I-043

PatchMatch を用いたアニメ画像のマッチングの高速化
PatchMatch for Efficient Matching of Cartoon Images

ガルシア トリゴ パブロ† 金森 由博‡ 楽 詠灝† 今井 星† 西田 友是†
Pablo Garcia Trigo† Yoshihiro Kanamori‡ Yonghao Yue† Sei Imai† Tomoyuki Nishita†

1. Introduction

Hand-drawn 2D animated cartoons are popular due to their expressiveness, but they require high amounts of work to be produced. There exist several methods for automating their production and most are based on *image registration* or *image matching*, a technique to find the correspondences between a *source* image and a *target* image. Image matching algorithms divide the source image into regions called *patches* and for each patch in the source image, search exhaustively for the most similar patch, called *match*, in the target image. Because many patch comparisons are performed, these algorithms tend to be slow, reducing their applications.

In this paper, we propose a method to accelerate the cartoon image matching method proposed by Sykora et al. [3] by the use of a random-based approximation algorithm proposed by Barnes et al. [4] that reduces the number of patch comparisons. Our initial tests show that our approach is successful at speeding up the matching.

2. Related Work

There exist several methods for automating or assisting the production of hand-drawn 2D animated cartoons.

Madeira et al. [1] and Qiu et al. [2] proposed methods that consist in segmenting cartoon images into *regions* (e.g., the face or the hand of a character), computing the features for each region and using these features to find the most similar region. These regions employ regions instead of patches, are focused mainly at coloring cartoons, and do not yield correspondences at the pixel level inside the regions, called *dense correspondences*, which are necessary for techniques such as retexturing and relightening.

To overcome the above limitation, Sykora et al. [3] presented a method based on the deformation of images. This approach assumes that characters in the input images are not occluded and the surface of the regions that make up the character does not vary greatly between frames. To deform the characters, this method divides the input frame into patches and searches exhaustively for the best matches of each, performing many patch comparisons and thus, becoming slow.

Recently, in the image processing field, Barnes et al. [4] proposed *PatchMatch*, a random-based algorithm to calculate the matches of a set of patches in photographs. It reduces the number of patch comparisons by not comparing to all patches but only to a few ones picked up randomly. To our knowledge, this method has not been used yet with cartoons.

In this paper, we propose a method to combine the approaches of Sykora et al. [3] and Barnes et al. [4] to obtain a matching algorithm with dense correspondences that preserves rigidity and is faster than the method of Sykora et al. [1].

3. Previous Approaches

In this section we review the methods by Sykora et al. [3] and Barnes et al. [4] for explaining later how to combine them.

Sykora et al. [3] calculate the matching of two input drawings or bitmap images F_1 (source) and F_2 (target), by deforming iteratively F_1 into F_2 using a grid. Figure 1 shows the two frames and the grids before and after the deformation using the girl character. Through interpolation, with the grids we can obtain the correspondences for all pixels and allow transformations such as recoloring, relightening and retexturing. The slowest part of the method proposed by Sykora et al. [3] is the step called *push*. For each grid point p in frame F_1 the algorithm constructs a patch A in F_1 around p, and uses it as reference for finding the correspondence of p in F_2. In F_2 it sets a search area centered at the same coordinates as p, with a size being several times larger than the grid cell size. Then, for each pixel q in search area S, a patch T centered at q is compared to A and the most similar is chosen as the best patch match, with its center q' is chosen as the match for point p'. It is all these patch comparisons inside the search area that slows down the algorithm. Figure 2 shows this in the left and center pictures. The grid is represented with red lines and the grid point p is highlighted in red. Around it, the patch A is the square drawn in red. The picture in center is the frame F_2 and the search area S is drawn in black.

If we analyze the matching we can see that each patch can fall in one of three cases: C1) The match of the patch is correct. C2) The most similar patch was chosen, but was a wrong one.

Figure 1: Inputs and outputs of our method. © CELSYS, Inc. Left: Input source frame $F_1$and red grid on top of it. Center: Input target frame F_2 . Right: Outputs of our algorithm, frame F_1 deformed to match F_2 with deformed red grid on top.

† 東京大学 The University of Tokyo
‡ 筑波大学 University of Tsukuba

C3) There were equally similar patches and the algorithm just picked one without further information. The most troubling case for the algorithm by Sykora et al. [3] is C2) which can lead to a wrong deformation.

The method by Barnes et al. [4] is an iterative random-based image matching algorithm intended to work over all the pixels of two photographs H_1 and H_2. Initially, each pixel in H_1 is assigned randomly another pixel H_2 as a match.Then, inside a loop, the algorithm tries to improve the match for every pixel i in H_1 in two steps: 1) In the *propagation* step, the pixel i with coordinates H_1 (x, y) receives from its neighbor pixels H_1 (x-1, y) and H_1 (x, y-1) their respective matches $H_2(x', y')$ and $H_2(x'', y'')$. Then, the neighbors $H_2(x'+1, y')$ and $H_2(x'', y''+1)$ of the matches are evaluated as possible match candidates for the pixel i. If one of them is more similar than the current match, the match is updated. 2) In the *random search* step, several pixels are picked up randomly and the current match for pixel i is updated if any of the randomly chosen pixels is a better match. In both steps, the similarity of the patches centered at the pixels being compared is used to determine if it is a better match.

4. Proposed Method

Our proposal consists in using the method by Sykora et al. [3] substituting the push step by a modified version of PatchMatch [4]. In order to be able to combine these two methods, there are two difficulties that need to be overcome: 1) When taking the points in the method by Sykora et al. [3] as the pixels in the method by Barnes et al. [4], they stop being contiguous and, in the propagation step, matches that are too far can fall in the case C2) and mislead the algorithm. To solve this, we decided to skip the propagation step altogether and rely only on the random search step. 2) In the random search step, even if the algorithm picks similar patches, it may never reach a stable configuration, and for areas with many matches equally similar such as those in Figure 2, they could lead to undesirable random local deformations. In this case we give preference to matches close to the center p', as shown in Figure 2, so that in the case of matches equally similar, the points tend to rest in the same place and are later displaced in the *regularize* step of the method proposed by Sykora et al. [3], resulting in smooth deformations. Thanks to these modifications, we covered cases C2) and C3) and were able to combine these two algorithms.

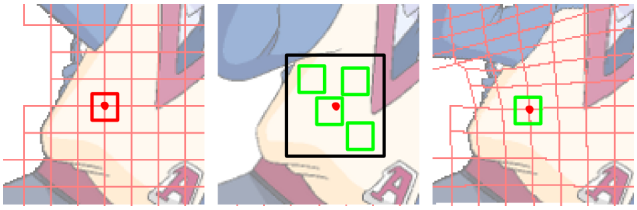

Figure 2: Random matches example © CELSYS, Inc. Left: The grid point and its patch on frame F_1, both in red. Center: Search area in black on frame F_2, centered at the same point. Random matches in green. Many patches are identical in the yellow part of the shirt. Right: Final match. When there is not a match that is more similar than others, our algorithm chooses the closest to the point for stability.

5. Results and Discussion

We have built a prototype of our method running on the CPU in a single core and compared to the method by Sykora et al. [3]. Our testing machine is a laptop equipped with an Intel Core i7 Q740 at 1.73GHz and 4 GB of RAM. In table 1 we show the running time for the cartoon images in Figure 1. We used the root-mean-square error between the deformed F_1 and the target F_2 as a stop criterion. We stopped them when the difference went down to 9%. The size of the frames is 500 x 375.

Table 1: Running time for the cartoon images in Figure 1

	Iterations	Total Time
Sykora	65	344.48*s*
Our Method	97	38.99*s*

In our experiment we have observed that our algorithm is faster, with a speed up of 8.84 times, but our method needs more iterations to converge to the target image. We think this is due to the lack of the propagation step. While there is a risk of introducing wrong matches, we want to evaluate its effect on speed and accuracy. Additionally, Sykora et al. use in their method a algorithm proposed by Li and Salari [5] to skip certain patch comparisons to satisfy an inequality. We plan to introduce it in both implementations and evaluate how the speedup changes.

6. Conclusion and Future Work

We have presented an acceleration method based on PatchMatch [4] for accelerating the cartoon image matching method by Sykora et al. [3] and have seen a speedup in our test. As future work, we would like to repeat the experiments incorporating the propagation step of PatchMatch [4] and the method by Li and Salari [5].

References

[1] J. Madeira, A. Stork, and M. Gross, “An approach to computer-supported cartooning," The Visual Computer,Vol. 12, No. 1, pp. 1-17, (1996).
[2] J. Qiu, H. S. Seah, F. Tian, Q. Chen, Z. Wu, and K. Melikhov, “Auto coloring with enhanced character registration”, International Journal of Computer Games Technology, Vol. 2008, pp. 2:1-2:7, (2008).
[3] D. Sykora, J. Dingliana, and S. Collins, “As-rigid-as-possible image registration for hand-drawn cartoon animations”, in Proceedings of the 7th International Symposium on Non-Photorealistic Animation and Rendering, pp. 25-33, (2009).
[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, “PatchMatch: A randomized correspondence algorithm for structural image editing”, ACM Transactions on Graphics (Proc. SIGGRAPH 2009), Vol. 28, No. 3, pp. 24:1-24:11, (2009).
[5] W. Li and E. Salari, “Successive elimination algorithm for motion estimation”, IEEE Transactions on Image Processing, Vol. 4, No. 1, pp. 105-107, (1995).