Laboratory of
Image Processing

Face Detection & Tracking using KLT Algorithm

Pier Luigi Mazzeo

pierluigi.mazzeo@cnr.it

Institute of Applied Sciences and Intelligent Systems ScienceApp

National Research Council of Italy

Face Detection & Tracking

- This example shows how to automatically detect and track a face using feature points. The approach in this example keeps track of the face even when the person tilts his or her head, or moves toward or away from the camera.

- Object detection and tracking are important in many computer vision applications including activity recognition, automotive safety, and surveillance. In this example, you will develop a simple face tracking system by dividing the tracking problem into three separate problems:
    - Detect a face to track
    - Identify facial features to track
    - Track the face

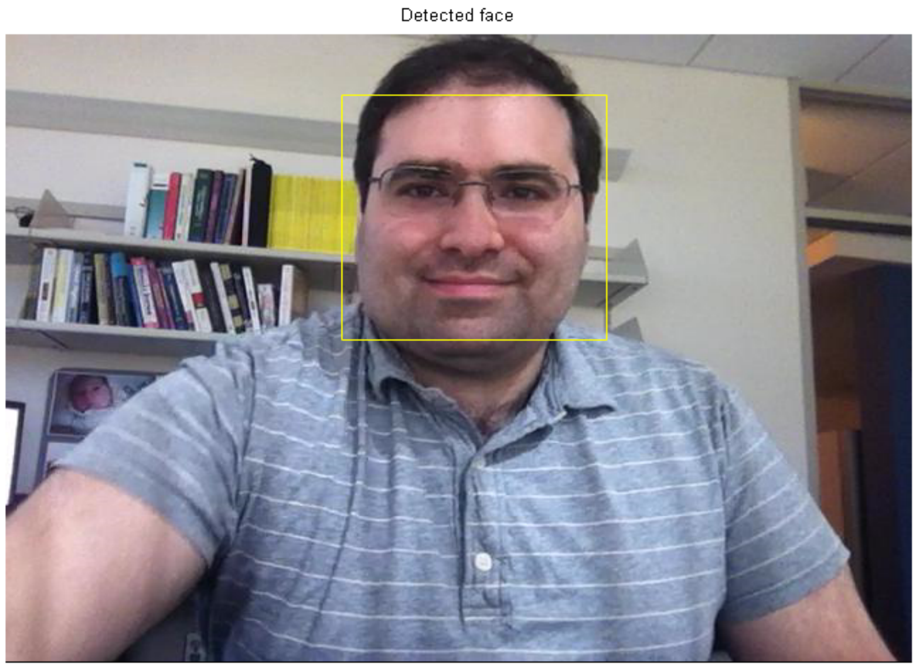
Detect a Face

First, you must detect the face. Use the vision.CascadeObjectDetector System object to detect the location of a face in a video frame. The cascade object detector uses the Viola-Jones detection algorithm and a trained classification model for detection. By default, the detector is configured to detect faces, but it can be used to detect other types of objects.

```
% Create a cascade detector object.
faceDetector = vision.CascadeObjectDetector();

% Read a video frame and run the face detector.
videoFileReader = vision.VideoFileReader('tilted_face.avi');
videoFrame      = step(videoFileReader);
bbox            = step(faceDetector, videoFrame);

% Convert the first box to a polygon.
% This is needed to be able to visualize the rotation of the
object.
x = bbox(1, 1); y = bbox(1, 2); w = bbox(1, 3); h = bbox(1,
4);
bboxPolygon = [x, y, x+w, y, x+w, y+h, x, y+h];
```

Detect a Face

```
% Draw the returned bounding box around the detected face.
videoFrame = insertShape(videoFrame, 'Polygon',
bboxPolygon);
figure; imshow(videoFrame); title('Detected face');
```



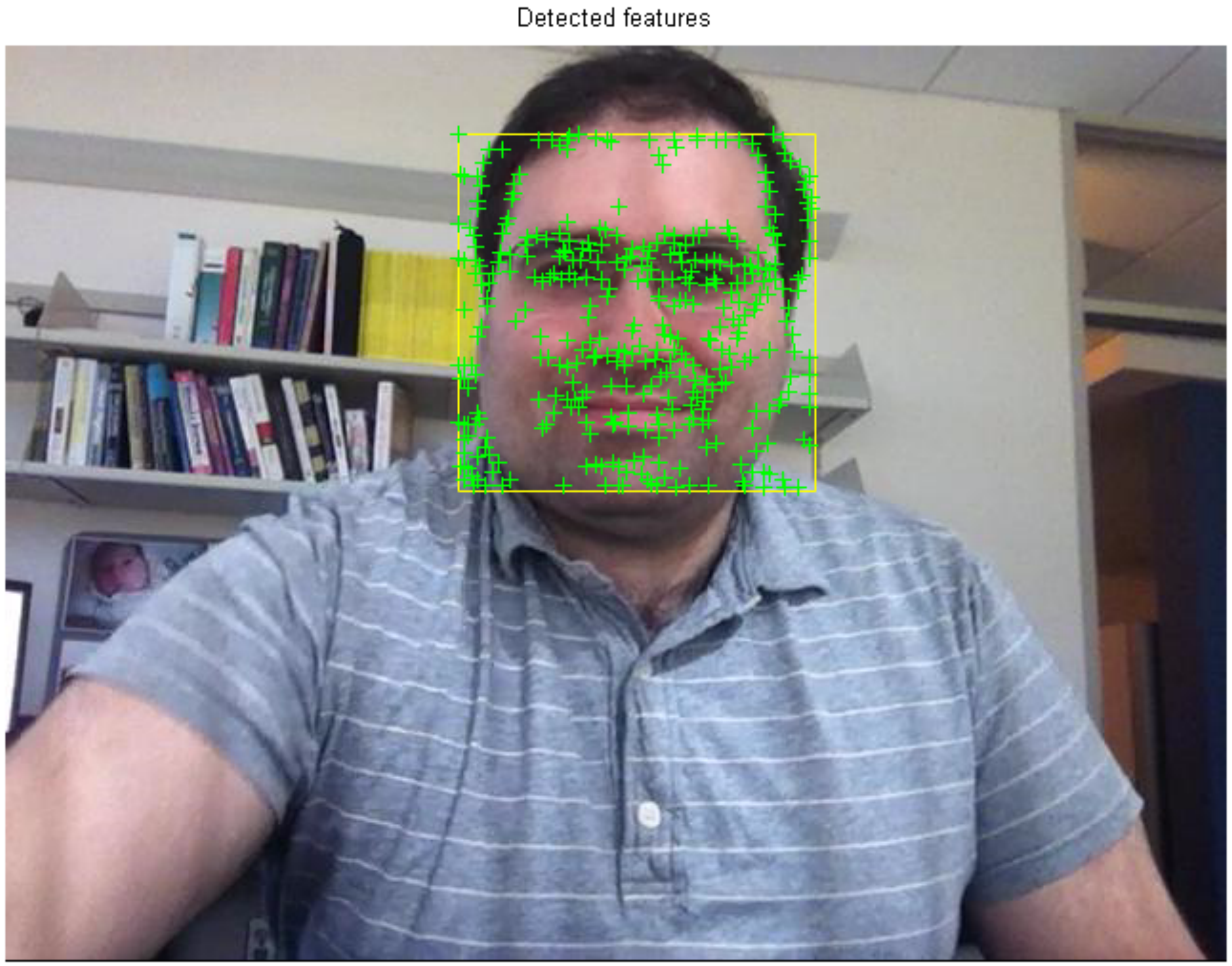
- To track the face over time, this example uses the Kanade-Lucas-Tomasi (KLT) algorithm. While it is possible to use the cascade object detector on every frame, it is computationally expensive. It may also fail to detect the face, when the subject turns or tilts his head. This limitation comes from the type of trained classification model used for detection. The example detects the face only once, and then the KLT algorithm tracks the face across the video frames.

Identify Facial Features To Track

The KLT algorithm tracks a set of feature points across the video frames. Once the detection locates the face, the next step in the example identifies feature points that can be reliably tracked. This example uses the standard, “good features to track”proposed by Shi and Tomasi.

```
% Detect feature points in the face region.
points = detectMinEigenFeatures(rgb2gray(videoFrame), 'ROI',
bbox);
% Display the detected points.
figure, imshow(videoFrame), hold on, title('Detected
features');
plot(points);
```

Identify Facial Features To Track



Initialize a Tracker to Track the Points

- With the feature points identified, you can now use the `vision.PointTracker` System object to track them. For each point in the previous frame, the point tracker attempts to find the corresponding point in the current frame. Then the estimateGeometricTransform function is used to estimate the translation, rotation, and scale between the old points and the new points. This transformation is applied to the bounding box around the face.

```
% Create a point tracker and enable the bidirectional
error constraint to make it more robust in the presence
of noise and clutter.
pointTracker =
vision.PointTracker('MaxBidirectionalError', 2);

% Initialize the tracker with the initial point locations
and the initial video frame.
points = points.Location;
initialize(pointTracker, points, rgb2gray(videoFrame));
```

Initialize a Video Player to Display the Results

- Create a video player object for displaying video frames.

```
videoInfo    = info(videoFileReader);
videoPlayer  = vision.VideoPlayer('Position',...
    [100 100 videoInfo.VideoSize(1:2)+30]);
```

Initialize a Geometric Transform Estimator

The estimateGeometricTransform function calculates the translation, rotation, and scale of the tracked face between frames. It computes the transformation between point locations in the previous frame, and corresponding locations in the current frame. The results are used to characterize the motion of the face.

```
% Make a copy of the points to be used for computing the geometric
% transformation between the points in the previous and the current frames
oldPoints = points;
```

Track the Points

```
while ~isDone(videoFileReader)
    % get the next frame
    videoFrame = step(videoFileReader);

    % Track the points. Note that some points may be lost.
    [points, isFound] = step(pointTracker, rgb2gray(videoFrame));
    visiblePoints = points(isFound, :);
    oldInliers = oldPoints(isFound, :);

if size(visiblePoints, 1) >= 2 % need at least 2 points

        % Estimate the geometric transformation between the old
points and the new points and eliminate outliers
        [xform, oldInliers, visiblePoints] =
estimateGeometricTransform(...
            oldInliers, visiblePoints, 'similarity',
'MaxDistance', 4);
```

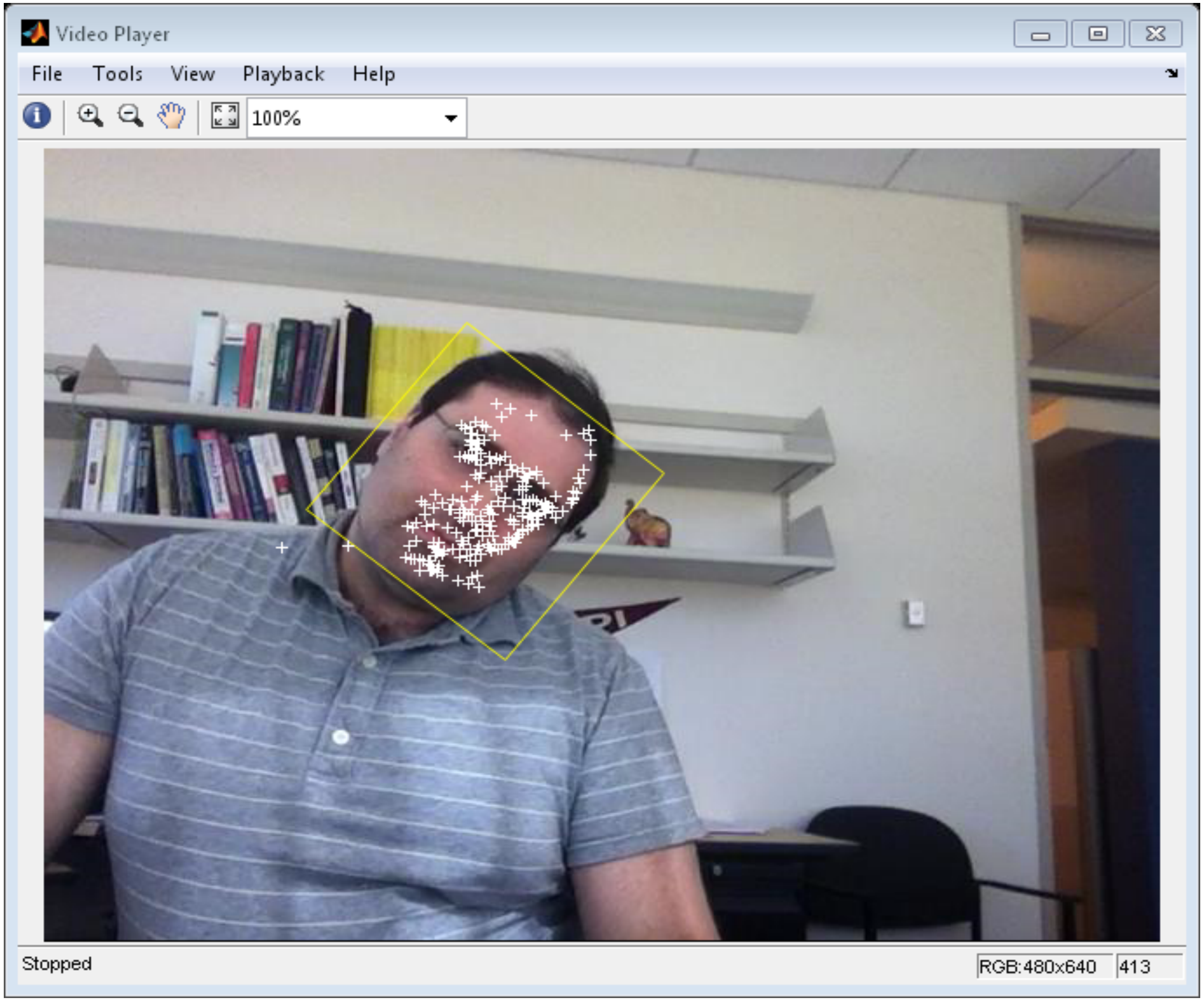
Track the Points

```
 % Apply the transformation to the bounding box
       [bboxPolygon(1:2:end), bboxPolygon(2:2:end)] ...
           = transformPointsForward(xform,
bboxPolygon(1:2:end), bboxPolygon(2:2:end));

       % Insert a bounding box around the object being tracked
  videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon);

       % Display tracked points
  videoFrame = insertMarker(videoFrame, visiblePoints, '+', ...
           'Color', 'white');

       % Reset the points
       oldPoints = visiblePoints;
       setPoints(pointTracker, oldPoints);
    end
```

Track the Points

```
% Display the annotated video frame using the
video player object
    step(videoPlayer, videoFrame);
end

% Clean up
release(videoFileReader);
release(videoPlayer);
release(pointTracker);
```

Track the Points



Track the Points

Summary

- In this example, you created a simple face tracking system that automatically detects and tracks a single face. Try changing the input video, and see if you are still able to detect and track a face. Make sure the person is facing the camera in the initial frame for the detection step.

References

- Viola, Paul A. and Jones, Michael J. "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.
- Bruce D. Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence, 1981.
- Carlo Tomasi and Takeo Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991.