

Big Brother: Multi-camera Object Tracking

Mohit Deshpande
The Ohio State University
deshpande.75@osu.edu

Brad Pershon
The Ohio State University
pershon.1@osu.edu

1 Introduction

Cameras are ever present in our society. They are on traffic lights, on top of buildings, inside buildings, and even on buses. Millions of dollars go to law enforcement and cities to help stop crime [5]. Even universities are including cameras in their bus systems [3]. With this massive coverage of cameras, many of them overlap with each other, e.g., the field-of-view of a camera in a bus might overlap with that of the camera on a building that the bus is stopped at. In the event of a crime, law enforcement check all of these cameras and must manually track the perpetrator through potentially hundreds of cameras [5]. Instead, we present an automated solution, called **BigBrother**, that can follow a target across any number of cameras.

2 BigBrother

The problem of object tracking through multiple cameras $\{\mathcal{C}_1, \dots, \mathcal{C}_N\}$, given a target T in one camera, is to find and track that target in the other cameras. For simplicity, we assume that the target T is first identified in \mathcal{C}_1 ; we call this the **primary camera**. We immediately start mean-shift tracking for the four points that define the bounding box around our target. This mean shift produces new coordinates that we can transform to other cameras. We use a variant of covariance tracking in the other cameras to match a target. The initial covariance matrix is taken from the feature matrix of the target T . Algorithm 1 describes our algorithm at a high level. The details are described in subsequent sections.

Algorithm 1 BigBrother Algorithm

```
1: procedure BIGBROTHER( $\mathcal{C}_1, \dots, \mathcal{C}_N, T$ )
2:    $\Sigma \leftarrow \text{feature\_cov}(T)$  ▷ Covariance matrix for other cameras
3:    $B_i \leftarrow \emptyset \quad \forall i \in \{2, \dots, N\}$  ▷ Position of the target in other cameras
4:   while true do
5:      $T \leftarrow \text{meanshift}(\mathcal{C}_1, T)$  ▷ Update the target's position
6:     for  $i \leftarrow 1$  to  $N$  do
7:        $T' \leftarrow H(\mathcal{C}_1, \mathcal{C}_i) \cdot T$  ▷ Transform coordinates using homography
8:       if  $T' \notin \mathcal{C}_i$  then
9:          $B_i \leftarrow \emptyset$ 
10:      continue
11:    end if
12:     $S \leftarrow \text{spanning\_bounding\_box}(T')$  ▷ Search region around meanshift estimate
13:     $B_i \leftarrow \text{multiscale\_cov\_tracking}(\Sigma, S)$ 
14:  end for
15: end while
16: end procedure
```

2.1 Homography

To track a target T across multiple cameras, we must have either complete 3D knowledge of the space of the cameras and their positioning in that space or, more simply, a homography between the cameras. We can only compute the homography if the field-of-view of the cameras overlap, which we assume. Specifically, for N cameras, we require only $N - 1$ homographies because we can compute the inverse homographies and use a sequence of them to translate between any two cameras.

We compute a homography offline by selecting corresponding pixels in the two camera views. We may use a corner or feature detector to make pixel selection easier. We require at least 4 corresponding coordinates to compute the homography, however, for a better homography we used ? points.

Each point is converted into inhomogeneous coordinates, and we solve the system of linear equations to produce the homography matrix H between two camera views.

For robustness, we use a normalized direct linear transform. We normalize our points to have zero mean and shift the average distance of the resulting points to the origin is $\sqrt{2}$. We could have additionally improved this algorithm using Random Sample Consensus (RANSAC) [2].

2.2 Mean-Shift Tracking

After computing the homography and identifying the region of interest in one camera frame, we can start mean-shift tracking [1]. We use ? color bins for the color quantization. We use circular neighbors with a radius of $r = ?$. For the Epanechnikov profile, we use $h = ?$.

At each frame of the mean-shift tracking, we use the homography to transform the mean-shift-tracked points into the other camera(s). If the homography is accuracy, we can determine the target is in a particular camera frame or not. If the transformed points are not in the frame, i.e., outside of bounds of the camera frame. However, if the points lie inside of the camera frame, we know that the target is present in a particular camera frame.

2.3 Covariance Tracking

To find the target in the other cameras, we use covariance tracking [4]. The primary issue with covariance tracking is speed: we have to search the entire image. However, we can the transformed mean-shift estimate to narrow our search space. In particular, the covariance matrix initialized from the original target in the primary camera. For our feature vector, we try several different representations, but use radial coordinate system.

Homographies are not completely accurate, hence, the transformed points will not be accurate. They may also be skewed, i.e., not aligned with the image coordinates. We determine the minimum spanning rectangle around our transformed points and use that as our search space. Then we can perform covariance tracking only in that space. This drastically limits our search space. We found a $XX\%$ reduction in the number of pixels we must search.

We do not start mean-shift tracking in the other camera frame since we want to maintain the relationship of the target between several cameras. If we were to mean-shift track in all cameras, these would operate independently. This makes it impossible to pick up a target in a new camera since mean-shift would stop tracking after the target leaves a frame. To remedy this, we use the homography to connect the primary camera to the other cameras. If the target moves into the field-of-view of another camera, we will notice the project points will lie in the camera's field-of-view.

2.4 Work Allocation

Mr. Pershon was tasked with computing the homography H between the two cameras. This was computed by manually matching correspondence points and solving the linear system of equations. He also coded a real-time implementation of mean-shift tracking for the primary camera.

Dr. Deshpande implemented the covariance tracking variance and the code for the cross-camera cooperation. He also implemented the multi-scale covariance searching across aspect ratios.

2.5 Challenges

Co-planar points. Since we use homographies for translating between different cameras, we only get reasonable results when the target is co-planar with the ground-plane. This is the largest limitation since many cameras, particularly indoor ones, do not have a high enough vantage point to satisfy the co-planar points, or even approximate it reasonably.

Inaccurate homography. Homographies or other measurements are not entirely accurate; there is always some error associated with them. When transforming points between cameras, the transformed points are not that accurate. However, we can ameliorate this inconsistency by using more points when computing our homography.

Loss of target in primary camera. One assumption our algorithm makes is that the target T remains in the same camera the entire during of our approach. If we lose the target in one camera but it is still present in another camera, we simply switch from covariance tracking to mean-shift; that camera becomes our primary camera. If there are multiple cameras,

Vastly varying target sizes. The covariance matrix is computed from a patch of fixed size. When we perform the search, we compute the covariance of several aspect ratios. However, if our target does not match one of these aspect ratios, then we may miss the detection. However, since they're computed as a function of the original target dimensions, we are likely to detect the target in the other cameras.

3 Results

Without loss of generality, we consider only two cameras for our experiments.

3.1 Lessons Learned

Vectorize code. Vectorized code tends to run faster than non-vectorized code due to parallelism and efficient algorithms for large matrix multiplication, e.g., Strassen's Algorithm. We learned a good approach is to start without vectorized code and remove loops by condensing data structures into matrices or tensors. Tensor arithmetic is also noticeably faster than non-vectorized code.

Use camera calibration files. Many datasets we worked provided their own camera calibration matrices. Working with these matrices is not entirely straightforward at times.

4 Future Work

Narrow search space with an object detector. Instead of creating the search space from the projected points, we can reduce our search space even further by running a object detector, such as Faster R-CNN, YOLO, or Single-Shot Detector, on each non-primary camera. These models draw bounding boxes on each object in a frame, depending on the frame rate. Then we can check the box of the object closest to the transformed points.

Construct a visual hull. Knowing the locations of cameras in 3D space allows us to use an algorithm such as VisualHull to construct a complete 3D wireframe of the target. This provides additional information, such as target height, that may be used to help narrow searching or constructing a description of the target.

Triangulate new cameras. Cameras on moving objects, such as buses, are difficult to incorporate in this system. However, if we can identify the position and pose of a camera using VisualHull, we can compute an approximation of the moving camera in the 3D space of the fixed cameras. This immediately adds the moving camera to the network of fixed cameras. The moving camera is removed from the network when it moves outside of the 3D space of the fixed cameras.

References

- [1] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 2003.
- [2] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.
- [3] New cameras enhance cabs security. <https://ttm.osu.edu/cabscameras>.
- [4] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on lie algebra. *CVPR*, 2006.
- [5] Privacy fears grow as cities increase surveillance. <http://www.nytimes.com>.