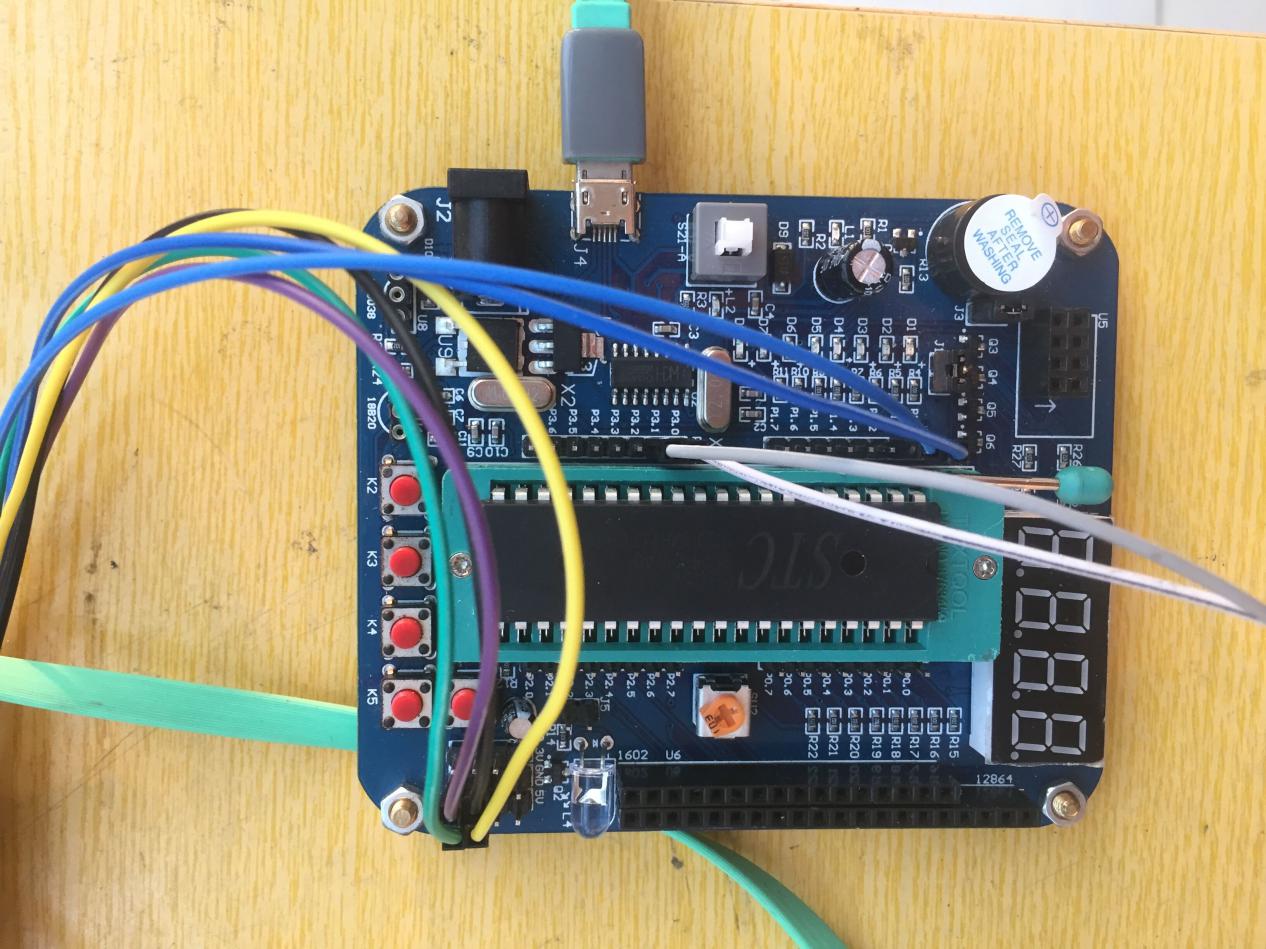
基于目标识别的云台移动追踪系统

1. 概述

摄像头把实时视频数据传输到电脑，在matlab中运行目标识别算法，获取目标的中心点位置，转化为舵机参数并传回给舵机，使云台跟随目标转动，从而实现移动追踪。（注：本实验采用人脸作为识别目标）

1. 硬件配置

Windows10 系统电脑，matlab2014a 版本以上。云台，摄像头等（待补充）



1. 算法步骤

1、初始化

1. 初始化摄像头对象
2. 初始化云台状态
3. 初始化目标检测与追踪方法
4. 初始化视频播放窗口对象
5. 初始化舵机参数
6. 初始化角度转换比例
7. 初始化摄像头到目标的映射距离

video = webcam(2);%初始化摄像头对象

s = serial('com4');

fopen(s);

fwrite(s,'15001500');%初始化云台状态

faceDetector = vision.CascadeObjectDetector(); % Finds faces by default

tracker = MultiObjectTrackerKLT;%初始化目标检测与追踪方法

videoPlayer = vision.VideoPlayer('Position',[200 100 fliplr(frameSize(1:2)+30)]);%初始化视频播放窗口对象

xPara = 1500;yPara = 1500;%初始化舵机参数

para = 2000/180;%初始化角度转换比例

distance = (width/2)/tan((angle/2)\*pi/180);%摄像头到目标的映射距离,angle为摄像头角度

2、循环读取视频帧

1）循环读取视频帧

2）运行目标识别算法，获取所有目标的boundingBoxes

3）选取最大目标作为对象，显示最大目标的boundingBoxes

4）计算目标中心位置，转化为基于图像中心点为原点的坐标，显示为红圆圈

while frameNumber<500

frameNumber = frameNumber + 1;

frame = snapshot (video);%循环读取视频帧

framegray = rgb2gray(frame);

if mod(T.frame\_number, 10) == 0

bboxes = 2 \* faceDetector.step(imresize(framegray, 0.5));

if ~isempty(bboxes)

tracker.addDetections(framegray, bboxes);

end

else

% Track faces

tracker.track(framegray);%运行目标识别算法

end

% only tracking the bigger face

tb = tracker.Bboxes;%获取所有目标的boundingBoxes

if size(tb,1)>1

tt = tb(:,3) .\* tb(:,4);

id = tt==max(tt);

tb = tb(id,:);%选取最大目标作为对象

end

% Display bounding boxes and tracker.

tb\_nb = size(tb,1);%boundingBoxes number

if tb\_nb==1 %显示最大目标的boundingBoxes

isplayFrame=insertObjectAnnotation(frame, 'rectangle',tb, 1);

else

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

tracker.Bboxes, tracker.BoxIds);

end

% Display objectCenter piont with 'o'

objectCenter = zeros(tb\_nb,2);objectCoordinate = zeros(tb\_nb,2);

for ii=1:tb\_nb

%计算目标中心位置

objectCenter(ii,:) = [tb(ii,1)+tb(ii,3)/2 tb(ii,2)+tb(ii,4)/2];

%转化为基于图像中心点为原点的坐标

objectCoordinate(ii,:) = [objectCenter(ii,1)-screenCenter(1) ...

screenCenter(2)-objectCenter(ii,2)];

%显示为红圆圈

displayFrame = insertMarker(displayFrame, objectCenter(ii,:),...

'o','color','red','size',8);

end

1. 发送指令
2. 判断时长是否大于间隔
3. 根据目标坐标转换舵机参数
4. 显示指令，传输指令给舵机
5. 显示图像中心点，输出图像
6. 释放视频窗口对象、指令传输端口

if (cputime-currentTime)>interval%set the time interval

%根据目标坐标转换舵机参数

if objectCoordinate

x = objectCoordinate(:,1);y = objectCoordinate(:,2);

if abs(x)>30%横坐标30px内就不要移动了

xAngle = atan(x/distance) \* 180 / pi;

xPara = round(xPara - xAngle \* para);

end

if abs(y)>20%横坐标20px内就不要移动了

yAngle = atan(y/distance) \* 180 / pi;

yPara = round(yPara + yAngle \* para);

end

end

instructs = [num2str(xPara) num2str(yPara)];

% if have objectCoordinate then give the instructs.

if objectCoordinate

displayFrame=insertText(displayFrame,objectCenter(ii,:),...

instructs,'FontSize',10);%显示指令

fwrite(s,instructs);%传输指令给舵机

end

currentTime = cputime;%更新当前时间

end

displayFrame = insertMarker(displayFrame, screenCenter,'+',...

'color','red','size',8);%显示图像中心点

videoPlayer.step(displayFrame);%输出图像

end

release(videoPlayer);

fclose(s)

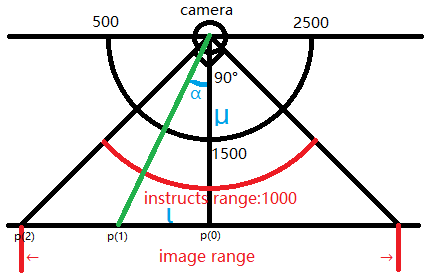
delete(s)

clear s

1. 算法说明
2. 设置指令传输的时间间隔是为了不让云台移动过于频繁，给人观感更佳，同时延长机器寿命。目标横坐标绝对值小于30px纵坐标绝对值小于20px时不更新指令也是出于这个考虑。
3. 限定指令范围，使得左右、上下移动指令介于舵机可处理之间。
4. **关于舵机指令计算问题***（核心问题）*

云台左右旋转有180度范围，舵机接收指令范围[500 2500]，上下旋转有90度范围，指令范围[500 1500]。初始化指令[1500 1500]让云台处于90度中间位置水平方向。

摄像头的左右与上下角度皆为90度，而云台左右旋转有180度。假如目标中心从图像的中心位置移动到最左边或最右边，那么云台应当相应向左或向右旋转45度，整个旋转的范围有90度。因此指令跨距应为指令范围的一半：



由上图可以看出：当目标中心从中间往左右移动，从等距移动上说，移动的范围越小，对应的角度增量越大，而移动的范围越大，对应的角度增量越小。如图所示，设在图像中处于左侧中点位置，从到与从到的图像跨距相同，但很显然到对应的转角大于到所对应的转角。从而可以得知角度与像素跨距的比例是不均匀的，在等像素跨距条件下，越靠近中心，角度越大，越靠近边缘，角度越小。因此不能用简单的比例对应来计算指令值，用指令跨距与图像长宽相比作为参数比例：是错误的。可靠做法是先根据像素跨距求得对应角度，再由角度转换成指令。

设为旋转角度，为像素跨距，为摄像头到目标的映射距离，为摄像头角度，为图像宽度，则：



因为像素跨距已知，，所以用反三角函数即可算出对应角度：



角度与指令跨距的比例是均匀的，对应比例关系为：



由于舵机接收指令后，云台状态跟随改变，目标坐标亦重新计算，因此下一次指令应基于上一次的指令，计算相对位置。本次应旋转的角度与角度比例相乘即为应格外移动的舵机指令值。由于目标向左（右）移动，图像中目标向右（左）移动目标坐标，因此舵机指令值：

