Video Inpainting Under Constrained Camera Motion
Kedar A. Patwardhan, Student Member, IEEE, Guillermo Sapiro, Senior Member, IEEE, and Marcelo Bertalmio

Abstract—A framework for inpainting missing parts of a video sequence recorded with a moving or stationary camera is presented in this work. The region to be inpainted is general: It may be still or moving, in the background or in the foreground, it may occlude one object and be occluded by some other object. The algorithm consists of a simple preprocessing stage and two steps of video inpainting. In the preprocessing stage, we roughly segment each frame into foreground and background. We use this segmentation to build three image mosaics that help to produce time consistent results and also improve the performance of the algorithm by reducing the search space. In the first video inpainting step, we re-construct moving objects in the foreground that are “occluded” by the region to be inpainted. To this end, we fill the gap as much as possible by copying information from the moving foreground in other frames, using a priority-based scheme. In the second step, we inpaint the remaining hole with the background. To accomplish this, we first align the frames and directly copy when possible. The remaining pixels are filled in by extending spatial texture synthesis techniques to the spatiotemporal domain. The proposed framework has several advantages over state-of-the-art algorithms that deal with similar types of data and constraints. It permits some camera motion, is simple to implement, fast, does not require statistical models of background nor foreground, works well in the presence of rich and cluttered backgrounds, and the results show that there is no visible blurring or motion artifacts. A number of real examples taken with a consumer hand-held camera are shown supporting these findings.

Index Terms—Camera motion, special effects, texture synthesis, video inpainting.

I. INTRODUCTION AND OVERVIEW

A. Introduction to the Video Inpainting Problem

The problem of automatic video restoration in general, and automatic object removal and modification in particular, is beginning to attract the attention of many researchers. In this paper we address a constrained but important case of video inpainting. We assume that the camera motion is approximately parallel to the plane of image projection, and the scene essentially consists of stationary background with a moving foreground, both of which may require inpainting. The algorithm described in this paper is able to inpaint objects that move in any fashion but do not change size appreciably. As we will see below, these assumptions are implicitly or explicitly present in most state of the art algorithms for video inpainting, but they still leave a very challenging task and apply to numerous scenarios. For a detailed discussion about these assumptions, including how they are actually relaxed in the real examples here presented, please refer to Section II-A.

A number of algorithms for automatic still image completion have been proposed in the literature [3], [5], [6], [11]. These cannot be generalized in a straightforward manner to address the challenging problem of video completion reported in this paper. There has also been some preliminary work on frame-by-frame partial differential equations (PDEs) based video inpainting [4], following [5]. In [4], the PDE is applied spatially, and completes the video frame-by-frame. This does not take into account the temporal information that a video provides, and its application is thereby limited. Also, the PDEs based methods interpolate edges in a smooth manner, but temporal edges are often more abrupt than spatial edges.

The authors in [24] recently proposed a method for spatetime completion of damaged areas in a video sequence. They pose the problem of video completion as a global optimization problem, which is inherently computationally very expensive. The work extends to space time the pioneering technique of nonparametric sampling developed for still images by Efros and Leung [13]. This implies the assumption that objects move in a periodic manner and also they do not significantly change scale, because otherwise the “copy and paste” approach of [13] would fail. Although the results are good, they suffer from several shortcomings. Only low-resolution videos are shown, and oversmoothing is often observed. This is due in part to the fact that pixels are synthesized by a weighted average of the best candidates, and this averaging produces blurring. Also, the camera is always static in all the examples in that paper. Though the reason for this is not discussed, it is probably due to the fact that the authors use a very simple motion estimation procedure involving the temporal derivative. We present results comparing with their approach in the experimental section.

An interesting probabilistic video modelling technique has been proposed in [10], with application to video inpainting. They define “epitomes” as patch based probability models that are learnt by compiling together a large number of examples of patches from input images. These epitomes are used to synthesize data in the areas of video damage or object removal. The video inpainting results are reported to be similar to those in [24], are primarily low resolution, and oversmoothing is also observed.

Very interesting work for repairing damaged video has been recently reported in [15]. Their method involves a gamut of different techniques that make the process of inpainting very complicated. There is an important amount of user interaction: the
user has to manually draw the boundaries of the different depth layers of the sequence. Also, the algorithm has to “learn” the statistics of the background. The motion of the objects in the background is restricted to be periodic, which implies that objects also do not change scale as they move, so movement is approximately on a plane parallel to the projection plane of the camera. All the examples shown involve either a static camera or a very smooth horizontal “lateral dolly” type of camera motion. The results are good, although not free from artifacts. Damaged moving objects are reconstructed by synthesizing a new un-damaged object, overlaying it on the sequence, and moving it along a new, interpolated trajectory. This approach produces very noticeable artifacts where objects move in an unrealistic way (for instance a walking person seems at some points to float over the ground). We here present results for videos of the same type as those in [15]. A related approach, also combining motion layer estimation and segmentation with warping and region filling in, has been reported in [25].

In [16], the authors propose a video inpainting technique also based in the nonparametric sampling of Efros and Leung [13]. Again, as in [24], this implies the assumption that objects move in a periodic manner and also they do not change scale. The authors use tracking to reduce the search space, and graphic cuts to merge the synthesized blocks. This approach can only deal with scenes from a static camera. And although the authors do not provide video examples, they report that their results suffer from artifacts at hole boundaries, and the filling process may fail when tracking is lost.

B. Key Contributions

Our approach is fundamentally related to the nonparametric sampling method proposed in [13] for the problem of 2-D texture synthesis. This method was further improved upon by using a priority and confidence based synthesis in [11]. We adapted and extended this technique for video inpainting for the static camera case in [20]. In this paper, we present the extension of our work in [20], including addressing the case when the camera moves. For this we introduce foreground, background, and optical-flow mosaics (see Section II-B), which not only help to produce good quality results, but also reduce the search space and lead to a faster implementation. Although the copy and synthesis components of the proposed framework are basically 2-D, the whole search and metric distances fully exploit the spatiotemporal information.

Our key contribution is the following: we present a simple and fast (compared with the literature) method to automatically fill-in video “holes” which shares the same assumptions of the state of the art works on the subject while being free of the common visual artifacts (blurring, unrealistic motion) those works present, and at the same time relaxing the static camera constraint. Fig. 1 gives an overview of our technique. The subsequent sections describe in detail each step in the proposed video inpainting process.

II. ASSUMPTIONS AND PREPROCESSING

A. Basic Assumptions

In this work, we make several assumptions on the kind of video sequences we are able to restore. As mentioned above, these assumptions are implicitly or explicitly shared by most state of the art works on the subject, often in an even more restrictive fashion.

Our basic assumptions are the following.

- The scene essentially consists of stationary background with some moving foreground.
- Camera motion is approximately parallel to the plane of image projection. This restriction ensures that background objects will not (significantly) change size, allowing for texture synthesis in the spirit of [13], which cannot deal with changes in size nor perspective.
- Foreground objects move in a repetitive fashion. In order to recover occluded or damaged foreground, and without the use of probabilistic models or libraries (used for instance in [1]), the vital information must be present in the video itself. Hence, this “periodicity” assumption.

```
Pre-processing:

| Segment each frame into |
| "Static Background + Moving Foreground" |
| Spatially align all frames, compute optical flow and create background and foreground mosaic |

Motion Inpainting:

| Compute the highest priority location for motion filling-in |
| Search in the foreground mosaic, to find the possible candidate frames |
| - Search each candidate frame for best match, |
| - Copy moving object from match to damaged frame, and |
| - Constrain priorities for next iteration |
| Continue until all pixels either are filled-in or have zero-priority |
| Repeat this block for all frames requiring motion inpainting |

Background Filling-in:

| Temporal copying |
| Priority based texture synthesis to fill in the remaining hole |
```

Fig. 1. Overview of the proposed video inpainting algorithm.
Moving objects do not significantly change size. Again, this restriction is imposed by the use of the nonparametric texture synthesis of [13]. This constraint can be removed by using a multiscale matching algorithm which can address the change in size when the object moves away from or towards the camera.

All the examples in this paper are taken with a hand-held camera, thereby complying with these assumptions only partially, while still producing very satisfactory results.

B. Preprocessing

The simple assumptions that we make allow us to compute a rough “motion confidence mask” $M_c$ for each frame just by comparing it with the following frame using block matching.\footnote{We only take into account blocks that have no information missing.} The median shift of all the blocks in the image gives a good estimate of the camera shift in this case. Any block that has considerable shift after subtracting this median camera motion is assumed to belong to the moving foreground. Hence, given that the motion of the camera does not produce any transformation of the static background besides translation, $M_c$ can be easily computed by a simple thresholding of the block-matching result. We should note that we could use course more advanced techniques to detect moving objects and to separate foreground from background, see for example [7], [19], [21], [23], and references therein, but all the examples in this paper were obtained with the very simple method of computation for $M_c$ described above.\footnote{We wish to clarify that the $M_c$ motion masks are different from the masks that indicate the area of inpainting (the “hole”): the latter are given by the user, as in every (image or video) inpainting procedure.}

As we mentioned earlier, we use image mosaics to be able to deal with some camera motion and to speed up the inpainting process. A mosaic is a panoramic image obtained by stitching a number of frames together. In the preprocessing stage we build three mosaics: a background mosaic, a foreground mosaic, and an optical-flow mosaic. The computation of $M_c$ gives us a segmentation of the sequence into foreground and background layers, as well as a good estimate of the camera shift for each frame. We use this camera shift estimation to align (register) the frames. Each mosaic is built from the set of aligned overlapping frames in the following way: each pixel of the mosaic is the average of the overlapping components. This is straightforward in the case of the foreground and background mosaics. In Fig. 3, we can see the mosaics obtained from a video sequence shown in Fig. 2. For the optical flow mosaic, which contains data used for the Sum of Squared Difference (SSD) computations as shown below, we use a two-channel image to store the horizontal and vertical components of the residual optical flow, that is, the motion vectors from which we have subtracted the camera shift. In Fig. 3 we use color coding to represent the direction of these 2-D vectors: green tones indicate horizontal motion and red tones indicate vertical motion. We must mention that there are more sophisticated mosaic generation techniques in the literature to handle camera motion, like [2], [12], and [22], but our simple approach has been satisfactory for the results reported here.

This mosaic generation step allows us to do a quick search for possible candidate frames from where to copy information when filling in the moving object, thereby speeding up the implementation by limiting our search to only these candidate frames instead of the entire sequence. The next section discusses the moving foreground completion step in detail.

III. Motion Inpainting

The algorithm consists of a preprocessing stage and two steps of video inpainting. In the first video inpainting step, that of Motion Inpainting, we reconstruct foreground (moving) objects that are “occluded” by the region to be inpainted. To this end we fill the gap as much as possible by copying information from the moving foreground in another frame, using a priority-based scheme and the above mentioned three mosaics. Here, we are using the technique introduced for still images by Efros and Leung in [13] and refined in [11] by Criminisi et al., so let us start this section by briefly reviewing that procedure.

A. Review of Nonparametric Sampling

Given the problem of inpainting an image “hole” $\Omega$ in a still image $I$, Efros and Leung proposed in [13] a simple yet extremely effective algorithm. For each pixel $P$ in the boundary of $\Omega$, consider its surrounding patch $\Psi_P$, a square centered in $P$. Compare this patch, using a simple metric such as the sum of squared differences (SSD), with every possible patch in the image. There will be a set of patches with small SSD distance to $\Psi_P$. Randomly choose a patch $\Psi_Q$ from this set, and copy its central pixel $Q$ to the current pixel $P$. We have filled $P$, so next we proceed to the following pixel in the boundary of $\Omega$. Criminisi et al. noted that the order in which the pixels of $\Omega$ are
filled is crucial, so in [11], they proposed an inpainting procedure which is basically that of Efros and Leung with a new ordering scheme that allows to restore long structures “occluded” by the hole Ω. The ordering scheme proposed by Criminisi et al. is as follows. They compute a priority value for each pixel in the boundary of Ω, and at each step the pixel chosen for filling is the one with the highest priority. For any given pixel $P$, its priority $Pr(P)$ is the product of two terms: a confidence term $C(P)$ and a data term $D(P)$: $Pr(P) = C(P)D(P)$. The confidence term $C(P)$ is proportional to the number of undamaged and reliable pixels surrounding $P$. The data term $D(P)$ is high if there is an image edge arriving at $P$, and highest if the direction of this edge is orthogonal to the boundary of $Ω$. We, thus, get higher priority values at significant edges that need to be continued, as in [5].

At this point, we suggest the reader to take a close look at Figs. 4–6, in order to have a further understanding of our algorithm. It is important to note that data from the mosaics is not used to fill in the damaged frames. The mosaics are only used to search for the “candidate-undamaged-frames,” from where information can be copied into the damaged frame $F$.

B. Initial Guess Search

Coming back to our problem, we start by restoring moving objects “occluded” by the gap in our video sequence. We want this restoration to be done by copying suitable information from other frames, but searching the entire sequence for a good match would be computationally very inefficient. Hence, we need to first have a small set of candidate frames which can provide information to complete those moving objects. To achieve this, we first search in the foreground mosaic, since we are inpainting foreground objects, to find the candidate frames, i.e., a small subset of frames where we will look for the best match. This “initial guess search” is implemented using the following steps (refer to Figs. 4–6).

Pseudo-code for completing moving object in one frame ($F$):

1. $Ψ_p = \text{getHighestPriorityPatch}(F, M_c)$;
2. While ($Ψ_p$ exists):
   - $Ψ_{pm} = \text{MosaicCoord}(Ψ_p, \text{Cam-Shift})$;
   - $Ψ_{pund} = \text{mosaicSearch}(Ψ_{pm})$;
   - $(f_1, f_n) = \text{getCandidates}(Ψ_{pund})$;
   - $Ψ_q = \text{getMatch}(F, Ψ_p, (f_1, f_n))$;
   - copy ($Ψ_q$, $Ψ_p$);
   - constrainPriority ($Ψ_p$);
   - updateConfidence ($Ψ_p$);
   - updateMosaics ($Ψ_p$);
3. Repeat the above procedure for each frame requiring moving object completion.

Fig. 6. Motion inpainting scheme. Green dots indicate highest priority. (Frame A) Red squares indicate the patch to be inpainted and (frame B) the corresponding best match. Areas filled with red are constrained to have zero priority.

1) In the current damaged frame under consideration, find the highest priority location $P$ and its surrounding patch $Ψ_P$.
2) Using the already available camera shifts computed during the preprocessing step, find the corresponding location $P_{m}$ for $P$ and also its surrounding patch ($Ψ_{p_{m}}$) in the foreground mosaic.
3) Using $Ψ_{p_{m}}$ as a template perform a search in the foreground mosaic to find the matching patch(es) $Ψ_{p_{m}}^{\text{match}}$.
4) Now, using the camera shifts and the motion confidence masks for each frame, identify the frames that have motion at the location corresponding to the mosaic area specified.

3We start our filling in with the “temporally outermost” damaged frame (in the 3-D video cube) and move toward the center of the “hole” in the video cube. This approximately gives more priority to the frames that have more undamaged information in the temporal vicinity.
by the matching patch(es) \( \Psi_{P_{\text{final}}} \). These frames are the candidate frames for searching a matching patch for the highest priority location \( P \) in the current frame.

Now some details on the above steps. First, for the data term \( D(P) \) in the priority computation, we use the following formula:

\[
D(k) = \frac{\|( \nabla M^L_k \| \alpha \cdot n_k \|}{\alpha}
\]

where \( \alpha \) is a normalizing constant (usually 255) and \( n_k \) is the normal to the hole boundary. The inner product of the rotated gradient of \( M^L \), \( \nabla M^L \), and the normal \( n_k \) is computed using central differences.\(^4\) Second, when looking for a (2-D) match for the template patch \( \Psi_{P_{\text{ref}}} \), we follow the approach used in [24]: We use a SSD metric involving a 5-D vector value composed by the color values \( R \), \( G \), and \( B \) and the optical flow values \( V_x^0 \) and \( V_y^0 \).\(^5\) The optical flow components are computed using the simple approximation \( V_x = I_t/I_x \) and \( V_y = I_t/I_y \), where \( I \) is the grayscale frame under consideration,\(^6\) and \( I_x \), \( I_y \), and \( I_t \) are its horizontal, vertical, and temporal derivatives, respectively (computed with a very simple numerical scheme, like central differences). The optical flow components can be computed with more recent, robust and fast state of the art techniques such as [8] and [9], but all our results were obtained with the very simple approximation just described.

Adding optical flow to the SSD vector helps us to ensure motion consistency. For example, if the moving person in a video has his right leg going forward and left going backward, there is no way to get a similar match without using optical flow, because in a 2-D image this situation would look similar (in R,G,B) to the situation when the two legs are in the same position but moving in the opposite direction (i.e., left leg moving forward and right moving backward).

\[ C(p) = \sum_{q \in \Psi_P \cap (M_c \setminus \Omega)} C(q) \]

where \( \Psi_P \) is the area of the patch and \( \Omega \) is the region of inpainting.

Finally, we update the foreground and the optical flow mosaics with the newly filled-in data.

\[ C(p) = \sum_{q \in \Psi_P \cap (M_c \setminus \Omega)} C(q) \]

where \( \Psi_P \) is the area of the patch and \( \Omega \) is the region of inpainting.

E. Ending the Foreground Inpainting

We repeat the above steps (Sections III-B–D) until all the pixels in the inpainting area are either filled in or have zero priority for motion inpainting (i.e., are “disabled” as explained above). This is precisely our indication that moving objects have been fully inpainted in the current frame. We now repeat this process for all the frames that require motion inpainting. This gives us a sequence with only moving objects filled in, and the rest of the missing region needs to be filled in with background.

IV. Background Inpainting

Once we have finished the stage of Motion Inpainting, we enter the stage where we inpaint the background. To accomplish this we first align the frames and directly copy whenever possible, while the remaining pixels are filled in by extending spatial texture synthesis techniques to the spatiotemporal domain. Let us see this in a little more detail.

When there is camera motion involved, often the background is less occluded in one frame than another (see [17] and [18]). When filling in the background, we align all the frames using the precomputed shifts, and then look for background information available in nearby frames. We then copy this temporal information using a “nearest neighbor first” rule, that is, copy available information from the “temporally nearest” frame (for more details refer to [20]). Note that this will, of course, be faster and of better quality than a simple block synthesizing procedure.

In cases where the occluder is stationary (refer to Fig. 7), there is a considerable part of the background that remains occluded in all of the frames. This shows up as a hole in the background mosaic. We fill in this hole directly on the background mosaic using the priority based texture synthesis scheme in [11] (extended to use a 5-D vector of colors and motion as explained in the previous section). The missing information in each frame is then copied from the the inpainted background mosaic, by spatially aligning the frame with the mosaic using the precomputed

\[ C(p) = \sum_{q \in \Psi_P \cap (M_c \setminus \Omega)} C(q) \]
Fig. 7. (Left) Missing part of the background is (right) filled in using a priority based texture synthesis scheme derived from [11].

Fig. 8. Left: A frame from the video in Fig. 11 shown in large scale. Right: Its inpainted result. Resolution is 640 × 480. Notice how there is no blur in the inpainted region, even at this full resolution.

shifts. This leads to a consistent looking background throughout the sequence.

V. EXAMPLES

All the videos referred to in this section have been captured using a consumer hand-held camera, providing a video resolution of 640 × 480 pixels per frame. The natural motion of a hand-held camera is a very common filming scenario. These and other video examples may be seen at http://www.tc.umn.edu/~patw0007/video-inpainting, some at full resolution (640×480), some at half resolution (320×240). Please note that, even if we display the inpainted videos at full resolution, no blurring artifacts appear, Fig. 8 shows at large scale a restored frame. Also, in the video results, it can be observed that inpainted moving objects have a consistent, natural looking trajectory. These results are state of the art, lacking the visual artifacts present in recent works on the subject, and with a faster and generally simpler technique.

In Fig. 9, we created an artificial rectangular hole in each frame at the same location. This presents not only a challenging task but also models practical scenarios like a camera with a damaged set of CCDs, or a speckle in the camera lens or on the film stock. Notice also that the camera is in motion throughout the sequence. The moving person has been successfully inpainted and the filled-in background is consistent along the sequence, thanks in part to the mosaic filling process. Fig. 10 shows another real-life video sequence where a moving person is occluded by a stationary pole, which also occludes considerable amount of the static background in all the frames. Notice that the camera does not move exactly parallel to the plane of projection while tracking the person of interest, which shows that our method is not very sensitive to mild relaxations of the assumptions stated in Section II-A. We have successfully removed the pole and the motion of the person is seamlessly continued through the region of inpainting. Again, Fig. 7 illustrates the hole in the background due to the static occluder, which is inpainted directly on the background mosaic, as described earlier. Fig. 11 shows a challenging example where the region to inpaint \( \Omega \) is a moving object that changes shape constantly. Results in Fig. 12 show that our algorithm works well even when the captured video does not strictly adhere to the assumptions mentioned in Section II-A. The moving car moves at an angle to the plane of projection, thereby changing size. The occluder is removed and the filled-in background is consistent throughout the video, in spite of appreciable
Fig. 11. Person running in from the left (occluder) is removed and the person walking (object of interest) is completed. In the final result (d), we have used the average background (from the background mosaic), in all frames, to compensate for subtle light variations. See video at http://www.tc.umn.edu/~patw0007/video-inpainting. (a) Original sequence with a moving occluder. (b) Sequence with occluder removed. (c) The moving person is filled in. (d) The area of occlusion is completely filled in.

Fig. 12. Red car moves at an angle to the camera, thereby slightly changing size as it moves towards the right. The proposed algorithm can easily handle such deviations from the motion-constraints mentioned in Section II-A. See video at http://www.tc.umn.edu/~patw0007/video-inpainting. (a) Original sequence with a moving occluder. (b) Moving car is inpainted. (c) Background is also filled in.

hand-held camera motion and small parallax. Figs. 13 and 14 show a comparison between the proposed approach and results shown in [24]. It should be observed that our technique compares favorably even in the presence of the moderate dynamic background in Fig. 14, though our algorithm was not designed to specifically address dynamic background.\footnote{Since the background is dynamic, our simple segmentation technique did not always give us the correct boundaries for the moving person in this video. We had to manually prune the $M_d$ masks in such cases.}

Fig. 13. Comparison of results with video from [24]. Jumping girl moving from left to right is occluded by another person. The proposed technique inpaints the occluder and fills-in the background without the oversmoothing observed in [24]. (a) Frames from original sequence. (b) Results from [24]. (c) Results using the proposed approach. (d) Details: result from (left) [24], (right) proposed approach.

Fig. 14. Comparison of results with video from [24]. Lady moving towards left is occluded. Proposed approach inpaints the occluder and does a good job of filling in the moderate dynamic background. Also note the better performance of our technique in restoring a small moving object such as the hat in the woman’s hand. (a) Frames from original sequence, with dynamic background. (b) Results from [24]. (c) Results using the proposed approach. (d) Detail of second frame: result from [24] (left), proposed approach (right).
by incorporating optical-flow in the SSD computation for synthesizing background also. Also note the better performance of our technique in restoring small moving objects such as the hat in the woman’s hand, or her left leg. The inpainted region in Fig. 13 is sharp and no oversmoothing is observed.

The complete video inpainting algorithm was implemented using C++, on a P-4 machine, with run-times of about 15 min (including preprocessing) for sequences with 50 frames at 320×240 resolution (with experimental nonoptimized code). The table below gives more details about the accompanying result videos.

### VI. CONCLUDING REMARKS

We have presented a simple framework for filling in video sequences in the presence of camera motion. The technique is based on combining motion based inpainting with spatial inpainting, using three image mosaics that allow us to deal with camera motion and speed up the process as well. If there are moving objects to be restored, they are filled in first, independently of the changing background from one frame to another. Then the background is filled in by extending spatial texture synthesis techniques to the spatiotemporal domain.

Currently, we are working on removing the assumptions stated in Section II-A, to be able to deal with arbitrary camera motion (including zooms), changes of scale in the moving objects, and dynamic backgrounds. Currently our algorithm does not address complete occlusion of the moving object as in [15]. We are working towards adapting our technique to such scenarios. Also to be addressed are the automated selection of parameters (such as patch size, mosaic size, etc.), and dealing with illumination changes along the sequence. Results towards adapting to illumination changes have recently appeared as an extension of [15], see [14].

### ACKNOWLEDGMENT

The authors would like to thank Dr. E. Shechtman and Prof. M. Irani for permitting the use of their original videos in [24] to generate the comparative results in Figs. 13 and 14. The presentation of this paper has tremendously benefited from the critical comments of the anonymous reviewers.

### REFERENCES


Guillermo Sapiro (SM’95) was born in Montevideo, Uruguay, on April 3, 1966. He received the B.Sc. (summa cum laude), M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, The Technion—Israel Institute of Technology, Haifa, in 1989, 1991, and 1993, respectively.

After postdoctoral research at the Massachusetts Institute of Technology, Cambridge, he became a Member of the Technical Staff at the research facilities of HP Labs, Palo Alto, CA. He is currently with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, where he holds the position of Distinguished McKnight University Professor. He works on differential geometry and geometric partial differential equations, both in theory and applications in computer vision, computer graphics, medical imaging, and image analysis. He has authored and co-authored numerous papers in image analysis and has written a book published by Cambridge University Press in January 2001.

Dr. Sapiro is a member of SIAM. He co-edited a special issue of the IEEE TRANSACTIONS ON IMAGE PROCESSING and the Journal of Visual Communication and Image Representation. He was awarded the Gutwirth Scholarship for Special Excellence in Graduate Studies in 1991, the Ollendorff Fellowship for Excellence in Vision and Image Understanding Work in 1992, the Rothschild Fellowship for Postdoctoral Studies in 1993, the Office of Naval Research Young Investigator Award in 1998, the Presidential Early Career Awards for Scientist and Engineers (PECASE) in 1998, and the National Science Foundation Career Award in 1999.

Marcelo Bertalmio was born in Montevideo, Uruguay, in 1972. He received the B.Sc. and M.Sc. degrees in electrical engineering from the Universidad de la República, Uruguay, in 1996 and 1998, respectively, and the Ph.D. degree from the University of Minnesota, Minneapolis, in 2001. He is currently an Associate Professor at Universidad Pompeu Fabra, Spain.

Dr. Bertalmio received the Femlab Prize 2002 for research done as a student in scientific fields related to Prof. S. Osher’s contributions, the Siemens “Excellent Student Paper Award” at the 2001 IEEE Workshop on variational and level set methods in computer vision, the 2001 Programa Ramón y Cajal by the Spanish Ministry of Science, and several fellowships and scholarships in the U.S. and Uruguay.