Real-Time Tracking of Video Sequences in a Panoramic View for Object-Based Video Coding

Matthijs Douze[1] and Vincent Charvillat[1]

[1]IRIT/ENSEEIHT,
2 rue Camichel,
31071 Toulouse cedex 7, France
{douze, charvi}@enseeiht.fr

Abstract. We present a real-time tracking system that uses a panoramic view of the scene as a reference. This view is captured with a specialized camera we developed. The tracking is robust, as it is not fooled by small objects that move differently from the background. Therefore, we can segment those objects, and code the different video object planes separately.

Introduction

In this article, we present a tracking technique for video frames with respect to a panorama captured by a specific camera. Within very favourable experimental conditions, we can solve in real time the tricky problems of the new geometric approaches of video coding, such as those considered in the MPEG-4 norm (dominant motion compensation, spatiotemporal object segmentation, object-based coding of video).

To do this, we implemented a sparse tracking method based on homographies.

Firstly, we present the way the data is captured, and the corresponding geometric model. Then we present our tracking method. Finally, we describe the video coding application of the method.

1 Shooting a Panorama

Realizing a panorama from a range of views has several drawbacks:

- the images are captured at different moments, so they are sensitive to environmental variations (especially illumination);
- the elementary images must be centred carefully;
- geometric aberrations are difficult to compensate.

Our panoramic camera partially addresses these drawbacks.

A continuous scan provides in one pass a panorama without any join. Joining the two sides of the panorama is not a geometric issue. The only geometric

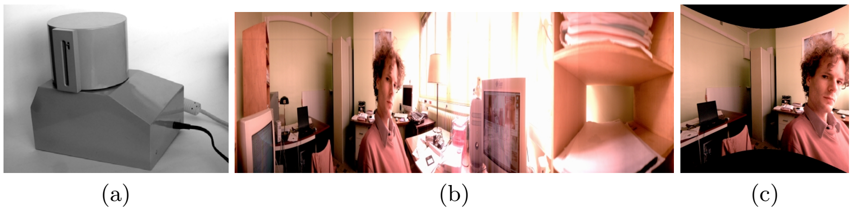

Fig. 1. Panorama (b) as seen by our camera (a), and one of the four remapped views (c).

aberrations are in the vertical direction: all non-vertical lines in the scene appear curved. This curvature is regular and continuous, it can be corrected easily on partial views.

A linear three-band CCD is mounted vertically and turns around a vertical axis, that crosses at right angles to the optical axis. The horizontal stroke covers 380°, and the optics provide a vertical view angle of 60°. The vertical resolution of the picture is that of the CCD, 2500 pixels. The horizontal resolution depends on the rotation speed. Typically, a 3500 pixel wide picture takes 70 seconds to capture.

The image produced by the panoramic camera is a cylindrical projection. However, we need a classical perspective projection to do homography-based matching. Therefore, the panorama has to be rectified to simulate this type of projection. We convert the panorama to four usable pictures, each of them covering 90° of the horizontal track (figure 1).

2 The tracking algorithm

After this, we shoot the scene with a classical video camera and we want to remap the image sequence with respect to the panorama. This is possible if the scene is motionless (this condition is relaxed in section 3.3) and if the optical centre coincides with the optical centre of the panoramic camera (in other words, the camera may only turn and zoom in and out).

2.1 Description of the problem

We can remap the transformed panorama I with a frame I_t of the video sequence using a homography. There exists $\mu = [h_1 \quad \cdots \quad h_8]^\top$ such that, for any 3D point which has projection (x, y) on I and (x', y') on I_t:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = h\left(\mu, \begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} \frac{h_1x+h_2y+h_3}{h_7x+h_8y+1} \\ \frac{h_4x+h_5y+h_6}{h_7x+h_8y+1} \end{bmatrix}$$

The parameter vector μ must be estimated for each frame of the video sequence. To solve this problem, we first investigate a classical tracking problem. Then, we will adapt the solution to our panoramic case.

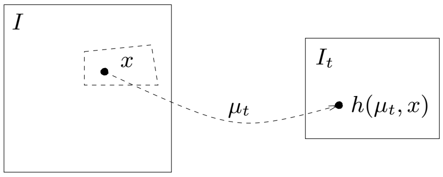

Fig. 2. Remapping of panorama I with frame I_t of the video sequence.

2.2 Point-based tracking with a homography model

The data of the problem are an image sequence $(I_t)_{t\in\mathbb{N}}$ and an interest region, J_0, on I_0. This region, quadrilateral-shaped, is inside the image of the object that has to be located in the following frames.

Modeling We suppose that the movement of the object's image can be described by a homography. That is, at frame t, the homography with parameters μ_t maps from J_0 to I_t. This model is correct, in particular, when the object is planar.

We assume that the image of a 3D point does not change colors (graylevels in our case). Then, we can say:

$$\forall x \in J_0, I_t(h(\mu_t, x)) = I_0(x)$$

where $I(x)$ is the graylevel of point x on image I. $h(\mu_0, x) = x$.

We assume also that, between the shots, the motions of the object and the camera are small. This induces that the difference $\Delta\mu_t = \mu_t - \mu_{t-1}$ is small, so that we can use μ_{t-1} as a close initial estimate of μ_t. We are going to do sparse tracking: we use only a subset of J_0, $K_0 = (x_1, ..., x_n)$ (called *reference points*). Thus, tracking consists in finding the series $(\mu_t)_{t\in\mathbb{N}}$ that mimimizes the quadratic differences

$$\mathcal{O}_t(\mu) = \sum_{x\in K_0} (I_t(h(\mu, x)) - I_0(x))^2 \qquad \mu_t = \operatorname*{argmin}_{\mu} \mathcal{O}_t(\mu)$$

Resolution We implement three methods to solve the problem: a non-linear method [4], the method found by Hager and Belhumeur [1], and that by Jurie and Dhome [2].

It is convenient to break down each method in two stages:

1. Learning stage: uses only I_0. The algorithm chooses the reference points K_0. It records their graylevels $G_0 = (I_0(x))_{x\in K_0}$.
2. Tracking stage: using graylevels $G_t = (I_t(h(\mu_{t-1}, x)))_{x\in K_0}$, the algorithm attempts to find $\Delta\mu_t$ that minimizes $\mathcal{O}_t(\mu_{t-1} + \Delta\mu_t)$. This stage executes in real time.

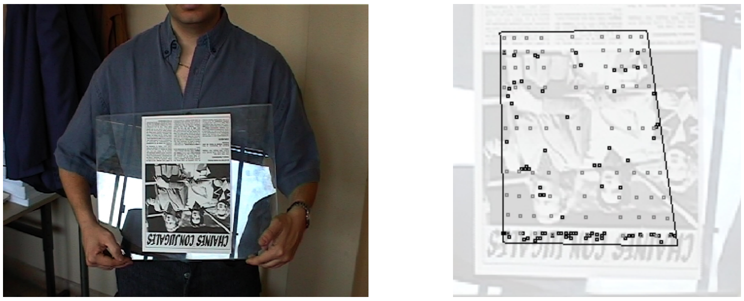

Fig. 3. Frame of a video sequence and its reference points. The quadrilateral represents J_0. The interest points are in black, the regular ones are in grey.

How to chose the reference points We need tens of points to get a reliable estimate of the homography. We use a melt of Harris' interest points [5] and arbitrarily spread *regular points* (figure 3).

Estimating $\boldsymbol{\Delta\mu}$ The *non-linear* (NL) method [4] directly minimizes the criterion $\mathcal{O}_t(\mu)$. This is a non-linear least squares problem, thus it can be solved with an iterative Levenberg-Marcquart algorithm [6].

Basically, the Hager and Belhumeur [1] method (HB) consists in doing (only) one Gauss-Newton optimization step (as used in the Levenberg-Marcquart algorithm). However, they compute the derivative of the objective function on I_{t-1} instead of I_t. This simplifies the computations, as μ_{t-1} can be used to boil down to I_0.

The performance of both algorithms in terms of robustness and precision depend largely on how the image and its gradient are sampled. For the NL method, we are using a precise implementation, which is based on a bilinear interpolation. In this case, the algorithm is very accurate (sub-pixel), but it needs a close estimate of the solution to converge. For the HB method, we use gaussian-filtered samples.

In the Jurie and Dhome [2] method (JD), we begin with a costly learning stage. During this stage, we simulate experiments that consist in applying a disruption on the points of K_0 by random homographies $\mu' = \mu_0 + \Delta\mu$, and measuring the resulting graylevel variation vector at the reference points $\Delta G = (I_0(h(\mu', x))_{x \in K_0} - G_0$. When the disruption is not too big, $\Delta\mu$ and ΔG are related linearly:

$$\Delta\mu \approx A \Delta G$$

by carrying out N ($\gg n$) such experiments, we can estimate matrix A in the least squares sense.

During the tracking stage, we measure $\Delta G_t = G_t - G_0$, and compute $\Delta\mu' = A\Delta G_t$, which is a disruption of μ_0. By a composition of homographies similar to a coordinate change, we can compute the corresponding μ_t:

$$\mu_t = \mu_{t-1} * \mathrm{inv}(\mu_0 + \Delta\mu')$$

where "$*$" and "inv" denote homography composition and inversion, respectively.

Depending on the standard deviation σ of the random disruptions during the learning stage, we can adjust the robustness-precision tradeoff: if σ is big, the tracking is less prone to "loose" its target, but it shakes more, and vice versa.

Synthesis One of our contributions has been to combine the three methods. We apply the more robust methods first, then the more precise ones, as described in figure 4. The $(A_i)_{i=1..e}$ matrices are obtained during the learning stage by applying disruptions with decreasing standard deviations: $\sigma_1 > \sigma_2 > \cdots > \sigma_e$. The function `sample` samples the graylevels at the reference points; `HB_tracking` and `NL_tracking` respectively implement the HB and NL (three iterations) methods, using μ as an inital estimate. At each stage, we make sure that the new estimate lowers the criterion.

We carefully tuned the parameters of the algorithms to get the best robustness-precision compromise (section 3.1). If the number of iterations is reasonable, this technique can be carried out easily in real time.

2.3 Adapting to the panoramic case

In our panoramic case, I corresponds to I_0. An ad hoc method provides the position of the first frame. The problem is: what will the interest region J_0 be? the same object is not guaranteed to be seen on the whole image sequence.

The tiles Therefore, we use *various* interest regions (*tiles*) that cover the whole panoramic image. The tiles are not necessarily disjoint. We choose tiles small enough so there is at least one seen on each video frame, but big enough for the estimation to remain relevant. For each tile, we choose a set of interest points, and we do a learning stage (figure 5(b)).

During the tracking stage, we use μ_{t-1} to calculate which tiles are wholy seen on image I_{t-1}. We compute a μ_t for each of these tiles.

Combining the results The corners of each tile provide 4 points. Thus, if we use r tiles, we have $4r$ point correspondences, and $8r$ equations in the components of μ_t. This problem could be soved using least squares, but a more robust method is described in section 3.3.

3 Experiments and applications

3.1 Finding optimal parameters

We tested the method on a synthetic image (of 1024×1024 pixels) that rotates faster and faster (from $2°$ to $20°$ between two frames). We know the ground

```
Tracking stage
input: $\mu_{t-1}, I_t$.
from learning stage: $K_0, G_0, (A_i)_{i=1..e}$.

$\mu \longleftarrow \mu_{t-1}$                      -- current estimate of $\mu_t$
$G \longleftarrow$sample$(I_t, K_0, \mu)$
$\varepsilon \longleftarrow \|G - G_0\|^2$            -- the criterion for the current $\mu$
For $i$ from $1$ to $e + 2$ do
    If $i \leq e$ then                          -- $e$ Jurie and Dhome computations
        $\mu' \longleftarrow \mu * \text{inv}(\mu_0 + A_i(G - G_0))$
    elsif $i = e + 1$ then                      -- one Hager and Belhumeur computation
        $\mu' \longleftarrow$HB_tracking$(I_t, K_0, \mu)$
    else                                        -- some non-linear iterations
        $\mu' \longleftarrow$NL_tracking$(I_t, K_0, \mu)$
    endif
    $G' \longleftarrow$sample$(I_t, K_0, \mu')$
    $\varepsilon' \longleftarrow \|G' - G_0\|^2$
    If $\varepsilon' < \varepsilon$ then                         -- retain the new $\mu$ if it lowers the criterion
        $\mu \longleftarrow \mu'$
        $\varepsilon \longleftarrow \varepsilon'$; $G \longleftarrow G'$
    endif
endfor
$\mu_t \longleftarrow \mu$
```

Fig. 4. Tracking stage of our algorithm.

truth in this case, so we can define a more reliable validation criterion than $\mathcal{O}_t$, which is:

$$\varepsilon_t = \sum_{i=1}^{4} \| \, w_i - h(\text{inv}(\mu_t), h(\widehat{\mu_t}, w_i)) \, \|^2$$

where (w_1, w_2, w_3, w_4) are the coordinates of the corners of J_0, μ_t and $\widehat{\mu_t}$ are respectively the ground truth and the estimated parameters. When ε_t grows above an arbitrary threshold ($s = 400$, corresponding to an error of 10 pixels per corner) we consider the target is “lost” and stop tracking. The time t^* before the target is lost indicates the robusteness of an algorithm. The average value $\bar{\varepsilon}$ of $(\varepsilon_t)_{t<t^*}$ indicates its precision.

We found the best combination of regular and interest points, and N, the number of experiments of the JD learning stage. Then we tested different sequences of tracking methods (table 1). The sequence offering the best robustness-precision compromise is: $(\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (12.5, 10, 5, 2.5)$.

3.2 Implementation

To handle real images, we made a program that processes a digital video stream on a 866 MHz PowerPC. The decompression of the 25 fps DV stream leaves about 20 ms per frame to carry out the tracking computations.

methods applied	t^*	$\bar{\varepsilon}$
NL	9	50.41
HB	59	74.55
JD 5	103	80.42
HB, NL	42	48.79
JD 5, NL	105	61.35
JD 5, HB	163	33.00
JD 5, HB, NL	154	33.52
JD 5, JD 2.5, HB, NL	332	43.81
JD 10, JD 5, JD 2.5, HB, NL	305	10.30
JD 12.5, JD 10, JD 5, JD 2.5, HB, NL	451	11.03
JD 15, JD 10, JD 5, JD 2.5, HB, NL	453	12.44
JD 20, JD 10, JD 5, JD 2.5, HB, NL	175	10.58

Table 1. t^* and $\bar{\varepsilon}$ for different method sequences. JD σ=JD method with standard deviation σ during the learning stage.

We optimized the most costly computations with vector instructions, so that we need about 2 ms per tile. There are usually 6 to 12 tiles per frame, so we can handle the stream without dropping frames too often (dropping frames makes tracking more difficult). The display is asynchronous. To take into account the difference of CCD responses, we apply a chromatic correction calibrated during the learning stage.

3.3 Outlier detection

We want our tracking technique to be robust with respect to moving objects that hide part of the scene. We do this during the tile merging stage. We use an adapted LTS [7] technique to find the tiles yielding results that are incoherent with the majority of the tiles. This works when the objects are not too big, i.e. few tiles are contaminated.

3.4 Video coding

We want to use this method to stream the video sequence. To do this, we first send the background panorama. Then, in a separate stream, we send the moving objects video, with the homography parameters used to combine the two. This a VOP method described in MPEG-4 [3].

To isolate the changing objects between two images, we use a thesholded pixel-to-pixel difference between the panorama and the frames. We also remove the regions that are too small to be considered objects with a morphological opening (figure 5(f)).

Conclusion

The contributions of our tracking method are:

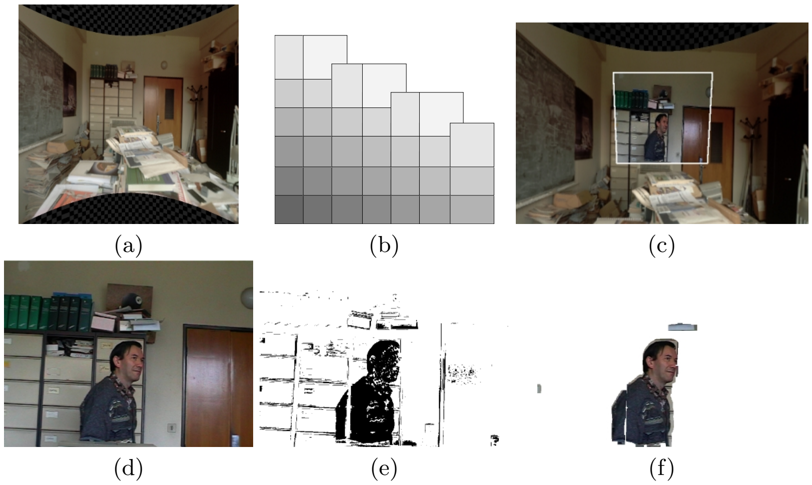

Fig. 5. Panorama (a), some of its tiles (b); frame of a video sequence (d), remapped to the panorama (c); thresholded difference (e), and extracted "object" (f).

- it uses a panorama as a reference image,
- it improves existing tracking techniques while remaining in real time,
- it is the beginning of a real time foreground/background MPEG-4 encoder.

We need to improve the tuning of the parameters and the speed of the robust estimation. We want to map the first frame with the panorama automatically during the learning stage.

References

1. Gregory D. Hager and Peter N. Belhumeur, *Efficient Region Tracking With Parametric Models of Geometry and Illumination*, IEEE PAMI, Vol. 20, oct 1999, p. 1025-1039
2. F. Jurie and M. Dhome. *Hyperplane approximation for template matching.* IEEE PAMI, 24(7), pages 996-1000, 2002.
3. ISO, *Overview of the MPEG-4 Standard*, Melbourne, October 1999.
4. M. Black and A. Jepson, *Eigentracking: Robust matching and tracking of articulated objects using a view-based represesentation*, IEEE PAMI, 20(10):1025-1039, 1998.
5. C. Schmid and R. Mohr and C. Bauckhage, *Evaluation of Interest Point Detectors*, IJCV, 37(2):151–172,2000.
6. J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, NJ, 1983.
7. P. J. Rousseeuw, and K. Van Driessen, *Computing LTS Regression for Large Data Sets*, Technical Report, University of Antwerp, 1999.