GPU Based Robust Image Registration for Composite Translational, Rotational and Scale Transformations

Semih Dinc
Department of Computer Science
University of Alabama in Huntsville
Huntsville, AL USA
sd0016@uah.edu

Ramazan S. Aygun
Department of Computer Science
University of Alabama in Huntsville
Huntsville, AL USA
aygunr@uah.edu

Farbod Fahimi
Mechanical and Aerospace Engineering
University of Alabama in Huntsville
Huntsville, AL USA
ff0002@uah.edu

***Abstract*—This paper presents a GPU based image registration algorithm that utilizes Hough Transform and Least Square Optimization to calculate the transformation between two images. In our approach, we calculate the transformation parameters of all possible combination solutions of matched feature points by exploiting parallel processing power of the GPU. We applied our algorithm on a variety of images including the problem of mosaic image generation. Experimental results show that our method is robust to the outliers (incorrect matches) and it can achieve very accurate registration (numeric and visual) results with much faster (up to 20 times) than CPU implementation.**

***Keywords*-Image Registration, GPU Programming, CUDA, Hough Transform, Least Squares Optimization**

I. Introduction

Image registration is the process of aligning multiple images using distinctive visual features of the images [1]. It is commonly used in many applications such as medical imaging, geospatial visualization or mosaic image generation and in the literature, there are a large number of image registration studies [2] [3] having different perspectives. All these CPU based methods suffer from the trade-off between speed and accuracy, since extracting and matching image features is not a trivial task to complete. Moreover, these algorithms are generally iterative and they cannot be parallelized directly. GPU architectures may provide better accuracy without sacrificing the speed. There are a few image registration studies using GPU based solutions. However, most of these studies are limited to a problem domain or do not support significant transformations. Kubias et al. [4] studied the 2D/3D image registration for only translation and rotation on X-Ray like images. Kohn et al. [5] proposed both rigid and non-rigid image registration system on medical images but the registration accuracy is not provided in the paper. Sinha et al. [6] proposed a GPU based feature tracking technique using SIFT and KLT. In that paper, matching features are tracked but image frames are not registered.

We propose an alternative GPU based image registration technique using NVIDIA CUDA libraries [7]. Our system computes the transformation by exhaustively checking all possible combinations of matched feature points using the parallel power of GPU and the Hough transform [8]. Since feature points are processed exhaustively, our method supports transformations with significant translation, scaling, or rotation. Our experiments show significant speed-ups with respect to CPU time while generating robust image registration results.

II. Calculation of Transformation Parameters

Image registration requires determining a geometric transformation including *Translation*, *Rotation*, and *Scale*, which involves 4 parameters to calculate: t_x, t_y (translation), s (scale), and θ (rotation) given in Equation (1),

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1}$$

where (x', y') is a pixel coordinate in original image and (x, y) is the corresponding pixel in transformed image. When 4 transformation parameters are known, this transformation matrix can be used to register moving image to the fixed image space. Equation (1) can be solved by reorganizing its terms such that all unknowns stay as a single vector T with the coefficient matrix Q on one side of the equation, and constant vector W stays on the other side (Eq. (2)). At least 2 pixel matches are necessary to solve this system, since Q must be non-singular.

$$\overbrace{\begin{bmatrix} x' \\ y' \end{bmatrix}}^{W} = \overbrace{\begin{bmatrix} x & -y & 1 & 0 \\ y & x & 0 & 1 \end{bmatrix}}^{Q} \overbrace{\begin{bmatrix} s\cos\theta \\ s\sin\theta \\ t_x \\ t_y \end{bmatrix}}^{T} \tag{2}$$

Solving all unknowns individually requires nonlinear calculations. In order to keep all computations linear, we assumed the entries of T are our 4 unknowns because solving T is sufficient to find the transformation. Based on these considerations, if we rewrite Eq. (2) for the following unknowns as $a = s\cos\theta$, $b = s\sin\theta$, $c = t_x$, $d = t_y$, and add the second match to the equation, the final form of the equation becomes

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \end{bmatrix} = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{3}$$

The equation can be solved using any technique as long as Q has full rank. We prefer Least Squares Optimization (LSO) because it is simple and effective method. LSO is independent from the number of matches in the equation and robust to uniform or Gaussian noise on pixel coordinates. Equation (4) shows the calculation of the T vector using least squares.

$$T = (Q'Q)^{-1}(Q'W) \tag{4}$$

When T is calculated, the original transformation parameters can be obtained. c and d are assigned to t_x and t_y, respectively. s and θ are obtained using the following calculations: $s = \sqrt{a^2 + b^2}$ and $\theta = atan2(b/a)$.

III. GPU Based Robust Image Matching

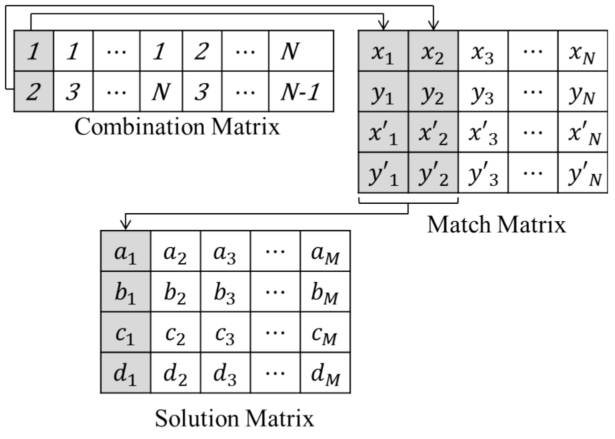
LSO may be negatively affected in the presence of incorrect matches (outliers). Feature extraction techniques (such as SIFT) usually extract outliers, which may cause a failure in the calculations. In the literature, RANSAC [9] like methods are preferred to eliminate the outliers, however, they are mostly CPU based solutions and our experiments showed that using such methods still may not be accurate enough. To solve this problem, we propose a robust approach that calculates all possible solutions and searches for a common solution in the solution space. This method is an exhaustive approach following the idea of Hough transform, which transforms the problem into the parameter space and searches for a common solution. As mentioned earlier, this idea may not be practical for CPU implementation but can be a good solution for GPU implementation.

A. Method

Assume that N matching features are extracted using SIFT descriptor. Since only 2 matches are needed to solve unknowns, we can choose all possible combinations of 2 matches out of N. So there will be $M = \binom{N}{2}$ number of equation systems, for which we solve 4 unknowns and store the results. Our assumption is that the majority of matches are correct and those will return correct solution. Actually, the majority assumption can be relaxed further, and our method may work even if the correct combinations are less than the half of the matches but if they are the most popular. In this manner, we can eliminate the incorrect matches that cannot agree on a single solution. After all combinations are solved, correct solutions will be closer to each other in the solution space and incorrect solutions will be outliers. Finally, if we calculate the histogram in 4 dimensional space and group the solutions based on their distances, the most populated histogram bin (with the most number of matches) will be the final solution to the problem. In our experiments, we prefer the range of [-500, 500] with 0.01 interval for the histogram bins of t_x and t_y. Histogram for s is selected in [0, 3] range using 0.001 interval. And finally histogram for θ has $[-\pi, \pi]$ range with 0.001 interval. The complexity of our solution is $O(n^2)$, where n is the number of matches and the power 2 is related to the number of combinations necessary.

B. GPU Implementation

In CUDA kernel, we employ two input matrices and one output matrix as given in Figure 1. One CUDA thread is assigned to one combination in the *combination matrix* that stores the match indices. The thread loads the corresponding matching pixels from the *match matrix* and calculates the unknowns for that equation. Finally it stores the solution into the corresponding index of the *solution matrix*.



1 | 1 | ⋯ | 1 | 2 | ⋯ | N
2 | 3 | ⋯ | N | 3 | ⋯ | N-1
Combination Matrix
x_1 | x_2 | x_3 | ⋯ | x_N
y_1 | y_2 | y_3 | ⋯ | y_N
x'_1 | x'_2 | x'_3 | ⋯ | x'_N
y'_1 | y'_2 | y'_3 | ⋯ | y'_N
Match Matrix
a_1 | a_2 | a_3 | ⋯ | a_M
b_1 | b_2 | b_3 | ⋯ | b_M
c_1 | c_2 | c_3 | ⋯ | c_M
d_1 | d_2 | d_3 | ⋯ | d_M
Solution Matrix

Fig. 1. Pixel matches generated by SIFT

There are $M = \binom{N}{2}$ number of solutions in the *solution matrix*. Since the calculation of a solution is independent, the equations can be solved simultaneously. The best solution is determined by selecting the most populated bin center of the solution histogram.

IV. Experiments

A. Registration Results

1) Real Image Registration: Since our registration algorithm calculates 4 transformation parameters (a rigid transform of images), real images can be problematic due to perspective or affine effect. For this reason, we captured images where there is only rotation, translation, and scale. Figure 2 shows the registration results. In the first example, two images are registered where there is a significant rotation difference (about 45 degrees) and very minor translation and scale difference. In the second example, two images are captured where there is a significant scale and translation.

2) Application to Mosaic Image Generation: Our second experiment is mosaic image generation. A mosaic image is large composite view of an object or a scene, which is formed by a collection of small images that partially covers the object [10]. For the experiments, 3 synthetic video datasets (having 50, 150×150 resolution frames) are generated. All

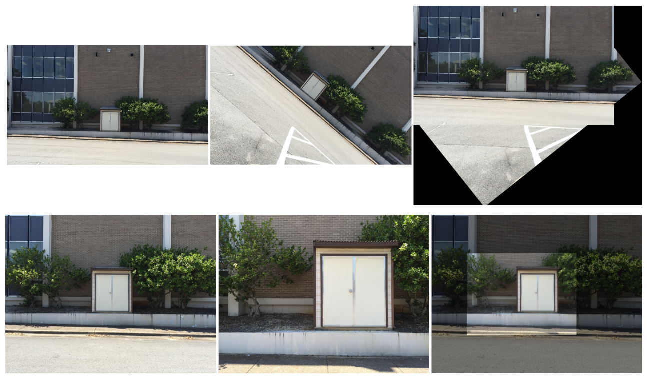


Fig. 2. Image registration for significant rotation (top row), for significant translation and scale (bottom row)

3 transformations (translation, rotation, and scale) are applied to subimages and then they are saved in the synthetic video. In Figure 4, the first column shows the original images, while the rest of images in each row are sample frames. In the first dataset *(painting)*, we performed a composite transformation with significant amount of translation (0 to 400 pixels in x and y directions), rotation (0 to π), and scale (1 to 0.5). In the second dataset (*shuttle*), only translational transformations (-150 to 150 pixels in x and y directions) are applied with a circular motion on the original image. Finally, in the third dataset (*earth*), significant translational (0 to 300 pixels in x, -150 to 150 pixels in y direction) and scale (1 to 1.7) factors are applied. Figure 3 shows the captured frames from the original image in white rectangular regions.

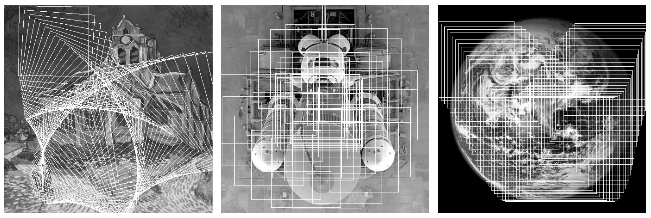


Fig. 3. Generating video frames from 3 datasets. White rectangular windows represent transformed frames on the synthetic videos

To generate the mosaic image, video frames are processed in order, using our algorithm. A weighted averaging strategy is used for overlapping pixels after registration. To evaluate the accuracy, the mean of squared error (*MSE*) is calculated for each transformation parameter and also the peak signal-to-noise ratio (*PSNR*) is measured between the original and the mosaic image. The results are compared with another GPU based image registration method called *imregdemons* provided in MATLAB (M-GPU). *MSE* results in Table I show a significant improvement in the second dataset for all parameters. In the first dataset, there is a good amount of improvement in s and θ and slightly better results for t_x and t_y. Finally in the third dataset, significant accuracy increase is achieved for t_x, t_y, and s. M-GPU could not generate proper mosaic for this dataset, since it made a transformation error that propagated to the rest of the frames. In general, both methods generate acceptable *PSNR* values, however, there is a clear superiority of the proposed method over M-GPU on the second dataset. Visual results of our method are also provided in Figure 5 that shows no significant defect on the mosaic images. In such applications, it is often unavoidable to lose information (or quality) due to scaling up frames. Particularly, this problem can be seen in the central regions of *earth* dataset, where the frames need to be enlarged up to 1.7 times.

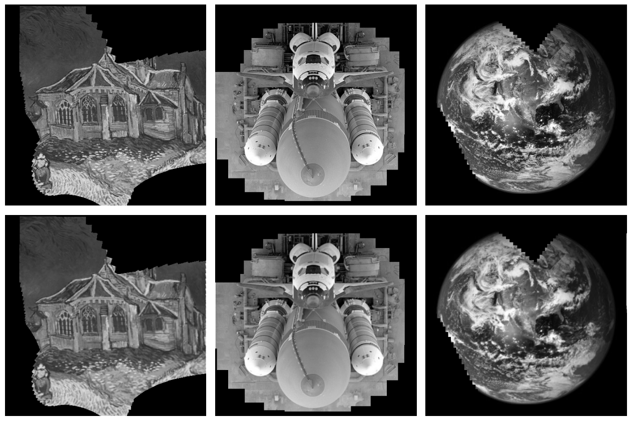


Fig. 5. First row shows the ground truth images and second row shows the results of our algorithm.

TABLE I
MSE AND PSNR VALUES AND IMPROVEMENT RATIO OF THE PROPOSED METHOD

		e_{t_x}	e_{t_y}	e_s	e_θ	PSNR
Proposed Method	Painting	0.192	1.895	0.001	0.000	28.341
	Shuttle	0.079	0.034	0.000	0.008	27.630
	Earth	0.012	0.005	0.000	0.000	27.383
M-GPU	Painting	0.227	2.009	0.001	0.006	28.965
	Shuttle	0.777	0.168	0.015	0.035	22.984
	Earth	0.057	0.073	0.005	0.005	27.891
Imp(%)	Painting	15.1%	5.6%	92.2%	93.5%	N/A
	Shuttle	89.7%	79.31%	99.7%	97.9%	N/A
	Earth	79.2%	93.1%	97.6%	-6.9%	N/A

B. Performance Results

1) Processing Times: Total processing time of our algorithm depends on feature extraction and image transformation, while the complexity of M-GPU depends on directly image size since it uses a hierarchical registration method without feature extraction. In Table II, the processing time of two methods are measured by examining 10 different sizes of the same image pair from 326×245 (10%, 65 feature points) to 3264×2448 (100%, 1543 feature points). In the table feature extraction times are not included, however, this stage can also be done quite fast [11], in which it takes around 0.11 seconds to extract 2500 features from a 1080p video frame. So, even if feature extraction is included in timing, our method performs significantly faster than M-GPU.

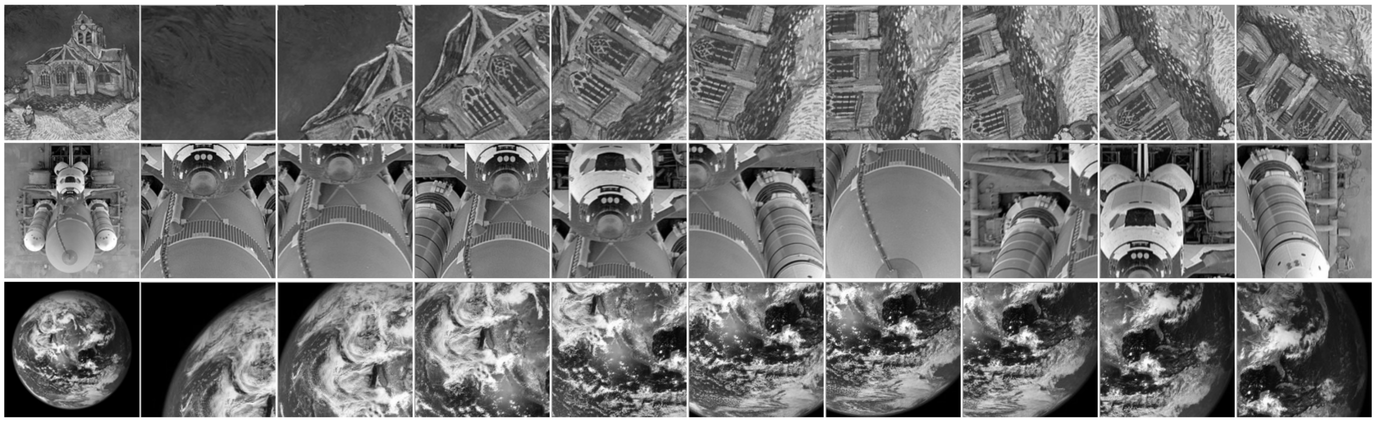


Fig. 4. A subset of all frames in our 3 datasets. First frames of the rows represent the original image and rest of them are transformed frames for that dataset. In *painting* (first row), there is significant amount of rotation, translation and scale. In *shuttle* (second row), there is only translation. Finally in *earth* (third row), there is significant translation and scale applied to the frames.

TABLE II
TOTAL PROCESSING TIMES OF THE PROPOSED METHOD AND M-GPU FOR VARYING IMAGE SIZE (IN SECONDS)

	10%	20%	40%	60%	80%	100%
Proposed*	0.043	0.044	0.048	0.053	0.061	0.072
M-GPU	1.727	2.208	3.998	6.842	10.402	15.213

* without feature extraction time.

2) CPU vs. GPU Implementation: In the GPU side of the program, first data is transferred from RAM to GPU memory and then kernel function is executed. After the execution is done, results are sent back to the memory. We assigned 1 dimensional block of threads and 1 grid in the CUDA kernel. The block size is assigned to 256, which is a proper value for our hardware. In CUDA, we used "shared memory" and "tiling" techniques to reduce the total data transfer time. A sequential CPU version of our algorithm is also implemented for a CPU vs GPU comparison. We ran the code on NVIDIA GeForce 320M graphics card and Intel Core 2 Duo 2.4 GHz CPU. According to the results in Table III, CPU runs faster than GPU up to 160 matching features. After that point, GPU becomes faster and the difference is getting larger as the number of matching pixels gets larger.

TABLE III
CPU AND GPU PROCESSING TIMES DEPENDING ON THE NUMBER OF FEATURES (IN MILLISECONDS)

#Features	CPU (ms)	GPU Total (ms)	GPU Kernel (ms)
100	13.926	48.02	0.045
200	56.261	38.355	0.049
300	127.992	44.607	0.066
400	226.608	45.184	0.084
500	352.415	49.598	0.09
600	504.607	53.366	0.071
700	686.567	48.19	0.077
800	891.314	51.646	0.071

V. CONCLUSION AND FUTURE WORK

This paper presents a GPU based image registration method using the idea of Hough transform to calculate the transformation. Massively parallel execution capability of GPU is exploited to solve the combinations of matching image pixels extracted by SIFT descriptor. Our algorithm is applied to real and synthetic images achieving high precision on registration and much fast processing times compared to CPU. As future work, we plan to enhance our method for affine and perspective transformations.

REFERENCES

[1] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[2] Lisa Gottesfeld Brown, "A Survey of Image Registration Techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992.

[3] Yudong Zhang and Lenan Wu, "Rigid Image Registration by PSOSQP Algorithm," *Advances in Digital Multimedia*, vol. 1, no. 1, pp. 4–8, 2012.

[4] A Kubias, F Deinzer, T Feldmann, D Paulus, B Schreiber, and Th. Brunner, "2D/3D image registration on the GPU," *Pattern Recognition and Image Analysis*, vol. 18, no. 3, pp. 381–389, 2008.

[5] Alexander Köhn, Johann Drexl, Felix Ritter, Matthias König, and Heinz-Otto Peitgen, "GPU accelerated image registration in two and three dimensions," in *Bildverarbeitung f{ü}r die Medizin 2006*, pp. 261–265. Springer, 2006.

[6] Sudipta N Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc, "Feature tracking and matching in video using programmable graphics hardware," *Machine Vision and Applications*, vol. 22, no. 1, pp. 207–217, 2011.

[7] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron, "Scalable Parallel Programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008.

[8] D H Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[9] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[10] Yi Chen and Ramazan Savas Aygün, "Synthetic Video Generation for Evaluation of Sprite Generation," *International Journal of Multimedia Data Engineering and Management*, vol. 1, no. 2, pp. 34–61, Jan. 2010.

[11] Hannes Fassold and Jakub Rosner, "A real-time gpu implementation of the sift algorithm for large-scale video analysis tasks," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2015, pp. 940007–940007.