分析和提升

执行图像统计，FIR滤波，频率和霍夫变换，形态，对比度增强和噪声去除

分析和增强技术使您能够提高信噪比和强调功能。 工具箱功能包括过滤，多维滤波，以及执行卷积和相关。

## 算法进行面部识别和追踪

这个例子展示了如何通过细节点自动识别和追踪面部。例子里的方法可以持续追踪面部甚至是当人倾斜ta的头，或者是拉近或远离相机。

介绍

物体的识别和追踪在众多计算机图像应用中都很重要，包括活动辨别，汽车安全，和监控。在这个例子里，你将会学到一个简单的面部识别系统，它将识别过程分为了三个部分。

1. 检测面部
2. 识别面部细节并追踪
3. 追踪面部

检测面部

首先，你必须检测到面部。用vision.CascadeObjectDetector系统来探测脸在摄像机框架内的位置。那个系统用Viola-Jones识别算法和一个为检测设计的分类模板。默认情况下，侦探是探测面部的，但是它也可以被用于探测其他类型物体。

% 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);

% Draw the returned bounding box around the detected face.

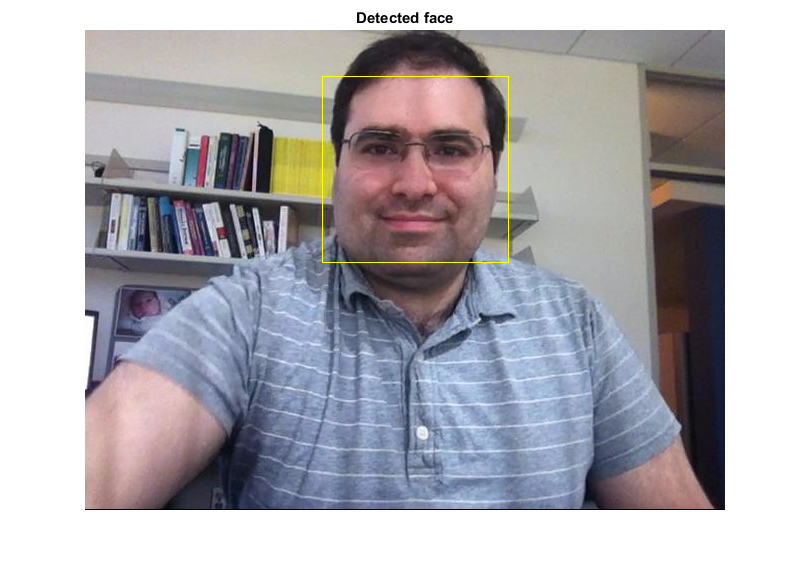
videoFrame = insertShape(videoFrame, 'Rectangle', bbox);

figure; imshow(videoFrame); title('Detected face');

% Convert the first box into a list of 4 points

% This is needed to be able to visualize the rotation of the object.

bboxPoints = bbox2points(bbox(1, :));



为了能时刻追踪面部，这个例子用了Kanade-Lucas-Tomasi (KLT)算法。当可以在所有框架里应用串联物体侦探时，计算量是很庞大的。它有可能检测脸部失败，当转动或倾斜头部的时候。这个限制来自于为检测设计的分类模型。这个例子只能瞬间检测面部，然后，KLT算法会在摄像机框架中追踪面部

面部细节的识别

KLT算法追踪一系列的摄像机框架内的细节点。一旦探测器定位了面部，在例子里的下一个步骤就是识别那些可以被追踪的细节点。这个例子用了标准的被Shi和Tomasi推荐的，“追踪的好细节”。

% 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);



将追踪器初始化来追踪点

用被识别的细节点，你可以用vision.PointTracker（点的追踪器）系统来追踪他们。对于每一个在之前框架里的点，点追踪器尝试去找到现在框架内一致的点。之后estimateGeometricTransform（估计几何变换）功能将被用来估计新旧点之间的变换，旋转和比例。这个变换被应用于脸部附近的识别框。

% 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, videoFrame);

初始化摄像机播放器来呈现结果

创建一个摄像机播放器展示计算机框架

videoPlayer = vision.VideoPlayer('Position',...

[100 100 [size(videoFrame, 2), size(videoFrame, 1)]+30]);

追踪面部

从一个框到另一个框中追踪细节点，然后用estimateGeometricTransform功能来测算面部的移动

% 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;

while ~isDone(videoFileReader)

% get the next frame

videoFrame = step(videoFileReader);

% Track the points. Note that some points may be lost.

[points, isFound] = step(pointTracker, 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);

% Apply the transformation to the bounding box points

bboxPoints = transformPointsForward(xform, bboxPoints);

% Insert a bounding box around the object being tracked

bboxPolygon = reshape(bboxPoints', 1, []);

videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, ...

'LineWidth', 2);

% Display tracked points

videoFrame = insertMarker(videoFrame, visiblePoints, '+', ...

'Color', 'white');

% Reset the points

oldPoints = visiblePoints;

setPoints(pointTracker, oldPoints);

end

% Display the annotated video frame using the video player object

step(videoPlayer, videoFrame);

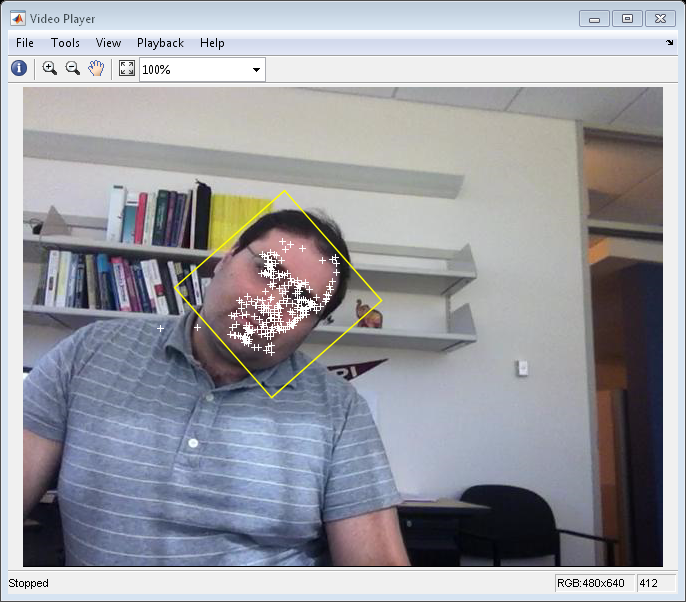
end

% Clean up

release(videoFileReader);

release(videoPlayer);

release(pointTracker);



总结

在这个例子中，你可以创造一个简单的面部追踪系统，可以自动探测和追踪一个单个的脸。尝试变换输入的相机，看看是不是仍然可以检测并追踪到面部。确定那个人在检测步骤中在初始框框里面对摄像机

引用

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.

Jianbo Shi and Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, 1994.

Zdenek Kalal, Krystian Mikolajczyk and Jiri Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. International Conference on Pattern Recognition, 2010

## 用现场摄像机获取来进行面部检测和追踪

这个例子展示了如何用KLT算法在现场的摄像机中检测和追踪面部

概述

对象检测和跟踪在许多计算机视觉应用中是重要的，包括活动识别，汽车安全和监视。在本例中，您将开发一个简单的系统，用于在网络摄像头捕获的实时视频流中跟踪单个面部。 MATLAB通过硬件支持包提供网络摄像头支持，您需要下载和安装才能运行此示例。支持包可通过支持包安装程序获得。

在该示例中的面部跟踪系统可以是两种模式之一：检测或跟踪。在检测模式下，您可以使用vision.CascadeObjectDetector对象来检测当前帧中的面。如果检测到面部，则必须检测面部上的角点，初始化vision.PointTracker对象，然后切换到跟踪模式。

在跟踪模式下，您必须使用点跟踪器跟踪点。当您跟踪点时，其中一些将因为遮挡而丢失。如果被跟踪的点的数量落在阈值以下，这意味着不再跟踪该面部。然后必须切换回检测模式以尝试重新获取脸部。

设置

创建一个物体来检测面部，追踪点，获取和展现摄像机框架

% Create the face detector object.

faceDetector = vision.CascadeObjectDetector();

% Create the point tracker object.

pointTracker = vision.PointTracker('MaxBidirectionalError', 2);

% Create the webcam object.

cam = webcam();

% Capture one frame to get its size.

videoFrame = snapshot(cam);

frameSize = size(videoFrame);

% Create the video player object.

videoPlayer = vision.VideoPlayer('Position', [100 100 [frameSize(2), frameSize(1)]+30]);

检测和追踪

捕获和网络摄像机框架过程是一个检测的循环，然后追踪面部。这个循环将会进行400个框架或直到播放器窗口关闭

runLoop = true;

numPts = 0;

frameCount = 0;

while runLoop && frameCount < 400

% Get the next frame.

videoFrame = snapshot(cam);

videoFrameGray = rgb2gray(videoFrame);

frameCount = frameCount + 1;

if numPts < 10

% Detection mode.

bbox = faceDetector.step(videoFrameGray);

if ~isempty(bbox)

% Find corner points inside the detected region.

points = detectMinEigenFeatures(videoFrameGray, 'ROI', bbox(1, :));

% Re-initialize the point tracker.

xyPoints = points.Location;

numPts = size(xyPoints,1);

release(pointTracker);

initialize(pointTracker, xyPoints, videoFrameGray);

% Save a copy of the points.

oldPoints = xyPoints;

% Convert the rectangle represented as [x, y, w, h] into an

% M-by-2 matrix of [x,y] coordinates of the four corners. This

% is needed to be able to transform the bounding box to display

% the orientation of the face.

bboxPoints = bbox2points(bbox(1, :));

% Convert the box corners into the [x1 y1 x2 y2 x3 y3 x4 y4]

% format required by insertShape.

bboxPolygon = reshape(bboxPoints', 1, []);

% Display a bounding box around the detected face.

videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);

% Display detected corners.

videoFrame = insertMarker(videoFrame, xyPoints, '+', 'Color', 'white');

end

else

% Tracking mode.

[xyPoints, isFound] = step(pointTracker, videoFrameGray);

visiblePoints = xyPoints(isFound, :);

oldInliers = oldPoints(isFound, :);

numPts = size(visiblePoints, 1);

if numPts >= 10

% Estimate the geometric transformation between the old points

% and the new points.

[xform, oldInliers, visiblePoints] = estimateGeometricTransform(...

oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);

% Apply the transformation to the bounding box.

bboxPoints = transformPointsForward(xform, bboxPoints);

% Convert the box corners into the [x1 y1 x2 y2 x3 y3 x4 y4]

% format required by insertShape.

bboxPolygon = reshape(bboxPoints', 1, []);

% Display a bounding box around the face being tracked.

videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);

% Display tracked points.

videoFrame = insertMarker(videoFrame, visiblePoints, '+', 'Color', 'white');

% Reset the points.

oldPoints = visiblePoints;

setPoints(pointTracker, oldPoints);

end

end

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

step(videoPlayer, videoFrame);

% Check whether the video player window has been closed.

runLoop = isOpen(videoPlayer);

end

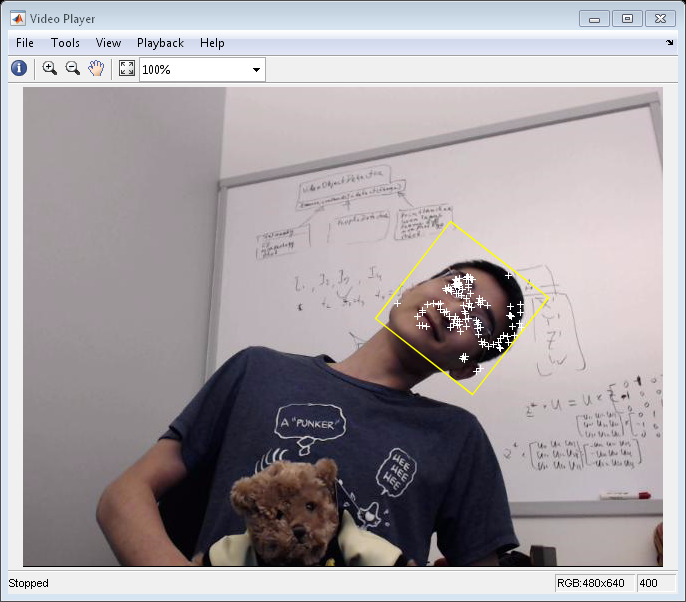
% Clean up.

clear cam;

release(videoPlayer);

release(pointTracker);

release(faceDetector);



引用

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.

Jianbo Shi and Carlo Tomasi. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition, 1994.

Zdenek Kalal, Krystian Mikolajczyk and Jiri Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. International Conference on Pattern Recognition, 2010

## 基于动作的多重物体追踪

这个例子展示了如何演示自动检测并基于动作追踪在摄像机中移动的物体

检测物体的移动和基于动作追踪是很多计算机视觉应用中很重要的能力，包括活动识别和交通疏导还有汽车安全。基于动作来追踪物体的问题可以分为两个部分：

1. 在每个框框中检测移动物体
2. 随时联系相同物体间检测的相同的点

对于移动物体的检测用了背景差值算法，基于Gaussian混合模板。形态学操作被应用在排除干扰以得到最显著的面部。最终，一点分析有关联的可能与移动物体一致的像素点检测组。

检测器对相同物体的关联仅仅基于移动。每个追踪的移动都被Kalman filter测算。过滤器被用于预测每个框框中追踪的位置，并决定每个被追踪指定的探测器的可能性。

在这个例子中，探测器的维修护理成为了一个重要的方面。在每个给定的框架中，一些检测器可能被追踪过程制定，同时其他检测器和追踪可能仍然没被指定。被制定的追踪会被一致的检测覆盖。没被指定的会被无形标记。没被指定的检测器将开始一个新的追踪。

了解更多信息，请看Multiple Object Tracking

这个例子是一个在顶端有主体的功能，并且帮助的形式如下：

function multiObjectTracking()

% Create System objects used for reading video, detecting moving objects,

% and displaying the results.

obj = setupSystemObjects();

tracks = initializeTracks(); % Create an empty array of tracks.

nextId = 1; % ID of the next track

% Detect moving objects, and track them across video frames.

while ~isDone(obj.reader)

frame = readFrame();

[centroids, bboxes, mask] = detectObjects(frame);

predictNewLocationsOfTracks();

[assignments, unassignedTracks, unassignedDetections] = ...

detectionToTrackAssignment();

updateAssignedTracks();

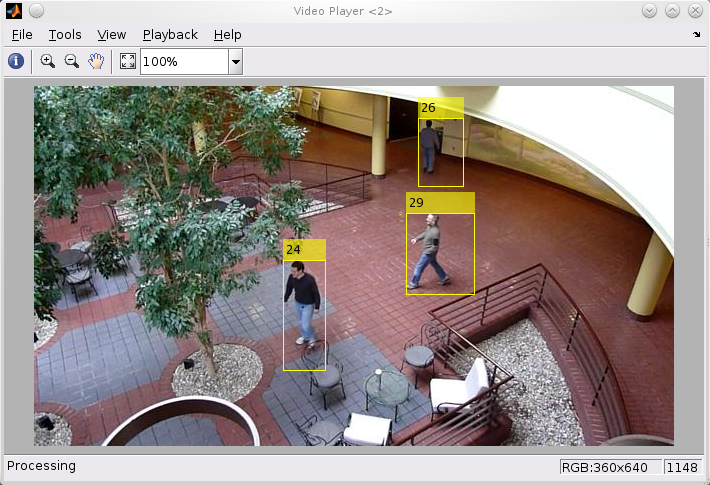
updateUnassignedTracks();

deleteLostTracks();

createNewTracks();

displayTrackingResults();

end



创建一个物体系统

创建一个系统用来读取摄像机框框，检测显著的物体，并展示结果

function obj = setupSystemObjects()

% Initialize Video I/O

% Create objects for reading a video from a file, drawing the tracked

% objects in each frame, and playing the video.

% Create a video file reader.

obj.reader = vision.VideoFileReader('atrium.mp4');

% Create two video players, one to display the video,

% and one to display the foreground mask.

obj.videoPlayer = vision.VideoPlayer('Position', [20, 400, 700, 400]);

obj.maskPlayer = vision.VideoPlayer('Position', [740, 400, 700, 400]);

% Create System objects for foreground detection and blob analysis

% The foreground detector is used to segment moving objects from

% the background. It outputs a binary mask, where the pixel value

% of 1 corresponds to the foreground and the value of 0 corresponds

% to the background.

obj.detector = vision.ForegroundDetector('NumGaussians', 3, ...

'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.7);

% Connected groups of foreground pixels are likely to correspond to moving

% objects. The blob analysis System object is used to find such groups

% (called 'blobs' or 'connected components'), and compute their

% characteristics, such as area, centroid, and the bounding box.

obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort', true, ...

'AreaOutputPort', true, 'CentroidOutputPort', true, ...

'MinimumBlobArea', 400);

end

初始化追踪

initializeTracks功能创建了一个追踪的箭，每一个追踪都是一个在摄像机中展现的移动的物体。这个结构的目的是概括追踪物体的陈述。陈述包括为了用来追踪的检测的信息，追踪结果和展示

结构包括以下几个方面

Id：追踪的数字

Bbox：物体的现在的边界框，用于展示

kalmanFilter：用于基于移动的追踪

age：从第一次检测并追踪开始，框框的数量

totalvisiblecount：在被检测并追踪的过程中所有的框框数量

consecutiveinvisiblecount：没被检测到的追踪产生的连贯的框框数量

干扰检测倾向于得到短期追踪的结果。由于这个原因，这个例子仅仅展示了一个在被几个框追踪后的物体。这个当totalvisiblecount超过一个具体的临界值时发生。

当没有检测器被被几个连贯的框框追踪联系时，这个例子假设了物体丢失了视野并检测到了追踪。这个发生在consecutiveInvisibleCount超过了某个临界值时。一个追踪可能也被当成干扰删除，如果这个追踪时间太短并标记了很多不可见的框框。

function tracks = initializeTracks()

% create an empty array of tracks

tracks = struct(...

'id', {}, ...

'bbox', {}, ...

'kalmanFilter', {}, ...

'age', {}, ...

'totalVisibleCount', {}, ...

'consecutiveInvisibleCount', {});

end

读取摄像机框框

从视频文件中读取视频框框

function frame = readFrame()

frame = obj.reader.step();

end

检测物体

detectObjects功能返回到图心和检测物体的边界框。它也返回到二进制的面部，和输入框大小一样。像素点伴随着数值为1的和前景同样的点，和像素点数值为0的和背景一致的点

这个功能展示了用前景检测移动分割。它之后展示了在得出二进制面具来减少干扰像素点的形态学操作和填充洞洞。

function [centroids, bboxes, mask] = detectObjects(frame)

% Detect foreground.

mask = obj.detector.step(frame);

% Apply morphological operations to remove noise and fill in holes.

mask = imopen(mask, strel('rectangle', [3,3]));

mask = imclose(mask, strel('rectangle', [15, 15]));

mask = imfill(mask, 'holes');

% Perform blob analysis to find connected components.

[~, centroids, bboxes] = obj.blobAnalyser.step(mask);

end

预测出现的追踪的新位置

用Kalman过滤器来预测每个在当前框里的追踪的图心，并更新边界点

function predictNewLocationsOfTracks()

for i = 1:length(tracks)

bbox = tracks(i).bbox;

% Predict the current location of the track.

predictedCentroid = predict(tracks(i).kalmanFilter);

% Shift the bounding box so that its center is at

% the predicted location.

predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;

tracks(i).bbox = [predictedCentroid, bbox(3:4)];

end

end

给追踪分配检测器

在当前的框架中给出现的追踪分配检测器被极小的花费完成。这个花费被定义为消极的给追踪的检测器一致的记录可能性

算法分为两步

第一步：计算用vision.KalmanFilter系统的距离步骤来给每个追踪分配检测器的花费。距离计算预测的追踪图像图心和检测器图心的距离。它也包括预测的自信，通过Kalman过滤器持续。结果被储存在M\*N矩阵，M是追踪数量，N是检测器数量。

第二步：解决用assignDetectionsToTracks功能花费矩阵的分配问题。这个功能带来矩阵的花费，和不分配检测器给追踪的花费

后者的价值依赖于distance method of the vision.KalmanFilter.返回的价值范围。这个价值一定被多次试验后调好。设置太低增加了创建新追踪的可能性，可能导致追踪碎片。设置太高导致单一追踪，将一个物体分离

assignDetectionsToTracks功能用Hungarian算法的Munkres’ 视觉来计算分配的最小总价值。它返回M\*2矩阵包含一致的表明分配的追踪和检测器分两列。它也回到检测器和没被分配的追踪目录

function [assignments, unassignedTracks, unassignedDetections] = ...

detectionToTrackAssignment()

nTracks = length(tracks);

nDetections = size(centroids, 1);

% Compute the cost of assigning each detection to each track.

cost = zeros(nTracks, nDetections);

for i = 1:nTracks

cost(i, :) = distance(tracks(i).kalmanFilter, centroids);

end

% Solve the assignment problem.

costOfNonAssignment = 20;

[assignments, unassignedTracks, unassignedDetections] = ...

assignDetectionsToTracks(cost, costOfNonAssignment);

end

更新追踪分配

updateAssignedTracks功能更新每一个被分配的追踪和一致的检测器。它是正确计算位置的vision.KalmanFilter的正确步骤。之后，它储存在新的边界框里，增加追踪年龄和总体被数为1的可见数。最后不可见的数成0

function updateAssignedTracks()

numAssignedTracks = size(assignments, 1);

for i = 1:numAssignedTracks

trackIdx = assignments(i, 1);

detectionIdx = assignments(i, 2);

centroid = centroids(detectionIdx, :);

bbox = bboxes(detectionIdx, :);

% Correct the estimate of the object's location

% using the new detection.

correct(tracks(trackIdx).kalmanFilter, centroid);

% Replace predicted bounding box with detected

% bounding box.

tracks(trackIdx).bbox = bbox;

% Update track's age.

tracks(trackIdx).age = tracks(trackIdx).age + 1;

% Update visibility.

tracks(trackIdx).totalVisibleCount = ...

tracks(trackIdx).totalVisibleCount + 1;

tracks(trackIdx).consecutiveInvisibleCount = 0;

end

end

更新未被分配的追踪

function updateUnassignedTracks()

for i = 1:length(unassignedTracks)

ind = unassignedTracks(i);

tracks(ind).age = tracks(ind).age + 1;

tracks(ind).consecutiveInvisibleCount = ...

tracks(ind).consecutiveInvisibleCount + 1;

end

end

删除丢失的追踪

deleteLostTracks功能删除很多不可兼得连续框架的追踪。他也删除刚刚创建的被很多不可见框框覆盖的追踪

function deleteLostTracks()

if isempty(tracks)

return;

end

invisibleForTooLong = 20;

ageThreshold = 8;

% Compute the fraction of the track's age for which it was visible.

ages = [tracks(:).age];

totalVisibleCounts = [tracks(:).totalVisibleCount];

visibility = totalVisibleCounts ./ ages;

% Find the indices of 'lost' tracks.

lostInds = (ages < ageThreshold & visibility < 0.6) | ...

[tracks(:).consecutiveInvisibleCount] >= invisibleForTooLong;

% Delete lost tracks.

tracks = tracks(~lostInds);

end

创建一个新的追踪

从未被分配的检测器里创建一个新追踪。假设任何未被分配的检测器开始于新追踪。事实上，你可以用其他办法测算干扰。

function createNewTracks()

centroids = centroids(unassignedDetections, :);

bboxes = bboxes(unassignedDetections, :);

for i = 1:size(centroids, 1)

centroid = centroids(i,:);

bbox = bboxes(i, :);

% Create a Kalman filter object.

kalmanFilter = configureKalmanFilter('ConstantVelocity', ...

centroid, [200, 50], [100, 25], 100);

% Create a new track.

newTrack = struct(...

'id', nextId, ...

'bbox', bbox, ...

'kalmanFilter', kalmanFilter, ...

'age', 1, ...

'totalVisibleCount', 1, ...

'consecutiveInvisibleCount', 0);

% Add it to the array of tracks.

tracks(end + 1) = newTrack;

% Increment the next id.

nextId = nextId + 1;

end

end

展现追踪结果

displayTrackingResults功能画了一个边界框，标签ID给每个视频框里的追踪

function displayTrackingResults()

% Convert the frame and the mask to uint8 RGB.

frame = im2uint8(frame);

mask = uint8(repmat(mask, [1, 1, 3])) .\* 255;

minVisibleCount = 8;

if ~isempty(tracks)

% Noisy detections tend to result in short-lived tracks.

% Only display tracks that have been visible for more than

% a minimum number of frames.

reliableTrackInds = ...

[tracks(:).totalVisibleCount] > minVisibleCount;

reliableTracks = tracks(reliableTrackInds);

% Display the objects. If an object has not been detected

% in this frame, display its predicted bounding box.

if ~isempty(reliableTracks)

% Get bounding boxes.

bboxes = cat(1, reliableTracks.bbox);

% Get ids.

ids = int32([reliableTracks(:).id]);

% Create labels for objects indicating the ones for

% which we display the predicted rather than the actual

% location.

labels = cellstr(int2str(ids'));

predictedTrackInds = ...

[reliableTracks(:).consecutiveInvisibleCount] > 0;

isPredicted = cell(size(labels));

isPredicted(predictedTrackInds) = {' predicted'};

labels = strcat(labels, isPredicted);

% Draw the objects on the frame.

frame = insertObjectAnnotation(frame, 'rectangle', ...

bboxes, labels);

% Draw the objects on the mask.

mask = insertObjectAnnotation(mask, 'rectangle', ...

bboxes, labels);

end

end

% Display the mask and the frame.

obj.maskPlayer.step(mask);

obj.videoPlayer.step(frame);

end

总结

这个例子创建了一个基于动作系统的检测和追踪多物体移动系统。尝试用不同的摄像机看看还能不能检测和追踪物体。尝试修改检测器，分配和删除步骤的参数

在这个例子里的追踪仅仅基于动作，假设所有物体都以恒定速度沿直线运动。当物体移动脱离模型，这个例子可能会出错。注意追踪人物标签#12的错误当他被树挡上时。

追踪错误的可能性可以通过用更多的复杂的移动模型减小，想匀变速，或给每个物体用复合Kalman过滤器。也可以通过其他方法随时联系检测器，像型号，形状和颜色。

## 用Kalman 过滤器来追踪物体

此示例显示如何使用vision.KalmanFilter对象和configureKalmanFilter函数来跟踪对象。

这个例子是一个函数，它的主体在顶部和辅助程序以嵌套函数的形式在下面。

介绍

卡尔曼滤波器具有许多用途，包括在控制，导航，计算机视觉和时间序列计量经济学中的应用。 本示例说明了如何使用卡尔曼滤波器跟踪对象并关注三个重要功能：

预测对象的未来位置

减少由不准确检测引入的噪声

促进多个对象与其轨迹的关联过程

对象跟踪的挑战

在展示使用卡尔曼滤波器之前，让我们首先检查在视频中跟踪对象的挑战。 下面的视频显示一个绿色的球在地板上从左到右移动。

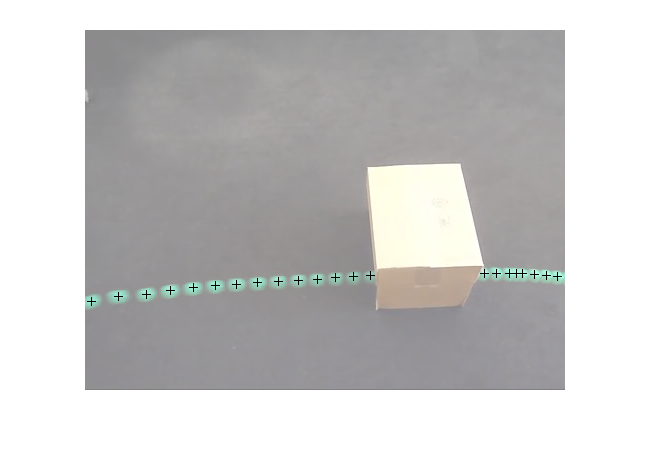
showDetections();



球上的白色区域突出显示使用vision.ForegroundDetector检测到的像素，该操作将移动对象与背景分离。 背景减法仅找到球的一部分，因为球和地板之间的低对比度。 换句话说，检测过程不理想，并且引入噪声。

为了容易地可视化整个对象轨迹，我们将所有视频帧覆盖到单个图像上。 “+”标记表示使用斑点分析计算的质心。

showTrajectory();



可以观察到两个问题：

该区域的中心通常不同于球的中心。 换句话说，在球的位置的测量中存在误差。

当球被盒阻塞时，球的位置不可用，即测量值丢失。

这两个挑战都可以通过使用卡尔曼滤波器来解决。

使用卡尔曼滤波器跟踪单个对象

使用前面看到的视频，trackSingleObject函数显示如何：

使用configureKalmanFilter创建vision.KalmanFilter

在序列中使用预测和正确的方法，以消除跟踪系统中存在的噪声

使用预测方法本身估计球的位置，当它被盒子遮挡

卡尔曼滤波器参数的选择可能是有挑战性的。 configureKalmanFilter函数有助于简化此问题。 有关此的更多细节可以在示例中进一步找到。

trackSingleObject函数包含嵌套辅助函数。 以下顶级变量用于在嵌套函数之间传输数据。

frame = []; % A video frame

detectedLocation = []; % The detected location

trackedLocation = []; % The tracked location

label = ''; % Label for the ball

utilities = []; % Utilities used to process the video

跟踪单个对象的过程如下所示。

function trackSingleObject(param)

% Create utilities used for reading video, detecting moving objects,

% and displaying the results.

utilities = createUtilities(param);

isTrackInitialized = false;

while ~isDone(utilities.videoReader)

frame = readFrame();

% Detect the ball.

[detectedLocation, isObjectDetected] = detectObject(frame);

if ~isTrackInitialized

if isObjectDetected

% Initialize a track by creating a Kalman filter when the ball is

% detected for the first time.

initialLocation = computeInitialLocation(param, detectedLocation);

kalmanFilter = configureKalmanFilter(param.motionModel, ...

initialLocation, param.initialEstimateError, ...

param.motionNoise, param.measurementNoise);

isTrackInitialized = true;

trackedLocation = correct(kalmanFilter, detectedLocation);

label = 'Initial';

else

trackedLocation = [];

label = '';

end

else

% Use the Kalman filter to track the ball.

if isObjectDetected % The ball was detected.

% Reduce the measurement noise by calling predict followed by

% correct.

predict(kalmanFilter);

trackedLocation = correct(kalmanFilter, detectedLocation);

label = 'Corrected';

else % The ball was missing.

% Predict the ball's location.

trackedLocation = predict(kalmanFilter);

label = 'Predicted';

end

end

annotateTrackedObject();

end % while

showTrajectory();

end

卡尔曼滤波器有两种不同的情况：

当检测到球时，卡尔曼滤波器首先预测其在当前视频帧处的状态，然后使用新检测到的对象位置来校正其状态。 这将生成已过滤的位置。

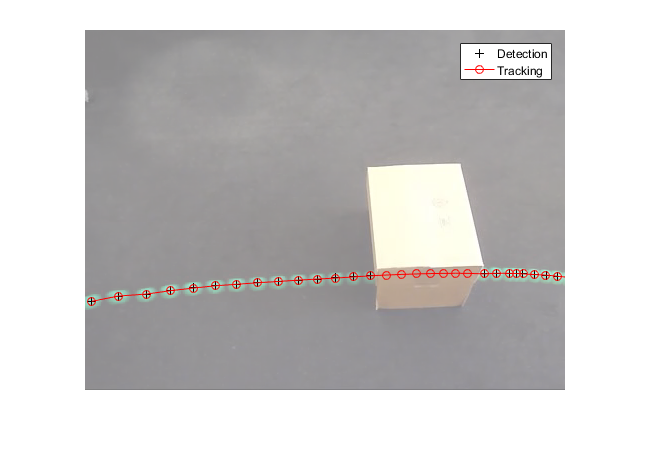
当球丢失时，卡尔曼滤波器仅依靠其先前的状态来预测球的当前位置。

您可以通过叠加所有视频帧来查看球的轨迹。

param = getDefaultParameters(); % get Kalman configuration that works well

% for this example

trackSingleObject(param); % visualize the results



探索卡尔曼滤波器配置选项

配置卡尔曼滤波器可能非常具有挑战性。 除了对卡尔曼滤波器的基本理解，它经常需要实验以得到一组合适的配置参数。 上面定义的trackSingleObject函数可以帮助您探索由configureKalmanFilter函数提供的各种配置选项。

configureKalmanFilter函数返回一个卡尔曼滤波器对象。 您必须提供五个输入参数。

kalmanFilter = configureKalmanFilter（MotionModel，InitialLocation，

           InitialEstimateError，MotionNoise，MeasurementNoise）

MotionModel设置必须对应于对象运动的物理特性。您可以将其设置为恒定速度或恒定加速度模型。以下示例说明了做出次优选择的后果。

param = getDefaultParameters(); % get parameters that work well

param.motionModel = 'ConstantVelocity'; % switch from ConstantAcceleration

% to ConstantVelocity

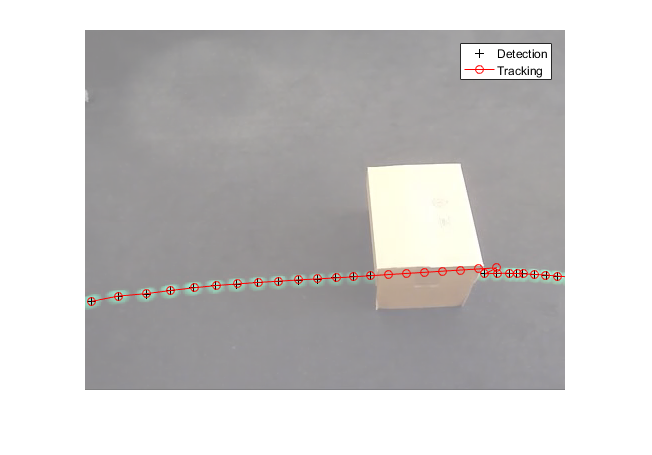
% After switching motion models, drop noise specification entries

% corresponding to acceleration.

param.initialEstimateError = param.initialEstimateError(1:2);

param.motionNoise = param.motionNoise(1:2);

trackSingleObject(param); % visualize the results



注意球出现在与预测位置完全不同的地方。 从球被释放的时间起，由于来自地毯的阻力，它经受恒定的减速。 因此，恒定加速度模型是更好的选择。 如果你保持恒定速度模型，跟踪结果将是次优的，无论你为其他值选择了什么。

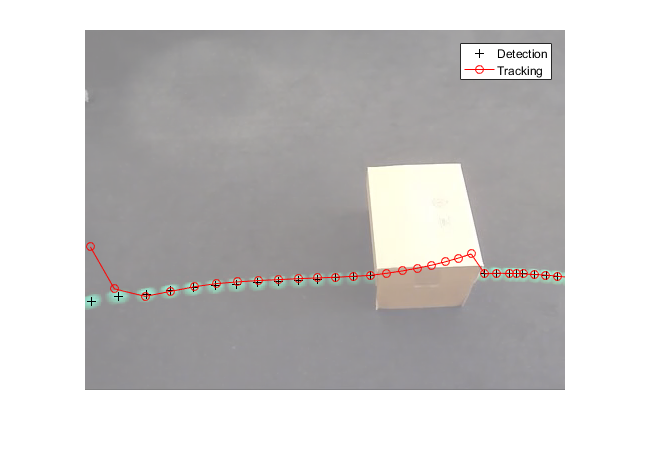
通常，您可以将InitialLocation输入设置为首次检测到对象的位置。您还将InitialEstimateError向量设置为大值，因为初始状态可能非常嘈杂，因为它是从单个检测导出的。 下图演示了将这些参数错误配置为对象运动的物理特性的影响。 您可以将其设置为恒定速度或恒定加速度模型。 以下示例说明了做出次优选择的后果。

param = getDefaultParameters(); % get parameters that work well

param.initialLocation = [0, 0]; % location that's not based on an actual detection

param.initialEstimateError = 100\*ones(1,3); % use relatively small values

trackSingleObject(param); % visualize the results



使用错误配置的参数，在卡尔曼滤波器返回的位置与对象的实际轨迹对准之前需要几个步骤。

MeasurementNoise的值应根据检测器的精度选择。 对于较不精确的检测器，将测量噪声设置为较大的值。 以下示例说明了错误配置的分段阈值的噪声检测。 增加测量噪声导致卡尔曼滤波器更多地依赖于其内部状态而不是进入的测量，并且因此补偿检测噪声模型。 以下示例说明了做出次优选择的后果。

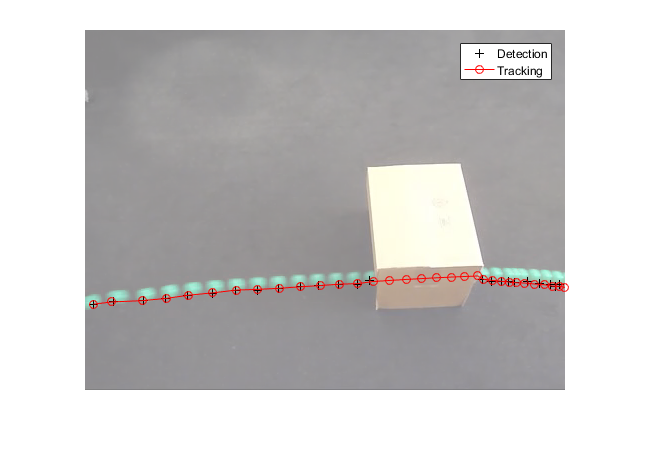
param = getDefaultParameters();

param.segmentationThreshold = 0.0005; % smaller value resulting in noisy detections

param.measurementNoise = 12500; % increase the value to compensate

% for the increase in measurement noise

trackSingleObject(param); % visualize the results



通常，对象不以恒定加速度或恒定速度移动。 使用MotionNoise可指定与理想运动模型的偏差量。 当增加运动噪声时，卡尔曼滤波器对输入测量更加依赖于其内部状态。 尝试使用MotionNoise参数进行试验，以了解有关其效果的更多信息。

现在您已经熟悉如何使用卡尔曼滤波器以及如何配置它，下一节将帮助您了解如何将其用于多个对象跟踪。

注意：为了简化上面例子中的配置过程，我们使用了configureKalmanFilter函数。 这个函数做了几个假设。 有关详细信息，请参阅函数的文档。 如果您需要对配置过程进行更高级别的控制，可以直接使用vision.KalmanFilter

使用卡尔曼滤波跟踪多个对象

跟踪多个对象提出了几个额外的挑战：

多个检测必须与正确的轨道相关联

您必须处理场景中出现的新对象

当多个对象合并为单个检测时，必须维护对象标识

vision.KalmanFilter对象与assignDetectionsToTracks函数一起可以帮助解决问题

将检测分配给轨道

确定检测是否对应于新对象，换句话说，确定轨迹创建

正如在遮挡的单个对象的情况下，预测可以用于帮助分离彼此接近的对象

要了解有关使用卡尔曼滤波器跟踪多个对象的更多信息，请参阅标题为“基于运动的多对象跟踪”的示例。

示例中使用的实用函数

效用函数用于检测对象和显示结果。本节说明示例如何实现这些函数。

获取创建卡尔曼滤波器和分割球的默认参数。

function param = getDefaultParameters

param.motionModel = 'ConstantAcceleration';

param.initialLocation = 'Same as first detection';

param.initialEstimateError = 1E5 \* ones(1, 3);

param.motionNoise = [25, 10, 1];

param.measurementNoise = 25;

param.segmentationThreshold = 0.05;

end

Read the next video frame from the video file.

function frame = readFrame()

frame = step(utilities.videoReader);

end

Detect and annotate the ball in the video.

function showDetections()

param = getDefaultParameters();

utilities = createUtilities(param);

trackedLocation = [];

idx = 0;

while ~isDone(utilities.videoReader)

frame = readFrame();

detectedLocation = detectObject(frame);

% Show the detection result for the current video frame.

annotateTrackedObject();

% To highlight the effects of the measurement noise, show the detection

% results for the 40th frame in a separate figure.

idx = idx + 1;

if idx == 40

combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);

figure, imshow(combinedImage);

end

end % while

% Close the window which was used to show individual video frame.

uiscopes.close('All');

end

Detect the ball in the current video frame.

function [detection, isObjectDetected] = detectObject(frame)

grayImage = rgb2gray(frame);

utilities.foregroundMask = step(utilities.foregroundDetector, grayImage);

detection = step(utilities.blobAnalyzer, utilities.foregroundMask);

if isempty(detection)

isObjectDetected = false;

else

% To simplify the tracking process, only use the first detected object.

detection = detection(1, :);

isObjectDetected = true;

end

end

Show the current detection and tracking results.

function annotateTrackedObject()

accumulateResults();

% Combine the foreground mask with the current video frame in order to

% show the detection result.

combinedImage = max(repmat(utilities.foregroundMask, [1,1,3]), frame);

if ~isempty(trackedLocation)

shape = 'circle';

region = trackedLocation;

region(:, 3) = 5;

combinedImage = insertObjectAnnotation(combinedImage, shape, ...

region, {label}, 'Color', 'red');

end

step(utilities.videoPlayer, combinedImage);

end

Show trajectory of the ball by overlaying all video frames on top of each other.

function showTrajectory

% Close the window which was used to show individual video frame.

uiscopes.close('All');

% Create a figure to show the processing results for all video frames.

figure; imshow(utilities.accumulatedImage/2+0.5); hold on;

plot(utilities.accumulatedDetections(:,1), ...

utilities.accumulatedDetections(:,2), 'k+');

if ~isempty(utilities.accumulatedTrackings)

plot(utilities.accumulatedTrackings(:,1), ...

utilities.accumulatedTrackings(:,2), 'r-o');

legend('Detection', 'Tracking');

end

end

Accumulate video frames, detected locations, and tracked locations to show the trajectory of the ball.

function accumulateResults()

utilities.accumulatedImage = max(utilities.accumulatedImage, frame);

utilities.accumulatedDetections ...

= [utilities.accumulatedDetections; detectedLocation];

utilities.accumulatedTrackings ...

= [utilities.accumulatedTrackings; trackedLocation];

end

For illustration purposes, select the initial location used by the Kalman filter.

function loc = computeInitialLocation(param, detectedLocation)

if strcmp(param.initialLocation, 'Same as first detection')

loc = detectedLocation;

else

loc = param.initialLocation;

end

end

Create utilities for reading video, detecting moving objects, and displaying the results.

function utilities = createUtilities(param)

% Create System objects for reading video, displaying video, extracting

% foreground, and analyzing connected components.

utilities.videoReader = vision.VideoFileReader('singleball.mp4');

utilities.videoPlayer = vision.VideoPlayer('Position', [100,100,500,400]);

utilities.foregroundDetector = vision.ForegroundDetector(...

'NumTrainingFrames', 10, 'InitialVariance', param.segmentationThreshold);

utilities.blobAnalyzer = vision.BlobAnalysis('AreaOutputPort', false, ...

'MinimumBlobArea', 70, 'CentroidOutputPort', true);

utilities.accumulatedImage = 0;

utilities.accumulatedDetections = zeros(0, 2);

utilities.accumulatedTrackings = zeros(0, 2);

end

end

## 使用CAMShift进行面部检测和跟踪

此示例显示如何自动检测和跟踪面部。

介绍

对象检测和跟踪在许多计算机视觉应用中是重要的，包括活动识别，汽车安全和监视。 在这个例子中，您将开发一个简单的面部跟踪系统，将跟踪问题分为三个独立的问题：

检测要跟踪的面

识别要跟踪的面部特征

跟踪面部

步骤1：检测要跟踪的面

在开始跟踪脸部之前，您需要先检测脸部。 使用vision.CascadeObjectDetector检测视频帧中面部的位置。 级联对象检测器使用Viola-Jones检测算法和训练的分类模型进行检测。 默认情况下，检测器配置为检测面，但可以为其他对象类型进行配置。

% Create a cascade detector object.

faceDetector = vision.CascadeObjectDetector();

% Read a video frame and run the detector.

videoFileReader = vision.VideoFileReader('visionface.avi');

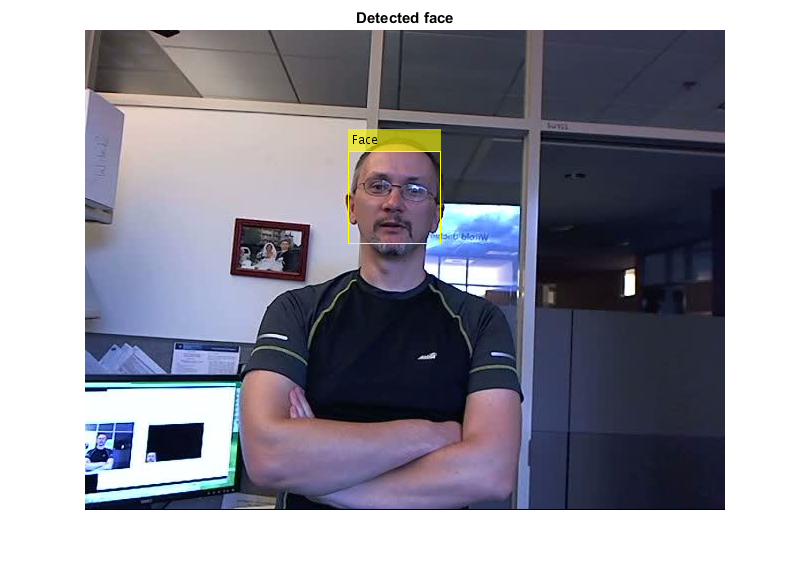
videoFrame = step(videoFileReader);

bbox = step(faceDetector, videoFrame);

% Draw the returned bounding box around the detected face.

videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'Face');

figure, imshow(videoOut), title('Detected face');



您可以使用级联对象检测器在连续的视频帧之间跟踪面部。然而，当面部倾斜或者人转过头时，可能会失去跟踪。这种限制是由于用于检测的训练分类模型的类型。为了避免这个问题，并且因为对每个视频帧执行面部检测是计算密集的，所以该示例使用简单的面部特征来进行跟踪。

步骤2：识别要跟踪的面部特征

一旦脸部位于视频中，下一步就是确定一个可以帮助您跟踪脸部的功能。例如，您可以使用形状，纹理或颜色。选择对象唯一的功能，并且即使对象移动也保持不变。

在本例中，您使用肤色作为要跟踪的功能。肤色在面部和背景之间提供了大量的对比度，并且不随着面部旋转或移动而改变。

% Get the skin tone information by extracting the Hue from the video frame

% converted to the HSV color space.

[hueChannel,~,~] = rgb2hsv(videoFrame);

% Display the Hue Channel data and draw the bounding box around the face.

figure, imshow(hueChannel), title('Hue channel data');

rectangle('Position',bbox(1,:),'LineWidth',2,'EdgeColor',[1 1 0])



步骤3：跟踪面部

选择肤色作为要跟踪的特征，您现在可以使用vision.HistogramBasedTracker进行跟踪。 基于直方图的跟踪器使用CAMShift算法，其提供使用像素值的直方图来跟踪对象的能力。 在该示例中，从检测到的面部的鼻子区域提取色调通道像素。 这些像素用于初始化跟踪器的直方图。 该示例使用该直方图在连续的视频帧上跟踪对象。

% Detect the nose within the face region. The nose provides a more accurate

% measure of the skin tone because it does not contain any background

% pixels.

noseDetector = vision.CascadeObjectDetector('Nose', 'UseROI', true);

noseBBox = step(noseDetector, videoFrame, bbox(1,:));

% Create a tracker object.

tracker = vision.HistogramBasedTracker;

% Initialize the tracker histogram using the Hue channel pixels from the

% nose.

initializeObject(tracker, hueChannel, noseBBox(1,:));

% Create a video player object for displaying video frames.

videoInfo = info(videoFileReader);

videoPlayer = vision.VideoPlayer('Position',[300 300 videoInfo.VideoSize+30]);

% Track the face over successive video frames until the video is finished.

while ~isDone(videoFileReader)

% Extract the next video frame

videoFrame = step(videoFileReader);

% RGB -> HSV

[hueChannel,~,~] = rgb2hsv(videoFrame);

% Track using the Hue channel data

bbox = step(tracker, hueChannel);

% Insert a bounding box around the object being tracked

videoOut = insertObjectAnnotation(videoFrame,'rectangle',bbox,'Face');

% Display the annotated video frame using the video player object

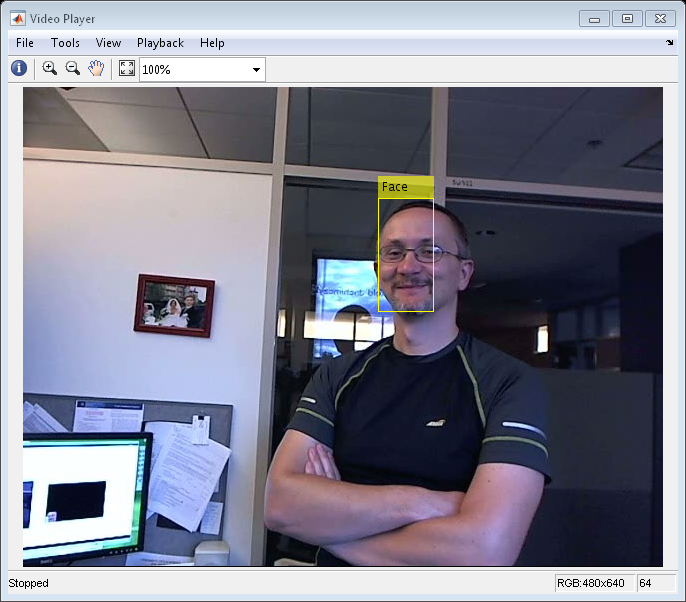
step(videoPlayer, videoOut);

end

% Release resources

release(videoFileReader);

release(videoPlayer);



概要

在此示例中，您创建了一个自动检测和跟踪单个面部的简单面部跟踪系统。 尝试更改输入视频，并查看您是否能够跟踪面部。 如果您发现跟踪结果不佳，请检查Hue通道数据，以查看面部区域和背景之间是否有足够的对比度。

Reference

[1] G.R. Bradski "Real Time Face and Object Tracking as a Component of a Perceptual User Interface", Proceedings of the 4th IEEE Workshop on Applications of Computer Vision, 1998.

[2] Viola, Paul A. and Jones, Michael J. "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.

## 车道偏离警告系统

此示例显示如何检测视频流中的道路车道标记以及如何突出显示车辆行驶的车道。该信息可用于检测从车道意外偏离并发出警告。

介绍

该示例检测并跟踪视频序列中的道路车道标记，并通知驾驶员他们是否正在车道上移动。该示例说明了如何使用HoughTransform，HoughLines和LocalMaximaFinder系统对象来创建线检测和跟踪算法。该示例使用以下步骤实现此算法：

·检测当前视频帧中的车道标记。

·将当前车道标记与前一视频帧中检测到的车道标记相匹配。

·找到左右车道标记。

·如果车辆移动穿过任一车道标记，则发出警告消息。

为了处理低质量视频序列，其中车道标记可能难以看到或隐藏在对象后面，示例等待车道标记出现在多个帧中，之后它认为标记有效。该示例使用相同的过程来决定何时开始忽略车道标记。

初始化

Use these next sections of code to initialize the required variables and System objects.

DrawPoly = 1; % Set to 0 to draw only lines

NumRows = 120; % Number of rows in the image region to process.

MaxLaneNum = 20; % Maximum number of lanes to store in the tracking repository.

ExpLaneNum = 2; % Maximum number of lanes to find in the current frame.

Rep\_ref = zeros(2, MaxLaneNum); % Stored lines

Count\_ref = zeros(1, MaxLaneNum); % Count of each stored line

TrackThreshold = 75; % Maximum allowable change of lane distance

% metric between two frames.

LaneColors = single([0 0 0;1 1 0; 1 1 0; 1 1 1;1 1 1]);

% Minimum number of frames a lane must be detected to become a valid lane.

frameFound = 5;

% Maximum number of frames a lane can be missed without marking it invalid.

frameLost = 20;

% For selecting Rho values 35:45 (1-based index: 415:424)

startIdxRho\_R = 415;

NumRhos\_R = 11;

% For selecting Theta values -90:-70deg (1-based index: 1:21)

startIdxTheta\_R = 1;

NumThetas\_R = 21;

% For selecting Rho values 379:415 (1-based index: 1:36)

startIdxRho\_L = 380;

NumRhos\_L = 36;

% For selecting Theta values 55:85deg (1-based index: 146:176)

startIdxTheta\_L = 146;

NumThetas\_L = 21;

% Offset for displaying the lines

offset = int32([0, NumRows, 0, NumRows]);

Create a VideoFileReader System object™ to read video from a file.

hVideoSrc = vision.VideoFileReader('viplanedeparture.mp4');

Create HoughLines System objects to find the Cartesian coordinates of the lines defined by the lane markers.

hHoughLines1 = vision.HoughLines('SineComputation', 'Trigonometric function');

hHoughLines3 = vision.HoughLines('SineComputation', 'Trigonometric function');

define parameters for inserting lane departure warning text.

warnText = {sprintf('Right\nDeparture'), '', sprintf(' Left\n Departure')};

warnTextLoc = [120 170;-1 -1; 2 170];

define parameters for inserting text specifying lane marker color/type.

lineText = {'', ...

sprintf('Yellow\nBroken'), sprintf('Yellow\nSolid'), ...

sprintf('White\nBroken'), sprintf('White\nSolid')};

Create a VideoPlayer System object to display the output video.

hVideoOut = vision.VideoPlayer;

Initialize the variables used in the stream processing loop.

Frame = 0;

NumNormalDriving = 0;

OutMsg = int8(-1);

OutMsgPre = OutMsg;

Broken = false;

流处理循环

Create the processing loop to perform lane detection on the input video. This loop uses the System objects you instantiated.

warningTextColors = {[1 0 0], [1 0 0], [0 0 0], [0 0 0]};

while ~isDone(hVideoSrc)

RGB = step(hVideoSrc);

% Select the lower portion of input video (confine field of view)

Imlow = RGB(NumRows+1:end, :, :);

% Edge detection and Hough transform

Imlow = rgb2gray(Imlow); % Convert RGB to intensity

I = imfilter(Imlow, [-1 0 1], 'replicate','corr');

% Saturate the values to be between 0 and 1

I(I < 0) = 0;

I(I > 1) = 1;

th = multithresh(I); % compute threshold

[H, Theta, Rho] = hough(I > th);

% Convert Theta to radians

Theta = Theta \* pi / 180;

% Peak detection

H1 = H;

% Wipe out H matrix with theta < -78 deg and theta >= 78 deg

H1(:, 1:12) = 0;

H1(:, end-12:end) = 0;

Idx1 = houghpeaks(H1, ExpLaneNum, 'NHoodSize', [301 81], 'Threshold', 1);

Count1 = size(Idx1,1);

% Select Rhos and Thetas corresponding to peaks

Line = [Rho(Idx1(:, 1)); Theta(Idx1(:, 2))];

Enable = [ones(1,Count1) zeros(1, ExpLaneNum-Count1)];

% Track a set of lane marking lines

[Rep\_ref, Count\_ref] = videolanematching(Rep\_ref, Count\_ref, ...

MaxLaneNum, ExpLaneNum, Enable, Line, ...

TrackThreshold, frameFound+frameLost);

% Convert lines from Polar to Cartesian space.

Pts = step(hHoughLines1, Rep\_ref(2,:), Rep\_ref(1,:), Imlow);

% Detect whether there is a left or right lane departure.

[TwoValidLanes, NumNormalDriving, TwoLanes, OutMsg] = ...

videodeparturewarning(Pts, Imlow, MaxLaneNum, Count\_ref, ...

NumNormalDriving, OutMsg);

% Meaning of OutMsg: 0 = Right lane departure,

% 1 = Normal driving, 2 = Left lane departure

% Detect the type and color of lane marker lines

YCbCr = rgb2ycbcr(double(RGB(NumRows+1:240, :, :)));

ColorAndTypeIdx = videodetectcolorandtype(TwoLanes, YCbCr);

% Meaning of ColorAndTypeIdx:

% INVALID\_COLOR\_OR\_TYPE = int8(0);

% YELLOW\_BROKEN = int8(1); YELLOW\_SOLID = int8(2);

% WHITE\_BROKEN = int8(3); WHITE\_SOLID = int8(4).

% Output

Frame = Frame + 1;

if Frame >= 5

TwoLanes1 = TwoLanes + [offset; offset]';

if DrawPoly && TwoValidLanes

if TwoLanes(4,1) >= 239

Templ = TwoLanes1(3:4, 1);

else

Templ = [0 239]';

end

if TwoLanes(4,2) >= 239

Tempr = TwoLanes1(3:4, 2);

else

Tempr = [359 239]';

end

Pts\_poly = [TwoLanes1(:,1); Templ; Tempr; ...

TwoLanes1(3:4,2); TwoLanes1(1:2,2)];

% Draw Polygon for lane

RGB = insertShape(RGB,'FilledPolygon',Pts\_poly.',...

'Color',[0 1 1],'Opacity',0.2);

end

% Draw lane marker lines

RGB = insertShape(RGB,'Line',TwoLanes1',...

'Color',{'yellow','magenta'});

% Insert Departure warning text (empty text will not be drawn)

txt = warnText{OutMsg+1};

txtLoc = warnTextLoc(OutMsg+1, :);

txtColor = single(warningTextColors{mod(Frame-1,4)+1});

RGB = insertText(RGB,txtLoc,txt,'TextColor', txtColor, ...

'FontSize',20, 'BoxOpacity', 0);

% Insert text indicating type and color of left and right lanes

for ii=1:2

% empty text will not be drawn

txtLoc = TwoLanes1([1 2], ii)' + int32([0 -35]);

lineTxt = lineText{ColorAndTypeIdx(ii)};

txtColor = LaneColors(ColorAndTypeIdx(ii), :);

RGB = insertText(RGB,txtLoc,lineTxt,'TextColor',txtColor, ...

'FontSize',14, 'BoxOpacity', 0);

end

% Draw third lane if needed

if OutMsgPre ~= OutMsg

ColorType = ColorAndTypeIdx(2-(OutMsg == 2));

Broken = ColorType == 2 || ColorType == 4;

end

ShowThirdLane = Broken && (OutMsg~=1);

if ShowThirdLane

if OutMsg == 0

% Find right third lane

Idx2 = houghpeaks(H(startIdxRho\_R:startIdxRho\_R+NumRhos\_R-1, ...

startIdxTheta\_R:startIdxTheta\_R+NumThetas\_R-1), ...

'NHoodSize', [7 7], 'Threshold', 1);

Rhor = Rho(Idx2(:,1) + startIdxRho\_R);

Thetar = Theta(Idx2(:,2) + startIdxTheta\_R);

ThirdLane = step(hHoughLines3, Thetar, Rhor, Imlow);

else

% Find left third lane

Idx3 = houghpeaks(H(startIdxRho\_L:startIdxRho\_L+NumRhos\_L-1 , ...

startIdxTheta\_L:startIdxTheta\_L+NumThetas\_L-1),...

'NHoodSize', [7 7], 'Threshold', 1);

Rhol = Rho(Idx3(:,1) + startIdxRho\_L);

Thetal = Theta(Idx3(:,2) + startIdxTheta\_L);

ThirdLane = step(hHoughLines3, Thetal, Rhol, Imlow);

end

OutThirdLane = videoexclude3rdlane(ThirdLane, ShowThirdLane,...

TwoLanes, TwoValidLanes, YCbCr);

OutThirdLane = OutThirdLane(:) + offset(:);

RGB = insertShape(RGB,'Line',OutThirdLane.','Color','green');

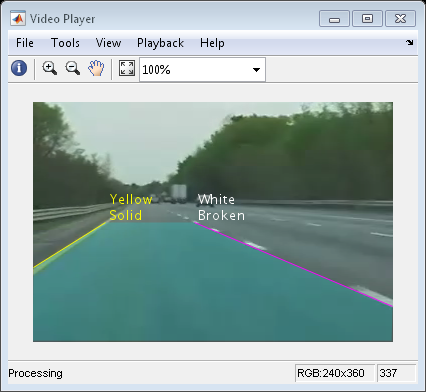
end

end

OutMsgPre = OutMsg;

step(hVideoOut, RGB); % Display video

end



发布

这里，您调用系统对象上的释放方法以关闭任何打开的文件和设备。

release(hVideoSrc);

概要

在“视频播放器”窗口中，您可以看到该示例检测到车辆前方的车道，并绘制了一个青色多边形以标记其位置。 该示例还指示车辆何时离开其车道并且检测到车道标线的类型。

附录

此示例中使用以下辅助函数。

* [videolanematching.m](matlab:edit('videolanematching.m'))
* [videodeparturewarning.m](matlab:edit('videodeparturewarning.m'))
* [videodetectcolorandtype.m](matlab:edit('videodetectcolorandtype.m'))
* [videoexclude3rdlane.m](matlab:edit('videoexclude3rdlane.m'))

## 从一辆车跟踪行人

此示例显示如何使用安装在移动车中的摄像头跟踪行人。

概述

该示例示出了如何对来自移动的相机的视频中的人执行自动检测和跟踪。 它演示了适用于移动摄像机的跟踪系统的灵活性，这是汽车安全应用的理想选择。 与固定相机示例，基于运动的多对象跟踪不同，本示例包含几个附加的算法步骤。 这些步骤包括人检测，定制非最大抑制和启发式以识别和消除假报警轨迹。 有关更多信息，请参阅多对象跟踪。

这个例子是一个函数，主体在顶部，辅助程序以嵌套函数的形式在下面。

function PedestrianTrackingFromMovingCameraExample()

% Create system objects used for reading video, loading prerequisite data file, detecting pedestrians, and displaying the results.

videoFile = 'vippedtracking.mp4';

scaleDataFile = 'pedScaleTable.mat'; % An auxiliary file that helps to determine the size of a pedestrian at different pixel locations.

obj = setupSystemObjects(videoFile, scaleDataFile);

% Create an empty array of tracks.

tracks = initializeTracks();

% ID of the next track.

nextId = 1;

% Set the global parameters.

option.ROI = [40 95 400 140]; % A rectangle [x, y, w, h] that limits the processing area to ground locations.

option.scThresh = 0.3; % A threshold to control the tolerance of error in estimating the scale of a detected pedestrian.

option.gatingThresh = 0.9; % A threshold to reject a candidate match between a detection and a track.

option.gatingCost = 100; % A large value for the assignment cost matrix that enforces the rejection of a candidate match.

option.costOfNonAssignment = 10; % A tuning parameter to control the likelihood of creation of a new track.

option.timeWindowSize = 16; % A tuning parameter to specify the number of frames required to stabilize the confidence score of a track.

option.confidenceThresh = 2; % A threshold to determine if a track is true positive or false alarm.

option.ageThresh = 8; % A threshold to determine the minimum length required for a track being true positive.

option.visThresh = 0.6; % A threshold to determine the minimum visibility value for a track being true positive.

% Detect people and track them across video frames.

cont = ~isDone(obj.reader);

while cont

frame = readFrame();

[centroids, bboxes, scores] = detectPeople();

predictNewLocationsOfTracks();

[assignments, unassignedTracks, unassignedDetections] = ...

detectionToTrackAssignment();

updateAssignedTracks();

updateUnassignedTracks();

deleteLostTracks();

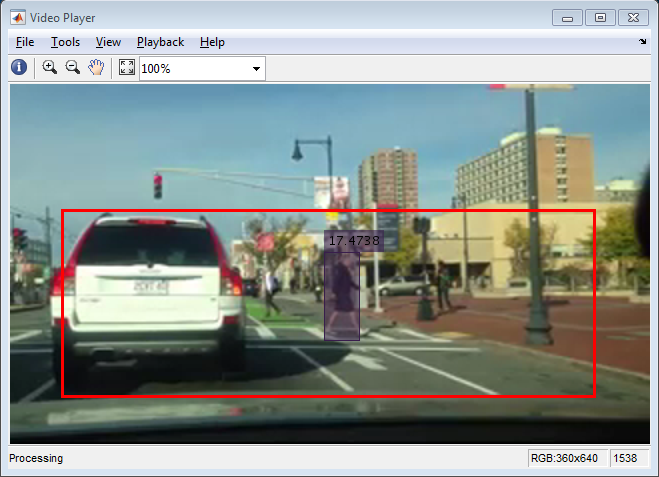
createNewTracks();

displayTrackingResults();

% Exit the loop if the video player figure is closed by user.

cont = ~isDone(obj.reader) && isOpen(obj.videoPlayer);

end



跟踪系统的辅助输入和全局参数

该跟踪系统需要包含将图像中的像素位置与标记行人位置的边界框的大小相关联的信息的数据文件。这个先验知识存储在向量pedScaleTable中。 pedScaleTable中的第n个条目表示成人人的估计高度（以像素为单位）。索引n引用行人脚的近似Y坐标。

为了获得这样的矢量，从与测试环境相同的视点和类似的场景中采集训练图像的集合。训练图像包含在距相机不同距离处的行人的图像。使用trainingImageLabeler应用程序，图像中行人的边界框手动注释。边界框的高度与行人在图像中的位置一起用于通过回归生成比例数据文件。这里是一个帮助函数，显示适合线性回归模型的算法步骤：helperTableOfScales.m

还有一组全局参数，可以调整以优化跟踪性能。您可以使用以下说明了解这些参数如何影响跟踪性能。

·ROI：以[x，y，w，h]的形式的感兴趣区域。它将处理区域限制到接地位置。

·scThresh：刻度估计的容差阈值。当检测到的标度和期望标度之间的差超过容差时，候选检测被认为是不现实的，并从输出中移除。

·gatingThresh：距离测量的门控参数。当匹配检测到的边界框和预测的边界框的成本超过阈值时，系统从跟踪考虑中移除两个边界框的关联。

·gatingCost：分配成本矩阵的值，以阻止可能的跟踪到检测分配。

·costOfNonAssignment：不分配检测或轨道的分配成本矩阵的值。将其设置得太低增加了创建新轨道的可能性，并且可能导致轨道碎片。将其设置得太高可能导致对应于一系列单独移动对象的单个轨道。

·timeWindowSize：估计轨道置信度所需的帧数。

·confidenceThresh：确定轨道是否为真的置信度阈值。

·ageThresh：轨道的最小长度为真正的。

·visThresh：确定轨道是否为真的最小可见度阈值。

创建跟踪系统初始化的系统对象

setupSystemObjects函数创建用于读取和显示视频帧并加载缩放数据文件的系统对象。

存储在比例数据文件中的pedScaleTable向量对我们对目标和场景的先验知识进行编码。一旦你从你的样本训练回归，你可以计算图像中每个可能的Y位置的预期高度。这些值存储在向量中。 pedScaleTable中的第n个条目表示我们估计的成年人的高度（以像素为单位）。索引n引用行人脚的近似Y坐标。该示例是具有主体在顶部的函数，并且辅助例程以嵌套函数的形式在下面。

function obj = setupSystemObjects(videoFile,scaleDataFile)

% Initialize Video I/O

% Create objects for reading a video from a file, drawing the

% detected and tracked people in each frame, and playing the video.

% Create a video file reader.

obj.reader = vision.VideoFileReader(videoFile, 'VideoOutputDataType', 'uint8');

% Create a video player.

obj.videoPlayer = vision.VideoPlayer('Position', [29, 597, 643, 386]);

% Load the scale data file

ld = load(scaleDataFile, 'pedScaleTable');

obj.pedScaleTable = ld.pedScaleTable;

end

初始化曲目

initializeTracks函数创建轨道数组，其中每个轨道是表示视频中的移动对象的结构。该结构的目的是维持被跟踪对象的状态。状态包括用于检测到轨道分配，轨道终止和显示的信息。

该结构包含以下字段：

·id：轨道的整数ID。

·color：用于显示的轨道的颜色。

·bboxes：一个N乘4矩阵，用于表示对象的边界框，当前框在最后一行。每行具有[x，y，width，height]的形式。

·score：N-by-1向量，用于在最后一行记录来自人员检测器的分类分数与当前检测分数。

·kalmanFilter：用于基于运动的跟踪的卡尔曼滤波器对象。我们跟踪图像中对象的中心点;

·age：自轨道初始化以来的帧数。

·totalVisibleCount：检测到对象的总帧数（可见）。

·confidence：一对两个数字，表示我们信任轨道的自信程度。它在预定义的时间窗内存储过去的最大和平均检测分数。

·predPosition：在下一帧中预测的边界框。

function tracks = initializeTracks()

% Create an empty array of tracks

tracks = struct(...

'id', {}, ...

'color', {}, ...

'bboxes', {}, ...

'scores', {}, ...

'kalmanFilter', {}, ...

'age', {}, ...

'totalVisibleCount', {}, ...

'confidence', {}, ...

'predPosition', {});

end

读取视频帧

从视频文件中读取下一个视频帧。

function frame = readFrame()

frame = step(obj.reader);

end

检测人

detectPeople函数返回检测到的人的质心，边界框和分类分数。 它对detectPeopleACF的原始输出执行过滤和非最大抑制。

·质心：N×2矩阵，每行的形式为[x，y]。

·bboxes：一个N乘4矩阵，每行的格式为[x，y，width，height]。

·分数：具有每个元素的N乘1向量是在相应帧处的分类分数。

function [centroids, bboxes, scores] = detectPeople()

% Resize the image to increase the resolution of the pedestrian.

% This helps detect people further away from the camera.

resizeRatio = 1.5;

frame = imresize(frame, resizeRatio, 'Antialiasing',false);

% Run ACF people detector within a region of interest to produce

% detection candidates.

[bboxes, scores] = detectPeopleACF(frame, option.ROI, ...

'Model','caltech',...

'WindowStride', 2,...

'NumScaleLevels', 4, ...

'SelectStrongest', false);

% Look up the estimated height of a pedestrian based on location of their feet.

height = bboxes(:, 4) / resizeRatio;

y = (bboxes(:,2)-1) / resizeRatio + 1;

yfoot = min(length(obj.pedScaleTable), round(y + height));

estHeight = obj.pedScaleTable(yfoot);

% Remove detections whose size deviates from the expected size,

% provided by the calibrated scale estimation.

invalid = abs(estHeight-height)>estHeight\*option.scThresh;

bboxes(invalid, :) = [];

scores(invalid, :) = [];

% Apply non-maximum suppression to select the strongest bounding boxes.

[bboxes, scores] = selectStrongestBbox(bboxes, scores, ...

'RatioType', 'Min', 'OverlapThreshold', 0.6);

% Compute the centroids

if isempty(bboxes)

centroids = [];

else

centroids = [(bboxes(:, 1) + bboxes(:, 3) / 2), ...

(bboxes(:, 2) + bboxes(:, 4) / 2)];

end

end

预测现有轨道的新位置

使用卡尔曼滤波器来预测当前帧中的每个轨道的质心，并相应地更新其边界框。 我们将前一帧中边界框的宽度和高度作为当前的大小预测。

function predictNewLocationsOfTracks()

for i = 1:length(tracks)

% Get the last bounding box on this track.

bbox = tracks(i).bboxes(end, :);

% Predict the current location of the track.

predictedCentroid = predict(tracks(i).kalmanFilter);

% Shift the bounding box so that its center is at the predicted location.

tracks(i).predPosition = [predictedCentroid - bbox(3:4)/2, bbox(3:4)];

end

end

将检测分配给轨道

通过最小化成本来将当前帧中的对象检测分配给现有轨道。使用bboxOverlapRatio函数计算成本，并且是预测的边界框和检测到的边界框之间的重叠比率。在该示例中，我们假设人将由于视频的高帧速率和人的低运动速度而在连续帧中逐渐移动。

该算法包括两个步骤：

步骤1：使用bboxOverlapRatio度量计算为每个轨道分配每个检测的成本。当人们朝向或远离相机移动时，它们的运动将不能仅通过质心点来精确地描述。成本考虑了图像平面上的距离以及边界框的比例。这防止将远离摄像机的检测分配给更靠近摄像机的轨道，即使它们的质心重合。这种成本函数的选择将减轻计算，而不诉诸更复杂的动态模型。结果存储在M×N矩阵中，其中M是磁道数，N是检测数。

步骤2：使用assignDetectionsToTracks函数解决由成本矩阵表示的分配问题。该函数采用成本矩阵和不向轨道分配任何检测的成本。

不向轨道分配检测的成本值取决于成本函数返回的值的范围。该值必须通过实验调整。将其设置得太低增加了创建新轨道的可能性，并且可能导致轨道碎片。将其设置得太高可能导致对应于一系列单独移动对象的单个轨道。

assignDetectionsToTracks函数使用Munkres版本的匈牙利算法来计算最小化总成本的分配。它返回一个M×2矩阵，其包含在其两列中分配的轨道和检测的对应索引。它还返回保持未分配的轨道和检测的索引。

function [assignments, unassignedTracks, unassignedDetections] = ...

detectionToTrackAssignment()

% Compute the overlap ratio between the predicted boxes and the

% detected boxes, and compute the cost of assigning each detection

% to each track. The cost is minimum when the predicted bbox is

% perfectly aligned with the detected bbox (overlap ratio is one)

predBboxes = reshape([tracks(:).predPosition], 4, [])';

cost = 1 - bboxOverlapRatio(predBboxes, bboxes);

% Force the optimization step to ignore some matches by

% setting the associated cost to be a large number. Note that this

% number is different from the 'costOfNonAssignment' below.

% This is useful when gating (removing unrealistic matches)

% technique is applied.

cost(cost > option.gatingThresh) = 1 + option.gatingCost;

% Solve the assignment problem.

[assignments, unassignedTracks, unassignedDetections] = ...

assignDetectionsToTracks(cost, option.costOfNonAssignment);

end

更新已分配曲目

updateAssignedTracks函数使用相应的检测更新每个分配的轨道。 它调用正确的vision.KalmanFilter方法来校正位置估计。 接下来，它通过取最近（最多）4个框的大小的平均值来存储新的边界框，并且将轨迹的年龄和总可见计数增加1.最后，该函数调整轨迹的置信分数 基于先前的检测分数。

function updateAssignedTracks()

numAssignedTracks = size(assignments, 1);

for i = 1:numAssignedTracks

trackIdx = assignments(i, 1);

detectionIdx = assignments(i, 2);

centroid = centroids(detectionIdx, :);

bbox = bboxes(detectionIdx, :);

% Correct the estimate of the object's location

% using the new detection.

correct(tracks(trackIdx).kalmanFilter, centroid);

% Stabilize the bounding box by taking the average of the size

% of recent (up to) 4 boxes on the track.

T = min(size(tracks(trackIdx).bboxes,1), 4);

w = mean([tracks(trackIdx).bboxes(end-T+1:end, 3); bbox(3)]);

h = mean([tracks(trackIdx).bboxes(end-T+1:end, 4); bbox(4)]);

tracks(trackIdx).bboxes(end+1, :) = [centroid - [w, h]/2, w, h];

% Update track's age.

tracks(trackIdx).age = tracks(trackIdx).age + 1;

% Update track's score history

tracks(trackIdx).scores = [tracks(trackIdx).scores; scores(detectionIdx)];

% Update visibility.

tracks(trackIdx).totalVisibleCount = ...

tracks(trackIdx).totalVisibleCount + 1;

% Adjust track confidence score based on the maximum detection

% score in the past 'timeWindowSize' frames.

T = min(option.timeWindowSize, length(tracks(trackIdx).scores));

score = tracks(trackIdx).scores(end-T+1:end);

tracks(trackIdx).confidence = [max(score), mean(score)];

end

end

更新未分配的曲目

updateUnassignedTracks函数将每个未分配的轨道标记为不可见，将其年龄增加1，并将预测的边界框追加到轨道。 置信度设置为零，因为我们不确定为什么它不分配给轨道。

function updateUnassignedTracks()

for i = 1:length(unassignedTracks)

idx = unassignedTracks(i);

tracks(idx).age = tracks(idx).age + 1;

tracks(idx).bboxes = [tracks(idx).bboxes; tracks(idx).predPosition];

tracks(idx).scores = [tracks(idx).scores; 0];

% Adjust track confidence score based on the maximum detection

% score in the past 'timeWindowSize' frames

T = min(option.timeWindowSize, length(tracks(idx).scores));

score = tracks(idx).scores(end-T+1:end);

tracks(idx).confidence = [max(score), mean(score)];

end

end

删除丢失的曲目

deleteLostTracks函数删除对于太多连续帧不可见的轨道。 它还会删除最近创建的，对于许多帧总体不可见的轨道。

噪声检测往往导致伪轨迹的产生。 对于此示例，我们在以下条件下删除轨道：

·跟踪对象的时间很短。 这通常发生在错误检测显示几个帧并且为其启动轨道时。

·该轨道对于大多数帧被标记为不可见。

·它未能在过去几帧内接收到强检测，这表示为最大检测置信分数。

function deleteLostTracks()

if isempty(tracks)

return;

end

% Compute the fraction of the track's age for which it was visible.

ages = [tracks(:).age]';

totalVisibleCounts = [tracks(:).totalVisibleCount]';

visibility = totalVisibleCounts ./ ages;

% Check the maximum detection confidence score.

confidence = reshape([tracks(:).confidence], 2, [])';

maxConfidence = confidence(:, 1);

% Find the indices of 'lost' tracks.

lostInds = (ages <= option.ageThresh & visibility <= option.visThresh) | ...

(maxConfidence <= option.confidenceThresh);

% Delete lost tracks.

tracks = tracks(~lostInds);

end

创建新曲目

从未分配的检测项创建新曲目。 假设任何未分配的检测是新轨道的开始。 在实践中，您可以使用其他线索来消除噪声检测，如大小，位置或外观。

function createNewTracks()

unassignedCentroids = centroids(unassignedDetections, :);

unassignedBboxes = bboxes(unassignedDetections, :);

unassignedScores = scores(unassignedDetections);

for i = 1:size(unassignedBboxes, 1)

centroid = unassignedCentroids(i,:);

bbox = unassignedBboxes(i, :);

score = unassignedScores(i);

% Create a Kalman filter object.

kalmanFilter = configureKalmanFilter('ConstantVelocity', ...

centroid, [2, 1], [5, 5], 100);

% Create a new track.

newTrack = struct(...

'id', nextId, ...

'color', 255\*rand(1,3), ...

'bboxes', bbox, ...

'scores', score, ...

'kalmanFilter', kalmanFilter, ...

'age', 1, ...

'totalVisibleCount', 1, ...

'confidence', [score, score], ...

'predPosition', bbox);

% Add it to the array of tracks.

tracks(end + 1) = newTrack; %#ok<AGROW>

% Increment the next id.

nextId = nextId + 1;

end

end

显示跟踪结果

displayTrackingResults函数为视频帧上的每个轨道绘制一个彩色边界框。 框的透明度水平以及显示的分数表示检测和轨迹的置信度。

function displayTrackingResults()

displayRatio = 4/3;

frame = imresize(frame, displayRatio);

if ~isempty(tracks),

ages = [tracks(:).age]';

confidence = reshape([tracks(:).confidence], 2, [])';

maxConfidence = confidence(:, 1);

avgConfidence = confidence(:, 2);

opacity = min(0.5,max(0.1,avgConfidence/3));

noDispInds = (ages < option.ageThresh & maxConfidence < option.confidenceThresh) | ...

(ages < option.ageThresh / 2);

for i = 1:length(tracks)

if ~noDispInds(i)

% scale bounding boxes for display

bb = tracks(i).bboxes(end, :);

bb(:,1:2) = (bb(:,1:2)-1)\*displayRatio + 1;

bb(:,3:4) = bb(:,3:4) \* displayRatio;

frame = insertShape(frame, ...

'FilledRectangle', bb, ...

'Color', tracks(i).color, ...

'Opacity', opacity(i));

frame = insertObjectAnnotation(frame, ...

'rectangle', bb, ...

num2str(avgConfidence(i)), ...

'Color', tracks(i).color);

end

end

end

frame = insertShape(frame, 'Rectangle', option.ROI \* displayRatio, ...

'Color', [255, 0, 0], 'LineWidth', 3);

step(obj.videoPlayer, frame);

end