Video Static Foreground Detection Method for Desktop Robotic Arm

Jianxin Wu, *Liyun Dai, Zhiming Liu
China College of Computer and Information Science, Southwest University, Chongqing 400715, China
wjx19960525@email.swu.edu.cn

Abstract. Although there are many efficient methods to detect the foreground from video frames, these methods are not completely applicable to the application scenario of desktop robotic arm. This paper presents a video static foreground detection method suitable for the desktop robotic arm application scenario. In our method, an auxiliary board is used to decrease false detection caused by camera-shake and work area movement, and to effectively eliminate non-target foreground interference such as human hands. In addition, with the auxiliary board we can also accurately calculate the 2-dimensional position and posture of the foreground. This method is general applicable to most desktop robotic arm application scenarios. Furthermore, compared with the traditional foreground detection method, this paper presents the 3-frames threshold method to solve the problem of multi-texture background environment appears many noise. What's more, for the problem that it is difficult to classify when the foreground colors are similar to the background colors, this paper presents the difference degree to describe pixels.

1. Introduction
The idea of background subtraction[1] is to build a background model with the first frame of the video or the previous frame, and then to compare the current frame with the background model, so as to classify the foreground and background. This method can be divided into two categories: (1) parametric techniques that use a parametric model for each pixel location[1][2], and(2)samples-based techniques that build their model by aggregating previously observed values for each pixel location[3][4]. These methods are widely used for object detection when the camera is fixed and the background is static, which are adapted for the application scenario of desktop robotic arm. Unfortunately, this idea is only used to detect the moving foreground. In the application scenario of desktop robotic arm, we need to detect the static foreground to obtain its static position and posture. In addition, we need to eliminate the false detection of non-target foreground, such as robotic arm and human hands in the process of interacting with the work area. Furthermore, we need to overcome the camera-shake and work area-movement caused by the movement of the robotic arm.

Among the static foreground detection methods, some methods directly compare the current frame with the background frame[5][6], but these methods can get all the moving and static foreground, and can't distinguish whether the foreground is moving. There are also some methods which use double foreground to calculate all the foreground and moving foreground respectively, and subtract the moving foreground from all the foreground to get the static foreground[7][8]. However, these methods are only effective for the continuously moving foreground, and moving objects that static for a short time or move slowly will be mistakenly detected. For example, in this application scenario, robotic arms and
human hands will be mistakenly detected when they static in the process of interacting with the work area.

In this paper, we add an auxiliary board to the work area of the robotic arm to skillfully solve the above problems (the specific installation method is given in Section 2), and presents a video static foreground detection method suitable for desktop robotic arm, which is divided into three steps. The first step is to extract and correct the area in the auxiliary board (the work area). The second step is foreground detection, and in this step, a background subtraction method based on samples is designed for the application scenario of desktop robotic arm. What is new in this method is that the auxiliary board is introduced to solve the problem of static foreground detection and eliminate the interference of non-target foreground. This method can work smoothly in various environments, such as light changes, the foreground colors are similar to background colors, and the camera-shake or work area-movement.

The rest of the paper is organized as follows. Section 2 introduces the installation method of the desktop robotic arm. Section 3 presents method in detail. The experimental results are given in section 4. Section 5 gives a summary and future work.

2. Installation of Desktop Robotic Arm

![Installation way of robotic arm](image)

Fig.1 Installation way of robotic arm

The eye to hand installation method of desktop robotic arm is: the camera and robotic arm are both fixed on the desktop, and the camera is directly above the work area of robotic arm and shoots downward. The specific installation method is shown in Fig.1(a).

In our method, we need to install a rectangular auxiliary board (cardboard is used in this paper) in the camera's field of vision, which should be placed in the camera's field of vision and occupy the camera's field of vision as much as possible, as shown in Fig.1(b). If the color of the auxiliary board is similar to the desktop, it is necessary to draw a obvious dividing line on the boundary edges of the auxiliary board, and no object interference can be placed on the board boundary. When it is inconvenient to install the auxiliary board, an obvious rectangular boundary mark can be made on the desktop instead of the auxiliary board. The auxiliary board can be directly regarded as the workbench of the desktop robotic arm. If there are other workbench, the workbench can be fixed on the auxiliary board at will, but the size of the auxiliary board should be slightly larger than that of the workbench, and the boundary of the workbench should not exceed that of the auxiliary board. We stipulate that the work area of the desktop robotic arm is the area within the auxiliary board.

3. Algorithm steps

3.1. Extraction and Perspective Transformation of Target Region

3.1.1. Extraction for Auxiliary Board Region

Although the camera is fixed, the background will change due to camera-shake and work area-movement, which will make the background subtraction method invalid. We have to find a way to fix the background, that is, extract the image only containing the auxiliary board region from the video frame, and the background subtraction processing is only for the image. In this way, not only the background can be fixed, but also the interference of non-work area can be eliminated to reduce the amount of calculation.
To extract the auxiliary board region, we must first extract the outer boundary contour of the auxiliary board, which is defined as $C_*$, as shown in Fig.2. We can take advantage of the obvious boundary between the auxiliary board region and the desktop. Firstly, we use Canny Edge Detection Operator to find the edges of video frames, and then find the contours of these edges and denote them as sets $\{C_1, C_2, \ldots, C_m\}$. Because the auxiliary board occupies the field of view of the camera as much as possible during installation, we only need to find the contour $C_{max}$ with the largest area in the set $\{C_1, C_2, \ldots, C_m\}$, and the contour $C_{max}$ has a great probability of being the contour $C_*$. However, because edge detection may lead to noise, sometimes the contour $C_*$ is unusable (not a clean and complete quadrilateral contour), and $C_*$ may be incomplete or have obvious noise interference, even $C_{max}$ may not be $C_*$. This will cause mistakes in our subsequent detection, so we need to do correctness detection. Because these mistakes have a small probability, we can just eliminate them, see Section 3.1.3 for details.

Next, we need to find the four vertices of the contour $C_*$. First do polygon fitting on the contour $C_*$ with Douglas-Peucker algorithm[9], and then do corner detection with Shi-Tomasi algorithm[10] to get the coordinates of the four vertices of the contour $C_*$, which are denoted as $(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)$, as shown in Fig.3(b). Finally, according to these four vertices, the pixels in the contour $C_*$ can be cut into a quadrilateral image from the video frame, which is denoted as $I_p$, which is the auxiliary board region, as shown in Fig.3(c).

3.1.2. Correction for Auxiliary Board Region

Although the auxiliary board is rectangular, due to the installation error of the camera, the optical center of the camera cannot be completely guaranteed to be perpendicular to the desktop, so the $I_p$ we obtained is an irregular quadrilateral image, as shown in Fig.4(a). This can cause distortion of the foreground, which is inconvenient for subsequent processing and calculating the 2-dimensional position and posture of the foreground. Therefore, it is necessary to transform the irregular quadrilateral image $I_p$ into a rectangular image $I_r$ through perspective transformation. In 3.1.1, we calculated the vertex coordinates $(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)$ of $I_p$, and let the four vertex coordinates of $I_r$ after perspective transformation be $(0,0), (w,0), (w,h), (0,h)$, among them:

$$h = \max (b_4 - b_1, b_3 - b_2)$$

$$w = \max (a_2 - a_1, a_3 - a_4)$$

Use $(a_1, b_1), (a_2, b_2), (a_3, b_3), (a_4, b_4)$ and $(0,0), (w,0), (w,h), (0,h)$ these four pairs of points can directly solve the linear equations to calculate the homography matrix $H$ of perspective transformation.
Then we can perspective transform $I_p$ into a rectangular image $I_r$ through $H$. In this process, we also need to use linear interpolation algorithm to fill the vacant part in the rectangular image.

Finally, we get a rectangular image which only contains the auxiliary board region, as shown in Figure 4 (b). After that, we only focus on the rectangular region, which can effectively reduce the impact of camera-shake and work area-movement. Moreover, on the premise of knowing the size of the auxiliary board, we can accurately and easily calculate the 2-dimensional position and posture of the foreground.

3.1.3. Correctness Check for Auxiliary Board Region

In chapter 3.1.1, we emphasized $C_*$ may be unusable, which will lead to $I_r$ is wrong, so we need to make a correct judgment for $C_*$. In fact, we can easily see $C_*$ with the naked eye whether it is usable (clean and complete quadrilateral contour), but it is impossible to judge whether each frame is available with the naked eye. So we take the first transformed image as the initial frame $I_{r, init}$ and confirm it is availability with naked eyes, then use $I_{r, init}$ to build the background model (see 3.2.1 for details), then use $I_{r, init}$ as a reference to check the image $I_{r, t}$ at the time $t$ whether it is available.

The method is as follows: firstly, we resize $I_{r, t}$ to the size of $I_{r, init}$, then respectively search ORB feature[11] points of $I_{r, init}$ and $I_{r, t}$ images , and their feature points are matched by FLANN[12] to obtain point pair set $\{(p_1, p'_1), (p_2, p'_2), \ldots, (p_q, p'_q)\}$. Because the background in this application scenario is static, and $I_{r, init}$ is the background frame, and the feature point pairs on matching are all static background points, so the coordinate values of these point pairs in pixels should be the same. If the pixel coordinate value of any point pairs are not equal, the obtained $I_{r, t}$ is considered to be wrong, so we should eliminate it.

$$\begin{cases} 
\text{retain } I_{r, t}, & \forall i \in \{1, 2, \ldots, q\} \text{ position}(p_i) = \text{position}(p'_i) \\
\text{eliminate } I_{r, t}, & \exists i \in \{1, 2, \ldots, q\} \text{ position}(p_i) \neq \text{position}(p'_i)
\end{cases}$$

3.1.4. Intrusion Detection for Auxiliary Board Region

In this method, we need to judge whether the foreground in the auxiliary board is static, and also need to eliminate the interference of non-target foreground. Based on the conventional idea, we need to detect all the static foreground first, and then judge whether it is a non-target foreground such as human hands or robotic arms. This requires the design of the corresponding algorithm to judge whether the foreground is a target foreground, which increases the complexity of the algorithm. However, in this application scenario, the foreground objects to be detected are placed in the work area by human hands or robotic arms, and the work area is in the auxiliary board. The process of human hands and robotic arms interacting with the work area is an intrusion process, and the intrusion will destroy the boundary contour $C_*$ of the rectangular auxiliary board(as shown in Fig. 5 (c)). And we default that the foreground will not move by themselves when hands or robotic arms does not interact with the work area. So we transform the two problems of whether the foreground is static and eliminating non-target foreground into intrusion detection problem, and intrusion detection can be transformed into the completeness detection of the $C_*$. Judge contour the completeness of $C_*$ is very simple, using the first frame of non-intrusion frame to calculate the length of contour $C_*$ (as shown in Fig.5(b)) and denote as $len_{init}$, if an intrusion occurs, its length will change(as shown in Fig.5(c)).
Therefore, we only need to judge whether the length \( l_{e1} \) of the current frame contour \( C_\star \) has a obvious change with \( len_{init} \), and then we can judge whether there is intrusion:

\[
\begin{align*}
&\text{There is an invasion at time } t, \quad |len_{init} - len_t| < \frac{len_{init}}{10} \\
&\text{There is no invasion at time } t, \quad |len_{init} - len_t| \geq \frac{len_{init}}{10}
\end{align*}
\]

We detect the foreground of \( I_{r,t} \) only when there is no intrusion, this solve the problems of static foreground and non-target foreground interference.

### 3.2. Strategy of Foreground Detection

#### 3.2.1. Background Modeling

Here, we build a background sample set for image \( I_{r,init} \). We denote \( p_t(x, y) \) as the point of image \( I_{r,t} \) at the time \( t \) coordinate \((x, y)\). In order to initialize the background sample set quickly, we use the VIBE algorithm[4] for reference, and randomly select \( N \) points in the 8 neighbourhood of point \( p_{init}(x, y) \) as the sample set of the point, which is denoted as \( \{s_1(x, y), s_2(x, y), ..., s_N(x, y)\} \).

We denote \( v_t(x, y) \) as the description value of point \( p_t(x, y) \). Which is different from the previous methods, in addition to \( r,g,b \) information, we add the difference degree \( d \), that is:

\[
v_t(x, y) = [r_t(x, y), g_t(x, y), b_t(x, y), d_t(x, y)]
\]

Among them:

\[
d_t(x, y) = |r_t(x, y) - b_t(x, y)| + |r_t(x, y) - g_t(x, y)| + |b_t(x, y) - g_t(x, y)|
\]

The addition of \( d \) makes it possible that the foreground colors are similar to background colors. \( d \) can enlarge the distance of RGB color space. When the \( rgb \) distance of two pixels is very close, \( d \) still has a large distance. At the same time, it can resist the change of brightness. When the brightness changes, the \( rgb \) value will increase or decrease at the same time, so \( d \) will not change. The Fig.6 can be a good illustration.

#### 3.2.2. Classification of Foreground and Background

This section introduces the method of background classification. When the background is fixed, according to the conventional idea, if a point is close to at least one sample point in its corresponding sample set, then we think that the point is a background point. For point \( p_t(x, y) \), we calculate the distance between it and \( N \) sample points in the corresponding background sample set (Euclidean distance of the description value), and then find the smallest distance and denote the distance as \( diff_t(x, y) \):

\[
\text{(a) Background video frame (b) Video frame (c) Distance when not add } d \text{ (d) Distance when add } d
\]

**Fig.6** Comparison results of add \( d \)
\[
diff_t(x, y) = \min_i (\text{distance}(v_t(x, y), s_i(x, y)))
\]

If \( \diff_t(x, y) \) is less than the dynamic threshold \( \dt_t \), the pixel is judged as the quasi foreground. The dynamic threshold \( \dt_t \) is described in 3.2.3.

\[
\begin{align*}
\text{sign}_t(x, y) &= 1, \quad \diff_t(x, y) \leq \dt_t(x, y) \\
\text{sign}_t(x, y) &= 0, \quad \diff_t(x, y) > \dt_t(x, y)
\end{align*}
\]

Considering the spatial correlation of pixels, the probability that the neighborhood of the foreground is also the foreground will be higher, so we use this feature to vote on the 8 neighborhood of each pixel. If more than half of the quasi foreground appears in the 8 neighborhood, we think that the point is the foreground, which can effectively filter the noise and holes.

\[
\begin{align*}
\text{sign}_t(x, y) &= 1, \quad \sum_{i=-1}^{1} \sum_{j=-1}^{1} \text{sign}_t(x + i, y + j) > 4 \\
\text{sign}_t(x, y) &= 0, \quad \sum_{i=-1}^{1} \sum_{j=-1}^{1} \text{sign}_t(x + i, y + j) \leq 4
\end{align*}
\]

### 3.2.3. Dynamic Threshold and Background Model Updating

In order to deal with the change of ambient light, it is necessary to update the background sample set regularly. Here we also refer to VIBE\[4\] update strategy. When updating the sample set, a random sample point in the corresponding sample set is replaced by a background point, so as to ensure the smooth life cycle of the sample set. At the same time, in order to avoid salt noise foreground is updated as the background, only the points detected as the background in three consecutive frames update their background samples.

If the threshold value is fixed, although it is simple, there will be a lot of texture noise in the application scenario with too many textures, as shown in Fig.7 (c). This is because our foreground classification method is based on pixels, some pixels will inevitably be misclassified, especially in the place where the pixel value gradient is larger, we call these points flicker points. So the dynamic threshold is very necessary. Considering the temporal correlation of the image, we presents the 3-frames threshold method. For point \( p_t(x, y) \), we take the maximum value of \( \diff_{t-1}(x, y) \), \( \diff_{t-2}(x, y) \) and \( \diff_{t-3}(x, y) \) plus the preset radius \( R \) as the threshold value:

\[
\begin{align*}
\dt_t(x, y) &= R, t = 1 \\
\dt_t(x, y) &= \diff_{t-1}(x, y) + R, t = 2 \\
\dt_t(x, y) &= \max \left( \diff_{t-1}(x, y), \diff_{t-2}(x, y) \right) + R, t = 3 \\
\dt_t(x, y) &= \max \left( \diff_{t-1}(x, y), \diff_{t-2}(x, y), \diff_{t-3}(x, y) \right) + R, t > 3
\end{align*}
\]

In this way, the characteristics of the recent image are reasonably used, and the interference of texture noise is effectively avoided. Consistent with the background update, only the points detected as the background in three consecutive frames update their dynamic thresholds. The foreground and background classification results after using the 3-frames threshold method are shown in Fig.7(d).
Because the junction of foreground and background is also a place with large pixel gradient, flickering points are easy to appear, and same flickering points will appear in the foreground. These flickering points will make the points in the foreground be updated as background samples, which will cause ghosts at the flickering points when the foreground is moved or removed. Based on this, we presents the reject update region strategy, in which all the pixels in the convex hull region of the foreground connected domain join the reject update region. In order to avoid texture noise being added to the refused update region, considering that texture noise has the characteristics of small area, the foreground whose connected domain is less than the preset threshold $T_{\text{A}}$ is not added to the reject update region. The value of preset threshold $T_{\text{A}}$ is set as 1/50 of the area of image $I_{r,t}$ in this paper.

### 3.3. Calculating the 2-dimensional Position and Posture of the Foreground

Based on the above foreground detection results, we can easily get the center pixel coordinates and 2-dimensional angles of each foreground. The specific method is as follows: Firstly, we find the outer contour point sets of each foreground connected domain. Then we find the convex hull of these point sets. Finally, we use the Rotating Calipers algorithm to find the minimum circumscribed rectangle of these convex hull. Let the center position of the minimum circumscribed rectangle be the center position of the foreground pixel, denoted as $(x_{\text{pixel}}, y_{\text{pixel}})$. The angle between the minimum circumscribed rectangle and the X axis of the image is regarded as the 2-dimensional posture of the foreground pixel, denoted as $\text{angle}_{\text{pixel}}$.

Next, we solve the physical 2-dimensional position and posture of each foreground. According to the principle of camera imaging, the relationship between the pixel coordinate $(x_{\text{pixel}}, y_{\text{pixel}})$ and its world physical 2-dimensional position coordinates $(X, Y, Z)$ is as follows:

$$
\begin{align*}
x_{\text{pixel}} &= \frac{\alpha f}{Z} X + c_x \\
y_{\text{pixel}} &= \frac{\beta f}{Z} Y + c_y
\end{align*}
$$

$\alpha$ and $\beta$ are the scale factors of the horizontal and vertical axes of the image, $f$ is the focal length of the camera, $c_x$ and $c_y$ are the compensation values caused by the non-coincidence between the optical center coordinates of the camera and the origin coordinates of the image, all of them are constants. The imaging region is a desktop, and desktop is a plane, then the $Z$ coordinate value can also be regarded as a constant, so the physical coordinates of the foreground has a linear mapping relationship with its pixel coordinates, and the pixel angle is the same as the physical angle. The physical size $(W_{\text{real}}, H_{\text{real}})$ of the auxiliary board is known, and its pixel size $(w, h)$ is also obtained in 3.1.2. According to the above reasoning, we can get the corresponding physical 2-dimensional position and posture of the foreground pixel with the upper left corner of the auxiliary board as the origin:
4. Experimental Results

In this section, in order to verify the effect of our method, we have carried out experiments on our method. In the experiment, we select the sample number N as 5, the preset radius R as 10. The code is written in Python using OpenCV library, and running on a computer with 3.3 GHz Intel i7 CPU and 16 GB RAM, which is Linux operating system. The installation method of the robotic arm, camera and auxiliary board is same with Fig.2, and the pixel size of the camera is 720 * 1280.

The experimental result gives bounding box of the foreground and foreground’s physical 2-dimensional position \((x_{\text{real}}, y_{\text{real}})\) and posture \(\text{angle}_{\text{real}}\). Moreover, 2-dimensional position and posture are marked in the upper right corner of the bounding box with \((x_{\text{real}}, y_{\text{real}})\), \(\text{angle}_{\text{real}}\) style format.

In order to verify that this method can deal with complex background, we use a multi-texture chessboard and a landscape image as the background, and in order to verify that this method is still effective in the case of camera-shake and work area-movement, after modeling the background, we move the auxiliary board and camera. The results are as follows:

\[
\begin{align*}
    x_{\text{real}} &= \frac{H_{\text{real}}}{h} \times x_{\text{pixel}} \\
    y_{\text{real}} &= \frac{W_{\text{real}}}{w} \times y_{\text{pixel}} \\
    \text{angle}_{\text{real}} &= \text{angle}_{\text{pixel}}
\end{align*}
\]

Conclusion: this method can accurately detect the foreground and give the 2-dimensional position
and posture in the multi-texture and complex background, and in the case of camera-shake and work area-movement (the video frame shows that the work area has obvious movement in the camera field of vision) this method still work well. In addition, after added the difference degree \( d \), we can detect the foreground that colors is very close to the background, which is difficult to classify even with the naked eye.

5. Conclusion
In this paper, a video static foreground detection method for desktop robotic arm is presented. With the help of auxiliary board, the background is fixed, which can effectively resist the camera-shake and work area-movement. Through the intrusion detection of auxiliary board, the interference of non-target foreground such as human hands on the detection results is eliminated. Finally, the 2-dimensional position and posture of the foreground can be accurately calculated by linear mapping between the corrected video frame and the rectangular auxiliary board. For the problem that multi-texture background environment has many noise, this paper presents the 3-frames threshold method. And for the problem that it is difficult to classify when the foreground colors are similar to background colors, this paper presents a fourth dimensional information difference degree in addition to RGB 3-dimensional information, which can effectively enlarge the slight differences between similar colors.

This method can do target detection directly in the application scenario that does not need to know the class of foreground. If we needs to know the category of foreground, this method can cooperate with Few-shot learning, and use the way of N-way K-shot[13][14] to classify. The foreground without background obtained which way can be directly used as the training set, the classification effect will be better.

This method slows down the work efficiency of the robotic arm, because the robotic arm needs to be moved out of the work area during the foreground detection. At the same time, due to the elimination of some wrong frames, the real-time performance of the detection will be affected. Our future work will address these issues.

Acknowledgments
This work was financially supported by the National Natural Science Foundation of China(62032019, 61732019, 61672435), and the Capacity Development Grant of Southwest University (SWU116007).

References
[1] C. Stauer, W. E. L. Grimson. Adaptive Background Mixture Models for Real-time Tracking. In Proceedings[C]. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999.
[2] T. Bouwmans, F. E. Baf, and B. Vachon. Background Modeling using Mixture of Gaussians for Foreground Detection A Survey[C]. Recent Patents on Computer Science, 2008.
[3] Z. Zivkovic, F. V. Heijden. Efficient Adaptive Density Estimation Per Image Pixel for the Task of Background Subtraction[J]. Pattern Recognition Letters, 2006.
[4] Barnich, M. V. Droogenbroeck. ViBE: A Powerful Random Technique to Estimate the Background in Video Sequences[C]. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.
[5] A. Bayona, J. C. San Miguel, and J. M. Martinez. Comparative Evaluation of Stationary Foreground Object Detection Algorithms Based on Background Subtraction Techniques[C]. 2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance, 2009.
[6] S. Guler, J. A. Silverstein, and I. H. Pushee. Stationary Objects in Multiple Object Tracking[C], 2007 IEEE Conference on Advanced Video and Signal Based Surveillance, 2007.
[7] L. Jinhui, J. Yaoliang, F. Guoliang. Real Time Automatic Obstacle Detection Method for Trae Surveillance in Urban Tra[J]. Journal of Signal Processing Systems, 2016.
[8] R. B. Waqqasur, S. Martin. Static Object Detection and Segmentation in Videos Based on Dual
Foregrounds Difference with Noise Filtering[J]. CoRR abs/2012, 2020.
[9] D. H. Douglas, T. K. Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature[J]. The Canadian Cartographer, 1973.
[10] J. Shi, C. Tomasi. Good Feature to Track[C]. Proc Of IEEE Conference on Computer Vision and Pattern Recognition, 1994.
[11] G. Bradski, K. Konolige, V. Rabaud and E. Rublee. ORB: An Efficient Alternative to SIFT or SURF[C]. 2011 IEEE International Conference on Computer Vision (ICCV 2011), 2011.
[12] M. Marius, G. L. David. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration[C]. International Conference on Computer Vision Theory and Applications, 2009.
[13] J. Snell, K. Swersky, and R. Zemel. Prototypical Networks for Few-Shot Learning[C]. Advances in Neural Information Processing Systems, 2017.
[14] F. Sung, Y. X. Yang, and L. Zhang. Learning to Compare: Relation Network for Few-Shot Learning[C]. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.