Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Treitlstr. 3/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: rmegret@ens-lyon.fr
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-060 October 18, 1999

Building the background mosaic of an image sequence

Rémi Mégret, Caterina Saraceno

Abstract

Images sequences present a high degree of redundancy because objects are repeated over the successive images. When their apparent displacements are well approximated by a simple parametric model, the whole sequence can be summed up by pasting together all the images onto a so called mosaic image. Then each image of the original sequence can be considered as a part of the final mosaic. When the displacements of distinct objects differ, we must choose which objects have to be represented and how.

In this report a framework is presented that produces the mosaic image corresponding to the background object of an image sequence. It is based on the dominant motion assumption, that is the background motion is parametric and the background occupates the main part of the images. The foreground objects are localised by their different motion. This localisation is computed together with the background motion in an iterative method. The regions corresponding to the background are then pasted onto the mosaic image using classic methods adapted to our problem or a new pasting method based on the distance to the foreground objects that achieve clearer mosaics.

Contents

List of Figures

1 Introduction

In this paper, we propose a framework to build a panoramic view representing the background of a given image sequence, by discarding the foreground objects. The segmentation foreground/background is based on motion estimation, and computed in relation to the alignment of images, which uses a model for the background motion. A new method is exposed that combines the source images into a clear mosaic in presence of foreground objects, and under the dominant motion assumption.

Image mosaicing and motion segmentation Video sequences generally present a high temporal redundancy, because the background and the foreground objects are repeated over the consecutive images. The mosaicing technique allows to produce a single image that represents a whole shot, by eliminating this temporal redundancy. The general structure of such algorithms is the following [IAH95]: first, the images are aligned using a parametric motion model, then they are pasted together to produce the mosaic image.

Alignement of an image pair consists in finding global parametric transformation that maps one of the images onto the other. We can distinguish two kind of methods.

The first ones align the whole images by minimizing an error function associated with the parametric model. When multiple coherent motions are present in the sequence this framework can only be used to find the dominant motion in a given region of interest. In [BAHH92, PH97, RPFRA98] the presence of multiple motions is ignored; a multiresolution framework [BAHH92] and the dominant motion assumption are needed to obtain the convergence to a real motion. In [IRP92] this assumption is maintained and the detection of outlier objects is introduced to avoid taking them into account, thereby reducing their influence on the computed motion.

Another kind of methods are based on the preliminary computation of a local motion field. Regression techniques are then applied to these data to segment images into regions with coherent motions, and evaluate their motion. A clustering into two regions (foreground/background) is exposed in [MBK98, DML95], while the number of regions is not defined a-priori for the layered representation in [WA94a, WA94b].

Whatever the alignment method for image pairs may be, global alignment frameworks [SHR98, Dav98] may improve the final precision by decreasing the accumulation of small errors.

When images are aligned, they can be pasted onto the mosaic space, that is each pixel value in the mosaic space is determined by combining the values of the corresponding pixels from the aligned images.

When object elimination is not an issue, values are averaged or a median is computed [IAH95]. Clearer mosaics can be obtained by segmenting the mosaic image into regions and copying all the pixels in one region from the same source image; the risk is to produce mosaics with discontinuities at the boudaries of the regions. The segmentation may be guided by global motion, as in the striping technique [PH97, RPFRA98], or computed from local displacements as in [Dav98].

Apart from the combining method object elimination was introduced in [WA94b] with the layered representation. In each source image a mask selects the pixels associated with

a given layer. The layer mosaic is then produced by combinating pixels in the masks, thereby discarding objects situated outside the masks.

Assumed background properties The background is expected to have the three following properties. *It is situated behind the rest of the scene*, so there may be parts occluded by the other objects. Appearance remains *constant over the time*, the only changes in the gray levels are due to global motion. Background pixels *occupate the main part of the image*. This property is not always true, since a big object can go close past the camera and occlude most part of the background. Nevertheless, the reconstruction of the background of a video sequence makes sense only if it is not too much occluded by other objects. So there should be images in the sequence in which the background occupates a main part.

To sum up, the background is a *big* object *at the back* of the scene, following the movement of the camera, and with eventually other objects in front of it. These properties will be taken into account for the choice of the used methods.

Proposed method The method we implemented is based on motion segmentation. Objects with a motion different from the background motion are not taken into account for the composition of the mosaic. The problem is to locate the background and keep it aligned within the sequence, then to combine its different views into a mosaic image.

In section 2 we present an overview of the whole mosaicing framework, that detects the foreground objects and eliminate them. This framework mainly consists in two parts: analysis of image sequence, and then synthesis of the mosaic image. The analysis part is described in detail in section 3, where the alignent and motion segmentation algorithms are presented, as well as how we combine them to achieve a robust alignment and the localisation of background. The construction of the mosaic given a set of aligned images and their associated background mask is tackled in section 3. All across the article, we will also present results obtained with our implementation, which is described in annex D.

2 General framework

The basic structure of the algorithm is derived from the classical mosaicing algorithm [IAH95], by introducing detection and removing of foreground objects.

We replace the classical alignment step by a module that aligns images and detects foreground pixels using an iterative framework described in this section. Background is supposed to be the main object in the scene, so that the dominant motion corresponds to the background motion. Then the segmentation of foreground objects is based on motion: pixels associated with a local motion that is different from the dominant motion constitute the foreground mask.

The pasting step takes one more parameter: the foreground masks associated with each images, and that are computed in the alignment/segmentation step.

2.1 Structure

The video flow is first analysed. This involves registration, but also segmentation to detect moving objects. Using the motion parameters and background masks, the mosaic view is then synthesized from the input images in the pasting module. Figure 1 shows a chart flow of the program structure.

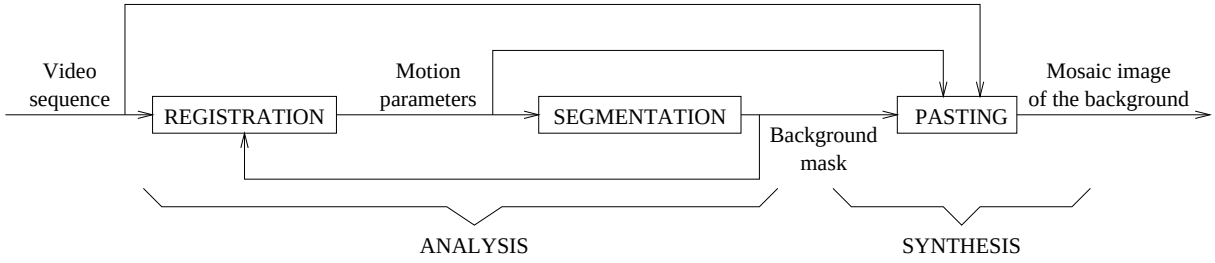


Figure 1: Chart flow of the mosaicing framework

2.2 Registration and Segmentation

Finding the background motion (*registration*) and its location (*segmentation*) are two related problems. Indeed if the general motion is known, we can compute a more precise local motion on aligned images. Reversely, the motion computation can be biased if the parameters are evaluated on an image that contains no coherent motion or more than one.

These two tasks (registration and segmentation) constitute an independent module that takes the video flow in input and gives the background motion and location. Which task is processed first is a conception choice. Because of the expected properties of the background, we decided to use a dominant motion approach (see section 3). That is, the background motion is computed. The segmentation is then performed based on motion information. The used method is explained in section 3.

2.3 Pasting

Once the registration is performed, and that we have the background mask for each image, we can past the images to the mosaic. First, from motion parameters between image pairs, global warping parameters are computed, to warp each source image onto the mosaic. Using these parameters images are aligned towards the mosaic coordinates. Then the values of the mosaic pixels are computed by combining the associated pixels from the original images. Only pixels which were classified as background by the segmentation are used. Classical pasting techniques that do not handle moving objects will be presented, as well as proposed extensions, in section 4.

3 Registration and Segmentation

In this section we will precise the base framework used to align an image pair images, while detecting the moving objects (section 3.1). We will then explain more particularly the alignment, and the motion segmentation we used (sections 3.2 and 3.3). Finally the method used to produce a global alignment out of pairwise alignment is exposed (section 3.4).

Related existing methods To handle the registration of background motion and the segmentation of foreground objects, several approaches can be used.

In [WA94b] a local motion is computed for each image directly from the video sequence with an optical flow. Then the image is partitioned into regions that have a coherent local motion with an iterative refining framework. Optical flow is also used in [MDK95], to directly partition the image into two regions, using a clustering technique. The bigger region is assimilated to the background, and its motion can be computed precisely using the region mask, and a parametric model.

In [IRP92] the *dominant motion* of the image pair is first evaluated to find a rough estimation of the background motion. Then a background mask is computed, by segmenting the aligned images on local motion intensity. The first motion estimation may be modified by some moving objects. For this reason a refinement is performed by computing extra aligment and segmentation, where each registration is computed only for the previously segmented region.

Our approach We chose to use a dominant motion approach with an associated segmentation as introduced in [IRP92] for the registration of images. Local motion estimation is computed only on aligned images, which increase their accuracy. The background is expected to occupy the main part of the image, so the dominant motion effectively represents its motion. As the background is supposed to be static in the real world its apparent motion is just the effect of the camera movement. Furthermore it is the furthest object in the image, so the image need only a foreground/background segmentation, and a single motion model can represent the whole background motion, provided it is of a sufficient order.

3.1 Iterative framework

The dominant motion approach is not biased by foreground objects for translation, by using the hierarchical method exposed in [BAHH92]. But it is less robust as soon as higher order models (affine, planar) are used. We used an adaptation of the framework proposed by [IRP92] to overcome this problem.

The background motion and localisation are computed in an iterative framework involving two submodules: registration and segmentation.

1. Initialize the background mask to be the whole image

2. Repeat steps 3.,4.,5.,6. several times to converge towards the motion and localisation of the background. For each motion model, the first pass gives a rough result for the parameters, and the second step affines it using the segmentation mask.

3. Find the dominant motion between the two original images using the background mask. Only pixels belonging to this mask are taken into account.

4. Align the two images using the parameters computed in step 3.

5. Segment out pixels with a local motion over some threshold in images from step 4. This threshold can be fixed (typically 1.0 pixel), or decrease at each step to take into account the inaccurate approximation due to too low order models in the first steps.

6. Set the background mask from the result of step 5.

The inner structure of the registration and segmentation is not really important. But the registration shouldn't be too perturbated by the presence of objects moving differently from the background.

The model in step 3. changes during the iterations. For the first passage in the loop, a dominant translation is computed. The mask resulting of the segmentation then contains only pixels with the same motion. In the following passages higher order models can be used, since they are computed only on pixels belonging to the background region.

3.2 Pairwise alignment

Given two images I_1 and I_2, we would like to find a parametric transformation such that warped I_1 is the closest to I_2. With each pixel $\vec{x}$ in the second image coordinates we associate a *motion* vector $\vec{u}(\vec{x})$ that represents the displacement of pixel $\vec{x}$ by the transformation.

A parametric model is chosen to represent the whole motion with a small number of parameters $\vec{p}$. The registration of an image pair consists in finding the parameters that minimizes the difference between the first image warped using these parameters and the second image. Translational, affine and planar motion models are generally used [IAH95] (see tab.1). They are sufficient to approximate the motion of objects when parallax is small. This hypothesis is supported by the background property to be the furthest object from the camera.

We use a least-square error as a measure for the difference between aligned images. Given a parameters vector p, this error can be computed for support regions $\mathcal{R}_1$ and $\mathcal{R}_2$ (resp. in first and second image) as follows:

$$E(\vec{u_p}) = \sum_{\vec{x}} \{I_2(\vec{x}) - I_1(\vec{x} - \vec{u}_p(\vec{x}))\}^2 \tag{1}$$

where $\vec{x}$ and its corresponding pixel in image 1 are inside the support regions: $\vec{x} \in \mathcal{R}_2$ and $\vec{x} - \vec{u}_p(\vec{x}) \in \mathcal{R}_1$.

model	motion form
translational	$\vec{u}(x,y) = \begin{pmatrix} a \\ d \end{pmatrix}$
affine	$\vec{u}(x,y) = \begin{pmatrix} a+bx+cy \\ d+ex+fy \end{pmatrix}$
planar	$\vec{u}(x,y) = \begin{pmatrix} a+bx+cy+gx^2+hxy \\ d+ex+fy+gxy+hy^2 \end{pmatrix}$

Table 1: Expression of parametric motion models

E is minimum when the images are well aligned. Registering the images according to the motion model given by $\vec{u}_p$ consists in minimizing E with respect to the parameters $\vec{p}$. This is a non-linear problem.

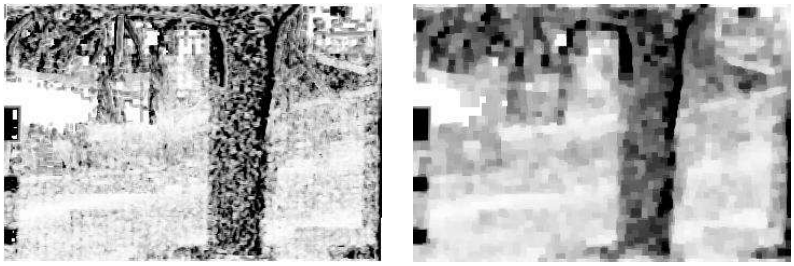
In [BAHH92], a framework is presented to solve this problem, using multi-resolution pyramids within an iterative gradient-based refining loop. This method was implemented. See annex A for its description. The advantages of such a framework is to be relative robust to local minima, allow large displacements, and although give precise results at subpixel scale. It is less robust when the number of parameters for the parametric motion increase.

3.3 Segmentation

Given images pairs aligned with respect to the background motion, our purpose in segmentation is to classify pixels into moving and stationary ones. This can be done using the framework of appendix C, where the normal-flow intensity is computed using a hierarchical gradient based approach. In our implementation we used a 3×3 neighborhood around each point to compute spatial gradients.

The implementation chosen has the advantage of being computationaly efficient, and to involve only local computations. However it is not very robust to changes in light, and is not a real motion intensity. Indeed, as the computation is local and because of the *aperture problem*, only the normal component of the local gradient is computed. As a consequence the motion intensity field obtained with this method is not very smooth, and depends on the direction of spatial gradient. To correct this, we post-process the motion using morphological operators. We used morphological opening [aS94] and closing both to eliminate small artifacts, and to smooth the values. An example of such a postprocessing of the motion values is shown in figure 2. The advantage of these operators is that they preserve the edges accuracy, as opposed to convolution methods which blur the image.

This local motion intensity is finally thresholded to produce a foreground mask. This mask can be used for the next registration, to determine on which region the global motion should be computed (see subsection 3.1), and also in the pasting step, to discard the foreground pixels (section 4).



a b

Figure 2: Postprocessing of the motion intensity using morphological opening and closing. Original motion intensity image (a), postprocessed image (b). (High motions in black, low motion in white.) The values are made more uniform, while keeping the edges sharp.

3.4 Global registration

Registration is based on pairwise alignment, but we need to know the transformation between the images and the mosaic coordinates.

Registration of original image pairs A first solution consists in registering consecutive original images by pairs, and composing the parameters to align all the images to one of them, in the way presented in figure 3.

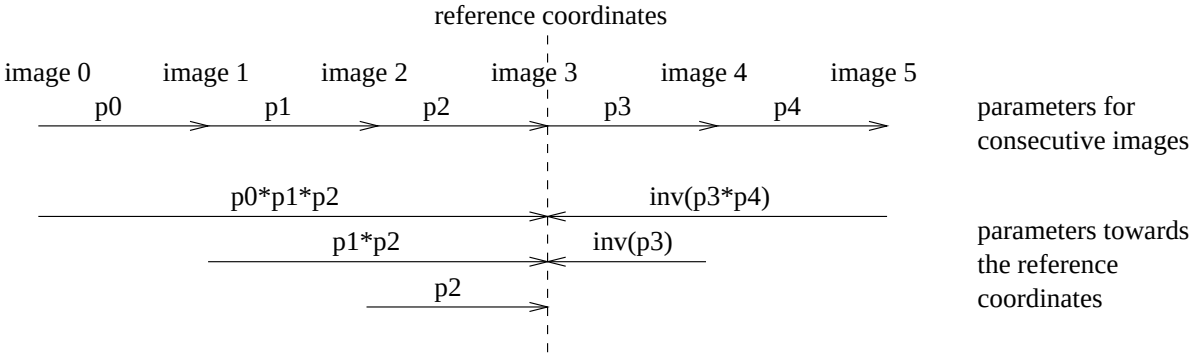


Figure 3: Global registration by composition of pairwise registrations

Appendix B describe some operations on composition of parametric motions.

Integrating reference image over time By using source images directly, foreground pixels are influencing the motion estimation. The technique can be made more robust by considering the alignment of each new source image with an integrated reference image, instead of with the previous source image.

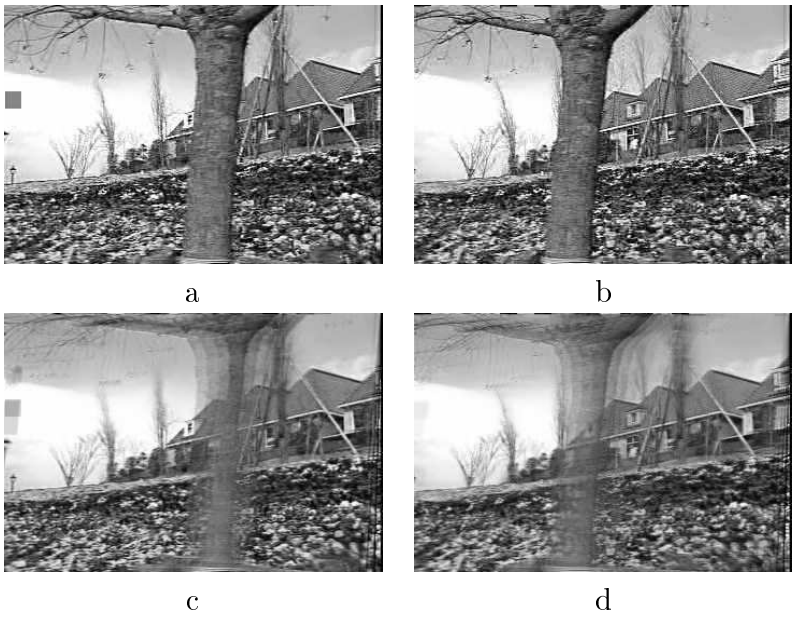
The reference image is obtained by combining the images already registered, aligned on the global motion. Moving objects then appear dissolved, so that the algorithm is less likely to align a foreground object with its blurred version from the reference image.

The method proposed by [IRP92] is to combine the new image I_{t+1} with the old reference image I_t^{ref}:

$$I_{t+1}^{ref} = \lambda W(I_t^{ref}) + (1 - \lambda) I_{t+1} \tag{2}$$

where λ is a real coefficient between 0 and 1, and W is the warping transformation to keep the reference image in the current image coordinates. When $\lambda = 0$, we come back to the case with no temporal integration.

Figure 4 shows the effect of the integration scheme on the reference image.



a b c d

Figure 4: Effect of integration on the reference image. First (a) and last (b) image of the integrating sequence. Integrated image corresponding to the third image (c), and to the seventh one (d).

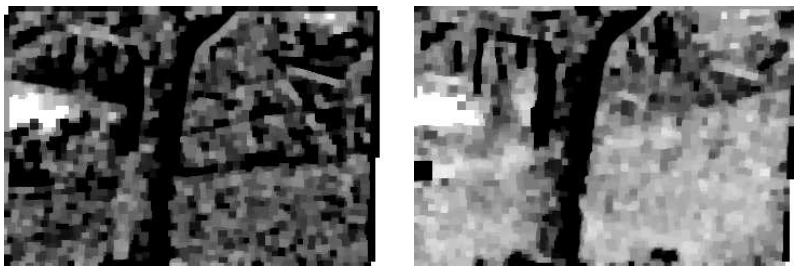
Use of the old segmentation mask as a cue In the alignment/segmentation process, each alignment is performed by taking into account only pixels belonging to the background mask. This mask is initialized to the whole image for the first step of an image pair alignment, and updated by each segmentation.

The risk is that the first steps converge towards the motion of a foreground object instead of the background. This can happen if the foreground motion is better approximated by the low order parametric model than the background. For example, if the background is rotating whereas an object is translating, then the first alignment will give the translational motion, even if the background occupates the main part of the image.

If we suppose that the moving objects are not too fast, the background mask doesn't vary much between consecutive images. We could avoid the first detection of foreground objects for each new image pair by using the background mask computed for the previous image pair. With this method:

- The convergence is faster, because the background location has just to be updated. Indeed we don't need the first pass that is supposed to give a rough idea of the parameters.
- The result is more robust to moving objects because less foreground surface influences the motion. The only foreground pixels to be taken into account for the global motion estimation belong to newly occluded regions.

The improvement of the stability is shown in figure 5 where we applied the registering framework on a sequence made of the images $3 \times i$ of the flower garden sequence. Choosing non consecutive images should increase the difficulty of finding the true global motion.



(a) without mask cue (b) with mask cue

Figure 5: Improvement of registration stability using the mask of previous segmentation as a cue for the next alignment. (see the text for experiment description) Local motion between the 5th and 6th aligned images of the sequence, without (a) and with (b) a mask cue. (White = low motion, Black = High motion)

We implemented the two methods (with and without mask cue) in order to compare them. The algorithms are exactly the same, excepted for the initialization of the alignment mask and the number of alignment/segmentation steps. In the method without cue, the mask is set to the whole image for each new image pair, whereas in the method with cue, it is copied from the background mask of the previous pair. The first alignment/segmentation step is suppressed in the cue method, as it should be replaced by the mask cue.

The first method does not use the segmentation mask for the first registration. The iteration number is set to 2: first alignment with a translation model over the whole image, then segmentation with a motion threshold of 1.0 pixel, alignment with an affine model.

The second method uses the segmentation mask of a pair for the first alignement of next pair. The translational alignement is then skipped: registration simply consist in an affine model alignment over the region determined by the mask.

The alignement of images without mask cue is wrongly executed. Indeed it takes into account for each image pair the tree for the first alignment. This is avoided with our method. In spite of the use of only one image out of three the good alignment is found.

4 Pasting

At this stage, the following data is available:

- *warping parameters* between each image and the mosaic. For each pixel $\vec{x}$ of the mosaic, we can compute the corresponding pixel $\vec{y_t} = P^t(\vec{x})$ in the image I^t.
- *background mask* for each image in the sequence. $\mathrm{bg}^t(\vec{y_t})$ is equal to 1 if $\vec{y_t}$ is a background pixel in image I^t, and to 0 otherwise.

Our purpose is to combine the image pixels to get a mosaic image without the moving objects.

In the following $M(\vec{x})$ will denote the value of the local motion intensity for the pixel $\vec{x}$ in the static mosaic, where $\vec{x}$ is expressed in the mosaic coordinates. y_t is the coordinates of the pixel corresponding to $\vec{x}$ in the image at time t, expressed in this image coordinates.

4.1 Classical solutions

Most of the mosaicing algorithms don't handle the presence of moving objects in the pasting step. They must be modified to exclude foreground objects pixels.

4.1.1 Simple combination

Basic integration The simplest method consists in combining the pixels values $I^t(\vec{y_t})$ corresponding to the mosaic pixel $\vec{x}$ with a simple combining function f, as in eq.3.

$$M(\vec{x}) = f(I^1(\vec{y_1}), \ldots I^N(\vec{y_N})) \tag{3}$$

The simple combining function f is usually a *mean* or a *median* [IAH95]. With this method, moving objects pixels are combined with background pixels. Small or fast moving objects have less pixels in temporal combination, and so their pixel values have a small influence on the mosaic pixel values. With a mean, big objects are blurred, and small objects leave a ghost like track. When using a median, there is less blur, and small moving objects can be totally eliminated if there are enough images in which the corresponding region shows the background. However, unless they are very small and moving fast, the foreground objects are still present in the mosaic.

Extension to exclude foreground pixels This method can be naturally extended to take into account the background mask $bg(\vec{y})$, by combining only pixels corresponding to the background.

$$M(\vec{x}) = g\left(\left(I^1(\vec{y_1}), bg^1(\vec{y_1})\right), \ldots \left(I^N(\vec{y_N}), bg^N(\vec{y_N})\right)\right) \tag{4}$$

where the extended combining function g can be expressed as function of a simple function f:

$$g\left((v_t, bg_t)_t\right) = f\left((v_t)_{\{t|bg_t=1\}}\right) \tag{5}$$

Figure 6 shows the mosaics obtained with the basic integration on the first 30 images of the flower garden sequence, using a median combiner. Image (a) is a simple combination of original images. The tree leaves a blurring mark, where it was in the original images. On image (b), motion information was used to segment out the foreground (the tree), so that only the background is present.

4.1.2 Partitioning

To avoid the blurring caused by the combination of many pixel values, a solution is to choose for each pixel only one source images. The mosaic image is partitioned into regions, and each region is associated with one source image.

The combining process is then expressed by:

$$\begin{aligned} M(\vec{x}) &= p_{s(\vec{x})}\left(I^1(\vec{y}_1), \ldots I^N(\vec{y}_N)\right) \\ &= I^{s(\vec{x})}\left(\overrightarrow{y_{s(\vec{x})}}\right) \end{aligned} \tag{6}$$

where

- $s(\vec{x})$ is the number of the source image from which the image mosaic pixel $\vec{x}$ is copied.
- p is a projection function indexed by s: $p_s(v_1 \ldots v_N) = v_s$.

The region partitioning is expressed entirely by the index $s(\vec{x})$.

An usual way to partition the mosaic is to associate each pixel with the closest center image, using a *Voronoi tesselation* like in [PH97]. The mosaic image is then partitioned into *stripes*, that is regions orthogonal to the general motion. The moving objects, are not blurred, but they may be sliced and have non coherent appearance on the strips boundaries, because of their displacement. A possibility to exclude foreground objects consists in choosing for each pixel the image whose center is the closest among those where the corresponding pixel is in the background mask.

In [Dav98], an original method is proposed to choose the strips according to the presence of moving objects. The region are computed to minimize the presence of motion on their boundaries. Moving objects are copied from a single source image, thus avoiding blurring, and incoherent stripes boundaries. But foreground objects are still in the mosaic, whereas we want to segment them out and reconstruct the background behind them. Such a method could be used as a last stage of our pasting step, to lower uncoherences in the background mosaic, when the parametric model does not describe the background motion precisely enough.

4.2 Extended formulation

We propose here a general framework that should englobe the previous extended classic combination methods (simple combination and Voronoi tessellation, excluding foreground pixels), and that will allow us to better control the pasting of the images.

a

b

Figure 6: Mosaic of the flower garden sequence using median combining. Simple combination (a). Combination of pixels belonging to the background masks (b). The black regions on the sides and in the tree's branches don't belong to the background.

4.2.1 General expression

Each pixel $\vec{y}$ in each source image I^t is associated a positive confidence coefficient $C^t(\vec{y})$ that tells in which measure the pixel should appear in the final mosaic. During combination more confident pixels will be given a more important influence.

The combination process is only based on the confidence of corresponding pixels, as expressed in eq. 7.

$$M(\vec{x}) = g\left(\left(I^1(\vec{y}_1), C^1(\vec{y}_1)\right), \ldots \left(I^N(\vec{y}_N), C^N(\vec{y}_N)\right)\right) \tag{7}$$

where

- $C^t(\vec{y}_t)$ is the confidence coefficient of pixel $\vec{y}_t$ in image at time t
- g is a general combination function, such that pixels with a null confidence ($C^t(\vec{y}_t) = 0$) don't influence the mosaic pixel value $M(\vec{x})$.

We can control two parameters: the combination function g, and the definition of the confidence coefficients $C^t(\vec{y})$.

4.2.2 Classical methods as particular cases

This general method can handle the two more particular techniques described in section 4.1 by properly choosing g and C.

Simple combination Eq. 7 is quite similar to eq. 4. A simple combination of pixels is then done by taking the same g function, and a confidence equal to 1 for background pixels and 0 for foreground pixels.

Voronoi striping Lets take for confidence a function null for foreground pixels and decreasing with respect to the distance to the center of the image for background pixels. For example:

$$C^t(\vec{y}) = \begin{cases} 0 & \text{if } \vec{y} \text{ is outside the image or} \\ & \text{in the foreground} \\ (1 + d_t(\vec{y}))^{-1} \quad \text{or} \quad \exp(-d_t(\vec{y})/\sigma) & \text{else} \end{cases}$$

where $d_t(\vec{y})$ is the distance between pixel $\vec{y}$ and the center of the image t. Such a confidence image is shown in figure 7b.

Let be g the function selecting the pixel value with the highest confidence.

Then applying the general framework with these functions leads to a Voronoi tessellation, where each mosaic pixel is copied from the corresponding pixel in the image with the closest center.

4.2.3 Framework possibilities

Choosing g and C lead to a particular behaviour of the framework.

Combination function g There are roughly two kinds of combination functions: weighting, and selecting.

Weighted mean, or *weighted median* are of the first kind. All the pixels with a non null confidence are used, but their influence is weighted by the confidence coefficient.

Selecting functions pick among the most reliable pixels to give a value for their combination. Choosing the *highest confidence pixel* will lead to a partition. By using a *combination of a few most reliable pixels*, transitions are smoother.

Confidence coefficient $C(\vec{y})$ This coefficient expresses the likelihood that a pixel should appear in the mosaic. It can be computed from different criteria, to compensate some errors:

- segmentation errors: the background mask may contain some foreground pixels, because of incertainty in segmentation. By defining the confidence as increasing with the distance from the foreground mask, we lower the participation of such pixels.
- optical deformations on borders: the images are more deformed far from their centers. By setting high confidence near the center and dicreasing with the distance, we give the priority to the central regions, which are less deformed.

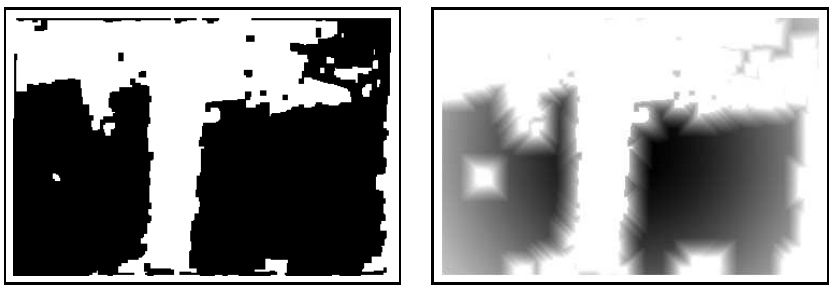
Combinations of those parameters are also possible. Indeed an extension of the Voronoi striping with a better compensation of segmentation errors could be defined, by adding the confidence related to segmentation, and the inverted distance to center of image:

$$C(\vec{y}) = \begin{cases} 0 & \text{if } C_{\text{bg}}(\vec{y}) = 0 \\ C_{\text{center}}(\vec{y}) + \lambda C_{\text{bg}}(\vec{y}) & \text{else} \end{cases}$$

where C_{center} is high for pixels close to the center of images, and $C_{\text{bg}}(\vec{y})$ is low for pixels too close from the foreground objects. λ is intended to control the relative influence of each parameter. With this confidence coefficient, the framework would create the mosaic by giving the priority to pixels close to the center, but not close to foreground objects.

Figure 7 shows the foreground mask of an image from the flowerbed sequence, and the associated confidence image

The confidence of figure 7b was used to produce the mosaic in figure 8. We can compare it with the mosaic obtained by a masked median combination (fig. 6). The advantage of striping is to avoid blurring, because on one region, all pixels come from the same source image, which is not the case with a median. The problem is to define the stripes so that there is no discontinuity at their boundaries. In our implementation there is no explicit treatment to avoid this. But by defining a confidence depending on the distance to the center of the image, pixels that are close on the mosaic come from source images that are close in time, and that are well aligned. This was not the case with the classical median technique.



a b

Figure 7: Confidence images for the 14^{th} image of the flower garden sequence using different rules. White is low confidence, and black a high one. Background mask (a) can be considered as a confidence image for simple combination (median). Confidence based on distance to the foreground objects and to the center on the image (b)

Figure 8: Mosaic obtained using a striping method, with a confidence based on distance to the foreground objects and to the center on the image. The black regions on the sides and in the tree's branches don't belong to the background.

5 Conclusion

In this report a framework was proposed to produce a background mosaic from a video sequence, and detailed algorithms were described to implement it. Our method involves two steps: 1) alignment of images and localisation of foreground objects, 2) pasting of the images onto the mosaic.

The structure of the first step relies on the assumption that the background is the dominant object in the source image , so that a dominant approach can be used (see section 3). Localisation of foreground objects is based on local motion intensity between aligned images. To improve accuracy and robustness, these two modules are run in an iterative refining process where alignment is computed on the segmented region, and segmentation use aligned images.

Concerning the pasting step (section 4), we reviewed two classical methods that do not take into account the presence of moving objects, and extended them to eliminate foreground objects. A new framework was proposed to achieve clearer mosaics, using a striping depending on the distance to foreground objects.

The whole framework was implemented and tested on the well-known flower-garden sequence. The proposed pasting method revealed to achieve a clearer mosaic than usual methods. The limitations of the whole framework come from the algorithm we chose for object detection (namely the background should be the main object in the sequence, and have a motion that is approximated by a parametric motion). This could be improved by using more accurate object detection.

Applications of such an algorithm can be found in video indexation, where a whole sequence can be summed up by a single mosaic image that represents its background.

References

[aS94] M. Pardàs and P. Salembier. Three-dimensionnnal morphological segmentation and motion estimation for image sequences. In *EURASIP Signal Processing*, volume 38(2), pages 31–33, September 1994.

[BAHH92] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *European Conference on Computer Vision*, pages 282–287, 1992.

[Dav98] J. Davis. Mosaics of scenes with moving objects. In *Proceedings of CVPR*, June 1998.

[DML95] F. Dufaux, F. Moscheni, and A. Lippman. Spatio-temporal segmentation based on motion and static segmentation. In *IEEE COnference on Image Processing*, volume 1, pages 306–309, Washington DS USA, Oct 1995.

[IAH95] M. Irani, P. Anandan, and S. Hsu. Mosaic based representation of video sequences and their applications. In *Fifth International Conference on Computer Vision*, pages 605–611, June 1995.

[IRP92] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *European Conference on Computer Vision*, pages 282–287, 1992.

[IRP94] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, Feb 1994.

[MBK98] F. Moscheni, S. Bhattacharjee, and M. Kunt. Spatio-temporal segmentation based on region merging. *IEEE-PAMI*, September 1998.

[MDK95] F. Moscheni, F. Dufaux, and M. Kunt. A new two-stage global/local motion estimation based on a background/foreground segmentation. In *Proceedings of the IEEE International Conference on Acoustic Speech and SIgnal Processing*, volume 4, pages 2261–2264, Detroit USA, May 1995.

[PH97] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *CVPR*, pages 338–343, June 1997.

[Ros84] A. Rosenfeld. *Multiresolution image processing and analysis*. Springer-Verlag, 1984.

[RPFRA98] B. Rousso, S. Peleg, I. Finci, and A. Rav-Acha. Universal mosaicing using pipe projection. In *International Conference on Computer Vision*, 1998.

[SHR98] H. S. Sawhneg, S. Hsu, and R.Kumar. Robust video mosaicing through topology inference and local to global alignment. In *European Conference on Computer Vision*, pages 103–119, 1998.

[WA94a] J. Wang and E. Adelson. Spatio-temporal segmenation of video data. In *Proceedings of the SPIE: Image and Video Processing II*, volume 2182, San Jose, February 1994.

[WA94b] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing Special Issue: Sequence Compression*, 3(5):625–638, September 1994.

A Hierarchical dominant motion computation

In this section, a particular solution to the registration problem of section 3 is exposed. It is based [BAHH92].

A.1 Linear approximation

A standard Gauss-Newton method, to solve such minimizations problems is to approximate the quantity under brackets to the first-order:

$$E_i(\vec{u}) = \sum_{\vec{x}} \left\{ \Delta I(\vec{x}, \vec{u}_i) + \vec{\nabla} I(\vec{x}) \cdot (\vec{u} - \vec{u_i}) \right\}^2 \tag{8}$$

where

$$\Delta I(\vec{x}, \vec{u}_i) = I(\vec{x}, t) - I(\vec{x} - \vec{u_i}(\vec{x}), t-1)$$

$$\vec{\nabla} I(\vec{x}) = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$$

Then the solution is found iteratively:

$\overrightarrow{u_0}$	is an initial guess for $\vec{u}$
$\overrightarrow{u_{i+1}}$	minimizes E_i

A.2 Multi-resolution framework

[BAHH92] proposed a hierarchical approach to avoid local minima and to provide an efficient convergence. A Gaussian or Laplacian pyramid is first computed ([Ros84]). Then parameters are computed sequentially for each level, in a coarse to fine way. The parameters found for one level i are used as a starting point to find the parameters at level $i-1$ (see fig. 9).

The pyramidal approach makes the iterative approach efficient and robust to noise and local minima. On coarser levels, the motion is at the scale of some pixels, so a rough approximation of the motion is quickly found. At each level the parameters become more precise. On the finest level, the parameters are finally more accurate than the pixel.

A.3 Realization

We have implemented the registration for translation motion and affine motion using the framework previously exposed.

Resolution of registration equations Each motion model leads to different equations. The affine motion englobes the translation motion, but it is more sensitive to noise. The translation model has less parameters and is more robust.

- For translation, the motion model is uniform:

$$\vec{u}(\vec{x}, \vec{p}) = \vec{p} = \begin{pmatrix} a \\ d \end{pmatrix}$$

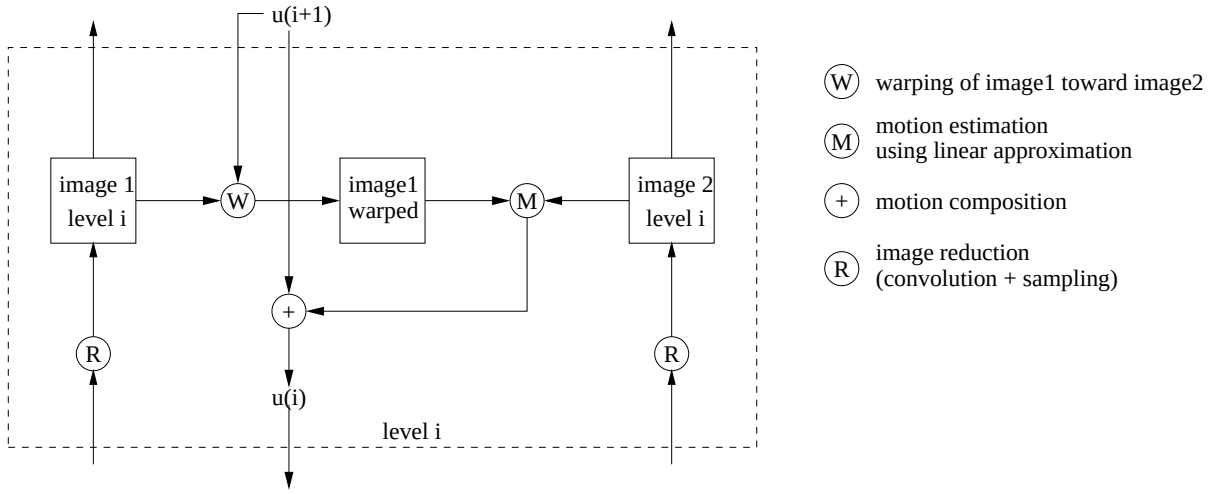


Figure 9: Multi-resolution framework for registration

Equation 8 simplifies itself into

$$E_i(\vec{p}) = \sum_{\vec{x}} \left\{ \Delta I(\vec{x}, \vec{u}_i) + \left[\frac{\partial I}{\partial x} \; \frac{\partial I}{\partial y} \right] \vec{p} \right\}^2 \tag{9}$$

Finding its minimum is equivalent to solving a 2×2 linear system.

- With affine motion, the motion field is:

$$\vec{u}(\vec{x}, \vec{p}) = \mathbf{X}\vec{p} = \begin{pmatrix} a + bx + cy \\ d + ex + fy \end{pmatrix}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \quad \text{and} \quad \vec{x} = \begin{pmatrix} x \\ y \end{pmatrix}$$

and

$$\vec{p} = (a \; b \; c \; d \; e \; f)^T$$

Then Equation 8 can be simplified into:

$$E_i(\vec{p}) = \sum_{\vec{x}} \left\{ \Delta I(\vec{x}, \vec{u}_i) + \left[\frac{\partial I}{\partial x} \; \frac{\partial I}{\partial y} \right] \mathbf{X}\vec{p} \right\}^2 \tag{10}$$

Its minimum is solution of a 6×6 linear system.

In both cases, the system is solved by a Gauss-pivot method.

B Manipulation of parametric motion fields

Motion fields used in the registration step should be composed and inverted. If we know the transformation from I_1 to I_2 and the transformation from I_2 to I_3, then the transformation from I_1 to I_3 can be obtained by composing the two previous ones. Also, when the transformation from I_1 to I_2 is known, I_2 can be warped towards I_1 using the inverse transformation.

It is straightforward for translations, and also possible for the affine motion model when considering motions due to camera movement. Affine model has a good property: composing and inverting affine fields leads to affine fields.

B.1 Composition of affine motion fields

Let us note p_1 the parameters of the affine transformation from I_1 to I_2 and p_2 those between I_2 and I_3. Then the transformation from I_1 to I_3 is affine, with parameters p such that (with the notations of eq.1):

$$\left\{\begin{array}{lcllll} a & = & a_1 & +a_2 & +b_1a_2 & +c_1d_2 \\ b & = & b_1 & +b_2 & +b_1b_2 & +c_1e_2 \\ c & = & c_1 & +c_2 & +b_1c_2 & +c_1f_2 \\ d & = & d_1 & +d_2 & +e_1b_2 & +f_1a_2 \\ e & = & e_1 & +e_2 & +e_1e_2 & +f_1b_2 \\ f & = & f_1 & +f_2 & +e_1f_2 & +f_1c_2 \end{array}\right. \tag{11}$$

B.2 Inversion of affine motion fields

Let us note p the parameters of the affine transformation from I_1 to I_2. Then the transformation from I_2 to I_1 is affine, with parameters p'. p' is the solution of the following equation:

$$\left[\begin{array}{ccc} 1 & 0 & 0 \\ a' & b'+1 & c' \\ d' & e' & f'+1 \end{array}\right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ a & b+1 & c \\ d & e & f+1 \end{array}\right]^{-1} \tag{12}$$

C Local motion intensity computation

C.1 Motion evaluation

As pointed out in [IRP94], a motion intensity can be computed out of the spatial and temporal gradients.

$$M(\vec{x},t) = \frac{\sum_{\vec{x}\in N(\vec{x}_0)} |I(\vec{x},t) - I(\vec{x},t-1)|.|\vec{\nabla} I(\vec{x},t)|}{\sum_{\vec{x}\in N(\vec{x}_0)} |\vec{\nabla} I(\vec{x},t)|^2 + C} \tag{13}$$

where $N(\vec{x})$ is a neighborhood of $\vec{x}$, and C is a constant to avoid numerical instabilities.

Because it is gradient based, this measure is valid only for motions at the scale of the pixel.

C.2 Reliability coefficient

For each pixel, a reliability coefficient is computed too. This coefficient is the inverse of the condition number of the matrix in the optical-flow equation 14, indicating its numerical stability.

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix} \tag{14}$$

where I_x (resp. I_y and I_t) represents the partial derivative of I with respect to x (resp. y and t). The sums are computed over a neighborhood of $\vec{x}$.

Noting λ_{min} and λ_{max} the largest and smallest eigenvalues of the matrix, the reliability coefficient is:

$$R(\vec{x}, t) = \frac{\lambda_{min}}{\lambda_{max}} \tag{15}$$

where:

$$\lambda = \frac{\left(\sum I_x^2\right) + \left(\sum I_y^2\right)}{2} \pm \sqrt{\left(\left(\sum I_x^2\right) - \left(\sum I_y^2\right)\right)^2 + 4\left(\sum I_x I_y\right)^2} \tag{16}$$

C.3 Hierarchical framework

To handle motions with an speed greater than one pixel a pyramidal approach is used. A Gaussian pyramid is first built[Ros84]. The gradient based motion and the reliability are computed for each level.

Starting at the coarser level, the definitive motion measure $M_f(\vec{x}, t)$ is computed as follows.

- $M_f(\vec{x})$ becomes $M(\vec{x})$ if $\begin{cases} M(\vec{x}) \text{ is high enough} \\ \text{or} \\ M(\vec{x}) \text{ is low and } R(\vec{x}) \text{ is high} \end{cases}$
- $M_f(\vec{x})$ is obtained by scaling the $M_f(\vec{y})$ corresponding in the coarser level in other cases.

D Implementation of the mosaicing framework

The framework described in this article has been implemented as a C program [1] and run over video sequences. It takes in input a sequence of gray-scale images and produces the mosaic image associated, using the methods exposed.

The following options can be set to modify the methods used in the mosaicing process.

[1] Our implementation is available on the Internet at http://www.prip.tuwien.ac.at/~megret/

Alignment/Segmentation

- Compute the dominant motion with only translationnal model, or with translationnal model, and then affine model.(see section 3.2)
- Use a background mask cue for each first alignment of an image pair.(see section 3.4)
- Number of alignment/segmentation steps, and eventually the thresholds used at each step by the motion segmentation.(see section 3.1)
- Use an integrated image as reference for alignment or not, and set the λ value (see section 3.4) that controls the importance of new images in the reference image.

Pasting

- Select the type of pasting among median (with confidence or not) or striped
- Select the type of confidence computed among background mask, Voronoi confidence or Voronoi confidence with penalty to pixels close to foreground objects (see section 4.2.3).
- Give the color images associated to the gray-level input to produce a color mosaic.

Controls

- Save alignment parameters to file, and retrieve them to skip the alignment steps.
- Save segmentation masks and parameters to file, to skip the analysis process.
- Produce a mosaic or not
- Save temporary files for each separate step