Live Face Verification with Multiple Instantialized Local Homographic Parameterization

Chen Lin[1], **Zhouyingcheng Liao**[1], **Peng Zhou**[1], **Jianguo Hu**[2] and **Bingbing Ni**[1*]
[1]Shanghai Jiao Tong University [†]
[2]Minivision
linchen040329@sjtu.edu.cn, patrickliao2007@gmail.com, zhoupengcv@sjtu.edu.cn,
hujianguo@minivision.cn, nibingbing@sjtu.edu.cn

Abstract

State-of-the-art live face verification methods would be easily attacked by recorded facial expression sequence. This work directly addresses this issue via proposing a patch-wise motion parameterization based verification network infrastructure. This method directly explores the underlying subtle motion difference between the facial movements re-captured from a planer screen (e.g., a pad) and those from a real face; therefore, interactive facial expression is no longer required. Furthermore, inspired by the fact that, we embed our network into a multiple instance learning framework, which further enhance the recall rate of the proposed technique. Extensive experimental results on several face benchmarks well demonstrate the superior performance of our method.

1 Introduction

Live face verification has been a very important building block for modern face recognition applications such as online banking, online shopping etc. The purpose of live face verification is to recognize whether the face under examination is being captured from a real person or some types of , e.g., showing the recorded face image/video on a pad to the camera. Previous methods [Pan *et al.*, 2007] mainly adopt an strategy to distinguish a real face from a fake one. Namely, the recognition systems ask the user to present different facial expressions such as smile, shaking head, close/open eye and then perform emotion recognition to test whether the predicted emotion matches the given order. However, such a strategy has inherent drawbacks. First, completing the entire verification process usually requires several minutes, which is NOT user friendly. Second, such a system could be very easily attacked by recording down all possible emotions of the user. To this end, recent works [Tan *et al.*, 2010; Yang *et al.*, 2013; Kollreider *et al.*, 2005] have started exploring a solution.

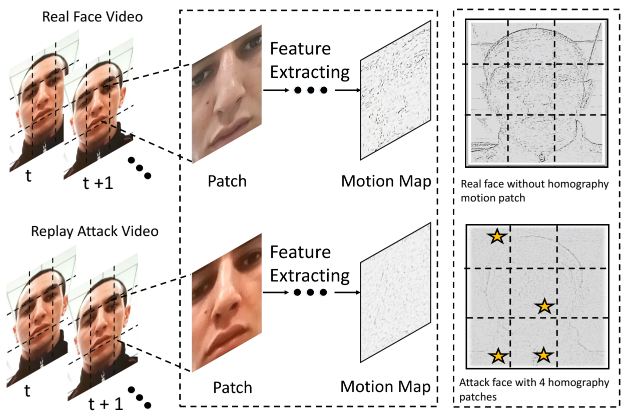

Figure 1: Visualization of our motivations. Each video is a bag containing 9 patches. Each patch is an instance, from which a motion map is extracted. The homography motion maps are labeled with yellow stars. With the motion maps, we can infer that the video in the first row is a real face video and the video in the second row is an attack face video.

Namely, users are allowed to perform any facial movements in a natural way, and the algorithm extract static motion features based on a single frame or several consecutive frames, to directly perform verification. No emotion actions are required.

Although more user friendly, it is very challenging for this method to achieve very low miss detection rate as well as false alarm rate, due to the following reasons. Most of these methods extract various holistic image features (e.g., such as Local Binary Patterns [Määttä *et al.*, 2011], 3D face shape analysis [Lagorio *et al.*, 2013], Fourier spectra [Li *et al.*, 2004]) or motion features between frames (e.g., such as optical flow [Bao *et al.*, 2009] motion magnification [Bharadwaj *et al.*, 2013] etc.) and then perform ensemble learning for improving the verification performance on a certain dataset. This approach is harmful for at least two reasons. First, as no physical geometric prior knowledge is used for exploring informative static/motion features, the learned features

[*]corresponding author: Bingbing Ni.

[†]Shanghai Institute for Advanced Communication and Data Science, Shanghai Key Laboratory of Digital Media Processing and Transmission, Shanghai Jiao Tong University, Shanghai 200240, China.

might not necessary correspond to the underlying between a real face and a fake one, i.e., the learned features might overfit. Second, these methods usually treat the entire image as a whole, however, to recognize faces discriminatively, local subtle features are more helpful. For example, [Sun *et al.*, 2014] extract features from multiple patches to do face identification. Using global feature representations might degrade the sensitivity of these algorithms. This work proposes a patch-wise motion parameterization based verification network to explicitly address these issues. This network is motivated by two observations. First, we note that although for the entire face, there exist NO explicit (specific) motion patterns when the user is performing certain emotions, there DO exist some local motion patterns when the faces are viewed from a planer objects (e.g., pad). In a short time-span, local facial patches could be approximately considered as a rigid planer object, therefore local homography holds. This motivates us to construct a network which extracts local motion maps (shown in Figure 1) from local facial patches (in our cases, a STN [Jaderberg *et al.*, 2015] network component), therefore to distinguish a real face from a fake one by attending to local motion patterns. Furthermore, inspired by the fact that, we embed our network into a multiple instance learning framework. In this sense, local subtle facial movements is more reasonable for our model, and therefore further enhance the recall rate of the proposed technique. Extensive experimental results on several face benchmarks well demonstrate the superior performance of our method.

The rest of this paper is organized as follows. We first review the previous work in three fields: the live face verification, the homography transformation, and the multiple instance learning. Then, we design a method to do the face verification task. Finally, we perform extensive experiments on several benchmarks with in-depth analysis.

2 Related Work

In live face verification , many motion features based methods [Li *et al.*, 2004; Bharadwaj *et al.*, 2013; Bao *et al.*, 2009] try to detect the live face clues. However, these live clues could also be easily detected from a replaying planar screen. On the contrary, we focus on finding ”fake” clues by detecting the homography holding regions. If we find any ”fake” face clues in a video, we think it is an attack video.

Two consecutive frames are related by homography if the captured object is a rigid body [Szeliski, 2004]. For a whole frame, global homography doesn’t hold when there contains moving objects while the background stays static. However, we can respectively explore homography for each local patch. Early homography detecting methods, such as [Hinterstoisser *et al.*, 2008], usually first detect the matching pixels between two consecutive frames. In our case, the matching process will lead to computing cost. To this end, we utilize the Spatial Transform Network (STN) [Jaderberg *et al.*, 2015]. It meets our conditions for the following reasons: 1) to estimate the homography, the first thing is to settle down the corresponding homography matrix, which is determined by six parameters. The six parameters allow rotation, translation and scaling. The STN which trains six warp parameters, is usually applied to do affine transformation; 2) the STN embedded to Convolutional Neural Network (CNN) don’t need the process of finding the matching points between two frames.

multiple instance learning (MIL) has improved the performances of many tasks in computer vision including segmentation [Yang *et al.*, 2017], tracking [Babenko *et al.*, 2009] and posing detection [Babenko *et al.*, 2008], since [Dietterich *et al.*, 1997]. MIL collects all instances into a bag, and one instance is not singleton. A bag is positive if there exists at least one positive instance; otherwise it is negative. In our work, we collect several patches from a whole frame, if we find any homography patches, we can regard the video as an attacking video and this naturally meets the MIL conditions.

3 Method

3.1 Motivation and Overview

As introduced above, we are motivated by the following two observations:

- O1: Local facial patches from the recorded video present significant motion pattern, e.g., local homographic property between adjacent frames, while real ones do not possess this nature.
- O2: For a fake face, there exist multiple patches which are easily recognized as “fake”, while all local patches of a real one must be recognized as “real”.

Following these two motivations shown in Figure 1, our proposed framework is detailed as follows. The input is two adjacent frames containing face instances. After face detection for each frame, each face is resized to the fixed size and then divided equally into 9 local patches. So we obtain nine pair-wise local facial patches from two adjacent frames. All pair-wise patches are input into a *Local Motion Discovery Module* which explores the underlying motion characteristic and outputs some local motion parameterization maps. Then, each of these local motion parameterization maps concatenated with the corresponding pair-wise local facial patch is input into a classification sub-network to evaluate whether the pair-wise patch is from a real video or not. To take full advantage of O2, we consider each face as a bag of local patches and propose a multiple instance based novel loss function to improve the recall rate of attack video. Our proposed network is illustrated in Figure 2.

3.2 Prerequisite

Live face verification can be defined as a two classification problem, that is, given a video containing a face, to determine whether the video is a real video or an attack video. Let $\{(V_i^{tr}, y_i)\}_{i=1}^N$ denotes the training set, where V_i^{tr} represents a training face video, y_i is the related ground truth, and N is the number of videos in the training set. $y_i \in \{0, 1\}$, 0 represents real video and 1 represents attack video. First, the videos are disassembled into frames, i.e., $V_i = \{F_{i,j}\}_{j=1}^{M_i}$, where $F_{i,j}$ denotes the j-th frame in video V_i, M_i is the number of frames in V_i. Then, we get the bounding boxes of face regions in the whole frames by applying the Dlib’s face detection algorithm. We further crop the face regions and

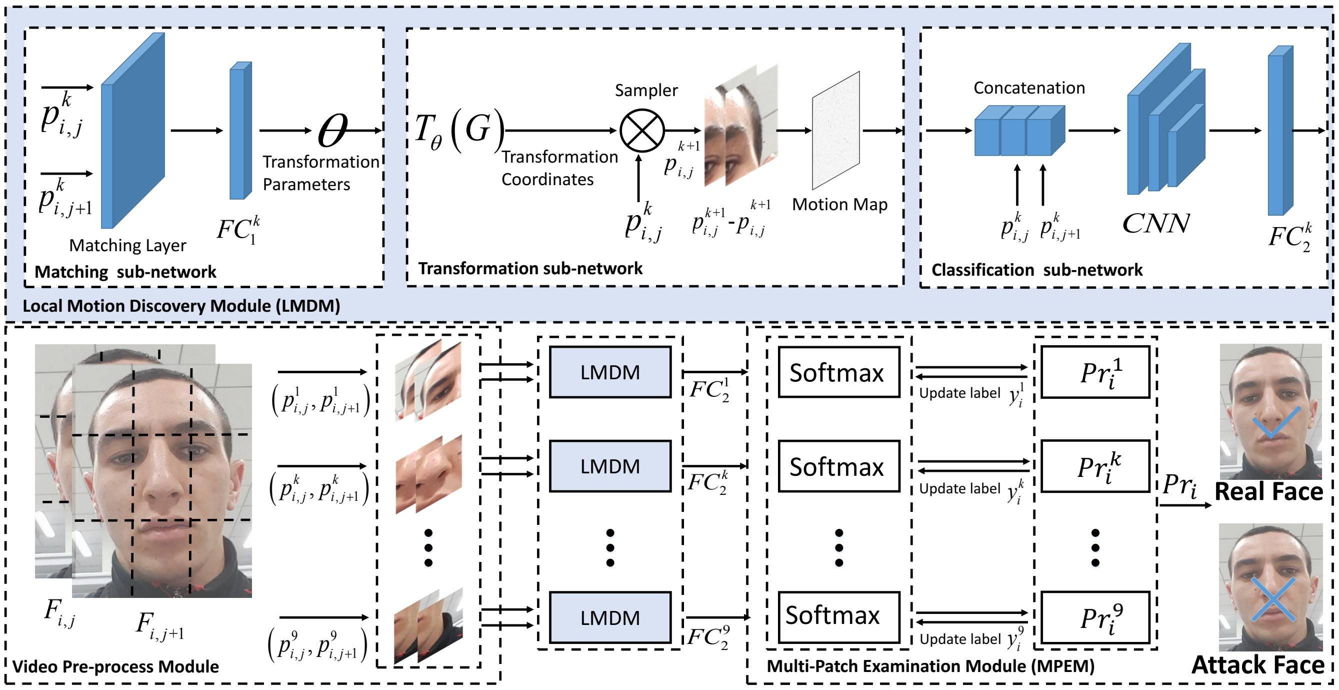

Figure 2: **Overview of the proposed approach**. For live face verification, we first apply the Video Pre-process Module to the original face video and we get 9 patch-pairs. We then separately extract motion pattern for each patch-pair in the Local Motion Discovery Module (LMDM). In LMDM, it takes the patch-pair and outputs the transformation parameters θ between the two patches. With θ and the patch-pair we obtain a patch level Motion Map. After that concatenation of the Motion Map and the patch-pair is fed to a CNN network. Finally, we regard the collection of the 9 patch-pairs as a bag and each patch-pair as an instance in the Multi-Patch Examination Module (MPEM). The MPEM outputs the final probability of the original face video.

resize them to 192 × 192. In order to get local subtle features of the faces, we divide the face into 9 average patches, $F_{i,j} = \{p^1_{i,j}, \ldots, p^k_{i,j}, \ldots, p^9_{i,j}\}$ shown in Figure 1.

3.3 Local Motion Discovery Module

We use the planar homography to parameterize local motion patterns. A planar homography is a transformation from one rigid plane to another, which is very useful to estimate the image transformation. Images of points on a plane in one view are related to corresponding image points in another view by a planar homography.

A planar homography can be represented as a 3×3 matrix. The coordinate transformation is

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \tag{1}$$

where (x_1, y_1) and (x_2, y_2) are the corresponding coordinates for a rigid planar object in two different views.

In the live face verification application, the frames can be regarded as different views of a face, due to the relative motion between the camera and the captured face. For a real face video, because the captured face is not a rigid plane, the corresponding coordinates in different frames do not satisfy the transformation Eq.(1), i.e., the frames are not related by homography. While for a re-captured face video, the re-playing screen or the attacking photo is a rigid plane, so all the frames are related by homography. But when the user is performing certain emotions, there exists no explicit global homography patterns. An example is shown in Figure 3. However, in a short time span, some facial patches could be approximately considered as a rigid planar object, there do exist some local homography patterns for these patches. Thus the local homography character is a useful feature in the live face verification task.

In order to extract local motion parametrization features from local facial patches, we develop a local motion discovery module similar to the Spatial Transform Network (STN) [Jaderberg *et al.*, 2015], which is shown in Figure 2. The local motion discovery module consists of a corresponding points matching sub-network, a transformation sub-network and a classification sub-network. Different from the original STN which takes one image as input, we input two patches $p^k_{i,j}$ and $p^k_{i,j+1}$, which are the k-th patches from consecutive j-th and j+1-th frames, to this module.

- The matching sub-network is a regular CNN. Its input is the two patches $p^k_{i,j}$ and $p^k_{i,j+1}$ and it outputs a homography matrix.

- The transformation sub-network has no trainable parameters. It only applies the homography matrix to patch $p^k_{i,j}$, and obtains a prediction patch $\widehat{p}^k_{i,j+1}$. The output of this sub-network is the difference between the pre-

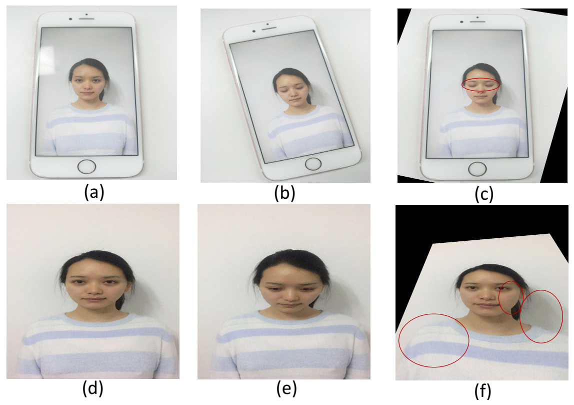

Figure 3: In the first row, (a) and (b) are the frames at T1 and T2 from a replaying video, (c) is the homography transformation from (b) to (a). In the second row, (d) and (e) are the frames at T1 and T2 from a live face video . (f) is the homography transformation from (e) to (d). The regions in the red circles are obviously deformed after transformation in (c) and (f). For the replaying video, we can apply a homography matrix to transform frame (b) to frame (a) approximately; while for the live face video, we can't make such transformation.

dicted patch $\widehat{p}^k_{i,j+1}$ and the patch $p^k_{i,j+1}$,

$$X^k_{i,j} = \widehat{p}^k_{i,j+1} - p^k_{i,j+1} \tag{2}$$

where $X^k_{i,j}$ represents the local motion parametrization features of the k-th local patch.

- The classification sub-network consists of a series of convolution layers, pooling layers and ends with a fully connected layer following a softmax layer. The input of the classification sub-network is a concatenation of $X_{i,j}$, $p^k_{i,j}$ and $p^k_{i,j+1}$. It outputs two probabilities indicating whether the two patches are from a real face video or an attack video. The classification sub-network is supervised by the ground truth y_i for video V^{tr}_i. The gradient is back propagated to both the weights of the classificat0ion sub-network and the matching sub-network.

It is worth noting that our matching sub-network is never explicitly learned from the two local patches. Instead, it learns automatically the homography transformation parameters that directly enhances the final classification accuracy.

3.4 Multi-Patch Examination Module

In the local motion discovery module, each face video is divided into 9 pair-wise local facial patches. We can regard each video

$$V_i = \left\{ \left(p^1_{i,j}, p^1_{i,j+1}\right), \ldots, \left(p^k_{i,j}, p^k_{i,j+1}\right), \ldots, \left(p^9_{i,j}, p^9_{i,j+1}\right) \right\}$$

as a bag, where $\left(p^k_{i,j}, p^k_{i,j+1}\right)$ represents an instance, and $p^k_{i,j}$ is the k-th patch of j-th frame in video V_i. Note that all instances(local patches) in a real face bag must be recognized as "real", therefore all the instances labels are available. But for a fake face, not all instances are recognized as "fake", because the homography motion patterns don't exist in some patches. The labels of instances in fake face bags are unknown for we

Algorithm 1 Learning algorithm for multi-patch examination module

```
Input: Training bags $B_{tr} = \{(V^{tr}_i, y_i)\}^N_{i=1}$ , where $V^{tr}_i$
       represents a training video and $y_i \in \{0, 1\}$
Output: Multiple instance model
 1: Initialize instance labels $\{y^1_i, \cdots, y^k_i, \cdots, y^9_i\}$ with the
    value of $y_i$
 2: Train local motion discovery module with cross-entropy
    loss shown by Eq. 3 for one epoch and get a pre-trained
    model;
 3: repeat
 4:     Input one batch data in the pre-trained model and out-
        put the probabilities $\{Pr^1_i, \cdots, Pr^k_i, \cdots, Pr^9_i\}$ for
        every instance of $V^{tr}_i$;
 5:     for each sample in the batch do
 6:         if $y_i = 1$ then
 7:             $y^k_i \leftarrow 1(Pr^k_i > 0.5)$ for every instance
                label in $\{y^1_i, \cdots, y^k_i, \cdots, y^9_i\}$;
 8:         else
 9:             $\{y^1_i, \cdots, y^k_i, \cdots, y^9_i\} \leftarrow y_i$;
10:         end if
11:     end for
12:     Use $\{y^1_i, \cdots, y^k_i, \cdots, y^9_i\}$ as ground truth and train
        local motion discovery module
13:     if iterating over an epoch then
14:         Update the pre-trained model with current model.
15:     end if
16: until max iteration is reached.
17: Output the final multiple instance model.
```

don't know which patches are "fake" in the bags. To find out the specific "fake" patches, we embed our work into a multiple instances framework.

Multiple Instances Learning (MIL) deals with the task in which the bag labels are available, whereas instance labels are unknown. A bag is positive if it contains at least one positive instance, and negative otherwise. The positive instances are also called the key instances. We propose a multiple instance based novel loss function which is embedded into the local motion discovery module to obtain an end-to-end multiple instance learning framework.

Algorithm 1 shows the details of learning process. The learner is given a training data set with N bags $B_{tr} = \{(V^{tr}_1, y_1), \ldots, (V^{tr}_i, y_i), \ldots, (V^{tr}_N, y_N)\}$, where $V^{tr}_i = \left\{ \left(p^1_{i,j}, p^1_{i,j+1}\right), \ldots, \left(p^k_{i,j}, p^k_{i,j+1}\right), \ldots, \left(p^9_{i,j}, p^9_{i,j+1}\right) \right\}$ is a bag with $\left(p^k_{i,j}, p^k_{i,j+1}\right)$ representing an instance and $p^k_{i,j}$ is the k-th patch of j-th frame in video V^{tr}_i. $y_i \in \{0, 1\}$ is the label of V^{tr}_i, where 1 represents the bag is a "fake" face video and 0 represents the bag is a "real" face video. If there exists any positive instance in bag V^{tr}_i, then V^{tr}_i is a positive bag ("fake" face video) with $y_i = 1$; otherwise it is a negative bag ("real" face video) with $y_i = 0$. Our goal is to find the positive instances, that is, the homography motion pattern patches. For the classification sub-network in the local motion discovery module, the cross-entropy loss for one batch data can be expressed as

$$\mathcal{L} = -\frac{1}{B \times P} \sum_{i}^{B} \sum_{k}^{P} (y_i^k log Pr_i^k + (1 - y_i^k) log(1 - Pr_i^k)) \tag{3}$$

where B represents batch size; P is the number of patches, which is equal to 9 in our case. y_i^k is the instance label. Pr_i^k is the probability of V_i^{tr} as a "fake" video predicted by the classification sub-network. Our multiple instance based loss function adjusts the labels y_i^k of the instances.

In the first epoch, we initialize the instance labels $\{y_i^1, \cdots, y_i^k, \cdots, y_i^9\}$ with the value of y_i and update parameters of the local motion discovery module to get a pre-trained model. In the second epoch, we first input batch data into the pre-trained model and get the corresponding probabilities $\{Pr_i^1, \cdots, Pr_i^k, \cdots, Pr_i^9\}$ for each instance in V_i^{tr}. Pr_i^k is the positive probability of the k-th instance in V_i^{tr}. When V_i^{tr} is a "fake" video, i.e., $y_i = 1$, let y_i^k be 1 if Pr_i^k is greater than 0.5, otherwise let y_i^k be 0. When V_i^{tr} is a "real" video, let y_i^k be equal to y_i directly. We think this is more in line with the phenomenon that for a "fake" video, not all patches are easily recognized as "fake", while all local patches of a real one must be recognized as "real". Once we get the ground truth for each instance, we update the model parameters to get a new pre-training model. We continue the training process until the maximum number of iterations is reached.

In testing phase, for each video we will get 9 probabilities that indicate whether the video is "fake" or not. We pick the biggest instance probability as the final score of the video as a "fake" video. Experimental results show that our multiple instance learning framework can effectively improve the live face verification accuracy.

3.5 Implementation Details

In our experiments, we use Stochastic Gradient Descent (SGD) to optimize our model. 8 videos are randomly chosen and adjacent two frames are sampled from each video to form a batch. The ResNet in our model is initialized from a pre-trained model for ImageNet [Deng *et al.*, 2009] classification. We start training with a learning rate of 0.005 and a weight decay of 0.001, and decrease the learning rate by 1/10 when the loss goes steady. All training and testing codes are built on Pytorch [Paszke *et al.*, 2017]

4 Experiments

Live face verification must be robust across different types of attacks. We evaluate our proposed approach on the OULU-NPU database [Boulkenafet *et al.*, 2017] and the Replay Attack database [Chingovska *et al.*,].

4.1 Databases

The OULU-NPU database contains 990 real face videos and 3960 attack face videos. And it is divided into training, development and testing subsets. It claims that the models should be learned on the training subset, fine-tuned on the development subset, and tested on the testing subset. Further, it designs 4 protocols to evaluate the generalization of the live face verification methods by controlling the illumination, background scenes and the sensor devices of videos. For instance, in Protocol IV, models should be evaluated across unseen illumination, background conditions and sensor devices in the training and fine tuning phases. The performance metrics are the Attack Presentation Classification Error Rate (APCER) and the Bona Fide Presentation Classification Error Rate (BPCER).

The Replay Attack database contains 200 real face videos and 1000 attack videos. It is divided into training, development and testing subsets. And it follows the same training and evaluating rules as the OULU-NPU database. The performance metric is the Half Total Error Rate(HTER).

4.2 Results and Analysis

Here, we evaluate our model on the OULU-NPU and the Replay Attack database. Results are listed in Table 1. It shows that our model yields 1.8% HTER on the Replay-attack database and 4.6% ACER in Protocol I, 5.4% ACER in Protocol II, 4.0 % ACER in Protocol III, and 12% ACER in Protocol IV on the OULU-NPU database, which outperforms the previous method *i.e.* LBP [Boulkenafet *et al.*, 2015].

To evaluate the effectiveness of each part of our model, we train a simple two-class classification network without motion map and multiple instance learning as our baseline. It yields 5.1% HTER on the Replay-attack database and 18.0% ACER on Protocol IV, OULU-NPU.

To demonstrate the effectiveness of motion map, we embed an STN network, which produce motion map from two adjacent frames, into the classification network. As we can see, CNN+STN, which yields 5.6% HTER on the Replay-attack database and 18.3% ACER on Protocol IV, OULU-NPU, performs much better than the baseline model(CNN). Moreover, we train the classification network with multiple instance learning. This CNN+MIL model yields 4.5% HTER on the Replay-attack database and 18.8% ACER on Protocol IV, OULU-NPU. Full results are listed in Table 1.

4.3 Generalization Analysis

We conduct a cross-database evaluation experiment to test the generalization capability of our proposed algorithm. All methods are evaluated on one database, while trained and fine-tuned on the other database.

It can be observed from table 2 that in terms of the ACER, our model gets the best generalization capability.

4.4 Hyper Parameters Analysis

We design an experiment to demonstrate why we divide face frames into 9 patches. In this setting, face frames are divided into average 1, 4, 9, 16 patches and tested on both data sets.

As is shown in Figure 4, the ACER (%) on both databases decreases as the number of patches increases. However, when the number of patches goes greater than 9, the ACER (%) does not increase obviously. When the patch numbers are set to 9, the ACER (%) reach 12% and 1.8% on the OULU-NPU database and the Replay-Attack database, respectively. Therefore, we set the number of patches to 9, which is a good trade-off between effectiveness and efficiency.

Methods	Replay-Attack		OULU-NPU					
	DEV	Test	DEV	Test				
				Display	Print	Overall		
	EER	HTER	EER	APCER	APCER	APCER	BPCER	ACER
			Protocol I					
LBP	0.4	2.9	4.4	5.0	1.3	5.0	20.8	12.9
CNN(baseline)	2.1	8.1	6.7	8.8	13.1	7.8	22.3	10.1
CNN +STN	0.4	5.6	1.4	6.3	9.6	9.6	0.8	5.2
CNN + MIL	3.5	4.5	4.4	3.3	2.1	3.3	9.2	6.3
CNN + STN+ MIL	0.4	**1.8**	2.2	7.5	8.3	8.3	0.8	**4.6**
			Protocol II					
LBP	-	-	4.1	15.6	22.5	22.5	6.7	14.6
CNN(baseline)	-	-	5.8	8.1	7.6	7.6	2.6	8.1
CNN +STN	-	-	3.3	8.6	6.1	8.6	7.5	8.1
CNN + MIL	-	-	5.6	12.0	9.2	12.0	7.2	9.6
CNN + STN+ MIL	-	-	2.4	5.6	1.1	5.6	5.3	**5.4**
			Protocol III					
LBP	-	-	3.9± 0.7	9.3±4.3	11.8±10.8	14.2±9.2	8.6±5.9	11.4±4.6
CNN(baseline)	-	-	4.9± 0.8	1.0±4.7	12.9±6.7	18.3±6.7	8.4±5.9	13.4±4.3
CNN +STN	-	-	2.5±0.7	1.3±1.6	2.2±3.4	2.4±3.3	11.4±10.3	6.9±4.0
CNN + MIL	-	-	3.8±1.5	1.5±1.3	1.8±1.4	2.5±1.4	9.4±10.1	6.0±4.5
CNN + STN+ MIL	-	-	2.4±1.1	1.0±1.0	1.0±1.3	1.5±1.2	6.4±6.6	**4.0±2.9**
			Protocol IV					
LBP	-	-	4.7± 0.6	19.2±17.4	22.5±38.3	29.2±37.5	23.3±13.3	26.3±16.9
CNN(baseline)	-	-	7.1±2.15	13.4±8.9	18.7±23.4	19.3±26.4	26.6±4.6	23.0±6.9
CNN +STN	-	-	4.7±2.5	11.7±10.3	16.7±18.9	17.5±18.4	19.2±19.0	18.3±6.8
CNN + MIL	-	-	4.8±3.5	14.2±17.2	17.5±12.9	21.7±13.3	15.8±15.3	18.8±7.0
CNN + STN+ MIL	-	-	3.4±1.4	10.8±12.8	13.3±14.4	15.8±12.8	8.3±15.7	**12.0±6.2**

Table 1: Experiment results on the Replay-Attack and the OULU-NPU databases. Here, CNN + STN refers to the model with motion map and CNN + MIL refers to the model with multiple instance learning.

Method	Replay-Attack		OULU-NPU	
	DEV	Test	DEV	Test
	EER	ACER	EER	ACER
LBP	0.3	15.7	4.7	13.9
CNN(baseline)	2.1	22.1	7.1	17.4
CNN+STN+MIL	0.4	15.1	3.4	11.1

Table 2: Inter-Test results in terms of EER(%) and ACER(%) on the Replay-Attack and the OULU-NPU databases.

	T	APCER	BPCER
LBP	0.04	24	36
Our model	0.18	8	20

Table 3: The average time T(s) taken in execution a video and the performance in terms of APCER(%) and BPCER(%) for both methods.

4.5 Time Complexity Analysis

To estimate whether the proposed method can response quickly in practical application scenarios, we test the execution time of the LBP and our model, which are trained on the OULU-NPU database. Our set up consists of a intel i5-5350 CPU, a 8GB Memory and a Logitech HD C920 Pro webcam. The camera captures 50 real face videos and 50 replay attack videos to do live face verification. Table 3 shows, in terms of the execution time, our method is comparable to LBP. Our model shows superior APCER and BPCER performance in practical application scenarios.

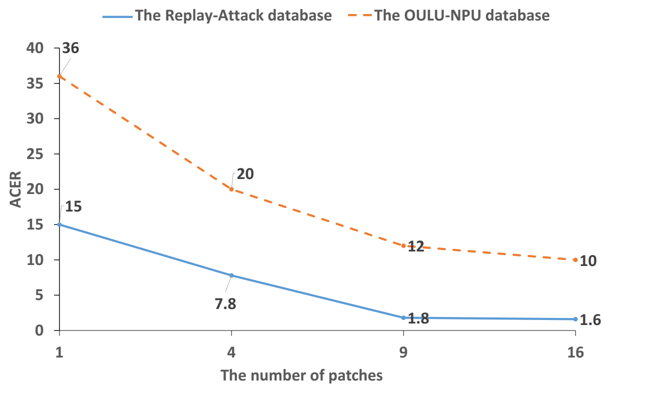

Figure 4: The results of ACER performance by dividing face frames into different number of patches. The yellow dot line and the blue line are the ACER performances on the OULU-NPU database and the Replay-Attack database, respectively.

5 Conclusion

In this paper, we present an accurate and robust method to tackle the live face verification task. The benefits of our proposed method are two-fold: 1) the Local Motion Discovery Module contributes to the verification accuracy, and 2) the Multi-Patch Examination Module enhances the recall rate of the attack videos. Extensive experimental results demonstrate the effectiveness of our proposed method.

Acknowledgments

This work was supported by National Science Foundation of China (U1611461,61502301,61521062). The work was partially supported by China's Thousand Youth Talents Plan, STCSM 17511105401 and 18DZ2270700.

References

[Andrews and Hofmann, 2003] Stuart Andrews and Thomas Hofmann. Multiple instance learning via disjunctive programming boosting. In *NIPS*, pages 65–72. 2003.

[Babenko *et al.*, 2008] Boris Babenko, Piotr Dollár, Zhuowen Tu, and Serge J. Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. 2008.

[Babenko *et al.*, 2009] Boris Babenko, Ming-Hsuan Yang, and Serge J. Belongie. Visual tracking with online multiple instance learning. *CVPR*, pages 983–990, 2009.

[Bao *et al.*, 2009] Wei Bao, Hong Li, Nan Li, and Wei Jiang. A liveness detection method for face recognition based on optical flow field. In *Image Analysis and Signal Processing*, pages 233–236, April 2009.

[Bharadwaj *et al.*, 2013] Samarth Bharadwaj, Tejas I. Dhamecha, Mayank Vatsa, and Richa Singh. Computationally efficient face spoofing detection with motion magnification. In *CVPR Workshops*, 2013.

[Boulkenafet *et al.*,] Z. Boulkenafet, J. Komulainen, Z. Akhtar, A. Benlamoudi, SE. Bekhouche, F. Dornaika, A. Ouafi, Amir Mohammadi, Sushil Bhattacharjee, and Sébastien Marcel. A competition on generalized software-based face presentation attack detection in mobile scenarios. In *IJCB, 2017*.

[Boulkenafet *et al.*, 2015] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid. face anti-spoofing based on color texture analysis. *CoRR*, 2015.

[Boulkenafet *et al.*, 2017] Z. Boulkenafet, J. Komulainen, Lei. Li, X. Feng, and A. Hadid. OULU-NPU: A mobile face presentation attack database with real-world variations. May 2017.

[Chen *et al.*, 2006] Yixin Chen, Jinbo Bi, and J. Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *PAMI*, 28(12):1931–1947, Dec 2006.

[Chingovska *et al.*,] Ivana Chingovska, André Anjos, and Sébastien Marcel. On the effectiveness of local binary patterns in face anti-spoofing.

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[Dietterich *et al.*, 1997] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1):31 – 71, 1997.

[Hinterstoisser *et al.*, 2008] Stefan Hinterstoisser, Selim Benhimane, Vincent Lepetit, Pascal Fua, and Nassir Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *BMVC*, 2008.

[Jaderberg *et al.*, 2015] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[Kollreider *et al.*, 2005] K. Kollreider, H. Fronthaler, and J. Bigun. Evaluating liveness by face images and the structure tensor. In *Fourth IEEE Workshop on Automatic Identification Advanced Technologies*, pages 75–80, Oct 2005.

[Lagorio *et al.*, 2013] A. Lagorio, M. Tistarelli, M. Cadoni, C. Fookes, and S. Sridharan. Liveness detection based on 3d face shape analysis. In *2013 International Workshop on Biometrics and Forensics (IWBF)*, pages 1–4, April 2013.

[Li *et al.*, 2004] Jiangwei Li, Yunhong Wang, Tieniu Tan, and A. K. Jain. Live face detection based on the analysis of fourier spectra. In *In Biometric Technology for Human Identification*, pages 296–303, 2004.

[Luong and Faugeras, 1996] Quan-Tuan Luong and Olivier D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *IJCV*, pages 43–75, 1996.

[Määttä *et al.*, 2011] J. Määttä, A. Hadid, and M. Pietikäinen. Face spoofing detection from single images using micro-texture analysis. In *IJCB*, pages 1–7, 2011.

[Pan *et al.*, 2007] G. Pan, L. Sun, Z. Wu, and S. Lao. Eyeblink-based anti-spoofing in face recognition from a generic webcamera. In *ICCV*, pages 1–8, Oct 2007.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[Sun *et al.*, 2014] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 classes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[Szeliski, 2004] Rick Szeliski. Image alignment and stitching: A tutorial. Technical report, October 2004.

[Tan *et al.*, 2010] Xiaoyang Tan, Yi Li, Jun Liu, and Lin Jiang. Face liveness detection from a single image with sparse low rank bilinear discriminative model. In *ECCV*, pages 504–517, 2010.

[Trucco and Verri, 1998] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[Yang *et al.*, 2013] J. Yang, Z. Lei, S. Liao, and S. Z. Li. Face liveness detection with component dependent descriptor. In *2013 ICB*, pages 1–6, June 2013.

[Yang *et al.*, 2017] Rui Yang, Bingbing Ni, Chao Ma, Yi Xu, and Xiaokang Yang. Video segmentation via multiple granularity analysis. *CVPR*, pages 6383–6392, 2017.

[Zhang and Zhou, 2017] Ya-Lin Zhang and Zhi-Hua Zhou. Multi-instance learning with key instance shift. In *IJCAI*, pages 3441–3447, 2017.