Artificial intelligence video clips based on dual-view camera calibration

Author: Bofeng Cao

Instructor: Professor R.Thamas

2023.4.16

Problem Statement

With the continuous development of artificial intelligence technology, video processing and analysis has become a research hotspot in the field of computer vision. In today's live broadcast and replay of sports events, intelligent editing has become a trend. Dual-view camera calibration (dual-view camera calibration) is one of the key techniques, which involves estimating the intrinsic and extrinsic parameters of the camera in order to establish an accurate geometric correspondence between two cameras with different viewing angles. Higher quality and more attractive sports video content can be achieved by using dual-view camera calibration technology. The goal of this research is to develop an artificial intelligence editing technology based on dual-view cameras applied to different sports events, using the correspondence between dual views for event detection and video frame quality assessment. However, in practical applications, this technology still faces some challenges, such as environment complexity, dynamic scene changes, and computing resource constraints, etc. To achieve the above goals, we will face the following problems:

1. How to accurately estimate the intrinsic and extrinsic parameters of the camera?

2. How to deal with the change of dynamic scene and track the target accurately?

3. How to define the occurrence of an event and accurately locate the video frame where the event occurred?

4. How to compare the quality of video content from different perspectives during the event?

By solving the above problems, we expect to develop an artificial intelligence-based video editing technology that supports various sports events. This will help to bring viewers a more vivid, compact and engaging sports viewing experience.

Input:

- Video streams from two different perspectives
- The definition of the event
- Extrinsics parameters of camera
  - Height
  - Angle
- Intrinsics parameters of camera
  - Focal length
  - Pixel size

Output:

- An edited video with different perspectives

Assumption:

Due to time constraints, I made a simplification of the above question.

- Events happen in simple scenarios
- The timelines of two video streams from different perspectives are consistent
- The internal parameters of cameras with different viewing angles are consistent
- Events are defined as the occurrence and collision of a single object

Literature Search

In January 2022, Nian Liu, Lu Liu, and Zengjun Sun published the article **Football Game Video Analysis Method with Deep Learning**[1]. A deep learning approach is used to build an event detection model to detect events contained in football videos. The whole model is divided into two stages, where the first stage is used to generate candidate event segments. It divides the football video to be detected into a certain length of frame sequence, and uses a sliding window to scan. Multiple frame sequences within a sliding window form a segment, and each segment is a prediction unit. The frame sequence features in the clip are obtained through the three-dimensional convolutional neural network, and used as the input of each time point of the bidirectional recurrent neural network, and further fused to generate the event prediction of the clip. The second stage is to further process the above results to remove all segments predicted to be non-events. Set thresholds according to the detection effects of various events, filter out event fragments with high probability values, obtain the start and end positions of events by merging, classify and mark, and finally output complete event fragments.

Limitations: This method requires the neural network model to be trained on different football videos. It takes a long time to extract features from video frames. Also, since the method does not track and identify the target, it is not possible to edit clips of a specific target, such as a player.

Background Information

3.1 Principles of Subsystem Calibration

Define a point in camera and world coordinates as $(X_c, Y_c, Z_c)^T$ and $(X_w, Y_w, Z_w)^T$.

According to the camera calibration technology, the relationship of the coordinate system can be known as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Since Xw and Yw of all marked points in the calibration plate are known precisely, the positions of all feature points in the camera coordinate system can be found through this marked point. Arbitrarily place the calibration board at several positions in the public market of cameras and projectors, and let the middle position be the reference plane, then the reference plane can be described as:

$$AX_C^r + BY_C^r + Z_C^r + C = 0,$$

Among them, A, B, C are the parameters of the plane equation, $(X_C^r, Y_C^r, Z_C^r)^{\mathrm{T}}$ is any feature point on the reference plane. Although only three points are needed to form a plane, the accuracy can be improved by using the least square method to solve the plane parameters. Once the parameters of the plane equation are determined, the height of the point on the calibration plate at any other position corresponding to the reference surface can be expressed as:

$$\Delta h = \frac{AX_c + BY_c + Z_c + C}{\sqrt{A^2 + B^2 + 1}}$$

3.2 Dual-view global calibration principle

To realize the global calibration of dual-view, the coordinate transformation of the dual-camera coordinate system must be deduced first. In Figure 1, the left camera is called camera 1, the corresponding plane calibration plate on the left of the dual-plane target is called calibration plate 1, the right camera is called camera 2, and the corresponding plane calibration plate is called calibration plate 2. Taking the camera 1 as the reference coordinate system, set the homogeneous coordinates of a feature point P on the calibration board 2 as P_{b2}^{b2} on its coordinate system, and the homogeneous coordinates on the calibration board 1 as P_{b1}^{b2} . The homogeneous coordinates in the camera 1 coordinate system are P_{c1}^{b2}, and the homogeneous coordinates in the camera 2 coordinate system are P_{c2}^{b2}. Then there is the transformation:

$$\begin{cases} P_{b1}^{b2} = T_{b2,b1} P_{b2}^{b2} \\ P_{c2}^{b2} = T_{b2(i),c2} P_{b2}^{b2} \\ P_{c1}^{b2} = T_{b1(i),c1} P_{b1}^{b2} = T_{c2,c1} P_{c2}^{b2} \end{cases}$$

In the formula: $T_{b2,b1}$ represents the transformation matrix from the calibration board 2 coordinate system to the calibration board 1 coordinate system, $T_{c2,c1}$ represents the transformation matrix from the camera 2 coordinate system to the camera 1 coordinate system, $T_{b2(i),c2}$ represents the ith pendulum In the placement position, the conversion matrix from the calibration board 2 coordinate system to the camera 2 coordinate system, $T_{b1(i),c1}$ is the same. It can be obtained from the above formula:

$$T_{b1(i),c1} T_{b2,b1} = T_{c2,c1} T_{b2(i),c2}$$

Considering the i-th and j-th placement positions of the biplane target, combined with the two constraint conditions that the positional relationship between the two calibration boards and the two cameras is also the same, it can be obtained :

$$\begin{cases} T_{b2,b1} = {T_{b1(i),c1}}^{-1} T_{c2,c1} T_{b2(i),c2} \\ T_{c2,c1} = T_{b1(i),c1} T_{b2,b1} {T_{b2(i),c2}}^{-1} \\ T_{b2,b1} = {T_{b1(j),c1}}^{-1} T_{c2,c1} T_{b2(j),c2} \\ T_{c2,c1} = T_{b1(j),c1} T_{b2,b1} {T_{b2(j),c2}}^{-1} \end{cases}$$

Make $T_A = T_{b1(j),c1} {T_{b1(i),c1}}^{-1}$, $T_B = T_{b2(j),c2} {T_{b2(i),c2}}^{-1}$, $T_C = {T_{b1(i),c1}}^{-1} T_{b1(i),c1}$, $T_D = {T_{b2(i),c2}}^{-1} T_{b2(i),c2}$, $T_X = T_{c2,c1}$, $T_Y = T_{b2,b1}$. Then the system of equations can be simplified as:

$$\begin{cases} T_A T_X = T_X T_B \\ T_C T_Y = T_Y T_D \end{cases}$$

$T_A$andT_B in $T_A T_X = T_X T_B$ can be obtained through camera calibration. In this way, the problem of solving T_X is transformed into the problem of solving the hand-eye calibration equation AX=XB proposed by Tsai et al[2].

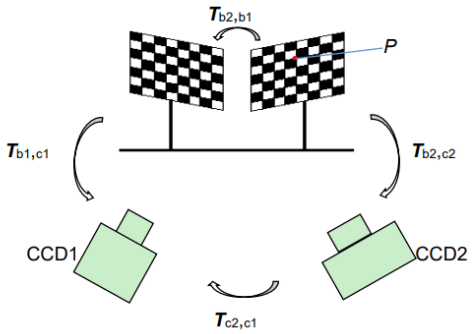


fig.1: Global calibration diagram

3.3 Recognition and Tracking of Moving Targets

3.3.1 Temporal Difference

Since the target in the scene is moving, the position of the target's image in different image frames is different. This algorithm performs differential calculation on two or three consecutive frames of images in time, and subtracts the pixels corresponding to different frames to determine the absolute value of the gray level difference. When the absolute value exceeds a certain threshold, it can be judged as a moving target, thus Realize the detection function of the moving target.

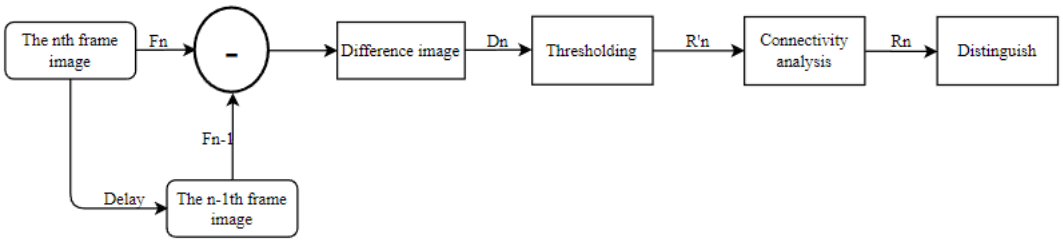


Fig.2 Two frame difference method

The operation process of the two-frame difference method is shown in the figure above. The images of the nth frame and the n-1th frame in the video sequence are recorded as f_n and fn-1, and the gray value of the corresponding pixel of the two frames is recorded as $f_n(x, y)$ and $f_{n-1}(x, y)$, subtract the gray values of the corresponding pixels of the two frames of images, and take their absolute values to obtain the difference image D_n:

$$D_n(x,y) = |f_n(x,y) - f_{n-1}(x,y)|$$

The threshold T is set, and the pixel points are binarized one by one to obtain a binarized image R_n'. Among them, the point with a gray value of 255 is the foreground (moving target) point, and the point with a gray value of 0 is the background point; the connectivity analysis of the image R_n' can finally obtain the image R_n containing the complete moving target.

$$R'_n(x,y) = \begin{cases} 255, D_n(x,y) > T \\ 0, else \end{cases}$$

3.3.2 Histogram Projection

The frame difference method can usually detect the changed area in the image, that is, the foreground area. However, in the foreground area, there may still be some noise or misdetected objects, so further processing is required on the foreground area to determine the precise position and shape of the object.

Foreground regions can be further analyzed using a histogram projection. First, the foreground area is binarized to separate the target from the noise. Then, histogram projection is performed on the target area to calculate the distribution of pixel values in the target area. Finally, the location and shape of the target are determined according to the histogram projection results. This method can effectively improve the accuracy of target recognition, especially when the target color is relatively stable. To sum up, it can be divided into the following steps:

1. Calculate the histogram M of the model

2. Calculate the histogram I of the target image

3. Divide M by I to get the ratio histogram R

4. Cycle each pixel point I(x, y) according to the pixel value mapping into the distribution probability of the histogram R

5. Convolution operation

6. Normalization and realistic output

Implementation

In MATLAB, we can use the camera calibration function in Computer Vision Toolbox to calibrate the dual-view camera. The following is the MATLAB dual-view camera calibration process:

1. Prepare data: First, you need to prepare a set of calibration images taken by two cameras. These images usually include photographs of a calibration object (such as a checkerboard or circular calibration plate) in different positions and orientations.

2. Import data: In MATLAB, use the imageSet function to import the calibration images of the two cameras:

```
imageDir1 = 'calibration_camera1';
imageDir2 = 'calibration_camera2';

images1 = imageSet(fullfile(imageDir1));
images2 = imageSet(fullfile(imageDir2));
```

3. Detect feature points: Use the detectCheckerboardPoints function to detect checkerboard corner points in the calibration image:

```
[imagePoints1, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames1);
[imagePoints2, ~, ~] = detectCheckerboardPoints(imageFileNames2);
```

4. Generate world coordinates: Use the generateCheckerboardPoints function to generate the world coordinates of the corner points of the checkerboard:

```
worldPoints = generateCheckerboardPoints(boardSize, squareSize);
```

5. Single camera calibration: Use the estimateCameraParameters function to estimate the internal parameters and external parameters of the two cameras respectively:

```
[cameraParams1, ~, estimationErrors1] = estimateCameraParameters(imagePoints1, worldPoints, 'WorldUnits', 'mm', 'EstimateSkew', false, 'EstimateTangentialDistortion', false, 'NumRadialDistortionCoefficients', 2);
[cameraParams2, ~, estimationErrors2] = estimateCameraParameters(imagePoints2, worldPoints, 'WorldUnits', 'mm', 'EstimateSkew', false, 'EstimateTangentialDistortion', false, 'NumRadialDistortionCoefficients', 2);
```

6. Dual-view camera calibration: use the stereoParameters function to estimate the rotation and translation matrix between the dual-view cameras:

```
stereoParams = stereoParameters(cameraParams1, cameraParams2);
```

Afterwards, an attempt was made to stitch panoramic images using the results of the camera calibration described above. But this is not the main purpose of the technique, so I won't discuss it here.

In the next step, I used matlab to implement a method based on frame difference method and histogram projection to detect and track moving objects in the video.

```
% read video file
video = VideoReader('1.mp4');
frame = readFrame(video);
% define static area
imshow(frame);
staticRegion = round(getPosition(imrect));
% Convert to grayscale image
grayFrame = rgb2gray(frame);
% Initialize the frame difference method
prevFrame = grayFrame;
% Initialize the output video
outputVideo = VideoWriter('output_video.avi');
open(outputVideo);
% Flag variable, record whether it is the first frame
isFirstFrame = true;
% is used to save the position of a frame where the object first enters the video
firstFramePos = [];
% is used to save the position of the frame where the collision occurred
collidedFrames = [];
% Initialize the counter
frameCounter = 1;
while hasFrame(video)
    % Read current frame and convert to grayscale image
    frame = readFrame(video);
    grayFrameNext = rgb2gray(frame);
    % If the current frame is a multiple of 3, perform target detection and tracking
    if mod(frameCounter, 3) == 0
        % frame difference method
        diffFrame = imabsdiff(grayFrameNext, prevFrame);
        diffThresh = graythresh(diffFrame);
        binaryDiff = imbinarize(diffFrame, diffThresh);
        binaryDiff = bwareaopen(binaryDiff, 50);
        % Histogram projection
        hsvFrame = rgb2hsv(frame);
```

```
        hueFrame = hsvFrame(:, :, 1);
        histHue = histcounts(hueFrame(binaryDiff), 0:1/255:1);
        hueMask = histeq(hueFrame, histHue);
        % Binarized histogram projection results
        hueMask = imbinarize(hueMask, graythresh(hueMask));
        hueMask = bwareaopen(hueMask, 50);
        % Detect moving objects using area attributes
        [labeledRegions, numRegions] = bwlabel(hueMask);
        regionProps = regionprops(labeledRegions, 'Centroid', 'BoundingBox', 'Area');
        % Extract the object with the largest area
        if numRegions > 0
            [~, maxAreaIdx] = max([regionProps.Area]);
            target = regionProps(maxAreaIdx);
            % If it is the first frame, record the position of the frame where the object first entered the video
            if isFirstFrame
                isFirstFrame = false;
                firstFramePos = video.CurrentTime;
            end
            % Determine whether the target enters the static area
            if rectint(target.BoundingBox, staticRegion) > 0
                collidedFrames = [collidedFrames; video.CurrentTime];
            end
            % Superimpose the detected objects on the original frame
            frameWithTarget = frame;
            frameWithTarget = insertShape(frameWithTarget, 'Rectangle', target.BoundingBox, 'LineWidth', 2, 'Color', 'green');
            frameWithTarget = insertMarker(frameWithTarget, target.Centroid, 'x', 'Color', 'red', 'Size', 8);
          else
              frameWithTarget = frame;
        end
```

```
         % update grayscale image
         prevFrame = grayFrameNext;
    end
    % Superimpose the static area on the original frame
    frameWithTarget = insertShape(frameWithTarget, 'Rectangle', staticRegion, 'LineWidth', 2, 'Color', 'red');
    % Show results
    imshow(frameWithTarget);
    drawnow;
    % Add current frame to output video
    writeVideo(outputVideo, frameWithTarget);
    % Increment frame counter
    frameCounter = frameCounter + 1;
end
% close output video
close(outputVideo);
if isFirstFrame
    disp('Target did not enter the video!');
else
    disp(['Target entered the video at ' num2str(firstFramePos) ' seconds.']);
end
if ~isempty(collidedFrames)
    disp('Target collided with the static region at the following times:');
    disp(collidedFrames);
else
    disp('Target did not collide with the static region.');
end
```

This code is a MATLAB based object detection and tracking algorithm. It uses frame difference method to detect moving objects in video, and uses histogram projection to track objects. Here is a detailed explanation of the code:

First read the video file and get the first frame. Then define the static area, display the first frame, let the user use the mouse to draw a rectangular area on the image as the static area. Convert the first frame to a grayscale image. Initialize the frame difference method, and initialize the first grayscale image as the previous frame.

Initialize the output video, creating an output video file named 'output_video.avi'. Initializes a counter for counting the number of frames processed.

Next is a while loop that first reads the current frame and converts it to a grayscale image. Perform object detection and tracking every 3 frames to speed up detection. Use the frame difference method to calculate the difference between the current frame and the previous frame, and binarize it. Convert the current frame to HSV color space using a histogram projection and compute the hue histogram in the difference image. Matches the hue channel of the current frame to the hue histogram in the difference image using a histogram equalization method. Binarize the histogram projection result and remove small connected regions. Use the area attribute to detect moving objects and extract the object with the largest area. If an object is detected, superimpose the object on the original frame. If it's the first frame, record the position of the first frame the object entered the video. Determine whether the target has entered the static area, and if so, record the position of the frame where the collision occurred. Update the grayscale image. Superimpose the static area on the original frame and display the result. Add the current frame to the output video. Increment frame counter.

After the loop finishes, turn off the output video. If the target does not enter the video, a prompt message is displayed. If the target entered the video, the time when they first entered the video is displayed. If the target collides with the static area, the time at which the collision occurred is displayed. Otherwise, a prompt message that no collision occurs is displayed.

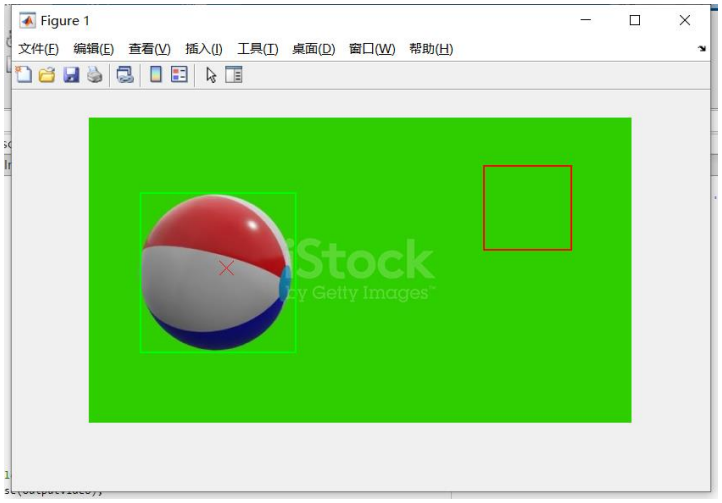
Also, I have some simple implementations for comparing the quality of different video frames.

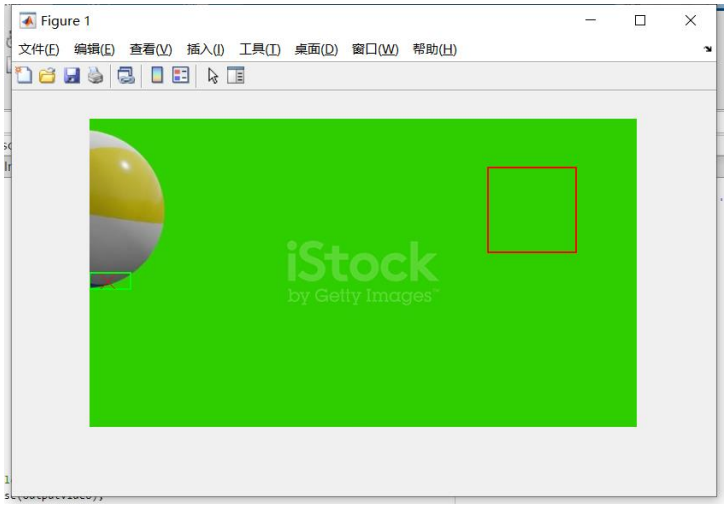
```
% Read image
image = imread('image.jpg');
if size(image, 3) == 3
% Convert to grayscale image
grayImage = rgb2gray(image);
else
grayImage = image;
end

% Compute horizontal and vertical gradients using Sobel operator
Gx = imgradient(grayImage, 'Sobel', 'horizontal');
Gy = imgradient(grayImage, 'Sobel', 'vertical');

% Compute gradient magnitude
G = sqrt(Gx.^2 + Gy.^2);

% Compute average gradient magnitude
meanGradient = mean(G(:));

% Output result
fprintf('Average gradient magnitude: %.2f\n', meanGradient);
```

The code above reads an image, converts it to grayscale if it's a color image, and then computes the gradient magnitude using the Sobel operator. Finally, it calculates the average gradient magnitude and prints the result. Computing the average gradient magnitude of an image can help us evaluate the overall texture and edge information of the image. Higher gradient magnitudes generally indicate more pronounced edge and texture changes in the image, while lower gradient magnitudes indicate smoother images.

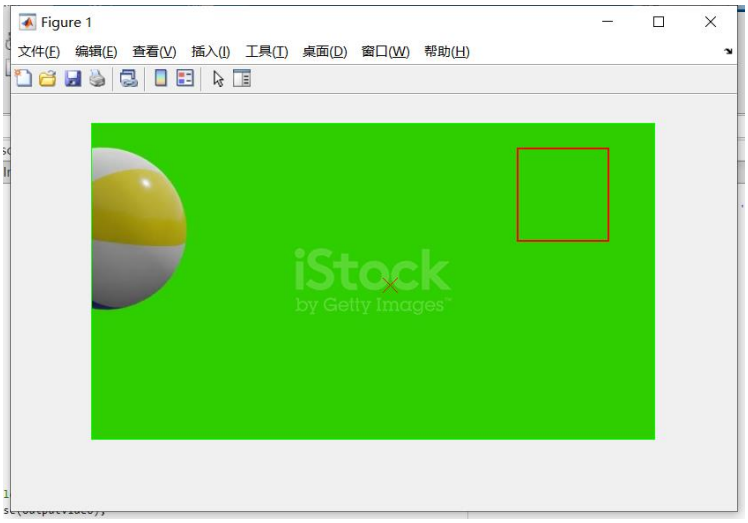
Simulation results

After running the above matlabcode, we can quickly locate and track the ball in the video. And record the time when the ball appears and the position of the corresponding video frame (event start) and the time when the ball enters the selected area (subject of the event). Through these records, we can quickly edit the video and output the video corresponding to the target event.





Although I optimized the detection area using histogram projection and erosion and dilation. However, when the target appears incomplete for a few frames, this method is very inaccurate in selecting the target area.



I think the possible reasons for this are:

Frame difference method sensitivity: When the target is moving slowly or the motion pattern changes, the frame difference method may have difficulty detecting significant changes. This may cause part of the target area to be misjudged as a static background, thereby affecting the subsequent target detection process.

Threshold setting: use the graythresh() function in the algorithm to determine the binarization threshold. This function automatically calculates the optimal threshold using Otsu's method, but in some cases this threshold may be too high or too low, resulting in incomplete target regions.

Noise and clutter: Noise and clutter in the video can affect the performance of frame difference and histogram projection. For example, lighting changes, shadows, or other moving objects can interfere with object detection.

To improve this problem, try the following:

Adjust Threshold: Try manually setting different binarization thresholds to find the best threshold for the current scene.

Use morphological operations: In the binarization image processing stage, you can try to use morphological operations (such as dilation, erosion, opening operations, closing operations, etc.) to improve the integrity of the target area.

Use more complex object detection algorithms: You can try to use more complex object detection algorithms, such as methods based on machine learning or deep learning (such as YOLO, SSD, etc.), these methods are usually more robust to incomplete objects.

Tracking algorithm: Introduce tracking algorithms (such as Kalman filter, optical flow method, etc.) to track the movement of the target in consecutive frames. This can help the algorithm use information from previous and subsequent frames to improve object detection when the object is incomplete.

Conclusion and Future Work

This paper aims to develop a fast editing technique based on dual-view camera calibration. By calibrating two cameras with different viewing angles, it is possible to better judge the occurrence of events and compare the quality of video frames from different viewing angles. At present, I have completed the calibration of two cameras with different viewing angles, and used the frame difference method combined with histogram projection to track the target and simply judge and record the occurrence and end of the event. However, the frame difference method combined with histogram projection cannot track the target completely and accurately, which leads to inaccurate judgment of the node where the event occurs and ends. In the future, I hope to add a neural network model to this algorithm to better handle event detection in complex scenes.

In addition, I also want to make some summary of my learning process in the whole 4OJ4. Throughout the semester, I've discovered that there is a difference between researching and developing a technology. To develop a technology, it is often necessary to consider more different aspects of technology and integrate these parts. And research is often more aimed at a detailed part for exploration. Throughout the 4OJ4 learning process, I first learned how to identify research topics. I think my first attempt at this was not successful. My topic selection was too large and there were too many technical details that needed to be paid attention to. There is no way to do it well. Research on a certain aspect (such as how to track objects more accurately, etc.). What's more, I learned how to explore and try an unknown field independently, which is different from the way I learn knowledge in other courses. I need to start from the generality of an article and then gradually refine it to the different technical details. The above are some summaries of my learning process for the whole semester.