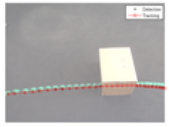

January 2013 "Computer Vision with MATLAB" webinar demo files

by Bruce Tannenbaum
05 Feb 2013

MATLAB code used in the computer vision webinar held on January 29, 2013.

visionfacetrackingKLT.m

```
%% Face Detection and Tracking Using the KLT Algorithm
% 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.
%
%   Copyright 2013 The MathWorks, Inc.

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

% Open a video file
videoFileReader = vision.VideoFileReader('tilted_face.avi');

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

% Convert the box to a polygon. This is needed to be able to visualize the
```

Highlights from January 2013 "Computer Vision with MATLAB" webinar demo files

playControls()
Copyright 2013 The MathWorks, Inc.
MatchCard.m

ForegroundDetection.m
Copyright 2013 The MathWorks, Inc.
FacePeopleDetection.m

```
% rotation of the object.
x = bbox(1); y = bbox(2); w = bbox(3); h = bbox(4);
bboxPolygon = [x, y, x+w, y, x+w, y+h, x, y+h];

% Draw the returned bounding box around the detected face.
shapeInserter  = vision.ShapeInserter('Shape', 'Polygons', 'BorderColor','Custom',...
    'CustomBorderColor',[255 255 0]);
videoFrame = step(shapeInserter, videoFrame, bboxPolygon);
figure; imshow(videoFrame); title('Detected face');

%% Identify Facial Features To Track
% Crop out the region of the image containing the face, and detect the
% feature points inside it.
cornerDetector = vision.CornerDetector('Method', ...
    'Minimum eigenvalue (Shi & Tomasi)');
points = step(cornerDetector, rgb2gray(imcrop(videoFrame, bbox)));

% The coordinates of the feature points are with respect to the cropped
% region. They need to be translated back into the original image
% coordinate system.
points = double(points);
points(:, 1) = points(:, 1) + double(bbox(1));
points(:, 2) = points(:, 2) + double(bbox(2));

% Display the detected points.
markerInserter = vision.MarkerInserter('Shape', 'Plus', ...
    'BorderColor', 'White');
videoFrame = step(markerInserter, videoFrame, points);
figure, imshow(videoFrame), title('Detected features');

%% Initialize a Tracker to Track the Points
% 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.
initialize(pointTracker, double(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]);
```


visionfacetrackingKLT.m

Face Detection and Tracking Using the KLT Algorithm

View all files

```
%% Initialize a Geometric Transform Estimator
geometricTransformEstimator = vision.GeometricTransformEstimator(...
    'PixelDistanceThreshold', 4, 'Transform', 'Nonreflective similarity');

% 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 = double(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 ~isempty(visiblePoints)
        % Estimate the geometric transformation between the old points
        % and the new points.
        [xform, geometricInlierIdx] = step(geometricTransformEstimator, ...
            double(oldInliers), double(visiblePoints));

        % Eliminate outliers
        visiblePoints = visiblePoints(geometricInlierIdx, :);
        oldInliers = oldInliers(geometricInlierIdx, :);

        % Apply the transformation to the bounding box
        boxPoints = [reshape(bboxPolygon, 2, 4)', ones(4, 1)];
        boxPoints = boxPoints * xform;
        bboxPolygon = reshape(boxPoints', 1, numel(boxPoints));

        % Insert a bounding box around the object being tracked
        videoFrame = step(shapeInserter, videoFrame, bboxPolygon);

        % Display tracked points
        videoFrame = step(markerInserter, videoFrame, visiblePoints);

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

    % Display the annotated video frame using the video player object
```

```
    step(videoPlayer, videoFrame);
end

%% Clean up
release(videoFileReader);
release(videoPlayer);
release(geometricTransformEstimator);
release(pointTracker);
close all
```

Contact us