Auto-Determination of Camera Parameters for Scenario Simulations in Visual Surveillance Applications

Richard Chang Nam Pham Trung Karianto Leman Wang Yue
Institute for Infocomm Research (A*STAR)
1 Fusionopolis Way
#21-01 Connexis 138632 Singapore
rpchang@i2r.a-star.edu.sg

Abstract

We propose a novel method for determining geometric parameters of general cameras to generate augmented reality images. Visual surveillance systems are usually evaluated by using datasets from different camera configurations. Then, the system is often fixed and one cannot ensure that the same methods can work in all configurations. Our approach deals with the auto-determination of geometric parameters from the images to generate virtual views situations, and evaluate quantitatively visual surveillance systems in any configuration. The method is based on geometry-free characterization of the images using a common codebook and random ferns to find correspondence points. Experimental results are presented on real data from two different locations for people collapse scenarios.

1 Introduction

Many computer vision applications require calibration information in order to provide 3D information about the scene. For visual surveillance systems, it may be relevant to display camera topology or 3D location of persons or objects. Baker and Aloimonos [2], Faugeras et al [4] introduced the pioneering approaches of single and multi-camera calibration. This pre-processing step is usually done offline but is very constraining since the cameras must be stationnary and cannot be moved once the parameters have been computed. Different approaches were used to calibrate the cameras. Some methods are based on feature points [3], known pattern [5] or trajectories of the objects [1]. These methods may be difficult to apply due to practical conditions such as the size of area or the number of the cameras to process. Autocalibration methods have been also introduced [4] based on camera motion [8] or object motion. These approaches are less constraining since it does not require a known calibration object but rely on other parameters like camera or object motion. Then, some assumptions have to be made on the system which limits the range of applications. These methods may also fail in case of major changes of the images. The camera position can influence the results of the computer vision algorithms since the view could be completely different and a same shape could appear differently. Testing tracking and computer vision algorithms to ensure the robustness of the method may be troublesome since all the configurations have to be set. Simulate the data can then be an efficient and convenient way to evaluate the performance of the methods.

In this paper, we introduce a new method to compute the geometric calibration parameters for tracking and event detection system evaluation. Given these parameters, we can simulate any position or orientation of the cameras in the scene and generate new views including virtual people or objects accordingly. It can be also used to generate training data to increase the global performance of the system. Our method relies on the global features of the image. Recent works on bags-of-features [6] representations have become popular as they introduce geometry-free features to characterize local subimage using statistical tools. The paper presents a common description visual language to determine calibration parameters from a dataset of images. It can deal with small and big changes. This paper is organized as follows. Section 2 presents the general overview of the method. Section 3 gives the characterization of the image, Section 4 describes the determination of the parameters, and finally Section 5 shows experimental results.

2 System Overview

The system is designed to evaluate visual surveillance methods for tracking or event detection applications. A training phase is firstly performed to model the parameters corresponding to the system configuration using a small dataset of images. Then using these parameters, the calibration data are determined in order to generate simulations. Each query image I is characterized according to the common codebook of the dataset. Random Ferns [7] are then used to find the nearest image I_n in the dataset. The geometric parameters of the image I are then computed using the image I_n. Secondly, we use calibration information to insert virtual objects and people according to scenarios. Fig. 1 shows the overview of the method.

3 Camera Parameters retrieval

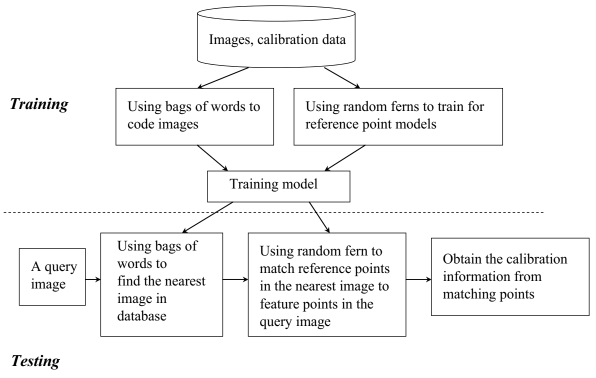
3.1 Characterizing texture

Our approach to measure the texture relies on the computation of a histogram of the difference between the value of pixels of images. Given an image I, each value of its histogram of difference h_I is given by :

$$h_I(i) = \sum_{x,y,x',y' \in I}^{x \neq x' \vee y \neq y'} diff(I,x,y,x',y',i),\ i \in [0,255] \tag{1}$$

with

$$diff(I,x,y,x',y',i) = \begin{cases} 1 & \text{if } |I(x,y) - I(x',y')| = i \\ 0 & \text{else} \end{cases} \tag{2}$$



Images, calibration data
Training
Using bags of words to code images
Using random ferns to train for reference point models
Training model
A query image
Using bags of words to find the nearest image in database
Using random fern to match reference points in the nearest image to feature points in the query image
Obtain the calibration information from matching points
Testing

Figure 1. General overview of the method. A first training phase corresponds to model the parameters of the configuration, then the calibration parameters are computed from this model.

In second stage, the histogram h_I is normalized, to ensure the invariance according to the size of I.

3.2 Generating codebooks

Let $F_z(I)$ be a function allowing the decomposition of an image I into several textured patches :

$$F_z(I) = \{z_0, z_1, ..., z_n\} \text{ with } I = \bigcup_{i=0}^{n} z_i \tag{3}$$

Let $T = \{h_{z_0}, h_{z_1}, ...h_{z_n}\}$ be the set containing all texture descriptors of patches z_i of I. The idea is to sample T to reduce the number of descriptors to $m \leq n$. We then add to T a metric function expressed by $dist(h_{z_i}, h_{z_j})$ and a reference texture patch h_{ref}. The reference patch is set to a patch containing a single color, corresponding to a uniform region. In second stage all the representation of patches contained in T are compared to h_{ref} and sorted according to their texture. The Mahalanobis distance is used as a metric function and is set so for the rest of the paper. At this point, T_s is then sampled into m areas. For each area, only the median patch is selected. The resulting selection gives the codebook V :

$$V = \{h_{ref}, h'_{z_0}, h'_{z_1}, ...h'_{z_m}\},\ V \subset T_s \tag{4}$$

that corresponds to the most representative patches.

3.3 Decomposing images into known patches

Let I_{acq} be an acquired image, I_{acq} is decomposed into z_{acq_i} patches. Each computed patch must be compared to the content of V, we then set a function G that transforms the patches of acquired images into patches of the codebook V. In case a new patch is detected, it is added to the codebook as a new entry. The acquired image I_{acq} is then codified using the patches of the codebook, the resulting image I_{cod_i} given by:

$$I_{cod_i} \in \cup G(z_{acq_i}, V) \tag{5}$$

3.4 Optimal decomposition of images

An efficient decomposition must produce an optimal and possibly unique partitioning of images. In addition it would be interesting to produce less patches, but of variable size so that they can cover homogeneous texture zones. In order to achieve an optimal generation of patches, a quadtree-like algorithm is set up. The quadtree algorithm cuts recursively images into subimages. Starting from the initial image, each subimage is cut into four equal subimages. The idea is to use the same principle, but in contrast to of the regular quadtree approach, the division of subimage will be driven by an entropy measure. A subimage is cut at the location were the difference of the quantity of information between possible subimages is minimal. This quantity of information is given for an image I by :

$$Q(I) = -\sum_{c=0}^{c=255} P(c) \log P(c) \tag{6}$$

where $P(c)$ is the probability of appearance of the grey value c within I. Finally the optimal (x, y) position is the one minimizing the following sum of differences:

$$min_{x,y}(\sum_{a=1,b=1}^{a=2,b=2} (Q(I_m) - (Q, I_{ab}, x, y)(x, y))^2) \tag{7}$$

An illustration of the algorithm is given in Fig. 2. It appears clearly that the image is decomposed more coherently, a complete overview of the method can be found in [6]. The presented work is used as it is to build the common vocabulary and to codify the images. It is a tool to characterize the frames provided by the cameras and represents a part of our structure.

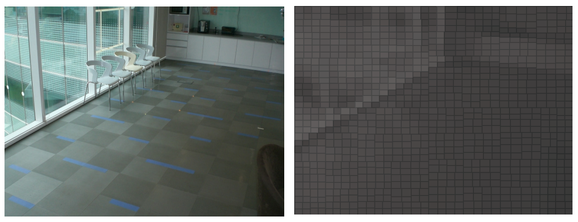

(a) (b)

Figure 2. Quadtree Decomposition of the image. At each step, the four regions have the same entropy value, the image is coded using 1024 patches. (a) Original Image (b) Codified image.

3.5 Finding the nearest image in the database

A geometric-free method is used to determine the geometric parameters. First of all, a dataset of images is built offline. This database corresponds to a sampling of the scene. Every codified query image $I_{cod_i} = \{z_i\}$ is then compared to the codified images of the database using the cross-correlation score:

$$Corr(z_i, z_j) = \frac{\sum (z_i - \bar{z_i}) \cdot (z_j - \bar{z_j})}{\sqrt{\sum (z_i - \bar{z_i})^2} \cdot \sqrt{\sum (z_j - \bar{z_j})^2}} \tag{8}$$

The selected image is then the one with the highest score:

$$I_n = max_{i,j}(Corr(z_i, z_j)) \tag{9}$$

4 Projection matrix computation

For a given camera, the projection matrix is computed using the parameters of nearest image I_n in the database. Correspondence points between the query image I_q and I_n are then determined to compute the geometric parameters of I_q. The image I_n is then set as a reference image.

4.1 Find correspondence points

Correspondence points from the nearest image I_n and the current view I_q are found in a two-steps method. First, random ferns [7] are applied to find a set of correspondence points between the two views. A fern is defined as a non-hierarchical structure that includes a set of binary tests. Given a patched image, a fern will output a binary number based on its binary tests. For example, $F_k = \{f_{k,1}, f_{k,2}, ..., f_{k,S}\}$ is the k-th fern, where $f_{k,j}$ is 0 or 1 depending on the test j of this fern. In [7], for each binary test j, two locations from the image I_n are chosen. If the intensity of the first location is larger than the intensity of the second location, then $f_{k,j}$ will be 0. Otherwise, $f_{k,j}$ will be 1. For each image patch, fern features will be extracted and classified to each class by using the naive Bayes classification method. Because the fern classification is based on the statistical approach, it can handle challenging cases such as rotation, lighting, affine deformations provided that the training set is large enough.

In step 2, covariance features [10] of patched images from each pair of correspondence points in the first step are extracted to verify the classification. The covariance feature is the covariance matrix of feature vectors of pixels in a patched image. The feature vector is defined by:

$$Feat(x,y) = \left\{ x, y, R(x,y), G(x,y), B(x,y), \left|\frac{\partial I(x,y)}{\partial x}\right|, \right. \tag{10}$$
$$\left. \left|\frac{\partial I(x,y)}{\partial y}\right|, \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right|, \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right| \right\}$$

If the distance of covariance features from two correspondence points is smaller than a threshold, these two points are confirmed in the same class. Otherwise, the correspondence pair is excluded.

4.2 Estimate projection matrix

Let P_r be the projection matrix of the reference view chosen from the database in Section 3. P_r can be represented as $P_r = K_r [R_r \;\; T_r]$

where K_r is the intrinsic parameters matrix, R_r is the rotation matrix, and T_r is the translation vector. We need to find the projection matrix of the current view I_q , P_c. From [5], the rotation matrix R and translation vector T from P_r to P_c can be obtained from the fundamental matrix F between the reference image I_n and the current image I_q. The matrix F is calculated from correspondence points in the previous section.

Let E be the essential matrix. E can be obtained from F: $E = K_c^T F K_r$ Here, we assume that the changing of views does not affect to the intrinsic parameters. Hence, $K_c = K_r$. Let U, S, V be the singular value decomposition of $E = USV^T$. The rotation R and translation T from P_r to P_c are

$$\begin{aligned} R &= UWV^T \text{or } UW^TV^T \\ T &= v_3 \text{ or } -v_3 \end{aligned} \tag{11}$$

where $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, and v_3 is the last column of V. From R and T, projection matrix P_c can be obtained. The details of the algorithm are shown in Fig. 3.

```
Step 1. Apply the characterization method to find
        the reference view that is close to the current
        view (Sec. 2)
Step 2. Using random ferns to find correspondence
        points between the reference view and
        the current view (Sec. 3)
Step 3.
   n = 0
   While (n < number of trials)
      Use RANSAC to find the fundamental matrix F
      Calculate $R, T$ (Eq. (11))
      $\tilde{R}_c = R \times R_r$
      $\tilde{T}_c = R \times T_r + T$
      $d$ = average distance of points to epipolar lines
            calculated from correspondence points
      If $\left(\tilde{R}_r, \tilde{T}_r\right)$ is near to $(R_r, T_r)$ and $d < d_{\min}$
         $d_{\min} = d$
         $R_c = \tilde{R}_c$
         $T_c = \tilde{T}_c$
      EndIf
      $n = n + 1$
   End
Step 4. Output $R_c, T_c$
```

Figure 3. Our method to find the projection matrix of the current view.

5 Experimental Results

Our method has been tested in two locations. The first one is a pantry ($5m \times 12m$) and the second one is a lift lobby ($3.5m \times 15m$). A small database of 10 images is used for each location. A random query static image is taken from a basic viewpoint. This image is then codified using the common codebook built from the database. The cross-correlation measure is used to find the nearest image in the database. Fig. 4 shows the 5 nearest images and the one selected by the system. The selected frame has the highest correlation value (0.97). Then, we compute some correspondence points between both images. Using our method, we have 24 inliers points out of 26 points shown in Fig. 5. Finally, the performance of estimating the projection matrix is shown in Table 1. In this table, the

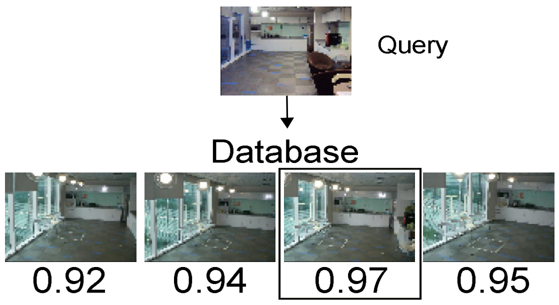


Figure 4. Cross-correlation scores between the query image and the ones in the database. The selected one has the highest value (0.97).

error of translation is the Euclidean distance between the translation vector from estimation methods and ground truth. The error of rotation is the distance between the Rodrigues rotation vectors. Ground truth is computed by using the camera calibration toolbox [3]. We compare results of the projection matrix estimation between our method and KLT feature tracking. Since our method to find correspondence points is based on the training database for ferns, it can handle variations of features. Hence, correspondence points are chosen more accurately. This lead to better estimations of the projection matrix. This table shows that our method has smaller error in both translation (2.8%) and rotation vector (3.18%) compared with using KLT [9] feature tracking. This estimation error is low and allows to simulate some scenarios in both locations. Then, we added several virtual people in the scene corresponding to a people collapse scenarios using XNA Game Scenario. Each person has an independent behavior. Fig. 6 shows that objects are fitted in the scenes correctly. The accuracy is good for applications such as creating data for verifying visual surveillance system.

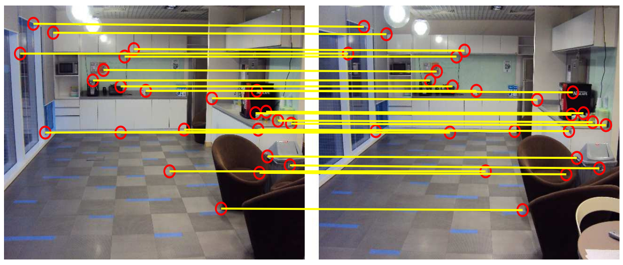


Figure 5. Finding correspondence points from our system between the reference (left) and the query (right) images. 24 inliers points have been retrieved out of 26 points.

Methods	Translation Min. Errors	Rotation Min. Errors
Our method	17cm (2.8%)	0.1 rad (3.18%)
KLT feature	43cm (7.2%)	0.11 rad (3.5%)

Table 1. Results for estimating projection matrix

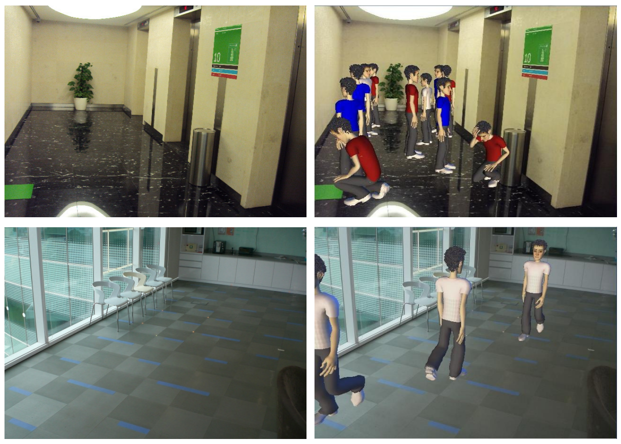


Figure 6. Results of fitting virtual objects to the scene. (left) Original images in the pantry and the lift lobby. (b) Generated Images.

6 Conclusion

We propose a new method to compute the geometric calibration parameters for tracking and event detection system evaluation. The contribution of this paper focuses on determining the camera geometric parameters using a common codebook and random ferns to generate simulations for visual surveillance system evaluations. A first training phase is performed to get the model of the camera, this model is then used to otain the calibration parameters from correspondence points. Virtual objects and people can then be added in the images in order to simulate any scenario. Experimental results on two different locations show the robustness of the method and the average error in estimation is about 3% which is less than 8% of KLT method.

References

[1] N. Anjum and A. Cavallaro. Single camera calibration for trajectory-based behavior analysis. In *AVSS*, 2007.
[2] P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *Omnivis*, 2000.
[3] J. Y. Bouguet. Camera calibration toolbox for matlab.
[4] O. Faugeras, Q.T. Luong, and S.J. Maybank. Camera self-calibration: Theory and experiments. In *ECCV*, 1992.
[5] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Univ. Press, 2004.
[6] L Lacheze and R Benosman. Visual localization using an optimal sampling of bags-of-features with entropy. In *IROS*, 2007.
[7] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *CVPR*, 2007.
[8] M. Pollefeys, L.V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. In *IJCV*, 2004.
[9] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR*, 1994.
[10] O. Tuzel, F. Porikli, and P. Meer. Region covariance: a fast descriptor for detection and classification. In *ECCV*, 2006.