View-Invariant Representation and Learning of Human Action

Cen Rao and Mubarak Shah
Computer Vision Lab
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
{rcen, shah}@cs.ucf.edu

Abstract

Automatically understanding human actions from video sequences is a very challenging problem. This involves the extraction of relevant visual information from a video sequence, representation of that information in a suitable form, and interpretation of visual information for the purpose of recognition and learning. In this paper, we first present a view-invariant representation of action consisting of dynamic instants and intervals, which is computed using spatiotemporal curvature of a trajectory. This representation is then used by our system to learn human actions without any training.

The system automatically segments video into individual actions, and computes view invariant representation for each action. The system is able to incrementally learn different actions starting with no model. It is able to discover different instances of the same action performed by different people, and in different viewpoints.

In order to validate our approach, we present results on video clips in which roughly 50 actions were performed by five different people in different viewpoints. Our system performed impressively by correctly interpreting most actions.

Keywords: Video Understanding, Action Recognition, View-invariant Representation, Spatiotemporal curvature, Events, Activities

1. Introduction

What do we mean by an action? Webster's dictionary defines action: the doing of something; state of being in motion; the way of moving organs of the body; the moving of parts: guns, piano; military combat; appearance of animation in a painting, sculpture, etc. More or less, hand gestures, sign language, facial expressions, lip movement during speech, human activities like walking, running, jumping, jogging, etc, and aerobic exercises are all actions. Consider a typical office scene, at a given time a person can be performing either one of the following actions: reading, writing, talking to other people, working on a computer, talking on a phone, opening and closing cabinets, leaving or entering the office.

Actions can be classified into three categories: *events*, *temporal textures*, and *activities* [1]. Motion *events* do not exhibit temporal or spatial repetition. Events can be low-level descriptions like a sudden change of direction, a stop, or a pause, which can provide important clues to the type of object and its motion. Or they can be high level descriptions like "opening a door", "starting a car", "throwing a ball", or more abstractly "pick up", "put down", "push", "pull", "drop", "throw", etc. Motion verbs can also be associated with motion events. For example, motion verbs can be used to characterize trajectories of moving vehicles [2], or normal or abnormal behavior of the heart's left ventricular motion [3]. The temporal textures exhibit statistical regularity but are of indeterminate spatial and temporal extent. Examples include ripples on water, the wind in the leaves of trees, or a cloth waving in the wind. Activities consist of motion patterns that are temporally periodic and possess compact spatial structure. Examples include walking, running, jumping, etc.

Recognition of human actions from video sequences is very popular in computer vision. This work has applications in video surveillance and monitoring, human-computer interfaces, model-based compression, and augmented reality. One standard approach for human action recognition is to extract a set of features from each frame of a sequence, and use those features to train Hidden

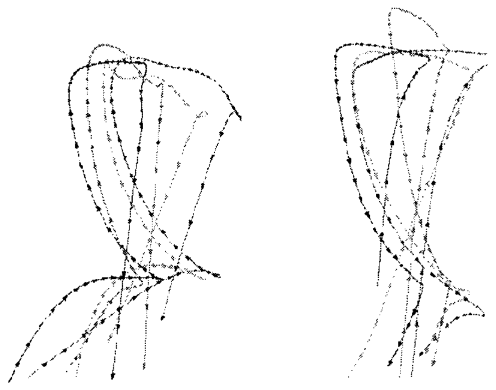


Figure 1. Several trajectories of "opening overhead cabinet" (a), and "closing overhead cabinet" (b) actions.

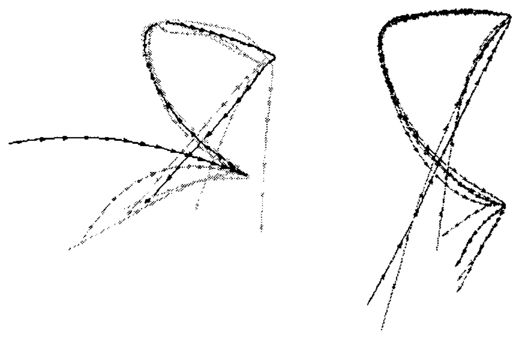


Figure 2: Several trajectories of "opening overhead cabinet" (a), and "closing overhead cabinets" (b) actions after converting them to the normal view using affine transformation.

Markov Models (HMMs) to perform recognition. The features can be an image location of a particular point on the object, a centroid of image region, moments of an image region, gray levels in a region, optical flow in a region (used as magnitude of optical flow, or concatenated u and v in a vector), sum of all changed pixels in each column (*XT* trace), 3-D locations (X_i,Y_i,Z_i) of particular point on the object, joint angels; how the parts of body move with respect to time, muscle actuations, properties of optical flow in a region like curl, divergence, etc, coefficients used in the eigen decomposition of above features, etc. A HMM consists of a set of states, a set of output symbols, state transition probabilities, output symbol probabilities, and initial state probabilities. The model works as follows. The features extracted from video sequences are used to train the HMMs. Matching of an unknown sequence with a model is done through the calculation of the probability that a HMM could generate the particular unknown sequence. The HMM giving the highest probability is the one that most likely generated that sequence.

In previous research, the most emphasis has been on discovering appropriate features. Therefore, not much work has been done on HMMs; they have been treated as a black box. There are several important issues related to HMMS. First, since HMMs rely on probabilities they require extensive training, therefore one needs to have a large number of training sequences for each action to be recognized. Second, for each action to be recognized, a separate HMM needs to be built. Therefore, this approach can only recognize some predefined set of action. It does not have a capability to learn new action. Third, since HMM is treated as a black box, it does not explain what a particular action is? It just outputs the probability an unknown action is recognized as a model action. Regarding features, the issue of representation of features has mainly been ignored.

In this study, we focus our attention on human actions performed by a hand. These actions include: opening and closing overhead cabinets, picking up and putting down a book, picking up and putting down a phone, erasing a whiteboard, etc. While performing an action, a hand essentially generates a 3-D trajectory in (x,y,z) space with respect to time. Our aim is to first compute a compact representation of 2-D projection of this trajectory from a video sequence, and then to learn human actions using this representation.

In this paper, we first present a new representation scheme based on sptiotemporal curvature of a trajectory. A trajectory is represented by a sequence of *dynamic instants* and *intervals*. This representation is then used to automatically learn human actions. The system starts with no model, and incrementally builds and refines models by watching people perform actions. Ultimately the system is able to recognize new actions using the learned actions. We present results on a video sequence depicting five different people performing roughly 50 different actions. The system is automatically able to segment the video into different actions, and learn actions.

2. Hand trajectories

In this section, we discuss how to compute motion trajectories from video sequences. In our method, hand is located in each frame, and centroids of a hand in each frame are connected to obtain a trajectory.

We apply skin detection [6] to locate a region corresponding to the hand in an image sequence. Skin detection uses pixel color value. Based on the color predicate, the system labels the incoming pixel as skin or non-skin. During the training phase a color histogram is generated. The pixels are manually labeled as skin or non-skin, and a 3-D Gaussian function for every pixel is generated. If the given pixel is a skin pixel, then a wide Gaussian (• = 2) is added to the color histogram. If the pixel is labeled as non-skin pixel, then a narrow Gaussian (• = 1) is subtracted from the color histogram. At the end of training, a threshold is applied to color histogram in order to divide histogram bins into skin and non-skin. This way, a predicate is generated, which takes a color pixel value as an input, and outputs the skin or non-skin label, based on which histogram bin the pixel color falls in. Then during detection, we just check the pixel flags in color predicate to decide its label. This process runs very fast, since only lookup table operations are involved.

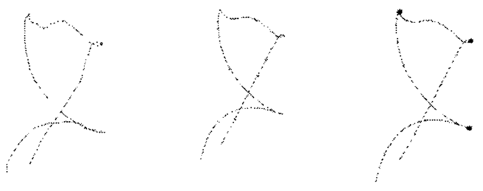
After skin detection, a connected component algorithm is applied to obtain largest connected component, which is hypothesized to be a hand. We assume that the only skin color object is hand. However, if this is not the case, then we can introduce some other constraint, for example fastest moving skin region is a hand. Next, the centroid of this skin region is computed for each frame, and trajectory of hand is created by joining the centroids.

2.1. Smoothing trajectories

A trajectory is a spatiotemporal curve defined as: $(x[1],y[1],1)$, $(x[2],y[2],2),\ldots,$ $(x[n],y[n],n)$. There are essentially two functions: $x(t)$ and $y(t)$ in the definition of a trajectory. The trajectory for action "opening overhead cabinet" is shown in the Figure 2.a. This trajectory contains some noise due to errors in skin detection, lighting conditions, projection distortions, occlusion, etc. Also, since the centroid of hand region is not always a true centroid of a hand, the trajectory obtained by connecting centroids of skin regions contains some errors. In order to deal with this noise, we use anisotropic diffusion to smooth $x(t)$ and $y(t)$ coordinates of trajectory. Anisotropic diffusion was proposed in the context of scale space [4]. This method iteratively smoothes the data (I) with a Gaussian kernel, but adaptively changes the variance of Gaussian based on the gradient of a signal at a current point as follows:

$$I_i^{t+1} = I_i^t + \lambda[c_N \bullet \nabla_N I + c_S \bullet \nabla_S I]_i^t \quad (1)$$

where $0 \leq \lambda \leq 1/4$. We choose λ=0.2 in our experiments. t represents the iteration number, and



(a) (b) (c)

Figure 3:"Opening overhead cabinet" trajectory (a) smoothed version of the trajectory (b) *dynamic instants* (marked by "*") and *intervals* (c)

$$\begin{aligned} \nabla_N I_i &\equiv I_{i-1} - I_i \\ \nabla_S I_i &\equiv I_{i+1} - I_i \end{aligned} \quad (2)$$

The conduction coefficients are updated at every iteration as a function of the gradient:

$$\begin{aligned} c_N^t &= g(|\nabla_N I_i^t|) \\ c_S^t &= g(|\nabla_S I_i^t|) \end{aligned} \quad (3)$$

where $g(\nabla I) = e^{-(\|\nabla I\|/k)^2}$.

The constant k can be fixed either manually at some fixed value, or can be estimated from the "noise estimator". We choose k=10 in our experiments

Figure 3.b shows a trajectory after anisotropic diffusion of x and y coordinates. Notice that now the trajectory is much smoother.

2.2. Computing spatiotemporal curvature

We use spatiotemporal curvature to compute view invariant representation of an action. The spatiotemporal curvature of a trajectory is computed by a method described by Besl and Jain [5]. In this case, a 1D version of the quadratic surface fitting procedure is used. The spatiotemporal curvature, k is given as follows:

$$k = \frac{\sqrt{A^2 + B^2 + C^2}}{\left((x')^2 + (y')^2 + (t')^2\right)^{3/2}}$$

$$\text{where} \quad (4)$$

$$A = \begin{vmatrix} y' & t' \\ y'' & t'' \end{vmatrix}, B = \begin{vmatrix} t' & x' \\ t'' & x'' \end{vmatrix}, C = \begin{vmatrix} x' & y' \\ x'' & y'' \end{vmatrix}.$$

The notation $|\bullet|$ denotes the determinant, and

$$x'(t) = x(t) - x(t-1), x''(t) = x'(t) - x'(t-1). \quad (5)$$

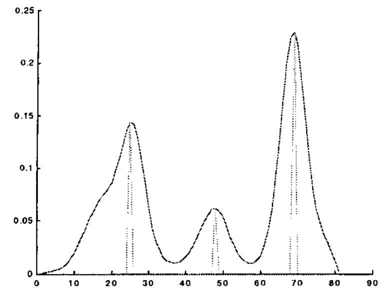


Figure 4. Spatiotemporal curvature, and detected maxima (*dynamic instants*) in "opening overhead cabinet" trajectory.

Since the time interval is constant, so $t'=1$, and $t''=0$.

Spatiotemporal curvature captures both the speed and direction changes in one quantity. Moreover, a special case of spatiotemporal curvature when $t' = t'' = 0$, in the above equation is the spatial curvature, commonly used in 2-D shape analysis, so that the time information is ignored. The sptiotemporal curvature of "opening overhead cabinet" trajectory is shown in Figure 4.

3. Representation

Representation is very important and sometimes difficult aspect of an intelligent system. The representation is an abstraction of a sensory data, which should reflect a real world situation, be view-invariant and compact, and be reliable for latter processing. We propose a new representation scheme based on sptiotemporal curvature of a trajectory. A trajectory is represented by a sequence of *dynamic instants* and *intervals*. A *dynamic instant* is an instantaneous entity, which occurs for only one frame, and represents an important change in motion characteristic: speed, direction, acceleration, and curvature. An *instant* is detected by identifying maxima (a zerocrossing in a first derivative) in the spatiotemporal curvature. An *interval* represents the time-period between any two *dynamic instants*, during which the motion characteristics pretty much remain constant. In our representation, *instants* and *intervals* have physical meanings. Therefore, it is possible to explain an action as a sequence of meaningful instants and intervals.

Dynamic instants and *intervals* for "opening overhead cabinet" action are shown in Figure 3.c.

A dynamic instant is characterized by a frame number, the image location, and the sign. The frame number tells us precisely in which frame, the dynamic instant occurs; the image location provides the location of the hand in the image when the dynamic event occurs; and the sign represents the sign of change of motion characteristic at the instant. The intervals are described by an average sptiotemporal curvature. Examples of dynamic instants include: touching, twisting, loosening; and the examples of intervals include approaching, lifting, pushing, and receding. Consider an opening overhead cabinet action (Figure 4.c). This action can be described as: hand approaches the cabinet ("approaching" interval), hand makes a contact with the cabinet ("touching" instant), hand lifts the cabinet door ("lifting" interval), hand twists ("twisting" instant) the wrist, hand pushes ("pushing" interval) the cabinet door in, hand breaks the contact ("loosening" instant) with the door, and finally hand recedes ("receding" interval) from the cabinet. Similarly, "picking up a phone" action can be explained by two *intervals* and one *dynamic instant* as: hand approaches the phone ("approaching" interval), hand touches the phone ("touching" instant), and finally hand lifts up the phone towards the ear ("lifting" inverval).

3.1. View invariance

It is very important for a representation of action to be view invariant. Since an action takes place in 3-D, and is projected on 2-D image, depending on the viewpoint of the camera the projected 2-D trajectory may vary. Therefore, trajectories of the same action may have very different trajectories, and trajectories of different actions may look the same. This may create a problem in interpretation of trajectories at the higher level. However, if the representation of action only captures characteristics, which are view invariant, then the higher level interpretation can proceed without any ambiguity. Instants, which are the maxima in spatiotemporal curvature of a trajectory are view-invariant. A dynamic instant in 3-D is always projected as a dynamic instant in 2-D, except in limited cases of accidental alignment. By accidental alignment, we mean a viewpoint, which is parallel to the plane, where the action is being performed. In that case, the centroid of hand in all frames is projected at the same location in image plane, resulting in a 2-D trajectory, which is essentially a single point. In Figure 1.a, we show trajectories of opening overhead cabinet action from several viewpoints. Even though these trajectories look quite different, in all cases three dynamic instants are detected by the proposed method.

The trajectories of the same action from different viewpoints look different even though all of them contain the same number of instants, because the location of instants and intervals are different. In order to deal with this view dependence, we propose a notion of a *normal view*. For each action, an arbitrary view is selected as a *normal view*, and the representation consisting of instants and intervals is computed. The trajectory of the same action performed under the camera view different from the normal view will still contain the same number of instants, but the characteristics of intervals may vary. We propose to use correspondence between instants in a *normal* and a novel view to fit the affine transformation. Since the order and number of instants in both trajectories are the same, the correspondence can easily be determined. Once the affine transformation is computed, the trajectory can be transformed into the normal view using this affine transformation. Figure 2 shows the trajectories of opening and closing overhead cabinet, transformed to the *normal views*. Compare these with trajectories shown in Figure 1.

Note that in order to use affine transformation at least three point correspondences are needed. In our approach, we do not employ the start and end instants, since they may vary depending on the field of view of camera, and the variations due to approach and receding intervals. Therefore, the affine transformation can only be applied to actions consisting of five or more instants. We also want to mention here that the projective transformation will be better than affine transformation, however we can only apply the projective transformation to actions containing six or more instants.

4. Learning

Once representation has been defined, the next step is to use this representation to learn human actions. As stated earlier, our aim is to start with no model, and incrementally build model of actions by continuously watching. This is the way, we believe, how children learn to recognize different actions by repeatedly observing adults perform actions.

We assume the camera is fixed, however, people can enter the field of view from any side. The system is continuously analyzing video stream captured by the camera. The system detects hand using skin detection, determines hand trajectory, and computes a view invariant representation of each action.

The continuous video stream can be easily segmented into individual actions. One particular action begins as soon as the hand enters the field of view, and ends when the hand goes out of the field of view. When the system detects the hand again, the second action begins, and so on.

For each action, the system builds a view-invariant representation, and places it into a corresponding category of actions, depending on the number of instants. The system also compares each action with all other actions in

At the higher level of abstraction, the system also determines sets of similar actions based on the match scores. For example, different instances of "opening overhead cabinet" action can be automatically determined to be similar. For each such set only one prototype representation is maintained, since all other instances convey the same information. For each prototype we associate a confidence, which is proportional to the cardinality of the set represented by this prototype. When more evidence is gathered, the confidence of some actions is increased, while the confidence of others remain the same. The prototypes with small confidence can ultimately be eliminated.

The smallest action consists of three instants. Since, currently we are considering actions performed by a single hand only, the actions consisting of three instants usually are either "pick up an object", or "put down an object". As we stated earlier we do not use beginning and end instant in affine warping, therefore for a three instant action, we are left with only one instant which is not enough for affine warping. Consequently, we interpret "pick up" and "put down" actions as follows.

We subtract the frame corresponding to the first instant from the frame corresponding to the last instant, and compute absolute difference for each pixel. Then we apply Gaussian mask centered around the location of hand in the image corresponding to the second instant This step will emphasize the pixels in the neighborhood of the location of hand during the pick up or put down. Then threshold is applied to the weighted difference picture to identify relevant pixels in the middle frame. The window enclosing such pixels signifies the region in the image, which have changed due to pick or put down action. Next, we need to determine if this action is a pick up or put down. We apply edge detector to pixels in the window in the frames corresponding to the first and last instants, and compute a difference of two edge images. Note that edge images are binary images, the difference picture will consists of pixels with value, *0*, *-1* or *1*. The sum of these pixels is determined, if the sum is *+ve* then the action is pick up, if it is *–ve* then it is put down, if it is zero then nothing was picked up or put down. Please see Figure 6 and 7 for representative examples of pick up and put down actions.

4.1. Matching

Given two viewpoint invariant representations of some actions, how can we determine if these are the same actions? It is obvious that two actions with different number of instants cannot be the same. Therefore, we should only match representations, with equal number of instants. We want to note that one action can be a sub-action of the other. In this case, these actions won't have the equal number of instants, however, this match is meaningful. At this point, we are not going to deal with it.

We propose to use a simple method for matching two representations. We compute the average difference between locations of instants. Assume actions we want to match are represented by location of instants:

$$((x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)) \text{ and}$$
$$((x'_1, y'_1), (x'_2, y'_2) \ldots, (x'_n, y'_n))$$

Then the match score is computed:

$$\eta = \sum_i (x_i - x'_i)^2 + (y_i - y'_i)^2 \qquad (6)$$

We compare each action with all other actions with the same number of instants, and compute the match score η. For each action, we need to select closely matched actions. All the matches, which are below some threshold are eliminated first, and only three best matches for each action are maintained. Also if a particular action does not match closely to any action of its category then it is declared as a unique action. Its label may change as more evidence is gathered.

The best matches for individual actions are merged into a compact list using transitive property. That is, if action 1 is similar to actions: 14, 21, and 29; and action 4 is similar to actions: 43, 1, and 14; then actions: 1, 4, 14, 21, 29, and 43 are all similar actions due to transitive property.

5. Experiments

We digitized a 8-minute video clip recorded at 24 fps captured by a stationary camera consisting of more than 11000 frames. Five people performed total of 48 different actions in front of the camera, the complete list of actions is given Table 2, and two representative video sequences are shown in Figure 8 and 9. People were not given any instructions, and entered and exited from arbitrary directions. Therefore, the viewpoints of these actions were very different. The system automatically detected hand using skin detection, generated trajectories of actions.

The actions were segmented by the system into 48 actions, and are shown in Figure 7. Trajectories of these actions were used to generate the view invariant representation proposed in this paper. These

Table 1. Interpretation results. The bold face font in column indicates incorrect match.

Action index	**Three Best matches**	**Evaluation & comments**
1	14 21 29	Correct
2	Pick up	Correct
3	18 6 23	Correct
4	43 1 14	Correct
5		Unique action
6	3 18 23	Correct
7		colinear points.
8		colinear points.
9	Pick up	Correct
10	Put down	Correct
11	Pick up	Correct
12	Put down	Correct
13		Unique action
14	1 29 38	Correct
15		Unique action
16	38 21 29	Correct
17	**Pick up**	Incorrect, object hidden
18	3 23 32	Correct
19	Pick up	Correct
20		Unique random motion
21	16 38 29	Correct
22	Pick up	Correct
23	3 18 6	Correct
24	Pick up	Correct
25	Put down	Correct
26		Unique action
27		Unique action
28	**42 5 16**	Incorrect
29	38 16 21	Correct
30	39	Correct
31	36 **16 38**	Two incorrects
32		Unique action
33		Incorrect, colinear points
34		Random motion, unique
35	Put down	The action is confusing
36	31 **16 42**	Two incorrects
37		Unique
38	16 29 21	Correct
39	30	Correct
40	46 **15**	One incorrect
41		Unique action
42		Unique action
43	14 29 1	Correct
44	**Pick up**	Incorrect, object too small
45		Unique action
46	40 **15**	One incorrect
47		Unique action
48	**36 31 42**	Incorrect, colinear points.

representations were interpreted by the system to learn these actions.

There were eleven actions consisting of three instants. The system attempted to interpret these actions either "pick up" or "put down" actions as describe in section 4. All except two actions (35 and 44) were correctly interpreted. Action 35 is quite confusing as it is clear from its trajectory shown in Figure 7. In action 44, the object which was picked up occupied too small region in the image, therefore it was not detected.

The remaining actions contained four to thirteen instants. Each of these actions, was matched using method discussed in section 4.1. The results are shown in Table 1. We are pleasantly surprised to see our simple matching technique worked quite well. Only two matches were completely wrong (actions 28 and 48). Five matches (31, 40, 36, 46) were partially incorrect. In action 48, the instants were collinear, therefore they did not provide independent constraint for the affine transformation. Action 28, has an extra instant, which created problem.

Note that these matches are based on only single instant of an action, therefore the performance of our approach is remarkable.

The system was able to learn that actions 1, 14, 16, 21, 29, 43, and 38 are the same. Note that even though trajectories of these actions shown in Figure 7, are different, but due to the strength of our representation, the system was able to learn they represent the same action. Similarly, the system was able to discover that actions 3, 18, 6, 23, and 32, which represent "put down the object, and then close the door", are all the same using matching and the transitive property. Therefore, the confidence for this action is quite large.

Several actions were identified as unique, because they did not match well with other actions having the same number of instants. Therefore, their confidence is quite low. Since we assume that the system is continuously watching in its field of view, if more instances of these unique actions are performed , the system will be able to increase the confidence.

6. Related work

Siskind and Morris [7] use HMMs to classify 6 gestures: pick up, put down, push, pull, drop, and throw. This requires training, and features used are not view invariant. Kojima et al [10] propose an approach to generate a natural language descriptions of human behavior from real video images. First, a head region of a human is extracted from each frame. Then, using a model-based method, 3-D pose and position of head are estimated. Next, the trajectory of head is divided into segments, and the most suitable verb is selected. Bobick and Davis [8] describe a method to recognize aerobic exercises from video sequences. They need training, and multiple views to peform recognition. Stauffer and Grimson [9] use simple classification based on aspect ratio of tracked objects. Seitz and Dyer [11] proposed an affine view-invariant trajectory matching method to analysis cyclic motion.

7. References

[1] Polana, R.., "Temporal textures and activity recognition" ,Ph.D. thesis, University of Rochester.

[2] Koller, D., Heinze, D, and Nagel, H-H. "Algorithmic characterization of vehicle trajectories from image sequences by motion verbs", CVPR-91, pp 90-95.

[3]. Tsotsos, J.K., et al, "A Framework for visual motion understanding", IEEE PAMI, 2(6):563-573, Novr, 1980.

[4]. Pietro Perona and Jitendra Malik, "Scale-space and Edge Detection Using Anisotropic Diffusion", IEEE PAMI, vol. 12 No. 7. July 1990.

[5] Besl, P. J., and Jain, R. C., "Invariant surface characteristics for 3D object recognition in range images", CVGIP, 33, 1986, 33-80.

[6] R Kjeldesn andJ Kender, "Finding skin in color images", Int workshop on Automatic face and gesture recogn, pp 312-ition317, 1996.

[7] Siskind J., M., and Moris, Q., "A maximum likelihood approach to visual event classification", ECCV-96, 347-360.

[8] Aaron Bobick and James W. Davis. Action recognition using temporal templates, pages 125--146.CVPR-97, 1997.

[9] C. Stauffer and W.E.L. Grimson, "Learning patterns of activity using real-time tracking", PAMI, 22(8):747--757, August 2000.

[10] M. Izumi A. Kojiama "Generating natural language description of human behavior from video images", ICPR-2000, 4: 728--731, 2000.

[11] S. M. Seitz and C. R. Dyer. *View-invariant analysis of cyclic motion.* International Journal of Computer Vision, 25:1--25, 1997.

Table 2: List of actions.

1st open the cabinet
2nd pick up an object (umbrala) from the cabinet.
3rd put down the object in cabinet, then close the door.
4th open the cabinet, with touching the door an extra time.
5th pick up an object (disks) with twisting hand around.
6th put back the object (disks) and then close the door.
7th open the cabinet door, wait, then close the door.
8th open the cabinet door, wait, then close the door.
9th pick up an object from top the of the cabinet.
10th put the object back to the top of cabinet.
11th pick up an object from the desk.
12th put the object back to the desk.
13th pick up an object, then make random motions.
14th open the cabinet.
15th pick up an object, put it in the cabinet, then close the door.
16th open the cabinet.
17th pick up an object (umbralla) from the cabinet.
18th put the object (umbralla) back to the cabinet.
19th pick up a bag from the desk.
20th make random motions.
21st open the cabinet.
22nd pick up an object (a bag of disks).
23rd put donw an object (a bag of disks) back to the cabinet, then close the door.
24th pick up an object from the top of the cabinet.
25th put the object back to the cabinet top.
26th make random motions with two hands.
27th continue the action 26.
28th close the door, with some random motion.
29th open the cabinet.
30th pick up an object (remote controller) from the cabinet, put it down on the desk, pick up another object (pencil) from the desk, put it in the cabinet, then close the door.
31st open the cabinet door, with the door half pushed, pick up an object (pencil) from the cabinet.
32nd pick up an object (remote controller) from the desk, put it in the cabinet, then close the door.
33rd open the cabinet door, wait, then close the door.
34th open the cabinet door, make random motions, then close the door.
35th pick up some objects.
36th open the door, pick up an object, with the door half opened.
37th close the half opened door.
38th open the cabinet door.
39th pick up an object, move it within the cabinet, pick up another object, move it, then close the door.
40th open the cabinet door, wait, then close the door.
41st pick up an object from the top of the cabinet.
42nd close the cabinet.
43rd open the cabinet.
44th put down a disk.
45th close the half closed door.
46th open the door, wait, then close the door.
47th open the cabinet door, pick up an object, then put it back, then close the cabinet door.
48th open, then close the cabinet door.

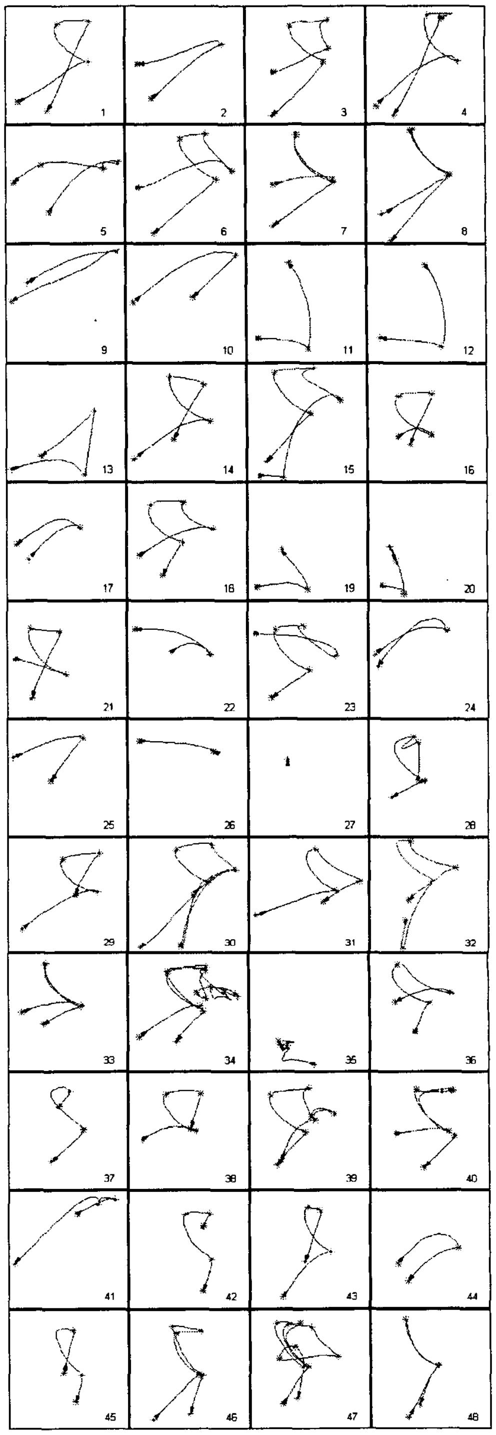


Figure 7. Trajectories of all 48 actions. The instants are shown with red "*".

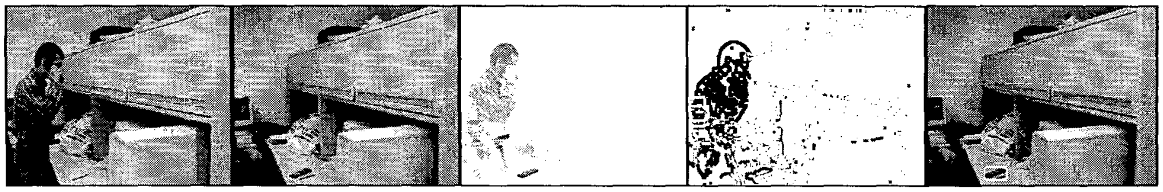


Figure 6: Action 10, put down. From left to right: first image, last image, the difference picture of two images, difference of two edge images (the blue pixels represent –1 and red pixels represent +1, white is 0.), small window (white box) shows the object.

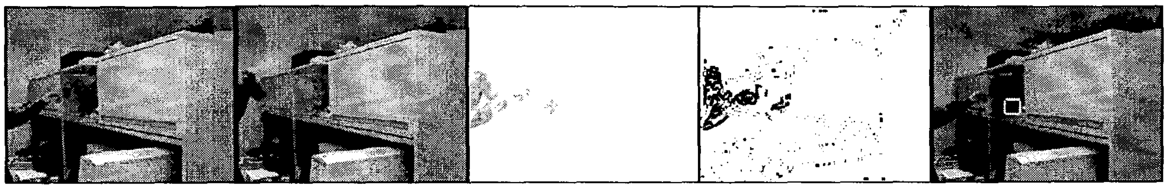


Figure 7: Action 11, pick up. Captions are the same as in Figure 8.

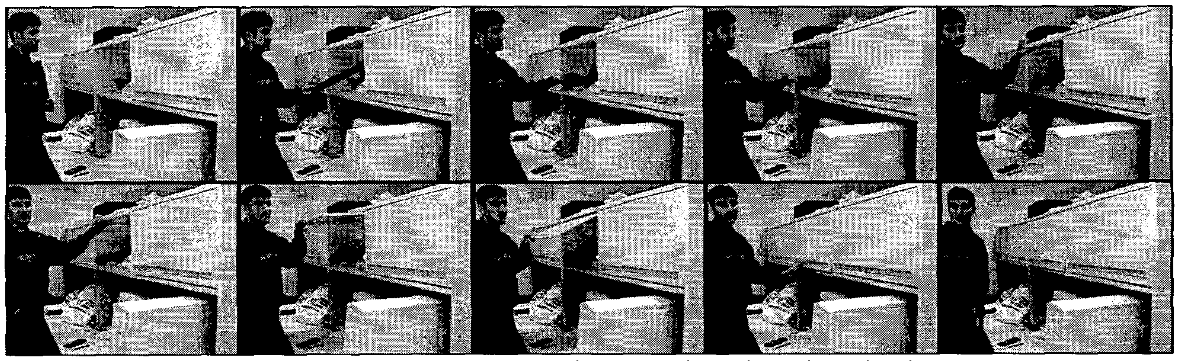


Figure 8. Sequence showing Action 3, put down the object in cabinet, then close the door.

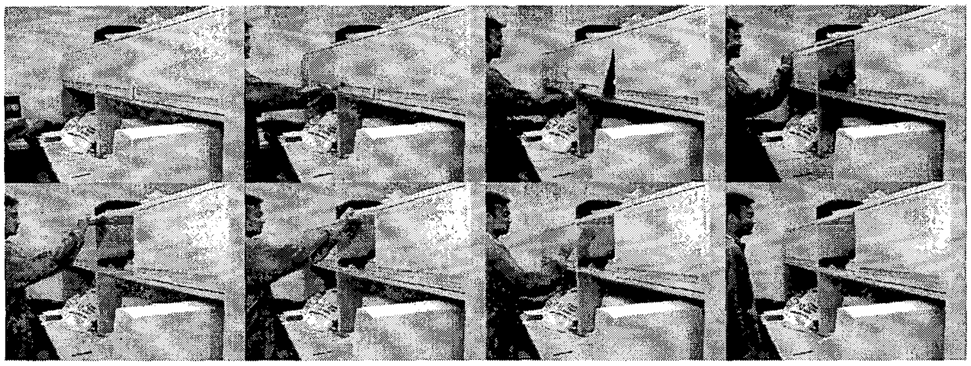


Figure 9. Sequence showing Action 29, opening a cabinet.