Integration of MediaPipe and SIFT Technologies for Accurate Exercise Posture Recognition

You-En Wang
Department of Electrical Engineering
National Central University
Taoyuan city 320, Taiwan

Ping-Hsuan Hsueh
Department of Electrical Engineering
National Central University
Taoyuan city 320, Taiwan

Chieh Yen
Department of Electrical Engineering
National Central University
Taoyuan city 320, Taiwan

***Abstract*—This article aims to identify and determine whether there are any changes in the posture of athletes during exercise. Incorrect exercise posture is the main cause of exercise-related injuries. We use Google MediaPipe to preprocess video images, extract feature points and feature descriptors using the SIFT algorithm on the processed images, and analyze the results to identify differences in posture during exercise. We did observe from the experimental results that there is a significant difference in similarity when the motion is abnormal. However, the accuracy of similarity recognition is less reliable when engaged in intense physical activity. We plan to further develop our research in the future to enable athletes to understand their upper limit of physical activity clearly. They can adjust their body parts appropriately during subsequent exercises by observing the quantified results of similarity measurement.**

***Keywords*—SIFT, MediaPipe, image recognition, pose recognition**

I. Introduction

This project focuses on applying the Scale-Invariant Feature Transform (SIFT) algorithm in image pose recognition. SIFT is a machine vision algorithm that is used to detect and describe local features in images. It first establishes the scale space of the image and finds the image extremum. Then, it obtains the feature description vector based on the extremum and finally identifies the result based on similarity matching using Euclidean distance.

SIFT has high robustness in terms of photometric and geometric transformations. In this project, Google MediaPipe is used to assist SIFT in recognizing whether each pose of the test subject during the same movement is similar. Correct movement posture can reduce abnormal joint wear and tear. Through our research, athletes can use the results to adjust and correct their actions to minimize the potential damage caused by incorrect posture during exercise.

II. Principle of Algorithm

A. MediaPipe Pose

Google MediaPipe Pose is an application within the MediaPipe framework that is primarily used for human pose estimation and tracking. It detects and tracks the human body pose from video streams captured by a camera and provides real-time information such as keypoint locations and pose angles.

MediaPipe Pose uses deep neural network technology and a dual-branch network architecture based on ResNet. The network's main objective is to predict the pose keypoints of the human body, including various body parts such as the head, shoulders, hands, elbows, knees, and more.

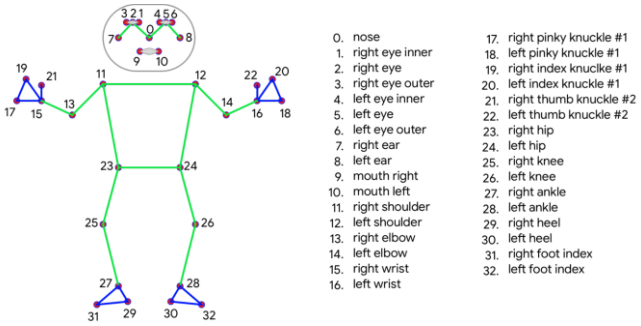

Fig.1 Landmarks Map of Google MediaPipe Pose [1].

The workflow of MediaPipe Pose mainly consists of three steps: pose detection, keypoint tracking, and angle calculation, which are as follows:

1) Pose detection: For each frame of the captured image from the camera, MediaPipe Pose first processes the image, converting it into a 224×224 sized image. Then, the image is passed through the neural network model, which returns the position of 33 keypoints.

2) Keypoint tracking: MediaPipe Pose uses a Kalman filter to track the keypoints after performing pose detection. The Kalman filter is a linear state estimator used to smooth and correct the position of keypoints, improving the tracking accuracy.

3) Angle calculation: Once keypoint tracking is completed, MediaPipe Pose calculates the pose angles of the human body by computing the distances and angles between adjacent keypoints. These angles can be used to estimate the human body's pose and applied to various scenarios such as sports analysis, health monitoring, gesture recognition, etc.

B. Scale-Invariant Feature Transform (SIFT)

1) Scale-Space Extrema Detection

The SIFT algorithm utilizes a Gaussian difference pyramid to detect local extrema in images. This pyramid is constructed by applying Gaussian filters and down-sampling the original image to create a series of images at different blur levels. The Difference of Gaussians (DoG) is computed between adjacent layers of the Gaussian pyramid, resulting in a DoG pyramid. This process allows for detecting local extrema in the image at various scales, making SIFT capable of detecting features of different sizes and orientations.

The following formula can be used to calculate Gaussian blurred images, where $I(x,y)$ represents the pixel value of the original image at (x,y), and the scale factor σ determines the degree of image smoothing.

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}*I(x,y) \quad (1)$$

The DoG image obtained by differentiating the Gaussian blurred image can be calculated using the following formula:

$$D(x,y,\sigma)=\big(G(x,y,k\sigma)-G(x,y,\sigma)\big)*I(x,y) \quad (2)$$

$G(x,y,\sigma)$ is the Gaussian function, k is the scale factor between adjacent layers in the pyramid, and σ refers to the level of blurring applied to an image. The larger σ is, the smoother the image is, and the corresponding scale is larger.

Figure 2 illustrates an effective method for constructing $D(x, y, \sigma)$. The initial image undergoes a gradual Gaussian convolution process to generate images spaced by a constant factor of k in scale space. In our research project, we will set the number of inner layers in each octave to 5 for the convenience of experimentation.

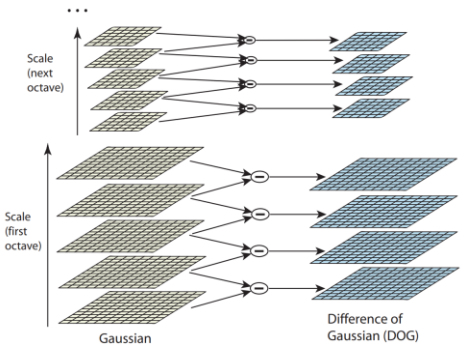

Fig.2 The set of scale space images is generated by performing iterative convolutions of the initial image with Gaussians for each octave in the scale space. The difference-of-Gaussian images are obtained by subtracting adjacent Gaussian images in the set.

After processing a full octave, we resample the Gaussian image with twice the initial σ value (i.e., the third image from the top of the stack) by selecting an even pixel in each row and column. This technique reduces computation while maintaining sampling accuracy relative to σ, similar to the beginning of the previous octave.

Next, we detect local extrema on each level of the DoG pyramid. To find local extrema, we first locate the maximum or minimum within a 3×3×3 neighborhood of each pixel in each DoG image. Then, we determine whether the candidate point is a local extrema by comparing its value to the values of all 26 neighboring pixels within the $3\times3\times3$ neighborhood. If the candidate point is a local extrema, it is labeled as a keypoint.

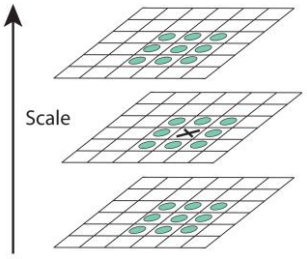

Fig.3 Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with circles).

2) Accurate Keypoint Localization

Once a candidate keypoint has been found by comparing pixels and their neighborhoods, the next step is to perform detailed fitting of nearby data, including position, scale, and principal curvature ratios. Brown developed a method (Brown and Lowe, 2002) [3] that uses a Taylor expansion (quadratic term) of the scale-space function $D(x, y, \sigma)$ with the origin set at the sample point.

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \quad (3)$$

In this equation, D and its derivatives are evaluated at the sampling point, $x = (x, y, \sigma)^T$ is the offset from this point. Differentiating both sides of the equation and setting it to 0 yields the extremum point. The position of the extremum point $\hat{x}$ can be determined by taking the derivative of the function concerning x and assigning it to zero.

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \left(\frac{\partial D}{\partial x}\right) \quad (4)$$

If the offset $\hat{x}$ is greater than 0.5 in any dimension, it means that the extremum is closer to a different sample point. In this case, the sample point is changed and interpolated. The final offset $\hat{x}$ is added to the position of its sample point to obtain the interpolated estimate of the extremum position.

The value of the extremum point's function $D(\hat{x})$ helps reject unstable extrema with low contrast. This result can be obtained by substituting equation (4) into (3).

$$D(\hat{x}) = D(\vec{0}) + \frac{\partial D^T}{\partial x} \hat{x} + \frac{1}{2} \hat{x}^T \frac{\partial^2 D^T}{\partial x^2} \hat{x} = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x} \quad (5)$$

However, these points have very low response values and are easily affected by noise interference, so use a corner detector for detection and removal. The Harris corner detector searches for areas with significant grayscale changes in an image, which may represent edges or corners in the image. It then distinguishes between corners and edges by analyzing the local grayscale changes in these areas.

Specifically, the Harris corner detector calculates a matrix that describes the local features of each pixel by computing the grayscale changes in the surrounding area. It then uses the matrix's eigenvalues and eigenvectors to determine whether the pixel is a corner point. First, compute the M matrix.

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \quad (6)$$

where $w(x, y)$ is a weighting factor, typically a Gaussian function. Then, compute the corner response value.

$$R = det(M) - k(trace(M))^2 \quad (7)$$

where $det(M)$ is the determinant of the M matrix, $trace(M)$ is the trace of the M matrix, and k is a constant parameter. If R is greater than a threshold and greater than the corner response values of its neighboring pixels, then mark the pixel as a corner. In this research, extreme points with contrast threshold $|D(\hat{x})|$ less than 0.03, which is consistent with Lowe's paper [2], were discarded (assuming that the image pixel values are within the range [0,1]).

In image processing, the DoG filter is sensitive to image edges and can produce extreme points at the image's edges. However, these points are unstable and need to be removed due to the difficulty in locating edge points and their susceptibility to noise. To address this, the algorithm searches for points with higher intensity in the neighborhood of each key point. If the key point is the local maximum among these points, it is retained; otherwise, it is removed. In this study, an edge threshold of 10 was used, which is consistent with Lowe's paper [2].

3) Orientation assignment

In order to achieve rotation invariance, the SIFT algorithm assigns a main orientation to each keypoint by calculating the direction of the image gradient. To compute the gradient direction, we first need to calculate the gradient magnitude and direction of the image. The gradient can be calculated using the following formula.

$$m(x, y) = \sqrt{(L(x+1, y)-L(x-1, y))^2 + (L(x, y+1)-L(x, y-1))^2} \quad (8)$$

$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \quad (9)$$

After calculating the gradient of the Gaussian image in the neighborhood of the feature point, a histogram is used to count the gradient direction and magnitude of the pixels in the neighborhood. The horizontal axis of the histogram represents the gradient direction angle, and the vertical axis represents the accumulated gradient magnitude corresponding to the gradient direction angle. The histogram divides the 0-360 degrees range into 36 bins, each bin representing a 10-degree range. The peak of the histogram represents the main direction of the image gradient in the neighborhood of the feature point, which is the main orientation of the feature point. If a peak has 80% energy of the main peak, it is considered as the secondary direction of the feature point, which can enhance the robustness of matching.

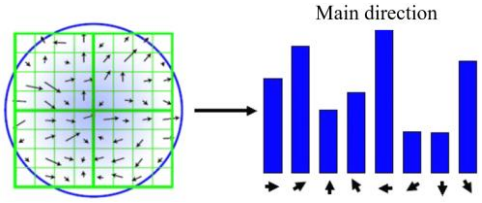

Fig.4 Orientation Histogram.

When computing the histogram, each sampled point added to the histogram is weighted using a circular Gaussian function, which performs Gaussian smoothing. Lowe suggested using a Gaussian weighted calculation with $\sigma = 0.5d$ for the gradient magnitude of the pixels in the subregion. This reason is that the SIFT algorithm only considers scale and rotation invariance, not affine invariance. By Gaussian smoothing, the gradient magnitudes near the keypoint have greater weight, partially compensating for the instability of keypoints caused by the lack of consideration of affine invariance.

After obtaining the main orientation of the image keypoint, each keypoint has three pieces of information *(x, y, σ, θ)*: position, scale, and orientation. From this, we can determine a SIFT feature region. Typically, a circle with an arrow or simply an arrow is used to represent the three values of the SIFT region: (a) the center represents the feature point position; (b) the radius represents the keypoint scale ($r = 2.5\sigma$); (c) the arrow represents the main orientation.

4) Keypoint descriptor

The SIFT descriptor is a feature vector composed of local images around keypoints. To compute the descriptor, we first divide the image around the keypoint into 16 sub-regions, each of size 4×4 and containing 16 pixels. Then, we use the same method as computing the gradient orientation histogram to calculate the gradient orientation histogram for each sub-region.

When computing the gradient orientation histogram, we divide the range of 360 degrees into 8 regions. We assign each pixel's gradient orientation to the closest directional region. Then, we assign the gradient orientation of each sub-region to the corresponding directional region, resulting in a gradient orientation histogram containing 8 elements. Because each sub-region has a gradient orientation histogram with 8 elements, there are a total of 128 elements for the 16 sub-regions around each keypoint These elements are combined into a 128-dimensional vector, which serves as the keypoint descriptor.

To achieve scale invariance, we also need to normalize the descriptor. We treat the descriptor as a vector and divide it by the Euclidean norm of the vector to obtain a unit vector. This result eliminates the influence of scale, making the descriptor scale-invariant.

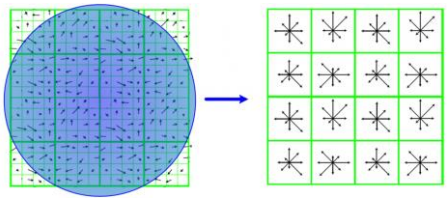

Fig.5 A total of 4×4×8=128 data, forming a 128-dimensional SIFT feature vector.

C. Matching descriptors

1) K-Nearest Neighbors

KNN is one of the simplest methods to classify objects from the closest training examples in feature extraction. KNN classification is based on measuring the distance between different eigenvalues. If a sample belongs to the majority class of its *k* most similar (i.e., closest) samples in the eigenspace, then it is also classified into that class. Here, *k* is usually an integer no greater than 20. In general, a bigger *k* value can reduce the noise effect on the classification, but it makes boundaries between classes less different [4]. KNN uses Euclidean distance as a distance-measuring instrument. The Euclidean distance formula is as follows, where *x* and *y* are Euclidean vectors.

$$d(x,y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \qquad (10)$$

2) FLANN

FLANN is an abbreviation for "Fast Library for Approximate Nearest Neighbors". The FLANN feature matcher is optimized for fast nearest-neighbor searches in huge datasets. This also works for features with high dimensions. It works faster than other matching techniques like Brute-Force Matcher for large datasets when fused with SIFT descriptor. The use of FLANN for search can be generally divided into two steps: indexing and searching. The first dictionary is the "indexParams", which configures the algorithm we want to use [5].

The best-known indexing method for high-dimensional similarity search is Locality-Sensitive Hashing (LSH) [6]. The basic idea of LSH is to use hash functions to transform and partition the original dataset into multiple subsets. The data in each subset are adjacent, and the number of elements in each subset is relatively small. By

doing so, the problem of searching for adjacent elements in a large dataset is transformed into the problem of searching for adjacent elements in a small subset. This use obviously reduces the computational cost.

D. RANSAC

RANSAC, which stands for Random Sample Consensus, is an algorithm used to calculate a model that fits a subset of data randomly selected from a larger dataset. In RANSAC, the "ransacReproThreshold" (abbreviated as "T_R") is a crucial parameter that is used to determine whether a data point is considered an inlier of the model. T_R is a threshold that is used to calculate the distance between a data point and its corresponding projected point on the model. If this distance is less than T_R, the data point is considered an inlier of the model; otherwise, it is considered an outlier. Because the subset of data is chosen randomly, RANSAC is a non-deterministic algorithm. However, after multiple iterations, according to probability, the model established has a certain probability of fitting most or all of the data, which becomes the optimal solution, i.e., the best model that represents the data.

Assuming the probability that each point belongs to the true inlier group is ω, then ω = number of true inlier points / total number of data points. Usually, we do not know what ω is. ω^n represents the probability that n randomly selected points are all inliers, and $(1-\omega^n)$ represents the probability that at least one of the n selected points is not an inlier. $(1-\omega^n)^k$ represents the probability that after k iterations, none of the n randomly selected points are all inliers. If the probability of success after k iterations is p, then the probability of being an outlier is $(1-p)$.

$$1-p = (1-\omega^n)^k \tag{11}$$

$$k = \frac{log(1-p)}{log(1-\omega^n)} \tag{12}$$

Therefore, if we want a high probability of success, say p = 0.99, and we keep n constant, increasing k leads to higher p. On the other hand, if we keep ω constant, increasing n requires a larger k. Since ω is usually unknown, it is better to choose a smaller n [7].

III. RESEARCH METHOD

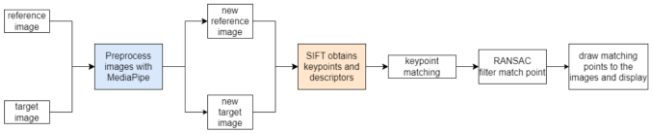

Fig.7 The flow chart of main program operation.

The study tested 5 different exercise movements, focusing on the common movements of planks and squats. The goal was to evaluate whether there were significant differences in exercise movements during prolonged activity that could potentially lead to exercise-related injuries. In order to achieve this, pre-recorded exercise videos were edited into two to three equal-length segments, with the first segment used as the baseline and the other segments used as target videos for comparison and analysis.

The main program flow chart for the study is shown in Figure 7. The video images were pre-processed using Google MediaPipe, and the processed images were then used to extract feature points and feature descriptors using the SIFT algorithm. Finally, appropriate matching points were selected, and the results were analyzed to identify differences in the subjects' postures during exercise movements. The advantage of this three-stage process is that it focuses the target feature points on body posture, reducing interference from unrelated information and improving the efficiency and accuracy of feature point extraction.

A. Pre-processing images with MediaPipe

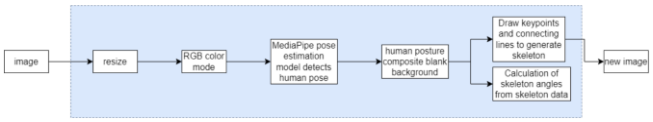

Fig.8 The flow chart of Preprocess images with MediaPipe [8].

In order to better recognize the keypoints of human posture in the later stages, we employed a two-stage preprocessing method for the images.

1) Resizing and changing color space of the images

We resize input images to 700×500 pixels for uniformity and faster program execution. Balancing the number of SIFT feature points detected and program speed is critical. Since there are no recording device specifications, resizing guarantees uniformity. Next, we convert the image to the RGB color space to ensure compatibility with Google MediaPipe, which helps to ensure accuracy and stability. This procedure represents completing the first stage of pre-processing.

Fig.9 The original image of squats and planks.

Fig.10 The image of squats and planks using RGB color space.

2) Extracting human posture to synthesize a new image

This study focuses on SIFT feature point extraction on human body appearance and simultaneously recognizes human body posture similarity. Google MediaPipe, trained with accurate machine learning, is used to extract appearance from the original image. The posture estimation model in the MediaPipe database is first used to detect 33 keypoints that are different from SIFT keypoints, necessary to build the skeleton. Then, a pre-established 700 × 500 pixels background is added to the detected human posture to remove noise and interference. Finally, the skeleton is established by connecting the 33 keypoints to the new image, which facilitates SIFT feature point extraction.

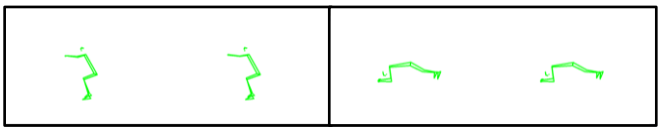

Fig.11 The body skeleton of squats and planks.

Fig.12 The new image of squats and planks.

B. SIFT obtains keypoints and descriptors

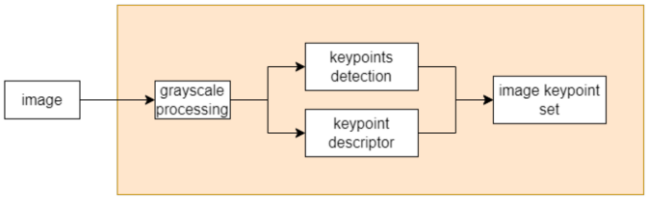

Fig.13 The flow chart of SIFT process [9], [10].

For the new reference and target images, we have undergone two-stage preprocessing, keypoint extraction and descriptor calculation are performed using the SIFT algorithm. First, the two images are converted to grayscale color space acceptable to the SIFT algorithm. Then, SIFT feature point extraction and descriptor calculation are performed to facilitate subsequent descriptor matching.

Fig.14 The new images of squats and planks using grayscale color space.

C. Keypoint matching and filtering

During the feature point matching stage, we used the Flann-based Matcher instead of the common Brute-force Matcher, as the Flann-based Matcher provides a more precise matching of feature points. In addition, we used the KNN-matching algorithm to perform the ratio test to achieve more accurate keypoint matching. Then the RANSAC algorithm was used to calculate a mathematical model to classify the matched keypoints.

D. Similarity Calculation

This study aims to examine whether the subject's posture is correct. Therefore, we used the number of feature point matches in each frame to judge similarity.

$$\text{match ratio} = \frac{\text{total number of accurately match points}}{\text{total number of match points}} \quad (13)$$

The total number of accurately matched points refers to the precise number of keypoints matching that have passed the ratio Test and RANSAC filtering. The match ratio obtained from this formula represents the matching rate of keypoints between the new reference image and the target image for each frame. It is worth noting that, under the premise that the motion of the new reference image is the standard motion, the high or low numerical value of the match ratio at a certain moment is not our concern. We only need to pay attention to the time when the similarity significantly drops.

$$\text{Similarity} = \text{match ratio} \times 100\% \quad (14)$$

In the above formula, "Similarity" refers to the similarity of each frame of the video. By displaying the Similarity on the window, the changes in the Similarity of each frame can be observed as the video is played. Assuming a certain value as a danger threshold, when the similarity falls below this threshold, the movement posture is likely to cause sports injury. The danger threshold can be set by the programmer (in this study, it was set as 15%).

Fig.15 The presentation of program results for squats and planks images.

The program results are shown in Figure 15. In this study, we plotted the similarity over time on a chart (Figure 17), which allows for objective quantification of the test subject's exercise performance. Suppose the similarity drops below a danger threshold, the color space of the target image turns red, and non-matching keypoints in the target image are marked to observe significant differences in the subject's movements.

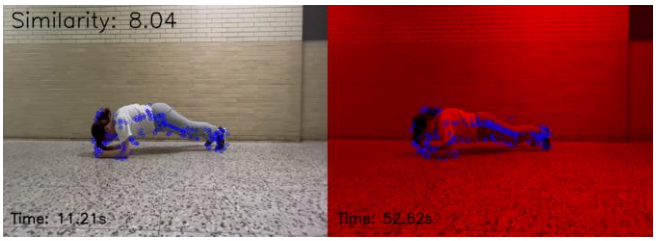

Fig.16 The decrease in similarity below the danger threshold was caused by the significant drop in the subject's hip height as shown in the image.

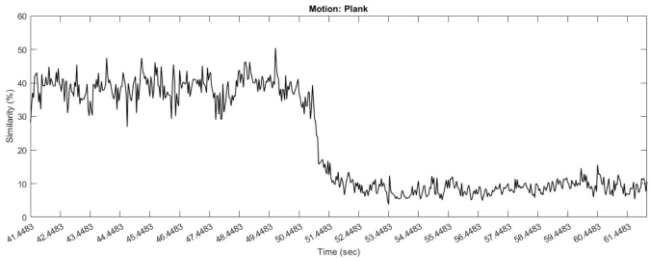

Fig.17 The similarity of plank exercises over time.

IV. Experimental Results

During the research process, adjusting different parameters may affect the recognition results. Therefore, in order to find the appropriate parameter settings, we designed two sets of experiments.

A. Varying the blur level of SIFT images

When operating SIFT, the σ value can affect the image blur and the number of extracted key points. Therefore, selecting an appropriate σ value for SIFT feature extraction is necessary. In the first experiment, we varied the σ value and observed changes in similarity to select a suitable σ value for similarity recognition of human posture.

Table 1. The relationship between the maximum number of keypoints and the σ value.

video \ σ	1.6	2.6	3.6	4.6	5.6	6.6
reference	369	178	92	68	56	47
target	363	176	90	64	53	50

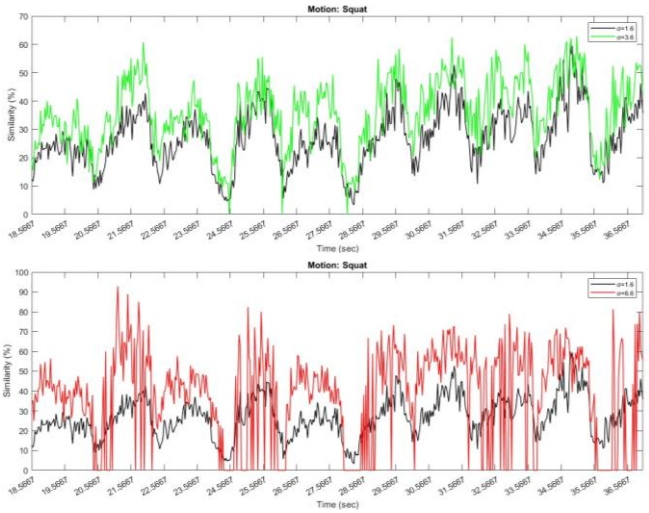

Fig.18 A higher σ value is more sensitive to changes in similarity (above: σ = 1.6, 3.6; below: σ = 1.6, 6.6).

Observing Table 1, we can see that the number of keypoints decreases as the σ value increases. When the number of keypoints is low, even a slight difference in the subject's movement may result in an insufficient number of matched keypoints, leading to a low or even zero similarity score. Figure 18 shows that a higher σ value is more sensitive to changes in similarity and has lower accuracy, making it less suitable for correcting the identification of movement posture. Considering the number of keypoints and the sensitivity to movement posture errors, we choose σ = 1.6 for SIFT feature extraction.

B. Varying the number of inliers in RANSAC

The number of inliers in RANSAC will automatically adjust to the optimal size with each frame played, which affects the number of matched descriptors and indirectly affects similarity. In the second experiment, we varied the RANSAC iteration count and T_R to observe the changes in the standard deviation of the similarity list.

Table 2 shows that as the iteration count and T_R value increase, the standard deviation of similarity also increases. By computing a greater number of iterations, the probability of a reasonable model being produced is increased. Considering the tolerance for errors in posture, we selected the iteration count and threshold (i.e., max iterations = 6000, T_R = 5.0) with the median standard deviation

(15.013) for similarity recognition of human posture.

Table 2. The relationship between the standard deviation of similarity and the RANSAC iteration times and T_R.

Iter \ Threshold	3.0	4.0	5.0	6.0	7.0
2000	14.052	14.468	14.989	15.283	15.367
4000	13.896	14.497	15.022	15.187	15.410
6000	13.944	14.591	15.013	15.218	15.450
8000	13.950	14.562	14.916	15.154	15.405
10000	13.932	14.633	15.023	15.243	15.432

V. Conclusion

A. Discussion

The research has achieved our desired goals, as evidenced by the changes in exercise posture similarity. This research has two key points for correct recognition of exercise posture accuracy. Firstly, the change in similarity should not be overly sensitive. Secondly, an appropriate value for the standard deviation of the similarity set needs to be selected. Through the two experiments mentioned above, we found that when the parameters are set to σ = 1.6, max iterations = 6000, and T_R = 5.0, these two requirements can be met. Furthermore, according to the experimental results, we found that this study performs better in evaluating static exercise postures (as shown in Figure 17).

Combining MediaPipe Pose and SIFT can achieve more precise and reliable pose estimation and image matching. Thus, it can be widely applied in many fields, such as human motion analysis, posture correction, virtual try-on, etc. However, there are still some limitations in this research. Since SIFT requires higher computing resources and complex algorithms; therefore, the speed is slower.

RANSAC can be used to estimate the parameters of a linear model even when there is a lot of noise in the data. It does not require feature scaling, which makes it easier to use for datasets with different feature scales. However, RANSAC may not always find the optimal set, and it is vulnerable to outliers when the data does not exhibit a clear linear relationship.

There are strict limitations on the input images in this research. Firstly, the target image must have the same motion posture as the reference image simultaneously and synchronously. Secondly, the duration of the video in the target image must be the same as that in the reference image. Due to these two limitations, this study is unsuitable for real-time image input.

B. Future Perspectives and Outlook

1) Accelerating computational operations: Due to the low computational efficiency of SIFT, future research can explore optimizations using algorithms such as PCA-SIFT, GPU-SIFT, BRIEF, and ORB to improve the computational efficiency of SIFT.

2) Expandability: In the future, MediaPipe Pose can accurately display the angles of body parts, allowing athletes to know how to correct their postures.

These are potential prospects for future research, aiming to improve the efficiency and real-time performance of the SIFT algorithm, broaden its application scope in this study, and promote its application in broader fields.

VI. Reference

[1] "On-device, Real-time Body Pose Tracking with MediaPipe BlazePose", https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html

[2] D.G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91-110 2004.

[3] M. Brown and D. G. Lowe, "Invariant features from interest point groups," *In British Machine Vision Conference*, Cardiff, Wales, pp. 656-665, 2002.

[4] D. I. H. Putri, Martin, Riyanto, and C. Machbub, "Object Detection and Tracking Using SIFT-KNN Classifier and Yaw-Pitch Servo Motor Control on Humanoid Robot," *2018 International Conference on Signals and Systems (ICSigSys)*, 01-03 May, 2018.

[5] V. Vijayan and P. Kp, "FLANN Based Matching with SIFT Descriptors for Drowsy Features Extraction," *2019 Fifth International Conference on Image Information Processing (ICIIP)*, 15-17 Nov., 2019.

[6] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimens ionality," *In Proc. of the 30th ACM Symposium on Theory of Computing*, pp. 604-613, 1998.

[7] O. Chum, "Two-View Geometry Estimation by Random Sample and Consensus," Ph.D. Thesis, 2005.

[8] "Holistic landmarks detection task guide," https://developers.google.com/mediapipe/solutions/vision/holistic_landmarker

[9] "cv:SIFT Class Reference," https://docs.opencv.org/4.x/d7/d60/classcv_1_1SIFT.html

[10] "Drawing Function of Keypoints and Matches," https://docs.opencv.org/4.x/d4/d5d/group__features2d__draw.html#ga5d2bafe8c1c45289bc3403a40fb88920