Research on LK Optical Flow Algorithm with Gaussian Pyramid Model Based on OpenCV for Single Target Tracking

Jing Bai\textsuperscript{1,a} and Lili Huang\textsuperscript{2,b}

\textsuperscript{1}School of Information Science and Technology, East China Normal University, 200062.
\textsuperscript{2}School of Information and Technology, Shanghai Jian Qiao University, 201319, China

\begin{itemize}
\item \textsuperscript{a}Phone: 18016297367, E-mail: 15008@gench.edu.cn
\item \textsuperscript{b}Phone: 15001817383, E-mail: 1577339166@qq.com
\end{itemize}

Abstract. There is a problem that the tracker of TLD algorithm in computer vision tracking will lose tracking target or follow the wrong one when the target is fast moving or blocked. To solve this problem, this paper proposes an improved LK optical flow algorithm with Gaussian pyramid model for single target tracking, through theoretical analysis and formula deduction. This paper applied this algorithm and the traditional LK optical flow algorithm with the unified video to compare the experimental results with the OpenCV visual library in Visual Studio 2012. The result shows that this algorithm has the effect of improving tracking accuracy and adaptability.

1. Introduction

Visual tracking is a hot topic in the field of computer vision. It is usually applied to lane departure detection, vehicle speeding detection, license plate recognition, behavior detection, falling objects detection and other scenes. However, there may be problems to visual tracking in some complex scenes during a long time, such as the target is moving fast or blocked and so on\cite{1-2}. TLD (Tracking-Learning-Detection) visual tracking algorithm combines detection and tracking while adding a learning mechanism, which making the overall target tracking is more stable and effective\cite{3}.

However, the tracker of the TLD algorithm is implemented with the LK optical flow algorithm. It will lose tracking target or follow the wrong one when the target is fast moving or blocked. This paper proposes an improved optical flow algorithm with Gaussian Pyramid model as the algorithm of TLD tracker, which improves the accuracy and adaptability of tracking.

2. Implementation of tracking algorithm

The TLD algorithm is a visual tracking algorithm which was proposed by Zdenek Kalal from 2008 to 2010 and constantly improved. The TLD algorithm is composed of tracker, detector and the online learning module. The tracker tracks the motion of the target between successive frames. The tracker can achieve its function when the target is always visible. The tracker can predict the region of the target of the current frame from the result of the previous frame, thus obtaining a positive sample trajectory. The detector corrects the tracker. Detector traverses the image of the current frame to find all possible positions similar to the target object, and obtains the positive and negative samples from the detection result. It takes the most trusted position of all detection windows as the TLD output. The
learning module generates the corresponding classifier by iterating and training according to the positive and negative samples obtained from tracker and detector so as to improve performance and correct the corresponding parameters[4]. The process of TLD algorithm is shown in Figure 1.

**Figure 1.** Process of TLD algorithm

### 2.1. LK (Lucas-Kanade) optical flow algorithm

The tracker of TLD is based on the LK optical flow algorithm which requires the following three assumptions:

- The brightness is constant. It is assumed that the surface color of the target is constant during the movement, that is, the brightness of the pixels in the two frames in the image is constant.
- Time is continuous or the movement is a movement of small amplitude. Changes in the motion of target in the image are slow, that is, the displacement of the target is small between the two continuous frames.
- The optical flows are consistent. In the image, the motion of adjacent pixels on the surface of the same target is consistent, and these pixels must be aggregated in the same region[5].

Assuming that the optical flows in the grayscale region centered on the M point are the same, the calculation formula of the optical flow can be converted to the formula (1) by giving different weights to different pixels.

$$\sum_{x \in \Omega} W(x,y)[I_t \cdot V_m + I_t]^2 = 0$$  

(1)

Ω represents a grayscale region centered on M points, and W(x,y) represents the weight function of the region. From the formula to draw a conclusion that the closer to the M, the higher the weight, the farther away from the M, the lower the weight.

The solution of the formula (1) can be calculated by the formula (2).

$$A^TW^2Av = A^TW^2b$$  

(2)

Assuming that there are N*N pixels in the region, the matrix variable s of the formula (2) can be expressed as formula (3), formula (4), and formula (5).

$$A = \begin{bmatrix} V_1(x_1,y_1) & V_1(x_1,y_2) & \ldots & V_1(x_1,y_N) \\ V_1(x_2,y_1) & V_1(x_2,y_2) & \ldots & V_1(x_2,y_N) \\ \vdots & \vdots & \ddots & \vdots \\ V_1(x_N,y_1) & V_1(x_N,y_2) & \ldots & V_1(x_N,y_N) \end{bmatrix}$$  

(3)

$$W = \begin{bmatrix} w(x_1,y_1) & w(x_1,y_2) & \ldots & w(x_1,y_N) \\ w(x_2,y_1) & w(x_2,y_2) & \ldots & w(x_2,y_N) \\ \vdots & \vdots & \ddots & \vdots \\ w(x_N,y_1) & w(x_N,y_2) & \ldots & w(x_N,y_N) \end{bmatrix}$$  

(4)

$$b = \begin{bmatrix} I_1(x_1,y_1) & I_1(x_1,y_2) & \ldots & I_1(x_1,y_N) \\ I_1(x_2,y_1) & I_1(x_2,y_2) & \ldots & I_1(x_2,y_N) \\ \vdots & \vdots & \ddots & \vdots \\ I_1(x_N,y_1) & I_1(x_N,y_2) & \ldots & I_1(x_N,y_N) \end{bmatrix}$$  

(5)

Solve equation to get the formula (6).

$$v = (A^TW^2A)^{-1}A^TW^2b$$  

(6)

Among, $A^TW^2A$ is a matrix of 2*2. Convert the form to get the formula (7).
\[
A^T W^2 A = \left[ \frac{\sum W^2 I_x^2}{\sum W^2 I_x I_y} \quad \frac{\sum W^2 I_x I_y}{\sum W^2 I_y^2} \right] \tag{7}
\]

If the \( A^T W^2 A \) of formula 7 is non-singular, it must have two characteristic roots \( \lambda_1 \) and \( \lambda_2 \) which are not equal to 0. When we set the \( \lambda_1 > \lambda_2 \), we can get the following conclusion:

- If \( \lambda_1 > \tau \), \( \lambda_2 < \tau \), we can calculate the optical flow using the formula (6).
- If \( \lambda_1 > \tau \), \( \lambda_2 < \tau \), we can’t get the complete optical flow information.
- If \( \lambda_1 < \tau \), \( \lambda_2 < \tau \), we can’t calculate the optical flow.

2.2. Gauss Pyramid LK optical flow algorithm

2.2.1 Principle of algorithm When calculate the optical flow, we first need to set up a window of adjacent areas. Calculation of the optical flow is more robust when the window is large, while calculation of the optical flow is more correct when the window is small. The reason is that when the motion of each part of the image is not consistent, if the window is too large, it is easy to go against the assumption which the optical flows must be consistent. This may not be realistic. So, smaller the window is and fewer pixels the window contains, more accurately we calculate. There is a problem that the above mentioned assumptions can not be achieved, when the movement range is large. To solve this problem, this paper introduces the Gaussian Pyramid model. Set a fixed window, and then generate a pyramid image. In each layer of the pyramid, calculate optical flow with the same size (resolution) of window. When the size of image is small, the window appears large. Then the optical flow can track the fast target. While, when the size of image is original size, the optical flow window is relatively small and the optical flow is more accurate.

Image pyramid is a set of images, and all multi-resolution images come from the same original image. This paper uses a down-sampled Gaussian Pyramid model. First, calculate the optical flow of the top level image of the pyramid. Then calculate the initial optical flow of the second-level according to the top level optical flow result, and further estimate the exact value of optical flow. Finally, calculate the initial optical flow of the next level, using the calculated second-level optical flow result and calculate its exact value. Put the calculated exact value into the next level for calculation until the bottom of the pyramid[6]. The principle is shown in Figure 2.

![Figure 2. Principle of Gaussian Pyramid](image-url)
Assume the original image is \( I(x, y) \) whose size is \( \times N \), it is to say that the image of Level 0 is \( I^0 \) which is at the bottom of pyramid. The size of \( I^0 \) is the same as the size of the original image, whose size is \( M \times N \) too. It is mean that \( I^0 = I \).

The way, the Gaussian pyramid model decreases downward, is as shown in formula (8).

\[
I^l(x, y) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m, n) I^{l-1}(2x + m, 2y + n) \tag{8}
\]

Wherein, for the \( i \)th level, \( I^l = \text{Reduce}[I^{l-1}] \).

\( w(m, n) \) is the generated kernel. The window function is a low-pass filter. The limitation of the generated kernel is to ensure the low-pass quality and to ensure that the brightness of the image, after expansion and contraction, is smooth without boundary cracks.

Pyramid is established in the way of downwards 1/2 sampling. Let \( L \) (\( L = 1, 2, \ldots \)) represent the number of levels in the pyramid model, and the images are expressed as \( I^1, I^2, \ldots \). The length and width of the \( L \) level image are represented by \( N^L \) and \( M^L \). \( L \) level image of the pyramid \((I^L)\) can be represented as a formula (9):

\[
I^L = \frac{1}{4}I^{l-1}(2x, 2y) + \frac{1}{8}[I^{l-1}(2x - 1, 2y) + I^{l-1}(2x + 1, 2y) + I^{l-1}(2x, 2y - 1) + I^{l-1}(2x, 2y + 1)] + \frac{1}{16}[I^{l-1}(2x - 1, 2y - 1) + I^{l-1}(2x + 1, 2y - 1) + I^{l-1}(2x + 1, 2y - 1) + I^{l-1}(2x - 1, 2y + 1)]
\]

The formula (9) performs Gaussian low-pass filtering on the image while downwards sampling to prevent image distortion. At the same time, some pixel value around the image, formulas (10), must be defined to satisfy the formula (9).

\[
\begin{align*}
I^{l-1}(-1, y) &= I^{l-1}(0, y) \quad (10a) \\
I^{l-1}(x, -1) &= I^{l-1}(x, 0) \quad (10b) \\
I^{l-1}(-1, -1) &= I^{l-1}(0, 0) \quad (10c) \\
I^{l-1}(N^{l-1}, y) &= I^{l-1}(N^{l-1} - 1, y) \quad (10d) \\
I^{l-1}(x, M^{l-1}) &= I^{l-1}(x, M^{l-1} - 1) \quad (10c) \\
I^{l-1}(N^{l-1}, M^{l-1}) &= I^{l-1}(N^{l-1} - 1, M^{l-1} - 1) \quad (10d)
\end{align*}
\]

2.2.2 Process of algorithm. Taking into account the amount of computation, the pyramid usually only calculated to 3 to 4 levels. The algorithm can deal with the problem of feature point tracking when optical flow is greater than the window, because of the moving image[7]. To increase the calculation speed of the optical flow field, algorithm with Pyramid LK optical flow is usually used to estimate the optical flow of feature point[8].

Given a pixel \( u \) in image \( I \), calculate the pixel \( v \) in image \( J \) that matches it, making \( v = u + d \).

Create a pyramid model of \( I \) and \( J \):

\[
\{I^l\}_{l=0, 1, \ldots, L_m} \text{ and } \{J^l\}_{l=0, 1, \ldots, L_m}
\]

Initialize the estimate of the optical flow:

\[
g^{L_m} = [0 \ 0]^T
\]

Determine the coordinates of the pixel point \( u \) on the image:

\[
I^l: u^l = [p_x \ p_y]^T = \frac{u}{2^l}
\]

Calculate the partial derivative of the image \( I^L \) in the \( x \) direction with \( u \) as the center:

\[
I^l_{x}(x, y) = \frac{A(xy+1)-A(xy-1)}{2}
\]
gradient matrix:

\[
G = \begin{bmatrix}
\Sigma I_x^2 & \Sigma I_x I_y \\
\Sigma I_x I_y & \Sigma I_y^2
\end{bmatrix}
\]  

(15)

Initialize LK optical flow vector : \( v = [0 \ 0] \) \(^T\). Calculate the pixel difference of the image :

\[
\delta I_k(x, y) = A(x, y) - I^k(x + g^x_k + v^{k-1}, y + g^y_k + v^{k-1})
\]  

(16)

Calculate image mismatch vector :

\[
b_k = [\delta \ \Sigma I_k I_k] \quad \delta = \Sigma I_k I_k
\]  

(17)

Calculate LK optical flow vector :

\[
\eta_k = G^{-1} b_k
\]  

(18)

Estimate the next iteration :

\[
v^k = v^{k-1} + \eta^k
\]  

(19)

The optical flow in the Lth level can be finally determined as :

\[
d^L = v^k
\]  

(20)

Calculate the light flow at L-1 level :

\[
g^{L-1} = [g^x_k^{L-1} \ g^y_k^{L-1}]^T = 2(g^k + d^L)
\]  

(21)

Calculate optical flow vector : \( d = g^0 + d^0 \). Then the corresponding point of the pixel \( u \) in the next frame image \( J \) is :

\[
v = u + d
\]  

(22)

3. Test

This LK optical flow algorithm with Gaussian pyramid model for single target tracking, proposed in this paper, was implemented with the efficient and flexible library function of OpenCV 3.0.0 in Visual Studio 2012. The source video for testing used a self-recorded video with a handheld yoghurt bottle moving. There are some scenes such as target blocked, disappeared, and fast moving in this video.

The tracking effect of the traditional LK optical flow algorithm in the TLD algorithm is shown in Figure 3. The result shows that the tracker will lose target when blocked or partially disappeared, as shown in Figure 3b. When the integral image is restored, the tracker may have a tracking error when it is tracked again, as shown in Figure 3c. When the tracked target is moving too fast, the tracker may also have erroneous tracking, as shown in Figure 3d.
4. Conclusion
There is a problem that the traditional LK optical flow algorithm in TLD can not capture fast moving or blocked objects. To solve this problem, this paper proposes an improved LK optical flow algorithm with Gaussian pyramid model. The test results show that the algorithm enhances the accuracy and adaptability of TLD algorithm in tracking the target under fast motion in the long time tracking. After the blocked target is recovered, the target can be accurately repositioned. So the effect of this algorithm is obviously improved for single moving target.

References
[1] Xiaojun Jia, Qingcang Yu. The Image Processing Based on Open Source Computer Vision Library[J]. Computer Applications and Software, 2008, 25(4): 276 ~ 278
[2] Mei Fang, Qingcang Yu, Huaqiang Li. Digital Image Processing Based on OpenCV in C++ Builder[J]. Computer Engineering and Design, 2008, 29(4): 882 ~ 884
[3] Kalal Z, Mikolajczyk K, Matas J. Tracking-Learning-Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 34(7): 1409
[4] Wei Zhu, Chunguang Zhao, Leqing Ou, Shoufeng Wang. Examples of OpenCV Image Processing Program[M]. Beijing: Publishing House of Electronics Industry, 2016: 233 ~ 240
[5] Xiaoyang Wu. Moving Target Detection and Tracking Based on OpenCV[D]. Hangzhou: Zhejiang University, 2008:48 ~ 55
[6] Boneng Lai, Xiyuan Chen, Qinghua Li. Speed Calculation with Pyramid Lucas-Kanade Optical Flow[J]. Measurement and Control Technology, 2016, 35(6): 145 ~ 148
[7] Hongyan Huang, Siwen Gao, Yanmei Yu, Xiaohai He. Visual Obstacle Avoidance System Based on Optical Flow Method[J]. Journal of Image and Signal Processing, 2016: 5(2), 66-72
[8] Anurag Ranjan, Michael J Black. Optical Flow Estimation using a Spatial Pyramid Network[C]. 2017 IEEE Conference on Computer Vision and Pattern Recognition, 2017