天津大学

《媒体计算》课程设计报告

视频特征表达与动作识别

姓　　名　郭清妍

学　　号　3018216068

2021 年 6 月 11 日

目录

摘要 3

Abstract 4

一、引言 5

二、视频特征表达方法 5

2.1 时空兴趣点 6

2.2 深度视频特征 8

三、视频动作识别方法 9

3.1 方法与步骤 10

3.2 结果及分析 19

四、总结 24

参考文献 25

摘要

随着视频获取设备和网络的发展，基于视频的人体动作识别获得了广泛的关注，从视频信息中分析和理解人体动作变得越来越重要。人体动作识别应用于视频监控、自动视频标签和人机交互等多个领域。

在本次课程设计中，我的任务如下：在数据集 KTH、HMDB51、DTDB 上使用 STIP、MoSIFT 特征提取方式，用 BoW 的方法对提取的特征首先使用 k-means 聚类算法聚类，然后对特征利用直方图进行量化，然后使用基于线性核函数、卡方核函数、RBF 核函数的 SVM 分类器，对视频进行分类。使用 Accuracy、Precision、Recall 三种指标来评估视频分类效果，最后对每个数据集中每个类别的第一段视频进行特征点可视化，在视频帧上可视化 STIP、MoSIFT 提取的兴趣点，然后合为一段视频，其中 MoSIFT 的可视化中，我还实现了 **MoSIFT 的方向变化情况**。

除此之外，我**对 BoW 量化之后的特征进行归一化处理，效果有所提升**。另外，分别从 SVM 核函数的选择、MoSIFT 特征选取方式、BoW 聚类类心个数以及是否对直方图特征进行归一化等几个方面来进行实验得出结论。

关键词：动作识别 视频特征提取 视觉单词模型 支持向量机

Abstract

With the development of video acquisition equipment and networks, video-based human action recognition has received widespread attention, and it has become more and more important to analyze and understand human actions from video information. Human action recognition is used in many fields such as video surveillance, automatic video tagging, and human-computer interaction.

In this course design, my tasks are as follows: use STIP and MoSIFT feature extraction methods on the data sets KTH, HMDB51, DTDB, and use the Bag of Visual Words(BoW) method to cluster the extracted features using the k-means clustering algorithm, and then the features are quantified using histograms. Then I use SVM classifiers based on linear kernel functions, chi-square kernel functions, and RBF kernel functions to classify the video. As for validation, I used Accuracy, Precision, and Recall to evaluate the performances of different methods of video classification. Finally I visualize the the points of interest extracted by STIP and MoSIFT on the video frame, of the first video, of each category, in each data set, and then combine them to a video.

In addition, I proposed an improvement in the qualification phase. I normalized the features after BoW quantization, and the accuracy has been improved apparently. In addition, experiments are conducted from several aspects: the selection of SVM kernel function, the selection method of MoSIFT features, the number of BoW cluster clusters, and whether to normalize the histogram features to draw conclusions, etc.

Key words: Human-Motion Recognition, Video Features Extraction, Bag of Visual Words, Support Vector Machine

一、引言

随着视频获取设备和网络的发展，基于视频的人体动作识别获得了广泛的关注，从视频信息中分析和理解人体动作变得越来越重要。人体动作识别应用于视频监控、自动视频标签和人机交互等多个领域。

人体动作识别本质上是视频分类任务，视频分类，最重要的是视频特征表达。本文首先对传统的基于时空兴趣特征点的视频特征表达方法 STIP、Chen 等提出的 MoSIFT、Dense Trajectory 以及近些年来发展较快的深度视频特征提取方式进行介绍，详细介绍两种广为使用的基于 3D 卷积的网络结构——C3D 网络和 Res3D。最后，本篇报告从以下几个方面设计实验，得出结论。

(1) 是否使用 BoW，结论：使用 BoW 明显好于不使用 BoW；

(2) MoSIFT 特征与 STIP 特征的比较：MoSIFT 特征表现显著好于 STIP 特征，其中 MoSIFT 特征使用提供的预提取特征，STIP 自行提取。

(3) MoSIFT 特征选取方式——随机选取、选取的比例、选取前若干个帧的特征，其中选取全部的特征，并且不打乱，分类效果最好；

(4) SVM 核函数的选取：大量实验证明，基于 Chi2 和基于 RBF 核函数的 SVM 分类器要好于基于线性核函数的分类效果；

(5) 对 BoW 词袋模型中的类心设置：类心不能过多也不能过少，实验中设置 128 表现最好；

(6) 本文还对 BoW 量化之后的特征进行归一化处理，实验证明性能有显著提升。

二、视频特征表达方法

传统的视频特征表达方法大多基于时空兴趣点，而近些年来，随着深度学习的发展，基于 3D 卷积的视频特征提取较为受欢迎。

2.1 时空兴趣点

这一部分主要对 STIP、MoSIFT、Dense Trajectory 时空兴趣点进行介绍。

2.1.1 STIP（Space-Time Interest Points）

Ivan Laptev[1]等人提出将空间兴趣点的概念扩展到时空域，基于 Harries 和 Forster 兴趣点算子的思想，在图像局部空间和时间区域像素值有显著变化的点作为时空兴趣点，来检测图像的时空局部结构。然后来估计检测到的事件的时空范围并且计算它们的尺度不变时空描述符，然后根据标记的时空点对事件分类，并且构建视频特征表示。

1. 检测时空兴趣点

检测出的兴趣点与选取的时间尺度与空间尺度有关，尺度与动作发生的范围（人本身的行为特点）有关。为了使该检测器能够自适应尺度变化，STIP 采用不同尺度的高斯滤波函数，对视频在时间和空间做了尺度变换。

时空兴趣点的求解采用如下思想：把视频看作三维的函数，寻找到一个映射函数，通过这个映射函数，将三维视频的数据映射到一维空间中，然后通过求此一维空间的局部极大值的点，而这些点就是兴趣点。

2. 时空尺度自适应

时空域两个尺度因子的选择不同对检测到的兴趣点有很大影响。时间尺度因子越大，则表明动作发生的时间越长，所以优先检测出动作持续时间长的兴趣点；反之，时间尺度因子越小，则优先检测动作持续时间短的兴趣点。在空间尺度因子的影响效果相同。

通过取归一化后的在时间尺度和空间尺度拉普拉斯算子极大值，可以得到时空域内动作发生的范围，即对应的时空尺度。在使拉普拉斯算子取得极大值的同时，也使响应函数（设置阈值来确定是否为兴趣点）取到极大值，就可以得到尺度自适应的时空兴趣点。

2.2.2 MoSIFT (Motion Scale Invariant Feature Transform)

MoSIFT 是 Charif 等人[3]提出的一种基于 SIFT 特征提取的时空特征算法，算法首先提取图像中的 SIFT 点特征，然后计算与 SIFT 关键点尺度相对应的光流大小。MoSIFT 算法能够检测空间上具有一定运动的、区分性强的兴趣点，运动强度由兴趣点周围的光流强度来衡量。

算法首先找到视频相邻帧中多个尺度下的 DoG 空间兴趣点（SIFT 特征点），然后计算与 SIFT 尺度相对应的光流，得到时空兴趣点。上图为 MoSIFT 算法的系统框图，输入为相邻的两帧图像，通过高斯差分，找出 DoG 空间中的局部极值点作为候选点，然后通过光流，分析判断这些候选点是否存在足够的运动信息，以决定是否作为兴趣点，最后提取特征点处的 MoSIFT 特征。将 SIFT 特征点与光流相结合，可以去除与运动无关的兴趣点，提炼出真正可以描述运动的特征。

2.2.3 Dense Trajectory（DT 算法）

DT 算法是提取基于轨迹的视频特征，在视频序列中对每一帧密集采样兴趣点进行跟踪，形成 dense trajectory。DT 算法包括密集采样特征点，特征点轨迹跟踪和基于轨迹的特征提取，然后基于 BoW 对特征点进行聚类，最后使用 SVM 分类器。

算法过程如下：

1. 得到 DT 特征

（1）对整个视频序列进行光流场计算（光流场是指图像中所有像素点构成的一种二维瞬时速度场，其中的二维速度矢量是景物中可见点的三维速度矢量在成像表面的投影）；

（2）对初始帧进行像素点密集采样，每隔 W 个像素点采样一个；

（3）对采样点进行跟踪：由光流判断跟踪点在下一帧的位置；

（4）对每个点跟踪都会形成一条 trajectory，为了避免长时间跟踪而产生的跟踪点漂移现象，可以对跟踪的长度 L 进行约束（L=15）

（5）现实视频中存在摄像头运动的缺陷，因此需要相应算法消除摄像头影响，得到最终的 DT;

每条 trajectory 都可以提取一个轨迹特征向量 S'（当 L=15，S'为 30 维），对局部动作模式进行编码，除了轨迹形状特征， DT 还使用了 HOF,HOG 和 MBH 三种特征来描述光流。在计算完后， DT 算法中对 HOG,HOF 和 MBH 均使用 L2 范数归一化。

2. 特征编码—Bag of Words

对于一段视频，存在着大量的轨迹，每段轨迹都对应着一组特征（trajectory, HOG, HOF, MBH），因此需要对这些特征组进行编码，得到一个定长的编码特征来进行最后的视频分类。在训练完 codebook 后，对每个视频的特征组进行编码，就可以得到视频对应的特征。

3. 分类

在得到视频对应的特征后，最终采用分类器进行分类。

2.2 深度视频特征

卷积神经网络（CNN）被广泛应用于计算机视觉中，包括分类、检测、分割等任务。这些任务一般针对图像进行，使用的是二维卷积。而对于基于视频分析的问题，2D 卷积不能很好得捕获时序上的信息。[4]最早提出 3D 卷积，用于行为识别。并且 3D 卷积也被用于受限玻尔兹曼机[6]和堆叠 ISA[7]来学习时空特征。基于 CNN 的动作识别最流行的方法之一是 two-stream 网络，Simonyan 等人提出了一种分别使用 RGB 和叠层光流帧作为外观和运动信息的方法[10]，表明将这两种流结合起来可以提高动作识别的精度。

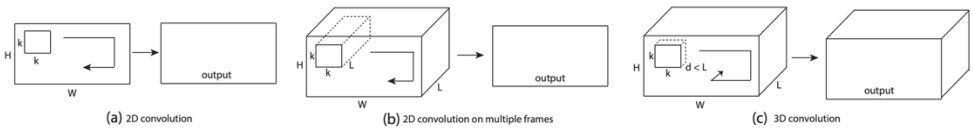

Figure 1. **2D and 3D convolution operations**. a) Applying 2D convolution on an image results in an image. b) Applying 2D convolution on a video volume (multiple frames as multiple channels) also results in an image. c) Applying 3D convolution on a video volume results in another volume, preserving temporal information of the input signal.

图 1 2D 与 3D 卷积操作

图 1 所示为 2D 卷积和 3D 卷积的区别，图 1(a)和(b)分别为 2D 卷积用于单通道图像和多通道图像的情况（此处多通道图像可以指同一张图片的 3 个颜色通道，也指多张堆叠在一起的图片，即一小段视频），对于一个滤波器，输出为

一张二维的特征图，多通道的信息被完全压缩了。而(c)中的 3D 卷积的输出仍然为 3D 的特征图。

现在输入一个视频段，其大小为 c*l*h*w ，其中 c 为图像通道(一般为 3)，l 为视频序列的长度，h 和 w 分别为视频的宽与高。进行一次 kernel size 为 333,步长为 1,并进行 padding 操作,滤波器个数为 K 的 3D 卷积后，输出的大小为 K*l*h*w*.池化同理。

这部分简单介绍两种深度视频提取的网络结构，分别为 C3D 网络和 3DRes。

(1) C3D 网络[5]

C3D 网络是一个通用网络，论文[5]中将其用于行为识别，场景识别，视频相似度分析等领域，关注三维卷积核 CNN，并且通过实验发现 3×3×3 卷积内核达到了最佳性能水平。除此之外，Varol 等人在[9]中提出扩大 C3D 输入的时间长度可以提高识别性能。如图 2 所示为 C3D 的网络结构。

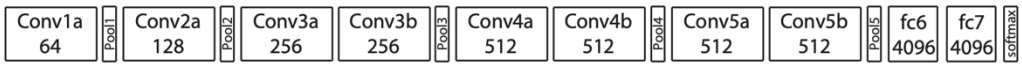

图 2 C3D 网络结构

C3D 网络有 8 个卷积层、5 个最大池化层和 2 个全连接层，然后是一个 softmax 输出层。所有 3D 卷积核都是 3×3×3，在空间和时间维度上的步长均为 1。每个框中都标出了过滤器的数量。3D 池化层从 pool1 到 pool5 表示。所有池化内核都是 2×2×2，除了 pool1 是 1×2×2。每个全连接层有 4096 个输出单元。

(2) Res3D

Res3D 仍然使用 3D ConvNets 来学习时空特征。然而，在 C3D 中，虽然工作仅限于寻找三维卷积时间维度上的卷积核核长度，但 Res3D 考虑许多了结构设计的其他方面。Res3D 相对于 C3D 在 5 种不同的基准测试中表现良好，运行时速度快了 2 倍，模型尺寸小了 2 倍。

三、视频动作识别方法

3.1 方法与步骤

一共分为四个部分，对于数据集中的视频，首先使用 STIP 或者 MoSIFT 进行特征提取，分割为测试集和训练集，用 BoW 构建 codebook 进行特征量化，生成直方图，然后使用 SVM 分类器进行分类，最后计算 3 个评估指标。

3.1.1 STIP 特征提取

1. 计算尺度归一化的二阶矩阵

首先在空间区域，用方差为σ_l^2的高斯核，来建模原始图像的线性尺度空间表示，即L^{sp}，其中*表示卷积运算，g^{sp}表示高斯核。然后对L^{sp}分别对 x，y 求偏导数得到L_x^{sp}和L_y^{sp}。代码实现的时候由于在卷积之前进行了扩充，所以在求导之前要先进行裁剪操作，保持图片尺寸不变。

$$L^{sp}(x,y;\sigma_l^2) = g^{sp}(x,y;\sigma_l^2) * f^{sp}(x,y)$$

$$g^{sp}(x,y;\sigma_l^2) = \frac{1}{2\pi\sigma_l^2} exp\big(-(x^2+y^2)\big)/2\sigma_l^2$$

$$L_x^{sp} = \partial x(L^{sp})$$

$$L_y^{sp} = \partial y(L^{sp})$$

对L_x^{sp}和L_y^{sp}进行高斯变换，将其作为特征值构造矩阵μ^{sp}，特征值构成了图像沿两个图像方向变化的描述符，代码中直接用 Lxm2smooth，Lxy2smooth，Lxmysmooth 来表示高斯变换之后的结果。

$$\mu^{sp}(\cdot;\sigma_l^2,\sigma_i^2) = g^{sp}(\cdot;\sigma_i^2) * \begin{pmatrix} (L_x^{sp})^2 & L_x^{sp}L_y^{sp} \\ L_x^{sp}L_y^{sp} & (L_y^{sp})^2 \end{pmatrix}$$

```
% 对原始图像周边扩充，同时用高斯核对原始图像卷积
L=mydiscgaussfft(extend2(f,4,4),sxl2);

% Lx 和 Ly 表示用局部尺度sxl2计算的高斯导数
% 这里进行crop是因为上一步的扩充，为了保持图片尺寸不变
% 求导之后再进行裁剪操作
Lx=crop2(filter2(dxmask,L,'same'),4,4)*sxl2^(1/2);
Ly=crop2(filter2(dymask,L,'same'),4,4)*sxl2^(1/2);
Lxm2=Lx.*Lx;
Lym2=Ly.*Ly;
Lxmy=Lx.*Ly;

% Gaussian transform
% mu
Lxm2smooth=mydiscgaussfft(Lxm2,sxi2);
Lym2smooth=mydiscgaussfft(Lym2,sxi2);
Lxmysmooth=mydiscgaussfft(Lxmy,sxi2);
```

Harris 和 Stephens 提出用角函数的正最大值

$$H^{sp} = det(\mu^{sp}) - k * trace^2(\mu^{sp}) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

```
if pointtype==1 % harris points
  detC=(Lxm2smooth.*Lym2smooth)-(Lxmysmooth.^2);
  trace2C=(Lxm2smooth+Lym2smooth).^2;

  %kparam=0.04;
  cimg=detC-kparam*trace2C;
end
```

2. 检测最大值

最后检测上一步经过处理的图像中的最大值，选择若干个响应值最强的点，并且舍弃在图像边界的点，返回其坐标以及响应值。

3. 特征提取主函数

（1） 不使用 BoW 方法

在实验过程中，取每个视频的前 50 帧（如果视频较小不足 50 帧，就取所有的帧）分别提取其 STIP 特征，返回然后对其取均值。

```
for i = 1:WantedFrames
     img=imread(['./Frames/',num2str(i),'.jpg']);
     f1=il_rgb2gray(double(img));  % 将图片转换为一个通道上
     [ysize,xsize]=size(f1);  % shape
     nptsmax=40;
     kparam=0.04;
     pointtype=1;
     sxl2=4;
     sxi2=2*sxl2;
     % detect points
     % 对于每一帧 提取STIP特征
     [posinit,valinit]=STIP(f1,kparam,sxl2,sxi2,pointtype,nptsmax);
     Test_Feat(i,1:40)=valinit;
end
Test_Feat = mean(Test_Feat); % 取平均值
```

分割测试集，训练集，并且在最后存储提取好的特征。

```
else
    Test_Feat = mean(Test_Feat); % 取平均值
    if video <= train_num
        train_features = [train_features;Test_Feat];
        train_labels = [train_labels;Test_Feat];
    else
        test_features = [test_features;Test_Feat];
        test_labels = [test_labels;Test_Feat];
    end
end
```

(2) 使用 BoW 的方法

不取平均值，而是将每一类视频中的每一个视频的所有 STIP 特征都加入特征矩阵中。是否使用 BoW，主要区别就在于此。注意这里要将同一类的视频的 STIP 特征放在一起，也就是一个 desc 结构体。

```
if do_bow
    desc = struct('features',Test_Feat);
    if video <= train_num
        train_features = [train_features;desc];
        train_labels = [train_labels;class];
    else
        test_features = [test_features;desc];
        test_labels = [test_labels;class];
    end
else
```

4. BoW 特征量化

(1) 对于训练集和测试集，都导入提取的 STIP 特征

```
desc_test(lasti)=struct('features',X_test(lasti).features,'class', Y_test(lasti))
desc_test(lasti).features = single(desc_test(lasti).features);
lasti=lasti+1;
```

(2) 构建 codebook，将所有视频的特征描述符全部拼接起来，然后进行随机采样.

```
%% build codebook
fprintf('\nBuild visual vocabulary:\n');
% 将所有视频的特征描述符全部拼接起来
DESC = [];
labels_train = cat(1,desc_train.class);
for i=1:n_class
    desc_class = desc_train(labels_train==i);
    [w,h]=  size(desc_class);
    randimages = randperm(h);
    randimages =randimages(1:5);
    DESC = vertcat(DESC,desc_class(randimages).features);
    %display(size(DESC));
end
display(size(DESC,1));
% 从训练描述符当中随机采样 构建codebook
r = randperm(size(DESC,1));
r = r(1:min(length(r),nfeat_codebook));
```

(3) 将数据集中的特征利用 K-means 聚类算法，聚为 K 类，便于之后生成直方图。

```
%% cunstruct codebook
% run k-means
K = nwords_codebook; % size of visual vocabulary
fprintf('running k-means clustering of %d points into %d clusters...\n',...
    size(X_train,1),K)
cluster_options.maxiters = max_km_iters;  % 最大迭代次数
cluster_options.verbose  = 1;

[VC] = kmeans_bo(double(X_train),K,max_km_iters);%visual codebook
VC = VC';%transpose for compatibility with following functions
clear DESC;
```

(4) 对于训练集、测试集中的数据，都利用 codebook 进行特征量化，即选择 codebook 当中，与当前特征向量欧式距离最近的一个 visual word 来表示。

```
%% feature quantization
fprintf('\nFeature quantization (hard-assignment)...\n');
% 训练集
for i=1:length(X_train)
  feature = X_train(i);
  dmat = eucliddist(feature,VC);  % 计算这一feature map与VC中所有向量的欧式距离
  [quantdist,visword] = min(dmat,[],2);  % 用VC中距离最近的簇平均值来量化这一个feature map
  % save feature labels
  desc_train(i).visword = visword;  % 距离最近的visual word
  desc_train(i).quantdist = quantdist;
end
```

(5) 这里我对生成的直方图（训练集和测试集）进行 L1 范数归一化处理，用这个特征的 visual word 标签的归一化直方图表示每个图像。在每个图像上计算单词直方图 H，将直方图 L1 范数归一化。这里给出训练集的代码，测试集同理。

```
N = size(VC,1); % visual words的个数

for i=1:length(desc_train)
    visword = desc_train(i).visword;  % 每个图片对应的visual word
    H = histc(visword,[1:nwords_codebook]);

    % L1范数归一化
    if norm_bof_hist
        H = H/sum(H);
    end

    % save histograms
    desc_train(i).bof=H(:)';
end
```

3.1.2 MoSIFT 特征提取

1. 对于预提取的 MoSIFT 特征，处理方式与 STIP 基本相同，只是数据格式不一样，STIP 特征我是自行提取的，格式为 mat，而在预提取的 MoSIFT 特征当中，前六列表示特征点的位置，后面的 256 列才表示真正的特征向量，所以需要删掉前六列的数据。如下：

```
fprintf('Loading %s \n',fname);
tmp = load(fname,ext);
[w,h] = size(tmp);
maxx = min(w, n_frame);
tmp = tmp(:,7:end); % 删掉前六列表示坐标的数据
```

2. 这里我设置了两种方式来抽取其中的特征，第一种是取前若干行，第二种是随机抽取一定百分比的数据，分别进行实验，如下：

```
if do_rand  % 随机抽取
    tmp = tmp(randperm(floor(w*ratio)),:);
    display(size(tmp));
else
    tmp = tmp(1:maxx,7:end);
end
```

3. 不同数据集的不同处理方式

由于所给的预抽取特征有的是 txt 格式，有的是 mat 格式，所以处理方式也不相同，需要因数据集而异。

```
if strcmp(dataset_dir ,'hmdb51') || strcmp(dataset_dir ,'dtdb')
    desc_test(test_id)=struct('features',tmp.sift, 'class',i);
    desc_test(test_id).features = single(desc_test(test_id).features);
else
    desc_test(test_id)=struct('features',tmp, 'class',i);
    desc_test(test_id).features = single(desc_test(test_id).features);
end
```

3.1.3 SVM 分类

一共使用基于三种不同的核函数的 SVM 分类器，分别是线性核函数、卡方核函数、RBF 核函数。除了线性核函数核函数，都要在参数空间中选择一个正确率最高的 gamma 值和惩罚系数，而线性核函数只需要选择一个最好的惩罚系数即可。

(1) 线性核函数，主要用于线性可分的情况

$$K(x, x_i) = x * x_i$$

```
if do_svm_linar_classification
    % cross-validation
    C_vals=log2space(7,10,5);  % 参数空间设定
    for i=1:length(C_vals);
        % 参数空间搜索 五折交叉验证
        opt_string=['-t 0  -v 5 -c ' num2str(C_vals(i))];
        xval_acc(i)=svmtrain(Y_train,X_train,opt_string);
    end
    % 选择最好的c
    [v,ind]=max(xval_acc);  % ind为C_vals中最好c值的索引

    model=svmtrain(Y_train,X_train,['-t 0 -c ' num2str(C_vals(ind))]); % 训练
    disp('*** SVM - linear ***');
    svm_lab=svmpredict(Y_test,X_test,model);  % 预测
    CM = confmatrix(Y_test,svm_lab,n_class); % 混淆矩阵
    acc = sum(Y_test==svm_lab)/length(Y_test);  % 准确率
    method_name='SVM linear';
    fprintf('OVERALL %s classification accuracy: %1.4f\n\n',method_name,acc);
end
```

(2) 高斯径向基核函数，即 RBF，局部性强，可以将一个样本映射到一个更高维的空间内，应用最广，无论大样本还是小样本都有比较好的性能。

$$K(x,x_i) = exp(-\frac{||x-x_i||^2}{\delta^2})$$

这里首先需要在参数空间内选择最好的 c（惩罚系数）和 gamma 值，将其作为 RBF 函数的参数训练 SVM 模型，得到预测结果。

```
if do_svm_rbf_classification
    % cross-validation
    C_vals=log2space(2,10,5);
    Gamma_vals = log2space(-5, 10, 10);  % 参数空间
    % 选择最好的c和最好的gamma值
    Acc_best = 0;
    for i=1:length(C_vals);
        for j=1:length(Gamma_vals)
            % 5折交叉验证
            opt_string=['-t 2 -v 5 -c ' num2str(C_vals(i)) ' -g ' num2str(Gamma_vals(j))];
            xval_acc=svmtrain(labels_train,bof_train,opt_string);
            if xval_acc > Acc_best
                Acc_best = xval_acc;
                C_best = C_vals(i);  % 最好的c
                Gamma_best = Gamma_vals(j);  % 最好的gamma值
            end
        end
    end

    % train the model and test
    model=svmtrain(labels_train,bof_train,['-t 2 -c ' num2str(C_best) ' -g ' num2str(Gamma_best)] );
```

(4) 卡方核函数

$$K(x,x_i) = 1 - \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\frac{1}{2}(x_i + y_i)}$$

原本的优化问题如下：

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i y_i = 0,\ \ \alpha_i \geq 0, i = 1, 2, \ldots, N$$

在加了核函数之后，优化问题变为以下形式，将原式中的内积，改为用核函数对两个特征向量的内积进行变换，这样做等价于先对向量进行映射然后再做内积：

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i^T \cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i y_i = 0,\ \ \alpha_i \geq 0, i = 1, 2, \ldots, N$$

```
if do_svm_rbf_classification
    % cross-validation
    C_vals=log2space(2,10,5);
    Gamma_vals = log2space(-5, 10, 10);

    % 选择最好的c和gamma值
    Acc_best = 0;
    for i=1:length(C_vals);
        for j=1:length(Gamma_vals)
            % -t=2:  表示选择线性RBF函数
            % -c: 表示惩罚系数
            opt_string=['-t 2 -v 5 -c ' num2str(C_vals(i)) ' -g ' num2str(Gamma_vals(j))];
            xval_acc=svmtrain(labels_train,bof_train,opt_string);
            if xval_acc > Acc_best
                Acc_best = xval_acc;
                C_best = C_vals(i);
                Gamma_best = Gamma_vals(j);
            end
        end
    end
    % 用最好的c和gamma值训练SVM并且预测
    model=svmtrain(labels_train,bof_train,['-t 2 -c ' num2str(C_best) ' -g ' num2str(Gamma_best)] );
```

3.1.5 评估

在评估部分，我主要实现了混淆矩阵、召回率、精确率的计算。其中 Precision = TP / (TP + FP)；Recall = TP / (TP + FN)，首先计算混淆矩阵，然后再根据混淆矩阵计算召回率、精确率。

混淆矩阵计算：

```
CM = zeros(ntypes);
for i=1:ntypes
    vals = IDXpred( IDXtrue==i );  % CM(i,j)表示真实值为i，预测值为j的个数
    for j=1:ntypes CM(i,j) = sum(vals==j); end;
end
```

部分实验的混淆矩阵结果见文件“混淆矩阵.xlsx”。

3.1.6 可视化

这一部分对 KTH，hmdb，dtdb 数据集中的每一类当中的第一个视频中的 STIP 特征、MoSIFT 特征进行可视化。

1. STIP 特征可视化

如图所示，为特征在 KTH 数据集中的第一张图片的可视化效果，提取代码与之前一致。

(1) 需要首先把每一帧提取出来存储。

```
for k = 1:WantedFrames
    mov(k).cdata = read( I, k);
    mov(k).cdata = imresize(mov(k).cdata,[256,256]);
    imwrite(mov(k).cdata,['./KTH_vis/',num2str(k),'.jpg']);
end
```

(2) 然后计算 STIP 特征，并且在先前提取的图片中画出，然后存储，结果如图 3 所示。

```
% title( 'Feature Points' ,'fontsize',12,'fontname','
F=getframe(gcf);
temp_path = ['./KTH_vis/',num2str(i),'/'];
imwrite(F.cdata,[temp_path,num2str(j),'.jpg']);
disp(['saving in',temp_path]);
```

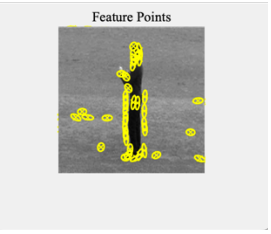

图 3 STIP 特征可视化

(3) 在最外层需要写一个循环，否则要手动选择每一张图片进行可视化。

```
filePath = [pwd, '/KTH/'];
dir = dir(fullfile(filePath, '*.mp4'));
each_num = 100;
n_class=6;
for i = 1:each_num:1
    aname = sprintf('%s%d.mp4',filePath,i);
```

2. MoSIFT 特征可视化

由于 mosift 使用了提供的预提取特征，所以主要的不同点也在得到特征点坐标的部分。前两列分别表示特征的 x，y 坐标，将其在提取的帧上标出即可。首先需要得到每一帧中的特征坐标。另外，有的数据集在提取帧的过程中，需要去掉 resize 的一步。最后，由于预提取的特征数据**除了位置信息，还有方向信息**，所以我还在图中可视化处特征的方向，如图 4 所示为可视化之后的 MoSIFT 特征。

```
mosift = load(['../mosift/', dataset, '/',[num2str(i), '.txt']]);
% 加载预提取特征
disp(['loading from ', '../mosift/', dataset, '/',[num2str(i), '.txt']]);
imshow(f1,[],'border','tight'), hold on
axis off;
pos = mosift(mosift(:,3)==j,:);  % 提取这一帧的mosift位置
for k=1:size(pos,1)
    plot(pos(k,1),pos(k,2),'ro');  % pos(k,1)表示x坐标，pos(k,2)表示y坐标
    quiver(pos(k,1), pos(k,2),pos(k,5), pos(k,6))  % 方向可视化
end
```

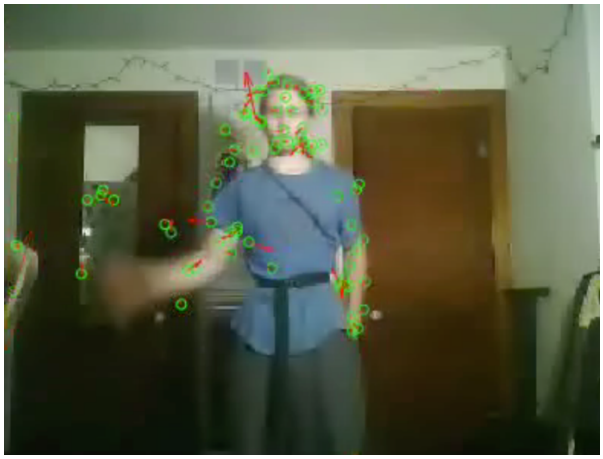

图 4 MoSIFT 特征可视化

(3) 在把每一个特征点打到视频的帧上之后，将输出的图片用 matlab 的 vedioWriter 函数进行视频合成，代码如下；

```
route='./dtdb_vis/';%基本路径
for dir=1:100:1800
    route='./dtdb_vis/';%基本路径
    mp4file = [route, num2str(dir), '_stip', '.mp4'];
    route = [route, num2str(dir), '/'];
    WriterObj=VideoWriter(mp4file,'MPEG-4');%待合成的视频的文件路径
    open(WriterObj);

    %n_frames=numel(d);% n_frames表示图像帧的总数
    for j=1:50
    filename=strcat(route,num2str(j),'.jpg');
    frame=imread(filename);%读取图像，放在变量frame中
    writeVideo(WriterObj,frame);%将frame放到变量WriterObj中
    %%为每一帧图像编号
    end
    close(WriterObj);
end
```

3.2 结果及分析

3.2.1 数据集

1. KTH 数据集：一共 6 类，分别为数据库包括在 4 个不同场景下 25 个人完成的 6 类动作：walking, jogging, running, boxing, hand waving and hand clapping，每一类视频有 100 个样本。

2. HMDB 数据集：一共有 20 类，每一类视频有 100 个样本，动作主要分为以下五类。

1）一般面部动作：微笑，大笑，咀嚼，交谈。

2）面部操作与对象操作：吸烟，吃，喝。

3）一般的身体动作：侧手翻，拍手，爬，爬楼梯，跳，落在地板上，反手翻转、倒立、跳、拉、推、跑，坐下来，坐起来，翻跟头，站起来，转身，走，波。

4）与对象交互动作：梳头，抓，抽出宝剑，运球、高尔夫、打东西，球、挑、倒、推东西，骑自行车，骑马，射球，射弓、枪、摆棒球棍、剑锻炼，扔。

5）人体动作：击剑，拥抱，踢某人，亲吻，拳打，握手，剑战。

3. dtdb 数据集：一共有 18 类，每一类 100 个样本，类别为：Chaotic_motion, Dominant_multiple_objects, Dominant_non-rigid, Dominant_rigid, Explosion_Divergence, Geyser, Pluming, Rotary_motion, Rotary_side_view, Scintillation, Stochastic_motion, Superimposed_motions, Superimposed_translation_and_stochastic, Turbulance, 'Underconstrained_aperture_problem, Underconstrained_blinking, Underconstrained_flicker, Wavy_motion

3.2.2 不同数据集中是否使用 BoW 以及不同特征计算方式的影响

超参数设置：

codebook 维数：128

在 mosift 特征提取当中，在这一部分当中，由于 MoSIFT 特征过多，我使用随机选取一定百分比的特征进行分类。这里召回率与精确率都取了平均值。只列出部分准确率以及参数设置的情况，在文件“实验数据.xlxs”中有更加详细的数据。

数据集	特征	SVM 核函数	是否使用 BoW	选取特征百分比	准确率（%）	精确率（%）	召回率（%）
KTH	STIP	线性	是	/	38.33	38.33	38.52
KTH	STIP	RBF	是	/	44.17	44.17	45.39
KTH	STIP	Chi2	是	/	41.67	41.67	42.61
KTH	STIP	线性	否	/	17.5	17.5	10.69
KTH	STIP	RBF	否	/	17.5	4.17	16.84
KTH	STIP	Chi2	否	/	15.83	3.33	10.69
dtdb	STIP	线性	是	/	13.33	13.33	12.82
dtdb	STIP	RBF	是	/	15.89	13.89	14.28
dtdb	STIP	Chi2	是	/	14.44	14.44	15.48
dtdb	STIP	线性	否	/	8.61	8.61	8.91
dtdb	STIP	RBF	否	/	10.83	10.83	11.65
dtdb	STIP	Chi2	否	/	11.67	11.67	10.72
hmdb51	STIP	线性	是	/	15.25	15.25	15.96
hmdb51	STIP	RBF	是	/	19.5	19.5	19.41
hmdb51	STIP	Chi2	是	/	18.5	18.5	19.4
hmdb51	STIP	线性	否	/	15.25	15.25	15.96
hmdb51	STIP	RBF	否	/	19.5	19.5	19.41
hmdb51	STIP	Chi2	否	/	15.5	15.5	18.06
KTH	MoSIFT	线性	是	50%	70.83	70.83	71.70
KTH	MoSIFT	RBF	是	50%	73.33	73.33	74.51
KTH	MoSIFT	Chi2	是	50%	70.00	70.00	71.70
KTH	MoSIFT	线性	否	50%	65.88	65.88	66.72
KTH	MoSIFT	RBF	否	50%	70.0	70.0	71.30
KTH	MoSIFT	Chi2	否	50%	69.8	69.8	70.52
dtdb	MoSIFT	线性	是	10%	21.94	21.94	22.50
dtdb	MoSIFT	RBF	是	10%	22.5	22.5	23.52
dtdb	MoSIFT	Chi2	是	10%	23.61	23.61	24.58
dtdb	MoSIFT	线性	否	10%	19.72	19.72	20.2
dtdb	MoSIFT	RBF	否	10%	20.28	20.28	21.88
dtdb	MoSIFT	Chi2	否	10%	11.11	11.11	10.62
hmdb51	MoSIFT	线性	是	10%	18.5	18.5	18.82
hmdb51	MoSIFT	RBF	是	10%	20.75	20.75	20.54
hmdb51	MoSIFT	Chi2	是	10%	23.75	23.75	23.20

hmdb51	MoSIFT	线性	否	10%	17.50	17.50	17.92
hmdb51	MoSIFT	RBF	否	10%	19.50	19.50	19.98
hmdb51	MoSIFT	Chi2	否	10%	21.75	21.75	22.80

从实验结果来看，在三个数据集中，MoSIFT 特征要远好于 STIP 特征，三种核函数中，RBF 和 Chi2 核函数不相上下，线性核函数表现相对较弱。

3.2.3 KTH 数据集——聚类类心个数的影响

由于 KTH 数据集规模较小，这里使用 KTH 数据集调整 BoW 中的超参数，以下为实验结果。其中 nfeat_codebook 表示 K-means 聚类中，用于 codebook 生成的描述符数量，即从很多个拼接起来的 sift 特征矩阵当中，随机抽取 nfeat_codebook 个特征向量。

STIP 特征：

特征	SVM 核函数	聚类类心个数	Nfeat_codebook	准确率（%）
STIP	线性	500	60000	38.33
STIP	线性	800	60000	30.00
STIP	线性	128	60000	**40.00**
STIP	线性	256	60000	35.00
STIP	线性	64	60000	30.83
STIP	RBF	500	60000	41.17
STIP	RBF	800	60000	43.33
STIP	RBF	128	60000	**44.17**
STIP	RBF	256	60000	37.50
STIP	RBF	64	60000	34.17
STIP	Chi2	500	60000	40.00
STIP	Chi2	800	60000	40.83
STIP	Chi2	128	60000	**48.33**
STIP	Chi2	256	60000	36.67
STIP	Chi2	64	60000	41.67

Mosift 特征：

特征	SVM 核函数	聚类类心个数	随机选取特征比例	Nfeat_codebook	准确率（%）
MoSIFT	线性	500	20%	60000	58.33
MoSIFT	线性	800	20%	60000	68.33
MoSIFT	线性	128	20%	60000	**70.83**
MoSIFT	线性	256	20%	60000	58.33
MoSIFT	线性	64	20%	60000	52.50

MoSIFT	RBF	500	20%	60000	60.00
MoSIFT	RBF	800	20%	60000	74.17
MoSIFT	RBF	128	20%	60000	**73.33**
MoSIFT	RBF	256	20%	60000	65.83
MoSIFT	RBF	64	20%	60000	61.67
MoSIFT	Chi2	500	20%	60000	**74.17**
MoSIFT	Chi2	800	20%	60000	73.17
MoSIFT	Chi2	128	20%	60000	70.00
MoSIFT	Chi2	256	20%	60000	73.33
MoSIFT	Chi2	64	20%	60000	63.33

由实验结果来看，类心个数并不是越多越好，也不是越少越好，在实验过程中证明取 128 的时候分类准确率较为理想。

3.2.4 KTH 数据集——mosift 特征选取方式的影响

由于每个视频的 mosift 特征包含了视频的所有帧的数据，所以这里我设计了两种选取特征的方式。一种是选取前若干个向量，另外一种方式是随机抽取一定百分比个特征。N_frame 表示的是在 mosift 特征提取当中，从预提取的特征当中取了前 n_frame 帧的特征向量。如果某个视频的总帧数小于设置的 n_frame，则取这个视频的全部特征。

但是考虑到在视频的不同时间点可能会有不同的信息，所以我后来使用随机抽取不同比例的特征向量来构建 codebook，进行实验对比。

特征	SVM 核函数	聚类心个数	Nfeat_codebook	选取特征方式	准确率（%）
Mosift	Chi2	500	60000	前 100	65.00
Mosift	Chi2	500	60000	前 1000	**72.50**
Mosift	RBF	500	60000	前 100	59.17
Mosift	RBF	500	60000	前 1000	**73.33**
Mosift	线性	500	60000	前 100	61.67
Mosift	线性	500	60000	前 1000	**70.00**
Mosift	线性	128	60000	随机 20%	66.67
Mosift	线性	128	60000	随机 50%	65.00
Mosift	线性	128	60000	全部随机	70.83
Mosift	线性	128	60000	全部不随机	**72.5**
Mosift	RBF	128	60000	随机 20%	63.33

Mosift	RBF	128	60000	随机 50%	69.17
Mosift	RBF	128	60000	全部随机	71.67
Mosift	RBF	128	60000	全部不随机	**78.33**
Mosift	Chi2	128	60000	随机 20%	68.44
Mosift	Chi2	128	60000	随机 50%	70.83
Mosift	Chi2	128	60000	全部随机	71.67
Mosift	Chi2	128	60000	全部不随机	**74.17**

(1) 对于选取前若干个特征的选取方式来说，**取前 1000 个的效果要远好于取前 100 个的效果，说明对于视频分类来说，还是特征越充足，分类越准确。**

(2) **对于随机选取一定百分比的特征选取方式来说，可以明显看到，取所有特征并且不打乱其顺序，分类的准确率最高，**与我们的预期也相同，这是因为视频本身具有时序的特征，如果将这些帧的顺序打乱，则会丢失掉时序的信息。并且，特征越多，可以把视频分类的更准确女，否则部分特征是无法精确代表这一类视频的。

3.2.5 KTH 数据集——对直方图量化的提升方法

这里我还对量化之后的直方图进行归一化处理，效果有所提升。

Nfeat_codebook = 60000，使用 KTH 数据集，特征提取方式为随机提取 50%的特征。

特征	SVM 核函数	聚类类心个数	是否归一化	准确率（%）
STIP	线性	128	是	**40.83**
STIP	线性	128	否	35.83
STIP	RBF	128	是	**42.50**
STIP	RBF	128	否	40.00
STIP	Chi2	128	是	**37.50**
STIP	Chi2	128	否	36.67
MoSIFT	线性	128	是	65.00
MoSIFT	线性	128	否	**68.44**
MoSIFT	RBF	128	是	69.17
MoSIFT	RBF	128	否	**70.83**
MoSIFT	Chi2	128	是	70.83
MoSIFT	Chi2	128	否	**71.67**

由实验结果来看，无论是 MoSIFT 特征还是 STIP 特征，在量化之后进行归一化处理，准确率都有提升，说明数据的归一化比较重要。

四、总结

在本次课程设计中，主要实现了两种视频特征 MoSIFT 以及 STIP 提取，然后使用 BoW 算法量化特征用来表征视频，并且实现了特征可视化。然后使用 SVM 分类器对测试集进行预测。并且我对 MoSIFT 和 STIP 量化之后的特征都进行归一化处理，通过实验对比，性能显著提升。除此之外，我对于可能影响分类结果的各种超参数设计实验，实验分析结果见以上实验部分。

在实验中我也遇到了一些困难，如提取特征较慢、电脑内存有限制，但是针对以上问题，我选取一部分特征来处理，所以在实验的精确程度方面会有折扣，如果不进行特征选择，选用全部特征进行实验，效果应该会好于本次报告当中的结果。

最开始遇到视频格式的解码问题，即 videoreader 读视频的时候会报错，以为是不同版本的 matlab 的问题，因此尝试了几种不同版本的 matlab，后来发现是 mac 不具有 avi 格式的解码器，于是想到的解决办法是将 avi 格式的视频转为 mp4 格式，如此来读取视频中的帧进而提取特征。

并且起初代码的 bug 我也花了很久时间去解决，最后发现是数据集分割的问题导致分类准确率很低，SVM 将测试集所有数据都分为同一类，实际上是我在生成数据样本标签的时候出了一个小错误，这是一个很小的问题，以后的学习过程当中应该注重细节，尽量每做一步都确定其准确性，减少 debug 时间成本。

通过本次实验，我对 BoW、基于传统视频特征提取的视频动作识别有了更深刻的认识，但是实验结果表明基于传统特征的视频分类准确率较低，尤其是多分类数据集，在之后的学习中，我会尝试提取深度视频特征进行视频分类。

参考文献

[1] Laptev, Ivan. "On space-time interest points." International journal of computer vision 64.2-3 (2005): 107-123.

[2] H. Wang, A. Kläser, C. Schmid and C. Liu, "Action recognition by dense trajectories," CVPR 2011, 2011, pp. 3169-3176, doi: 10.1109/CVPR.2011.5995407.

[3] Chen, Ming-yu, and Alexander Hauptmann. "Mosift: Recognizing human actions in surveillance videos." (2009).

[4] Ji S, Xu W, Yang M, et al. 3D convolutional neural networks for human action recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2013, 35(1): 221-231.

[5] Tran D, Bourdev L, Fergus R, et al. Learning spatiotemporal features with 3d convolutional networks[C]//Proceedings of the IEEE International Conference on Computer Vision. 2015: 4489-4497.

[6] G.W.Taylor,R.Fergus,Y.LeCun,andC.Bregler. Convolutional learning of spatio-temporal features. In ECCV, pages 140–153. Springer, 2010.

[7] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learn- ing hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In CVPR, 2011.

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convo- lutional neural networks. In CVPR, 2014.

[9] G. Varol, I. Laptev, and C. Schmid. Long-term tempo- ral convolutions for action recognition. arXiv preprint, arXiv:1604.04494, 2016.

[10] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 568–576, 2014.