


Course Project One for Digital Image Processing 2025

Dr. Xiaoming Peng

YOPO – You Only Pick Once

杨继一	2022190908033
任津铭	2022190908020
刘玉浩	2022190908022

 <https://github.com/Marcobisky/YOPO>

Introduction

This project aims to automatically track the red and blue cars as much as possible. The given image sequences contain 200+ frames from a continuously recorded video each. The detection method must be reliable without using advanced technology like deep learning and computer vision methods.

Objectives:

- Select an appropriate template from the first frame
- Design a mathematical algorithm to track the colored cars

Challenges:

- The size and color of the car is changing as the car is driving along the road
- Finding the best matching image in the frame leads to computational expense

Methodology

Algorithm overview

In our study, the logical structure of a fully detection algorithm is first designed. To begin with, 'kernel' is used as the core recognition reference, which is a matrix that contains the car's content. The first reference kernel is marked from the first frame. After that, for each of the remaining frames, the kernel will search candidates within a calculated area for a better kernel for the output detection of cars. If the candidate outperforms the previous kernel, the kernel will be updated for the next frame. By doing the detection operation repeatedly, the car can be located from each frame. Figure-1 shows the overview of our algorithm and the labor division of our team.

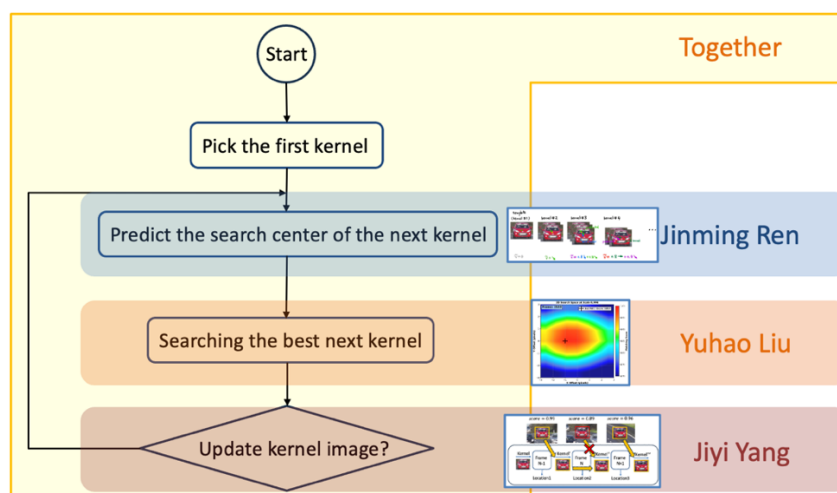


Figure-1: The overview algorithm

Kernel initialization

For the algorithm, the kernel is the key to continuous and reliable detection. Therefore, the selection of the first kernel is vitally important. In our study, the initial kernel is picked from the first frame. The content of the kernel consists of the RGB image of the car and its location. The RGB image is stored in 3 channels. The location information is stored with the x-y axis of the top-left corner of the rectangular frame and the width and height of the rectangular frame.

Motion prediction

Motion prediction is a key process of the algorithm. The reason for predicting the car's motion comes from two reasons. The first one is to save computation resources. During tracking, it is impossible to search the whole image frame with the kernel because it will lead to huge time latency and massive computational operations. The second reason is that the car is moving along a regular street continuously. Motion prediction is possible to better locate the car in a specific area.

As for the prediction algorithm, we develop a moving average method to predict a rough area where the car appears with a high possibility. Since the car is driving along a straight street, the position of the car is highly related to its velocity. Therefore, based on the previous tracks, the location of the subsequent frame can be easily predicted. Here is the algorithm of the moving average method:

$$v_{n+1} \approx \alpha v_n + (1 - \alpha)v_{n-1} \quad (\alpha \in [0, 1])$$

Where v_{n+1} is the velocity of the $n + 1$ frame, v_n is the velocity of the n frame, and v_{n-1} is the velocity of the $n - 1$ frame. α is the weight of the previous frame, referring to how seriously the previous positions influence the subsequent position.

In our study, by several trials, the weight is set to be 0.8, which means the last frame dominates at 0.8 weight, and all the frames before the last frame dominate with exponential weight. For a special case, the velocity of the car in the first frame is 0, and the motion prediction of the second frame is 0 as well, which means the center of the searching area of the second frame is the same as the first frame (reference frame). Figure-2 presents the demonstration of the moving average method frame by frame.

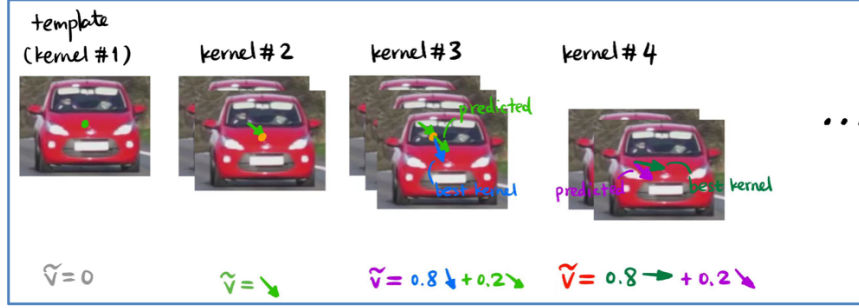


Figure-2: The moving average method

Car searching

After predicting the rough location of the car on the subsequent frame, the next process is to find the car by comparing it with the reference kernel which contains the RGB content of the colored car. Considering the changing location will lead to changing size and brightness of the car, the kernel will be enlarged and shrunk for a better match. Secondly, since we only predict a rough location of the searching area, the actual searching region is extended to three pixels per boundary, aiming to search the car reliably.

For the best match selection algorithm, Normalized Cross-Correlation (NCC) is mainly used. The NCC is a classic template matching algorithm that is invariant to linear illumination changes. It evaluates the similarity between two image regions by calculating their statistical correlation. The formula is given by:

$$NCC(u, v) = \frac{\sum_{x,y} [T(x, y) - \bar{T}] [I(u + x, v + y) - \bar{I}_{u,v}]}{\sqrt{\sum_{x,y} [T(x, y) - \bar{T}]^2 \sum_{x,y} [I(u + x, v + y) - \bar{I}_{u,v}]^2}}$$

Where I is the image region, and T is the template kernel.

The value of NCC ranges from -1 to 1. The closer the NCC value is to 1, the more similar the two regions are. Therefore, we can easily find out the best match image region as the detection result of this frame.

The template kernel is also resized into 0.98, 0.99, 1.01 and 1.02 scales for better selection. For each size, we repeat the NCC calculation and choose the best image region that matches. Figure-3 shows the selection process within a search area.

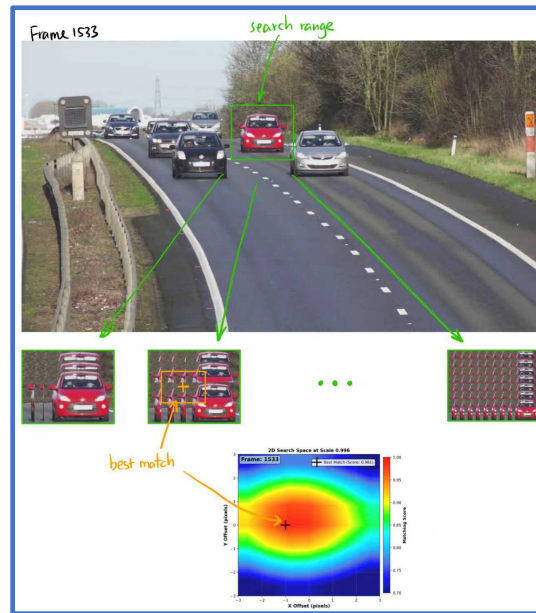


Figure-3: Demonstration of the search algorithm

Kernel updating

For the last step of our algorithm, we design an updating rule for the kernel for self-optimizing recognition. The reason for this design is to adjust the kernel along the track to better match the car. It is also a solution to meet the standard of reliable tracking. The core idea of the updating rule is to compare the NCC value of the best-matching one with the NCC threshold. If the score is greater than the threshold, the kernel will be updated to the present kernel, both the content and position label, otherwise only the position label of the kernel is updated.

Through several trials, the threshold of NCC is 0.94 is chosen. Figure-4 shows an example of the kernel updating rule in three frames.

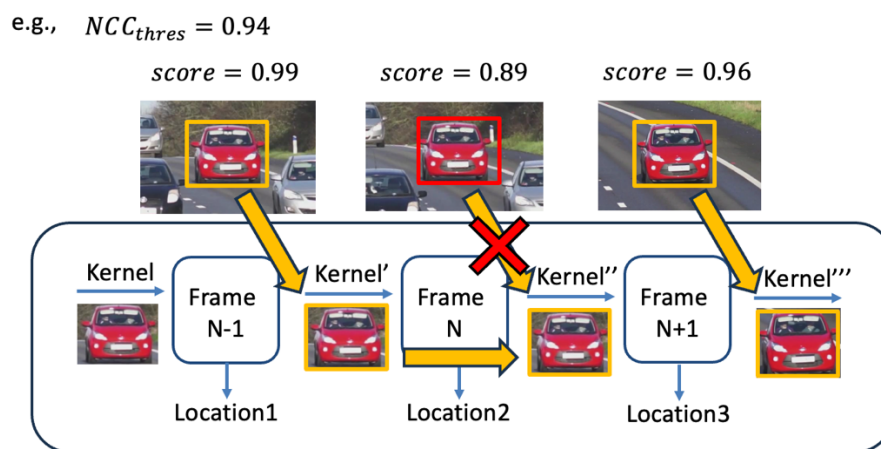


Figure-4: Kernel updating rule

Results

Through the steps of recognition, colored cars are easily located in each frame with pretty good performance. Figure-5 shows the final trajectory the car generates in the final frame. The full recognition dynamic graph can be found in the appendix files.



Figure-5: The trajectory of the car

At the same time, we provide a heat map of the searching area, showing in the NCC value of the best match kernel of each frame in different locations, where the deeper the color, the greater the value. The '+' mark on the heat map shows the local maximum value among all points. It is also a dynamic graph, appended to the appendix files.

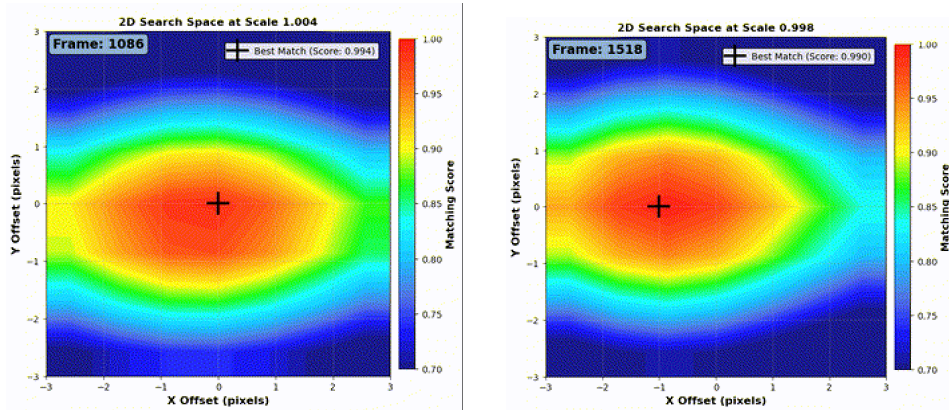


Figure-6: Heat map of the best match kernel

Conclusion

In conclusion, the tracking algorithm we designed performs very well in this task. The colored car is easily located with a bounding box that is very tight. For the two main challenges of this task, each of them is solved well. The problem of changing object size and color is solved by updating the template kernel in time. The computational expense in finding the best-matching image is solved by moving prediction to narrow the search area. All the pictures of the detected cars are collected in the appendix files.