Optimization and High-performance Hardware Implementation of Multipath Interference Compensation Algorithm

Chuan Liu¹, Shinan Lang¹*, Yiheng Cai¹ and Qiang Wu¹

¹Beijing University of Technology, No.100, Pingleyuan, Chaoyang District, Beijing, China

*Email: langshinan@gmail.com

Abstract. Our previous sparse regularization method for multipath interference (MPI) compensation in pulsed Time-of-Flight (ToF) depth camera works well on desktop computer. However, pulsed ToF depth cameras are mainly applied in fields such as autopilot vehicles and robots. In these situations, exporting depth images to a computer and running MPI compensation in MATLAB every time are not realistic. Therefore, according to the hardware structure of the multipath interference real-time compensation platform, we use QR decomposition to optimize the sensing matrix in our algorithm for better hardware implementation, while maintaining the accuracy of our algorithm. For the optimized algorithm, we designed an efficient parallel data processing hardware structure that matches the features of our algorithm to optimize the calculation efficiency. Then, we implemented it on the real-time compensation platform. The experimental results show that this algorithm architecture runs stably under the clock frequency and it runs faster than desktop computer.

1. Introduction
Active light source ranging systems have gradually replaced passive measurement systems recently and they are widely used in computer vision [1], autopilot vehicle [2], interactive game [3], smart home [4], medical [5], and other fields. The pulsed ToF imaging technology uses an active Near-InfraRed (NIR) light source. It is a leading range measurement method suitable for lidar imaging. Depth of every pixel on a surface is outputted simultaneously during a single measurement. The depth of each pixel represents the distance between the object in view and the camera lens. This technology enables direct measurement of three-dimensional structure of an object without the aid of traditional computer vision methods. Due to the short laser duration in the pulsed ToF depth imaging mechanism, this technology is superior to other competing technologies because of the inherent advantages of low power consumption, low heat generation, high frame rate, long measurable distance, and higher security. It is one of the leading technologies in the field of depth sensing.

However, there are a series of problems to be solved in this depth measurement technology, such as systematic error, lens distortion, flying pixel and multipath interference. Among them, multipath interference is our major concern in this paper. Multipath happens when the emitted laser returns through multiple paths. The cause of multipath is that the surface of an object is not a perfect reflective surface and the emitted laser will scatter when it reaches the object surface [6]. After multiple
scatterings, the depth camera sensor captures the combination of light from many paths, which causes serious distortion in depth estimation.

Previously, we proposed an algorithm for multipath interference compensation in pulse ToF depth cameras [7], and the experimental results showed improved accuracy in real scenes where the multipath normally causes range distortions. However, our cameras need to be applied in the fields of autopilot vehicles, robots, etc. It is not realistic to output the depth images to a desktop computer and process them in Windows and MATLAB environments every time. It results in large power consumption, high heat generation, and poor mobility. So, we need to implement the multipath interference compensation algorithm on the multipath interference real-time compensation platform. Real-time implementation of algorithms on embedded system has the advantages of fast speed, low power consumption, and good mobility.

This paper is divided into the following sections. Section 2 introduces our multipath interference compensation algorithm. Section 3 introduces the hardware structure of the multipath interference real-time compensation platform which we used to implement the algorithm. In Section 4, we present our optimization for the algorithm based on the hardware characteristics and propose the hardware architecture of our algorithm. At last, we show the results of hardware implementation in Section 5.

2. Multipath Interference Compensation Algorithm

In our previous paper [7], we introduced the multipath interference compensation algorithm. The focus of this paper is the implementation of our algorithm on embedded platform which we used to implement the algorithm. So we briefly introduce our algorithm theory in this section.

In pulsed ToF camera, depth is denoted as (1).

\[ d = \frac{c \cdot T_p}{2} \cdot \frac{S_1 - B(G)}{S_0 - B(G)} \]  

where \( T_p \) is half of the emitted pulsed light period, \( d \) is the distance to the object, \( c \) is the speed of light. \( T \Delta t \) is the time delay between emitted and reflected light. \( S_0 \) is the full reflected light collected by the first shutter, \( S_1 \) is the reflected light partially collected by the second shutter, and \( B(G) \) is the background light.

Multipath happens when the emitted laser returns through multiple paths. After multiple scatterings, the depth camera sensor captures the combination of light from many paths. When we take multiple measurements of the same scene, each under different emitted laser period \( T_p \), we get a measurement vector \( y \) which is denoted as (2).

\[
\mathbf{y} = \left( \begin{array}{c} d_1 \\ d_2 \\ \vdots \\ d_M \end{array} \right) = \left( \begin{array}{c} \Gamma_1 \cdot \frac{c \cdot T_p_1}{2} \cdot \frac{S_1(1)_{1} - B(G)(1)_{1}}{S_0(1)_{1} - B(G)(1)_{1}} + \cdots + \Gamma_N \cdot \frac{c \cdot T_p_1}{2} \cdot \frac{S_1(N)_{1} - B(G)(N)_{1}}{S_0(N)_{1} - B(G)(N)_{1}} \\
\Gamma_1 \cdot \frac{c \cdot T_p_2}{2} \cdot \frac{S_1(1)_{2} - B(G)(1)_{2}}{S_0(1)_{2} - B(G)(1)_{2}} + \cdots + \Gamma_N \cdot \frac{c \cdot T_p_2}{2} \cdot \frac{S_1(N)_{2} - B(G)(N)_{2}}{S_0(N)_{2} - B(G)(N)_{2}} \\
\vdots \\
\Gamma_1 \cdot \frac{c \cdot T_p_M}{2} \cdot \frac{S_1(1)_{M} - B(G)(1)_{M}}{S_0(1)_{M} - B(G)(1)_{M}} + \cdots + \Gamma_N \cdot \frac{c \cdot T_p_M}{2} \cdot \frac{S_1(N)_{M} - B(G)(N)_{M}}{S_0(N)_{M} - B(G)(N)_{M}} \end{array} \right) \]

where \( d_M \) represents the depth obtained by the \( M^{th} \) measurement. \( S_0(N)_M, S_1(N)_M \) and \( B(G)_M \) represent respectively the contribution of the \( N^{th} \) path to the intensity maps of \( S_0, S_1 \) and \( B(G) \) at the \( M^{th} \) measurement. \( \Gamma \) represents the reflection coefficient of the \( N^{th} \) path. \( T_p_M \) is half of the laser pulse at \( M^{th} \) measurement.

For a specific scene, the multipath effect is unchanged. The attenuation coefficient is the degree of light intensity attenuation after reflection. Assume that there are \( N \) paths, the light intensity attenuation coefficient of each path is different. Thus, there are \( N \) attenuation coefficients. These attenuation coefficients form an original signal vector:
\[ x = \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_N \end{pmatrix} \]  

(3)

Where \( I_N \) represents the reflection coefficient of the \( N^{th} \) paths.

We use measurement vector to observe the original signal. This process can be expressed as (4).

\[ y = \Phi x \]  

(4)

where \( y \) is the measurement vector obtained by us. \( \Phi \) is the sensing matrix. \( x \) is the original signal which is formed by attenuation coefficients of each light path.

The sensing matrix \( \Phi \) is constructed as (5).

\[ \Phi = \begin{pmatrix} \frac{c^T p_1}{2} * S(1)_1 - BG(1)_1 \\ \frac{c^T p_1}{2} * S(1)_2 - BG(1)_2 \\ \vdots \\ \frac{c^T p_M}{2} * S(1)_M - BG(1)_M \end{pmatrix} \]

\[ \begin{pmatrix} \frac{c^T p_1}{2} * S(2)_1 - BG(2)_1 \\ \frac{c^T p_2}{2} * S(2)_2 - BG(2)_2 \\ \vdots \\ \frac{c^T p_M}{2} * S(2)_M - BG(2)_M \end{pmatrix} \]

\[ \begin{pmatrix} \frac{c^T p_1}{2} * S(3)_1 - BG(3)_1 \\ \frac{c^T p_2}{2} * S(3)_2 - BG(3)_2 \\ \vdots \\ \frac{c^T p_M}{2} * S(3)_M - BG(3)_M \end{pmatrix} \]

\[ \vdots \]

\[ \begin{pmatrix} \frac{c^T p_1}{2} * S(N)_1 - BG(N)_1 \\ \frac{c^T p_2}{2} * S(N)_2 - BG(N)_2 \\ \vdots \\ \frac{c^T p_M}{2} * S(N)_M - BG(N)_M \end{pmatrix} \]

(5)

where \( S0(N)_M, S1(N)_M \) and \( BG(N)_M \) represent respectively the contribution of the \( N^{th} \) path to the intensity maps of \( S0, S1 \) and \( BG \) at the \( M^{th} \) measurement. \( I_N \) represents the reflection coefficient of the \( N^{th} \) path. \( T_p \) is half of the laser pulse at \( M^{th} \) measurement.

In order to reconstruct the original signal, we solve the \( L_1 \) norm problem:

\[ \min_{x \geq 0} \|x\|_1 \quad \text{s.t.} \quad y = \Phi x \]  

(6)

The overall flow of our original signal reconstruction algorithm is illustrated in Figure 1.

![Figure 1. Overall flow of our original signal reconstruction algorithm](image-url)

As discussed in our previous paper [7], our algorithm constantly optimizes the sensing matrix to find the closest estimation of the original signal \( x \). The output of the original signal reconstruction
algorithm is the optimal estimation of the original signal $x$, which consists of the attenuation coefficients of all paths, and the light intensity ratio corresponding to these coefficients.

The procedure of our signal reconstruction algorithm is to select the columns of the sensing matrix $\Phi$ which are most relevant to original signal $x$ by greedy iteration. We initialize the residual, index set, and number of iterations at first. During each iteration, the most relevant element between the residual and columns of the sensing matrix is found. Then we update the index set, record the reconstructed atom set, and calculate the estimation of $x$. When done, we update the residual and perform the next iteration until the number of iterations reaches the sparsity $K$.

After the original signal reconstruction algorithm is finished, we get the optimal estimation of the original signal $x$ and the optimized sensing matrix $\Phi$. Through positions of the non-zero attenuation coefficients and the corresponding light intensity ratio $\frac{S_{1-BG}}{S_{0-BG}}$, we can recover the true depth of the main path component using formula (1).

3. Optimization and Implementation of Multipath Interference Compensation Algorithm on the Real-time Platform

3.1. Hardware Architecture

Hardware architecture of our real-time compensation platform is illustrated in Figure 2.

![Figure 2. Hardware architecture of our real-time compensation platform](image)

As shown in Figure 2, we choose ZYNQ-7030-FFG676 FPGA chip as the core processor of the platform. There are two channels of differential clocks. The 156.25 MHz differential clock is provided on the platform, and there is an alternative clock provided on the backplane. There are two slave I2C signals and an EEPROM which is connected to the first I2C. EEPROM is for data buffer. Two DDR3 memory chip are connected to the ZYNQ chip for data storage which is needed during algorithm operations. UART interface is connected to the embedded system for downloading the input data.

3.2. Optimization of The Sensing Matrix

In order to accurately reconstruct the original signal $x$, the independence between columns of the sensing matrix needs to be as large as possible, and the correlation between the sensing matrix and the
original signal needs to be as small as possible [8]. Therefore, we need to optimize the sensing matrix constructed by us. We increase the column independence of the sensing matrix by QR decomposition.

According to the theory of compressed sensing [8], the greater the independence of the sensing matrix columns, the fewer the number of measurements needed. The column independence of the sensing matrix can be represented by the smallest singular value. The greater the singular value, the stronger the column independence.

We need to increase the smallest singular value of the sensing matrix without changing other properties. According to related literature [9, 10], QR decomposition meets our requirements. So we use the QR decomposition algorithm presented in [9] to optimize our sensing matrix.

3.3. Implementation of Multipath Interference Compensation Algorithm

We designed an efficient parallel data processing hardware structure that matches the OMP core for our algorithm. It uses pipelines to increase data throughput. The input of this algorithm is three series of variables: sensing matrix, measurement vector, and sparsity. The output is the reconstructed original signal and the sensor matrix after optimization. Our hardware architecture connects the input of the two-dimensional depth map to the signal reconstruction module with a highly efficient pipeline, and calculates the true depth map without interference using the output matrix of the signal reconstruction module.

Before executing the multipath interference compensation algorithm, we download the depth image input into our system using UART interface.

The key procedure of our signal reconstruction algorithm is to select the element from the columns of the sensing matrix which is most relevant to the current residual in each iteration [11, 12]. Then we add the selected element to the candidate subset, calculate a new estimation of the original signal, and update the residual. In the next iteration, the candidate subset is updated with the column of the sensing matrix which is most relevant with the new residual. After that, estimation of the original signal and residual are updated. We repeat this iteration until the times of iterations reaches the sparsity K.

The implementation steps of the signal reconstruction algorithm on the real-time compensation platform are presented next. Our hardware is designed for signal length of 256. The sparsity of the original signal is K = 8, and the length of the measurement vector is 64. The input data is 24-bit fixed-point format with ten integer bits and fourteen fractional bits. We initialize the residual with the value of the measurement vector, initialize the index set to empty set, and the times of iterations counter to 1.

The first step of our multipath interference compensation algorithm is dot product operation. Overall, there are 64 24-bit multipliers operating in parallel. Then the results are added together. The multiplier module used here is reused in the least square core later. Multiplication and addition are divided into three stages. The first stage performs the multiplication operation. The second stage performs eight addition operations in parallel, adding eight values each time. These eight results are added again in the third stage to get the final dot product result. This pipeline structure allows new data to be entered on the rising edge of each clock cycle.

Next step is finding the column of the sensing matrix which is most relevant to the current residual. This procedure is denoted as (7).

\[
\lambda_t = \arg \max_{j=1..N} | < r_{t-1} \cdot \phi_j > |
\] (7)

There is one dot product result output during each clock cycle. We find the column of the sensing matrix which is most relevant to the current residual by comparing all the values. After the maximum value is found, we update the index set and add the most relevant column to the optimized sensing matrix set. The product of the most relevant column and the corresponding coefficient is subtracted from the residual to update the residual. Through continuous iterations, the algorithm finds K most relevant columns. Our next step is solving the least square problem. This problem is denoted as (8).

\[
\hat{x}_t = \arg \min_x \| y - A_t \hat{x} \|_2
\] (8)
The steps of solving the least square problem involves finding the inverse matrix of $8 \times 8$ matrix. The inversion is used to estimate the original signal $x$ from (9).

$$ (A_t^T A_t) \hat{x} = A_t^T y $$

We assume $C = A_t^T A_t$ and the difficulty of finding inverse matrix of $C$ is proportional to the dimension of matrix. Thus, we choose QR decomposition process to find the inverse matrix. The inverse of $C$ is given by (10).

$$ C^{-1} = U^{-1} Q^T $$

Next, we refer to the method in [9, 13, 14] to find the inverse matrix using QR decomposition. When solving the norm in the method, eight multipliers are used in parallel. Here we reuse the eight multipliers we used in the dot product module. All inputs are connected to the multiplier through a multiplexer. The residual registers are not used after the optimization problem, so we reuse these registers for the matrices $C$ and $Q$, resulting in reduced use of resources and circuit area. The most important part in this hardware module is finding the inverse square root. We refer to Lomont’s fast inverse square root method [15]. The floating-point number is right shifted by one. The result after shifting is subtracted from a constant 0x5f3759df. A very close inverse square root approximation can be acquired using this method.

The final step is calculating an estimation of the original signal $x$. From the previous steps, we obtained $A_t$, $Q$, and $U^{-1}$. In the process of calculating the original signal estimation, we do not directly calculate $C^{-1}$. We calculate $A_t^T y$, then $Q^T A_t^T y$, and finally calculate (11).

$$ \hat{x} = U^{-1} Q^T A_t^T y $$

The purpose of skipping the direct calculation of $C^{-1}$ and breaking it into several steps is to improve efficiency of calculation.

4. Experiments and Results

We set up a scene which consists of two walls and the camera is facing the junction. The corner was located 60 cm in front of the pulsed ToF camera. The experimental setup and light intensity map collected is illustrated in Figure 3. The red line is the row of 640 pixels we chose for MPI compensation result verification.

![Figure 3](image-url)  
(a) Experimental setup, (b) light intensity map collected.

The hardware architecture of this paper is designed in Verilog and implemented on ZYNQ-7030-FFG676 board, which operates at 156.25 MHz. The development tool for the hardware is Xilinx Vivado 2018.3 design suite. Software simulation tool is the default simulation tool built in the Vivado 2018.3 design suite. Before the algorithm starts running, we download the required inputs using UART interface. For sparsity of eight, it takes 2600 clock cycles to find the eight most relevant columns of the sensing matrix. QR decomposition module takes 210 clock cycles. The number of clock cycles required to calculate the dot product of the sensing matrix column and the residual is relatively large. The dot product calculation requires $256 \times 8 = 2048$ clock cycles. The total time
required to run the whole multipath interference compensation algorithm is 31.09 μs. In our hardware structure, 64 points are reconstructed at a time. The 640 points we selected are divided into ten parts and processed in parallel. Uncalibrated depth, depth with multipath compensation in desktop computer, and depth with multipath compensation in our real-time platform are presented in Figure 4.

\[ RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\text{Reconstructed}_i - \text{GroundTruth}_i)^2} \]  \hspace{1cm} (12)

where the \( \text{Reconstructed}_i \) and the \( \text{GroundTruth}_i \) represents the reconstructed depth and ground truth, respectively.

The ground truth was acquired by commercial laser rangefinder. The final results are sent to a desktop computer through UART interface. We calculate the average RMSE of uncalibrated depth, the real-time compensation platform output, and the desktop computer output separately. RMSE of uncalibrated depth is 130 mm, RMSE of real-time compensation platform is 77 mm, and RMSE of desktop computer is 73 mm. The reconstruction accuracy of our hardware structure is close to that of desktop computer, but hardware structure runs faster. The comparison between the results of real-time compensation platform and the results of the same algorithm running in MATLAB on a quad-core Intel i5-4570 CPU is shown in Table 1.

| Device               | Signal Length | Sparsity | Frequency         | Time     | Average RMSE |
|----------------------|---------------|----------|-------------------|----------|--------------|
| Intel Core i5-4570   | 64            | 8        | 3.2 GHz           | 165 ms   | 73 mm        |
| Xilinx ZYNQ-7030-FFG676 | 64            | 8        | 156.25 MHz        | 31.09 μs | 77 mm        |

As can be seen from Table 1, the multipath interference compensation algorithm implemented on the real-time compensation platform runs faster than CPU without reducing reconstruction accuracy.

5. Conclusion
In this paper, we propose a highly pipelined multipath interference compensation algorithm hardware architecture implemented on the real-time compensation platform. This algorithm hardware architecture solves the problem of multipath interference in pulsed ToF depth images. The algorithm implemented on the platform runs faster than desktop computer, requiring only 31.09 μs for the whole algorithm. These results indicate the feasibility of real-time multipath interference compensation.
Acknowledgments
Thanks sincerely for the encouragement and support from my professor, family, and girlfriend during my research. You really helped me a lot in many ways. This work was supported by the National Natural Science Foundation of China (Nos. 41606219, 41776186), the Scientific Research Project of Beijing Educational Committee (No. KM 201910005027) and the Rixin Foundation of Beijing University of Technology.

References
[1] Marin, G., Zanuttigh, P., & Mattoccia, S. (2016, October). Reliable fusion of tof and stereo depth driven by confidence measures. In European Conference on Computer Vision (pp. 386-401). Springer, Cham.
[2] Liu, J., Sun, Q., Fan, Z., & Jia, Y. (2018, September). TOF Lidar development in autonomous vehicle. In 2018 IEEE 3rd Optoelectronics Global Conference (OGC) (pp. 185-190). IEEE.
[3] Lefloch, D. (2017). Real-time processing of range data focusing on environment reconstruction.
[4] Jiang, H., Cai, C., Ma, X., Yang, Y., & Liu, J. (2018). Smart home based on WiFi sensing: A survey. IEEE Access, 6, 13317-13325.
[5] Lee, C. C., Chen, C. J., Chen, S. C., Yang, H. C., Lin, C. J., Wu, C. C., ... & Wu, H. M. (2018). Gamma Knife surgery for clival epidural-osseous dural arteriovenous fistulas. Journal of neurosurgery, 128(5), 1364-1371.
[6] Dorrington, A. A., Godbaz, J. P., Cree, M. J., Payne, A. D., & Streeter, L. V. (2011, January). Separating true range measurements from multi-path and scattering interference in commercial range cameras. In Three-Dimensional Imaging, Interaction, and Measurement (Vol. 7864, p. 786404). International Society for Optics and Photonics.
[7] Liu, C., Lang, S., Wu, Q., & Zhang, J. (2019, October). Generalization of the Multipath Interference Compensation Algorithm in Pulsed Time-of-Flight Imaging. In 2019 IEEE Symposium on Product Compliance Engineering-Asia (ISPCE-CN) (pp. 1-5). IEEE.
[8] Donoho, D. L. (2006). Compressed sensing. IEEE Transactions on information theory, 52(4), 1289-1306.
[9] Stanislaus, J. L., & Mohsenin, T. (2013, January). Low-complexity FPGA implementation of compressive sensing reconstruction. In 2013 International Conference on Computing, Networking and Communications (ICNC) (pp. 671-675). IEEE.
[10] Abolghasemi, V., Jarchi, D., & Sanei, S. (2010, June). A robust approach for optimization of the measurement matrix in compressed sensing. In 2010 2nd International Workshop on Cognitive Information Processing (pp. 388-392). IEEE.
[11] Kulkarni, A., & Mohsenin, T. (2015, May). Accelerating compressive sensing reconstruction OMP algorithm with CPU, GPU, FPGA and domain specific many-core. In 2015 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 970-973). IEEE.
[12] Blache, P., Rabah, H., & Amira, A. (2012, July). High level prototyping and FPGA implementation of the orthogonal matching pursuit algorithm. In 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA) (pp. 1336-1340). IEEE.
[13] Yu, Z., Su, J., Yang, F., Su, Y., Zeng, X., Zhou, D., & Shi, W. (2016, May). Fast compressive sensing reconstruction algorithm on FPGA using Orthogonal Matching Pursuit. In 2016 IEEE International Symposium on Circuits and Systems (ISCAS) (pp. 249-252). IEEE.
[14] Bai, L., Maechler, P., Muehlberghuber, M., & Kaeslin, H. (2012, December). High-speed compressed sensing reconstruction on FPGA using OMP and AMP. In 2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012) (pp. 53-56). IEEE.
[15] Lomont, C. (2003). Fast inverse square root. Tech-315 nical Report, 32.