Proceedings of the Federated Conference on Computer Science and Information Systems pp. 293–301
DOI: 10.15439/2019F282
ISSN 2300-5963 ACSIS, Vol. 18

Creating See-Around Scenes using Panorama Stitching

Saja Alferidah
King Faisal University
Saudi Arabia, Alahsa
Email:saja.alferidah@gmail.com

Nora A. Alkhaldi
King Faisal University
Saudi Arabia, Alahsa
Email: nalkhaldi@kfu.edu.sa

***Abstract*—Image stitching refers to the process of combining multiple images of the same scene to produce a single high-resolution image, known as panorama stitching. The aim of this paper is to produce a high-quality stitched panorama image with less computation time. This is achieved by proposing four combinations of algorithms. First combination includes FAST corner detector, Brute Force K-Nearest Neighbor (KNN) and Random Sample Consensus (RANSAC). Second combination includes FAST, Brute Force (KNN) and Progressive Sample Consensus (PROSAC). Third combination includes ORB, Brute Force (KNN) and RANSAC. Fourth combination contains ORB, Brute Force (KNN) and PROSAC. Next, each combination involves a calculation of Transformation Matrix. The results demonstrated that the fourth combination produced a panoramic image with the highest performance and better quality compared to other combinations. The processing time is reduced by 67% for the third combination and by 68% for the fourth combination compared to stat-of-the-art.**

I. Introduction

THE STUDY of panoramic imaging is one of the advanced research topics in the field of computer vision, graphics and image processing [1]. Panorama Stitching is defined when two or more images of the same scene are taken by rotating a camera about its axis. As a result of this process a wider panorama image is created by overlapping the common contents of each component image [2]. In 1997, Szelinski and Shum defined creating a larger panorama image as the integration and overlapping the common contents of two or more images of the same scene by rotating the camera about its axis. In 2017, Wand et al. defined panorama stitching as taking multiple images with an overlapping area and stitching them together into a single wide image [3][4]. In 2015, Heekyeong Jeon et al. classified the panorama stitching process as the three core steps of detecting features, matching them, and stitching [2]. Early panorama images were created by sliding a slit-shaped aperture across a photographic film. The digital approach of today extracts thin, vertical strips of pixels from the frames of a sequence captured by a translating video camera. The resulting image is considered as multi-viewpoint (or multi-perspective), because different strips of the image are captured from multiple viewpoints [4]. Strip panoramas are created from a translating camera with many variants, such as "pushbroom panoramas" [5], "adaptive manifolds" [6], and "x-slit" images [7]. Contrary to the hardware-based approach, many researchers have explored the multi-perspective renderings of 3D models [8][9]. Yu and McMillan presented a model that describes a multi-perspective camera [10]. Panoramic image stitching is used in a variety of environment, including gaming, virtual reality, virtual museums, and map applications [11]. Microsoft Research, for example, is spending on research projects featuring panorama stitching techniques, and many algorithms are designed to efficiently facilitating the creation of panoramic images through stitching [12][13].

II. Background

Most researchers classify panorama stitching as either a direct technique or a feature-based technique [11][14]. The direct technique compares pixel to pixel between images and the feature-based technique compares all features within each image [14]. This paper applies the feature-based technique as it is more advanced, faster, and flexible when compared to the direct technique. Producing a panorama stitching for two or more images of the same object is divided into three steps. First, the process discovers the points of interest between several images (the keypoints) and extract vector features around each of these points of interest (the descriptors). Second, identifies the matching lines between several images using the extracted features after that match the correct features and remove incorrect features. Third, find the transformation matrix that satisfies matching with the other keypoints, and use this transformation to align the two images before merging. Panorama stitching is considered through two perspectives. The first is camera rotation, where images are acquired with the camera positioned at the same point while being rotated to provide multiple views of the same object. The second perspective is camera translation, where the camera is not fixed at the same position but is moved through a linear translation to capture the second image. This paper focuses on the second prospective where two images are taken for the same scene and with a slight linear displacement.

Consider a car moving towards an intersection with a large building on the corner obstructing its view. If an image is taken from a point ahead of its current position and stitched with another image at its current position, such that the integrated image shows the two overlapped as a semi-transparent view, then this image can enable drivers to have a partial view of the scene behind the building. This work helps to create a vision

effect around the image, Figure 1, graphically explains this scenario. The first camera (Camera 1) captures one image of

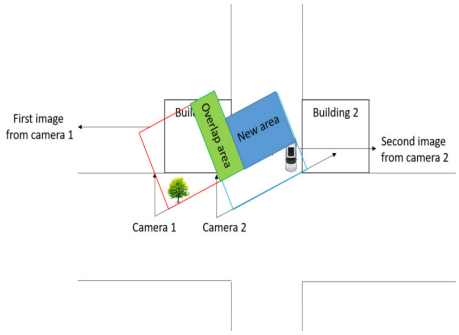


Fig. 1: The panorama stitching problem illustrated

building 1, which is shown as a red square. The blue square is the image captured from the second camera (Camera 2) that captures part of building 1 and part of building 2. The green rectangle represents the overlapping area between the two images and the blue area behind the green rectangle is the portion obscured by the building. This paper employs seven techniques, which are combined into four hybrid models, as shown in Figure 2, to create panoramic images. The four

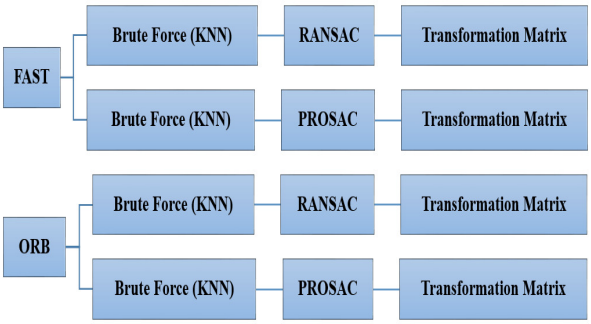


Fig. 2: The selected solution for four hybrid approaches

combinations include: (1) $FAST$, Brute Force (KNN) and $RANSAC$; (2) $FAST$, Brute Force (KNN) and $PROSAC$; (3) ORB, Brute Force (KNN) and $RANSAC$; (4) ORB, Brute Force (KNN) and $PROSAC$. Then each combination is followed by the calculation of a Transformation Matrix. The results of these models are compared to the model proposed in [2]. Basically, the model in [2] utilized ORB, Hamming distance, $PROSAC$ and the Transformation Matrix to produce a stitched image. This model will be referred to as the fifth combination here. Next sections will discuss the seven implemented techniques.

A. FAST

The $FAST$ technique is a high-speed corner detector method [15] defined by having a pixel A surrounded by a sufficient quantity of neighborhood pixels with a different grayscale value. In this scenario, the pixel A is recognized as

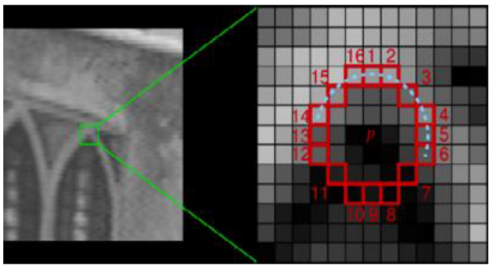


Fig. 3: $FAST$ Feature Point Detection [15]

a $FAST$ corner and applies to grayscale images. The $FAST$ neighborhood must contain enough pixels with values greater than, less than or similar to that of pixel A. We choose an arbitrary pixel as the center to establish a circular area, to be considered as the pixel point's neighborhood [16]. As shown in Figure 3, a discrete circle of radius 3 with pixel p as the central pixel has neighborhood pixels labeled 1 to 16. If pixel 16 has sequential n pixels that satisfy the equation [16]

$$|I_x - I_p| > t, \tag{1}$$

then, we consider p as a candidate feature point, t is a given threshold value, I_x is the gray value of the sequential n pixel, and I_p is the gray value of pixel p, [17]. For features extraction and descriptors computation in the first and second hybrid combinations based on using $FAST$, the Binary Robust Invariant Scalable Keypoints ($BRISK$) algorithm is incorporated, [23], because $FAST$ can only detect corner features but dose not compute the descriptors. Therefore, this paper uses $BRISK$ descriptor with $FAST$ keypoints.

B. BRISK

$BRISK$ is a binary descriptor that calculate the weighted Gaussian average over selected points near the keypoint [23]. For specific pairs of Gaussian windows $BRISK$ compare values that could be either a 1 or a 0 depending on which window in pair was greater [23]. $BRISK$ descriptor applies the sampling pattern around the keypoints [23]. The sampling pattern rotated α angle around the keypoint k. The α is calculated by [23]:

$$\alpha = \arctan 2(g_y, g_x), \tag{2}$$

where g_x and g_y are the gradients sum. The bit vector descriptor d_k is collected by execute for all point pairs the short distance intensity comparisons [23].

$$(p_i^{\propto}, \sigma_i) \in S, \tag{3}$$

such as every bit b corresponds to:

$$b = \begin{cases} 1, & I(p_j^{\propto}, \sigma_j) > I(p_i^{\propto}, \sigma_i) \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

$$\forall (p_i^{\propto}, p_j^{\propto}) \in S, \tag{5}$$

where $I(p_i^{\propto}, \sigma_i)$ is gray intensity after rotated α angle around the keypoint k and S is gray intensity for the short distance

pairs set. At the end, $BRISK$ uses a deterministic sampling pattern introduce a uniform sampling-point density [23].

C. ORB

The ORB technique is based on improved $FAST$ and the Binary Robust Independent Elementary Features $BRIEF$ feature detector techniques to extract points of interest by using a binary string [18]. Since $FAST$ and $BRIEF$ process quickly, the ORB will also be fast [15]. While the $FAST$ technique is not sensitive to noise and is highly reliable for identifying feature points, it does not provide an orientation. However, ORB incorporates orientation into $FAST$ with the $oFAST$ algorithm. The $BRIEF$ approach finds descriptors around each feature point by using a binary coding method [19], which is simple and requires less memory compared to $SIFT$ and $SURF$ [19]. Consider p is a smoothed image patch defined on the size of $S*S$ (Where S contains the coordinates of pixels) round feature points and a binary random selected test defined as τ,

$$\tau(p;x;y) = \begin{cases} 1, & \text{p(x)<p(y)} \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

where $P(x)$ is the pixel intensity at the $x = (u,v)^T$ point [3][20][16]. After filtering is performed, a set of points can uniquely identify one binary detection τ [16]. Therefore, the features defined as a vector of n binary strings is the same as,

$$f_n(p) = \sum_{1 \le i \le n} 2^{i-1} \tau(p;x;y), \tag{7}$$

[3][20][16]. Since $BRIEF$ is not scale invariant, ORB solves this issue by adding a direction into $BRIEF$ by defining patch moments as [3][20][16]

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y), \tag{8}$$

where p and $q \in \{0,1\}$ is binary selector for x and y direction and (x,y) is the position of the $FAST$ feature point. The circular neighborhood radians are $r,x,y \in [-r,r]$ [21][22], and the moment is reordered (centroid) as C [3][20][22], such that

$$C = \left(\tfrac{m_{10}}{m_{00}}, \tfrac{m_{01}}{m_{00}}\right). \tag{9}$$

When assuming a center vector O to the centroid $\vec{oc}$, then the offset is defined as

$$\Theta = \arctan[\frac{m_{01}}{m_{10}}] = \arctan[\frac{\sum_{x,y} yI(x,y)}{\sum_{x,y} xI(x,y)}], \tag{10}$$

[3][20][16]. Therefore, ORB extracts the $BRIEF$ descriptor based on the direction performed by Equation 6. The random ORB uses a greedy algorithm to find the random pixel block with low correlation and vector length equal to a 256-bit feature descriptor named $BRIEF$, [16], for which some previous research used a different type of test, such as the Gaussian distribution [3].

D. K-Nearest Neighbor

One image matching algorithm is KNN that take a set of query points Q and set of references point R [24]. Then, check for each query point $q \in Q$, compute distance between q and all $r \in R$, sort the computed distance in list [24]. Finally, select K nearest reference points corresponding to k smallest distance [24]. A threshold ratio value is then checked to determine if it is a good matching point, which requires the process to loop until at least four matches are found to compute the Homography [25]. This paper uses the Brute Force matcher, a simple version of the KNN, to match the descriptors of the images.

E. RANSAC

$RANSAC$ is a robust technique used to estimate the Homography and remove outlier points randomly from images to provide good matches [11] and increase quality [2]. $RANSAC$ randomly select a set of data required to calculate a mathematical model of data parameters [26][16]. Then, with an effective random sample [16], $RANSAC$ uses a small number of points to estimate the model and check if it agrees with the remaining points by calculating their distance to the fitted model. $RANSAC$ can be performed N times until a subset of the image is found with a good matching relationship [11].

F. PROSAC

The $PROSAC$ technique is used to remove outliers points progressively from images to obtain good matching results [26]. This algorithm performs the same steps as $RANSAC$ gradually and not randomly, which reduces the required operation time and the number of repetition when the sufficient process of verification is completed [2]. Two problems need to be addressed in $PROSAC$, first is the growth function [26],

$$n = g(t), \tag{11}$$

which is defined as the set U_n of n (where U is set of features) arranged progressively and sampled after trials t are selected [19]. Second, $PROSAC$ like $RANSAC$ provides guarantees about the stopping criterion for the optimal solution, which must be found [26].

G. Transformation Matrix

The transformation matrix defines x in an image B with x' as the final panorama image for calculating a new position of pixels [27][3],

$$x' \sim H_x, \tag{12}$$

where x is a new position of pixels in image B, x' is a position of pixels in the final panorama,$\sim$ finds the similarity up to scale, and H is a 3×3 matrix that can be calculated using the Direct Linear Transform algorithm [27]

$$H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}, \tag{13}$$

where points x and x' are defined as [27]

$$x_i = \begin{pmatrix} x_i \\ y_i \\ w_i \end{pmatrix}, x'_i = \begin{pmatrix} x'_i \\ y'_i \\ w'_i \end{pmatrix}, \quad (14)$$

where x_i , y_i is the keypoint position and w_i is set to 1. The final equation after subsequent transformation [27] becomes:

$$\begin{pmatrix} 0^T & -w'_i x_i^T & y'_i x_i^T \\ w'_i x_i^T & 0^T & -x'_i x_i^T \\ -y'_i x_i^T & x'_i x_i^T & 0^T \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0. \quad (15)$$

Two linearly independent equations. In addition, this can be written as [27]:

$$A \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} = 0. \quad (16)$$

We add two question on matrix A for each pair of points.

III. Experimental Results

The machine used in this work is Windows 10 with 64-bit operating system. The application uses two cameras with 1373×2382 image resolution to capture sets of images for testing the panorama stitching algorithms. The experiments are performed using 15 different scenes, each with two captured images. Contrast Differences is used to evaluate the quality of the four hybrid combinations to determine if the stitched images are seamless as the seam is considered poor when it is visible.

A. Contrast Differences

Contrast differences are the variances in luminance between neighboring pixels that make them distinguishable [28]. In this paper, the differences in the contrast value between the stitched images and the original image check the quality of the four stitched images. Equation 17 shows the image contrast calculation I'_k , where I_k is the image in the vertical direction, is determined by:

$$I'_k(i,j) = I_k(i+1,j) - I_k(i,j), \quad (17)$$

for $1 \leq i < H$ and $1 \leq j \leq L$. To evaluate the quality between the original and the four stitched images, the area of the original image and the four stitched images is divided [28] as illustrated in Figure 4 with the stitched image $I_{k,k+1}$ and the two halves of the overlapping area from the original images A and B and t_v is the horizontal translation.

The left half of the overlapping area is mainly contributed by the left half of the stitched image from A'. The right half of overlapping area is mainly contributed by the right half of the stitched image from B'. So, the contrast values of A and B are subtracted from contrast values of A' and B' to calculate the contrast difference values. The performance measures d_A and d_B for the A and B regions are then calculated as [28].

$$d_A = \sum_{i=1}^{H} \sum_{j=1}^{t_v} |I'_k(i, L_k - t_v + j) - I'_{k,k+1}(i, L_k - t_v + j)| \quad (18)$$

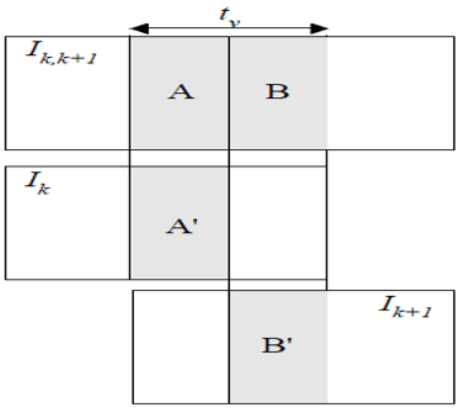


Fig. 4: Regions for comparison [28]

$$d_B = \sum_{i=1}^{H} \sum_{j=1}^{\frac{t_v}{2}} |I'_{k+1}(i, \frac{t_v}{2} + j) - I'_{k,k+1}(i, L_k - \frac{t_v}{2} + j)| \quad (19)$$

where the L and H are the width and height of the images, respectively [28]. Beside using Contrast Differences, this paper use Peak Signal-to-Noise Ratio ($PSNR$) and Root Mean Square Error ($RMSE$) to evaluate quality by calculating the error rate.

1) Mean Squared Error (MSE).

$$MSE(x,y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - y_i)^2, \quad (20)$$

where N is number of pixels, x and y are signals, the error signal $e_i = x_i - y_i$ is differences between two signal [29].

2) Peak Signal-to-Noise Ratio (PSNR).

$$PSNR = 10 \cdot log_{10}(\frac{L^2}{MSE}), \quad (21)$$

where $L = 2^8 - 1 = 255$ for an 8-bit per pixel image [29]. High value of PSNR means better quality and less noise.

3) Root Mean Square Error (RMSE).

$$RMSE(I,J) = \sqrt{MSE(I,J)} = \sqrt{\frac{\sum_{j=1}^{m} \sum_{i=1}^{n} (I_{ij} - J_{ij})^2}{m \times n}}, \quad (22)$$

Where I,J are two image matrices [29].

Figures 5 and 6 show two images for one building from different angle, referred to as Data 0. Figures 7 and 8 show two images captured for same scene, referred to as Data 1. Figures 9, 10, 11 and 12 show the stitched images for the first, second, third and Fourth combinations of Data 1, respectively. Figures 13, 14, 15 and 16 show the stitched images for the first, second, third and Fourth combinations of Data 1, respectively. Figure 17 Provides analysis of the processing time in seconds as shown in (a), the $PSNR$ as shown in (b) and $RMSE$ as shown in (c), for the five combinations using 10 different

Fig. 5: First Image of Data 0

Fig. 6: Second Image of Data 0

Fig. 7: First Image of Data 1

Fig. 8: Second Image of Data 1

Two images of the same building were taken from two different angles as shown in Data 0 and Data 1.

scenes. The fifth combination refers to the method proposed in [2], as mentioned earlier. In Figure 17 (a), it is clear that the processing time of third and fourth combination take minimum time to process stitched panorama images. From Figure 17 (b) and Figure 17 (c) it is apparent that the fourth combination produce better result on most $PSNR$ and $RMSE$. Table I shows the number of keypoints and matching points from testing Data 0. Table II shows the number of keypoints and matching points from testing Data 1. Table III show the first and second hybrid combinations results for 10 data (i.e. scenes) that contains two set of images with image resolution of 1373×2382. Table IV show the third and fourth hybrid combination results for 10 data. Table V show the fifth hybrid combination results for 10 data.

IV. Discussion

This paper provided four different combinations that used for panorama stitching and compared their output images based on processing time and quality. The third model reduced the processing time by 67% for the ORB, Brute Force (KNN), $RANSAC$, and Transformation Matrix compared to the fifth model, [2], that used ORB, Hamming distance, $PROSAC$, and the Transformation Matrix. The proposed fourth model reduced the processing time by 68% for the ORB, Brute Force (KNN), $PROSAC$, and Transformation Matrix compared to the fifth model, [2]. The fourth model that used ORB, Brute Force (KNN), $PROSAC$, and Transformation Matrix is shown better performance and quality results compared to other combinations. In particular, using ORB, Brute Force (KNN), $PROSAC$, and Transformation Matrix in the fourth model is shown better results of $PSNR$ and $RMSE$ compared to other combinations, as illustrated in Tables III, IV and V. Table I and Table II are showing the results for two different scenes that are referred to as Data 0 and Data 1. The four hybrid combinations are compared with regard to detector/descriptor type, where the $FAST$ technique is used

Fig. 9: First Hybrid Combination

Fig. 10: Second Hybrid Combination

Fig. 11: Third Hybrid Combination

Fig. 12: Fourth Hybrid Combination

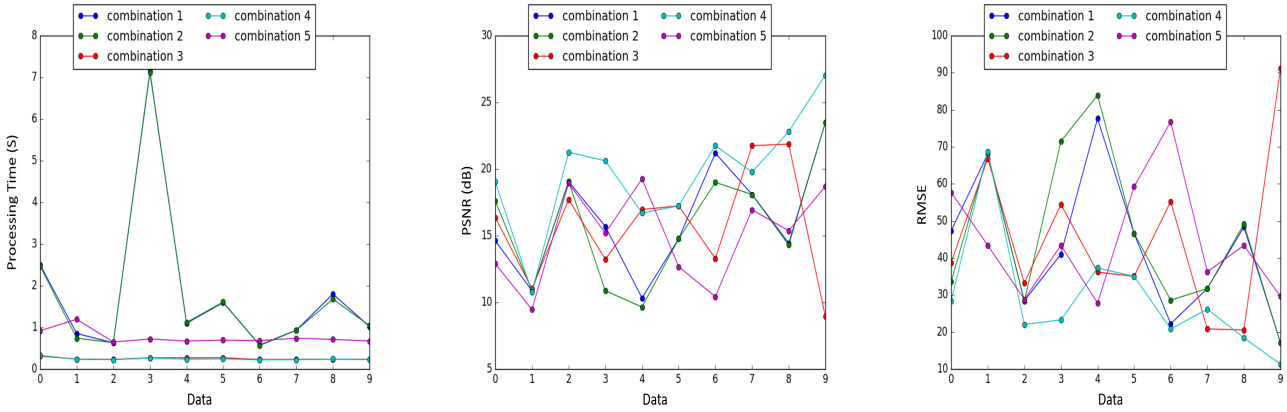
Fig. 13: First Hybrid Combination

Fig. 14: Second Hybrid Combination

Fig. 15: Third Hybrid Combination

Fig. 16: Fourth Hybrid Combination

Figures 9, 10, 11 and 12 show the resulted four stitched images using Data 0, while Figures 13, 14, 15 and 16 show the resulted four stitched images using Data 1



(a) Processing Time (b) $PSNR$ (c) $RMSE$

Fig. 17: Comparison of the five combinations in terms of: (a) Processing Time, (b) $PSNR$ and (c) $RMSE$ for 10 different data.

TABLE I: The Four Hybrid Combination Techniques Results of Data 0

	Data 0							
	Detector/ Descriptor	**Matching**	**Remove Outliers points**	**Alignment**	**Number of keypoints from first image**	**Number of keypoints from second image**	**Number of matched keypoints**	**Result**
Method 1	$FAST$ \ $BRISK$	KNN	$RANSAC$	Perspective Transform	7624	7731	31	Successful
Method 2	$FAST$ \ $BRISK$	KNN	$PROSAC$	Perspective Transform	7624	7731	31	Successful
Method 3	ORB \ ORB	KNN	$RANSAC$	Perspective Transform	500	500	31	Successful
Method 4	ORB \ ORB	KNN	$PROSAC$	Perspective Transform	500	500	31	Successful

TABLE II: The Four Hybrid Combination Techniques Results of Data 1

	Data 1							
	Detector/ Descriptor	**Matching**	**Remove Outliers points**	**Alignment**	**Number of keypoints from first image**	**Number of keypoints from second image**	**Number of matched keypoints**	**Result**
Method 1	$FAST$ \ $BRISK$	KNN	$RANSAC$	Perspective Transform	2482	1901	31	Successful
Method 2	$FAST$ \ $BRISK$	KNN	$RANSAC$	Perspective Transform	2482	1901	31	Successful
Method 3	ORB \ ORB	KNN	$RANSAC$	Perspective Transform	500	500	31	Successful
Method 4	ORB \ ORB	KNN	$PROSAC$	Perspective Transform	500	500	31	Successful

TABLE III: Comparison of the Four Hybrid Combination Techniques for 10 Data (1)

	Combination 1					**Combination 2**				
	Processing Time (s)	**Contrast Differences**		**PSNR (dB)**	**RMSE**	**Processing Time (s)**	**Contrast Differences**		**PSNR (dB)**	**RMSE**
		$\mathbf{d_A}$	$\mathbf{d_B}$				$\mathbf{d_A}$	$\mathbf{d_B}$		
Data 0	2.50309	365268	267576	14.631	47.312	2.46984	73218	112068	17.609	33.581
Data 1	0.85385	13023558	8068338	10.872	68.077	0.73972	13064142	8038308	10.872	68.077
Data 2	0.62836	85230	101964	19.053	28.437	0.63583	83136	91626	19.088	28.322
Data 3	7.14782	75186	87852	15.699	41.018	7.11124	1185936	1181022	10.875	71.48
Data 4	1.09822	1451064	1209600	10.332	77.618	1.11622	104508	115926	9.652	83.936
Data 5	1.59695	75132	72858	14.766	46.584	1.60874	99432	71598	14.766	46.586
Data 6	0.57794	85758	85872	21.194	22.224	0.56634	109380	102126	19.0	28.611
Data 7	0.93311	78144	93948	18.096	31.749	0.93277	80076	71814	18.096	31.749
Data 8	1.80077	122898	103104	14.439	48.371	1.67754	158718	122154	14.3	49.151
Data 9	1.02130	84066	84612	23.493	17.057	1.04310	84066	84516	23.492	17.058

TABLE IV: Comparison of the Four Hybrid Combination Techniques for 10 Data (2)

	Combination 3					**Combination 4**				
	Processing Time (s)	**Contrast Differences**		**PSNR (dB)**	**RMSE**	**Processing Time (s)**	**Contrast Differences**		**PSNR (dB))**	**RMSE**
		$\mathbf{d_A}$	$\mathbf{d_B}$				$\mathbf{d_A}$	$\mathbf{d_B}$		
Data 0	0.30996	80148	103404	16.368	38.739	0.32800	100956	100536	19.051	28.444
Data 1	0.24092	12035304	6983988	11.034	66.818	0.23476	12722154	8241258	10.788	68.739
Data 2	0.23189	124308	134700	17.702	33.222	0.22431	87174	92172	21.261	22.055
Data 3	0.26943	597486	576864	13.243	54.423	0.26545	105282	73854	20.623	23.27
Data 4	0.26986	104718	105714	16.961	36.182	0.23573	88644	98586	16.698	37.292
Data 5	0.27146	75126	85512	17.24	35.037	0.24401	77568	87516	17.243	35.028
Data 6	0.22970	1006734	801954	13.283	55.256	0.22012	83136	88596	21.759	20.825
Data 7	0.23260	65022	53256	21.752	20.843	0.22489	80082	69036	19.768	26.189
Data 8	0.23804	125682	110166	21.871	20.559	0.24186	108966	93636	22.808	18.456
Data 9	0.23218	315816	672126	8.934	91.165	0.23196	87876	89844	27.054	11.32

TABLE V: Comparison of the Four Hybrid Combination Techniques for 10 Data (3)

	Combination 5				
	Processing Time (s)	**Contrast Differences**		**PSNR (dB))**	**RMSE**
		$\mathbf{d_A}$	$\mathbf{d_B}$		
Data 0	0.91932	1203294	892212	12.906	57.705
Data 1	1.19511	49792386	35975232	9.475	43.431
Data 2	0.64943	944424	577704	18.934	28.829
Data 3	0.72190	1975944	1109010	15.203	43.431
Data 4	0.67391	552252	876252	19.258	27.774
Data 5	0.69564	1512900	1189476	12.681	59.225
Data 6	0.68438	1557420	729276	10.43	76.742
Data 7	0.73849	958362	1074060	16.943	36.258
Data 8	0.71170	2065578	1281894	15.384	43.385
Data 9	0.67448	2359956	832956	18.706	29.597

in first and second combination and the ORB technique is used in the third and fourth combinations. It can be seen that the FAST technique extracted more keypoints compared to the ORB technique. As minimum number of keypoints will reduce the processing time.

V. Conclusion and Future Works

Researchers have worked to improve panorama stitching techniques and minimize the computational requirements. This paper focused on a new application of panorama stitching for a 2D scenario. The proposed methods can generate a semi-transparent view as a solution for the project enabling drivers to effectively see around corners. This paper also compared the quality and processing time of the produced 2D views within the scope of state-of-the-art methods. For future work, the presented models can be extended to include 3D scenes. Another future work can provides a real-time processing for similar applications to assist drivers and pedestrians. Finally, additional novel techniques could be implemented to enhance the methods discussed in this work.

Acknowledgment

We would like to express our special thanks to King Faisal University and to Dr. Syed Afaq Husain, Dr. Muhammad Bilal Ahmad and Dr. Asrar Ul Haque for their feedback which helped to improve the project.

References

[1] Haque, M.J., "Improved Automatic Panoramic Image Stitching," *Lap Lambert Academic Publishing GmbH KG*, 2012

[2] H. Jeon and J. Jeong and K. Lee, "An implementation of the real-time panoramic image stitching using ORB and $PROSAC$," *International SoC Design Conference (ISOCC)*, 2015, pp. 91–92.

[3] M. Wang and S. Niu and X. Yang, "A novel panoramic image stitching algorithm based on ORB," *International Conference on Applied System Innovation (ICASI)*, 2017, pp. 818–821.

[4] A. Agarwala1 and M. Agrawala and M. Cohen and D. Salesin1 and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," *ACM $TRANSACtions$ on Graphics*, 2006, vol. 25, pp. 853–861.

[5] R. Gupta and R. I. Hartley, "Linear pushbroom cameras," *IEEE $TRANSACtions$ on Pattern Analysis and Machine Intelligence*, 1997, vol. 19, pp. 963–975.

[6] S. Peleg and B. Rousso and A. Rav-Acha and A. Zomet, "Mosaicing on adaptive manifolds," *IEEE $TRANSACtions$ on Pattern Analysis and Machine Intelligence*, 2000, vol. 22, pp. 1144–1154.

[7] A. Zomet and D. Feldman and S. Peleg and D. Weinshall, "Mosaicing new views: the Crossed-Slits projection," *IEEE $TRANSACtions$ on Pattern Analysis and Machine Intelligence*, 2003, vol. 25, pp. 741–754.

[8] M. Agrawala and D. Zorin and T. Munzner, "Artistic Multiprojection Rendering," *Proceedings of the Eurographics Workshop on Rendering Techniques*, 2000, pp. 125–136.

[9] J. Yu and L. Mcmillan, "A Framework for Multiperspective Rendering," *Proceedings of the 15th Eurographics Conference on Rendering Techniques*, 2004, pp. 61–68.

[10] J. Yu and L. Mcmillan, "General Linear Cameras," *Proceedings of the 8th European Conference on Computer Vision*, 2004, pp. 14–27.

[11] M. Z. Bonny and M. S. Uddin, "Feature-based image stitching algorithms," *International Workshop on Computational Intelligence (IWCI)*, 2016, pp. 198–203.

[12] R. Szeliski and H. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps," *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques [ACM Press]*, 1997, pp. 251–258.

[13] A. Wójcicka and Z. Wróbel, "The Panoramic Visualization of Metallic Materials in Macro- and Microstructure of Surface Analysis Using Microsoft Image Composite Editor (ICE)," *Proceedings of the Third International Conference on Information Technologies in Biomedicine*, 2012, pp. 358–368.

[14] P. Azad and T. Asfour and R. Dillmann, "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition," *International Conference on Intelligent Robots and Systems*, 2009, pp. 4275–4280.

[15] J. Jiao and B. Zhao and S. Wu, "A speed-up and robust image registration algorithm based on FAST," *IEEE International Conference on Computer Science and Automation Engineering*, 2011, vol. 4, pp. 160–164.

[16] L. Yu and Z. Yu and Y. Gong, "An Improved ORB Algorithm of Extracting and Matching Features," *International Journal of Signal Processing and Pattern Recognition*, 2015, vol. 8, pp. 117–126.

[17] E. Rosten and and T. Drummond, "Machine Learning for High-speed Corner Detection," *Proceedings of the 9th European Conference on Computer Vision - Volume Part I [Springer-Verlag]*, 2006, pp. 430–443.

[18] J.J. Anitha and S.M.Deepa, "Tracking and Recognition of Objects using SURF Descriptor and Harris Corner Detection," *International Journal of Current Engineering and Technology*, 2014, vol. 4, pp. 775–778.

[19] K. Dohi and Y. Yorita and Y. Shibata and K. Oguri, "Pattern Compression of FAST Corner Detection for Efficient Hardware Implementation," *21st International Conference on Field Programmable Logic and Applications*, 2011, pp. 478–481.

[20] E. Rublee and V. Rabaud and K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *International Conference on Computer Vision*, 2011, pp. 2564–2571.

[21] M. Brown and D.G. Lowe, "Automatic Panoramic Image Stitching using Invariant Features," *International Journal of Computer Vision*, 2007, vol. 74, pp. 59–73.

[22] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[23] S. Leutenegger and M. Chli and R. Y. Siegwart, "$BRISK$: Binary Robust invariant scalable keypoints," *International Conference on Computer Vision*, 2011, pp. 2548–2555.

[24] A. S. Arefin and C. Riveros and R. Berretta and P. Moscato, "GPU-FS-KNN: A Software Tool for Fast and Scalable KNN Computation Using GPUs," *PloS one*, 2012, vol. 7, pp. e44000.

[25] J. S. Beis and D. G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1000–1006.

[26] O. Chum and J. Matas, "Matching with $PROSAC$ - progressive sample consensus," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 220–226.

[27] P. Ostiak, "Implementation of HDR panorama stitching algorithm," *10th Central European Seminar on Computer Graphics for Students (CESCG)*, 2006.

[28] C.Y. Chen, "Image Stitching - Comparisons and New Techniques," *CITR-TR-30*, 1998.

[29] P. Ndajah and H. Kikuchi and M. Yukawa and H. Watanabe and S. Muramatsu, "An investigation on the quality of denoised images," *International Journal of Circuits, Systems and Signal Processing*, 2011, vol. 5, pp. 423–434.