A KEY POSE MODEL FOR HUMAN INTERACTION RECOGNITION AND COLOR FROM GRAY BY OPTIMIZED COLOR ORDERING

by

Arash Vahdat

B.Sc, Sharif University of Technology, 2009

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2011

APPROVAL

Name: Arash Vahdat

Degree: Master of Science

Title of thesis: A Key Pose Model for Human Interaction Recognition and Color from Gray by Optimized Color Ordering

Examining Committee:

Dr. Richard Vaughan
Chair

Dr. Greg Mori, Senior Supervisor

Dr. Mark Drew, Senior Supervisor

Dr. Ze-Nian Li, SFU Examiner

Date Approved:

Abstract

In this thesis, we attempt to address two different problems in computer vision. First, the human interaction recognition is discussed whose goal is to recognize the action type of interacting humans in a sequence of images. We model the interaction using a sequence of key poses, important atomic-level actions performed by the actors. Spatial arrangements between the actors are included in the model as is strict temporal ordering of the key poses. An exemplar representation is used to model the variability of the key poses, and spatial arrangement and temporal order of key poses are modeled in a structured latent model.

Second, we attack the problem of color from gray. In this problem, we are interested in construction of a gray level image which can be used to recover color image with the minimum amount of embedded information. We propose to use a parametric curve which maps the gray values to rich samples of color space. In that way, we merely have to transmit the parameters for the curve itself along with the gray image in order to recover an approximation of color. The method rests on an optimization of curve parameters in which gray is selected from nearby color points on curve such that the overall gray error is minimized whilst also minimizing the color error.

To my parents for their love,
endless support
and encouragement.

"The middle path is the way to wisdom."

— Rumi, 1207-1273

Acknowledgments

I would like to take advantage of this opportunity to appreciate numerous people who influenced this thesis and my studies at Simon Fraser University. I am truly grateful to my supervisors, Dr. Greg Mori and Dr. Mark Drew for their patience, support, encouragement, and help in both my education and life. I have been fortunate to work in Vision and Media Lab(VML) where constructive and supportive students and faculty are gathered. Specially, I wish to express my gratitude to Mani Ranjbar and Weilong Yang for their generous discussion and genuine directions. My warm thanks go to my colleague Bo Gao whose cooperation was essential for this thesis. Finally, I appreciate valuable comments by committee Dr. Ze-Nian Li and I am so grateful to Dr. Richard Vaughan for taking the time to serve as chair at my defense seminar.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In last decades, the study of various emerging computational problems has formed many new fields in computer science. Computer vision is one of those new fields formed to study automatic perception of the world through cameras. Computer vision widely discusses object recognition, digital photography, automatic surveillance, navigation, content based media retrieval and in general any application whose goal is to replace or improve human visual and recognition system with a (semi-) automatic system.

In this thesis, we try to address two different problems in computer vision. First, in Part. I the human interaction recognition is discussed whose goal is to recognize the action class of interacting humans in a sequence of images. Later, in Part. II we will attack the problem of color from gray. In this problem we are interested in construction of a gray level image which contains color information with the minimum amount of embedded information.

1.1 Human Interaction Recognition

Computer-vision based analysis of human movement is a broad active area of research. The solution to this problem can be used in automatic surveillance, computer game industry and content based video retrieval. Surveillance is usually needed in secured areas such as airports to decrease the emergency service arrival time. However, surveillance videos contain huge amount of visual data with rare occurrence of abnormal actions, and action recognition can be used to save the operator time by automatic detection of abnormal activities. In addition, action recognition helps game industry to create new controllers which enable players to play with their body instead of using small handheld devices. Content based video retrieval systems are used by content providers to search among huge

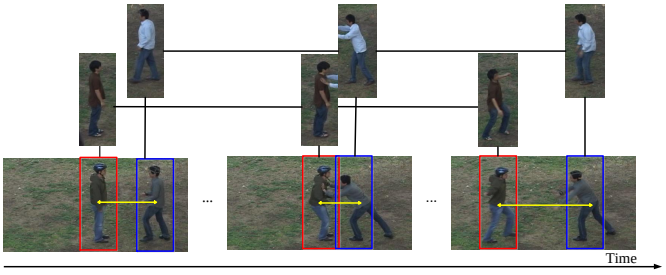


Figure 1.1: High level depiction of our model. Horizontal axis represents time. We attempt to recognize an interaction by localizing the key poses of its subjects. In the localization of key poses we consider the common distance of the poses for the action, and their order in temporal domain. We use exemplar representation of key poses which is visualized in upper row.

number of videos based on their content extracted automatically. Action labels of involving subjects in videos can be used in query of such systems which arise the need for automatic action recognition in visual database.

In this thesis, we focus on recognizing *activity*-level interactions between individuals, those composed of *atomic*-level poses or movements. We here distinguish between action, activity and interaction. Actions are atomic short movements or poses of human such as waving hand, bending and jumping. Activities are composed of several actions that occur in particular order such as shopping in a store or paying the bill. Finally, interaction is a subset of activities which is caused by several subjects. Examples of such interactions are people embracing, shaking hands, and pushing each other.

Similar to many recognition problems, interaction recognition also has several intrinsic challenges. First, intra class variation is natural due to difference of subject's behavior at different environments. For example, one may kick high in upper body; however, one may hit the opponent's leg. The speed of action and also body shape of actors can vary from a subject to another subject. In addition, environmental constraints such as obstacles or different illumination conditions change the appearance of an action. Second, extraneous movements or poses may occur in longer activity level scenarios. The subject may prepare before taking action or avoid an obstacle while performing. These intermediate poses are irrelevant for action recognition or can be shared between different classes.

The approach we take in this thesis models an interaction with a set of *key poses* of the individuals involved. In a sequence of human poses, key poses are defined as those extreme poses in human motion which are discriminative for action recognition. For example, in pushing action these

key poses can be considered as stepping forward and pushing done by a subject, as well as falling backward of other subject. Transitional poses between the key poses or initial and terminal standing poses are not considered as key poses since they do not have extra information than key poses or they are common between many classes. A high-level depiction of the model is shown in Fig. 1.1. We use an exemplar-based model of the instantiation of these key poses. We believe the spatial configuration of actors and temporal order of these key poses provide discriminative cues for interaction recognition. We will consider these cues when we are looking for key poses of an interaction in a sequence. In contrast with many exemplar-based methods that match exemplars to the whole sequence, we localize key poses in small subset of frames which enables us to discard irrelevant poses.

The main contribution of this thesis is the development of this model. This work was done in collaboration with Bo Gao, Mani Ranjbar, and Greg Mori. The contribution of the author was centered around development of sparse key pose sequence model and dynamic program for the inference.

1.2 Color From Gray

Color quantization consists of reducing the number of color levels, usually in a lossy fashion. This technique is used to compress an image to display on devices such as legacy monitors or handheld devices that support limitted number of colors. Typically, color quantization starts with a clustering of color points of an image, followed by representation of each color with the index of its cluster. In decoder side, image can be recovered using the color look-up table and indices of each pixel. Similarly, transforming a color image to gray scale takes a 24-bit color pixel into a 256-level (or fewer) gray scale value.

Here, we are concerned with the question of what is the minimum amount of auxiliary information necessary to transmit, along with gray levels, to be able to have the decoder side generate both an accurate gray image as well as an accurate color representation. Solution to this problem enables us to design scalable image compression standards that both devices with and without color display can use at the same time. Especially if the devices with gray display have limited computational power, they can display the image without any convertion from color to gray.

But we also wish to develop a color-to-grayscale transformation which as best as possible also encodes color information in the gray values, and we'll be willing to accept some gray level error provided we also obtain a good level of color accuracy. Here, we make use of a parametric curve,

which maps gray scale space to points in color space. Then one can reconstruct an approximation of the input color image from the gray scale image simply by also using the mapping from gray to color which can be rebuild from a few parameters of curve.

We make the parametric curve adaptive to the input image by optimizing its parameters such that the mapping produces a gray scale and a reconstructed color image with minimum amount of error. Here, we use a parametric expression of traversing color space that takes merely 13 parameters, and these would easily be included in a file header. We transform colored image to CIELAB color space which is designed in a way that the Euclidean distance of points measures the magnitude of observed perceptual difference.

The contribution here is the notion of utilizing a curve that visits gray value planes in color space, as well as the novel mapping used in the parametric curve: The curve is designed so as to minimize color error adaptively according to the data content of the input image. The intent of the thesis is to examine whether one can indeed use a small characterization of the gamut of an image, encompassed by a simple curve, to represent color using gray. Results are shown to be promising.

Part I

Human Interaction Recognition

Chapter 2

Previous Work

The action recognition literature in computer vision is immense. Weinland et al. [59] review the previous works in four groups including spatial action representations, temporal action representations, action segmentation and view-independent action recognition. Poppe [44] also provides recent works on action recognition in two category including image representation and action classification techniques. Moeslund et al. [41] review the work on human tracking, pose estimation, and action recognition. Forsyth et al. [16] provide a survey on human pose and motion analysis which share many similarity to action recognition taxonomies, and Krüger et al. [30] discuss the recent works on action recognition and mapping in robotics context.

In this chapter, we review the previous works in three sections. In Sec. 2.1, we focus on proposed features that describe human action. In Sec. 2.2, we review the action recognition models, and finally we discuss the interaction models in Sec. 2.3.

2.1 Action Representation

In this section we discuss the features that are proposed to represent an action. The desiderata of these features are robustness to background clutter, noise, occlusion, intera-class variation as well as being discriminative. Overall, proposed feature can be classified to three groups including: global features, local features and hybrid features. Global feature are those represent the action as a whole around the subject. Local features describes the action as a collection of local cuboids, and finally, hybrid features typically divides the image sequence in spatial and temporal domain and calculate local features in each segments.

2.1.1 Global Features

Global features represent the action as whole feature around the subject. In this group of features, the human is typically detected by background subtraction or a human detector and tracker. Next, the image around the subject is described using a feature. Global features are rich in represention of spatial and temporal configuration and appearance of action; However, they are sensitive to the performance of localizer, noise, and occlusion. A small shift in bounding box around the subject or partial occlusion will change the total feature and can cause recognition failure.

Silhouette images are one of the earliest global features proposed for action recognition. Bobick and Davis [4] proposed motion energy image (MEI) and motion history image (MHI) to recognize action. They use background subtracted images to calculate the silhouette image for each frame, and they aggregate them over time to construct the MHI. MEI is a binary version of MHI that can be built by threshholding it. As the final descriptor, they represent MEI and MHI using Hu moments [24] which were proposed for scale-, orientation- and translation-invariant visual pattern matching. Chen et al. [5] used the contour of human body and extract star skeleton by finding distance extrems of points on the contour to the centroid point which are likely to be legs, hands and head.

Gorelick et al. [20] created a 3D silhouette volume for a sequence by stacking silhouette images over time. Then, they use the Poisson equation to calculate the mean time that a random walker needs to reach to the boundary of 3D silhouette. Solution to the equation is a measure of distance to the boundary of 3D silhouette and can be used to extract local features such as space-time saliency, shape structure, and orientation of action. Their final global descriptor is formed by computing wighted moments over these local features.

A drawback of silhouette based features is their sensitivity to forground detection which may fail in the case of dynamic background. One way to overcome such a problem is to use Chamfer distance between pre-computed silhouette image and edge map. Weinland and Boyer [58] proposed an exemplar based action recognition and they represent exemplars using silhouetted image. In matching the exemplars to an input sequence, they use Chamfer distance which measures the distance between a point on edge map and the closest point on silhouette edges. In this way, they eliminate the background subtraction in test stage.

Moreover, a silhouette image is not discriminative for the poses with self occlusion. Many previous works used optical flow for action recognition which captures the movement of a subject's body part in consequent images. Even though optical flow may carry noise with a dynamic background or

moving camera, it reflects better cues of action than silhouette image. Efros et al. [12] calculated optical flow in a human centric bounding box to compensate camera movement. They divided motion flow vectors to four channels based on their sign in vertical and horizontal components, and, they filtered each channel using a Gaussian filter to overcome the camera jitter. Finally, all channels are flattened together to form the feature. Ali and Shah [1] used optical flow to define a set of kinematic features such as diversity, vorticity of optical flow field. They use Principal Component Analysis to find the dominant mode for each kinematic feature and they represent the action using bag of kinematic modes.

Another group of global features are formed using body part detection and tracking. In Ramanan and Forsyth [46], the pose of subject is represented by output of body parts detector and tracker. Rao et al. [47] also used the 2D track of hands for action recognition. The performance of body part-based features rely on human pose estimation which is still an open challenging problem in video data. Failure in pose estimation is caused due to the large degree of freedom of human body and background texture. In addition, 3D pose reconstruction suffers from ambiguity of back projection from 2D space to 3D space in the case we are using one camera. In contrast, optical flow or silhouette features do not rely on the track of body parts and represent an action using dense image features.

2.1.2 Local Features

Local features describe an action using a collection of local cuboids. Instead of human localizing, interest points are detected using an interest point detector or dense sampling. Next, image around the interest points are represented using a feature such as motion or image gradient, and the final action representation is typically formed as histogram of quantized visual words. Local features do not rely on a human localizer and are robust against partial occlusion. However, they usually discard all spatial and temporal configuration which can cause failure on complex activities.

Interest points can be detected either automatically using a interest point detector from image sequence or by dense sampling. Interest point detectors attempt to detect regions in spatiotemporal domain that a complex motion is taking place. A widely used interest point detector called Space-Time Interest Points (STIP) is proposed in Laptev [31]. The STIP is a generalization of the two dimensional Harris corner detector [21] to the three dimensional video. STIP is triggered at spatio-temporal points that indicate a change in the velocity of a moving corner in the video. For example in a hand waving action interest points are detected at the points that movement direction of hand is changed. Dollar et al. [11] proposed a different interest point detector using the Gabor Filter for

temporal domain of image sequences. Their feature is sensitive to periodic movements or any other complex motion. Rapantzikos et al. [48] proposed to use energy of discrete wavelet transformation on each spatil and temporal domain to find the salient cuboids.

Another group of local feature detectors are inspired from biologicall visual system. In Jhuang et al. [26] a hierarchical feed-forward architecture is proposed for action recognition. At their first level, a set of spatio-temporal filters are applied to the sequence. Then, templates are matched to the high scores of the first level, and, at upper level score maximization and template matching is performed again on the out put the previous level. The Final descriptor is formed using the maximum score of the last level which is a spatio-temporal position invariant feature.

Many sparse models use the Bag of Words (BOW) representation as the final descriptor of an action. In this representation a feature such as image gradient or optical flow is used to describe the image cube around the interest points. Then, a vocabulary of visual words is constructed by clustering large number of samples of interest points, and, the final descriptor histogram of occurrence of each word in the sequence. In contrast to global features, the bag of words representation is translation invariant, however it discards all global information about arrangement of visual words in spatial and temporal domains which is not desired in the case that configuration of those visual words is discriminative for action recognition. In order to overcome this problem, many hybrid features are proposed which is covered in the following section.

2.1.3 Hybrid Features

In order to have both configuration information of global features and translation invariance of local features several hybrid features are proposed. These features mainly take two approaches. One approach is to divide the spatio-temporal space of action to bins and calculate the translation invariant features inside the bins. Their final descriptor is formed by concatenation of features of all bins which is robust to small displacement of signal as long as the translation keeps the signal mainly inside their bins. The second approach is to detect local features first and capture the spatio temporal configuration of local points in the feature. This approach is invariant to larger displacements, however, with large number of interest points, the configuration extraction may become intractable.

As an example of the first approach, Yamato et al. [67] binned the silhouette image of each frame to non-overlapping cells and they count number of forground pixels in a cell as their feature. Moreover, in Laptev et al. [32] temporal and spatial domains are divided into non-overlapping segments and bag of words features are computed in each segment.

Among hybrid features proposed using first approach, the Histogram of Oriented Gradients (HOG) is a popular feature which was introduced by Dalal and Triggs [8] first for human detection. But, later it was used for human action recognition. The process of HOG feature extraction starts with Gamma normalization of input image, followed by a gradient computation in vertical and horizontal directions. Then, the gradient vectors vote into a histogram over oriented bins with different Gaussian weights centered at the pixel in the center of non-overlapping cells. For illumination invariance purpose, the histogram in each cell is normalized with respect to all four neighboring cells, and the final descriptor is made by concatenation of the histogram of oriented gradients for all cells with their different normalizations. Although original HOG is proposed for 2D image, Klaser et al. [28] extended 2D HOG feature to 3D video for action recognition. The HOG feature carries cues of human pose and in contrast to the optical flow feature, it has information on still parts of human which can not be captured in motion feature. However, it may carry some noise due to a cluttered background.

Kovashka and Grauman [29] take the second approach and propose a hierarchical representation to capture the spatial and temporal configuration of visual words. They start with interest points as zero level, and in first level they bin the 3D space around each interest point and the histogram of the closest points are calculated in each bin. Then, new words are created by clustering the histograms at level one. In further layers, histograms of words are created around the words of previous level. They show their method can model the spatial and temporal structure of local interest points in actions at different scales. Gilbert et al. [19] also proposed similar hierarchical technique to encapsulate the structure of local interest point. However, instead of clustering patterns of previous level, they mine the most frequent patterns, and use them as words for upper layer.

2.2 Action Classification Models

Having a representation of action, we need a classification model to recognize the action in an unknown sequence. In this section we review previous models proposed for action recognition. We divide them to template matching models, generative models, discriminative models, and key pose models.

2.2.1 Template Matching Models

Template models capture the temporal dynamic of an action implicitly in the templates, and classification is typically done using Nearest neighbor method that assign a test sample to the class of the

nearest training sample in the space of features. Gorelick et al. [20] used Euclidean distance in the space of their global features to find the nearest neighbor, and Bobick and Davis [4] used Mahalanbis distance which is a normalized Euclidean distance based on variance of each dimension.

In several works, frame level distance measure is used to match the whole sequence. In Efros et al. [12] a template is defined as a sequence of human centric images described by 4-channel blurred motion feature. In order to account for shift in a periodic action, the correlation of a sequence and a template for all different starting frames is calculated using sum of frame-to-frame correlation measure. Then, temporal smoothing is used to overcome difference in speed. In Lin et al. [34] each sequence is matched to a set of predefined shape and motion prototypes in a frame-based manner. Then, action is recognized using a dynamic prototype matching to the closest sequence in training data.

Overall, template matching models are capable of classification of short actions that have simple temporal dynamic. Classification of long complicated activities using template is infeasible due to limitation in total number of templates that one can generate. In addition, complex activities may have irrelevant visual information that can not be handled using templates. In order to over come this problem, several richer models are proposed that are discussed in the following sections.

2.2.2 Generative Models

Generative models are probabilistic models that specifies the joint probability distribution over features and action class. Sampling from these models will produce samples of classes, and classification can be done by calculating conditional probability of a class given an observation. A popular generative model used in action classification is the Hidden Markov Model (HMM). A HMM models an action as a sequence of hidden states which makes it suitable for modeling action dynamics. State transition and observation probabilities are considered in HMM. There are two typical assumptions in HMM to keep the parameter training tractable. The Markov assumption forces that a state transition is conditioned on only previous state, not on other states. Second, an observation is only dependent to the current state. In Yamato et al. [67], a Hidden Markov Model is used to recognize the action. They consider a HMM for each tennis stroke class, and they represent the observation using quantized silhouette over super pixels. The parameters are learned using Maximum Likelihood for each class and in test stage, a sequence is assigned to a class that has highest probability given that class's model. In Ramanan and Forsyth [46] 2D body joints are matched to an annotated 3D motion sequence using HMM. Another HMM is used to infere the class of frame using annotation

of 3D motion data. Lv and Nevatia [37] also try to recognize action using human motion capture data. They define several features produced from different combination of joint locations, and learn a HMM for each feature set. The probability of each HMM is then fed to an AdaBoost classifier.

In Xiang and Gong [64] Dynamic Bayesian Network is used to model complex activities containing multiple humans and objects in cluttered scenes. In this work, an activity is represented using atomic scene events and their temporal and causal correlation are modeled in a Dynamic Bayesian Network whose parameters and structure are learned by structured Expectation-Maximization (EM).

2.2.3 Discriminative Models

In contrast to generative models, discriminative models learn the conditional probability of a class given a sample point instead of modeling the hard problem of sample generation. In this class, conditional random field (CRF) is typically used to model dependency of different variables of a model in a discriminative graphical model. Sminchisescu et al. [53] used a chain CRF to recognize per-frame action in sequence of images. Wang and Mori [56] used a hidden conditional random field to combine local patch features with global feature for per-frame action recognition. Their hidden variables are part labels connected to the global label and patches in the frame. They learn parameters using gradient decent on log-likelihood.

Classifiers are another group of discriminative method which attempt to learn the boundary between classes typically in non-probabilistic manner. Support Vector Machines (SVM) learn a hyperplane in the feature space that has maximum margin to the closest samples of each class. Similar to many other classification tasks, SVM are also popular in action recognition, and in the simplest way it can be used to classify the features extracted from the sequence; such as in [26, 32, 11]. Boosting is also another discriminative technique that forms a classifier from several weak classifiers. Fathi and Mori [14] proposed a two level AdaBoosting for action recognition on a human centric motion feature. They first train AdaBoost classifiers in small regions of a bounding box, then the response of those mid-level weak classifiers are fed to another classifier.

Several discriminative methods train the parameters of their models in a max margin criterion which tries to maximize the gap between score of ground truth label and any other hypothetical labeling. Wang and Mori [57] proposed max margin hidden CRF to learn parameters of the same model in [56]. They showed the new training has better accuracy then previous log-likelihood based parameter training. Shi et al. [51] proposes a semi-Markov model to segment a sequence temporally and label segments with an action class. In this model, a set of features is defined for each segment,

as well as features for the boundary and transition between the actions of neighboring segments. Then, features are scored using a linear function and the parameters of the model are learned in a discriminative max margin framework. Niebles et al. [42] develop a similar discriminative model to recognize activity by defining atomic-action classifiers. Both [51] and [42] are similar to our model, but for recognizing single person activities, without hard temporal ordering, without the non-parametric exemplar matching.

2.2.4 Key pose Models

Action recognition models usually consider whole temporal domain in recognition. On the contrary, key pose models attempt to localize important poses of human called "key poses" to recognize action. The term "Key Pose" was originally initiated in animation to describe the critical configuration of a character or an object which carries the extreme points in its motion or expression. In the action recognition context, key poses are used for human poses that are discriminative for action recognition purpose.

In early attempts, key pose is used in Sullivan and Carlsson [54] for action recognition and human tracking. They localize key poses in a video sequence using shape matching to a particular pre-defined poses, and then, the manual part labels are used to re-initialize the human body tracker. However, the proposed key pose model is capable of action recognition in a frame, and can not be used for complex activity recognition.

Lv and Nevatia [38] also proposed a key pose based model for view-invariant human action recognition. They represent each action as sequence of synthetic 2D human poses rendered from different view points. Constraints on transition between key poses are represented using a state diagram called "Action Net" which is constructed based on the order of key poses of an action. Given the input, the silhouette of the input frames are matched to the key poses using a dynamic program, Viterbi Algorithm. The per-frame action recognition is done based on the annotation of matched key poses. Their method is similar to our method. However, we learn the order of discriminative key poses for each action, but, they fixed the key pose of an action and their order at the beginning.However we assume that key poses can be shared between actions and we learned the discriminative key poses of an action. We also match the key pose to subset of frames and we consider interaction instead of action.

2.3 Interaction Models

Much work focuses on atomic action recognition of individuals instead of interaction. Interactions were considered by Intille and Bobick [25] who used probabilistic techniques for recognizing hand-specified structured activities such as American football plays. They define low-level action primitives and temporal relationship between them, and they use Baysian Networks to combine uncertainty of feature evidence and temporal information. Medioni et al. [40] developed a system for recognizing events from aerial video surveillance data, for instance interactions between vehicles and road checkpoints. They track the objects in the scene, and they extract their location and speed features with respect to the checkpoint. A set of recursive scenarios is defined, and the likelihood of each primitive scenario is calculated using the spatial and temporal features. Then, they are used to calculate the likelihood of higher level scenarios.

In our experiments we use the UT-Interaction (SDHA) dataset introduced by Ryoo and Aggarwal [50]. Ryoo and Aggarwal develop a matching kernel that considers spatial and temporal relations between space-time interest points to detect and localize non-parametric activities. However, the proposed method relies on spatial and temporal interest point detector which can have very sparse output on a non-periodic action. Yao et al. [68] use dense local patches in a Hough transform voting scheme. They construct random trees to learn mapping from local feature space to spatial-temporal-action Hough space. In training, discriminative codebooks are formed in leaves of the trees that can be used for voting of action location based on local features in a probabilistic manner. Yu et al. [69] develop an efficient algorithm using semantic texton forests with a pyramidal version of Ryoo and Aggarwal's matching kernel. These methods are based on local patches and have less explicit modeling of the presence of individuals than our method, and our method obtains higher accuracy on the UT-Interaction dataset.

Chapter 3

Modeling Human Interaction

Our goal in this thesis is to recognize human interactions in videos. The interactions we consider are activities such as pushing, handshaking or hugging that involve two people interacting.

We will model these interactions by a sequence of key poses. For example, as shown in Fig. 1.1 a common scenario for pushing is the following: one person steps forward, raises his hands, and pushes the other person while he takes a defensive pose, steps backward, and falls back at the end. Similarly, we can decompose other interaction scenarios to sequences of key poses. Observing them and their chronological order can be used to recognize an interaction.

Given an input video and a putative interaction, four things are unknown:

1. **Who** is involved in the interaction? More specifically, which person is taking which role in the interaction – many asymmetric interactions, such as pushing or kicking, have distinct "subject" and "object" roles which can have different key pose sequence.

2. **When** do the key poses occur? We model each interaction by a fixed-length sequence of key poses, but we do not know a priori when these key poses occur in an input video.

3. **How** are the key poses executed? There is variation in appearance for the key poses of an interaction – e.g. is the push with one hand, two hands, a forceful push, or a weak push.

4. **Where** are the people when the key poses occur? The spatial arrangement of these key poses is important – interactions such as pushing or embracing have stereotypical relative distances between the people involved.

These are unknown and, while inferring them is useful, are not our direct goal of recognizing interactions. Hence, we treat them as latent variables in a novel constrained variant of a structured latent variable model.

Following the standard notation in structured latent variable models, we now provide a formulation of our model. Let $x \in \mathcal{X}$ be a video sequence that consists of people performing an interaction $y \in \mathcal{Y}$ where $\mathcal{Y}$ is the finite set of interactions. Given a set of video and interaction label pairs, our task in training is to learn a scoring function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \Re$ over these pairs. Following the usual latent variable formulation, we will assume F maximizes a model G that includes the latent variables H: $F(x, y) = \max_H G(x, y, \boldsymbol{H})$.

In our work, the latent variables $\boldsymbol{H}$ answer the four questions above. Namely, $\boldsymbol{H} = [b, t, e, p]$, where:

1. b specifies who takes which role in the interaction. In this work we assume we are provided roughly correct tracks of the people in a scene, and b denotes which person is the subject and object of the interaction. More generally, one could build b from tracklets, or infer it while tracking.

2. t specifies when the key poses occur. Our interaction model has a fixed number of key poses (e.g. 5 in experiments). t specifies when in the (much longer) input video x these key poses occur. This key pose sequence will be constrained to be in chronological order.

3. e specifies how the key poses are executed. We use an exemplar-based representation in which e specifies which discrete type of execution of a key pose is present in a video. Essentially, this is similar to an aspect or mixture model to account for key pose variation.

4. p specifies the spatial locations in the video frames for the key poses. As with b, we will rely on a tracker to assist with this information, allowing small shifts in position from tracker output to account for tracker error.

In the following sections we provide more precise details on these latent variables and the scoring function G. For ease of exposition, we start with a single subject key pose sequence model (Sec. 3.1), followed by a model for interactions (Sec. 3.2). We do not assume the key poses are provided as training data, and instead aim to automatically discover them. Algorithms for this learning, and associated inference, are in Chapter. 4.

3.1 Single Subject Key Pose Sequence Model

We start by describing a model of videos of a single person performing an activity. Given a set of such videos, we want to find a set of key poses in these sequences and use them to model the activity class. Key poses are meant to be important, infrequent actions; much of each video can consist of highly variable human action that can be misleading when attempting to build an activity model. Considering our *pushing* example, there are poses such as standing or walking at the beginning or

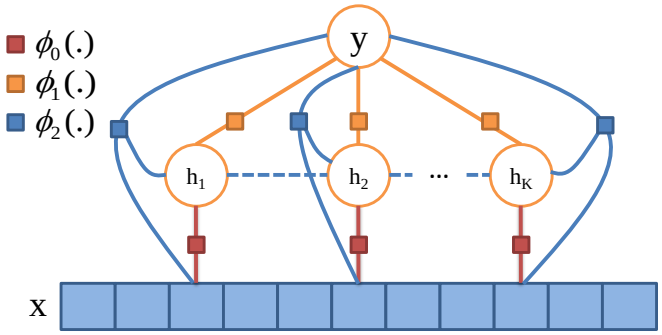


Figure 3.1: Graphical depiction of our model for single subject key pose sequence matching. The lower layer x is the observed sequence of frames, and the middle layer $\boldsymbol{h}$ is the key pose sequence layer and the top layer y is the activity label. Edges with boxes denote factors in our model. Dashed lines represent temporal constraints between key poses.

the end of the video that are variable and not discriminative. Further, each of the key poses will have variation in appearance. Finally, the spatial arrangement of these key poses is important (particularly for interactions), so the model will also include what spatial location in a video frame contains the key pose.

An instantiation of a single subject key pose model in a video sequence will consist of three things: **when** do the key poses occur, **how** is each key pose executed, and **where** in space do they occur. We assume that we are given a rough track of the subject, via human detection and tracking algorithms. We represent each key pose in a sequence by a triple $\boldsymbol{h} = [e, t, p]$. Variables t and p are its spatio-temporal locations, with p restricted to locations near the tracker output. The variable $e = 1, 2, \ldots, |\mathcal{E}|$ denotes which appearance variant of the key pose is taking place at time t and location p where $\mathcal{E}$ is a discrete set of exemplars used as a representation of the appearance of key poses – for instance, the different types of pushes noted above would each be represented by its own element of $\mathcal{E}$. As noted above, a model contains multiple key poses in sequence, and we denote the K key poses of a sequence by $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, .., \boldsymbol{h}_K]$, where each $\boldsymbol{h}_i$ is a triple $[e_i, t_i, p_i]$. Our model also has a constraint on the temporal component of the key poses $\boldsymbol{H}$. The key poses should be matched to the input sequence in chronological order, hence $t_i < t_j$ if $i < j$. This hard constraint will be enforced in inference via an efficient algorithm.

We now describe the scoring function $G(x, y, \boldsymbol{H})$ for a single subject model. A graphical depiction of our model is shown in Fig. 3.1. Factors in this model include terms measuring compatibility between input sequences and instantiations of key poses, between key poses and activity label, and

among the three. Based on this model, a sequence of key poses H is scored for the input x and the label y by $G(x, y, H) = \boldsymbol{\omega}^T \boldsymbol{\Phi}(x, y, \boldsymbol{H})$ which is a linear function on $\boldsymbol{\omega}$, the parameters of the model. We formulate the scoring function as:

$$\boldsymbol{\omega}^T \boldsymbol{\Phi}(x, y, \boldsymbol{H}) = \sum_{i=1}^{K} \boldsymbol{\alpha}^T \boldsymbol{\phi}_0(x, t_i, e_i, p_i) + \sum_{i=1}^{K} \boldsymbol{\beta}_i{}^T \boldsymbol{\phi}_1(y, e_i) + \sum_{i=1}^{K} \boldsymbol{\gamma}^T \boldsymbol{\phi}_2(x, y, t_i, p_i)$$

where $\boldsymbol{\phi}_0(\cdot)$, $\boldsymbol{\phi}_1(\cdot)$ and $\boldsymbol{\phi}_2(\cdot)$ are the potential functions defined on the links which will be described below. $\boldsymbol{\alpha}$, $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_K]$ and $\boldsymbol{\gamma}$ are the parameters of the model which are grouped in $\boldsymbol{\omega} = [\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}]$.

Exemplar Matching Link: $\boldsymbol{\alpha}^T \boldsymbol{\phi}_0(x, t_i, r_i, p_i)$ measures the compatibility between key pose with appearance of the ${e_i}^{th}$ exemplar and the image evidence of one track at time t_i and location p_i. It is formulated as:

$$\boldsymbol{\alpha}^T \boldsymbol{\phi}_0(x, t_i, e_i, p_i) = \sum_{e=1}^{|\mathcal{E}|} \boldsymbol{\alpha}_e{}^T \boldsymbol{D}(f(x, t_i, p_i), g(e_i)) \mathbb{1}_{\{e_i = e\}} \tag{3.1}$$

where $f(x, t, p)$ computes features for sequence x at the location p and time t. Similar to $f(\cdot)$, $g(\cdot)$ calculates the features for exemplars. The details of these features and distance measure $\boldsymbol{D}$ are described in Sec. 3.3. $\mathbb{1}$ is an indicator function selecting for the weight vector associated with the ${e_i}^{th}$ exemplar.

Activity-Key Pose Link: $\boldsymbol{\beta}_i{}^T \boldsymbol{\phi}_1(y, e_i)$ models the compatibility between activity y and exemplar e_i as the i^{th} key pose. It is a scalar for each activity and exemplar, and if it is high it means that particular type of exemplar is strongly associated with the activity label y:

$$\boldsymbol{\beta}_i{}^T \boldsymbol{\phi}_1(y, e_i) = \sum_{a \in \mathcal{Y}} \sum_{e=1}^{|\mathcal{E}|} \beta_{iae} \mathbb{1}_{\{y=a\}} \mathbb{1}_{\{e_i = e\}} \tag{3.2}$$

The activity-key pose term is indexed on key poses $\boldsymbol{\beta}_i$, and it means that an exemplar in a sequence of key poses may have different compatibility with the activity at different times. This models the fact that key poses have a particular order for each activity. For example bending starts with a standing pose, continues with bending until the subject reaches ground, and ends with a standing pose.

Direct Root Model: $\boldsymbol{\gamma}^T \boldsymbol{\phi}_2(x, y, t_i, p_i)$ measures the compatibility of global features extracted from x at time t_i and location p_i and activity class label y. This directly models the features of the input to the activity class label, without exemplars. It is parametrized as:

$$\boldsymbol{\gamma}^T \boldsymbol{\phi}_2(x, y, t_i, p_i) = \sum_{a \in \mathcal{Y}} \boldsymbol{\gamma}_a{}^T f(x, t_i, p_i) \mathbb{1}_{\{y=a\}} \tag{3.3}$$

3.2 Interaction Key Pose Sequence Model

Our goal is to recognize human interactions in a video. There are several ways to extend our model in Sec. 3.1 to capture interactions. The easiest way would be to learn parameters of the model for each individual of the interaction, and use them to score each participant separately. The problem with this method is that it cannot capture any information about interaction. For asymmetric activities such as kicking, pushing, or punching the model parameters should be different for each participant. The participants of these interactions are the subject of activity, the one who does the activity, and the object of the activity, the one to whom the activity occurs. The subject and object in an interaction may have different key poses. For example, in pushing the key poses for the subject are stepping forward, putting hands in front, and shoving actions. However, for the object who is pushed the key poses are a defensive pose, stepping backward, and falling back because of the push. So, we expect to see a different group of key poses for the subject and object trajectories. Further, as noted above the relative spatial position of the subject and object of an interaction is an important cue for recognition.

We modify our single subject model to incorporate this information: **who** is playing which role, and additional cues about **where** these people are. The model is depicted in Fig. 1.1. We assume we are given the rough trajectories of a potential subject and object of an interaction, and similar to our model in Fig. 3.1 we match key poses to each trajectory. However, we model the asymmetry in the interaction, and we define two different compatibilities between key poses and activity for subject and object tracks. In other words, in Eq. 3.1, we use $\boldsymbol{\beta}^s$ and $\boldsymbol{\beta}^o$ for subject and object trajectories. Further, we model the spatial distance of the key poses by an additional term in the scoring function, denoted by $\boldsymbol{\theta}$. The intuition is that the key poses of an activity occur at common spatial distances from each other. For example in hugging subjects open their hands at a certain distance and then embrace at very nearby spatial locations afterwards.

Let x be a video that contains two people interacting. In our interaction model the latent variables are $\boldsymbol{H} = [\boldsymbol{H}^1, \boldsymbol{H}^2, \boldsymbol{b}]$. $\boldsymbol{H}^1$ and $\boldsymbol{H}^2$ are the key pose configuration for each person. The variable $\boldsymbol{b} = (b^1, b^2)$ selects which person trajectories take the subject and object roles in the interaction. We assume a tracker provides the rough trajectories of the people in the video. We use $l(x, t, b^1)$ to denote the location of subject actor in sequence x at time t (same as $l(x, t, b^2)$ for object trajectory). Given a sequence, a latent variable configuration, and a class label, we calculate the score of each participant, and include the score of the spatial distance link. The scoring function for the interaction

model is formulated as:

$$L(x, y, \boldsymbol{H}; \boldsymbol{\omega}) = G(x, y, \boldsymbol{H}^1, b^1; \boldsymbol{\omega}^s) + G(x, y, \boldsymbol{H}^2, b^2; \boldsymbol{\omega}^o) + Q(x, y, \boldsymbol{H}; \boldsymbol{\mu}) \tag{3.4}$$

where we make explicit the dependence of G on different parameter subsets $\boldsymbol{\omega}^s = [\boldsymbol{\alpha}, \boldsymbol{\beta}^s, \boldsymbol{\gamma}]$, $\boldsymbol{\omega}^o = [\boldsymbol{\alpha}, \boldsymbol{\beta}^o, \boldsymbol{\gamma}]$ for different trajectories. The parameter b is used to select tracks (not considered in the single-subject model). Note that $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ are assumed to be identical for the subject's and object's trajectories, while $\boldsymbol{\beta}$, the compatibility of key poses and activity is different. $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ...\boldsymbol{\mu}_K]$, $\boldsymbol{\mu}_i$ are the parameters that measure the compatibility between activity y and binned distance between tracks at the time of the i^{th} key pose. $Q(x, y, \boldsymbol{H}; \boldsymbol{\mu})$ measures the relative distance of two tracks at the time of the key poses and is formulated as:

$$Q(x, y, \boldsymbol{H}; \boldsymbol{\mu}) = \sum_{i=1}^{K} \boldsymbol{\mu}_i{}^T \boldsymbol{\theta}(x, y, t_i^1, b) + \sum_{i=1}^{K} \boldsymbol{\mu}_i{}^T \boldsymbol{\theta}(x, y, t_i^2, b) \tag{3.5}$$

where

$$\boldsymbol{\mu}_i{}^T \boldsymbol{\theta}(x, y, t_i^j, b) = \sum_{a \in \mathcal{Y}} \boldsymbol{\mu}_{ia}{}^T \boldsymbol{bin}(\|l(x, t_i^j, b^1) - l(x, t_i^j, b^2)\|_2) \mathbb{1}_{\{y=a\}} \tag{3.6}$$

i.e., the distance between the tracks at the time of the i^{th} key pose in j^{th} trajectory. The function $\boldsymbol{bin}(\cdot)$ discretizes this distance. To summarize, the full set of parameters is $\boldsymbol{\omega} = [\boldsymbol{\beta}^s, \boldsymbol{\beta}^o, \boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\mu}]$. Note that the scoring function L is a linear function of $\boldsymbol{\omega}$.

3.3 Features

In order to match key poses to the input sequence, we choose the histogram of oriented gradients (HOG) and histogram of optical flow (HOF) features to capture shape and movement of human. For the HOG feature we used implementation of Felzenszwalb et al. [15] which is similar and efficient implementation of the original HOG feature proposed by Dalal and Triggs [8]. They calculate the gradient of an image in vertical and horizontal directions, and they bin them to different oriented bins. The histograms in each cell is normalized with respect to the L2-norm of histograms at four neighboring cell one by one. Then the final feature is projected to the 13 directions proposed in [15] which is similar to principal direction obtained by Principal Component Analysis(PCA). We calculate the HOF feature in exactly similar manner, except we use motion vector between two frames calculated by Lucas and Kanade [36] optical flow method instead of gradient vector. Thus,

we represent each frame using a concatenation of HOG and HOF features of 8×8 non-overlapping cells organized on a grid inside a bounding box around the subject.

In Eq. 3.1 we use a function $\boldsymbol{D}(\cdot, \cdot)$ to measure the distance of two bounding boxes. The inputs to $\boldsymbol{D}$ are HOG and HOF features of the two bounding boxes and the output is a vector with i^{th} component storing normalized Euclidean distance between HOG and HOF features at the i^{th} cell. In other words, $\boldsymbol{D}$ calculates the Euclidean distance of features at corresponding cells provided by HOG and HOF.

3.4 Selecting Exemplars

Our model requires an exemplar set consisting of instantiations of various discriminative key poses $\mathcal{E}$. Given the tracks of subjects in training sequences we have access to thousands of samples of cropped images of human subjects. A clustering algorithm such as k-means could be used to extract various human poses from cropped bounding boxes. But naive clustering methods focus on common rather than discriminative poses. In order to get varied, discriminative key poses, we trained a multiclass linear SVM classifier using LIBLINEAR [13] on top of all cropped bounding boxes from different activities. This classifier is used to score the training samples as a measure of how discriminative a sample is. Next, we clustered the samples with the highest scores using k-means. The k-means centers are mean feature of each class which is virtual poses and do not exist as training samples. We use the nearest samples of the training set to the centers provided by k-means as a set of key human pose candidates. This heuristic procedure is efficient and effective in our experiments; though other supervised clustering techniques could also be used (e.g. Lazebnik and Raginsky [33]).

Chapter 4

Key Pose Detection and Parameter Learning

Given the training set, we need to learn the parameters of the model to be able to find the key poses in a test sequence and recognize its activity class. The learning algorithm we use requires the inference procedure, so we first describe the inference procedure to find the key poses for a sequence, and then explain how we train the parameters of the model.

4.1 Inference

Given a video sequence x, model parameters $\boldsymbol{\omega}$, and a hypothesized activity label y, we score the sequence by finding the best sequence of key poses. The activity label for a sequence is the y that maximizes this score. We assume we are given a tracker that produces person trajectories, but we do not know which of these people takes which role in the activity. We define the scoring function $E(x, y)$:

$$E(x, y; \boldsymbol{\omega}) = \max_{(b^1,b^2)\in S} \max_{(\boldsymbol{H}^1,\boldsymbol{H}^2)\in\mathcal{H}_1\times\mathcal{H}_2} L(x, y, \boldsymbol{H}; \boldsymbol{\omega}), \tag{4.1}$$

with L being the dual-trajectory scoring function defined in Eq. 3.4. b^1 and b^2 select which person trajectories take the subject and object roles in the interaction. S is the set of all ordered pairs of actors in sequence x. In the case with many actors in a sequence, i.e. TRECVID experiment we limit S to pairs of temporally overlapping trajectories which are close spatially. Recall from Sec. 3.1 that key pose sequences are constrained by a chronological ordering. $\mathcal{H}_1$ and $\mathcal{H}_2$ are the sets of chronologically valid keypose sequences for the trajectories corresponding to people b^1 and

b^2.

Note that the interaction distance term Q in Eq. 3.5 measures the distance of a trajectory from the other one at time point of its key pose for both trajectories which is decomposable. Hence, the maximization in Eq. 4.1 can be decomposed into maximization for each trajectory. So, we can write

$$E(x,y;\boldsymbol{\omega}) = \max_{(b_1,b_2)} \Bigg\{ \underbrace{\max_{\boldsymbol{H}^1 \in \mathcal{H}_1} \left\{ G(x,y,\boldsymbol{H}^1,b^1;\boldsymbol{\omega}^s) + \sum_{i=1}^{K} {\boldsymbol{\mu}_i}^T \boldsymbol{\theta}(x,y,t_i^1,b) \right\}}_{\text{subject trajectory}} + \underbrace{\max_{\boldsymbol{H}^2 \in \mathcal{H}_2} \left\{ G(x,y,\boldsymbol{H}^2,b^2;\boldsymbol{\omega}^o) + \sum_{i=1}^{K} {\boldsymbol{\mu}_i}^T \boldsymbol{\theta}(x,y,t_i^2,b) \right\}}_{\text{object trajectory}} \Bigg\} \quad (4.2)$$

The score maximization for each trajectory consists of finding K key poses, $\boldsymbol{h}_i = (e_i, t_i, p_i)\,, \forall i \in 1, ..., K$ that match to the sequence. However, our model has a chronological ordering constraint on the key poses found in the input sequence, which states $t_1 < t_2 < \cdots < t_K$. The exemplar and spatial perturbation of the key pose are free from this constraint, so we can maximize the partial score of our model for the i^{th} key pose at frame t over possible exemplars and spatial perturbation:

$$A_i^t = \max_{e_i,p_i} \left\{ \boldsymbol{\alpha}^T \boldsymbol{\phi}_0(x,t,e_i,p_i) + {\boldsymbol{\beta}_i}^T \boldsymbol{\phi}_1(y,e_i) + \boldsymbol{\gamma}^T \boldsymbol{\phi}_2(x,y,t,p_i) + {\boldsymbol{\mu}_i}^T \boldsymbol{\theta}(x,y,t,b) \right\} \quad (4.3)$$

where $t = 1, 2, \cdots, T$, and T is the number of frames in x. Next, considering the constraint, we can rewrite the score maximization of a trajectory in Eq. 4.2 as:

$$\begin{aligned} \max \quad & \sum_{i=1}^{K} A_i^{t_i} \\ \text{s.t.} \quad & t_i < t_{i+1} \;\; \forall i = 1, 2, \cdots, K-1 \end{aligned} \quad (4.4)$$

We use an efficient dynamic programming algorithm to solve this maximization. We define M_j^τ as the best score using j elements of A until the τ^{th} frame:

$$\begin{aligned} M_j^\tau &= \max \sum_{i=1}^{j} A_i^{t_i} \\ \text{s.t.} \quad & 1 \le t_i < t_{i+1} \le \tau \;\; \forall\, i = 1, 2, \cdots, j-1 \end{aligned} \quad (4.5)$$

We can write M_j^τ as a recursive function:

$$\begin{aligned} M_j^\tau &= \max\{M_{j-1}^{\tau-1} + A_j^\tau, M_j^{\tau-1}\} && 1 < j \le K, j < \tau \le T \\ M_j^j &= M_{j-1}^{j-1} + A_j^j && 1 < j \le K \\ M_1^\tau &= \max\{A_1^1, A_1^2, \cdots, A_1^\tau\} && 1 \le \tau \le T \end{aligned} \quad (4.6)$$

The optimal solution of Eq. 4.5 is M_K^T, and can be calculated in time $O(KT)$, the number of keyposes multiplied by the number of frames in the video sequence. In summary, to solve the maximization in Eq. 4.2, given a (b^1, b^2) pair, we maximize the score for each track specified with b^1 and b^2 using dynamic program in Eq. 4.6, and then we maximize this score over all possible pairs of (b^1, b^2) (elements in S).

4.2 Learning

We use $y^* = \arg\max_y E(x, y; \boldsymbol{\omega})$ as the predicted label of x. Given $\{(x^1, y^1), (x^2, y^2), ..., (x^n, y^n)\}$, the set of training data, we aim to find parameters that score x^i and y^i higher than other activity types. Similar to Felzenszwalb et al. [15] and Wang and Mori [57] we formulate the training criteria in the Max-Margin framework. We set $\boldsymbol{\omega}$ by:

$$\min_{\boldsymbol{\omega}, \xi^i} \frac{\lambda}{2}\|\boldsymbol{\omega}\|^2 + \sum_i \xi^i \tag{4.7}$$
$$\text{s.t.} \quad E(x^i, y^i; \boldsymbol{\omega}) - E(x^i, y; \boldsymbol{\omega}) > \Delta(y^i, y) - \xi^i \;\; \forall i, \forall y \in \mathcal{Y}$$

where λ is a tradeoff constant and $\Delta(y^i, y)$ is 0-1 loss.

The constraint in Eq. 4.7 forces the score of the true labeling for each training sequence to be higher than the best score for an incorrect hypothesized label. The optimization problem in Eq. 4.7 is a non-convex optimization problem and we use the non-convex extension of the cutting plane algorithm using NRBM [10] to learn the parameters.

4.3 Initialization

Parameter initialization can be crucial in learning latent variable models since they can converge to a local optimum. We use some heuristic to initialize the parameters. In order to initialize $\boldsymbol{\beta}$, which affects the valid key pose sequence, each trajectory in class a is divided into K (number of key poses) equal length, non-overlapping temporal segments. Each frame of a trajectory in the i^{th} segment is matched to its nearest exemplar in $\mathcal{E}$, and β_{iae} is set to the frequency of matching exemplar e to the i^{th} segment of sequence in class a. Since, in training we only have involving people, not their rules, we set both β^s and β^o to the same before-mentioned β. The initial α and μ are set to a constant vector normalized based on the number of their components. In order to speedup the training procedure we learn the direct root model separately by a multi-class linear SVM using LIBLINEAR [13]. Next, we augment the SVM score with a constant for bias term and

we change γ to a two-component weighting vector for each action which is set to a constant vector in the initialization.

Chapter 5

Experiment Details and Results

We consider two datasets to gauge our model's effectiveness in classifying human interactions. First, we test our model on the UT-Interaction dataset [50], a publicly available benchmark with comparative results. Second, we construct a dataset for recognizing *embrace* interactions by selecting a subset of the TRECVID 2008 Surveillance Event Detection challenge [52] and demonstrate our model on a non-choreographed dataset. Fig. 5.2 shows sample frames from the UT-Interaction and TRECVID embrace datasets.

5.1 UT-Interaction Dataset

The UT-Interaction dataset contains videos of 6 classes of human-human interactions: *shake hands*, *hug*, *kick*, *point*, *punch*, and *push*. There are 20 video sequences in total. Each video contains at least one execution per interaction, providing 8 executions of human activities per video on average. The dataset is divided into two sets. Set 1 is recorded in a parking lot with a stationary background and set 2 is recorded on a lawn with slight background movement and camera jitter. We follow the experimental setting of the classification task described in the High-level Human Interaction Recognition Challenge [50] – bounding boxes are used as input and the performance of our model is evaluated using leave-one-out cross validation on each set. Note that no additional information is used – in particular roles in the interaction (b variables) are inferred both in learning and test time.[1]

[1]For the *point* activity, the ground truth in the UT-Interaction dataset only contains the person performing the activity without the other one being pointed at. We search horizontally for a person nearest to the one performing the point activity and include him as the other part of the activity.

5.1.1 Implementation Details

The bounding boxes provided as input contain the two humans performing an interaction, not tracks of individuals. We employ a pedestrian detector [8] to obtain initial positions of the people in the first frame of every video clip. We select a pair of detections with the minimum horizontal distance out of the three highest scoring detections, then run a tracker [49] to find trajectories of two individuals interacting with each other in the subsequent frames. To handle tracker jitter, we allow key pose positions at small spatial perturbations around the tracker output. We use a 20 pixel step size and allow up to 1 step horizontally, a 15 pixel step size and allow up to 1 step vertically to locate p, the position of key pose in the track. Considering camera zoom in set 1, we also perform multi-scale search at 2 scales. In other words, we assume we have another latent variable for scale which can have two different zoom rates for set 1. In inference, we maximize the score over this hidden variable similarly to other hidden variables.

5.1.2 Results

Confusion matrices of the two sets in the UT-Interaction dataset are shown in Fig. 5.1. The figure shows some confusion between the activities *push* and *punch* on set 2. This is consistent with the fact that pushing and punching are similar in both appearance and motion. Comparisons with other approaches are summarized in Table 5.1. A direct comparison is possible to the methods by Yao et al. [68] and Yu et al. [69]. Our method clearly outperforms the other methods.

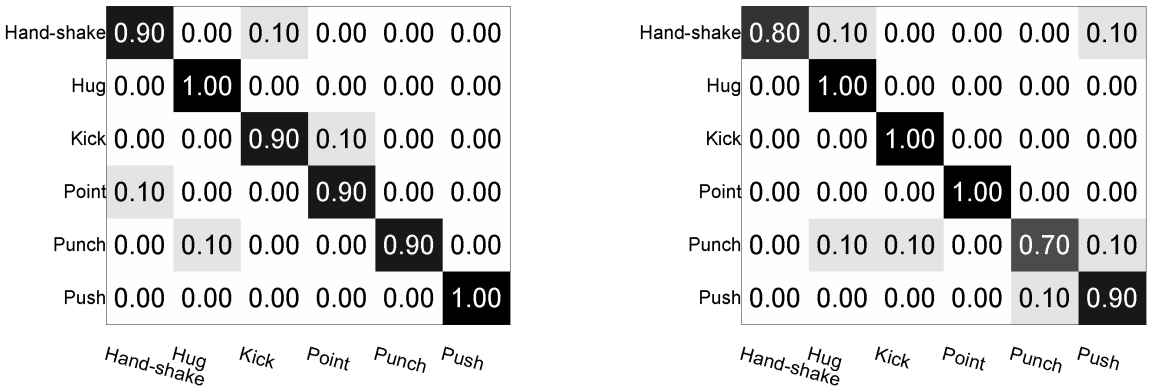
Table 5.1: Per-clip classification accuracy on UT-interaction dataset.

Method	Set 1	Set 2	Avg
Our method	**0.93**	**0.90**	**0.92**
Yu et al. [69]	N/A	N/A	0.83
Yao et al. [68]	0.88	0.80	0.84

5.1.3 Visualization of Model Weights

In this section we provide visualization of portions of our model to understand what it has learned.

We visualize the exemplar matching model, which is patch-based weights, to demonstrate that our model is able to localize key poses in the trajectory and fire on discriminative patches for pose. Figure 5.2 shows our exemplar-matching model. We show weights between exemplars and activity labels to show our model can handle pose variation via the exemplar representation. Figure 5.3



(a) set 1 (b) set 2

Figure 5.1: Confusion matrices of per-clip classification result on UT-Interaction dataset. Horizontal rows are ground truth and vertical columns are predictions.

visualizes our learned activity-key pose weights. We visualize the weights for distances between the localization of key poses in each trajectory to illustrate the contribution of spatial constraints. The first bin (bin 1) is assigned to distance smaller than a threshold, and the last bin (bin 5) is assigned to all distances larger than the maximum step size. Figure 5.4 shows the learned spatial distance weights.

5.2 TRECVID Embrace Dataset

We collected a subset from the development dataset of the TRECVID 2008 Surveillance Event Detection challenge [52] for the embrace event classification task. Our goal is to examine performance of our method on non-choreographed activities. The full TRECVID dataset is very challenging, and state-of-the-art methods perform poorly on it (>95% **miss** rate at 10 FP/hour). Considering the fact that human detectors and trackers have difficulty in challenging datasets like TRECVID, we manually select a subset of the dataset the detector/tracker perform well. This subset will certainly be easier than the full dataset, but it can be argued that with a better detector/tracker, performance should improve.

We choose five days of video from camera view 3, which contains 343 *embrace* events (63% of those in the whole dataset). We manually select a positive set of 36 *embrace* clips where our detector and tracker provide reasonable output. We randomly sample 300 video clips that do not temporally overlap with the *embrace* events, and use the same human detector/tracker used for

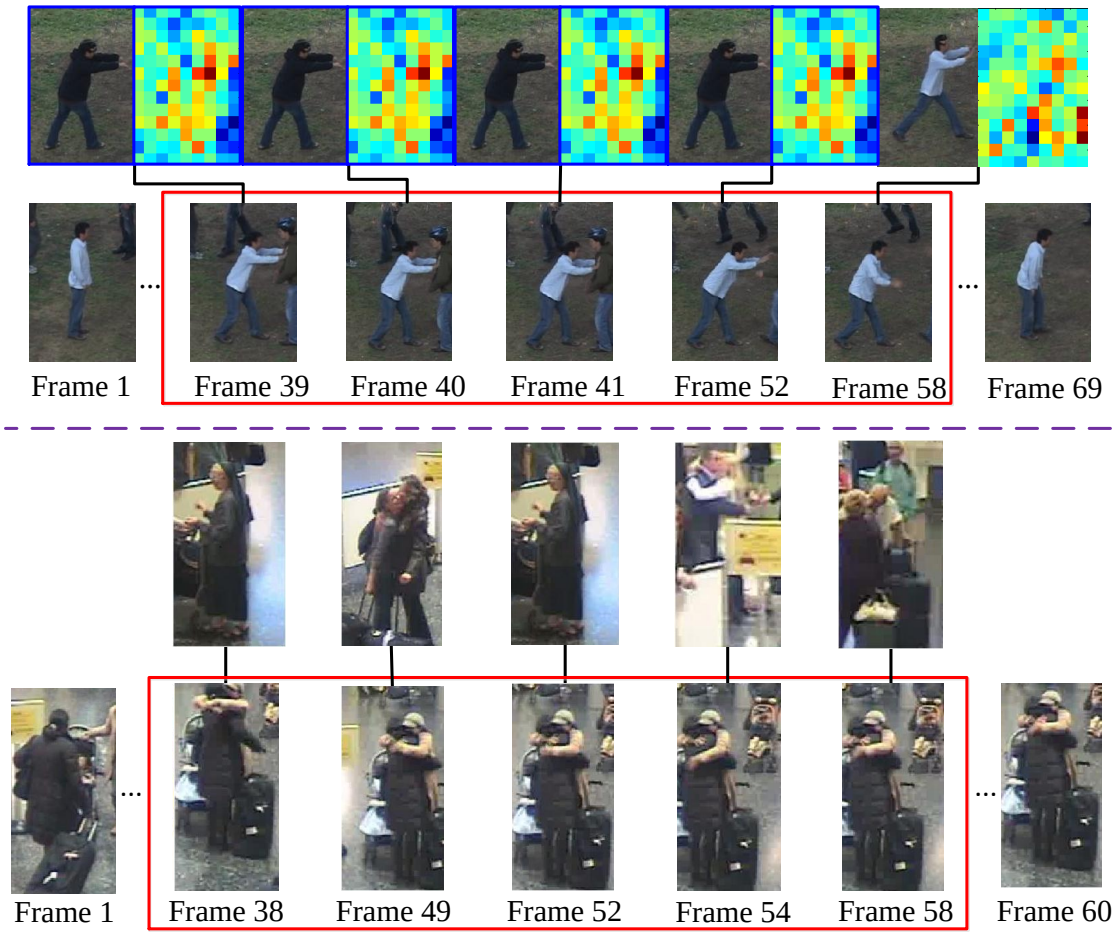


Figure 5.2: Discriminative frames of a trajectory are automatically extracted. Separated by a dashed line, the upper part of the figure comes from the UT-Interaction dataset and the lower part from the TRECVID embrace dataset. The localizations of key poses in trajectories are highlighted by red bounding boxes. In the upper part, our model localizes 5 key poses in a 69-frame long trajectory and selects exemplars for each of them. The frame number under each key pose localization indicates its time in the trajectory. Exemplars are selected based on similarity in appearance and localization of key pose. The similarity is defined as patch-weighted distance. The model learns to give high weights on patches where poses appear to be unique. Patch-based weights are shown beside each exemplar on the first row. The weights spread over the contour of each individual and concentrate on outstretched arms for the push activity. Similar visualizations are shown in the lower part for a trajectory from the TRECVID embrace dataset.

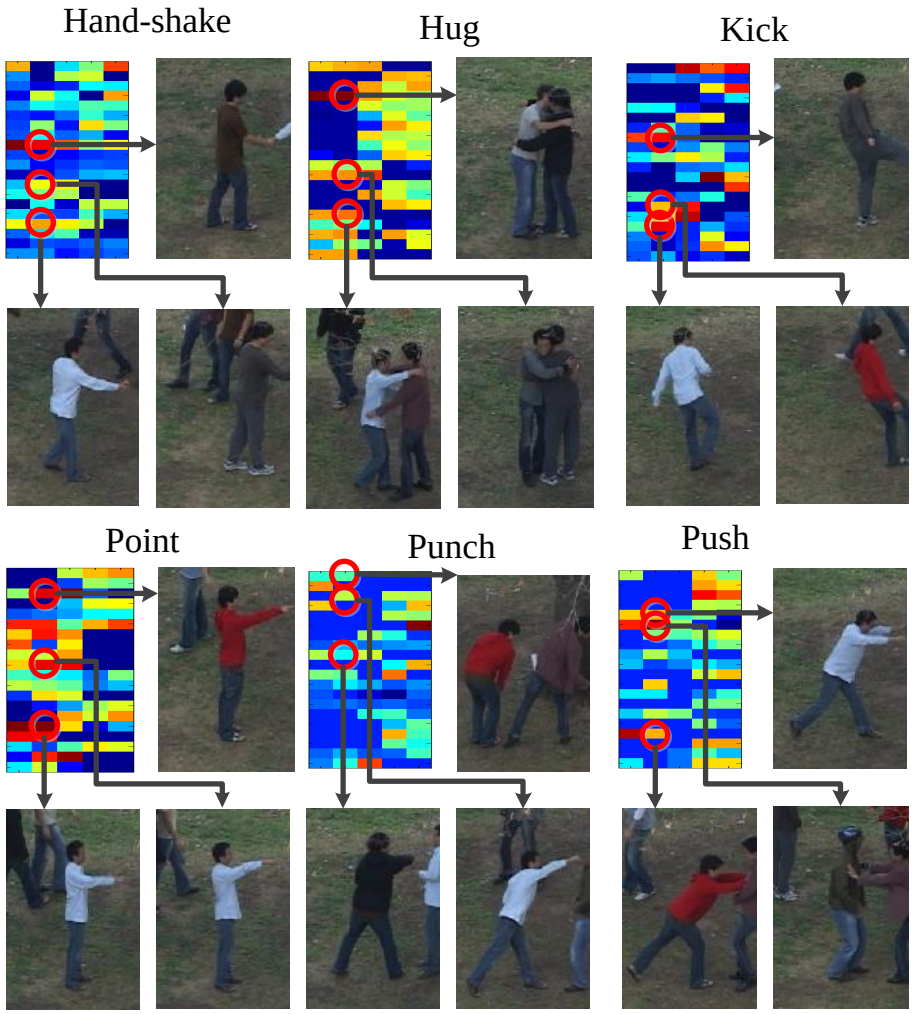


Figure 5.3: Visualization of activity-key pose model. For the heatmap of each activity, the horizontal axis is the concatenation of the 5 key poses in the activity and the vertical axis specifies 20 exemplars belong to the activity. Each pixel describes the score for an exemplar being matched to a key pose in the activity. The weights represent our model's preference for an exemplar in a key pose. For the second key pose in each activity, we also visualize the exemplars with highest weights. For each activity, selected exemplars have large pose variation.

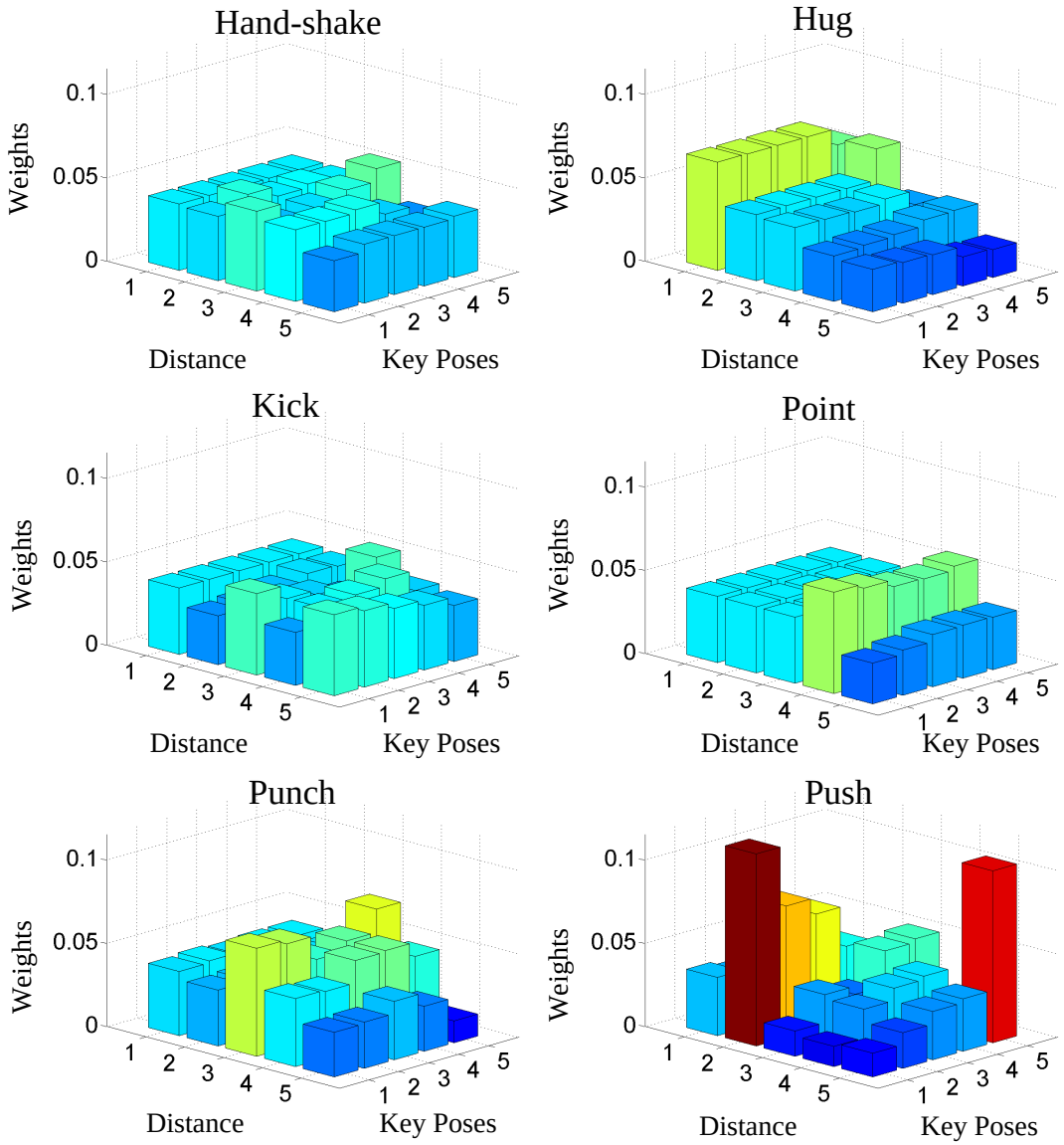


Figure 5.4: Spatial distance model for six activities in UT-Interaction dataset. Three axes are discrete distance, key poses and weights. For a key pose in each activity, the heights of bars indicate our model's preference among different distances. Bars are also colored according to height. The spatial distances in the *hug* activity are preferred to be smaller than that in the *point* activity, which illustrates the fact that people are closer to each other in hugging compared with pointing. For the *push* activity, the spatial distance preferred by the last key pose is much greater than previous ones, reflecting the separation of the two individuals at the end of the activity.

positive examples to obtain trajectories. There are 1074 pairs of trajectories that intersect spatio-temporally, but are not *embrace* events. We sample 108 pairs of trajectories to use.

The TRECVID Event Annotation Guidelines state that embrace starts at the latest time when subjects do not have physical contact prior to the embrace. However, we believe important and discriminative information is also present in frames before people have physical contact. For example, pairs of people with both arms outstretched strongly indicates the upcoming embrace event. So we decide to label the starting frame of *embrace* 20 frames earlier than the TRECVID ground truth. We also fix the length of *embrace* samples at 60 frames for both positive and negative samples. Note that the negative samples come from videos randomly sampled in time, hence is a fair comparison to non-embrace videos, though our dataset lacks the "near"-*embrace* events that would require non-maximum suppression. Our embrace dataset excludes groups hugging and other serious occlusions in which case one can barely see embrace event. However, the dataset still inherits the challenging characteristics of TRECVID videos: it contains large intra-class variation with a cluttered background. The precise dataset will be available for download at our website.

5.2.1 Preprocessing

Our dataset is created by collecting a set of trajectories from the TRECVID dataset. We run a HOF/HOG SVM human detector on the first frames of the clips and use a tracker [2] to obtain trajectories of individuals. The task is now a classification task – given a pair of trajectories, is there an *embrace* activity occurring or not.

5.2.2 Results

We evaluated our method using 6-fold cross-validation. To evaluate the effectiveness of different parts of our model, we introduce two baseline methods. The first baseline is our full model without the root model, the direct link between key poses and activity labels. The second baseline is our full model without the spatial distance model, the link between localizations of key poses in one trajectory and poses in the other trajectory simultaneously. The ROC curves in Fig. 5.5 show the effectiveness of our method relative to these baselines.

The 6% increase in AUR from the first baseline to our full model reflects the contribution of the root model to our full model. We only select trajectories that overlap spatio-temporally for negative examples, so as expected the models with and without spatial distance are similar.

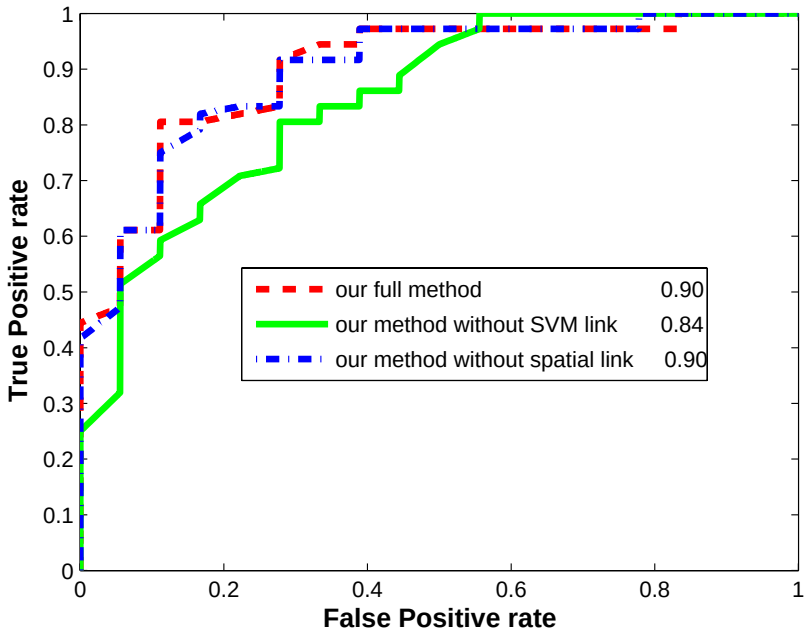


Figure 5.5: ROC curves on TRECVID *embrace* dataset. Legend shows Area Under ROC (AUR) for methods.

Our experiments are on a subset of the TRECVID *embrace* dataset, but we can extrapolate performance to the complete TRECVID dataset. Camera view 3 captures the majority of the *embrace* events. In the worst case, if we misclassify all other positive examples, the maximum achievable true positive rate (TPR) in ROC will be 63%. Due to failures of the human detector, tracker and ignorance of short positive samples, our TPR will at most decrease to $10\% \times 63\%$ of our reported TPR. However, our negative examples are randomly selected pairs of trajectories which overlap in space and time, they are a very difficult subset of negative examples. The results on our dataset indicate promising performance on the full TRECVID dataset.

Chapter 6

Conclusions

In this part we addressed human interaction recognition problem in computer vision. We presented a discriminative model for human interactions based on a sequence of key poses. Strict temporal ordering, and the spatial relation between actors in an interaction were modeled with structured-latent variables. Variability in instantiation of key poses are all enforced in this model using exemplar-based representation. An efficient dynamic programming algorithm for inferring key pose sequences was presented, and parameters were learned using the discriminative max-margin criterion. Experiments on the benchmark UT-Interaction dataset verified the effectiveness of the model. Further, non-choreographed activities were explored using a subset of the TRECVID dataset.

Our method has its limitations in handling the real world videos. The method uses human trajectories as input. As future works, examining the addition of tracking as an additional latent variable could alleviate this direct prerequisite. Moreover, due to the noise in the spatial domain and cluttered background, key pose localization can sometimes fail. Our current key pose representation is limited to the spatial domain and two successive frames in temporal domain due to the use of the HOG and the motion features. One can define richer segments in wider temporal domain. Exploring such rich segmentation and representation is another possible direction of future research.

In order to develop a tractable system we were limited to a simple model. Our strict ordering of key poses can not recognize the scale of an action in temporal domain or the length of irrelevant movements between the key poses. The temporal distance of key poses could be modeled rather than their order. In addition, a richer model can be developed by modeling the co-occurrence of key poses of actors or their temporal distance instead of distance between trajectories. But, both the modifications in temporal order of key poses and their spatial link will make the exact inference intractable which require further research on approximation techniques.

Part II

Color From Gray

Chapter 7

Previous Work

Color quantization techniques have been mainly explored in 1980s and 1990s, the period that the techniques were popular due to the limitation of display memory. Modern computers can now display milions of colors which limits the application of color quantization to mobile devices. In addition to display, color quantization may be used to compress images, since they can replace colors (typically 24bits) with their indices (8bits or less) in color look-up tables.

Color quantization methods can be divided to two groups: *fixed quantization* and *adaptive quantization*. In fixed color quantization, a fixed set of representative colors is used for all images. But, in adaptive methods representative colors are formed for an image such that the visual difference of output is as close as possible to the input color image. The advantage of fixed quantization is that it eliminates the need for attaching the colormap to the image which saves the required memory to store and transfer the image. For this reason, fixed color quantization is widely used on world wide web specially for small images such as logos or icons that have small size and attaching the look-up table is expensive e.g. web safe colors. However, a fixed color mapping has large amount of error on real world images which have broad range of colors. For this purpose most researchers focus on adaptive color quantization techniques.

In adaptive color quantization the set of representative colors is created based on the color points of input image such that the quantized image has small error. The error usually is measured by mean squared error, and the task can be considered as clustering of color points to groups that total error of groups is minimum. It has been shown in Gray et al. [17] that the finding the optimum solution to such problem is NP-complete. Hence, proposed methods uses heuristics to find an approximation of the optimal representative colors. These heuristics can be divided to two main categories: clustering methods and splitting methods. The clustering methods are typically bottom-up techniques that

iteratively group the data to clusters that have lower error. Splitting techniques usually start with all points, and divide them to smaller groups in each iteration such that each group has smaller error.

7.1 Clustering Methods

Clustering techniques are typically bottom-up iterative methods, that start with all color points and group them to clusters that have lower errors. Time complexity of these methods are usually higher than splitting methods. However, in practice they usually achieve better approximation of optimum solution.

The popularity algorithm is an early method proposed by Tom Boyle and Andy Lippman in 1978 which was implemented in Heckbert [22]. The method is not iterative, but is bottom-up, hence it is considered as clustering technique. This method divides the color space to small bins, and calculates the histogram of points using those bins. Then, the bins with the largest number of colors are chosen for color mapping, and the points in the non-empty bins which are not chosen are assigned to the closest bin. Finally, representative colors are set to the mean of each group. The method is sensitive to the initial bins size and it may fail for images with wide range of colors.

Several clustering methods are used for color quantization. Linde et al. [35] used popular clustering algorithm, k-means method, for vector quantization. They initialize centers of the clusters with random samples from colors. In each iteration they assign a point to the closest cluster, and they update the centers to the mean of the points in the cluster. Xiang and Joy [65] used the agglomerative clustering technique for color quantization. In this technique, each color point represents a cluster at the beginning of the algorithm. In each iteration, two clusters are chosen to merge such that the resulting cluster has the lowest error. They represent each cluster by surrounding box, and, they measure error by spread of color points in dimensions of the box. Xiand [66] minimized the maximum interclass distance in order to cluster the input color points. They proposed an iterative approach which starts with all points at the beginning, an in each iteration it calculates the distance of the representative of a cluster to all other points in the cluster. The furthest point is assigned as representative of a new cluster. The points closer to the new representative are then moved to its cluster.

Dekker [9] used the self organizing networks to cluster the color points. A network is defined by a set of nodes and edges. Nodes are representative colors in the color map, and the edges connects the neighboring color points in color space. In each iteration, the position of the nodes are updated toward weighted direction of both the closest color points of the input image, and the neighboring

nodes. At the end, the network spreads in the color space such that it represents the topology of color points of the input image. Generally speed of self organizing maps depends on the initialization of the network. Mavridis and Papamarkos [39] used the principal component analysis to initialize the network for color quantization.

7.2 Splitting Methods

Splitting methods typically start with all color points of an image as a cluster. In each iteration they choose a cluster, and divide it into small groups such that the resulting groups have the smallest error. These techniques differ based on the criteria of cluster selection, the splitting border location, and number of groups that are created in each iteration.

One of the most popular splitting methods is the median cut algorithm proposed in Heckbert [23]. The basic idea of the median cut algorithm is that each color entries in a colormap represents the same number of pixels in the original image. This method starts with a box that tightly encloses all color points of an image. In each iteration, a box with largest range on an axis is selected. Then, a plane perpendicular to the axis at the median of points along that axis divides the box to two smaller boxes which have the same number of color points. The process continues until the number of the rectangular subspaces reaches a desired number of entries in colormap. Wu and Witten [61] proposed mean-split algorithm which places the partitioning plane at the mean of a box, and Joy and Xiang [27] place the plane at the center of the box. However, Wan et al. [55] argued that considering the variance instead of spread for partitioning results in lower mean squared error. They chose the box with the highest variance, and they find the partitioning plane that reduce the variance most. Orchard and Bouman [43] proposed a binary splitting tree similar to Wan et al. [55]. However, they find the partitioning plane perpendicular to the principal direction of the data instead of the space axes.

Gervautz and Purgathofer [18] proposed the octree portioning method for color quantization. They saved the color map entries in the leaves at the the 8^{th} level of a tree whose its inner nodes are from degree 8. They scan the input image and they push down all colors of image to the corresponding leaves in the tree until the number of leaves exceeds the number of representative colors. Then, the leaves at the deepest level with the smallest number of nodes are merged to their parrents node. Finally, each color map entries is assigned to the mean of colors at the corresponding leaf.

Balasubramaian et al. [3] proposed the sequential color quantization which splits the color space sequentially along the axes of color space. First, they quantize a component of color according to the

marginal distribution of the component. Then, they quantize the second color component using the conditional distribution of the second color component given the first component, and similarly they quantize the third component using the first two components. Even though they derive the optimum quantization for each successive stage given an order of color components, but, they iterate the algorithm for each order to find the one that results in minimum mean squared error.

Some methods use both splitting methods and clustering techniques for color quantization. Wu and Zhang [62] find a splits the partitions similar to Wan et al. [55]. Then, two resulting partitions are fed to the algorithm proposed in Linde et al. [35] as initial clusters to find a locally optimum bi-partition with minimum sum of inner variance. Procedure chooses a partition with largest variance, and continue the partitioning and the refining until a desired number of clusters are achieved. In later work Wu [60] show that for color image data, points are usually spread along the intensity direction, and the first few partitioning planes are perpendicular to the first principal component of data. They proposed to split the color points to more than two partitions using parallel planes found by a dynamic program at the beginning of partitioning, and they use other binary partitioning afterwords such as the one in Wu and Zhang [62].

Overall splitting methods are greedy algorithm that in each iteration attempt to minimize the error by choosing the partition that has the highest decrease in error. Even though they are typically faster than clustering methods, but their error is greater than clustering techniques.

A common artifact of color quantization is contouring which is resulted from mapping smoothly varying regions to small of number of representative colors. Typically a subsequent processing is done on the quantized color image to reduce the degradation. The techniques are known as *Dithering* methods which are based on the properties of human vision system. Human beings perceive high frequency spatial variation as homogeneous colors, which can be considered as a low-pass or averaging filter. These techniques create larger number of colors by spatial mixing of colors.

Some color quantization algorithm model both the quantization and dithering together. Orchard and Bouman [43] reweight the squared error of quantization based on the area of a region with same quantized color after first color mapping, and repeat the process with new weights to decrease the contouring effect. Puzicha et al. [45] model both the quantization and the dithering in a cost function which is derived from the human visual system.

Chapter 8

Color from Gray by optimized color ordering

Our goal in this part is to store color information in gray values, and, to recover color later using small amount of extra information attached to the constructed gray image. However, the mapping from color image to gray scale typically deletes color information by assignment of the same gray intensity to different colors. For example, in the standard NTSC multimedia-standard mapping from RGB to gray scale luma is done by $Y' = 0.299R' + 0.587G' + 0.114B'$, where typically primed quantities are gamma-corrected values. For quantized gray levels this means that in RGB color space points on planes with normal vector $u = (0.299, 0.587, 0.114)$ all are assigned to the same gray scale value. As an extreme case, in Fig. 8.1 a color image and its corresponding, *single* gray level in this case, image are shown: Although the image consists of different colors, all colored points on it have the same gray scale value.

We also need to measure the error in the recovered color image, so we transform the image to CIELAB color space instead of using RGB space. CIELAB space is a perceptually uniform color space which means that distance of two color points reflects the magnitude of the perceptual difference observed by human visual system. CIELAB color space consists of three components. L^* component represents the lightness and both a^* and b^* reflect the color dimensions. We here use L^* and CIELAB color space respectively as our internal gray space and color space and we measure the error by the colorimetric error [63] which is Euclidean distance between two points in the gray or color space. We also assume that input-image colors are in standard, nonlinear sRGB color space [7], and visualization will be done in sRGB space.

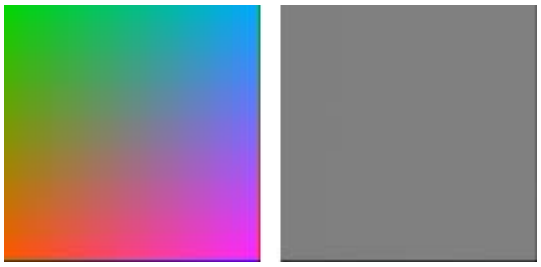


Figure 8.1: Left: an image containing different colors, all corresponding to the same gray. Right: Grayscale image of left image.

8.1 Color to Gray and Back

In order to encode color information in an L^* gray scale image, we assume that in a small color neighborhood each gray intensity corresponds to a fixed color point, representing all neighboring colors instead of whole plane visualized in Fig. 8.1. The task at hand, then, is to decide which representative color to use for each gray level, adaptive to the input image, and also allow this mapping changes if that will decrease color error at the expense of some gray scale error.

In Fig. 8.2 this mapping for a small region in color space is shown, using solid points for the representative color in each plane. Suppose we have created a curve traversing colors in color space in an orderly fashion so as to move steadily upward from L^* gray level 0 to gray level 255, visiting a rich color sampling in both hue and saturation. In the illustrative figure, suppose that for quantized gray scale g, the color assigned is a green, and for the next quantized gray value $(g-1)$, the color is a pink and for $(g-2)$ is a blue — filled points on a spiralling curve are the *initial* representative colors that correspond to each of the quantized gray levels.

In this case to encode an input color, that happens to correspond to gray levels g, we use the gray scale of the *closest* filled point in 3D; e.g., for input-image point A, which happens to be a red and has quantized L^* value g, and is not one of our representative colors on the mapping, we actually use a gray scale of $(g-1)$ instead of g, because (1): $(g-1)$ is within a small search range in quantized gray scale of g, and (2): the representative color for $(g-1)$ is closer to A's color. So,

we represent color point A with point B because it is the closest to A. We accept small amount of error in gray scale image by assigning gray value $(g-1)$ instead of g to point A. However, this small error enables us to recover approximate color of A in decoder side, since $g-1$ is assigned to a color point close to the original color of A. Hence, all that is necessary at the decoder side is (1) the gray level actually assigned to the pixel and (2) the mapping used, so that the color pertaining to that gray level is known.

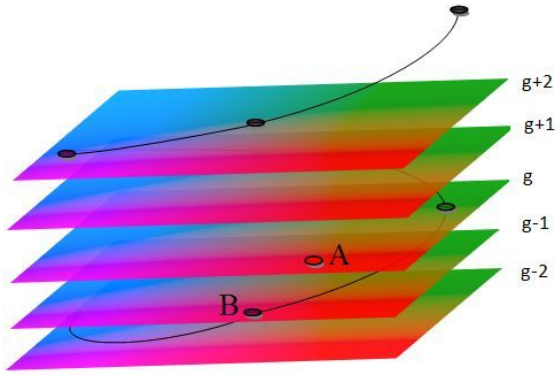


Figure 8.2: Color planes with uniform gray scale values. On each plane we fix a color point on the intersecting curve and during transformation from color image to gray scale value we use gray scale of the closest fixed point. So point A is transformed to gray scale $g-1$ and in reconstruction color point B is used.

8.2 Parametric curve

As illustrated in Fig. 8.2, we need an efficient way to store and represent the mapping between gray scale and color values. The standard approach is to use a palette to store RGB values in a look-up table. The image data itself then consists of look-up table index values, which cannot of course provide a sensible gray scale image itself – instead, gray is produced by regenerating an approximate color image and then transforming that to gray.

Here we take the tack of instead using a 3-space curve $C(g) : \Re \mapsto \Re^3$ which maps gray scale values to points in color space. By choosing an appropriate parametric function we can optimize its shape for each input image, and attach its parameters instead of a large look-up table. Not only do

we thus save on bandwidth, but in fact we show in Chapter. 9 that we achieve comparable results to the standard adaptive GIF file in the accuracy of gray images and recovered color.

The mapping function can be thought as a curve with gray scale value as its parameter. In order to have small error on transformation form color to gray and back, the mapping should cover different colors as gray scale changes. In other words, the curve should have different color samples in small a neighborhood of gray scale values. Hence, in CIELAB space, if we assume a curve with a vertical axis, equal to lightness component L^*, a helical curve will satisfy this requirement. As shown in Fig. 8.2, as g increases the curve travels through different colors on planes parallel to the a^*, b^* plane. However, the requisite curve should also cover different saturation values which corresponds to distance of curve from the middle axes. Thus, instead of using a fixed radius, we assume an alternating radius with a sliding Gaussian peak:

$$\text{radius} = r(g) = c_1 \exp(\frac{-(g-c_2)^2}{(c_3)^2}) \sin(c_4\, g + c_5) \; + \; c_6 \tag{8.1}$$

where g is gray level value.The mapping curve is then taken to be

$$\begin{aligned} a^* &= r(g)\sin(c_7\, g) \; + c_a \\ b^* &= r(g)\cos(c_7\, g) \; + c_b \\ L^* &= g \end{aligned} \tag{8.2}$$

where L^*, a^*, b^* are the color components and $c_1, c_2, \ldots, c_7$ are the coefficients which will be optimized later. c_a and c_b are the center of the curve in the a^*, b^* plane, which can be assigned to the mid-value 0 (128 in Matlab). However to allow the main L* axis to bend to suit the actual gamut of the image, instead of fixing c_a and c_b we add two further 3-vector parameters, $\boldsymbol{w}_1, \boldsymbol{w}_2$ expressing a polynomial fit of g to a* and b*:

$$\begin{aligned} \left[1,\; g,\; g^2\right] \boldsymbol{w}_1 \;&=\; a^* \\ \\ \left[1,\; g,\; g^2\right] \boldsymbol{w}_2 \;&=\; b^* \end{aligned} \tag{8.3}$$

Hence our final expression of curve C is as follows:

$$\begin{aligned} a^* &= r(g)\sin(c_7\, g) + \; \boldsymbol{w}_1 \cdot [1,\, g,\, g^2] \\ b^* &= r(g)\cos(c_7\, g) + \; \boldsymbol{w}_2 \cdot [1,\, g,\, g^2] \\ L^* &= g \end{aligned} \tag{8.4}$$

$c_1, c_2, \ldots, c_7, \boldsymbol{w}_1$ and $\boldsymbol{w}_2$ are 13 parameters of our curve. The radius and curve with initial parameters used in optimization are shown in Figs. 8.3 and 8.4.

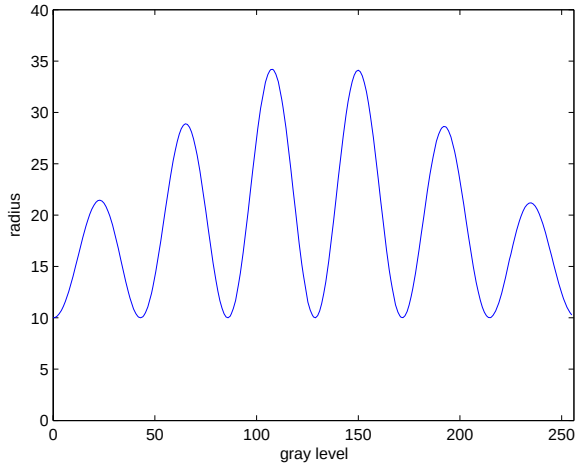


Figure 8.3: Radius function.

8.3 Transform to gray scale image and color image reconstruction

Each pixel in the input image can be thought as a point in 3-dimensional CIELAB space. In order to transform it to gray scale, first the closest point on the curve in 3D is found: then its gray level is used as that pixel's gray level. Due to the special shape of the curve that closest color point will also have similar gray scale value and the amount of error will be small. So, we can use this fact when we are looking for the closest point on the curve to a point by limiting the search range to a neighborhood of original gray scale value of that point. Hence, the gray scale value for each pixel p with color $\rho(p)$ is obtained by:

$$gray_{apex} \;=\; \arg\min_{g} \left(\|C(g) - \rho(p)\|\right) \tag{8.5}$$

where $g = L^*(\rho) - K, \ldots, L^*(\rho), \ldots, L^*(\rho) + K$ and K is the search range. In order to reconstruct the color image from the corresponding gray scale image constructed in this way, we simply use the corresponding color point on the curve for gray value $gray_{apex}$:

$$Lab_{recon} = C(gray_{apex}) \tag{8.6}$$

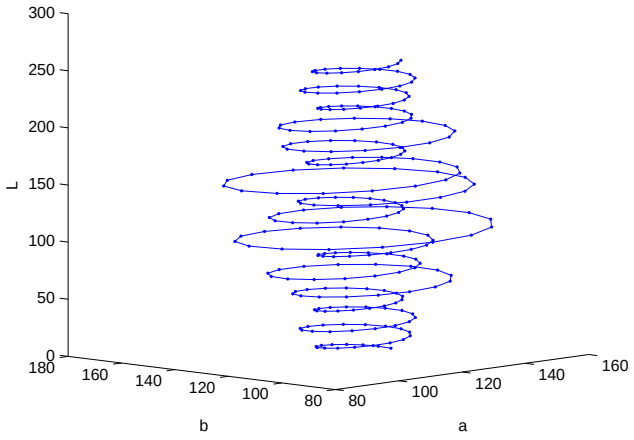


Figure 8.4: Parametric curve traversing color space.

8.4 Optimization

Each color image can consist of points in different regions of CIELAB space. So a fixed curve cannot always reconstruct the image very well. The main advantage of the proposed curve is that its shape can be changed by a few parameters. Thus, for each input image we adaptively optimize the parameters such that the error in both the gray image and the reconstructed color image is low. The parameters $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ captures the center of curve and can be found by a simple polynomial fit in Eq. 8.3. However, we need to find optimum values for $c_1, \ldots, c_7$. We limit the search over nearby gray scale values to $g \rightarrow g \pm k,\ k = 0..K$ (we use K=5 here). Then the appropriate objective function is as follows:

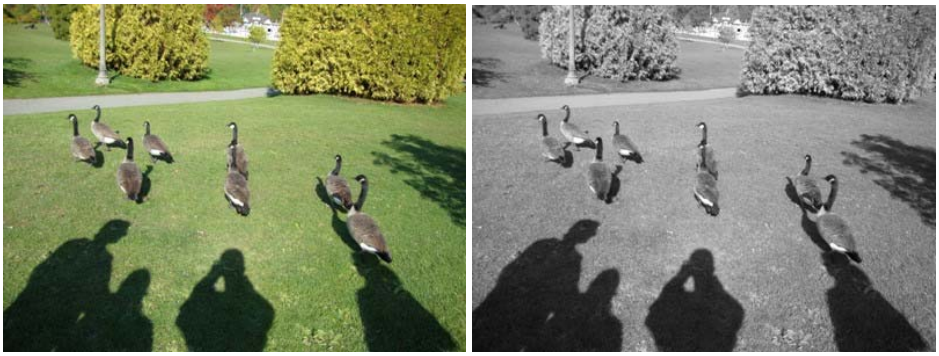
$$\min_{c1..c7} Cost(C,\, \rho) \;=\; \sum_{p \in \Omega} \left\{ \min_{k=-K..K} \|C(g+k) \,-\, \rho\| \right\}^2 \tag{8.7}$$

with $g = L^*(\rho)$ and where Ω is the image domain. Optimization in Eq. 8.7 forces the closest point on the curve in small neighborhood of original gray values get as close as possible to the original color. We use Matlab's built-in nonlinear least-squares curve fitting function which is implementation of the Trust Region Approach [6] to minimize the cost function in (Eq.8.7).

Chapter 9

Experiment Details and Results

Let us compare the performance of our algorithm with the standard GIF file format color quantization method, especially concentrating on low-bitrate representations for mobile-device applications. For fewer bits, we quantize to only 2^n gray levels and also search in color space only on those quantized planes. However, we should note that even comparing our method to the GIF file format is unfair. The GIF file type saves the color mapping in a look-up table. But, in our method, the color mapping is stored in a curve with 13 parameters. Our goal is to save the color information in gray level values, but the GIF file format has not such constraint.



(a) (b)

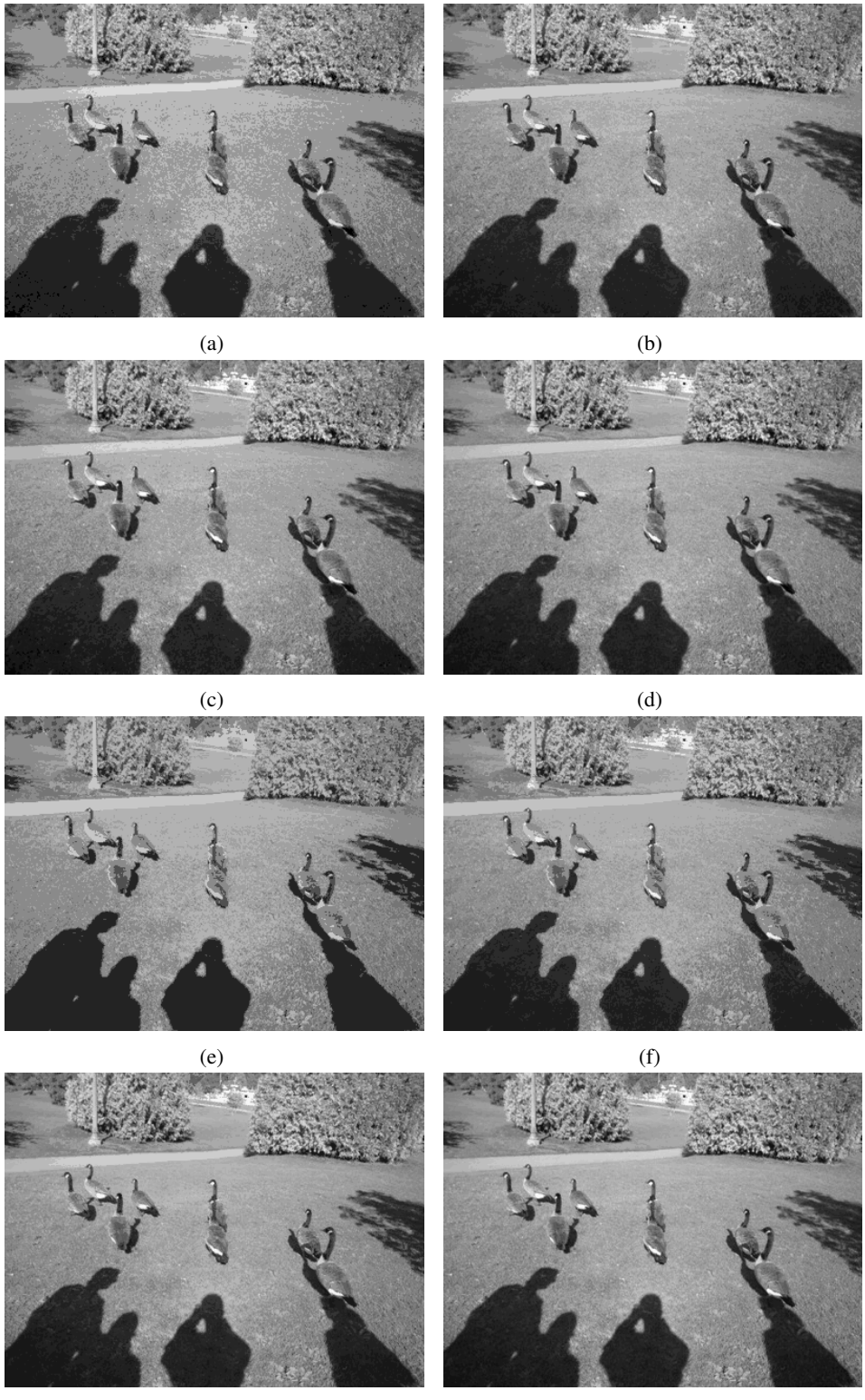
Figure 9.1: (a): Input image; (b): gray for input.

Consider the input image I_1 shown in Fig. 9.1(a): its corresponding gray is shown in Fig. 9.1(b). Now, Figs. 9.2(a-d) show the gray images for the present method, for bpp (bits per pixel) values 3,4,6,8; and Figs. 9.2(e-h) show the corresponding gray images using GIF. Figs. 9.3(a-d) show the

color versions for the present method, over these bpp values, and Figs. 9.3(e-h) show GIF color. Fig. 9.4 shows quantitative comparison of our method for a few images. Typically, the present method generates a better gray representation for low bitrates, and can also generate a comparable color as well. Nevertheless we see that the method performs remarkably well given the small number of curve parameters needed to be stored on top of the gray scale values.

The error measure used here is CIELAB error, for both gray images and color, and one should note that since this is a pixel-wise measure, it does not correctly take into account mechanisms of spatial integration, as in e.g. the s-CIELAB measure proposed by Zhang and Wandell [70]. Since for lower bitrates GIF tends to produce speckled images, a better error metric might tend to discount these speckles in favor of the overall fidelity of the image and thus numerical results might be different.

Fig. 9(a) shows how the fit to the gamut for Fig. 9.1 (shown in red, subsampled) is indeed well fit by the (blue) parametric curve. The GIF palette is shown in Fig. 9(b), as opposed to that for the proposed method in Fig. 9((c). For our method, the palette is of course well-ordered from L^*=0 to 255. Further results are shown in Fig. 9.6, Fig. 9.7 and Fig. 9.8.



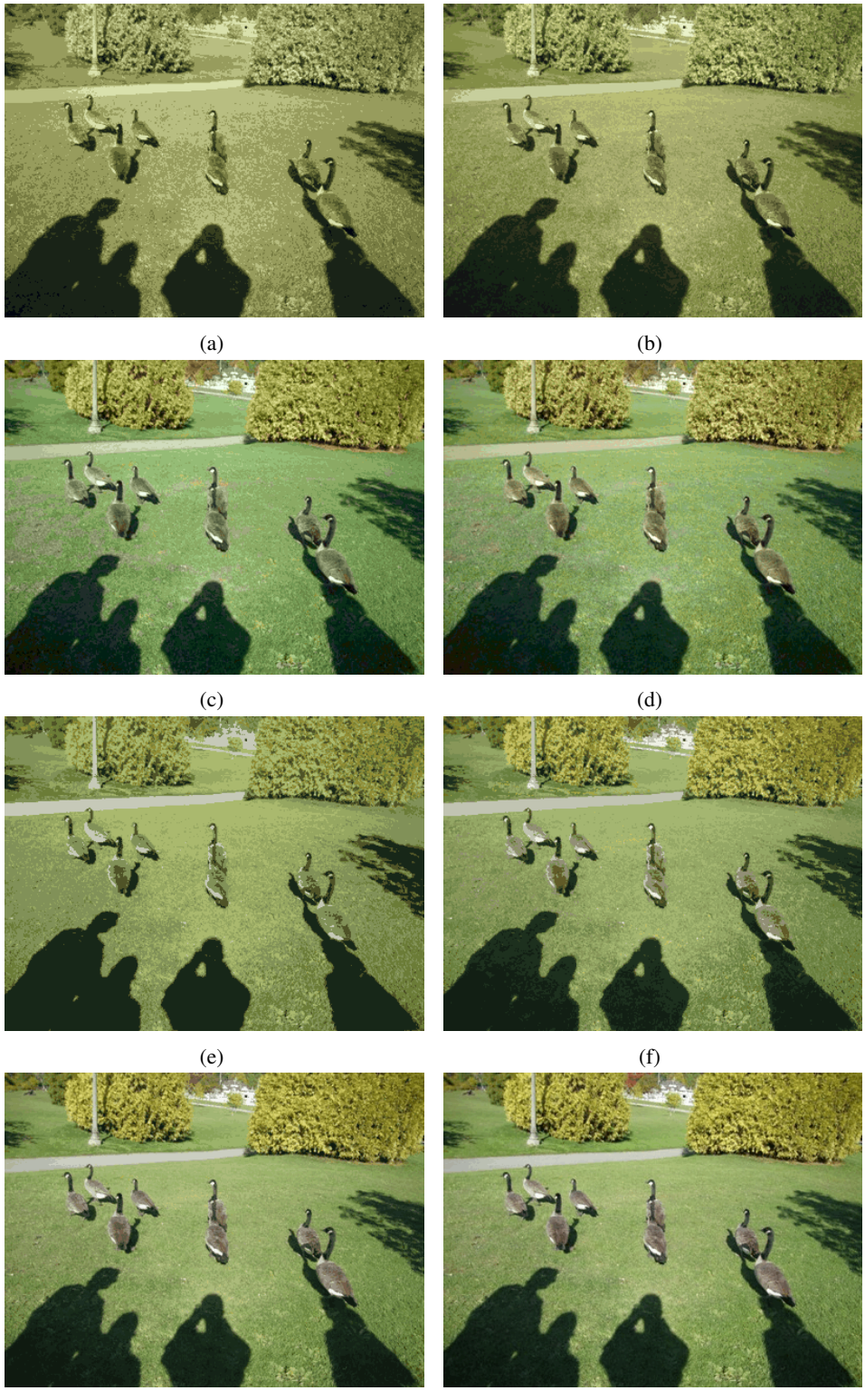
(a) (b)

(c) (d)

(e) (f)

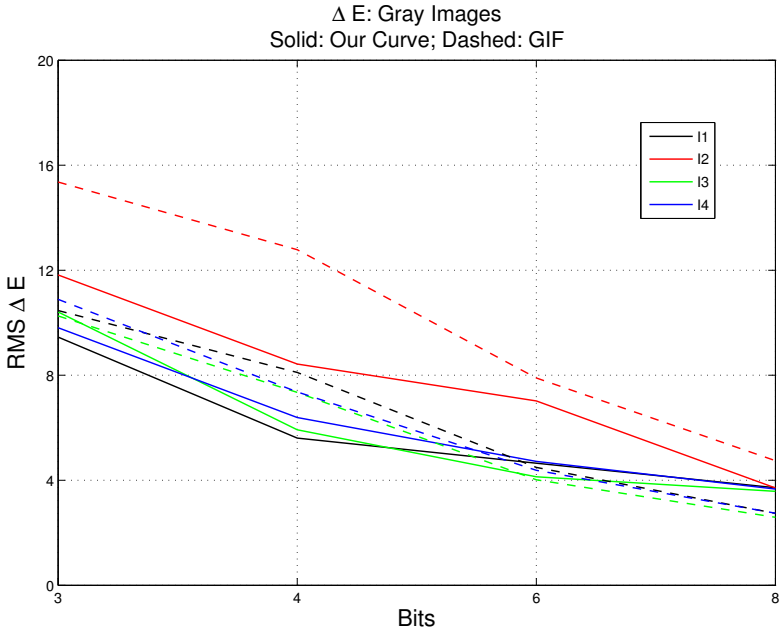
(g) (h)

Figure 9.2: (a-d): Gray for 3,4,6,8 bpp; (e-h): GIF gray.

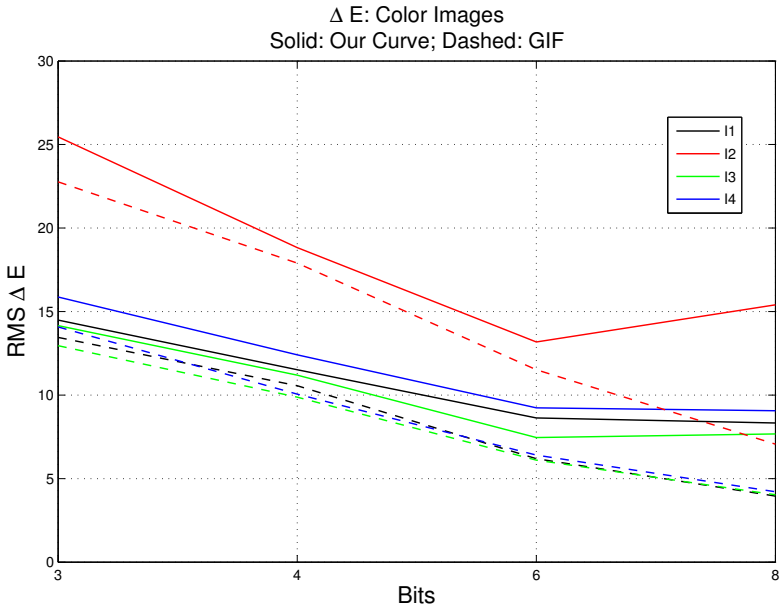


(a) (b) (c) (d) (e) (f) (g) (h)

Figure 9.3: (a-d): Color for 3,4,6,8 bpp; (e-h): GIF color.

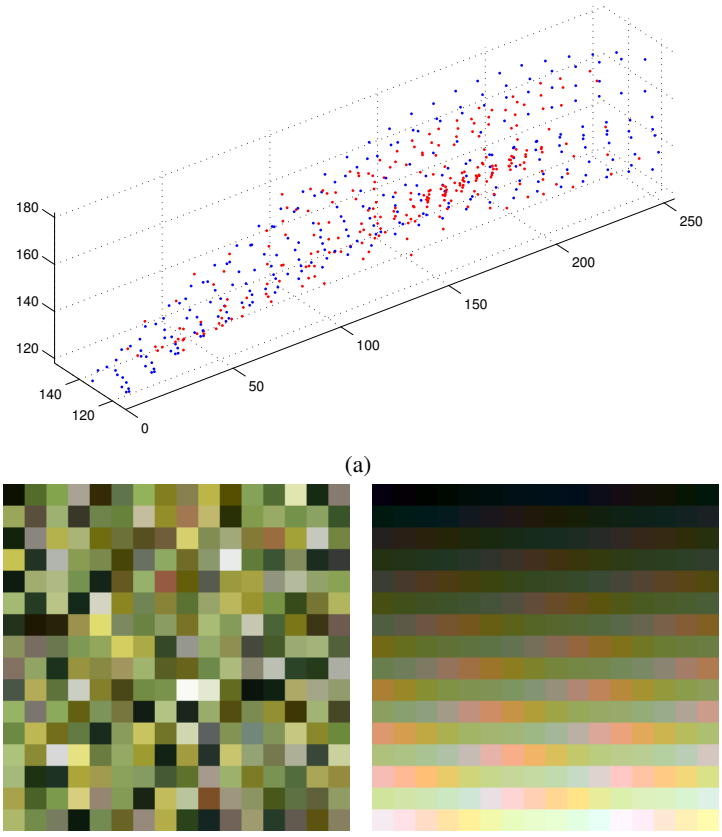


(a)



(b)

Figure 9.4: (a): CIELAB errors for gray images,; (b): Color error for input images I_1-I_4 visualized respectively in Fig. 9.3, Fig. 9.6, Fig. 9.7 and Fig. 9.8



(a)

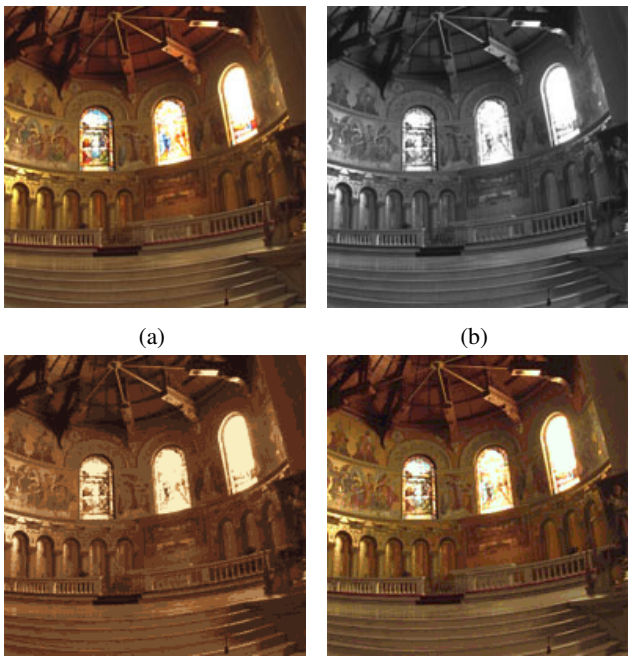
(b) (c)

Figure 9.5: (a): Gamut encompassed by parametric curve; (b): GIF palette; (c): Ordered colors along curve.

(a) (b)

(c) (d)

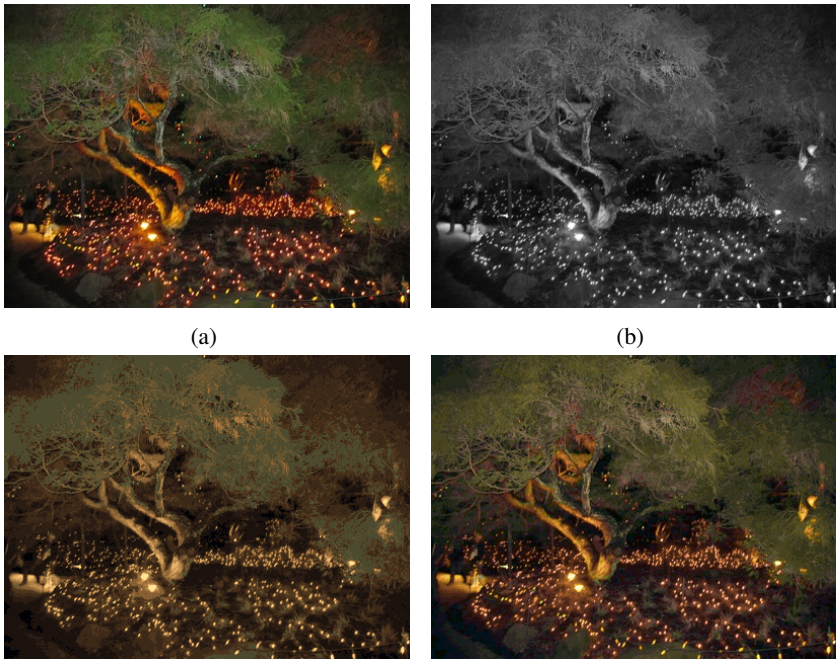
Figure 9.6: (a): Input image; (b): Gray for input; (c,d): Color output at 4bpp,8bpp.



(a) (b)

(c) (d)

Figure 9.7: (a): Input image; (b): Gray for input; (c,d): Color output at 4bpp,8bpp.



(a) (b)

(c) (d)

Figure 9.8: (a): Input image; (b): Gray for input; (c,d): Color output at 4bpp,8bpp.

Chapter 10

Conclusions

In this part, we proposed a novel method to reconstruct color from a gray scale image, by optimizing a mapping from gray scale to color using a parametric curve. Remarkably, for low bitrate results show that the method reconstructs gray scale with surprisingly small error, and the color version had comparable error.

Our method has several limitations. We used a constant quantization rate to sample from the curve. This results in dense samples at small radius or low saturation but sparse samples at large radius or saturated colors. So, the method has lower error for images with low saturation. In addition, the accuracy of our method declines for an image with narrow range of colors. Because typically the curve traverse over all color space, and some representative color are wasted. Different quantization methods can be explored to address these problems which is a possible direction of future research. We also ignored spatial domain in color quantization. One further study can be done on incorporating spatial information in optimization.

Overall, we have succeeded in the intent of this part, namely showing that a very low-complexity curve (just 13 parameters, here) may indeed describe the gamut of the input image sufficiently well. Clearly it is likely possible to improve the precise form of the curve, and as well user studies are called for to study the efficacy of the method. These are the subject of future work.

Bibliography

[1] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 32(2):288–303, 2010.

[2] B. Babenko, Ming-Hsuan Yang, and S. Belongie. Visual Tracking with Online Multiple Instance Learning. In *CVPR*, 2009.

[3] R. Balasubramaian, C. A. Bouman, and J. Allebach. Sequential Scalar Quantization of Color Images. *Journal of Electronic Imaging*, 3(1):45–59, 1994.

[4] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Trans. PAMI*, 23(3):257–267, 2001.

[5] H. S. Chen, H. T. Chen, Y. W. Chen, and S. Y. Lee. Human action recognition using star skeleton. In *International Workshop on Visual Surveillance and Sensory Networks*, pages 171–178, 2006.

[6] T.F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Optim.*, 6:418–445, 1996.

[7] International Electrotechnical Commission. Multimedia systems and equipment – colour measurement and management – part 2-1: Colour management – default RGB colour space – sRGB. IEC 61966-2-1:1999.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[9] Anthony H. Dekker. Kohonen neural networks for optimal colour quantization. *Network: Computation in Neural Systems*, 5:351–367, 1994.

[10] Trinh-Minh-Tri Do and Thierry Artières. Large margin training for hidden markov models with partially observed states. In *ICML*, 2009.

[11] Piotr Dollar, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[12] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003.

[13] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *J. of Machine Learning Research*, 9:1871–1874, 2008.

[14] A. Fathi and G. Mori. Action recognition by learning mid-level motion features. In *CVPR*, pages 1–8, 2008.

[15] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. PAMI*, 32(9), 2009.

[16] David A. Forsyth, Okan Arikan, Leslie Ikemoto, James F. O'Brien, and Deva Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2/3), 2005.

[17] M. R. Garey, David S. Johnson, and Hans S. Witsenhausen. The complexity of the generalized lloyd - max problem. *IEEE Transactions on Information Theory*, 28(2):255, 1982.

[18] M. Gervautz and W. Purgathofer. A simple method for color quantization: octree quantization. *Graphics Gems I*, pages 287–293, 1990.

[19] Andrew Gilbert, John Illingworth, and Richard Bowden. Fast realistic multi-action recognition using mined dense spatio-temporal features. In *ICCV*, 2009.

[20] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Transaction Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[21] C. Harris and M. J. Stephens. A combined corner and edge detector. In *Alvey Conference*, pages 147–152, 1988.

[22] P. S. Heckbert. Color Image Quantization for Frame Buffer Display. B.Sc. thesis, Architecture Machine Group, MIT, May 1980.

[23] Paul Heckbert. Color image quantization for frame buffer display. In *SIGGRAPH 82*, pages 297–, 1982.

[24] M. K. Hu. Visual pattern recognition by moment invariants. *IEEE Transaction on Information Theory*, 8(2):179–187, February 1962.

[25] Stephen S. Intille and Aaron Bobick. Recognizing planned, multiperson action. *CVIU*, 81:414–445, 2001.

[26] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.

[27] G. Joy and Z. Xiang. Center-Cut for Color-Image Quantization. *The Visual Computer*, 10(1):62–66, 1993.

[28] A. Klaeser, M. Marszalek, and C. Schmid. A spatio-temporal descriptor based on 3D-gradients. In *BMVC*, 2008.

[29] Adriana Kovashka and Kristen Grauman. Learning a hierarchy of discriminative space-time neighborhood features for human action recognition. In *CVPR*, pages 2046–2053. IEEE, 2010.

[30] Volker Krüger, Danica Kragic, Ales Ude, and Christopher Geib. The meaning of action A review on action recognition and mapping. *Advanced Robotics*, 21(13):1473–1501, 2007.

[31] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, September 2005.

[32] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2008.

[33] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Trans. PAMI*, 31(7):1294–1309, 2009.

[34] Zhe Lin, Zhuolin Jiang, and Larry S. Davis. Recognizing actions by shape-motion prototype trees. In *ICCV*, 2009.

[35] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Communications*, COM-28(1):84–95, January 1980.

[36] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *DARPA Image Understanding Workshop*, April 1981.

[37] F. J. Lv and R. Nevatia. Recognition and segmentation of 3-D human action using HMM and multi-class adaboost. In *ECCV*, pages IV: 359–372, 2006.

[38] F. J. Lv and R. Nevatia. Single view human action recognition using key pose matching and viterbi path searching. In *CVPR*, pages 1–8, 2007.

[39] Dimitris Mavridis and Nikos Papamarkos. Color quantization using principal components for initialization of kohonen SOFM. In *ICIP*, pages 1633–1636. IEEE, 2009.

[40] G. Médioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia. Event detection and analysis from video streams. *IEEE Trans. PAMI*, 23(8):873–889, 2001.

[41] T. B. Moeslund, A. Hilton, and V. Kruger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 103(2-3):90–126, November 2006.

[42] Juan Carlos Niebles, Chih-Wei Chen, , and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[43] M. T. Orchard and C. A. Bouman. Color Quantization of Images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[44] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[45] J. Puzicha, M. Held, J. Ketterer, J. M. Buhmann, and D. Fellner. On spatial quantization of color images. *IEEE Trans. Image Processing*, 9(4):666–682, April 2000.

[46] D. Ramanan and D. A. Forsyth. Automatic annotation of everyday movements. In *Advances in Neural Information Processing Systems 16*, 2003.

[47] Cen Rao, Alper Yilmaz, and Mubarak Shah. View-invariant representation and recognition of actions. *Int. Journal of Computer Vision*, 50(2), 2002.

[48] K. Rapantzikos, Y. Avrithis, and S. Kollias. Spatiotemporal saliency for event detection and representation in the 3D wavelet domain: Potential in human action recognition. In *Proceedings of the 6th ACM International Conference on Image and Video Retrieval, CIVR*, 2007.

[49] David Ross, Jongwoo Lim, and Ruei-Sung Lin. Incremental learning for robust visual tracking. *IJCV*, May 2008.

[50] M. Ryoo and J. Aggarwal. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *ICCV*, 2009.

[51] Qinfeng Shi, Li Cheng, Li Wang, and Alex Smola. Human action segmentation and recognition using discriminative semi-markov models. *Int. Journal of Computer Vision*, 2010. to appear.

[52] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR*, 2006.

[53] C. Sminchisescu, A. Kanaujia, and D. Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 103(2-3):210–220, November 2006.

[54] J. Sullivan and S. Carlsson. Recognizing and tracking human action. In *ECCV*, 2002.

[55] S. J. Wan, S. K. M. Wong, and P. Prusinkiewicz. An algorithm for multidimensional data clustering. *ACM Trans. Math. Softw.*, 14:153–162, June 1988.

[56] Yang Wang and Greg Mori. Learning a discriminative hidden part model for human action recognition. In *Advances in Neural Information Processing Systems (NIPS) 21*, 2008.

[57] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *CVPR*, 2009.

[58] D. Weinland and E. Boyer. Action recognition using exemplar-based embedding. In *CVPR*, pages 1–7, 2008.

[59] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *CVIU*, 2010. to appear.

[60] Xiaolin Wu. Color quantization by dynamic programming and principal analysis. *ACM Trans. Graph.*, 11:348–372, October 1992.

[61] Xiaolin Wu and Ian H. Witten. A fast K-means type clustering algorithm. Technical Report 85/197/10, University of Calgary, June 1985.

[62] Xiaolin Wu and Kaizhong Zhang. A better tree-structured vector quantizer. In *Data Compression Conference*, pages 392–401, 1991.

[63] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas*. Wiley, New York, 2nd edition, 1982.

[64] T. Xiang and S. Gong. Beyond tracking: Modelling activity and understanding behaviour. *IJCV*, 67(1):21–51, 2006.

[65] Z. Xiang and G. Joy. Color Image Quantization by Agglomerative Clustering. *IEEE Computer Graphics and Applications*, 14(3):44–48, May 1994.

[66] Z. G. Xiang. Color image quantization by minimizing the maximum intercluster distance. *ToG*, 16(3):260–276, July 1997.

[67] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *CVPR*, 1992.

[68] A. Yao, J. Gall, and L.J. Van Gool. A hough transform-based voting framework for action recognition. In *CVPR*, 2010.

[69] Tsz-Ho Yu, Tae-Kyun Kim, and Roberto Cipolla. Real-time action recognition by spatiotemporal semantic and structural forest. In *BMVC*, 2010.

[70] X. Zhang and B.A. Wandell. A spatial extension of CIELAB for digital color image reproduction. *SID Journal*, 5:61–63, 1997.