An accelerated SAR Back Projection Algorithm using Integer Arithmetic

Don Lahiru Nirmal Hettiarachchi* and Eric Balster†
* University of Dayton, Ohio, USA
E-mail: hettiarachchid1@udayton.edu
† University of Dayton, Ohio, USA
E-mail: ebalster1@udayton.edu

***Abstract*—Frequency-domain algorithms are used frequently to form Synthetic Aperture Radar (SAR) images from SAR video phase histories (VPH). Even though frequency-domain algorithms are computationally efficient, they work under many assumptions that do not hold for all imaging cases. Time-domain back projection (BP) algorithms can avoid such problems. The drawback of BP is that it requires a higher number of operations leading to a higher computational complexity in the order of N^3. Recently, various acceleration methods for time domain back projection have been developed in the radar processing community. This paper presents a new acceleration method for SAR BP using fixed-point arithmetic. It is shown that fixed-point based BP algorithm is faster than traditional algorithm and it maintains a high output image quality. The proposed algorithm process images with 15.69% speedup on average, while maintaining high quality image outputs.**

I. Introduction

Synthetic Aperture Radar (SAR) systems are designed to generate high resolution images. The resolution of an optical sensor depends on the distance to the target, but in SAR systems, the resolution depends on the nature of the transmitted signal. Therefore, SAR systems can maintain the same resolution even at a thousand meters or a thousand kilometers from the target. Due to these reasons, SAR processing is a very popular technique and has a wide variety of applications in both military and commercial fields. Once the SAR system receives the echoed signal from a target, image formation steps are carried out to generate the output image. There are many algorithms developed for SAR image formation and these algorithms are categorized as frequency-domain algorithms and time-domain algorithms.

Frequency-domain algorithms such as Range Doppler (RD), $\omega - k$ algorithm and Chirp Scaling (CS) are widely used due to their computational efficiency. Computational complexity of the frequency-domain is with order of $N^2 log_2 N$ [1]. However, frequency-domain algorithms work under many assumptions that do not hold for all imaging cases. Some algorithms depend on geometric approximations that neglect various limits (large image size, narrow angle swath, low squint angle, or wide bandwidth etc.). Few algorithms assume that the aircraft flies on a linear flight track. Due to this reason, off-track motion errors are only approximately compensated and that could result poorly focused images. Some frequency-domain algorithms need interpolation in frequency domain that could generate artifacts on the output image after conversion to time domain due to the interpolation errors [2].

Back projection (BP) is a time-domain algorithm that avoids the above mentioned problems with frequency-domain algorithms. One of the drawback of BP is that it requires a higher number of operations with order of N^3 which demands a significant processing time, where the N corresponds to the N pulses of echo data and $N \times N$ samples (per image). Due to its immense computational burden, the back projection algorithm is rarely used in real-time SAR processing or high resolution SAR image formation [3, 4].

In past two decades, many algorithms have been developed to accelerate traditional BP. Some algorithms are mainly derived by modifying the BP [1–6]. Other accelerations are achieved by utilizing high speed devices like general purpose graphics processing unit (GPGPU) and field programmable gate arrays(FPGA) [7–10].

In [2], Yegulalp et al., introduces a fast back projection (FBP) algorithm that reduces the computational complexity by $\sqrt{N}$. This modified back projection algorithm manages to retain all advantages of traditional back projection such as perfect motion compensation, unlimited scene size, perfect focus for higher bandwidth, and integration angle. The basic idea of this method is to divide the full synthetic aperture into sub-apertures. Each of these sub-apertures generate sub-images and finally, all sub-images are added coherently to get the final image. For a $N \times N$ pixel image with N_{pulses} of range compressed data, it is shown that approximately $N^{3/2} N_{pulses}$ operations are required. Compared with traditional back projection, this algorithm is faster by a factor of $\sqrt{N}$.

Ulander et al., modifies the traditional back projection algorithm and generalizes the FBP algorithm to introduce the fast factorized back projection (FFBP) algorithm [5]. FFBP method follows aperture divided to sub-aperture (SA) concept introduced in FBP. Then it assigns a local polar coordinate (LPC) system to each sub-aperture and follows a pyramid

computational architecture to complete the full aperture. Finally, a fusion technique and 2-D interpolation is carried out for all sub-apertures to get the back projected image.

Zheng et al., introduces an accelerated back projection (AFBP) algorithm by modifying the FFBP algorithm [3]. The AFBP technique also follows the basic concept introduced in FBP and FFBP methods. First, it divides the aperture into sub-apertures. Then, all sub-apertures are allocated a unified polar coordinate (UPC) system. To avoid 2-D interpolation, it converts all data into 2-D wavenumber domain. Finally, a sub-aperture fusion is carried out in 2-D WN domain and converts back to time domain in order to get the back projected output image.

Ref [6], develops a simulator for spotlight SAR image formation. Later in the same year, back projection algorithm of SAR simulator is accelerated using a GPGPU and OpenCL language [8]. The accelerated system obtained a speedup of 4X over single-threaded C++ implementations and a speedup of 19X over native MATLAB implementations.

This paper introduces a new method for back projection algorithm acceleration using fixed-point arithmetic. Back projection algorithm in the image processing module of spotlight SAR simulator [6] is used for modification. Proposed algorithm is optimized by incorporating multiple scale factors for fixed-point conversion and it is developed with OpenCL to test with multiple devices like CPUs, GPGPUs and FPGAs. Obtained results show that the proposed algorithm gains a $\sim$ 25% speed improvement for 512×512 sized images, a $\sim$ 12% for 256×256 sized images, and a $\sim$ 11% for 128×128 size images. Trade-off of accelerating back projection algorithm using fixed-point arithmetic is between image quality and speedup. Comparing the Peak signal-to-noise ratio (PSNR) values of output images, it is shown that the acceleration is obtained while preserving high image quality compared to traditional back projection.

Following the introduction, Section II includes an overview of SAR simulator and back projection. Section III introduces the proposed back projection algorithm. Section IV includes obtained results and analysis and Section V is dedicated for conclusion.

II. Spotlight SAR Model

A brief overview of image processing modules in spotlight SAR simulator [6] are introduced in this section. Fig. 1 shows the spotlight SAR imaging model that is used for derivation, where d_{so} is the standoff distance to the scene center, d_{alt} is the altitude, and θ is the horizontal angular displacement and ϕ is the incidence angle.

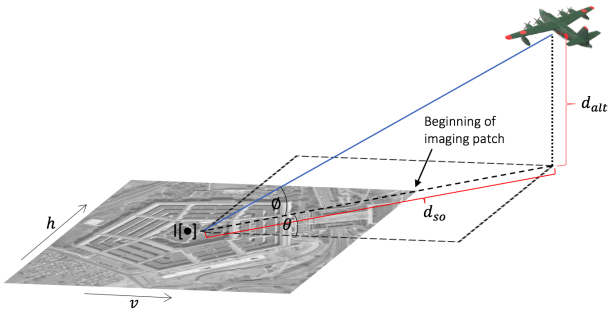


Fig. 1: Spotlight SAR imaging overview [6]

A. Generate Range Profile

The distance from aircraft to each pixel is given by

$$d_{ac}[v,h,\theta] = \sqrt{(vG - d_{so}\sin(\theta))^2 + (hG - d_{so}\cos(\theta))^2 + d_{alt}^2}, \quad (1)$$

where $v(\in [\frac{N}{2}, \frac{-N}{2}])$ and $h(\in [\frac{N}{2}, \frac{-N}{2}])$ are image indexes and G is the ground sample distance (GSD) and N represent number of rows and columns. Then the distance from each pixel to scene center is shown in (2).

$$d[v,h,\theta] = d_{ac}[v,h,\theta] - \sqrt{d_{alt}^2 + \left(d_{so} - \frac{NG}{\sqrt{2}}\right)^2} \quad (2)$$

The echoed pulse return based on individual pixel distances are shown in (3).

$$x_{ret}(t,\theta) = \sum_{v,h} I[v,h] x_p\left(t - \frac{2d[v,h,\theta]}{c}\right) \quad (3)$$

Where the $x_p(t)$ is the transmitted linear frequency modulated pulse and c is the speed of light. As shown in (4), the return signal ($x_{ret}(t,\theta)$) is demodulated by mixing with carrier signal ($c(t)$) and applying a low-pass filter.

$$x_{mix}(t,\theta) = F\{x_{ret}(t,\theta)c(t)\} \; ; \; c(t) = cos(2\pi f_l t) \quad (4)$$

The demodulated signal ($x_{mix}(t,\theta)$) is sampled and discretized to obtain $x_{mix}[n,\theta]$. Then, a matched filter is applied to generate the phase history ($X_{ph}[n,\theta]$). Equation (5) shows the match filter process in frequency domain.

$$X_{ph}[n,\theta] = X_{mix}[n,\theta] H_{mf}[n], \quad (5)$$

where $H_{mf}[n]$ is the impulse response of the matched filter. As shown in (6), the range profile ($R_p[k,\theta]$) is obtained by applying FFT and oversampling to the phase history.

$$R_p[k,\theta] = \sum_n X_{ph}[n,\theta] e^{-j\frac{2\pi n}{N}k} \quad (6)$$

B. Back Projection

The range profile ($R_p[k,\theta]$) is used to back project into a 2-D image space. After applying the phase correction, the projected data from all return signals are summed up to get the final image. The final back projected image output is obtained by

$$\tilde{I}[v,h] = \frac{1}{\Theta_N}\left|\sum_{\theta}\tilde{R}_p(i,\theta)e^{-j\frac{4\pi f_l d_R[v,h,\theta]}{c}}\right|, \quad (7)$$

where $d_R[v,h,\theta]$ is the differential range and $\tilde{R}_p(i,\theta)$ is the linearly interpolated range profile. Equations (8) and (9) show the d_R and $\tilde{R}_p$ respectively.

$$d_R[v,h,\theta] = d_{ac}[v,h,\theta] - \sqrt{d_{so}^2 + d_{alt}^2}, \quad (8)$$

$$\tilde{R}_p(i,\theta) = (\lfloor i+1\rfloor - i)R_p[\lfloor i\rfloor,\theta] + (i-\lfloor i\rfloor)R_p[\lfloor i+1\rfloor,\theta], \quad (9)$$

where $i = d_R[v,h,\theta]/d_s$ and d_s is the sample distance.

III. Proposed Back Projection Algorithm

The proposed back projection algorithm is introduced in this section. Idea is to convert all back projection variables into integers and use fixed-point arithmetic. First, new SAR model with new distance variables are introduced since the back projection method described in Section II-B uses angles to calculate distances. Converting variables associated with an angle into integers can increase the computational complexity and reduce the accuracy. Therefore, a new spotlight SAR system model with new distance variables are introduced and it is shown in Fig. 2.

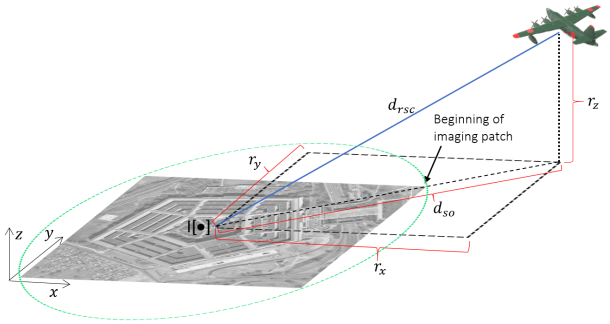


Fig. 2: SAR imaging overview for fixed-point processing

The fixed-point processing approach can be used to increase the speed of a calculation [11]. The precision of a floating-point variable can be preserved by multiplying with a constant scale value. Higher scale value yields a higher precision after the conversion. Finally, calculated fixed-point variables can be converted back to the floating-point variable by dividing with same constant scale value. The constant scale value is limited to power of 2 in order to achieve multiplication and division with binary shifts. Floating-point to fixed-point conversion is given by

$$\hat{F} = \lfloor 2^{\lambda} F \rfloor, \quad (10)$$

where F represents the floating-point variable, $\hat{F}$ is the fixed-point variable, and λ is the scale factor. In the proposed back projection method, all the floating-point variables used in back projection in Section II-B are converted to integers. The Table I consists of floating-point variables, fixed-point variables, and corresponding scale factors used for each conversion.

TABLE I: Floating-point variable, Scale factor, and Integer variable

Floating-point variables	Scale factor	Fixed-point variables
r_x, r_y, r_z	2^{λ_R}	$\hat{r}_x, \hat{r}_y, \hat{r}_z$
d_x, d_y, d_z	2^{λ_R}	$\hat{d}_x, \hat{d}_y, \hat{d}_z$
d_R	2^{λ_R}	$\hat{d}_R$
d_{rcp}	2^{λ_R}	$\hat{d}_{rcp}$
d_{rsc}	2^{λ_R}	$\hat{d}_{rsc}$
R_p	2^{λ_M}	$\hat{R}_p$
γ	2^{λ_C}	$\hat{\gamma}$
S_γ	2^{λ_R}	$\hat{S}_\gamma$
C_γ	2^{λ_R}	$\hat{C}_\gamma$
$\tilde{I}$	2^{λ_R}	$\hat{\tilde{I}}$

There are three scale factors defined in Table I which are used for the conversion. The λ_R scale is applied to distances from aircraft to scene center and aircraft to each pixel location. Since distances from aircraft to scene center and each pixel are large values and it plays a significant impact on overall accuracy, λ_R is considered to be a higher value. The λ_M scale is for range profile conversion and the values in range profile are significantly smaller compared to distance values measured from aircraft. Therefore, a smaller scale is used for λ_M. The λ_C value is fixed and it is used to calculate angles for phase correction.

A. Fixed-Point Conversion

First, the differential range $d_R[v,h,\theta]$ described in (8) is converted to integers. The new differential range, $\hat{d}_R[\xi]$ is shown in (11).

$$\hat{d}_R[\xi] = \hat{d}_{rcp}[\xi] - \hat{d}_{rsc}[\xi], \quad \xi \in (x,y,z), \quad (11)$$

where $\hat{d}_{rcp}[\xi]$ is the fixed-point variable distance between radar to current pixel and $\hat{d}_{rsc}[\xi]$ is the fixed-point variable distance between radar to scene center. The $\hat{d}_{rcp}[\xi]$ distance and $\hat{d}_{rsc}[\xi]$ distances are converted to fixed-points by applying (12).

$$\hat{d}_{rcp}[\xi] = \lfloor 2^{\lambda_R} d_{rcp}[\xi]\rfloor \quad \text{and} \quad \hat{d}_{rsc}[\xi] = \lfloor 2^{\lambda_R} d_{rsc}[\xi]\rfloor, \quad (12)$$

where,

$$d_{rcp}[\xi] = \sqrt{(r_x - d_x)^2 + (r_y - d_y)^2 + r_z^2}, \quad (13)$$

$$d_{rsc}[\xi] = \sqrt{r_x^2 + r_y^2 + r_z^2}. \tag{14}$$

The r_x, r_y, r_z in (13) and (14) represent the ranges from radar to scene center and current pixel to scene center (Fig. 2). The d_x, d_y represent the displacements from scene center to current pixel. Square root operations described in (13) and (14) are calculated using Newton's method (or Babylonian method) [12]. As shown in (15) and (16), all range values are converted to integers and the radicands are defined as $\hat{S}_a$ and $\hat{S}_b$. Iterative method is started by defining $\hat{X}_a$ and $\hat{X}_a$ as seed values. Since the radicands are already scaled with 2^{λ_R}, the $\hat{X}_a$ and $\hat{X}_b$ seed values are initiated with the same scaled constant as shown in (17). Then, the Newton's iterative method is applied for $\hat{S}_b$ and $\hat{S}_a$ using (18). The $\hat{X}_{k+1}$ is the $\hat{X}_k$ value for the next iteration.

$$\hat{S}_a = \hat{r}_x^2 + \hat{r}_y^2 + \hat{r}_z^2 \tag{15}$$

$$\hat{S}_b = (\hat{r}_x - \hat{d}_x)^2 + (\hat{r}_y - \hat{d}_y)^2 + \hat{r}_z^2 \tag{16}$$

$$\hat{X}_k = 2^{\lambda_R}, \quad k \in (a, b) \tag{17}$$

$$\hat{X}_{k+1} = \frac{1}{2}(\hat{X}_k + \frac{\hat{S}_k}{\hat{X}_k}), \quad k \in (a, b) \tag{18}$$

After Newthon's method converge to a fixed point, the values of $\hat{d}_{rsc}[\xi]$ and $\hat{d}_{rcp}[\xi]$ are obtained by considering,

$$\hat{d}_{rsc}[\xi] \equiv \hat{X}_{a+1} \quad \text{and} \quad \hat{d}_{rcp}[\xi] \equiv \hat{X}_{b+1}. \tag{19}$$

The differential range $\hat{d}_R[\xi]$ is obtained by using (11). After applying the phase correction, the range profile is back projected to 2-D image space. Back projected data from all return signals are summed up to get the final scaled-up image. Final scaled-up back projected image output is obtained by

$$\hat{\tilde{I}}[x, y] = \frac{1}{\Xi_N} \left| \sum_{\xi} \hat{\tilde{R}}_p(i, \xi) e^{-j\hat{\gamma}[\xi]} \right|, \quad \hat{\gamma}[\xi] = \frac{4\pi f_l \hat{d}_R[\xi]}{c}, \tag{20}$$

where $e^{-j\hat{\gamma}[\xi]}$ is the phase correction term and $\hat{\tilde{R}}_p(i, \xi)$ is a bi-linearly interpolated range profile given by

$$\begin{aligned} \hat{\tilde{R}}_p(i, \xi) = \\ \{(\lfloor i+1 \rfloor - i)\hat{R}_p[\lfloor i \rfloor, \xi] + \\ (i - \lfloor i \rfloor)\hat{R}_p[\lfloor i+1 \rfloor, \xi]\} 2^{\lambda_R - \lambda_M}, \end{aligned} \tag{21}$$

and i is given by

$$i = \frac{\hat{d}_R[\xi]}{\hat{d}_s}. \tag{22}$$

The $\hat{\tilde{R}}_p(i, \xi)$ in (21) is initially scaled up by 2^{λ_M} for range profile values and then by $2^{\lambda_R - \lambda_M}$ after interpolation to match the $e^{-j\hat{\gamma}[\xi]}$ scale. Finally, as shown in (23) the scaled-up back projected image output ($\hat{\tilde{I}}[x, y]$) is scaled-down by using the same constant scale value to obtain the final image output.

$$\tilde{I}[x, y] = \frac{\hat{\tilde{I}}[x, y]}{2^{\lambda_R}} \tag{23}$$

B. Angle Conversion with Fixed-Point Arithmetics

The Euler's formula is used to calculate the integer value of the phase correction term, $\hat{\gamma}[\xi]$ shown in (20). First, 2π range is divided by Q number of linearly spaced points, where $Q = \lceil 2\pi 2^{\lambda_C} \rceil$. Then, an integer array, $\hat{F}_{sin}[\cdot]$ is created to hold the Q number of 2^{λ_C} scaled values of sine angles.

$$e^{-j\gamma} = \hat{C}_\gamma - j\hat{S}_\gamma, \tag{24}$$

where $\hat{S}_\gamma = sin(\hat{\gamma}[\xi])$ and $\hat{C}_\gamma = cos(\hat{\gamma}[\xi])$. The value of $\hat{S}_\gamma$ is calculated by

$$\hat{S}_\gamma = \hat{F}_{sin}[\hat{\gamma}] 2^{\lambda_R - \lambda_C}, \tag{25}$$

and the value of $\hat{C}_\gamma$ is calculated by

$$\hat{C}_\gamma = \hat{F}_{sin}[(\hat{\gamma} + \frac{Q}{4})\%Q] 2^{\lambda_R - \lambda_C}, \tag{26}$$

where $\%$ is the modulus operator. Since $\hat{F}_{sin}[\cdot]$ holds 2^{λ_C} scaled values of sine angles, $\hat{S}_\gamma$ and $\hat{C}_\gamma$ in (25) and (26) are scaled up by $2^{\lambda_R - \lambda_C}$ to match the $\hat{\tilde{R}}_p(i, \xi)$ scale of (20). Fig. 3 shows the sine and cosine approximations calculated after scaling with $\lambda_C = 3$. An accurate approximation can be obtained by scaling up with a higher scale. After few accuracy tests, the scale value λ_C is fixed at 6 for better approximation.

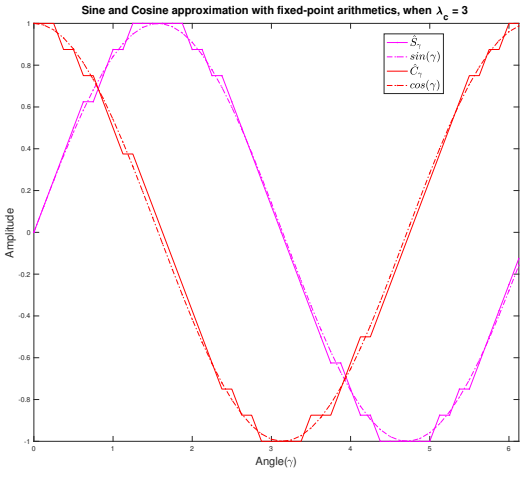


Fig. 3: Sine and Cosine approximation with fixed-point arithmetics

C. Scale Value Optimization

The λ_R scale value is used to scale large distances, i.e. aircraft to scene center and aircraft to each pixel location. The λ_M scale value is used to scale range profile distances. The main goal of this conversion is to accelerate the BP module. Therefore, several tests are carried out to find the optimum λ_R and λ_M values. Trade-off of the acceleration process is between output image quality and speedup. Image quality is calculated objectively using the peak signal to noise ratio (PSNR) of output images. In order to obtain optimum λ_R and λ_M values, the standoff distance(d_{so}), altitude (d_{alt}), and patch width (target area) parameters are varied to test on different configurations. Each parameter is varied 3 times to get a total of 27 test configurations per image. Varying standoff distance and altitude parameters change the range distances from aircraft to scene center and aircraft to each pixels. Scaling high varying distances with high scales like λ_R could result an arithmetic overflow. Therefore, to find the best PSNR value for each configuration, λ_R and λ_M are varied from 1 to 31. Fig. 4 shows the PSNR values calculated for all λ_R and λ_M combinations for one test configuration.

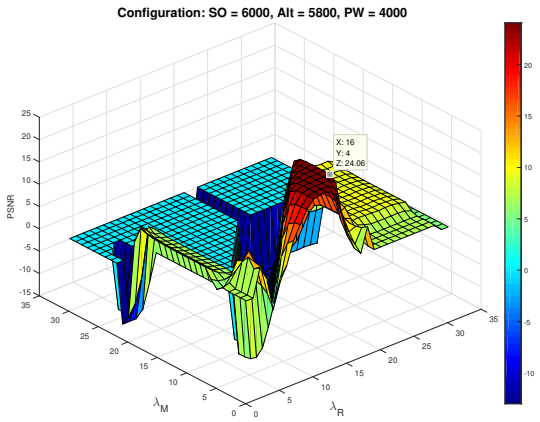


Fig. 4: PSNR variation with λ_R and λ_M. Test configuration: standoff distance $(SO) = 6000m$, altitude $(Alt) = 5800m$, patch width $(PW) = 4000m$

As shown in Fig. 4, best PSNR occurs when $\lambda_R = 16$ and $\lambda_M = 4$. Furthermore, it is shown in the plot that only a small area represents higher PSNR values throughout the grid. Some areas covered with low PSNR values are due to low quality (high noise) of images when scale values are low. Other low PSNR value areas are due to arithmetic overflow that generate lower quality images. All test configurations shows that the maximum PSNR value for each configurations is obtainable when λ_R is between 14 to 16 and λ_M is between 3 to 7. Analyzing PSNR values with all test configurations, λ_R is fixed to 16 and λ_M is fixed to 4 for further testing.

D. OpenCL Implementation of Proposed Back Projection Algorithm

Implemented BP module is shown in following algorithm. The algorithm uses the range profile ($\hat{R}_p[\cdot,\xi]$) input to create the BP image. To increase the processing speed each pixel in BP image is assigned as an independent work item.

Algorithm : The Proposed BP Module

1: `kernel void proposed_BP`$(\hat{\tilde{I}}[\cdot], \hat{R}_p[\cdot,\xi], P, N, \hat{d}_s, \hat{r}_x, \hat{r}_y, \hat{r}_z, \hat{G}, f_l)$
2: `int` $x, y, t, \hat{d}_R, \hat{S}_\gamma, \hat{C}_\gamma, \hat{\tilde{I}}_r, \hat{\tilde{I}}_i, \hat{J}_\gamma, \hat{\gamma}$;
3: `long` $\hat{X}_a, \hat{X}_b, \hat{d}_x, \hat{d}_y, \hat{S}_a, \hat{S}_b$;
4: $u =$ `get_global_id`(0);
5: $x = (int)(u/N)$;
6: $y = u\%N$;
7: $\hat{d}_x =$ `(long)` $(2 * y - N + 1) * \hat{G}$;
8: $\hat{d}_y =$ `(long)` $(N - 2 * x - 1) * \hat{G}$;
9: $\hat{J}_\gamma =$ `(int)` $(B * f_l)$;
10: `for`$(n = 0; n < N; n++)\{$
11: $\quad \hat{S}_a = \hat{r}_x[n] * \hat{r}_x[n] + \hat{r}_y[n] * \hat{r}_y[n] + \hat{r}_z[n] * \hat{r}_z[n]$;
12: $\quad \hat{S}_b = (\hat{r}_x[n] - d_x) * (\hat{r}_x[n] - d_x) + (\hat{r}_y[n] - d_y) * (\hat{r}_y[n] - d_y) + \hat{r}_z[n] * \hat{r}_z[n]$;
13: $\quad \hat{X}_a = (\hat{X}_a + S_a/\hat{X}_a) >> 1$;
14: $\quad \hat{X}_b = (\hat{X}_b + S_b/\hat{X}_b) >> 1$;
15: $\quad \hat{d}_R = \hat{X}_a - \hat{X}_b$;
16: $\quad i =$ `(int)` $\hat{d}_R/\hat{d}_s +$ `(int)` $(P + 1) >> 1$;
17: $\quad$ `if` $(\hat{d}_R \geq 0)$
18: $\quad\quad i += 1$;
19: $\quad$ `if` $(i > 0$ `&&` $i < P)\{$
20: $\quad\quad \hat{\gamma} = (\hat{F}_Y * \hat{d}_R) >> \lambda_R$;
21: $\quad\quad \hat{\gamma} = \hat{\gamma}\%Q$;
22: $\quad\quad$ `if` $(\hat{\gamma} < 0)$
23: $\quad\quad\quad \hat{\gamma} += Q$;
24: $\quad\quad \hat{S}_\gamma = \hat{F}_{sin}(\gamma) << (\lambda_R - \lambda_C)$;
25: $\quad\quad \hat{C}_\gamma = \hat{F}_{sin}[(\gamma + Q >> 2)\%Q] << (\lambda_R - \lambda_C)$;
26: $\quad\quad t = [i + (P + 1) >> 1] * \hat{d}_s - \hat{d}_R$
27: $\quad\quad \hat{\tilde{R}}_{pr} = (\hat{R}_{pr}[i-1] * t + \hat{R}_{pr}[i] * (\hat{d}_s - t))/\hat{d}_s$;
28: $\quad\quad \hat{\tilde{R}}_{pi} = (\hat{R}_{pi}[i-1] * t + \hat{R}_{pi}[i] * (\hat{d}_s - t))/\hat{d}_s$;
29: $\quad\quad \hat{\tilde{I}}_r = ((\hat{\tilde{R}}_{pr} * \hat{C}_\gamma - \hat{\tilde{R}}_{pi} * \hat{S}_\gamma) >> \lambda_M + \hat{\tilde{I}}_r * n)/(n+1)$;
30: $\quad\quad \hat{\tilde{I}}_i = ((\hat{\tilde{R}}_{pr} * \hat{S}_\gamma + \hat{\tilde{R}}_{pi} * \hat{C}_\gamma) >> \lambda_M + \hat{\tilde{I}}_r * n)/(n+1)$;
$\}\}$

First, Lines 1-3 initialize the kernel and variables. Then in Line 4, the pixel index (u) of the BP image is obtained and Line 5-6 show the (x, y) position indexes of the BP image. Line 7 and 8 calculate the distances from current pixel to the scene center in x and y directions. The term N represents the BP image size (i.e. $N \in \{128, 256, 512\}$) and $\hat{G}$ is the ground sampling distance. Line 9 calculates the constant part of $\hat{\gamma}$, where $B = \frac{4\pi}{c}$. Line 10 calculates the intensities for all pixels in BP image. Lines 11-15 show the implementation of Newton's fixed point iteration method to calculate the differential range $\hat{d}_R$. Lines 16-18 show the calculation steps

of the range profile sample indexes corresponding to the current pixel (u) in the BP image. Line 19 constraints the sample i within the range profile, where P represents the number of FFT samples. Lines 20-23 calculate the phase correction $\hat{\gamma}$, where $Q = \lceil 2\pi 2^{\lambda_C} \rceil$. Lines 24 and 25 calculate the sine and cosine values of $\hat{\gamma}$ angle. Lines 26-28 show the calculation steps for bi-linear range profile interpolation. Lines 19-30 calculate the BP image intensities after applying phase correction. Image output of BP module (in (20)) requires complex math calculation. Therefore, as shown in Lines 27-30, real samples ($\hat{\tilde{R}}_{pr}, \hat{\tilde{I}}_r$) and imaginary samples ($\hat{\tilde{R}}_{pi}, \hat{\tilde{I}}_i$) of range profile $\hat{R}_p[\cdot, \xi]$ and BP image $\hat{\tilde{I}}[\cdot]$ are calculated separately.

IV. Results

The proposed back projection module is tested with synthetically generated video phase history (VPH) from the SAR simulator [6]. The images tested are shown in Fig. 5. These images are processed with SAR simulator to generate synthetic VPH, then VPH of each image are used for SAR processing. Usage of synthetic VPHs are beneficial for performance evaluation for SAR imagery techniques, such as calculation of PSNR comparison with original images.

A. Speed increase with fixed-point processing

All images shown in Fig. 5 are 512×512 in size, and are downsampled to form 256×256 and 128×128 images to test the BP module with different image sizes. Furthermore, OpenCL 1.2 is used to develop the BP algorithm and measure the execution time for floating-point and fixed-point versions. All tests are carried out on an 4-core Intel core I7-4810MQ 2.8 GHz processor and 64-bit Microsoft Windows 7 Enterprise operating system. Modern CPUs calculations are optimized for floating-point arithmetics and therefore CPU optimizations are disabled to maintain a fair comparison. Execution time for back projection module with floating-point and fixed-point (proposed) methods shown in Fig. 6. Different sized images are tested and it is shown in Fig. 6 that for all images, execution time of fixed-point version of the back projection is faster than floating-point version. Also, note that higher the image size, faster the fixed-point back projection compared to floating-point version.

As shown in the Fig. 6, 512×512 sized images are processed with a 24.99% speedup on average, 256×256 sized images are processed with a 11.51% speedup on average, and 128×128 sized images are processed with a 10.57% speedup on average. Therefore, the average speedup is 15.69% for all images.

B. Image Quality Comparison

It is shown in Section IV-A, for all test images, the fixed-point BP module executes faster than floating-point BP module. To verify proper functionality of the fixed-point BP module, quality of test images are measured using PSNR index. All calculated PSNR (dB) values are shown in Table II. Note that the change of PSNR values from floating-point BP to fixed-point BP are very low. On average, for all images the PSNR value is decreased by 0.1924 dB. Moreover, it is shown that when the image size increases, speedup percentage increases (Fig. 6) and PSNR error decreases.

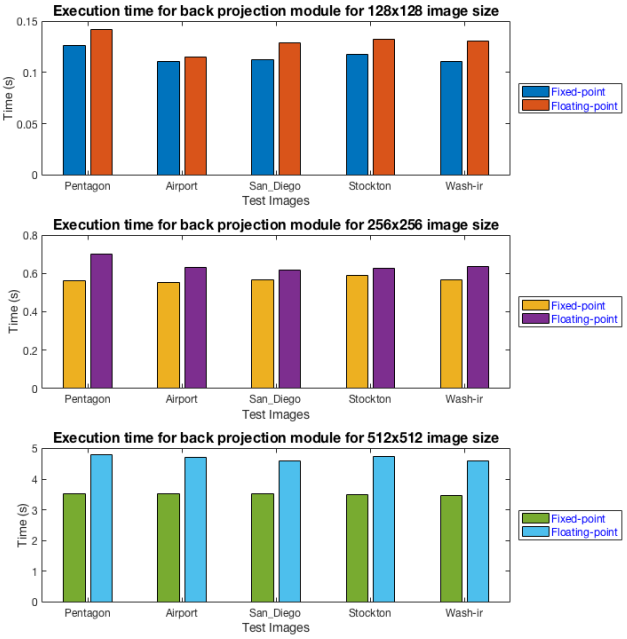


Fig. 6: Execution time for back projection module with floating-point and fixed-point (proposed) methods.

TABLE II: PSNR (dB) results of all test images.

Image Size	Test images	Floating-point	Fixed-point	Error (Δ)
128x128	Pentagon	24.2095	24.4420	-0.2325
	Airport	18.7690	18.3764	0.3926
	San Diego	19.6278	19.0534	0.5744
	Stockton	19.8428	19.5710	0.2718
	Wash-ir	17.9474	17.3517	0.5957
256x256	Pentagon	23.8985	23.6778	0.2207
	Airport	19.9064	19.7595	0.1469
	San Diego	23.0957	23.0090	0.0867
	Stockton	25.7948	25.6556	0.1392
	Wash-ir	18.3788	18.2248	0.1540
512x512	Pentagon	25.2779	25.2561	0.0218
	Airport	20.7656	20.6733	0.0923
	San Diego	23.8377	23.7267	0.1110
	Stockton	27.3165	27.3144	0.0021
	Wash-ir	19.5251	19.2161	0.3090
Average Error				**0.1924**

Floating-point back projection (BP) module outputs and fixed-point BP module outputs for test images are shown in Fig. 7 for a visual comparison. As expected, image outputs from both modules are visually very similar. However, above back projected output images are generated using synthetic VPH. Therefore, both BP modules are tested with real SAR imagery data. Fig. 8 shows the back projected final outputs from both modules. The 'parking lot' image from GOTCHA dataset, which is very common in the radar processing com-

(a) Left: Pentagon image, center: Airport image, right: San Diego image

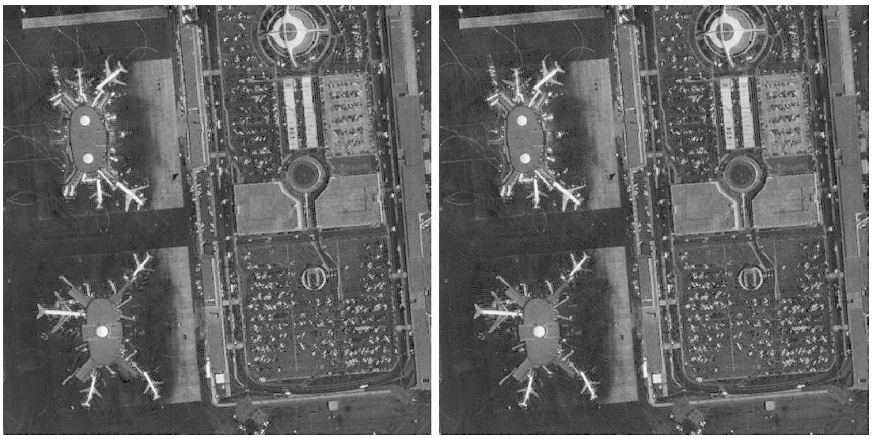
(b) Left: Stockton image, right: Wash-ir image

Fig. 5: Original images used to create synthetic VPHs

munity is used for testing. The PSNR value is not available because of no image reference. Floating-point BP module process SAR data and output the image in 84.27 seconds and fixed-point BP module process it in 74.63 seconds achieving a 11.4% speedup. As expected, fixed-point BP output is visually similar to the floating-point BP output.

V. Conclusion

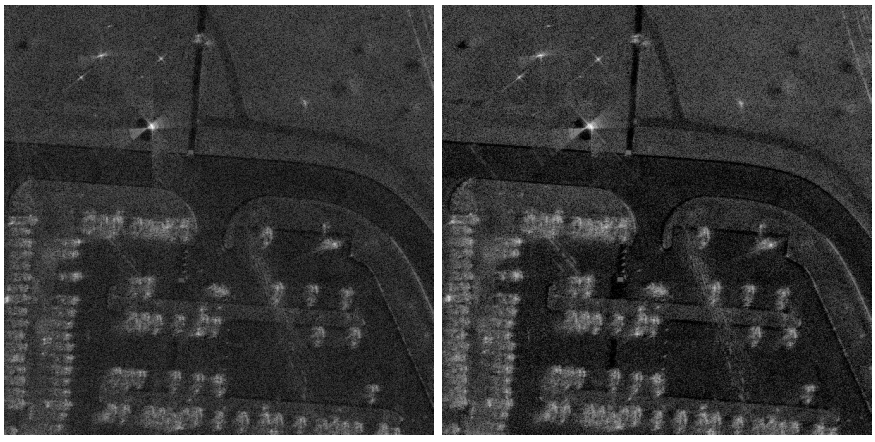
This paper presents a new acceleration method for SAR back projection using fixed-point arithmetics. Proposed algorithm, is tested with various images with different sizes and compared with traditional floating-point BP module. It is shown that fixed-point based BP algorithm is faster than traditional BP algorithm and it maintains a higher output image quality. Proposed algorithm process images with 15.69% speedup on average and average PSNR decrease for all images is 0.1924 dB.



(a) Airport image

(b) San Diego image

Fig. 7: Output image comparison. Left: floating-point BP output, Right: fixed-point BP output



(a) Left: Floating-point BP module output (b) Left: Fixed-point BP module output

Fig. 8: Back projected 'parking lot' image from GOTCHA dataset

REFERENCES

[1] H. L. Li, J. Li, Y. X. Hou, L. Zhang, M. D. Xing, and Z. Bao, "Synthetic aperture radar processing using a novel implementation of fast factorized back-projection," in *IET International Radar Conference 2013*, pp. 1–6, April 2013.

[2] A. F. Yegulalp, "Fast backprojection algorithm for synthetic aperture radar," in *Proceedings of the 1999 IEEE Radar Conference. Radar into the Next Millennium (Cat. No.99CH36249)*, pp. 60–65, 1999.

[3] L. Zhang, H. l. Li, Z. j. Qiao, and Z. w. Xu, "A fast bp algorithm with wavenumber spectrum fusion for high-resolution spotlight sar imaging," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, pp. 1460–1464, Sept 2014.

[4] L. Ran, Z. Liu, T. Li, R. Xie, and L. Zhang, "An adaptive fast factorized back-projection algorithm with integrated target detection technique for high-resolution and high-squint spotlight sar imagery," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, pp. 171–183, Jan 2018.

[5] L. M. H. Ulander, H. Hellsten, and G. Stenstrom, "Synthetic-aperture radar processing using fast factorized back-projection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 760–776, July 2003.

[6] E. J. Balster, F. A. Scarpino, A. M. Kordik, and K. L. Hill, "A simulator for spotlight sar image formation," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–5, Jan 2017.

[7] B. Ge, L. Chen, D. An, and Z. Zhou, "Gpu-based ffbp algorithm for high-resolution spotlight sar imaging," in *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–5, Oct 2017.

[8] E. J. Balster, M. P. Hoffman, J. P. Skeans, and D. Fan, "Gpgpu acceleration using opencl for a spotlight sar simulator," in *Proceedings of the 5th International Workshop on OpenCL*, IWOCL 2017, (New York, NY, USA), pp. 1:1–1:5, ACM, 2017.

[9] F. Cholewa, M. Wielage, P. Pirsch, and H. Blume, "Synthetic aperture radar with fast factorized backprojection: A scalable, platform independent architecture for exhaustive fpga resource utilization," in *International Conference on Radar Systems (Radar 2017)*, pp. 1–6, Oct 2017.

[10] D. Pritsker, "Efficient global back-projection on an fpga," in *2015 IEEE Radar Conference (RadarCon)*, pp. 0204–0209, May 2015.

[11] J. C. French and E. J. Balster, "A fast and accurate orthorectification algorithm of aerial imagery using integer arithmetic," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, pp. 1826–1834, May 2014.

[12] O. Kosheleva, "Babylonian method of computing the square root: Justifications based on fuzzy techniques and on computational complexity," in *NAFIPS 2009 - 2009 Annual Meeting of the North American Fuzzy Information Processing Society*, pp. 1–6, June 2009.