Dynamic Mosaic

CHEN BO
CHUA KOK BENG MARCUS
FENG JIMIN

Introduction

Aims and objectives

- Aim: Take a short video of a walking person and track the person by moving the camera. Then produce a video with a static mosaic background and the person walking in the mosaic video

- Objective: practice and apply the principles of computer vision to achieve the above aim

Tools used in this project

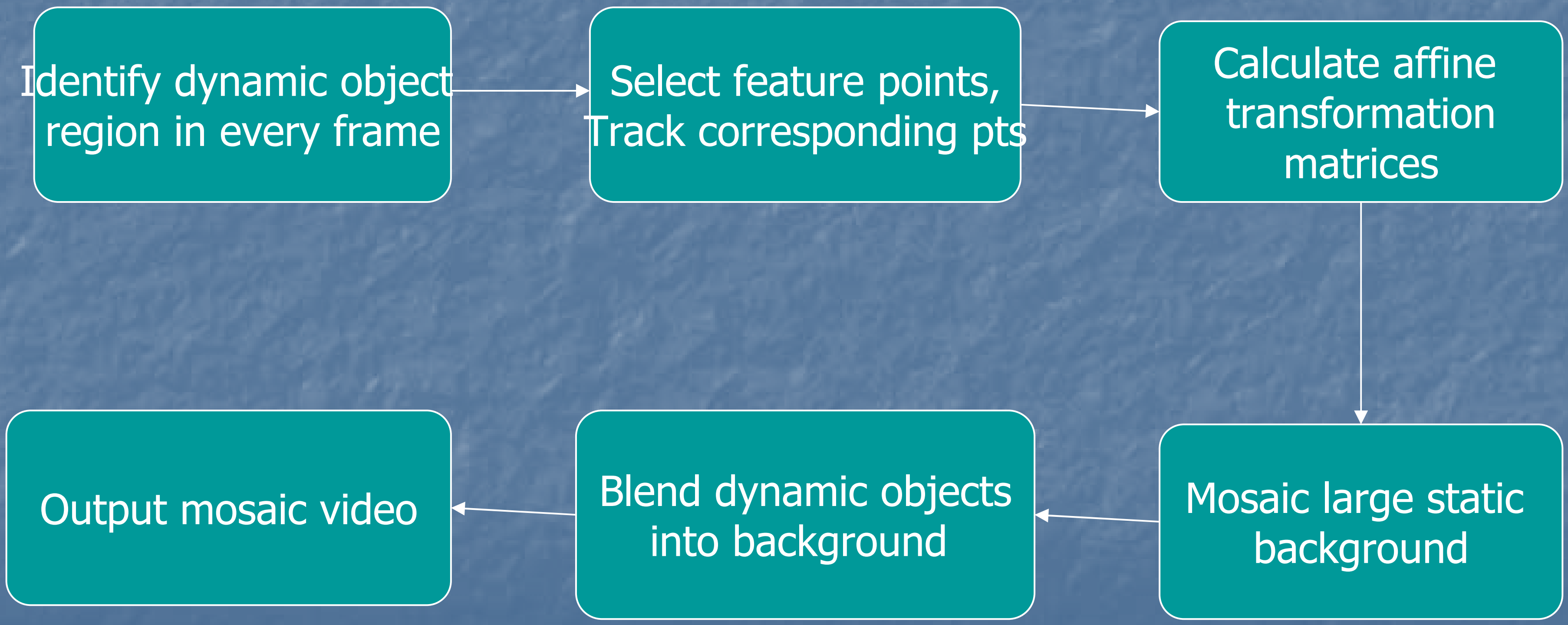
- Matlab 7.0 , particularly its image functions library
- Virtual dub software for video format conversion

Assumptions

- Camera without too much jitters
- Consecutive frames without sudden change of intensity
- Displacement of dynamic object in two consecutive frames should not be large.
- The first frame has overlapping region with the last frame
- There must be one frames that can cover the moving objects to form the static background

Our method of solving the problem

- Flow chart



Step 1: Track region around dynamic object

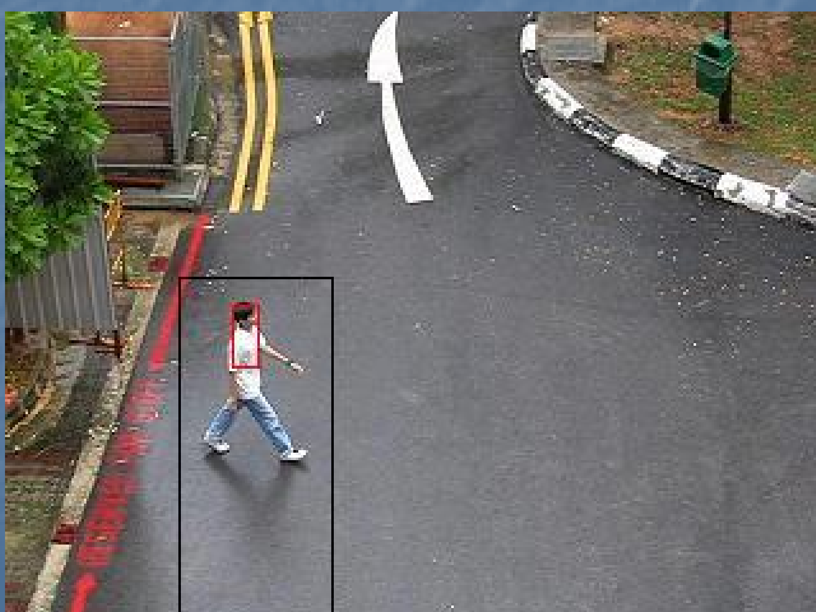
- Select general region around dynamic objects (moving region) in first frame

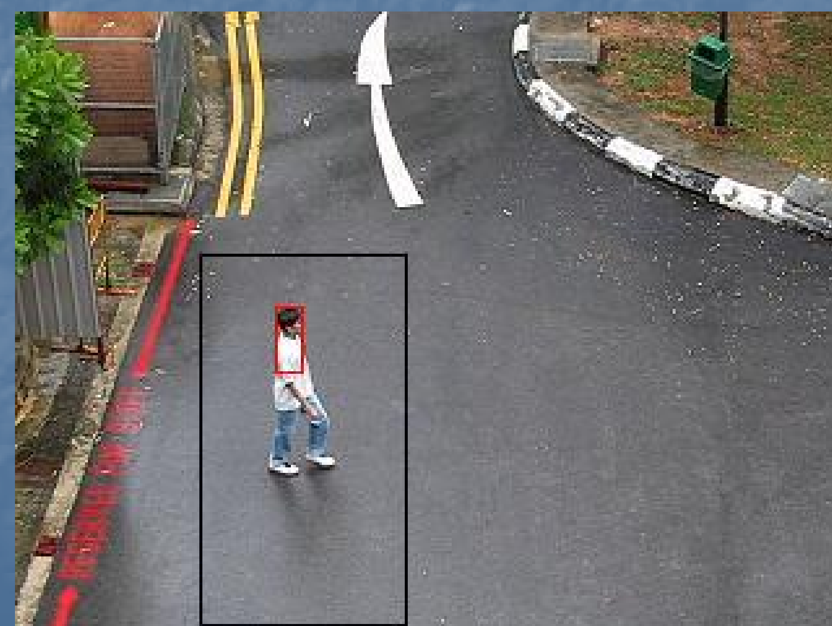
Step 1: Track region around dynamic object

- Use template matching to identify region around the objects in subsequent frames

Step 1: Track region around dynamic object

- Use template matching to identify region around the objects in subsequent frames





Step 1: Track region around dynamic object

- Use template matching to identify region around the objects in subsequent frames
  - Convert to gray image
  - Reduce size of frames
  - Template matching restricted to neighborhood region
  - Algorithm: find least square sum.

$$E(x,y) = \sum_i \sum_j (f(x+i, y+j) - k(i,j))^2$$

- Identify more dynamic objects

Step 2: Find corresponding points

Essential for auto image registration and mosaic. two sub steps:

- First auto detect good features from the first frame
- Then track the selected features in the subsequent frame

Step 2: Find corresponding points

- Auto Feature detection --- in first frame
  - The idea is to evaluate the cornerness of the image points by using its gradient

$$\frac{(\theta_x, \theta_y) \cdot (-I_y, I_x)}{\|\nabla I\|}$$

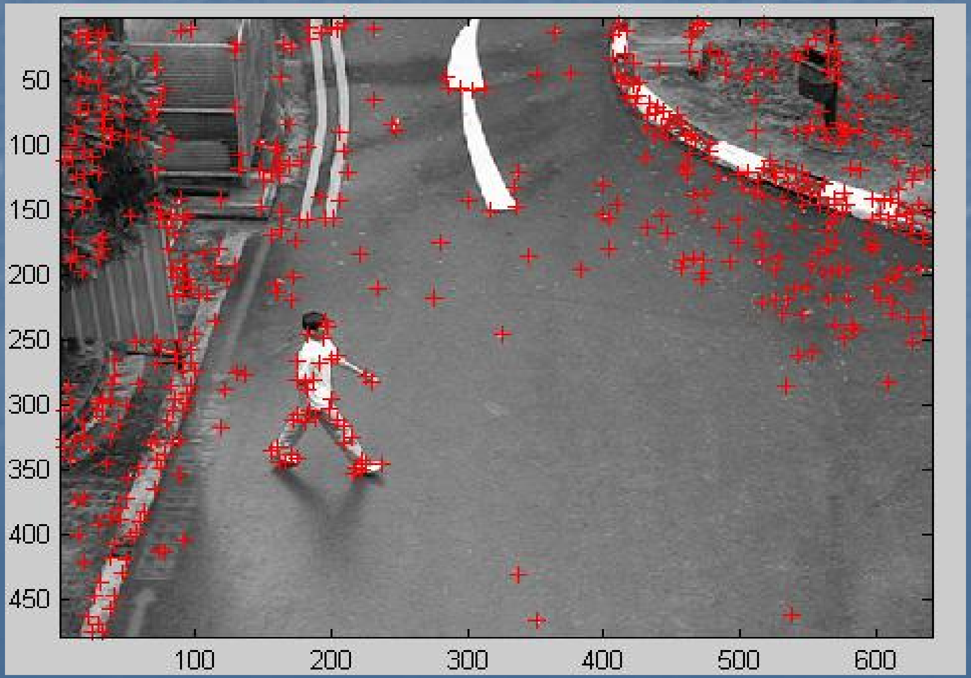
  - Also, use Tomasi Method to evaluate the quality of the grids by computing the eigenvalues (λ_1 , λ_2) of the following matrix, taking min ($\lambda 1$, $\lambda 2$) as its quality

$$\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Step 2: Find corresponding points

- Auto Feature detection (cont…)
  - • Next, perform nonmaximum suppression to find local maximums of the image
  - • local maximums with grid quality > threshold will be the good features to track

Threshold = max(quality(:))/10



Step 2: Find corresponding points

- Tracking of good features

the ideas:

- Lucas & Kanade method to calculate the displacement and find corresponding pts using

$$\mathbf{Z}\,\mathbf{d} = \mathbf{e} \qquad (26)$$

where

$$\mathbf{Z} = \begin{bmatrix} \sum_{\mathbf{x}\in W} w\, I_x^2 & \sum_{\mathbf{x}\in W} w\, I_x\, I_y \\ \sum_{\mathbf{x}\in W} w\, I_x\, I_y & \sum_{\mathbf{x}\in W} w\, I_y^2 \end{bmatrix} \qquad \mathbf{e} = \begin{bmatrix} \sum_{\mathbf{x}\in W} w\,(I - J)\, I_x \\ \sum_{\mathbf{x}\in W} w\,(I - J)\, I_y \end{bmatrix}$$

the steps:

- down sampling the image to form a image pyramid
- perform tracking of the features in coarser level, reject the features if out of boundary or quality of the tracked points is lower than threshold

Step 2: Find corresponding points

- Tracking of good features

  good features points tracked at the last frame can be used to select corresponding points for calculate affine transformation matrix

- Result: last frame

Step 2: Find corresponding points

- Choose good points: find a fixed number of good points from tracked points in last frame that can be used for calculating affine transformation
- Steps:
  - • Remove points inside the moving region of the frames
  - • calculate the displacements of the remaining pts, form a histogram, and choose pts that are in the peak
  - • also have to check the major direction of the displacements: chosen pts must have similar direction
  - • recursively apply the steps to obtain the best n points

Step 3: Calculate Affine transformation

- Calculate affine transformation matrix between each pair of consecutive frames using 2 sets of linear equations of the form

Ax = b

$$\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \end{bmatrix} = \begin{bmatrix} x'_1 \\ \vdots \\ x'_n \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} y'_1 \\ \vdots \\ y'_n \end{bmatrix}$$

Where x_1, y_1… x_n, y_n are points in the first image and x'_1, y'_1… x'_n, y'_n are the corresponding points in the second image.

Step 3: Calculate Affine transformation

- Requires at least 3 pairs of points.
- MATLAB is used to calculate the values of a11, a12, a13, a21, a22, a23 using the least square solution.

$$x = (A^T A)^{-1} A^T b$$

Step 4: Create large Static Background

- Warp the first frame using the transformation matrices calculated in previous step
- From the previous step:

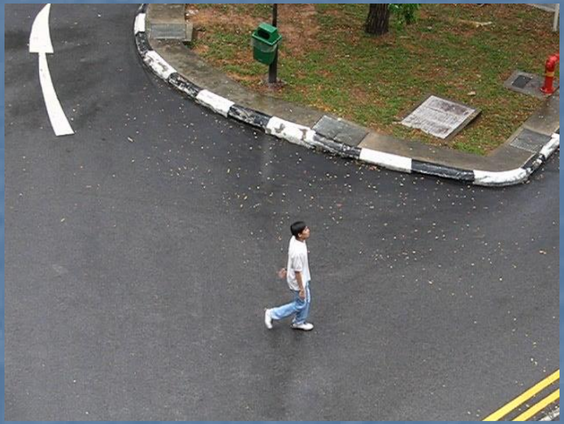
$$\text{Frame1} \xrightarrow{T_1} \text{Frame2} \xrightarrow{T_2} \text{Frame3} \; \ldots \xrightarrow{T_{n-1}} \text{FrameN}$$

$$\text{Frame1} \xrightarrow{T} \text{FrameN}$$

Where $T = T_{n-1} * T_{n-2} * \ldots * T_1$

Step 4: Create large Static Background

- Mosaic first and last frame



Step 4: Create large Static Background

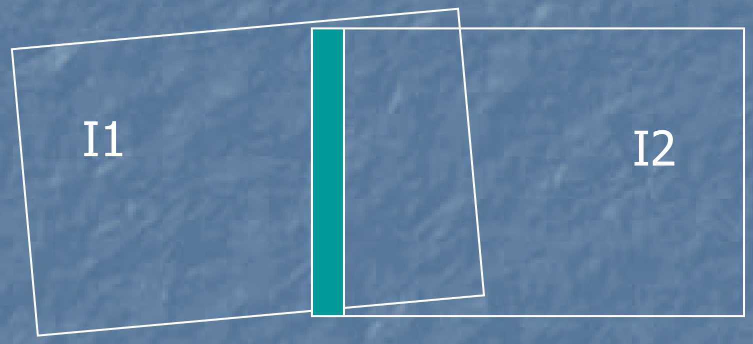
- Mosaic first and last frame

Step 4: Create large Static Background

- Mosaic first and last frame

Step 4: Create large Static Background

- Apply blending function near the image boundary.



- Use weighted average: Use the distance from the pixel to the boundary as weight

Step 4: Create large Static Background

- Before Blending

- After Blending

Step 4: Create large Static Background

- Cover moving objects in the mosaic image using selected frames to form the static background

1. Warped the selected frame (i^{th} frame) that will be used to cover up the region. The matrix, T used to do this transformation is as follows:

$$T = T_{n-1} * T_{n-2} * \ldots * T_i$$

2. Identify the regions where the dynamic objects are in the mosaic background. (Step 1)
3. Cover up the dynamic objects using the intensity of the selected frame.

Step 4: Create large Static Background

Mosaic background with dynamic objects

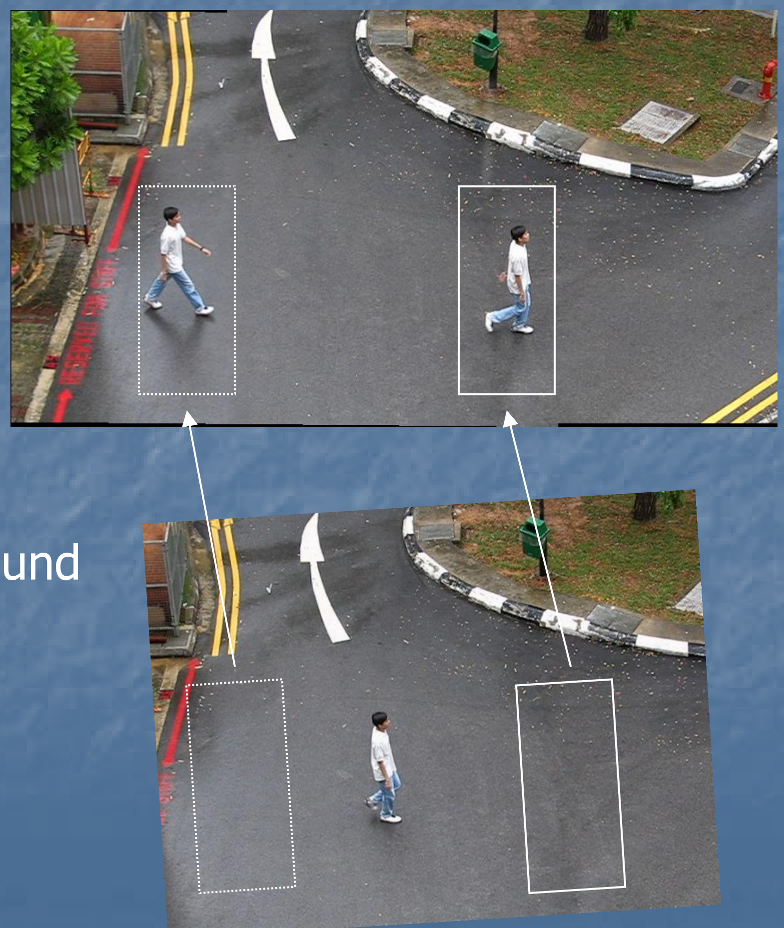
Step 4: Create large Static Background

Mosaic background with dynamic objects

1. Warp selected frame

Step 4: Create large Static Background

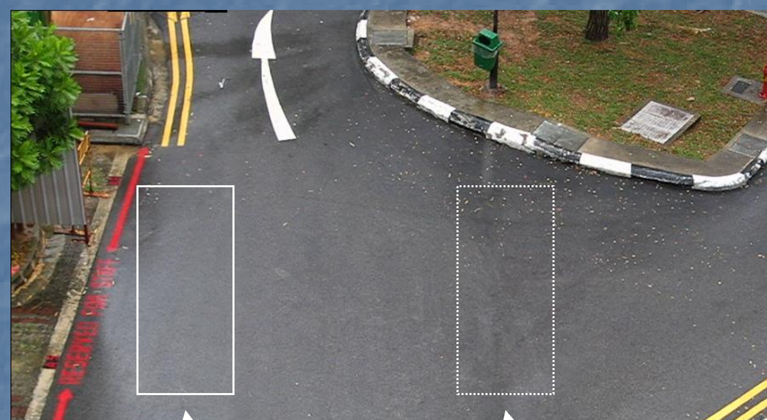
Mosaic background with dynamic objects



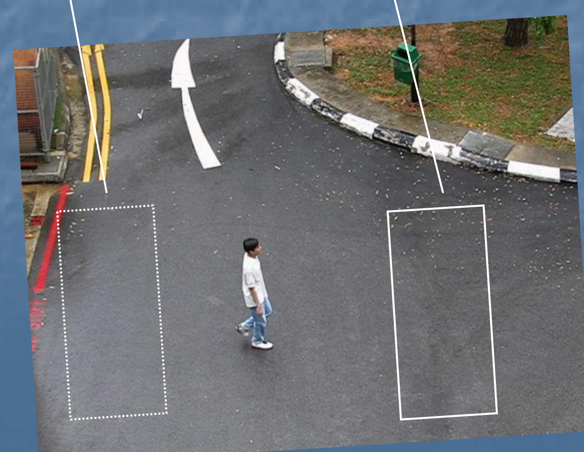
2. Identify region around dynamic objects

Step 4: Create large Static Background

Mosaic background without dynamic objects



3. Cover up the dynamic objects



Step 5: Blend dynamic Objects into background

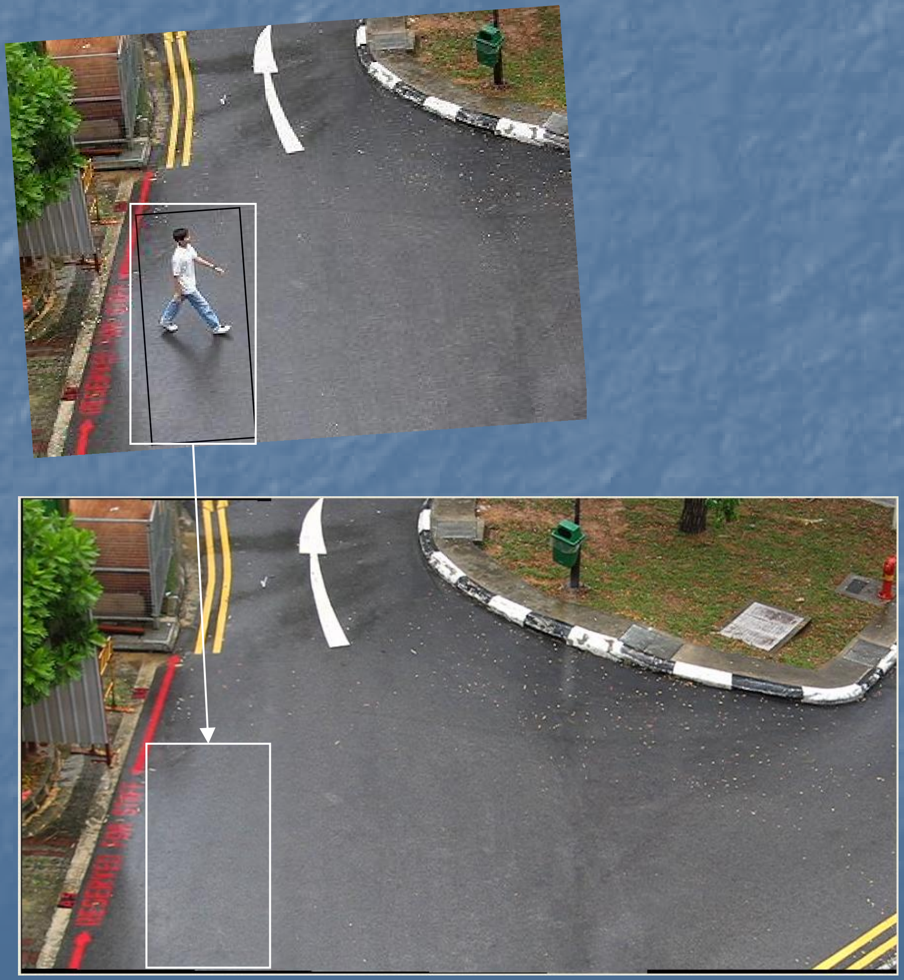
- Identify regions in the static background corresponding to dynamic objects in each frame

$$T_{i\text{-}bg} = T_{last\text{-}bg} * T_{i\text{-}last}$$

Step 5: Blend dynamic Objects into background

- Identify regions in the static background corresponding to dynamic objects in each frame

$$T_{i\text{-}bg} = T_{last\text{-}bg} * T_{i\text{-}last}$$

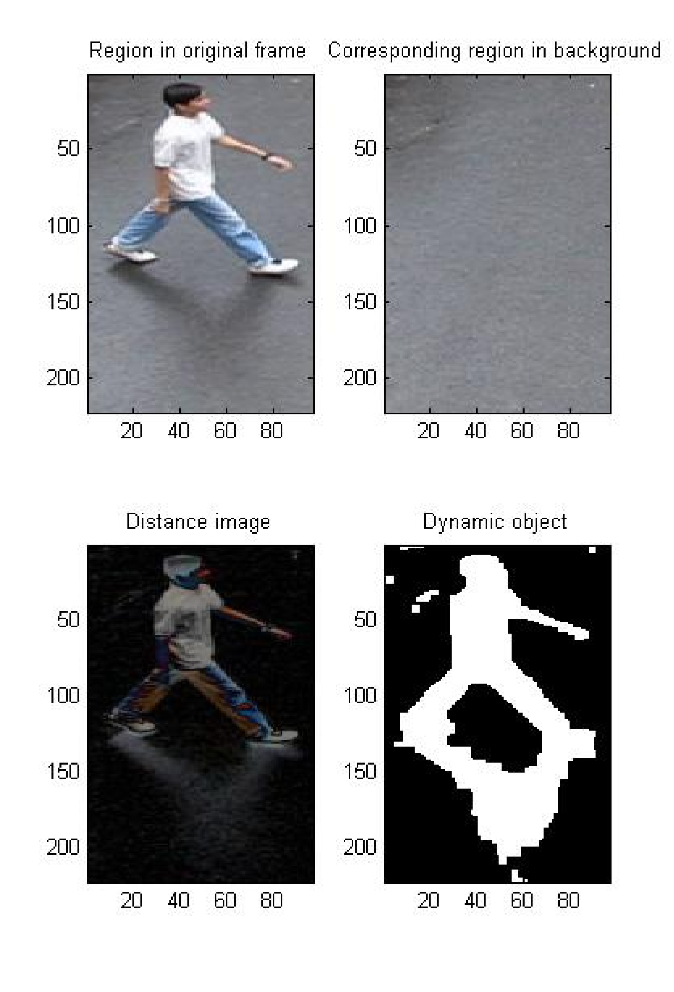


Step 5: Blend dynamic Objects into background

- Apply background removal to select the exact position of dynamic objects

  image subtraction + threshold

- Blend dynamic objects into background.



Conclusion

- Based on the assumptions we have been able to develop an almost fully automatic dynamic mosaic program
- We have applied the following computer vision techniques:
  - Features selection, tracking, registration, background removal, etc

Future work

- Apply projective transformation instead of affine transformation
- More sophisticated method for covering dynamic objects
- Remove the constraint of overlapping between first frame and last frame

References

- "An Introduction to 3D Vision", by Y. Ma, S. Soatto, J. Kosecka and S. Sastry (MASKS) http://vision.ucla.edu/MASKS/
- MATLAB Functions for Computer Vision and Image Analysis by Peter Kovesi http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/
- Mosaicing Tutorial

  http://www.pages.drexel.edu/~sis26/MosaickingTutorial.htm
- Shapiro & Stockman, *Computer Vision*, Prentice-Hall, 2001