial gaussian noise is added to initial pose ζ_k and particles with different poses and velocities are generated.

2.2 P1: Search-Free Gridding based Redundant Projection Removal

While extracting features from images, there are more features in areas with complex patterns if there is no constraint. Thus, the corresponding map points also gather together in such areas. This method uses 2D projections in the last frame to judge whether map points in 3D space are gathering together and remove some map points in dense areas to achieve a balanced distribution. The last frame's projection is used because it is simpler to judge in 2D space than in 3D space and difference between two adjacent frames in high frame video is very small.

Fig. 2. Detail of redundancy removal

In the proposed method, each projection is processed in 3 same steps. They are firstly assigned into grids, which is 32×24 for example. Then the 32×24 grid is called the top layer. As shown in Fig. 2, in each grid in top layer, there are possibly several projections. We only keep one projection whose corresponding map point is seen by the largest number of keyframes. If there are two projections observed by the same number of keyframes, we choose the firstly processed one. Map points seen by more keyframes exist longer than other map points and are more stable after bundle adjustment in mapping.

The bottom layer has larger grids than top layers. In case that there are still projections crowded together, we constraint the maximum number of projections in each grid in bottom layer. After that, map points are well-distributed in 2D space. The corresponding map points of the left projections are thought to be necessary and are processed in the projection step.

In hardware implementation, one projection is input in each clock. All projections are processed in pipeline. Gridding is based on binary search of projection's position. The result of gridding is stored in a RAM in which each element represents one grid. Thus, when the top layer has 32×24 grids and the input has N projections, 5 clocks are needed in binary search and one clock is needed in each layer to judge whether this projection should be kept. After all, this design has only $N + 6$ clocks latency.

2.3 P2: Non-iteration Linear Function based Division

In projection, the 2D projection of each map point in each particle's camera plane needs to be calculated. In camera projection model, firstly the map point division is needed when transforming from camera coordinate to file coordinate. Camera projection model is shown in equation (10):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} f_x x \\ f_y y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \quad (10)$$

Fig. 3. Map points' distribution and selection

$$[x, y, z]^T = R_w X_w + t_w, \quad (11)$$

where the dividend is the projection on z axis of distance between the camera center and map point. Vector $[u \ v]^T$ is the 2D projection of map point $[x, y, z]^T$. $[f_x \ f_y]$ and $[c_x \ c_y]^T$ are focus length and principle point of camera. $R_w \in SO(3)$ and $X_w, t_w \in R^3$ which represent map point's position and camera position separately. Subscript w means reference of world coordinate. Here scale is not considered.

In SLAM, if a map point is too close to the camera, it means that an object blocks the camera. This situation is almost impossible in normal sequences. And it is also impossible that a map point is too far away from the camera because it has large error to triangulate a far map point. Even though a map point which is too close to or too far away from the camera exists, the position of this map point is not converged and has large error. In a word, only map points which are in an appropriate range from the camera are supposed to have little error.

Considering this fact, map points' selection in Fig. 3 is proposed. In this method, piecewise linear function is used to replace reciprocal calculation in division:

$$\frac{1}{z} \approx \begin{cases} \frac{1}{th_1}, & z < th_1 \\ k_i z + b_i, & z_i \leq z < z_{i+1} \\ \frac{1}{th_2}, & z \geq th_2 \end{cases} \quad (11)$$

The piecewise function has 16 intervals in which k_i and b_i are different and satisfy:

$$\begin{cases} \frac{1}{z_i} = k_i z_i + b_i \\ \frac{1}{z_{i+1}} = k_i z_{i+1} + b_i \end{cases} \quad (12)$$

We first use binary search to find the interval, then calculate the approximation of reciprocal of z . After reciprocal is obtained, multiplications with x and y are done. The division of this value has little error with true value but has shorter latency than common iteration based division.

Fig. 4. Temporal 2D local matching

Fig. 5. Storage of projections and local search range

2.4 P3: Temporal 2D Local Matching

In SLAM problem, matching between feature points and map point is needed. In high frame rate video, the most important characteristic is that the difference between two frames is small. The movement of a feature point between two frames is less than 10 pixels. As shown in Fig. 4, this method firstly calculates the 2D projection of map points in the last frame's camera plane. The map points used here are only necessary map points from proposal 1.

When we do matching between feature points and map points, we only compare hamming distance of descriptors of feature points and map points whose projection in the last frame is in a local search range. If there are several projections in the search range, the one with smallest hamming distance is selected and this distance must be smaller than a threshold. A one-to-one matching is considered in which one map point is only matched with one feature point. Considering the dataflow, only the first feature point matched is stored.

In storage aspect, as shown in Fig. 5, we use a RAM whose address is directly related with the position of the projection position. Each element in the RAM represents a 16x16 pixels grid. It stores the address of the oldest map point whose projection in last frame is in this grid. after Proposal 1 assigns projections in the last frame into grids and removes dense map points, addresses of the map points left are assigned to RAM in this proposal. Only one map point is stored in grid here as well. The selection method is the same

Fig. 6. Test sequences examples

Table 1. Sequences explanation

Name	Max translation speed (m/s)	Average translation speed (m/s)	Max rotation speed (°/s)	Average rotation speed (°/s)
X1	4.4	1.0	176.8	70.2
X2	5.7	1.2	179.1	83.3
Y1	5.9	1.0	179.9	84.5
Y2	5.7	1.2	180.0	89.0
Z1	1.8	0.7	131.3	52.9
Z2	3.5	1.2	179.5	92.1
Circle	2.9	1.0	174.5	97.6
Rand 1	6.7	0.8	179.9	109.4
Rand 2	5.2	0.9	172.4	74.7

with proposal 1.

After preparation, when a feature point is input from feature extraction part, we firstly calculate the position of it in the RAM storing temporal projections. Then we compare hamming distance with map points whose address is in this RAM element and surrounding eight ones. So, one feature point is only compared with maximum nine map points. This reduces complexity and raise matching accuracy. This storage method is a sparse storage which has many empty elements but has a direct and fast accessing speed.

3. Experiment

3.1 Experiment environment

The algorithm is tested on a computer with Intel Core i7 CPU and 8 GB memory by Visual Studio 2013. Third party libraries are Opencv 2.4.12, g2o, Eigen 3.2.10 and DBow2. For hardware implementation, we used Vivado High Level synthesis 2017.2 and ISE 14.7. And the FPGA

device we use is Xilinx Kintex-7 xc7k480tffv1156-2.

Test sequences we use are 9 high frame rate sequences to simulate the high frame rate hardware implementation. The resolution is 640×480 and frame rate is 762 fps. Each sequence has 2000 frames. Four examples are shown in Fig. 6. Table 1 gives explanations of sequences. Rotation here is the angle from rotation matrix using exponential map. Names of sequences are the same with move direction of cameras. Circle means camera moves like circle. Rand 1 and 2 mean random movement. Rand 1 is zigzag movement and rand 2 is helix movement. These sequences all include strong rotations and fast translation.

3.2 Accuracy evaluation

Accuracy of the tracking algorithm is evaluated on software compared with ORB-SLAM⁽⁶⁾. Because this paper targets on hardware implementation, fixed point decimal with three integer bits and ten fractional bits are used. The particle number is 512. In each image, we extract 1000 feature points. Since the sequences are made by ourselves, we measured the distance of movement and then we know the scale in monocular SLAM. We firstly transform our result to the coordinate of the result of ORB-SLAM. After this the two trajectories have the same initialization. Then evaluate the accuracy according to root mean square error (RMSE) of rotation and translation separately:

$$RMSE_R = \sqrt{\frac{\sum_{k=n_0}^n E_{kR}^2}{n-n_0}}, \quad (13)$$

$$RMSE_T = \sqrt{\frac{\sum_{k=n_0}^n E_{kT}^2}{n-n_0}}, \quad (14)$$

where E_{kR} and E_{kT} are the absolute difference of rotation and translation. They are defined as equation (15), (16):

$$E_{kR} = \cos^{-1}\left(\frac{\text{Tr}(R_{kRP}R_{kRO}^{-1})-1}{2}\right), \quad (15)$$

$$E_{kT} = \|(x_{kP}, y_{kP}, z_{kP})^T - (x_{kO}, y_{kO}, z_{kO})^T\|, \quad (16)$$

where R_{kRP} is the rotation part of camera pose from particle filter at time k and R_{kRO} is the camera pose from ORB-SLAM. In equation (16), $(x_{kP}, y_{kP}, z_{kP})^T$ is the translation of camera pose from particle filter at time k and $(x_{kO}, y_{kO}, z_{kO})^T$ is that from ORB-SLAM. Because of the randomness of SLAM, each sequence is tested five times and we take the median value as the result. Compared with basic particle filter framework without proposals, after adding three proposals, total accuracy has little difference. The first proposal reduces some map points but make the map points' distribution more balanced. Rotation error decreases 0.04 degree compared with basic framework but translation error increase 0.05 cm. Although each sequence is tested 5 times

Table 2. Software accuracy rotation RMSE

	Rotation error (°)			
	Conventional	P1	P1+P2	P1+P2+P3
X1	1.38	0.75	1.56	1.88
X2	1.35	1.07	1.24	0.73
Y1	0.92	0.92	1.16	0.80
Y2	1.16	1.92	1.02	1.83
Z1	0.85	0.89	0.81	0.93
Z2	0.94	1.17	1.03	1.22
Circle	1.00	2.14	0.79	0.85
Rand 1	2.37	1.12	1.85	1.37
Rand 2	1.15	1.13	1.07	1.19
Average	1.24	1.23	1.17	1.20

Table 3. Software accuracy translation RMSE

	Translation error (cm)			
	Conventional	P1	P1+P2	P1+P2+P3
X1	2.39	1.44	2.14	1.65
X2	2.52	1.97	2.20	2.37
Y1	1.72	1.68	1.99	1.48
Y2	1.93	2.30	1.48	2.51
Z1	1.06	1.26	1.74	1.48
Z2	1.53	1.73	1.87	2.09
Circle	1.63	2.37	1.31	1.43
Rand 1	1.87	1.96	2.78	2.03
Rand 2	1.99	1.88	1.73	2.10
Average	1.85	1.84	1.91	1.90

and median error is taken as the result, there is still randomness so that proposed methods have less error in some sequences compared with basic framework while have larger error in other sequences. In nine sequences, Z1 is the slowest one. So, it has the smallest error compared with other sequences. Rand 1 is the fastest one and has larger error than other sequences in conventional method and after adding proposal 1 and 2.

3.3 Hardware performance

Three proposals are implemented in hardware and compared with conventional works. Fig.7 shows the data flow of hardware implementation.

For the input, temporal inputs are frame k-1 projections, map points and particles' camera poses. Temporal inputs are stored in RAMs. The update of temporal input is not considered. ORB features are input as a stream. For each

Fig. 7. Hardware framework

Table 4. Hardware performance

	Latency (ms)			
	Redundancy removal	Matching	Projection	Sum
Software	-	4-29	21-81	25-110
Conventional	0.0528	1.89	1.09	3.03
P1	0.00952	1.89	1.09	2.99
P1+P2	0.00952	1.89	0.271	2.17
P1+P2+P3	0.00952	0.177	0.271	0.457

Table 5. Performance with different parameters

	Latency (ms)			
	Redundancy removal	Matching	Projection	Sum
Parallelism				
16	0.00952	0.177	0.542	0.728
32	0.00952	0.177	0.271	0.457
64	0.00952	0.177	0.135	0.322

Table 6. Resource utilization with different parameters

	Utilization (used/total)		
	16	32	64
Parallelism			
# of LUT	21274 (3%)	42534 (7%)	99498 (16%)
# of Flip Flop	31076 (5%)	55743 (9%)	111508 (18%)
# of BRAM	74 (3%)	90 (5%)	106 (6%)
# of DSP	256 (13%)	480 (25%)	928 (48%)
Maximum frequency	135.969 MHz		

frame, redundancy removal is firstly done as a preparation. This part takes only 0.01 ms and is not a problem in practical system. After this preparation, when an ORB feature is inputted into observation, it is firstly matched with map points. Then the matched map point and the position of the feature are inputted into parallel projection to calculate reprojection error. To realize parallelism, particles' camera

poses are also stored in several RAMs so that they can provide poses simultaneously.

Hardware performance using 32 parallelisms is shown in Table 4. Compared with software result, hardware implementation has dramatic acceleration. Proposal 1 accelerates redundancy removal part. Proposal 2 accelerates projection part. Proposal 3 accelerates matching part. Using 32 parallelisms in projection is to balance the performance and resource utilization and is not fixed. Table 5 and Table 6 shows the performance and resource utilization using different parameters of parallelism. Because only projection part is in parallel, so when number of parallelism doubles, the processing time is not the half. Maximum frequency is tested using Xilinx ISE 14.7.

4. Conclusions

This paper proposes Search-free gridding based redundant projection removal, Non-iteration linear function based division and Temporal 2D local matching in particle filter based SLAM on FPGA. Search-free gridding based redundant projection removal reduces map points in complex pattern areas and reduce projection times. Non-iteration linear function based division is proposed to accelerate division in projection problem in SLAM. Temporal 2D local matching is proposed to reduce matching and keep accuracy. The proposed methods are tested on high frame rate sequences with challenging movement. Compared with ORB-SLAM⁽⁶⁾, the proposed method has 1.20° error in rotation and 1.90 cm error in translation averagely. Matching and projection parts which takes totally 88.6% time consumption in software are implemented on hardware and has 0.457 ms latency when 32 processing units are in parallel in projection. For future work, prediction and estimation in particle filter still need to be implemented. Resampling in estimation part is the only part in particle filter that is sequential and there are several methods to make it parallel. To make a complete hardware implementation, data communication of tracking with mapping, storage and update of particle filter's state vector and dataflow from camera are also needed to be considered.

Acknowledgment

This work was supported by KAKENHI (16K13006) and Waseda University Grant for Special Research Projects (2018B-234).

References

- (1) Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." *Mixed and Augmented Reality*, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007.
- (2) Hu, Tingting, and Takeshi Ikenaga. "Pixel Selection and Intensity Directed Symmetry for High Frame Rate and Ultra-Low Delay Matching System." *IEICE TRANSACTIONS on Information and Systems* 101.5 (2018): 1260-1269.
- (3) Kümmerle, Rainer, et al. "g 2 o: A general framework for graph optimization." *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on. IEEE, 2011.
- (4) Davison, Andrew J., et al. "MonoSLAM: Real-time single camera SLAM." *IEEE transactions on pattern analysis and machine intelligence* 29.6 (2007): 1052-1067.
- (5) Eade, Ethan, and Tom Drummond. "Scalable monocular SLAM." *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*. IEEE Computer Society, 2006.
- (6) Mur-Artal, Raul, and Juan D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." *IEEE Transactions on Robotics*, Vol. 33, No. 5, pp. 1255-1262, 2017
- (7) A. Davison. "Real time simultaneous localisation and mapping with a single camera". In *ICCV*, Nice, France, July 2003.
- (8) Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." *Computer Vision (ICCV)*, 2011 IEEE international conference on. IEEE, 2011.
- (9) D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- (10) Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2006.
- (11) Yuan Hu, Songlin Du, Takeshi Ikenaga, "3D Checking and Saturated Kernel Based GPU Accelerated Localization for ORB-SLAM", *International Workshop on Smart Info-Media Systems in Asia (SISA2017)*, Sep. 2017.
- (12) Yap, Teddy, et al. "A particle filter for monocular vision-aided odometry." *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on. IEEE, 2011.