Visual Comput (2005) 21: 601–610
DOI 10.1007/s00371-005-0313-3

ORIGINAL ARTICLE

Yun-Tao Jia
Shi-Min Hu
Ralph R. Martin

Video completion using tracking and fragment merging

Published online: 31 August 2005

Y.-T. Jia (✉) · S.-M. Hu
Tsinghua University, Beijing, China
jiayt@cg.cs.tsinghua.edu.cn

R.R. Martin
Cardiff University, Cardiff, UK
ralph@cs.cf.ac.uk

Abstract Video completion is the problem of automatically filling space–time holes in video sequences left by the removal of unwanted objects in a scene. We solve it using texture synthesis, filling a hole inwards using three steps iteratively: we select the most promising target pixel at the edge of the hole, we find the source fragment most similar to the known part of the target's neighborhood, and we merge source and target fragments to complete the target neighborhood, reducing the size of the hole.
Earlier methods were slow, due to searching the whole video data for source fragments or completing holes pixel by pixel; they also produced blurred results due to sampling and smoothing. For speed, we *track* moving objects, allowing us to use a much smaller search space when seeking source fragments; we also complete holes *fragment by fragment* instead of pixelwise. Fine details are maintained by use of a *graph cut algorithm* when merging source and target fragments. Further techniques ensure *temporal consistency* of hole filling over successive frames. Examples demonstrate the effectiveness of our method.

Keywords Video completion · Texture synthesis · Mean shift · Graph cut · Tracking

1 Introduction

Video completion is the problem of automatically filling *holes* (missing parts) in video sequences caused by the removal of unwanted objects. We solve it by using information from other parts of the sequence to suggest suitable in-fill. Video completion has become viable as hardware advances, as evidenced by [3, 15, 21, 23]. It has many applications, in areas such as video editing and film postproduction.

Image completion has been widely studied. *Image inpainting* methods [4, 8, 17] can quickly fill small nontextured holes in time proportional to the size of the hole. *Texture synthesis* methods [5, 11, 13, 20, 22] are more useful for larger, textured holes but generally take time proportional to the size of the image as suitable filling material is sought elsewhere in the image.

Existing methods for *video* completion include *video inpainting*, analogous to image inpainting [3], *space–time video completion*, which is based on texture synthesis and is good but slow [21], *motion layer video completion*, which splits the video sequences into different motion layers and completes each separately [23], and *video repairing*, which repairs static background with motion layers and repairs moving foreground using *model alignment* [15]. Much earlier work does not, however, adequately address important differences between image and video completion: there are much more data, and temporal inconsistencies are visually important in completed video.

We overcome these issues with a new approach based on texture synthesis. It is efficient and produces visually appealing results. It completes each hole iteratively.

Each iteration is divided into three steps. First we select the most promising target pixel at the edge of the hole. A space–time *target fragment* is defined around it; its contents are partially known. Next, we find the source fragment most similar to the known part of the target fragment in a carefully chosen search region of the video. Finally, we merge the source and target fragments to complete the latter, reducing the size of the hole. We use rules to measure each candidate target pixel's *merit* to select the best target. When searching for a suitable source fragment, we *track* moving objects to generate a much smaller relevant search space. We complete holes *fragment by fragment* instead of pixel by pixel to gain further speed. We use a *graph cut* algorithm to merge source and target fragments in a way that retains fine details. Further steps are taken to ensure the *temporal consistency* of completed results over successive frames.

The rest of the paper is organized as follows. We survey prior work in Sect. 2 and outline our new method in Sect. 3. Key ideas are then described in detail in Sects. 4–7: target pixel selection, tracking to quickly find matching source fragments from suitable parts of the video, use of the graph cut method to merge source and target fragments, and enforcement of temporal consistency. Results, a discussion, and conclusions are given in Sects. 8–10.

2 Related work

We now review prior work on *image* and *video completion*, as well as on *mean shift tracking* and *graph cut image merging*, both used as components of our approach.

2.1 Image completion

Video completion basically extends image completion to 3D space–time. We thus consider how existing image completion techniques are relevant to video completion.

There are two main approaches to image completion. *Image inpainting* [4] methods use PDEs to repair minor damage to images. Levin [17] extended this idea by measuring global image statistics and bases inpainting on prior image knowledge as well as local color information. For small, nontextured regions, such methods achieve visually satisfactory results. However, the lack of generated texture in larger more complex reconstructed areas is clearly visible.

Texture synthesis methods comprise the other approach. After selecting a *target* pixel whose neighborhood is partially inside the hole, a *source* fragment, with texture matching the target's *known neighborhood*, is sought elsewhere in the image. This source fragment is then merged into the neighborhood of the target pixel. Such methods are suited to filling *large* holes in images. The method in [13] uses these ideas together with hierarchical image approximation and adaptive neighborhood sizes, leading to impressive results, but at high computational cost. Zhang et al. [22] used a method to preferentially select pixels to be filled, choosing better known neighborhoods having low texturing. A graph cut algorithm is used to find the best way to merge the source fragment with the target fragment; we also do so. This approach completes natural images smoothly and quickly.

2.2 Video completion

Video completion is more challenging for two reasons. Firstly, the amount of data in video sequences is much greater, so texture synthesis methods cannot be directly applied to video completion: searching for a source fragment in the whole video dataset would be much too slow. Secondly, temporal consistency is a necessity; it is *more* important than spatial aliasing in images, due to the eye's sensitivity to motion [21]. Simply completing video sequences frame by frame using image completion methods leads to flickering and is inappropriate.

Bertalmío et al. [3] consider extending image inpainting techniques to video sequences using ideas from fluid dynamics. As before, such video inpainting is useful for filling small nontextured holes in video sequences, but is unsuitable for completing large space–time holes caused by removal of macroscopic objects.

Wexler et al. [21] treat video completion as a global optimization problem, to enforce global spatiotemporal consistency during video completion. They solve the problem iteratively: missing video portions are filled pixel by pixel. Multiple target fragments are considered at different locations for the unknown pixel; for each, it seeks the most similar space–time source fragment elsewhere in the video. The fragments are merged according to similarity criteria to complete the unknown pixel. For speed, this method is performed at several scales using *spatiotemporal pyramids* and nearest-neighbor search algorithms [2]. Overall, however, this approach is slow, and the results appear blurred due to the fragment merging and smoothing operations.

Zhang et al. [23] segment video sequences into different nonoverlapping motion layers, each of which is completed separately. After removal of unwanted video objects in each layer, the method selects a reference frame in each layer and completes that frame. The solution is then propagated to other frames using the known motion parameters. This yields good results but is limited to rigid bodies for which the transformation between frames can readily be determined: for example, their appearance may not vary with time by rotating in three dimensions.

2.3 Mean shift and graph cut

Our method tracks moving objects to limit the search space when trying to find the best source fragment for repair. We use the *mean shift* algorithm [10]. It can rapidly

and robustly track nonrigid objects in videos using features such as color or texture using a Bayesian framework to find the most probable location for the tracked object in each frame. The mean shift algorithm has been applied to various vision problems, including robust feature space analysis [9], spatiotemporal video segmentation [12], and video tooning [19].

To merge source and target fragments smoothly, we use the *graph cut* technique [7] to find the best boundary between them—we wish to minimize pixel differences across the boundary. It works by expressing the problem as having to find the min-cut in a weighted graph. This method was used by Boykov [6] to segment N-dimensional images and later applied to image and video texture synthesis [16], foreground extraction for images [18], and photomontage [1]. It has already found use in fragment merging for filling holes in images in [22] and is well suited to this purpose for video completion.

The next section explains our algorithm in outline.

3 Overview and contribution

Our algorithm is based on texture synthesis; it is efficient and produces temporally coherent results.

The *video completion* problem is: given an input *video sequence* $V = v(x, y, t)$ with holes H where some unwanted objects have been removed during a sequence of frames, represented by a *matte* M, a bitmask indicating locations of each hole, our goal is to fill the holes one by one with plausible (background) pixel values based on the known regions.

Each single hole is filled in an iterative manner. Each time on the iteration, we *complete* a single video *fragment* in the hole. A fragment is a cubical neighborhood around some pixel. The fragment size is chosen according to the scale of the underlying hole in the video sequence. Iteration terminates when the hole has been filled.

A video fragment is completed using texture synthesis. We select a *target fragment* T centered at a pixel at the boundary of the hole; its pixels are partly unknown. We then search appropriate known parts of the video for a *source fragment* S having the greatest similarity to the known part of T. If the similarity is too low, we exit this iteration, otherwise, we merge the source fragment S with the target fragment T to fill unknown pixels in the target fragment (see discussion below).

Video completion must address two main problems: the time taken given the large amount of data and the control of temporal consistency between frames for added pixels. We use three main ideas to resolve these issues:

- **Search pruning using tracking**: To avoid searching the entire video for the best source fragment S, we use the *mean shift* method to *track* moving objects. This quickly determines a much smaller search space for plausible source fragments with high similarity to the target fragment. This greatly improves speed.
- **Fragment completion using graph cut**: We complete holes in the video sequence *fragment by fragment*. When merging the source information with the target, we use a *graph cut* algorithm to ensure that pixel differences across the boundary between the original material and the synthesized material are kept as small as possible. This is much faster than *pixel by pixel* video completion yet maintains fine details and produces smoothly merged results.
- **Temporal consistency**: To ensure the temporal consistency of new material and avoid flickering, we do the following: if two target fragments T_1 and T_2 are neighbors in time, we favor choosing corresponding source fragments S_1 and S_2, which are also neighbors in time. This simple method performs well.

The main contributions of this paper are thus threefold. The first is to introduce tracking into video completion, for two purposes. When selecting target fragments, considering whether a target is trackable is useful for comparing the merit of different targets in order to get a good target T. Then, we use tracking to limit the size of the search space for source fragments.

Secondly, we introduce graph cut methods into video completion to find the best seam between a target fragment T and a source fragment S. This enables us to merge T and S with the least visible seam while retaining high resolution details. This is crucial when performing fragment-by-fragment, rather than pixel-by-pixel, filling. Thirdly, we preserve temporal consistency during video completion by ensuring consistency of source fragments at adjacent time steps.

The next three sections give further details of each step.

4 Optimal target selection

In order to select a good target video fragment T for each iteration, we consider the *merit* of the fragment centered at each pixel in the hole. We take into account two factors: how much information is known in the target fragment and how well the target fragment can be tracked through the video sequence. The former information is stored in an *info map*, I, of the same size as the video sequence (or at least as big as the holes) at each pixel that is at the center of a target fragment. The latter information is stored in a similar *trackability map*, C. Both are explained shortly in more detail. As holes are filled incrementally, these maps can be quickly updated *locally*, after initial construction.

Suppose I_T stands for the *info map* value for a target video fragment T and C_T for the *trackability map* value.

Fig. 1. Target selection

We define the overall merit O_T for the target T as:

$$O_T = I_T + k\,C_T, \tag{1}$$

where an optimum choice for k seems to be about 2 or 3, to give more importance to C_T. It is simple to keep O_T updated as filling occurs, using a sorted list for all targets, allowing us to quickly find the target fragment with maximum merit.

As explained, if a suitable source fragment cannot be found for a given target, we ignore this target and try again with a new target. In practice, many target fragments have almost the same maximum merit value. We thus first select the N best target fragments and randomly choose one of them as the target. (We set N to about 40; if no suitable source is found, we adaptively increase N.)

4.1 Info map

The idea of the *info map* is to tell us how much information is *known* in the target video fragment T [13]. Let M_v be the matte value at pixel $v(x, y, t)$. The *info map* value I_T for T is given by:

$$I_T = \sum_{v \in T} M_v. \tag{2}$$

Clearly, the *info map* can easily be calculated initially by applying an all-in-one filter of the same size as T to the matte and multiplying the result by the negation of the matte, as shown in Fig. 2.

Figure 2 shows that larger values in the *info map* correspond to target fragments having more known neighboring pixels. Such target fragments are preferred (Fig. 1, left, middle). Two candidate targets and surrounding video fragments are marked in red. We prefer the target in the middle figure to the left figure because there is more known information in this video fragment. (We assume for this example that this hole is fixed over time.)

4.2 Trackability map

The *trackability map* measures how well a target fragment can be tracked through the video sequence. Trackability is computed for every candidate target fragment. For a target fragment T it is measured by the number of unknown pixels that are *trackable* in it—an unknown pixel is trackable

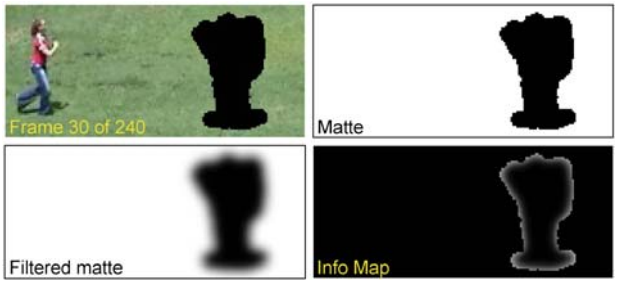

Fig. 2. *Top left* to *bottom right*: source frame, its matte, filtered matte, info map. Video data from [21]

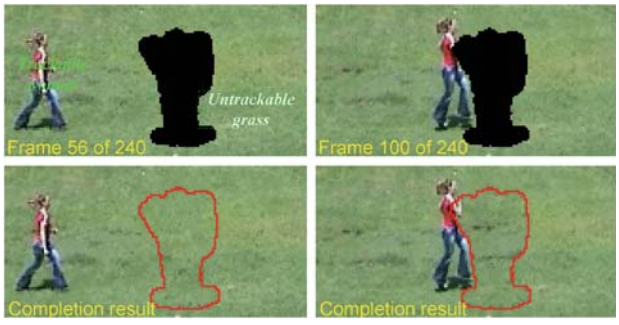

Fig. 3. *Top left* to *bottom right*: 56th source frame, 100th source frame, image completion of 56th frame, video completion of 100th frame

if and only if there is an adjacent known neighborhood that contains an object that can be tracked through the video. Let τ_v be a Boolean value saying whether pixel v is trackable. The trackability C_T is given by:

$$C_T = \sum_{v \in T,\, M_v = 0} \tau_v. \tag{3}$$

Using this information during target selection allows us to give priority to those rare objects which are trackable in the entire video. In Fig. 3, for example, a hole exists in the video across many frames. There are two kinds of objects in this video: a trackable woman and the grass. The left frame can be completed easily because sufficient information (about the grass) can be obtained from many other frames. On the other hand, only a few possibilities exist for correctly completing the right frame, as there are far fewer neighborhoods including the (trackable) woman

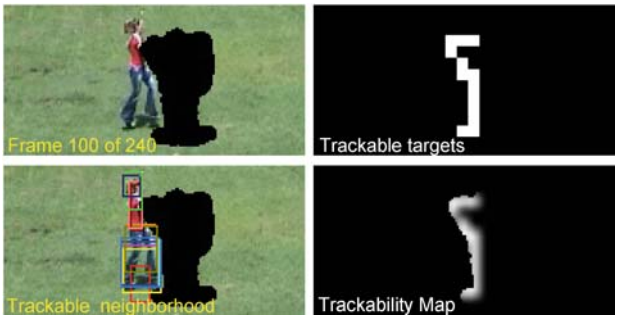

Fig. 4. *Top left* to *bottom right*: source frame, trackable target fragments, trackable neighborhoods of target fragments, trackability map of source frame

in an appropriate stance. If we select a target for filling this space–time hole using the left-hand frame, it will be very hard to complete the right frame in a globally temporally consistent way by the time we get to it. It is better to select trackable targets first like those shown in the right-hand frame. This is even more important than choosing a higher info map value. (Fig. 1, right). There is a neighborhood (green dashed lines) containing part of the woman, which can be tracked in the video sequence (Fig. 5). Thus, all unknown pixels in the target fragment are trackable, and we prefer this target fragment, in Fig. 1 (right), to the one in Fig. 1 (middle).

Trackability map computations are very quick for two reasons. Firstly, all pixels inside the same fragment can be considered to share the same neighborhood outside the fragment and so have common trackability information. The trackability map can thus be computed per fragment, rather than pixel by pixel, as shown in Fig. 4. The colored rectangles at the bottom left indicate trackable neighborhoods of various target fragments shown in white at the top right. Secondly, target fragments only exist at the edges of holes, limiting where trackability maps need to be processed.

5 Source selection using tracking

After choosing a target video fragment T, we now need to find the most appropriate source video fragment S.

We must avoid searching the whole video sequence, which is much too time consuming. Previous solutions [21] used spatial and temporal derivatives to estimate motion parameters of video fragments. The portion of the video to search for a suitable source fragment was restricted according to the motion parameters of the target fragment. This avoids much unnecessary computing, but can be improved. Spatial and temporal derivatives are useful for separating a moving foreground from the background, but less useful for processing static objects. Take the target in Fig. 1 (left), for example: its spatial and temporal rates of change are approximately zero. Using these motion parameters to limit the search range, *all* the grass must be searched. Generally, then, the search space in [21] is still highly redundant.

Instead, we use tracking to control the search, as shown in Fig. 5. Trackable and untrackable targets are treated differently. If a target is untrackable, as in Fig. 5 (1.1,right), the known video neighborhood around it is not trackable in the video sequence. Such a target fragment has unchanging color and texture throughout the whole sequence and belongs to the background. In part 2 of Fig. 5, for example, the woman is trackable, but all grass remains still and untrackable through the video. The need for global temporal consistency tells us that such background should be filled in the same way in each frame. A global search of the whole video for the best target is pointless, and instead we just search the frame containing the target pixel (any frame is as good as any other): see part 3 in Fig. 5 (right).

For a trackable target, e.g., the part of the woman shown in Fig. 5 (left), a known trackable neighborhood N overlaps it (green). This is active through the whole video. The target fragment belongs to a moving object in the video sequence, separate from the stationary background. In this situation, we apply the mean shift tracking algorithm to follow N through the video sequence, giving a precise space–time route for $N(t)$. This gives a set of small windows which include the moving $N(t)$ in each frame, as shown in part 2 of Fig. 5 (left). The areas to be searched are exactly determined by the tracked neighborhood, marked by green squares in part 3 of Fig. 5 (left).

In both trackable and untrackable cases, we only have to search a small portion of the whole video to find the best source fragment, giving high efficiency.

Apart from the two cases described above, we must also consider the case in which we cannot find a source fragment that is sufficiently similar to the target. For example, take the target fragment shown in the first row of Fig. 7. Using the criteria in Sect. 4, we do not select the target illustrated because of its low *trackability map* value: it only has grass (background) in the known neighborhood around it. The difference between this target and the best source is very large as part of the woman's leg is present in earlier frames in the space–time fragment. We skip such targets (Sect. 7).

6 Graph cut fragment updating

After we have determined both the target and source fragments T and S, we must combine them to produce an output video fragment. Simply copying target pixels where known, and source pixels otherwise, can lead to an obvi-

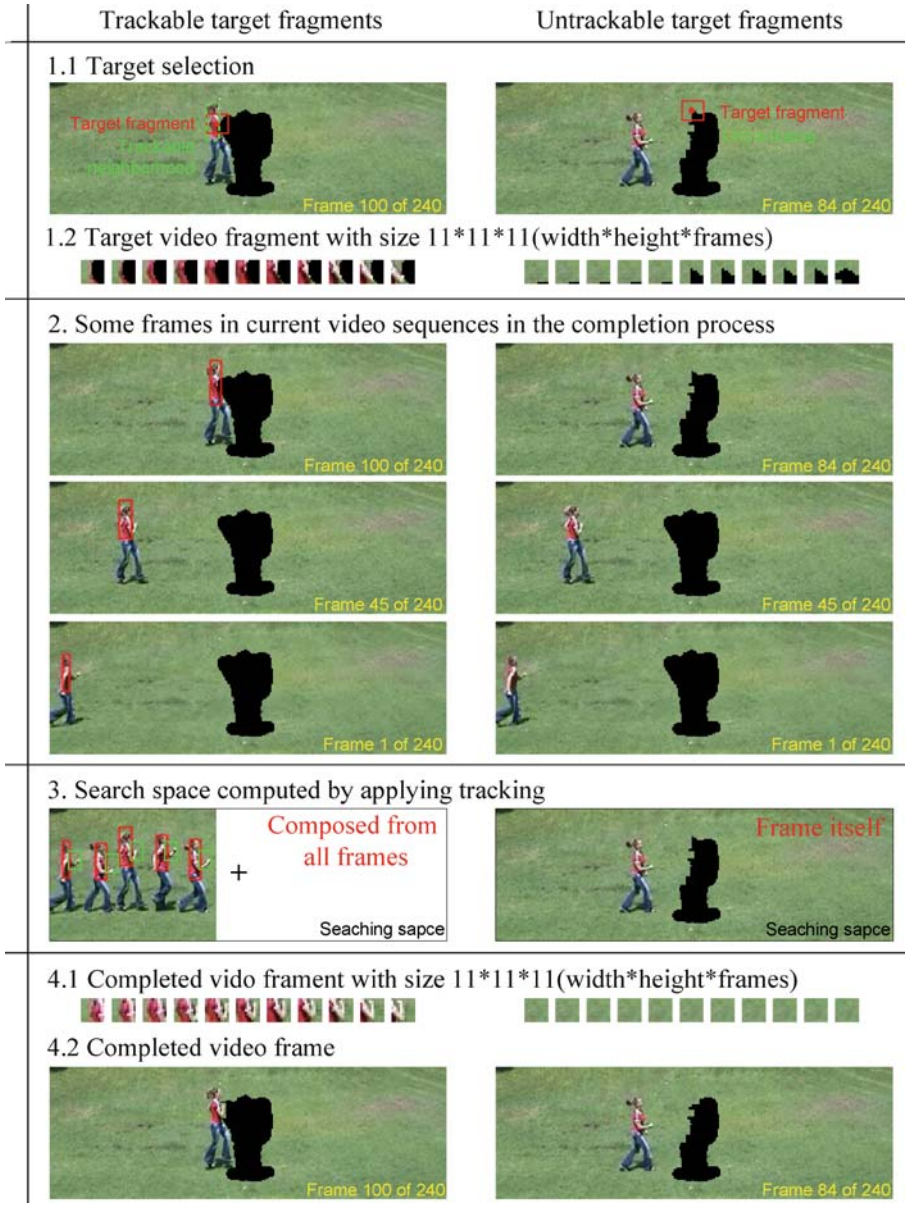

Fig. 5. Source search for trackable (*left*) and untrackable (*right*) target fragments

ous join in the output. We avoid this problem by finding the least visible seam between target and source fragments in the overlap region.

The least visible seam is the one for which pixel differences across the seam are as small as possible. The best seam can be found by finding the minimum cut of a weighted graph formed by joining neighboring pixels in a difference image. This *graph cut* method has already been used for merging textures in images [16]. Note that we only apply this method to the region of *overlap*, O, of the target and source fragments, which we first have to compute: $O = T \cap S$. We compute the color difference values c_o, expressed as an r, g, b vector, at each pixel $o(x, y, t)$ in the overlap region O:

$$c_o = |c_t - c_s|, \quad o(x, y, t), t(x, y, t), s(x, y, t) \in O, T, S. \tag{4}$$

We now build an undirected weighted graph G using the pixels in O. For each edge between a pair of connected pixels o_i and o_j in O, with color differences c_{o_i} and c_{o_j}, we define the weight w_{ij} of that edge to be:

$$w_{ij} = \begin{cases} \kappa \left(1 - \exp\left(-\frac{\|c_{o_i}\| + \|c_{o_j}\|}{2\sigma^2}\right)\right) & \text{if } N(o_i, o_j) \\ \infty & \text{otherwise,} \end{cases} \tag{5}$$

Fig. 6. Comparison of graph cut and direct fragment update

where $\|\cdot\|$ denotes the length of a vector, $N(.,.)$ returns true if the pixels are six-connected, and κ and σ are constants (about 10 and 5 in practice). This weight is less when corresponding adjacent pixels in T and S are similar, which is where we want the seam to be. Thus, the best seam is the one giving a minimum cut for graph G_S.

The advantages of using the graph cut method are shown in 2D in Fig. 6. On the left is an input image of a wall, with a hole to be filled. At the center is the completed result using graph cut, while on the right is the direct fragment update result from [22] showing a structural discontinuity inside the blue circle.

7 Achieving temporal consistency

Achieving temporal consistency is the other main requirement for video completion: people are highly sensitive to motion. Temporal aliasing is much more important than spatial aliasing in image completion. Consider Fig. 3. The image completion result for frame 56 and the video completion result for frame 100 are incompatible in the same video sequence. The large difference between their backgrounds will produce obvious flicker in the output video. In [21], temporal consistency is achieved using costly global optimization. The objective function forces coherence of all video fragments containing the same completed pixel. To complete one pixel, many source fragments are considered for many target fragments around any unknown pixel, and source fragments are merged according to similarity. In this way, the completed pixel maintains high coherence with all fragments around it, but at high computational cost.

To achieve temporal consistency, we use a simpler approach which encourages the *source fragments* to be temporally consistent with each other. The basic idea is to supplement the current search space for source fragments with an extra region R. These extra candidates are chosen to be temporally consistent with previously completed video filling. R must be computed explicitly because it is not necessarily included by default in the search region

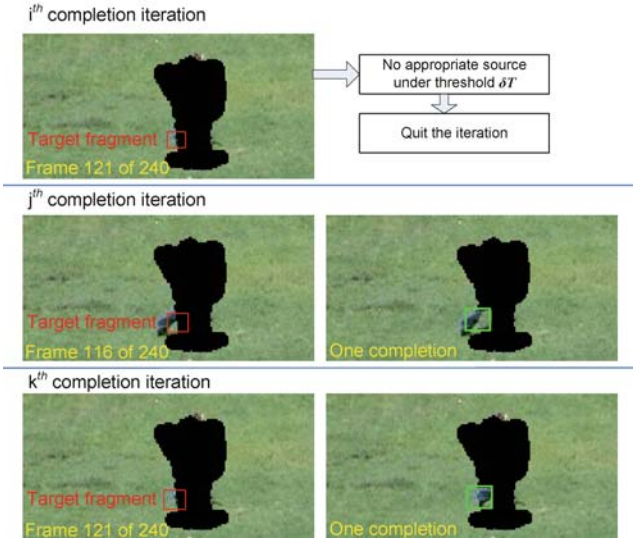

Fig. 7. Supplemental search region

found by the tracking algorithm. Take Fig. 7, for example. During the ith iteration of hole filling, we may encounter a target T with no trackable neighborhood around it, as explained in Sect. 5, and for which no appropriate source fragment can be found, so we skip this iteration. If later in the jth iteration we select a target fragment some frames before T, like that marked in Fig. 7, we can complete it successfully as it is trackable through the video sequence (it contains part of a moving leg). If later still, in the kth iteration, we select T again, this time we can complete it using candidates from the supplemental region R.

When searching for the best source fragment, we add a bonus to the similarity measurement between the target and *source fragments in the supplemental region R*. Doing so gives greater weight to such results than ones from elsewhere in the search space, providing temporal consistency in the results.

The idea of the supplemental region is illustrated in Fig. 8. Suppose during the previous completion step we found source fragment S_1 for target fragment T_1 (top line of Fig. 8). We record T_1 and S_1 as a pair in a list L. When we select a new target fragment, T_2, we check the list L to see if there is any filled target fragment overlapping with T_2 in space and within a certain time before or after T_2. In this example, we find T_1 occurring before the current target T_2, shown by the yellow arrow in Fig. 8. If more than one such target exists, we prefer ones with greater overlap or, if equal overlap, ones which are more trackable. There is a strong possibility that the best source fragment for T_2 is somewhere just after S_1 in time, in the known portion of the video sequence, and using it will ensure temporal consistency is maintained.

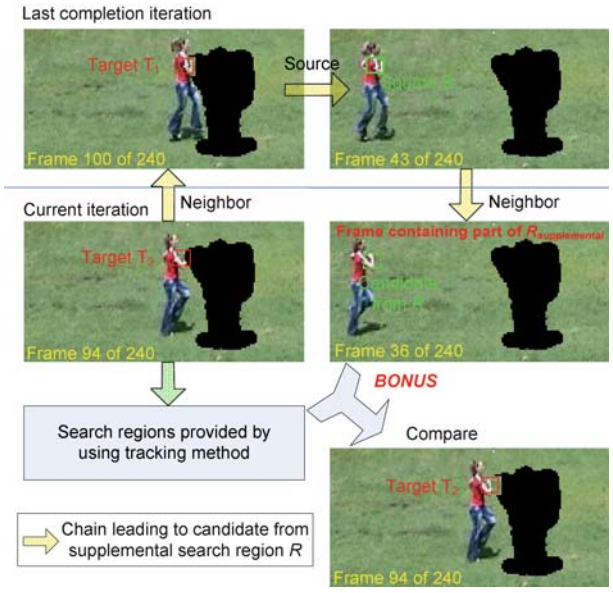

Fig. 8. Bonus for supplemental search region

8 Results

Our algorithm has been applied to various videos of complex dynamic scenes. Since the perceived quality of the completed video frames depends on human judgement, rather than mathematical measures, we show some frames taken from video sequences to demonstrate the effectiveness of our method. Processing times are given in Table 1. The time taken is proportional to both the hole size (pixels) and the video length (frames). The former decides the number of filling iterations, while the latter is related to the size of search space.

Figure 9 demonstrate the results from a two-woman video sequence after one woman has been removed. This sequence is tricky as the hole is large. More significantly, in addition to removing the stationary woman, we have also removed the woman walking in and around the hole for tens of frames, as an unwanted side effect. We wish to keep the moving woman, and indeed our filling method successfully puts her back. The visual results and the performance table show that our algorithm is efficient and robust.

9 Discussion

9.1 Comparison with Wexler's method

Compared to Wexler's method [21], the main advantage of our algorithm is efficiency. If we spend the same time on source patch searching, we can complete the video N^4/K times faster than their method, where N is the patch size in both algorithms, typically 5, and K is their speedup factor due to use of spatiotemporal pyramids (but which also causes blurring), typically 8. One factor of N^2 is due to pixel-by-pixel completion in their case—we fill a whole fragment at once; the second factor of N^2 is due to the number of source patches they must search for each pixel. In practice, our algorithm can find source patches more quickly because our search space is carefully selected by tracking, typically at the scale of a single frame, and theirs is much larger. Our implementation of their algorithm takes over 4 h for the "Space" video example, whereas ours takes 17 min. Another advantage compared to [21] is that our algorithm maintains finer details in the output, as seen, for example, in Fig. 10.

9.2 Handling dynamic cameras

We cannot currently deal with scenes from dynamic cameras: scaling, rotation, and other transformations occur between the target and candidate source fragments. Finding a suitable source fragment requires knowledge of the motion parameters of the camera, which is also a problem for [21]. Estimating motion parameters from video sequences *is* possible [14], but even so, the search would be more complex. Secondly, temporal consistency would be more difficult to maintain in dynamic scenes, as the neighborhood would need to take into account the motion parameters of the camera. Thus, extending our algorithm to handle dynamic cameras is not impossible in principle, but it needs further work.

9.3 Artifacts

When merging target and source fragments using graph cut, a special case arises at the boundary of the hole. In this case, we should not modify the known part of the video sequence adjacent to the hole but leave it as is and only fill in the unknown pixels. There is thus no need to apply graph

Table 1. Timings for video completion using a 2.4-GHz Pentium 4 CPU

Video	Length (frames)	Video size (pixels)	Hole size (pixels × frames)	Time for completion (min)
Hopping (from [21])	240	288 × 96	1768 × 240	75.25
Space	100	320 × 240	1942 × 100	16.75
Beach (from [21])	83	180 × 60	609 × 49	12.00

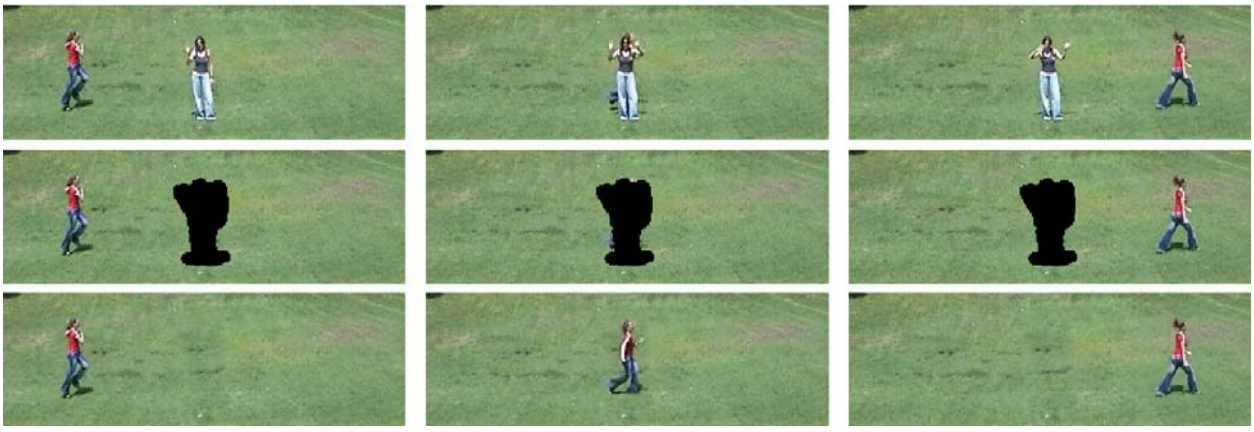

Fig. 9. *Top* to *bottom*: input video, removal of stationary woman, filling results

Fig. 10. *Left* to *right*: frames 45, 47 of input video, one lady removed, results from [21], our results

cut in this case, and we should just copy from the source fragment to the unknown part of the target fragment. However, this approach can lead to visible artifacts at the edge of the holes, as there is nothing to enforce smoothness of pixel intensities across the edge of the hole (see the earlier discussion).

A simple approach to diminishing such artifacts would be to apply the *border matting* technique from [18].

9.4 Loss of tracking

At the beginning of video completion, tracking works very well, as our method selects targets with high trackability. As completion proceeds, cases can arise in which we lose tracking, and we cannot find a good source fragment. In such situations, the supplemental regions explained in Sect. 7 often still provide an appropriate source fragment. If no good source fragment is found in the supplemental region either, we abandon filling this target fragment and select a new target (this happens in less than 10% of cases for all three example videos), as described in Sect. 3. Overall, even when tracking fails, we still get good results.

10 Conclusions and future work

We have given a novel, efficient, and visually pleasing approach to video completion. We carefully select suitable target fragments and limit the search for source fragments using tracking. Holes are filled fragment by fragment with a texture synthesis algorithm, using a graph cut algorithm to find good seams between target and source fragments. Temporal consistency is achieved by further control of source fragment selection to avoid flickering.

Good results have been achieved to date. We wish to extend the work to more complicated and dynamic scenes, involving, for example, complex camera and object motions in three dimensions.

Acknowledgement This work was supported by the Natural Science Foundation of China (Project No. 60225016, 60321002) and the National Basic Research Project of China (Project No. 2002CB312101).

References

1. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Trans. Graph. **23**(3), 294–302 (2004)
2. Arya, S., Mount, D.M.: Approximate nearest neighbor queries in fixed dimensions. In: SODA: ACM-SIAM Symposium on Discrete Algorithms, pp. 271–280 (1993)
3. Bertalmío, M., Bertozzi, A.L., Sapiro, G.: Navier-stokes, fluid dynamics, and image and video inpainting. In: Computer Vision and Pattern Recognition, vol. 1, pp. 355–362 (2001)
4. Bertalmío, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: SIGGRAPH, pp. 417–424 (2000)
5. Bertalmío, M., Vese, L.A., Sapiro, G., Osher, S.: Simultaneous structure and texture image inpainting. In: Computer Vision and Pattern Recognition, vol. 2, pp. 707–712 (2003)
6. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: International Conference on Computer Vision, pp. 105–112 (2001)
7. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In: Energy Minimization Methods in Computer Vision and Pattern Recognition, pp. 359–374 (2001)
8. Chan, T.F., Shen, J.: Mathematical models for local nontexture inpaintings. SIAM J. Appl. Math. **62**(3), 1019–1043 (2002)
9. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)
10. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: Computer Vision and Pattern Recognition, pp. 2142–2151 (2000)
11. Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based inpainting. IEEE Trans. Image Process. **13**(9), 1200–1212 (2004)
12. DeMenthon, D.: Spatio-temporal segmentation of video by hierarchical mean shift analysis. In: Statistical Methods in Video Processing Workshop,Image and Vision Computer (2002)
13. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. ACM Trans. Graph. **22**(3), 303–312 (2003)
14. Heuer, J., Kaup, A.: Global motion estimation in image sequences using robust motion vector field segmentation. In: ACM Multimedia **1**, 261–264 (1999)
15. Jia, J., Wu, T.P., Tai, Y.W., Tang, C.K.: Video repairing: inference of foreground and background under severe occlusion. In: Computer Vision and Pattern Recognition, vol. 1, pp. 364–371 (2004)
16. Kwatra, V., Schödl, A., Essa, I.A., Turk, G., Bobick, A.F.: Graphcut textures: Image and video synthesis using graph cuts. ACM Trans. Graph. **22**(3), 277–286 (2003)
17. Levin, A., Zomet, A., Weiss, Y.: Learning how to inpaint from global image statistics. In: International Conference on Computer Vision, pp. 305–312 (2003)
18. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**(3), 309–314 (2004)
19. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.F.: Video tooning. ACM Trans. Graph. **23**(3), 574–583 (2004)
20. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: SIGGRAPH, pp. 479–488 (2000)
21. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Computer Vision and Pattern Recognition, vol. 1, pp. 120–127 (2004)
22. Zhang, Y., Xiao, J., Shah, M.: Eurographics 2004 / Short Presentations, Region completion in a single image (2004)
23. Zhang, Y., Xiao, J., Shah, M.: Motion layer based object removal in videos. In: IEEE Workshop on Applications of Computer (2005) (in press)

Y.-T. JIA is currently pursuing a Master's degree in computer science at Tsinghua University. His current research interests include realistic rendering, video analysis, and synthesis.

S.-M. HU is currently a professor of computer science at Tsinghua University. His research interests include digital geometry processing, video-based rendering, rendering, computer animation, and computer-aided geometric design. He obtained his Ph.D. in 1996 from Zhejiang University. He is on the editorial boards of *Computer Aided Design*.

R.R. MARTIN obtained his Ph.D. in 1983 from Cambridge University and is now professor at Cardiff University. He has published over 140 papers and 9 books covering such topics as solid and surface modeling, intelligent sketch input, geometric reasoning, reverse engineering, and various aspects of computer graphics. He is on the editorial boards of *Computer Aided Design* and the *International Journal of Shape Modelling*.