Using Closed-Loop Detection to Improve Homography Estimation and Mosaicing

Michele Pratusevich

July 25, 2012

Contents

1 Introduction and Overview

The goal of this project is to improve an automatically mosaiced image by detecting a closed loop in a sequence of images and re-computing the calculated image homographies using nonlinear optimization. First homographies will be discussed, with some discussion about factoring homography matrices to determine the various components of a homography. Next the techniques for detecting a closed loop in an image sequence are discussed. Then the optimization problem for improving the homography matrices is described in terms of formulation and the constraints. Lastly, further considerations and ideas are discussed.

2 Homographies

2.1 In General

A homography H in this context is a 3 by 3 matrix $H_{i,j}$ that describes the relationship between image i and image j in an image sequence. In this application, the homographies $H_{i,i+1}$ are calculated between consecutive images i and $i+1$. When calculating an image mosaic, the cumulative homography for image k in the sequence is calculated by multiplying each of the previous homography matrices together until the current image:

$$H_{1,k} = H_{1,2} \cdot H_{2,3} \cdots H_{k-1,k}$$

In the application of creating a 2-D mosaic from a video, the homography is a 2D homography, transforming a point $[u, v, 1]^T$ to $[u', v', 1]^T$ up to a scale factor, normalized in OpenCV to be 1.

2.2 Factoring a Homography Matrix

As discussed by Sonka, et. al in [1], homographies form a group under multiplication as shown above. There are various important subgroups that can be used to factor any homography matrix H into it's components. As discussed in [1], any homography can be decomposed as $H = H_P H_A H_S$ where

$$H_P = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{a}^T & b \end{bmatrix}, \qquad H_A = \begin{bmatrix} K & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad H_S = \begin{bmatrix} R & -R\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

And K is upper triangular. Page 557 in [1] gives more information about these subgroups.

Using the factorization above as a starting point, a homography matrix is therefore written as follows:

$$H = H_P H_A H_S = \begin{bmatrix} KR & -KR\mathbf{t} \\ \mathbf{a}^T KR & -\mathbf{a}^T KR\mathbf{t} + \mathbf{b} \end{bmatrix}$$

Where t is the translation vector, R is the rotation matrix for rotation in the xy-plane, and K is the dilation matrix in the xy-directions. $\mathbf{a}$ and $\mathbf{b}$ are coefficients related to the rotation and dilation in the z direction.

2.3 Computing the Homography Matrices in Code: Overview

The homography matrices can be computed using an OpenCV method called `cvFindHomography`, described in [**?**], which needs to be tuned according to the application you are working with. For instance, the window size that is needed to find a correct homography depends on how close the camera is to the scene in question.

To use the `cvFindHomography` method you already need to have points picked out on both images that are known to be correspondent. In applications

with few images, these points can be picked out by hand and visual inspection. Because the goal of this project was to create a fully automated mosaicing algorithm, manually picking points was not an option. Instead, because the image sequence is taken from a video, the video can be sequenced in such a way to make the images overlapped by a significant amount. Given this information, we use optical flow in the OpenCV method `cvCalcOpticalFlowPyrLK` to calculate the new positions of interest points from the first image in the second image. The optical flow method has parameters to tune as well.

The interest points in the first image are found using the OpenCV methods `cvGoodFeaturesToTrack` and `cvFindCornerSubPix`, described in [3]. The parameters on these methods must be tuned as well.

The `cvFindHomography` method then uses correspondences in those points to compute an image-wide homography that is then used to construct the mosaic.

2.4 Parameters in OpenCV Methods

The various parameters tuned for the OpenCV methods are as follows (written in order as they appear on [2, 3]:

- `cvGoodFeaturesToTrack`
  - –
- `cvFindCornerSubPix`
  - –
- `cvCalcOpticalFlowPyrLK`
  - –
- `cvFindHomography`
  - –

3 Detecting a Closed Loop

TODO

4 Optimization Problem Formulation

This is the problem formulation for the optimization problem in closed-loop homographies.

There is a closed loop detected between image 1 and image n. Each homography computed by the mosaicing algorithm, $H_{i,i+1}$, is the homography from image i to the next image $i+1$, computed using OpenCV (and RANSAC). Once

the closed loop is detected, the homography $H_{1,n}$ is computed between image 1 and the overlap image n.

Overall, the goal of the optimization algorithm is to minimize the error between

$$H_{1,2} \cdot H_{2,3} \cdots H_{n-2,n-1} \cdot H_{n-1,n} - H_{1,n} = 0$$

Which can be written as

$$H_{1,n}^{\text{cumulative}} - H_{1,n} = 0.$$

Using a scalar formulation of the problem, the error is calculated by calculating the sum of the absolute values of the differences between the cumulative and new homographies for each component in the matrix.

$$\sum_{i,j} |(H_{1,n}^{\text{cumulative}})_{ij} - (H_{1,n})_{ij}|.$$

The variables given to the optimizer are the 8 parameters in each of the matrices $H_{i,i+1}$ (each entry of the homography matrix except for the $3,3$ entry, which is assumed to be 1. The values of the $H_{1,n}$ matrix are not variables in the optimization function, but are taken as truth. [1] The optimizer will yield n new homography matrices $H_{i,i+1}^{\text{optimized}}$.

The optimizer tries to minimize this nonlinear multivariable function according to the following constraints:

1. The new matrices that are computed must be homography matrices (i.e. still have determinant close to 1).

$$|\det(H_{i,i+1}) - 1| \leq \text{determinant\_threshold } \forall i \in (1, n-1)$$

2. The variables in each matrix can't change "too much." There are a few ways to implement this:

   (a) The individual variables can only change within a certain range (with this range being different depending on which component in the homography it is):

$$|(H_{i,i+1})_{1,1} - (H_{i,i+1}^{\text{optimized}})_{1,1}| \leq \text{change\_threshold}$$
$$|(H_{i,i+1})_{1,2} - (H_{i,i+1}^{\text{optimized}})_{1,2}| \leq \text{change\_threshold}$$
$$|(H_{i,i+1})_{2,1} - (H_{i,i+1}^{\text{optimized}})_{2,1}| \leq \text{change\_threshold}$$
$$|(H_{i,i+1})_{2,2} - (H_{i,i+1}^{\text{optimized}})_{2,2}| \leq \text{change\_threshold}$$

[1] It is possible to use the components of the closed-loop matrix as variables too, but for now I've implemented the code to consider it as truth. I will try using those components as variables as well.

$$|(H_{i,i+1})_{3,1} - (H_{i,i+1}^{\text{optimized}})_{3,1}| \leq \text{pixel\_threshold}$$

$$|(H_{i,i+1})_{3,2} - (H_{i,i+1}^{\text{optimized}})_{3,2}| \leq \text{pixel\_threshold}$$

$$|(H_{i,i+1})_{1,3} - (H_{i,i+1}^{\text{optimized}})_{1,3}| \leq \text{small\_threshold}$$

$$|(H_{i,i+1})_{2,3} - (H_{i,i+1}^{\text{optimized}})_{2,3}| \leq \text{small\_threshold}$$

The reason each entry (or group of entries) should have it's own threshold is that each component of the homography matrix is related to a different transformation and has different similarity tolerances.

(b) The total change in all the variables of a particular matrix can't exceed a certain value.

$$|\sum_{i,j}(H_{k,k+1})_{i,j}| \leq \text{sum\_thresh}$$

I don't like this way because it does not account for two very large changes in two variables - i.e. if component $(1,1)$ changes by -100 and component $(3,3)$ changes by $+99$, it seems that there wasn't that much change in total whereas in reality the new homography matrix is very different from the old homography matrix.

(c) Factor the new homography matrix and allow not "too much" of a change from the new translation and rotation components.

(d) Don't care about changes between each of the individual homographies but care about the changes in all the cumulative homographies.

Right now my code uses the 2a metric of "too much change."

3. The $3,3$ entry of all the intermediate homographies computed with $H^{\text{optimized}}$ cannot be very different from 1.

$$\text{Let } H_{1,k}^{\text{optimized}} = H_{1,2}^{\text{optimized}} \cdots H_{k-1,k}^{\text{optimized}}$$

$$|(H_{1,k}^{\text{optimized}})_{3,3} - 1| \leq \text{entry33\_threshold } \forall k \in (2,n)$$

5 Results

TODO

6 Discussion and Additional Points of Consideration

TODO

7 Conclusion

TODO

References

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision, 3rd ed.*, Thomson, USA, 2008.

[2] `http://opencv.willowgarage.com/documentation/camera_calibration_and_3d_reconstruction.html`

[3] `http://opencv.willowgarage.com/documentation/cpp/imgproc_feature_detection.html`