Fast Software Image Stabilization with Color Registration

Carlos Guestrin Fabio Cozman Eric Krotkov
{guestrin,fgcozman,epk}@cs.cmu.edu
Robotics Institute, Carnegie Mellon University

Abstract

We present the formulation and implementation of an image stabilization system capable of stabilizing video with very large displacements between frames. A coarse-to-fine technique is applied in resolution and in model spaces. The registration algorithm uses phase correlation to obtain an initial estimate for translation between images; then Levenberg-Marquardt method for non-linear optimization is applied to refine the solution. Registration is performed in color space, using a subset of the pixels selected by a gradient-based subsampling criteria. This software implementation runs at 5Hz on non-dedicated hardware (Silicon Graphics R10000 workstation).

1 Introduction

Consider a camera moving in the world; for example, a person using a hand-held camera or a camera mounted on a car. The images obtained are often shaky, particularly when the camera moves fast. When there is a significant amount of motion, it is hard to keep track of objects in the video sequence. Getting disoriented is easy when the camera moves quickly. When comparing two consecutive frames of a video sequence, the camera motion causes a motion of the contents. Figure 1 illustrates this effect. The goal of image stabilization is to estimate and compensate for this motion.

In this paper, we present a system capable of stabilizing video with very large displacements between frames. This system is applied to feature tracking and to generate stable overlays on the video.

This paper contributes with formulation and results of the fastest software-only implementation of image stabilization. The system presented here is capable of stabilizing video at 5Hz on non-dedicated hardware (Silicon Graphics R10000 workstation). This is also the only published work to use color information in the image stabilization and apply a gradient-based criteria to select the pixels to be used in the registration step.

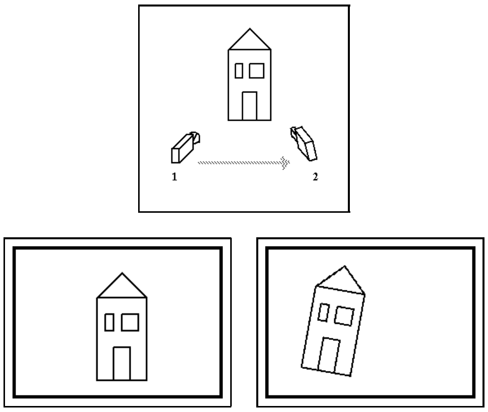

Figure 1: Effect of camera motion on images. Top, illustration of a camera moving in the world. Bottom, resulting images from positions 1 and 2.

This system was integrated with a visual position estimator [3]. The position estimator was developed for space robotics applications, where the video frame-rate is low. Thus, the most important factor for the image stabilization algorithm is to deal with large displacement between frames, a high frame-rate is not fundamental. In applications such as stabilizing a home video, on the other hand, a high frame-rate is fundamental, but displacements between frames tend to be of a smaller magnitude.

2 Related Work

There has been extensive work on image stabilization, [2, 5, 7, 9] are some examples available in the literature. Most of these, including our system, use the same general framework for image stabilization: the motion between consecutive frames of the video sequence is estimated (except for [2]) and compensated for. The main differences are in the motion estimation algorithm used, the type of motion compensation applied, the hardware and the final goal application.

Morimoto and Chellappa developed a system that performs motion estimation by tracking a set of feature points [9]. The system of Hansen et al. fits a global motion model to the optical flow [5]. These implementations achieved higher frame-rates (between 10 and 28Hz) using specialized hardware. These approaches use tracking or flow to stabilize video. Our approach stabilizes the video by directly estimating a global transformation by applying an image registration algorithm. This result could then be used for tracking or flow estimation. We believe our approach is less prone to local effects. Our approach achieves a lower frame-rate, but does not use specialized hardware. The performance of other approaches will probably be up to one or two orders of magnitude lower if used on non-dedicated hardware, while our system would have performance comparable to existing systems if implemented on dedicated hardware.

Irani et al. formulate a method for determining 3D ego-motion from the 2D motion obtained from stabilization [7]. Burt and Anandan demonstrate registration of a frame in the video sequence to a mosaic, rather than to the previous frame [2].

The registration algorithm presented in this paper is also applicable to mosaicing. A similar methodology was formulated and used for image mosaicing by Szeliski [12]. Work on image registration is extensive, Brown presents a survey of this field [1].

3 Registration Algorithms

In our system, image registration is used to estimate camera motion between consecutive frames of the video sequence. The effects of camera motion in the sequence are modeled by the camera motion model. In the literature, there are many models for camera motion [7, 9, 12]. The most common models are: translation, rigid, affine and projective [4].

In this Section, we present two methods used in our image stabilization system: phase correlation and minimization of image difference.

3.1 Phase Correlation

An estimate of image translation can be obtained by using the phase difference of the Fourier transform. This algorithm was proposed by Kuglin and Hines [8].

We will introduce this algorithm by analyzing the continuous one dimensional case. Given $f(t)$ and $g(t)$ such that $g(t) = f(t-a)$, that is, g is a translated version of f, their Fourier transforms are different by an exponential term on the translation, a:

$$\mathcal{F}[g(t)] = e^{-a\omega j}\mathcal{F}[f(t)] \ .$$

Therefore, the phases (Φ) are different by a linear term in a, magnitude is invariant to translation:

$$\Phi[g(t)] - \Phi[f(t)] = -a\omega \ .$$

The desired translation can be obtained by calculating the inverse Fourier transform of $\mathcal{F}[d(t)]$, where:

$$\Phi[d(t)] = \Phi[g(t)] - \Phi[f(t)] = -a\omega \ , \qquad ||\mathcal{F}[d(t)]|| = 1 \ .$$

The inverse is a delta function translated by a:

$$\mathcal{F}[d(t)] = e^{-a\omega j} \ \Rightarrow \ d(t) = \delta(t-a) \ .$$

Therefore, $d(t)$ will have value zero everywhere except at a, which is the desired translation.

This same reasoning can be extended to continuous 2-D functions. Consider, $g(x, y) = f(x-a, y-b)$, in this case, the function $d(x, y)$ is defined by:

$$\begin{aligned} \Phi[d(x,y)] &= \Phi[g(x,y)] - \Phi[f(x,y)] = -(a\omega_1 + b\omega_2) \ , \\ ||\mathcal{F}[d(x,y)]|| &= 1 \ . \end{aligned}$$

Therefore:

$$d(x,y) = \delta(x-a, y-b) \ .$$

An analogous methodology can be used to determine the translation between *discrete* 2D functions, for example images. In the case of images and other sensor data, it is usually the case that g is not a perfect translated copy of f. In this case, the matching function $d(x, y)$ will not have a single non-zero value. The estimate for the translation is obtained by determining the maximum value of $d(x, y)$.

An estimation of translation could also be obtained using an iterative method. These methods yield local estimates, thus, are prone to fall into local minima. This effect is accentuated when displacements are large. The advantage of this method over iterative optimization methods is that the estimate obtained with the phase correlation method is global, thus, overcoming the limitation of local methods. The phase correlation algorithm is capable of dealing with translations of up to 50% of the image size.

3.2 Minimizing Image Difference

The registration problem can also be modelled as the minimization of the intensity difference between the images [12]. It is common to formulate the error function as the sum of the square intensity difference:

$$E^2 = \sum_{x,y} [I_{t+1}(u,v) - I_t(x,y)]^2 ; \text{where } (u,v) = \mathcal{T}[(x,y)]. \quad (1)$$

$\mathcal{T}$ maps the coordinate system of image t to that of $t+1$. This transformation is usually a parametric model of the effect on the images of the camera motion. Thus, the problem can be defined as finding the values of the parameters of $\mathcal{T}$ that minimize E^2.

An iterative numerical optimization method is used to minimize E^2 [12]. These methods are prone to fall into local minima, instead of the desired global minima. To circumvent this problem, the method can be performed in a coarse-to-fine fashion, for example, using a Gaussian image pyramid [11]. Another interesting technique is to also perform a coarse-to-fine approach on model space. [6].

3.3 Optimizing in Color Space

When using grayscale images, a blue sky could be incorrectly registered to a brown mountain, if they have similar intensities. This kind of mismatch is very disturbing to us, as viewers and users of the registered images, because a blue sky and a brown mountain are very different; they don't even have the same color! We have worked on the hypothesis that color gives a strong cue to the registration process, the cue necessary to minimize this kind of mismatch. Therefore, it was incorporated in the formulation of the algorithm.

Consider an image divided into three bands, red ($\mathcal{R}$), green ($\mathcal{G}$) and blue ($\mathcal{B}$). This formulation incorporates color information into the error function:

$$\begin{aligned} E^2 &= K_R \sum_{x,y} [R_{t+1}(u,v) - R_t(x,y)]^2 + \\ &\quad K_G \sum_{x,y} [G_{t+1}(u,v) - G_t(x,y)]^2 + \\ &\quad K_B \sum_{x,y} [B_{t+1}(u,v) - B_t(x,y)]^2 \ ; \quad (2) \\ \text{where} \quad (u,v) &= \mathcal{T}[(x,y)] \ . \end{aligned}$$

The constants K_R, K_G and K_B are weights for each component. Again, the problem is defined as finding the values of the parameters of $\mathcal{T}$ that minimize E^2.

4 Motion Compensation

In this Section, we present an overview of some motion compensation techniques. [4]

- **overlays on raw video:** an application of this work is creating stable graphical overlays on video so that they appear to "stick" to world features as the camera moves. For example, buildings can be labeled in an aerial sequence, or the goal of a teleoperated rover could be indicated.
- **rigid frame:** the motion estimate is used to warp all frames to a reference frame. The viewer feels that camera motion has been removed.
- **smoothed camera motion:** the high-frequency component of the camera motion is removed, eliminating "jittering" in the video, and is used to stabilize the overlays. The low-frequency component of the camera motion, such as that caused by a rover turning, should not be altered. A possible approach is to remove the low-frequency component from the motion compensation. [4]

5 Feature Tracking

Feature tracking is used in many computer vision applications. The motion of the camera and of objects in the scene are the main causes of feature motion. Image stabilization can be used estimate the camera motion component.

In most tracking approaches, a window (or template) around the desired feature is matched using a local search algorithm. These methods are affected by local effects (local minima), such as textureless regions and local repetitive patterns. A coarse-to-fine approach is usually applied to minimize this problem, but it will not solve it completely.

For applications in which camera motion is the main source of motion in the video sequence, our approach is able to estimate this motion. Therefore, it is possible to use image stabilization to track features. This is achieved by applying the transformation $[T]$ to the original position of the feature, thus, obtaining a new estimate of feature location. [4]

The advantage of using a global method is that it is not affected by local effects. This allows us, for example, to track a region with no texture. However, the motion of the feature may not be perfectly estimated with a global motion model. In this case, image stabilization can be used to obtain an initial estimate of feature motion. This estimate can then be refined using a local method.

6 Implementation and Results

We implemented an image stabilization system using the formulation presented previously. This system was integrated into a *Visual Position Estimator* designed to aid operators of teleoperated rovers [3]. This system was able to stabilize a live video stream at 5Hz, using a software-only implementation on a Silicon Graphics R10000 workstation. This frame-rate is adequate for a space robotics application, such as that of the visual position estimation system.

The image stabilization system implemented is able to stabilize video with large displacements between frames, up to 50% of the image size. This is essential when the frame-rate is low.

Video clips and illustrations are available: *http://www.cs.cmu.edu/~guestrin/ImageStabilization/*

6.1 Overview

The outline of the system implement is represented below, for each frame, these steps were followed:

1. **Registration:** the parameters of the transformation $\mathcal{T}$ are determined in two steps:
   - **phase correlation:** first estimate of translation between images;
   - **minimization of SSD:** using Levenberg-Marquardt method in a coarse-to-fine approach on the model and on the image space.
2. **Motion Compensation:** stable overlays are created; one of these techniques of motion compensation is applied:
   - **overlays on raw video**;
   - **rigid frame**;
   - **smoothed camera motion**.
3. **Feature Tracking:** if desired, a local tracking can be performed around the position of the overlays to minimize the drift. The transformation is also used to pre-warp the tracking template.

6.2 Registration

Figures 2 and 3 illustrate the quality of a typical result of the algorithm. On inspection of the difference images, notice that the difference in the overlapping region of the stabilized frames is much smaller than the difference in the original frames.

The input to our image stabilization system is a live 320x240 pixels video stream. Since the Fast Fourier Transform (FFT) requires a window size of 2^n pixels, our algorithm used a 256x128 window.

Figure 2: The top two frames were selected from a video sequence. The bottom image is the square of the difference between the two.

Figure 3: The top two frames were extracted from a stabilized sequence. The bottom image is the square of the difference between the two.

6.2.1 Phase Correlation

Performing FFT on a large image can be computationally expensive. To increase performance, the images may be subsampled before applying the phase correlation algorithm. Subsampling, however, will have an effect on the accuracy of the algorithm. Thus, there is a tradeoff between increased performance and accuracy. Through experimentation, we found that, for this application, subsampling the images eight times offered the best performance to accuracy ratio.

Applying the phase correlation algorithm in this manner yielded a good estimate for the translation between images. This estimate was generally within 5 to 10% of the result obtained with the minimization of image difference using the translation motion model. Subsampling the image eight times decreased the time to apply the phase correlation algorithm to only about 3% of the total computation time of the image stabilization system. As expected theoretically, the phase correlation method was able to detect translations of up to 50% of the image size (in this implementation, up to 128 pixels horizontally and 64 vertically).

6.2.2 Minimizing Image Difference

To perform this minimization, the Levenberg-Marquardt method for non-linear optimization was chosen [10]. Using this iterative optimization method, sub-pixel accuracy can be achieved.

In our implementation, optimization is carried out in a coarse-to-fine approach in image space, using a Gaussian pyramid. A coarse-to-fine approach is also applied in model space, starting with a simpler model and using the result as the initial estimate for the next, more detailed, model. In many cases, the largest motion component in video sequences is caused by translation. It is common to see the camera panning, but not rotating about its optical axis. For this reason, we first estimate translation between images and then estimate other motion models, such as the rigid model (rotation+translation) used in our system.

Registration in Color Space

We chose to weigh each band equally. Therefore, the constant K_R, K_G and K_B in equation (2) were set to one. The weights for each band could be tuned through experimentation or with a physical analysis of the application.

Although, at first the color formulation seems two times more computationally expensive than the grayscale one, in practice this was not the case. The formulation color formulation was only 20% slower, because the optimization converged in fewer iterations.

Robustness, on the other hand, increased dramatically with the use of color information. For many test sequences, visually noticeable mismatches were much more frequent with the grayscale formulation than with color. This algorithm was also applied to image mosaicing. A large number of sequences were tested and the number of visually noticeable mismatches decreased by about 50%.

Gradient-based Subsampling

In order to increase performance, we optimize on a subset of the image pixels. If we chose the pixels in this subset randomly or in using a regular pattern, we may lose important features that help the algorithm converge to a correct solution. Therefore, we would like to select a subset of the image pixels using a criteria that maintains a large proportion of the information. Thus, a *gradient-based subsampling* was performed.

The image gradient information is used to determine the partial derivatives and the Hessian of the error function E^2 with respect to the model parameters. Intuitively, we can think that regions of low intensity variation in the images only make E^2 numerically larger. The gradient information, however, is necessary to determine how to converge to a solution. Therefore, gradient-based subsampling will decrease the amount of data used in the optimization, but maintain a large part of the information needed for convergence.

In our implementation, we only consider a band of a pixel (each pixel is composed of three bands) if its gradient is above a threshold. This threshold is dynamically determined. First, the percentage of the pixels that will be used in the computation is defined. Then, the threshold gradient value that binds that percentage of the pixels is determined. In our implementation, the threshold value is determined by, first, sorting the gradient pixels. The desired threshold is the pixel with index $n_{pixels} * \frac{100 - \text{binding percentage}}{100}$ in the sorted vector. For our application, we found that using 8% of the pixels yielded the best performance to accuracy ratio.

6.3 Motion Compensation

Figures 4 and 5 illustrate the results of image stabilization on a video sequence. The position of the flag was estimated using the global motion estimation calculated by the registration algorithm. Experiments were also performed in smoothing camera motion and motion coding, obtaining successful results (see [4]).

During our tests, we compared the visual result of smoothed camera motion to that of just using the raw video. We found that, for low frame-rates, the improvement is minimal. It seems that in the low frame-rate case the high-frequency motion we would like to remove has not been captured due to the lower acquisition frequency. Therefore, smoothing the camera motion is only necessary when the frame-rate is high.

6.4 Feature Tracking

A feature tracker, using the global motion estimation as an initial estimate for feature motion, was implemented as described in Section 5. First, the global motion calculated in the stabilization step presented previously is used to estimate the position of the feature in the current frame. Then, a local tracking algorithm is applied to refine this estimate. [4] Using this feature tracking improved, visually, the accuracy of the positioning of overlays over long video sequences.

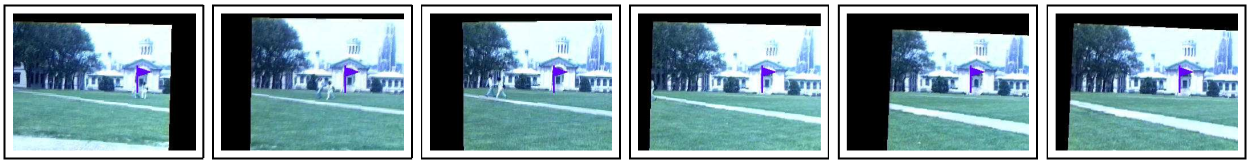

Figure 4: The rigid frame technique of motion compensation is applied to stabilize a video sequence.

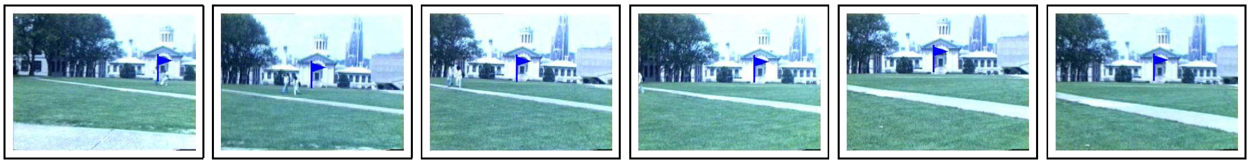

Figure 5: The motion compensation technique overlays on raw video is applied to stabilize the flag.

7 Conclusion

In this paper, we have presented the formulation and implementation of an image stabilization system. The formulation was applied to feature tracking and creation of stable graphical overlays. A software-only implementation stabilizes a live video stream at 5Hz.

The image registration algorithm used phase correlation to obtain an initial estimate for translation between images. The Levenberg-Marquardt method was then used to minimize the image difference in a second step. This minimization was performed in color space on a subset of image pixels selected using a *gradient-based subsampling* criteria. The system is able to deal with large displacements between the images (up to 128 pixels in the current implementation).

The phase correlation method yields a global measure that helps overcome the limitations of algorithms that tend to fall into local minima. The translation estimate was robust and also fast, using only 3% of the total time and being within 5 to 10% of the result of the iterative method.

Using color in the registration process improved the robustness of the algorithm in the order 50%. Registration in color space was 20% slower than grayscale. It would be useful to address questions such as when and why the algorithm fails. A challenging issue is to formulate and test physically founded ideas to tune the weights of each band in the color registration process.

Usual subsampling techniques will often affect robustness. In our experiments, using *gradient-based subsampling* had little effect on robustness and greatly increased the performance.

For feature tracking, it would be very useful to compare the results of the approach presented here with those of other methods. It is also important to improve the local tracker, possibly merging templates, instead of acquiring new ones on every frame.

References

[1] L. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[2] P. Burt and P. Anandan. Image stabilization by registration to a reference mosaic. *DARPA Image Understanding Workshop*, november 1994.

[3] F. Cozman and E. Krotkov. Automatic mountain detection and pose estimation for teleoperation of lunar rovers. *Int. Conference on Robotics and Automation*, 1997.

[4] C. Guestrin, F. Cozman, and E. Krotkov. Image stabilization for feature tracking and generation of stable video overlays. Technical Report CMU-RI-TR-97-42, Robotics Institute, Carnegie Mellon University, November 1997.

[5] M. Hansen, P. Anandan, K. Dana, G. Van der Wal, and P. Burt. Real-time scene atabilization and mosaic construction. *DARPA Image Understanding Workshop*, November 1994.

[6] M. Irani, B. Rousso, and S. Peleg. Computing occluding and transparent motions. *International Journal of Computer Vision*, 12(1):5–16, January 94.

[7] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *International Conference on Computer Vision and Pattern Recognition*, pages 454–460, March 94.

[8] C. Kuglin and D. Hines. The phase correlation image alignment method. *IEEE Conference on Cybernetics and Society*, September 1975.

[9] C. H. Morimoto and R. Chellappa. Automatic digital image stabilization. *IEEE International Conference on Pattern Recognition*, August 1996.

[10] W. Press, W. Vetterling, and S. Teukolsky snd B. Flannery. *Numerical Recipes in C*. Cambridge Press, 1992.

[11] A. Rosenfeld. *Multiresolution Image Processing and Analysis*. Springer-Verlag, 1984.

[12] R. Szeliski. Image mosaicing for tele-reality applications. Technical Report CRL 94/2, Digital Equipment Corporation, Cambridge Research Lab, May 1994.