360 Stitching from Dual-Fisheye Cameras Based on Feature Cluster Matching

Tancredo Souza, Rafael Roberto, Veronica Teichrieb [*]
[*] Voxar Labs - Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{tantan, rar3, vt}@cin.ufpe.br

João Paulo Silva do Monte Lima [†*]
[†] Departamento de Computação
Universidade Federal Rural de Pernambuco
Recife, Brazil
joao.mlima@ufrpe.br

Jonysberg Peixoto Quintino [‡]
[‡] Projeto de P&D CIn/Samsung
Universidade Federal de Pernambuco
Recife, Brazil
jpq@cin.ufpe.br

Fabio Q. B. da Silva, Andre L M Santos [§]
[§] Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{fabio, alms}@cin.ufpe.br

Helder Pinho [¶]
[¶] Samsung Instituto de
Desenvolvimento para a Informática
Campinas, Brazil
helder.p@sidi.org.br

***Abstract*—In the past years, captures made by dual-fisheye lens cameras have been used for virtual reality, 360 broadcasting and many other applications. For these scenarios, to provide a good-quality experience, the alignment of the boundaries between the two images to be stitched must be done properly. However, due to the peculiar design of dual-fisheye cameras and the high variance between different captured scenes, the stitching process can be very challenging. In this work, we present a 360 stitching solution based on feature cluster matching. It is an adaptive stitching technique based on the extraction of feature cluster templates from the stitching region. It is proposed an alignment based on the template matching of these clusters, successfully reducing the discontinuities in the full-view panorama. We evaluate our method on a dataset built from captures made with an existing camera of this kind, the Samsung's Gear 360. It is also described how we can extend these concepts from image stitching to video stitching using the temporal information of the media. Finally, we show that our matching method outperforms a state-of-the-art matching technique for image and video stitching.**

I. Introduction

A full-view panorama (or 360 panorama) creates an immersive experience for the users, especially in virtual reality applications. It is possible to use these full-view panoramas for educational purposes, among others. As an example, the Google Expedition[1] project aims to use these panoramas as a virtual teaching tool, enabling teachers to choose environments for children to visit in virtual reality. In this case, to create such environments, it is used a polydioptric camera. It consists of multiple cameras, each one pointing to a certain direction and having an overlapping field of view. The full-view panoramas obtained with this kind of camera have a great quality, *i.e.* any transition of point of view is almost imperceptible, which is essential to create an immersive experience.

However, while virtual reality devices are more accessible nowadays, such as the Google's Cardboard and Samsung's Gear VR, this does not apply to polydioptric cameras. Some setups may use more than 15 high-resolution cameras combined, which can be very expensive. Also, they can be very hard to handle, given their size and weight. Google Street View Trekker, for instance, weighs 20 kg. and is 1.2 meters tall.

[1]https://edu.google.com/expeditions/

In contrast, dual-fisheye lens cameras emerged as a means to popularize 360° captures. Dealing with size and weight issues, these smaller cameras have a more accessible price and can be held by one hand. Their design consists of two fisheye lenses pointing at opposite directions (Fig. 1). Each lens has a field-of-view greater than 180°, creating a region in both images of duplicate information. This area, called *overlap*, allows these images to be stitched.

Fig. 1. Samsung Gear 360 C200 (left) and R210 (right) dual-fisheye cameras.

On the other hand, the stitching process for this kind of camera is more challenging. The overlap region is much smaller than the ones from the polydioptric cameras. This means that there is less information to be extracted from these regions, making the alignment estimation of these images more difficult. We propose an adaptive stitching technique, being more robust to absence of information and to distortions caused by the image spherical representation.

The main contributions of this paper are:

1) An adaptive stitching method for dual-fisheye lens cameras that considers the image regions with more texture (Section III);

2) An extension of this method to stitch videos using temporal aspects (Section IV);
3) A qualitative evaluation of this method for both image and video stitching using different cameras. (Section V).

II. Related Work

The problem of stitching images dates before the popularization of virtual reality devices and 360° cameras. The goal was to automatically create panoramic images. One of the first works combined images by choosing the best corresponding lines in the two images, which is locally smoothed afterwards [1]. More recent solutions tackled this problem as a multi-image matching problem, using invariant local features to find matches between all of the images [2], which created a foundation for many further studies. Then, some works were proposed to improve this idea, such as addressing pure rotation [3], misalignment of the input images [4] and preserving image perspective [5].

Dealing with dual-fisheye images is more challenging due to the limited overlap region. However, some methods tried to address this problem before dual-fisheye cameras were widely available. For instance, Deng et al. [6] proposed using the focal length and line perspective on the spherical image to estimate the rotation and intrinsic parameters of two fisheye images. These two input pictures were captured using a fisheye lens mounted on a single special camera and pointing to opposite directions at different moments. A similar idea was used for mobile devices [7], where two fisheye lens were mounted on the front and rear camera simultaneously. The stitching method was optimized to reduce the operating area in order to improve the execution time on smartphones.

The recent release of cameras with two fisheye lenses made it easier to capture 360° images. Following this popularization, some studies were published aiming to stitch the output pictures of such cameras. Ni *et al.* [8] proposed a method that adapted the unwarping and blending procedures to address the characteristics of the fisheye lens. Ho *et al.* created a method that uses an initial static alignment combined with a post-refinement step based on a template matching of the overlap regions to stitch the images. They lately evolved this work to remove jitter from videos using the temporal coherence of the stitching boundary [9].

The main difference in our method is that we combine invariant local features with template matching, which is more suitable for images with small overlap region such as in dual-fisheye lenses. In summary, we extract features in the overlap region in order to find clusters that will be used to perform template matching. With this modification, it was not necessary to have preliminary alignments. Additionally, when stitching videos, we use point interpolation to avoid abrupt transition between different homographies.

III. Dual-Fisheye Image Stitching

The pipeline of the developed stitching method is shown in Fig. 2. Initially, the original dual-fisheye image is unwarped into two equirectangular projections (one for each lens). Afterwards, by detecting and matching their features, it is possible to analyze objects with relevant texture details. These features are then clusterized to create rectangular templates, which can be matched to estimate an homography that successfully aligns both images. Finally, it is applied a blending function to provide a better visualization of the stitched result. We now proceed to the details of each step.

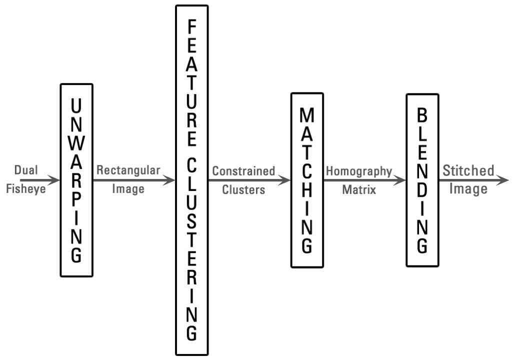

Fig. 2. The proposed stitching method pipeline for dual-fisheye images.

A. Unwrapping

The design of dual-fisheye cameras capable of capturing 360-degree panoramas is different from conventional digital cameras. First of all, it is necessary to understand how the camera captures the world around it. The image acquired by the dual-fisheye camera has the format shown in Fig. 3.

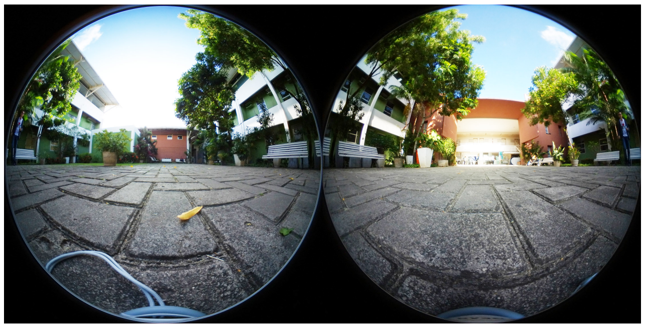

Fig. 3. The world seen by both fisheye lenses.

Given this format, some adaptations are required in order to work with this image in planar coordinates. We apply a warping function that comprises some changes of the coordinate system [10] to obtain a rectangular representation of the spherical image (Fig. 4).

The first step consists in normalizing the equirectangular coordinates to the $[-1, 1]$ range using the following equations:

$$x' = \frac{2x}{w} - 1,\ y' = 1 - \frac{2y}{h}, \tag{1}$$

where w and h are width and height of the equirectangular image, respectively, and (x, y) the point position. After this step,

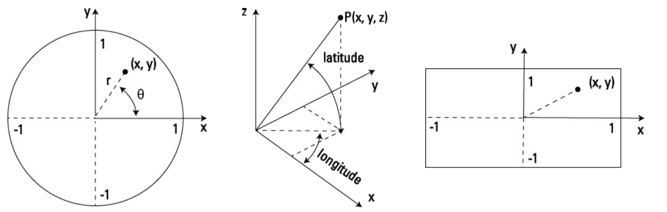

Fig. 4. Process to convert from fisheye coordinate (left) to 3D vector (center) and them to equirectangular image (right). Image adapted from Bourke et al. [10].

the 2D normalized equirectangular coordinates are represented as $longitude$ and $latitude$:

$$longitude = \pi x' + \lambda_0,\ latitude = \frac{\pi}{2} y', \tag{2}$$

where λ_0 is the central meridian of the equirectangular projection. Considering the rotation between the fisheye lenses as 180°, then $\lambda_0 = -\frac{\pi}{2}$ for one of the lens and $\lambda_0 = \frac{\pi}{2}$ for the the other.

The normalized 2D fisheye coordinates are obtained from the 3D spherical coordinates:

$$r = \frac{2 \arctan \frac{\sqrt{P_x^2 + P_z^2}}{P_y}}{aperture},\ \theta = \arctan \frac{P_z}{P_x}, \tag{3}$$

where

$$\begin{gathered} P_x = cos(latitude)cos(longitude), \\ P_y = cos(latitude)sin(longitude), \\ P_z = sin(latitude). \end{gathered} \tag{4}$$

Finally, to denormalize the 2D fisheye coordinates:

$$x'' = \frac{w(1-x)}{2},\ y'' = \frac{h(1-y)}{2}. \tag{5}$$

We obtain as a result two different projections of same dimensions: the center (E_c) and border (E_b) equirectangular projections, which are used to generate the full 360° panorama. For convenience, we define the width and height of these projections as, respectively, W and H. Since the field of view for fisheye cameras is greater than 180°, the equirectangular projections will have a left and right region with redundant information. These areas are defined as the left ($\cap_l$) and right ($\cap_r$) overlap regions (Fig. 5).

B. Feature Clustering

Given E_b and E_c, we initially apply the ORB feature extractor proposed by Rublee et al. [11] on their $\cap_l$ and $\cap_r$ regions. Afterwards, it is proposed a method to cluster these features into templates, based on their proximity. These created templates can be matched and used to estimate an homography matrix that aligns both images. We only consider these high texture areas instead of using $\cap_l$ and $\cap_r$ as a single template to be matched [12]. Doing this makes the homography more related to the scene-position of an object seen through different lenses, rather than only considering the similarity of the overlapped information. This avoids closer objects suffering from the parallax effect to worsen the homography estimation.

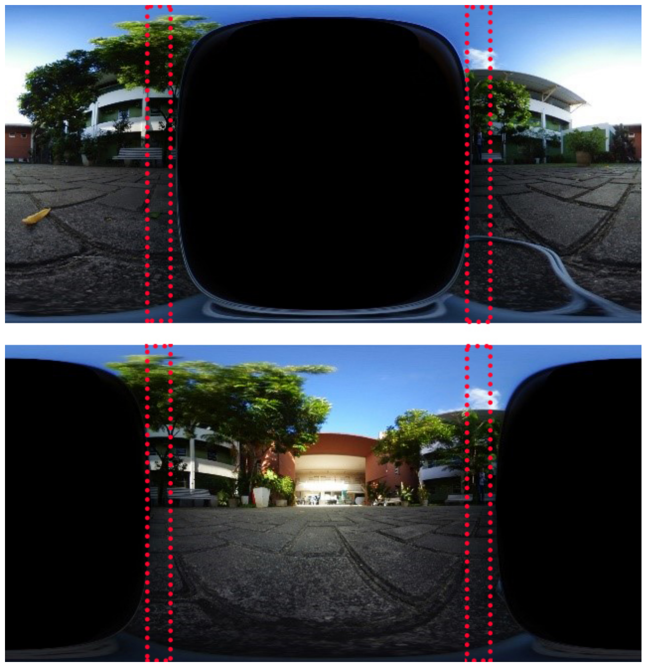
Fig. 5. Border (top) and center (bottom) equirectangular projections that compose the full 360° panorama and their respective overlap regions.

1) Feature Detection and Matching: This step detects and matches features related to the texture of an object, as shown in Fig. 6. These features can be interpreted as important points for the analysis of the misalignment between the two fisheye lenses. For such task, we performed a feature correspondence step with brute-force search for nearest-neighbors based on binary descriptors. It was used hamming distance to measure the disparity between descriptors. Finally, to remove spurious correspondences, it was considered the distance ratio between the two nearest neighbors.

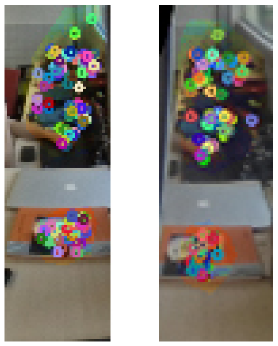
Fig. 6. Some extracted features from the overlap regions.

2) Feature Evaluation: After a set of features L is detected, we perform a feature evaluation. This step is important to guarantee which features should be used to obtain a consistent alignment. For a feature positioned at $l(x, y)$ and its correspondent $l'(x', y')$, two conditions must be met:

(i) $|l - l'| \leq \gamma$,
(ii) $(l')' = l$,

where $\gamma = 0.05 \times W$ is the maximum position displacement between the correspondent features. A feature is defined as *good*, if both conditions were met, *bad*, if both criteria fail, and *average*, otherwise.

3) Clustering: For the template matching step, regions must be created out of the set of features L. In this step, the features are grouped into clusters C_i (Fig. 7) based on the distance δ to their neighbors:

$$C_i = \{l \in L \mid \forall l_i \in C_i, |l - l_i| \leq \delta\}, \tag{6}$$

where C_i is the subset of features with the highest number of elements and $|C_i| \geq 4$. In our implementation, we defined $\delta = 0.01 \times W$. Based on the evaluation performed in Subsection III-B2, clusters are formed with features that have a *good* quality.

We now proceed to the definition of the templates. Given a cluster and its set of features, let $l_1(x_l, y_l)$ and $l_2(x_r, l_r)$ be the farthest features to the left and right, respectively. Finally, let $l_3(x_t, y_t)$ and $l_4(x_b, y_b)$ be the farthest upwards and downwards. Using these points, we can define a rectangular template T_i as described in Fig. 7.

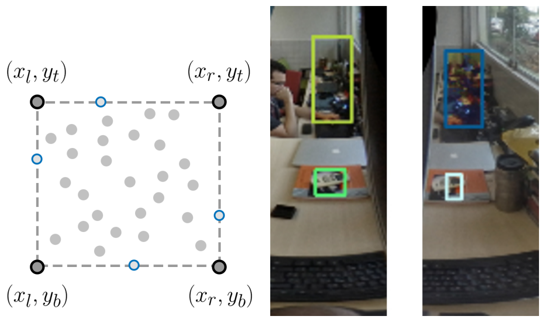

Fig. 7. Feature clusters turned into rectangular templates. The farthest features are represented as blue circles.

4) Template Expansion: In our experiments, we noticed that some generated templates were too small, making it infeasible to extract any information from that region. To avoid this issue, we apply a template expansion step that guarantees a minimum height and width for each cluster. Let h and w be the initial template height and width and $\cap_h$ and $\cap_w$ be the height and width of the overlap region, respectively. We define a height and width threshold, $t_h = 0.50 \times \cap_w$ and $t_w = \cap_w$, such that for a template T_i generated from a cluster C_i, $h \geq t_h$ and $w \geq t_w$. These templates are expanded from their center such that both conditions are satisfied.

$$T_i(h', w') = \begin{cases} h' = max(h, t_h), \\ w' = max(w, t_w). \end{cases} \tag{7}$$

It is important to still respect the limits of $\cap_l$ and $\cap_r$, otherwise the template will then include information outside of these areas (Fig. 8).

5) Template Balance: It is possible that one side of the overlap region has much more templates than its opposite. If this happens, the transformation may be biased in favor of the region with most templates. This, can result in a good alignment for that side while the other will be totally misaligned. To avoid this, we perform a balancing step, forming templates from clusters with less quality features to the short-numbered side until both have the same template amount.

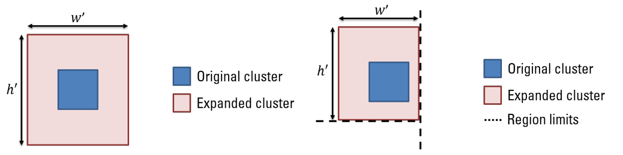

Fig. 8. The cluster expansion while also respecting the region limits.

C. Matching

In this work, we aim to minimize the discontinuity when transitioning from E_b and E_c using template matching. This matching can be summarized as moving a template T such that it perfectly aligns to its correspondent T'. Let T_b and T_c be, respectively, the sets of templates obtained from $\cap_{l,r}$ in E_b and E_c. Each template in $T \in T_b \cup T_c$ will be matched with a region in which the normalized cross-correlation value ρ is maximum [13]. This will cause a displacement in the template related to its original position. After this, there will be a set of templates T'_b matched in E_c, and, by analogy, a set T'_c matched in E_b. Let $P_c = T_c \cup T'_b$ and $P_b = T_b \cup T'_c$. These sets are built such that, for all $c_i \in P_c$ and $b_i \in P_b$,

$$c_i \longleftrightarrow b_i \tag{8}$$

where $\longleftrightarrow$ denotes a point correspondence.

At this moment, we estimate an homography H which relates E_c and E_b, based on the displacement information for all templates in P_c and P_b. For such estimation, we aim to minimize the back-projection error using RANSAC to eliminate outliers. As a result, we obtain a perspective transform that distorts E_c, which can then be overlayed to E_b with less apparent discontinuity.

Finally, it is important to assure the homography does not degenerate the final panorama. Indeed, even though RANSAC is a good filter for outliers, the matching imperfection may lead to a degenerated stitched panorama due to error propagation in correspondences. To avoid this issue, we validate the estimated homography, checking if all four farthest corners of E_c are still within a displacement limit $r = 0.15 \times H$ (Fig. 9) and, thus, did not degenerate. If that is not the case, the estimated homography is discarded and the identity matrix will be applied instead.

D. Blending

We apply a blending method to better visualize the stitching results. In this step, to create two blended overlap regions $B_{l,r}$ from $\cap_{l,r}$, we apply the ramp function proposed by Ho et al. [12], which is defined as:

$$\begin{aligned} B_l(x,y) &= \alpha(\cap_l, x) * E_b(x,y) + \beta(\cap_l, x) * E_c(x,y), \\ B_r(x,y) &= \alpha(\cap_r, x) * E_b(x,y) + \beta(\cap_r, x) * E_c(x,y), \end{aligned} \tag{9}$$

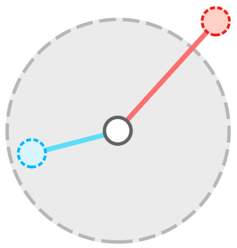

Fig. 9. The original corner position in white, a valid displacement in blue and invalid in red.

where (x, y) represents the pixel position from $\cap_{l,r}$, α and β are defined as

$$\alpha(\cap, x) = \begin{cases} \frac{\cap_w - x + 1}{\cap_w}, \; for \; \cap = \cap_l \\ \frac{x}{\cap_w}, \; for \; \cap = \cap_r \end{cases}, \tag{10}$$

$$\beta(\cap, x) = \begin{cases} \frac{x}{\cap_w}, \; for \; \cap = \cap_l \\ \frac{\cap_w - x + 1}{\cap_w}, \; for \; \cap = \cap_r \end{cases}.$$

This function gradually changes the pixels in the overlap region, as illustrated in Fig. 10.

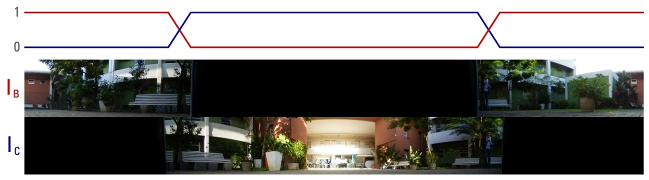

Fig. 10. The ramp function applied for blending.

We summarize the proposed method for image stitching in Algorithm 1.

Algorithm 1 Dual-Fisheye Image Stitching

```
1:  variables
2:      $F$, Dual-fisheye frame to be stitched
3:      $S$, Stitched frame
4:  end variables
5:  procedure IMAGESTITCH
6:      $E_b, E_c \leftarrow equirectangularProjection(F)$
7:      $T_b, T_c \leftarrow buildTemplates(E_b, E_c)$
8:      $T'_b, T'_c \leftarrow templateMatching(T_b, T_c)$
9:      $H \leftarrow findHomography(T_c \cup T'_b \rightarrow T'_c \cup T_b)$
10:     $validateHomography(H)$
11:     $E'_c \leftarrow warpPerspective(E_c, H)$
12:     $S \leftarrow blend(E_b, E'_c)$
```

IV. Dual-Fisheye Video Stitching

The pipeline for video stitching is shown in Fig 11. A video is composed of individual frames as in Fig. 3. Thus, the approach proposed in Section III can also estimate, for each video frame f, an alignment matrix H_f. However, some challenges arise due to the temporal information of a video, such as dealing with the jittering effect [9]. Applying a different H_f every frame would make the video uncomfortable to watch, since each minor displacement of E_c will be perceptible. We describe a scoring system used to obtain a refined affine matrix H_m as the video progresses, instead of constantly changing it. Finally, it is used point interpolation to gradually change to H_m, reducing the jittering effect.

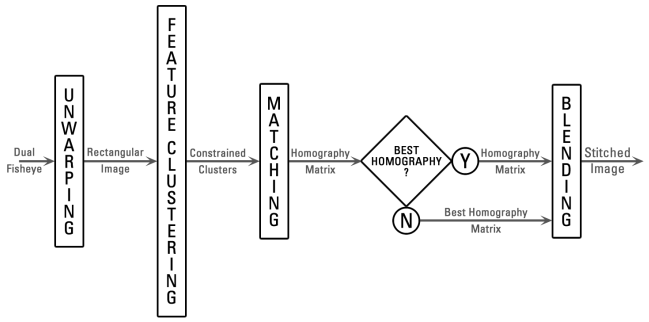

Fig. 11. The proposed stitching method pipeline for dual-fisheye videos.

A. Homography Refinement

In this step, it is performed a weighted-average of all templates in the frame based on the number of inliers considered by RANSAC when estimating the affine matrix. The frame score s_f is based on the number of inliers w_i that a given template T_i has when estimating H_i:

$$s_f = \frac{\sum_{i=1}^{|T|} w_i * \rho_i}{\sum_{i=1}^{|T|} w_i}, \tag{11}$$

where ρ_i is the normalized cross-correlation value for the template matching of the template T_i. If the homography is valid and the score increases, this indicates a change in the homography, which will be gradually applied, avoiding the jitter effect.

B. Interpolation Method

Let H_f and H_{f+1} be the homographies estimated for the frames f and $f+1$. As previously discussed, changing immediately from H_f to H_{f+1} results in an unpleasant effect to the viewer. In this step, we aim to reduce this effect using point interpolation (Fig. 12). We create a set of intermediary points p_i using the following equation:

$$p_i = (1 - \Delta) * p_f + \Delta * p_{f+1}, \text{ for } \Delta = [0, 0.02, ...1] \tag{12}$$

where $p_f = w_f * H_f^{-1}$ and $p_{f+1} = w_{f+1} * H_f^{-1}$ and w_f and w_{f+1} are the inliers for the frames f and $f+1$.

We summarize the proposed method for video stitching in Algorithm 2.

V. Results

The proposed stitching methods were evaluated on a diverse picture and video dataset, which contained different scenarios (indoor/outdoor captures, varying illumination and objects

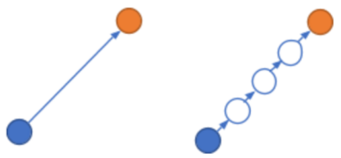

Fig. 12. Points before interpolation (left). Points after interpolation (right).

Algorithm 2 Dual-Fisheye Video Stitching

```
1:  variables
2:      $f$, an individual frame of the dual-fisheye video
3:      $s_m$, maximum current score
4:      $H_i$, homography to be applied to the frame
5:      $\Delta$, interpolation step
6:  end variables
7:  procedure VIDEOSTITCH
8:      $H_i \leftarrow I$
9:      while frame $f$ not empty do
10:         $E_b, E_c \leftarrow \textit{equirectangularProjection}(f)$
11:         $T_b, T_c \leftarrow \textit{buildTemplates}(E_b, E_c)$
12:         $T'_b, T'_c \leftarrow \textit{templateMatching}(T_b, T_c)$
13:         $H_f \leftarrow \textit{findHomography}(T_c \cup T'_b \rightarrow T'_c \cup T_b)$
14:         $s_f \leftarrow \textit{estimateScore}(f)$
15:         if $s_f > s_m$ and valid $H_f$ then
16:             $H_m \leftarrow H_f$
17:             $s_m \leftarrow \rho_f$
18:             $\Delta \leftarrow 0.0$
19:         if $\Delta < 1.0$ then
20:             $H_i \leftarrow \textit{interpolatePoints}(\Delta, H_m)$
21:         $\Delta \leftarrow min(1.00, \Delta + 0.02)$
22:         $E'_c \leftarrow \textit{warpPerspective}(E_c, H_i)$
23:         $\textit{blend}(E_b, E'_c)$
24:         $f \leftarrow \textit{getNextFrame()}$
```

closer/farther away from the camera in the same scene). It consisted of 107 pictures and 6 videos registered with the Samsung Gear 360 C200 and 31 pictures and 4 videos captured with the Samsung Gear 360 R210 (Fig. 1). Table I shows the picture and video resolutions for both models.

TABLE I
MAXIMUM RESOLUTION FOR THE DIFFERENT GEAR 360 MODELS.

Camera model	Image	Video
C200	7776 x 3888	3840 x 2160
R210	5792 x 2896	4096 x 2048

The resulting stitched panoramas are shown in Fig 15. We compared our results with Ho *et al.* [12], which presents state of the art results. In most cases, the template matching step resulted in a consistent alignment. The reduction of the ghosting effect caused by the blending function also shows the robustness of this work. Fig. 13 highlights only the overlap area of different images in order to emphasize the stitching quality. We can see that our method is able to perform a good stitching. This can be noted by looking at the lines, which run continuously from the border image to the center one.

Concerning Video Stitching, our method was also able to perform a good stitching in scenes when the camera is both stationary or in movement[2]. Also, the stitching is not harmed when there are changes in the overlap area, such as people walking. In the scenes in which the camera the camera is stationary, it is possible to note more clearly that changing the homography improves the stitching quality in most of the times. Moreover, the interpolation provided a smooth transition when the homography changes, which produces a more comfortable experience.

Since we rely on features available on overlap areas $\cap_{l,r}$, our method is expected to fail when there are few or no features available. Fig. 14 shows some of these cases. Typically, one of the overlap areas has some features while the other does not. This results in cases such as the one on Fig. 14 (top), in which only one of the overlap areas are correctly aligned. It is worth to mention that in some of these cases there are not many features in both overlap areas, which result in incorrect homographies. Our method was able to identify these invalid homographies, which means that there is no large misalignment between the center and border images.

It was also measured the execution time of the proposed technique. The hardware used for these evaluations was a laptop with an Intel Core i7-6820 @ 2.70GHz processor and 16GB RAM. The proposed 360 stitching based on feature cluster matching was implemented in C++ using the OpenCV library. The execution times are shown in Table II.

VI. CONCLUSION

In this work, we proposed a 360 stitching technique based on feature cluster matching for dual-fisheye cameras. In our evaluation dataset, the method was able to deal with existing challenges for this kind of cameras, by creating consistent full-view panoramas. Initially, it projects the spherical captures into two equirectangular projections. Then, using ORB, it creates templates by detecting and matching relevant features of the objects. These templates are then used to estimate an alignment matrix that creates a full-view panorama with smooth transitions. The method was able to obtain qualitative good results based on regions containing very high texture information. Also, it was shown how to extend the proposed method for video stitching, progressively estimating a better alignment for the video. It was discussed whether the proposed cluster matching technique is more robust than considering the entire overlap regions as single templates.

For future works, it is important to apply an illumination normalization in the images captured from the different lenses. This can improve feature matching in the overlap areas, especially when the lenses are pointing to different places with different lighting conditions. Another improvement is the development of a more robust blending technique, which can reduce most of the ghosting effect that appeared in some situations [14].

[2]A video showing these results is available at https://goo.gl/dzS7Rq.

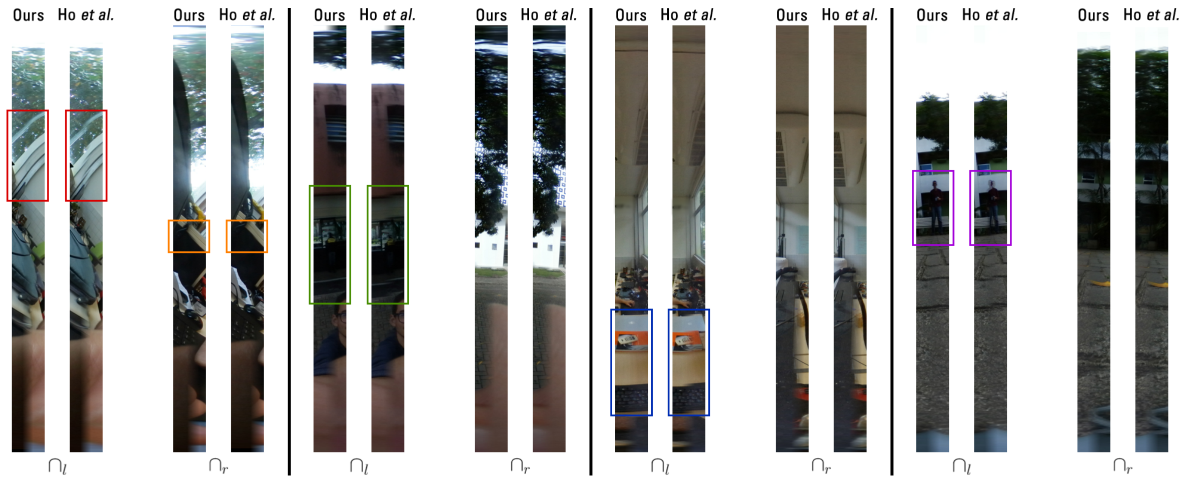

Fig. 13. Comparison between the results of Ho et al. [12] and ours.

TABLE II
AVERAGE EXECUTION TIME PER IMAGE/FRAME IN SECONDS.

	Unwarping	Cluster & Matching	Blending	Total Time
Image	0.640	0.792	0.046	1.4848
Video	0.500	0.732	0.034	1.2663

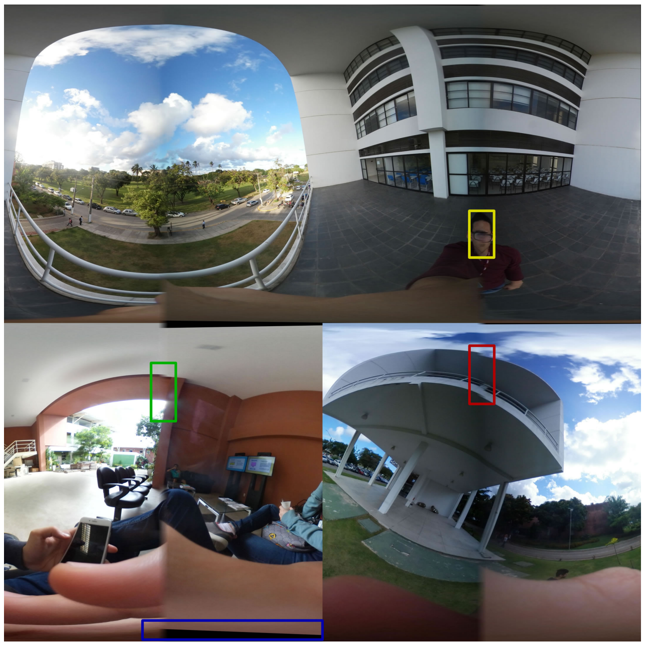

Fig. 14. Failure cases obtained using our method.

REFERENCES

[1] D. L. Milgram, "Computer methods for creating photomosaics," *IEEE Transactions on Computers*, vol. C-24, no. 11, pp. 1113–1119, Nov 1975.

[2] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, Aug 2007. [Online]. Available: https://doi.org/10.1007/s11263-006-0002-3

[3] J. Zaragoza, T. J. Chin, Q. H. Tran, M. S. Brown, and D. Suter, "As-projective-as-possible image stitching with moving dlt," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1285–1298, July 2014.

[4] F. Zhang and F. Liu, "Parallax-tolerant image stitching," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 3262–3269.

[5] C. H. Chang, Y. Sato, and Y. Y. Chuang, "Shape-preserving half-projective warps for image stitching," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, June 2014, pp. 3254–3261.

[6] X. Deng, F. Wu, Y. Wu, and C. Wan, "Automatic spherical panorama generation with two fisheye images," in *2008 7th World Congress on Intelligent Control and Automation*, June 2008, pp. 5955–5959.

[7] T.-M. Liu, C.-C. Ju, Y.-H. Huang, T.-S. Chang, K.-M. Yang, and Y.-T. Lin, "A 360-degree 4kx2k panoramic video processing over smart-phones," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2017, pp. 247–249.

[8] G. Ni, X. Chen, Y. Zhu, and L. He, "Dual-fisheye lens stitching and error correction," in *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Oct 2017, pp. 1–6.

[9] T. Ho, I. D. Schizas, K. R. Rao, and M. Budagavi, "360-degree video stitching for dual-fisheye lens cameras based on rigid moving least squares," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 51–55.

[10] P. Bourke. (2016) Converting dual fisheye images into a spherical (equirectangular) projection. http://paulbourke.net/dome/dualfish2sphere. [Online] Last access 2017-10-23.

[11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[12] T. Ho and M. Budagavi, "Dual-fisheye lens stitching for 360-degree imaging," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2172–2176.

[13] J. P. Lewis, "Fast template matching," in *Vision interface*, vol. 95, no. 120123, 1995, pp. 15–19.

[14] H. Hejazifar and H. Khotanlou, "Fast and robust seam estimation to seamless image stitching," *Signal, Image and Video Processing*, vol. 12, no. 5, pp. 885–893, Jul 2018. [Online]. Available: https://doi.org/10.1007/s11760-017-1231-3

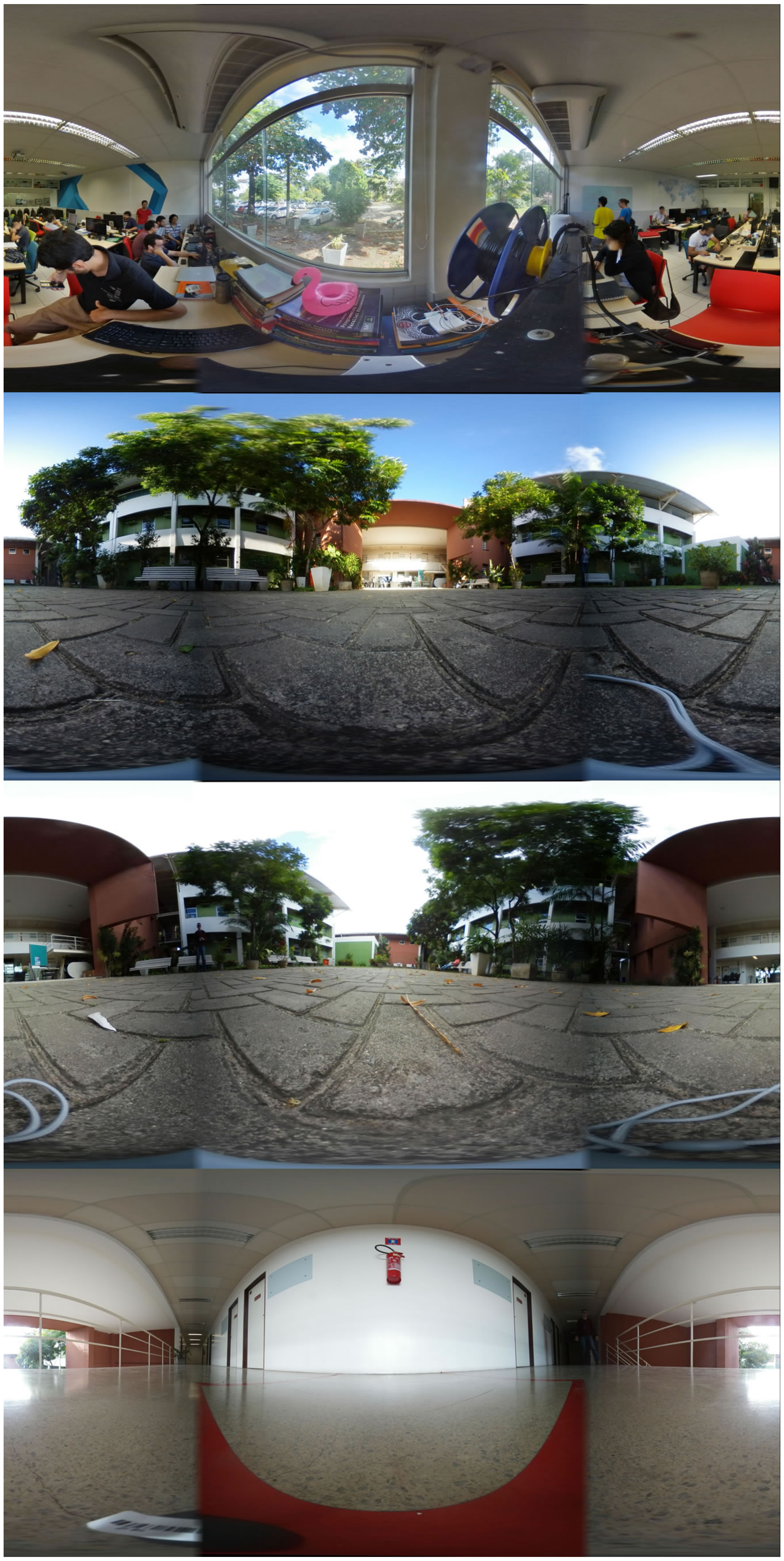
Fig. 15. Results obtained using the proposed method.