Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Project No. 611391

DREAM

Development of Robot-enhanced Therapy for Children with Autism Spectrum Disorders

Agreement Type: Collaborative Project
Agreement Number: 611391

D4.2 Evaluation of multi-sensory data perception

Due Date: **01/04/2015**
Submission date: **01/04/2015**

Start date of project: **01/04/2014** Duration: **54 months**

Organisation name of lead contractor for this deliverable: **University of Portsmouth**

Responsible Person: **Honghai Liu** Revision: **1.0**

Project co-funded by the European Commission within the Seventh Framework Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including the Commission Service)	
RE	Restricted to a group specified by the consortium (including the Commission Service)	
CO	Confidential, only for members of the consortium (including the Commission Service)	

Contents

Executive Summary

Deliverable D4.2 presents evaluation of multi-modal sensory perception, including algorithm specification, design, implementation, and validation of a suite of multi-modal sensory data acquisition modules derived from the sensory requirements set out in deliverable D1.1 Interaction definition (with particular regard to the sensory cues that characterize the action triggers, action components, and action goal states.)

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order)

Haibin Cai, University of Portsmouth
Yinfeng Fang, University of Portsmouth
Dongxu Gao, University of Portsmouth
Xiaodong Jiang, University of Portsmouth
Zhaojie Ju, University of Portsmouth
Honghai Liu, University of Portsmouth
Ting Wang, University of Portsmouth
Yiming Wang, University of Portsmouth
Hui Yu, University of Portsmouth
Wei Zeng, University of Portsmouth
Shu Zhang, University of Portsmouth

Revision History

Version 1.0 (Ju, Z., Cai, H., Gao, D., Y., Wang, Y., Zhang, S., 28-03-2015)

Introduction

This deliverable describes the process and analyse of the signals, including images and sound, obtained from the multi-camera system designed, and use them for further multi-data fusion. To achieve acceptable performance in sensing and data interpretation, practical issues have to be prioritized when selecting data pre-processing and multisensory fusion methods from a variety of estimation, classification, and inference methods. Since regions of interest for detecting eye gaze, head and human body have been decided experimentally, trade-off will be handled to achieve the balance between efficiency and effectiveness of the multi-camera data acquisition and interpretation. Evaluation of multiple sensor fusion, camera pose estimation, gaze estimation, body action analysis, expression recognition, object tracking, and speech recognition is presented.

Multiple Sensors Capturing and Fusion

To synchronise and fuse the multiple sensor data, a framework for coordinating multiple sensors is presented in Figure 1. This project employs five individual sensors: Camera 1, Camera 2, Camera3, Kinect 1 and Kinect 2.

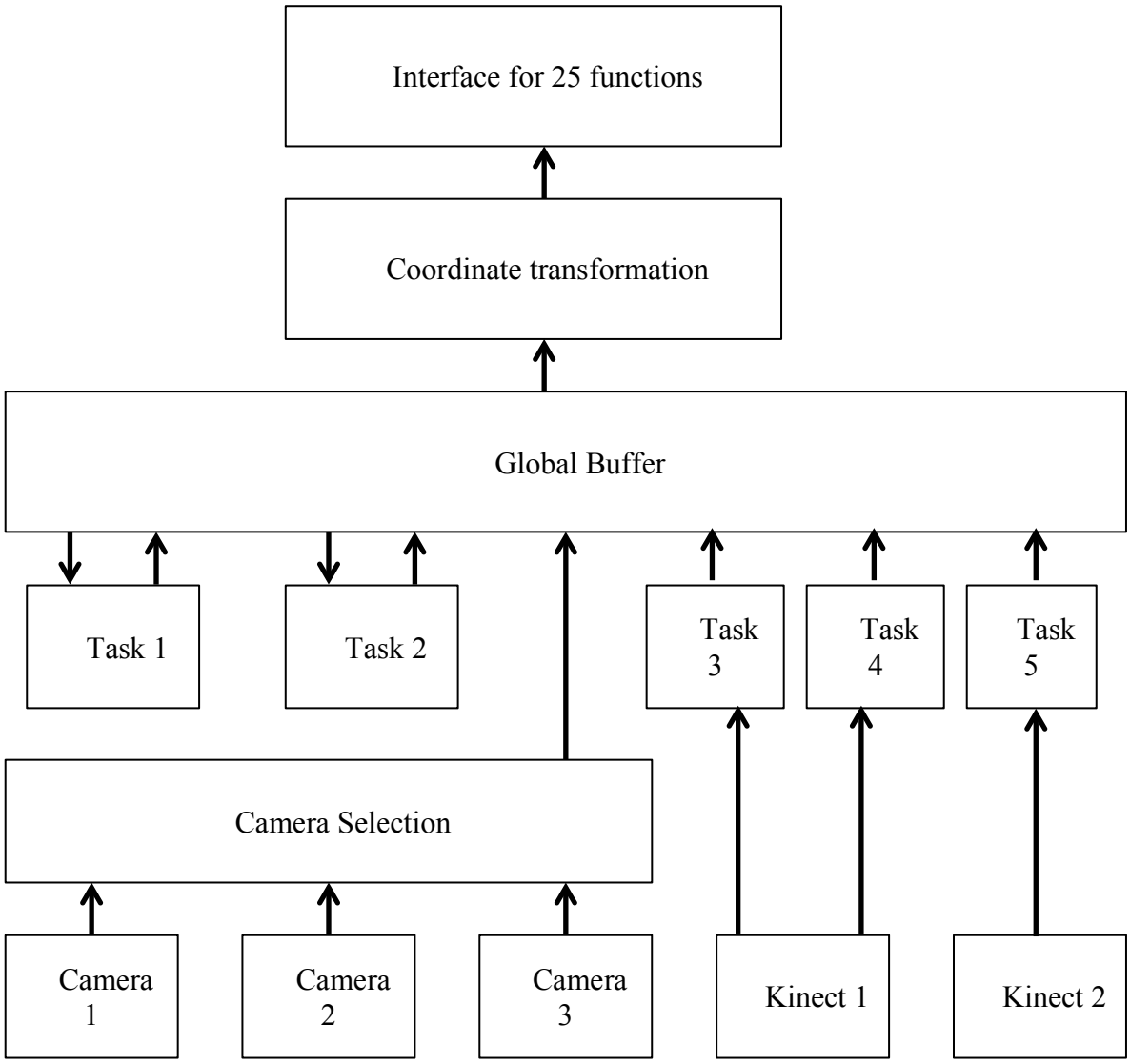


Figure 1: a framework for coordinating multiple sensors

Camera Selection Module

Camera 1, Camera 2 and Camera 3 form a functional unit to get the face location, eye locations, gaze direction, head direction, etc. The Camera Selection Module (CSM) captures

image frames from three sensors and selects one camera by the highest face detection probability, and meanwhile CSM module also functions to obtain facial feature points from the selected frame. The selected camera ID, the original frame and the calculated feature points will be simultaneously saved in the Global Buffer and be updated according to the speed (fps). Task 1 and task 2 serve to implement the primary functions, like calculating face/eye location, head/gaze directions, face ID, facial expression ID and etc. These separated tasks will run through the main algorithms, which has been proposed/employed by this project. It should be noted that Task 1 and Task 2 could communicate with each other by sharing data in the global buffer.

The function of Kinect1 are two-folds: voice analysis (task 3) and subject's skeleton joints extraction (task 4). Task 3 can be further separated into two parts: speech recognition and speech direction tracking. Kinect2 focuses on object tracking, and the objective is to get the location of an object (toys), object ID and robot's head location.

Global Buffer Module

An example definition of the global buffer is given below, and the number before each variable will be used in Table 1.

global_buffer

{

V1: *cv::VideoCapture cap;*
V2: *CameradeviceID*
V3: *Mat ::image*
V4: *Eyes(rightx, righty, rightz, left x,left y, left z)*
V5: *Head pose(roll, yaw, pitch)*
V6: *Face(vector(x,y,z))*
V7: *Gaze(roll, yaw, pitch)*
V8: *Coordinate transform Mat(R,T)---camera1,camera2,camera3,Kinect2*
V9: *Frame 3D points(x,y,z)*
V10: *Object position(x,y,z)*
V11: *Head position(x,y,z)*
V12: *Hand position(x,y,z)*
V13: *cv::Mat X; //face 49 feature points*
V14: *int numberkernel; //number of kernel used for eye center detection*
V15: *vector<cv::Mat> kernel_filter; //kernel used for eye center detection*
V16: *Robot head position*
V17: *Sound Direction*
V18: *Face sorce*
V19: *Desk_point vector // 3 points*
V20: *Skelton joint vector*
V21: *Object_location*
V22: *Object_id*
V23: *Face_id*
V24: *Face_expression_id*
V25: *Object_id*
V26: *Object_history_location vector*

V27: *Voice_descriptor_id*
V28: *Voice_text_id*
V29: *Skelton_history_joint vector*

}

Coordinate Transformation Module

In the coordinate transformation component, we need to determine the position and orientation of each camera, given its intrinsic parameters and a set of n correspondences between 3D points and their 2D projections. Our implementation is based on Efficient Perspective-n-Points (EPnP) algorithm proposed by Vincent, Francesc and Pascal [1], with the 2D-3D correspondence obtained by Kinects and Cameras. Our camera pose estimation method has robust results for different camera poses. It only requires the user to mark the corresponding points between the Kinect image and Camera image manually for about 20 pairs. This approach is preferred as it's far more reliable than any other feature-matching algorithm. With the intrinsic parameters of the cameras, the poses of those cameras related to the Kinect can be found reliably. And we also provide an interface to transform the rotation across different coordinate systems.

The Class and Its Support File

Class:

CalibrateObjects
KinectToolsWithOpenni2
CoordTransform

Class Interfaces:

// *Read configuration parameters from file with extension of ".conf".* // *The standard format of this file can be found in the project folder named "console.conf".* KinectToolsWithOpenni2::ReadConf (std::string config_file_full_path);
// *Main process of the calibration* KinectToolsWithOpenni2::Estimate ();
// *Transform a 3D point in local coordinate system to global coordinate system with local* // *camera's Rotation Matrix and Translation Matrix.* CoordTransform::PointTransform (Local_Point, Global_Point, Cam_Rot_Mat, Cam_Tran_Mat);
// *Transform a 3D point in global coordinate system to local coordinate system with local* // *camera's Rotation Matrix and Translation Matrix.*

CoordTransform::PointInverseTransform (Local_Point, Global_Point, Cam_Rot_Mat, Cam_Tran_Mat);
// Transform a Rotation Matrix in global coordinate system to local coordinate system with local *// camera's Rotation Matrix.* CoordTransform::TransformRotationFromGlobalToLocal(Global_Rot, Local_Rot, Cam_Rot);
// Transform a Rotation Matrix in local coordinate system to global coordinate system with local *// camera's Rotation Matrix.* CoordTransform::TransformRotationFromLocalToGlobal(Local_Rot, Global_Rot, Cam_Rot);
(Noted: Detailed instructions about the members and functions can be found in the source files of each class as comments.)

Support File:

This implementation is based on the viz module of the OpenCV, and OpenNI2

viz Module from OpenCV for point cloud visualization Add directories of "$(OPENCV249_ROOT)\sources\modules\viz\include", and "$(OPENCV249_ROOT)\sources\modules\viz\include\opencv2\viz" to the include path; Add opencv_viz249d.lib and opencv_viz249d.dll to the program for compilation and running.
OpenNi2 for Kinect usage Install the OpenNi2 lib to the system; Add directory of "$(OPENNI2_ROOT)\Include" to the include path; Add directory of "\$(OPENNI2_ROOT)\Lib" to the lib path; Add OpenNI2.lib to the program for compilation and running.

All dependencies can be found in the directory of "Dependency". In each folder in "Dependency", there is a Readme file that contains short instruction.

Functions and Global Variables

While running the program, all the variables would be updated by the tasks and CSM in real-time. With these 29 variables, predefined 25 interface functions can be easily implemented through accessing the variables, arithmetic and coordinate transformation. Table 1 lists relations between these 25 interface functions (described in Deliverable D3.1.3) and the defined variables.

Table 1 Functions and global veriables

25 Interface functions	Related Variables
F1: checkMutualGaze	V7,V8,V17
F2: getArmAngle	V20

F3: getBody	V20
F4: getBodyPose	V20
F5: getEyeGaze	V4,V7,V8,V13
F6: getEyes	V3,V4,V6,V8,V13,V14,V15
F7: getFaces	V3,V6,V8
F8: getGripLocation	V22,V21,V22
F9: getHands	V20
F10: getHead	V6,V8,V13
F11: getHeadGaze1	V6,V8,V11,V13,V19
F12: getHeadGaze2	V6,V8,V13
F13: getObjects1	V21,V22
F14: getObjects2	V19,V21,V22
F15: getObjectTableDistance	V21,V22
F16: getSoundDirection	V8,V17
F17: indntifyFace	V11,V23
F18: IdentifyFaceExpression	V11,V23,V24
F19: identifyObject	V21,V25
F20: identifyTrajectory	V25,V26
F21: identifyVoice	V27
F22: recognizeSpeech	V28
F23: trackFace	V6,V8,V18
F24: trackHand	V29
F25: trackObject	V21,V22

Gaze Estimation

Method

Face Location Method

The boosted cascade face detector is employed with default parameters in order to obtain the approximate location of the face [2]. This method corresponds to the getFaces(x,y,z) function.

Eye Location Method

This correspond to the getEyes(eyeLx, eyeLy, eyeLz, eyeRx, eyeRy, eyeRz) function. In our project, we present an improved integro-differential solution to localize the eye centers.

The proposed method is computationally much cheaper than the original integro-differential method [3] and also achieves a higher accuracy in relatively lower-resolution images.

It is evident that Daugman's method has been highly cited, especially in iris recognition area. It is assumed that an iris is with a circular form and the integro-differential operator is applied as defined:

$$\max_{(r,x_0,y_0)} \left| G_{\square}(r) * \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\square r} ds \right| \qquad (1)$$

Where $G_{\square}(r)$ is a smoothing Gaussian with a scale of $\square$ and $*$ represents convolution, $I(x,y)$ is the image of an eye and ds is the contour of the r radius circle with the center point of (x_0, y_0). The operator searches for the maximum along the circle path in the blurred image via the Gaussian kernel, partial derivative with respect to increasing radius r. In order to deal with the obscure of upper and lower limbus by the eyelids, the angular arc of contour integration s is restricted in range to two opposing 90 cones centered on the horizon.

In discrete implementation of the integro-differential operator, the order of convolution and differentiation is interchanged and concatenated to improve the speed. After replacing the convolution and contour integrals with sums, the equation is derived as follows,

$$\max_{(n\Delta r, x_0, y_0)} = \left| \frac{1}{\Delta r} \sum_k \left\{ \left(G_\sigma\big((n-k)\Delta r\big) - G_\sigma\big((n-k-1)\Delta r\big) \right) * \sum_m I[(k\Delta r \cos(m\Delta\theta) + x_0), (k\Delta r \sin(m\Delta\theta) + y_0)] \right\} \right| \qquad (2)$$

Where Δr means small increment in radius, $\Delta\theta$ is the angular sampling interval along the circular arcs.

Gaze Estimation Method

We propose a real-time feature-base gaze estimation method by constructing multi-sensor fusion system to handle the large head movement. Three cameras and two Kinects are using in this system. All the image data are captured simultaneously by creating 8 handles in programming. Each handle deals with difference kinds of data. The data captured in each handle are two Kinect RGB image data, two Kinect depth data, three camera data and one Kinect audio data. The resolution of the camera, Kinect RGB image, Kinect depth image are 1280*960, 640*480 and 640*480 separately.

In order to estimation the gaze using feature-based methods, the eye center position should be accurately located. The eye location methods can be roughly divided into three categories, namely appearance-based, shape-based and hybrid methods.

We proposed a feature-based method to estimation the gaze direction. Firstly the eye center location is determined using the above method. In order to deal with large head movements, the head poses need to be determined. In pure method, we employ the method proposed by Xiong et al. [4] to locate the feature points in the human face. Then the method of POSIT proposed by Dementhon et al. [5] is used to calculate the direction of head gaze (corresponding to the getHead(headx, heady, headz) function). It should be noted that the gaze direction differs from the head gaze by two angles, the horizontal direction θ and the vertical direction φ. The final gaze direction is finally determined by adding the angles to the

head gaze. The following is the equation to calculate the gaze direction (corresponding to the getEyeGaze(eye, x, y, z) function).

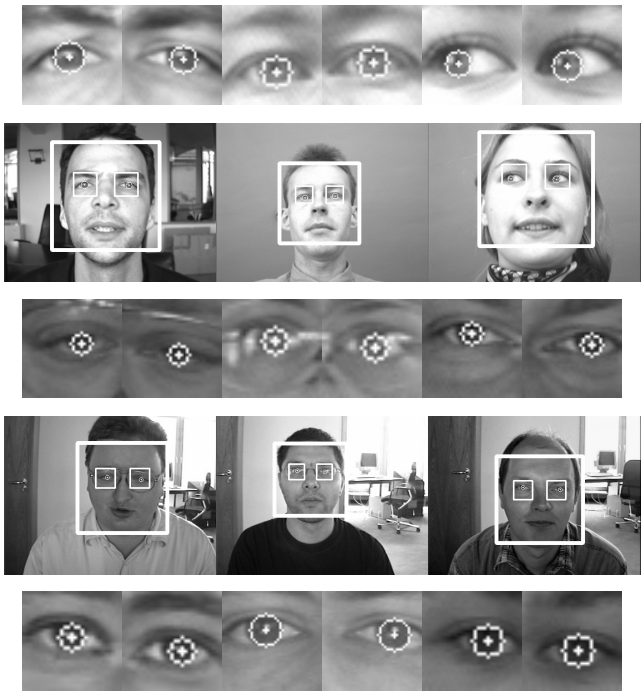
$$\begin{cases} \theta = \tan^{-1}(\gamma * \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} * \frac{\cos\alpha}{L}) \\ \varphi = \tan^{-1}(\varepsilon * \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} * \frac{\cos\beta}{H}) \end{cases} \quad (4)$$

Where (x_c, y_c)means the the center of two eye corners, (x_p, y_p)means the center eye pupils, α is the angle between the line of two eye corners and the line of two centres. β is the complementary angle of α. L is the distance of the two eye corners, γ and ε are determined through experiment.

Results

Eye Location Results

Rough eye regions are extracted through anthropometric relations with the face as stated in [6] and [7]. The proposed method is validated on the BioID face database which consists of 1521 grayscale images of 23 different people with a resolution of 384*288 pixels [8]. The images in the database are head and shoulder frontal view images with a large variety of illumination, background, scale and pose. Some people in the database are wearing glasses. In some images the eyes are closed or completely hidden by reflections on the glasses. Because of these conditions, the BioID database is considered to be challenging and realistic. Some snapshots are provided in Fig.2, the number of detected image is 1469.



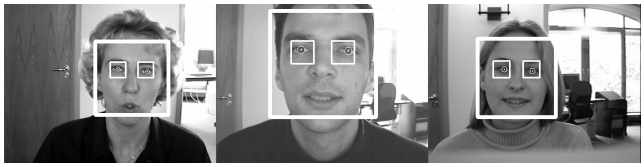
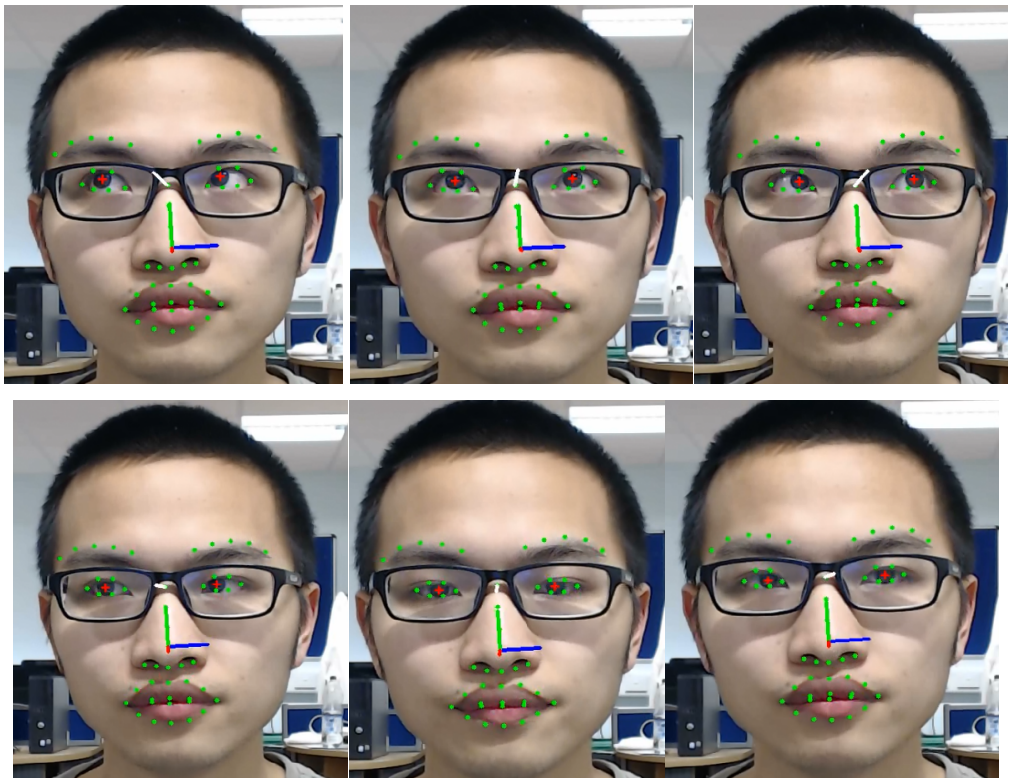
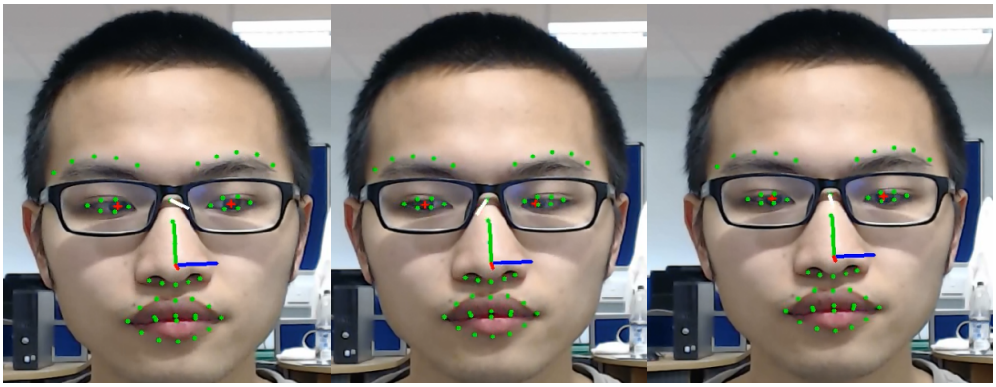


Figure 2 Snapshots with accurate eye center estimation

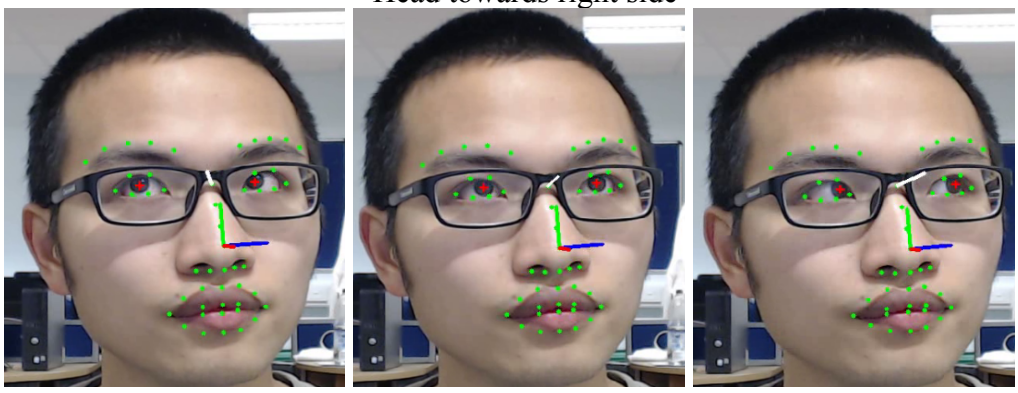
Gaze Estimation Results

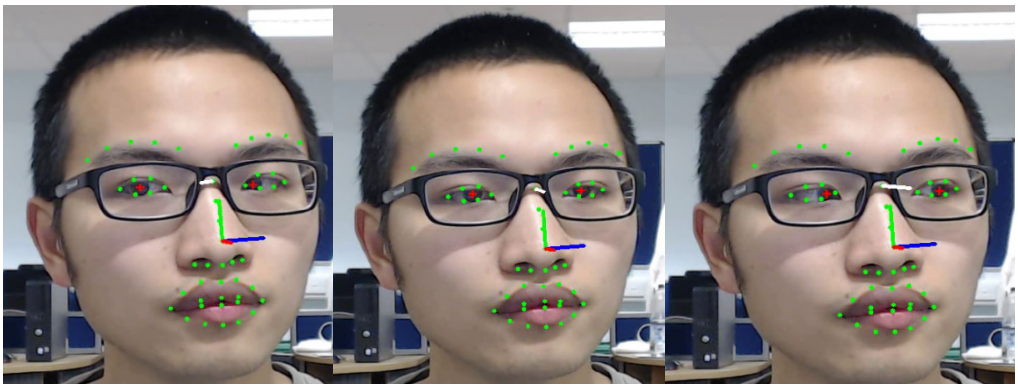
The following image shows the gaze estimation results of our project. The Kinects are used to acquire the color image and also the depth image, three high-resolution cameras are used to acquire high quality color image. And the images we used to calculate the gaze direction are selected according to the angle of head poses. The most frontal face image is the image that has the smallest angle of yaw. This image is then chosen as the final image to estimate the gaze direction.

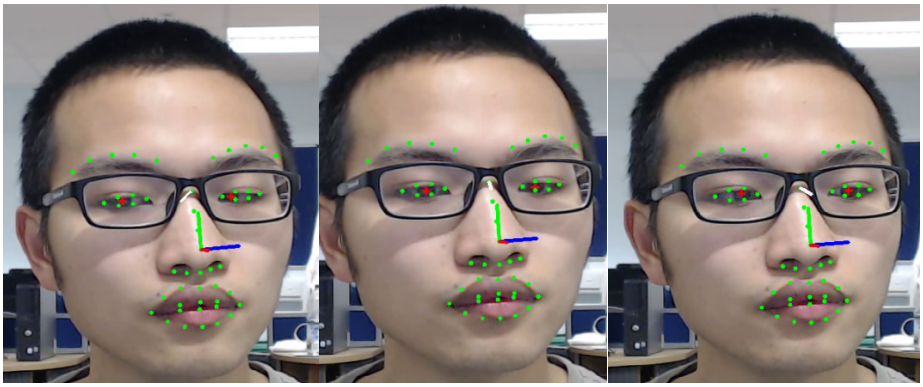




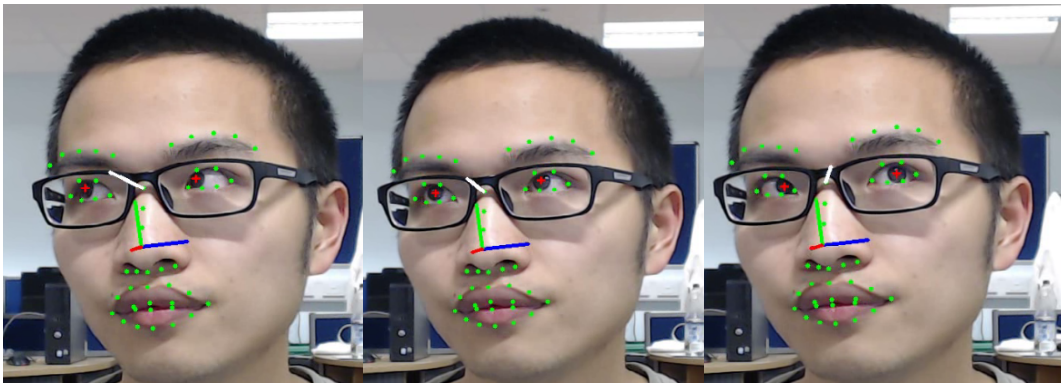
Head towards right side





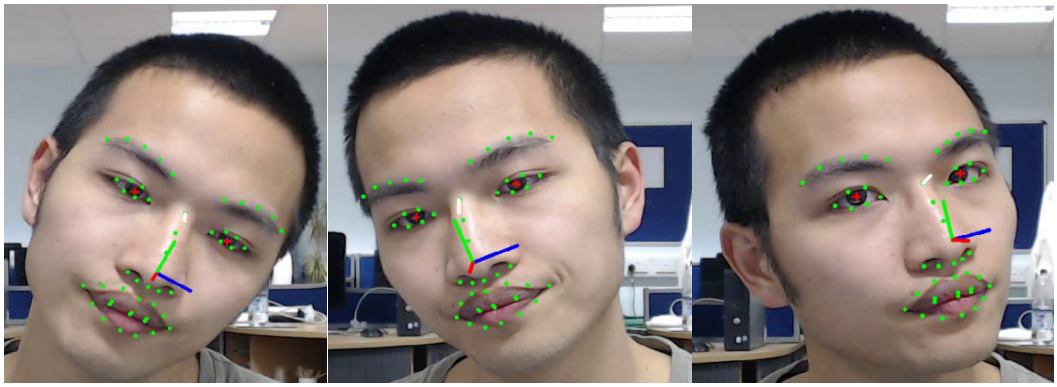


Head toward left side

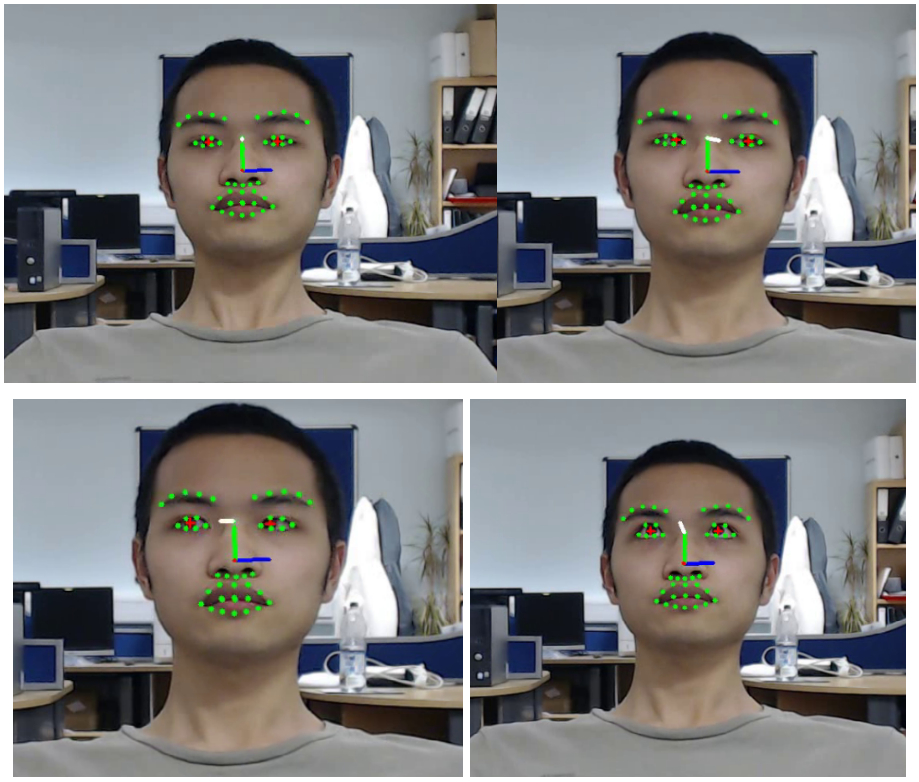


Head rotation

Without glasses



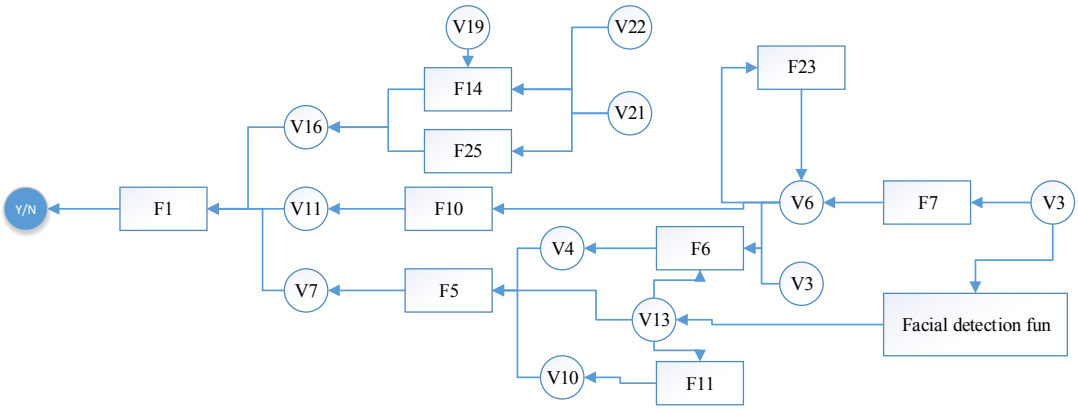
Longer distance



Functions

Related functions are F1, F5, F6, F7, F10, F11, F12 and F23.

Relation of above functions and global variables are shown as follows.



Human Action Analysis

Method

In this project, the 3D joint position feature will be utilised to represent the actions. As these features are invariant to the translation of the human body and robust to noise and temporal misalignment. One key observation is that representing the human movement as the pairwise relative positions of the joints results in more discriminative features. For a human subject, 21 joint positions are tracked by the skeleton tracker and each joint i has 3 coordinates $p_i(t) = (x_i(t), y_i(t), z_i(t))$ at a frame t. The illustration of the skeleton joints are shown in Fig. 1. The coordinates are normalized so that the motion is invariant to the initial body orientation and the body size. For each joint i, we extract the pairwise relative position features by taking the difference between the position of joint i and any other joint j:

$$p_{ij} = p_i - p_j$$

The 3D joint feature for joint i is defined as:

$$p_i = \{p_{ij} | i \neq j\}$$

Although enumerating all the joint pairs introduces some information that may be irrelevant to our classification task, our system is capable of selecting the joints that are most relevant to our recognition task. Relative joint position is actually a quite intuitive way to represent human motions. Consider, for example, the action "waving". It can be interpreted as "arms above the shoulder and moving left and right". This can be effectively characterized through the pairwise relative positions.

Results

Using the skeleton information of Kinect shown as Fig.3 to Fig.5, as well as the 3D joint positon, the action could be obtained.

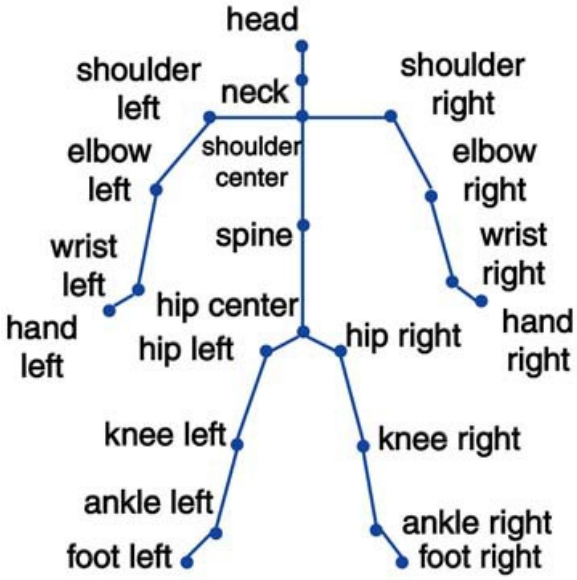


Figure 3 Human joints tracked with the skeleton tracker

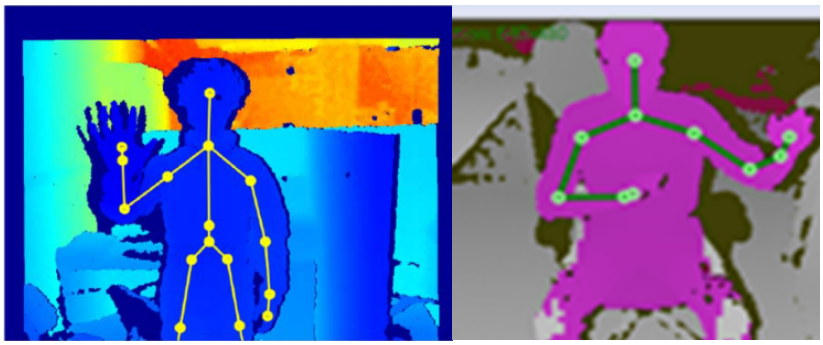


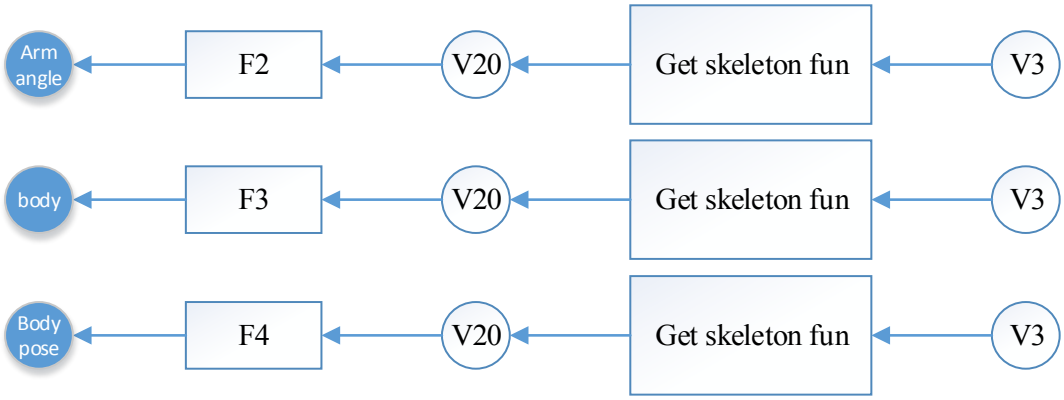
Figure 4 Action recognition using depth information and Kinect skeleton

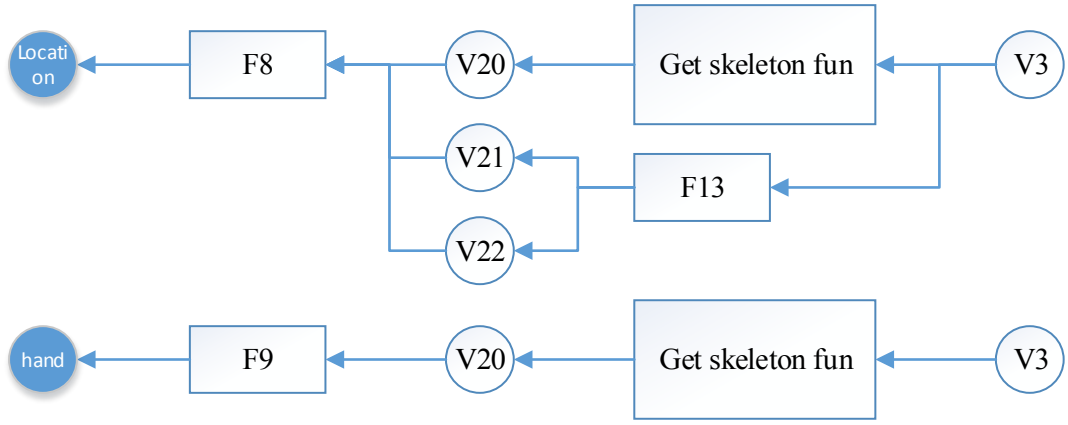
Figure 5 Action recognition using Kinect skeleton

Functions

Related functions are F2, F3, F4, F8, F9 and F24.

Relation of above functions and global variables are shown as follows.





Face Expression Analysis

Method

Local Binary Pattern

LBP is a nonparametric method and has been proved as a powerful descriptor in representing local textural structure [9]. The main advantages of LBP are its strong tolerance against illumination variations and computational simplicity. This method has been successfully used in both spatial and sptio-temporal domains in face recognition and facial expression recognition.

The original LBP operator labels the pixels of an image with decimal numbers. Each pixel is compared with its eight neighbors in a 3×3 neighbourhood, considering the center pixel value as threshold; bigger values are encoded with 1 and the others with 0. A binary number is obtained by concatenating all these values. Its corresponding decimal number is used to compute LBP histogram. Figure 6 shows an example of LBP operator.

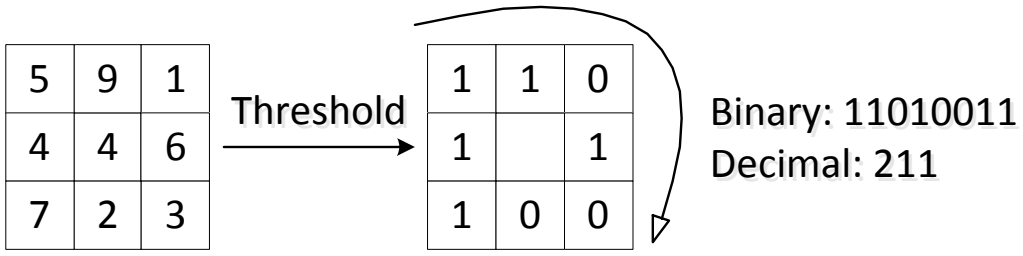


Figure 6. Example of LBP operator.

Support Vector Machine

SVM is considered as one of the most powerful machine learning techniques for data classification. It achieves a good balance between structural complexity and generalization error. It offers great performance under the circumstance of very few training samples, high dimensionality and nonlinear classification.

In a two-class learning task, SVM find a maximal margin hyperplane as its decision boundary. For a linear separable dataset, SVM assumes that the best classification results are

obtained by maximizing the margin of hyperplane between two classes. It allows not only the best partition on the training data, but also leaves much room for the correct classification of the future data. In order to guarantee the maximum margin hyperplanes to be actually found, an SVM classifier attempts to maximize the following function with respect to $\vec{w}$ and b:

$$L_P = \frac{1}{2}\|\vec{w}\| - \sum_{i=1}^{t} \alpha_i y_i (\vec{w} \cdot \vec{x_i} + b) + \sum_{i=1}^{t} \alpha_i$$

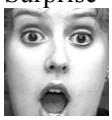
where t is the number of training examples, α_i are the Lagrange multipliers. The vector $\vec{w}$ and constant b define the hyperplane.

SVM only makes binary decisions. For multi-class classification problem, one-vs-rest technique provides a computational simpler and flexible strategy, which trains binary classifiers to one class from all the others

Results

We evaluate our system on CK+ database using 10-fold cross-validation. CK+ is a set of image sequences in which the facial expressions of subjects are displayed from neutral to target emotions. For our experiment, the first neutral face and three peak frame are used, which results in 1236 images (135 Angry, 177 Disgust, 75 Fear, 207 Happy, 84 Sadness 249 Surprise and 309 Neutral). The confusion matrix is shown below.

	Neutral 	Angry 	Disgust 	Fear 	Happy 	Sadness 	Surprise 
Neutral 	50.15	15.32	9.61	3.60	9.61	8.11	3.60
Angry 	26.15	36.15	9.23	4.62	1.00	9.23	4.62
Disgust 	20.44	6.63	56.91	0.55	8.84	4.42	2.21
Fear 	13.33	8.33	6.67	43.33	13.33	10.00	5.00
Happy 	13.04	4.83	8.21	6.76	59.42	3.38	4.35

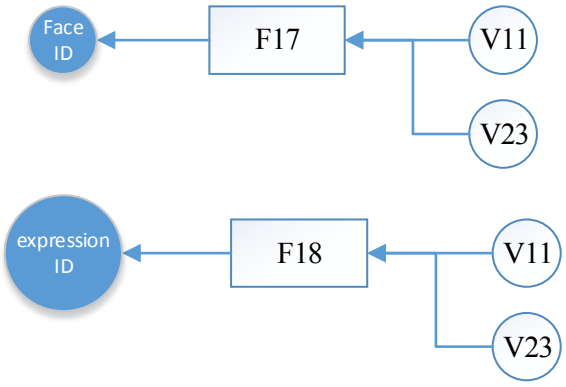
Sadness 	31.17	7.79	10.39	12.99	9.09	24.68	3.90
Surprise 	4.84	1.61	0.40	2.42	3.23	2.02	85.48

The overall recognition rate is 53.15%.

Functions

Related functions are F17 and F18.

Relation of above functions and global variables are shown as follows.



Object Tracking

Method

Despite that numerous algorithms [10] have been proposed in the literature, object tracking remains a challenging problem due to appearance change caused by pose, illumination, occlusion, and motion, among others. In this project, a compressive tracking algorithm will be used which can handle referred problems. The tracking problem is formulated as a detection task and the main steps of the proposed algorithm. We assume that the tracking window in the first frame is given by a detector or manual label. At each frame, we sample some positive samples near the current target location and negative samples away from the object center to update the classifier. To predict the object location in the next frame, we draw some samples around the current target location and determine the one with the maximal classification score.

To account for large scale change of object appearance, a multiscale image representation is often formed by convolving the input image with a Gaussian filter of different spatial variances. The Gaussian filters in practice have to be truncated which can be replaced by rectangle filters.

For each sample $Z \in R^{\omega \times h}$, its multiscale representation is constructed by convolving Z with s set of rectangle filters at multiple scales $\{F_{1,1}, \ldots, F_{\omega,h}\}$ defined by

$$F_{\omega,h}(x,y) = \frac{1}{\omega h} \times f(x) = \begin{cases} 1, & 1 \le x \le 0, 1 \le y \le h, \\ 0, & otherwise. \end{cases}$$

Where ω and h are the width and height of a rectangle filter respectively.

Then we represent each filtered image as a column vector in $R^{\frac{\omega h}{}}$ and concatenate these vectors as a very high-dimensional multiscale image feature vector $x = (x_1, \ldots, x_m)^T \in R^m$ where $m = (\omega h)^2$. The dimensionality m is a typically in the order of 10^6 to 10^{10}. We adopt a sparse random matrix **R** to project x onto a vector $v \in R^n$ in a low-dimensional space. The random matrix **R** needs to be computes only once offline and remains fixed throughout the tracking process. For the sparse matrix **R**, the computational load is very light. And we only need to store the nonzero entries in **R** and the positions of rectangle filters in an input image corresponding to the nonzero entries in each row of **R**. Then, v can be efficiently computed by using **R** to sparsely measure the rectangular features which can be efficiently computed using the integral image method.

It is easy to show that the low-dimensional feature v is scale invariant. Each feature in v is a linear combination of some rectangle filters convolving the input image at different positions. Therefore, without loss of generality, we only need to show that the j-th rectangle feature x_j in the i-th feature v_i in v is scale invariant. We have

$$\begin{aligned} x_j(xy) &= F_{sw_j,sh_j}(sy) \otimes Z(sy) \\ &= F_{sw_j,sh_j}(a) \otimes Z(a)|_{a=sy} \\ &= \frac{1}{s^2 \omega_i h_i} \int_{u \epsilon \Omega_s} Z(a-u) du \\ &= \frac{1}{s^2 \omega_i h_i} \int_{u \epsilon \Omega} Z(y-u)|s^2 du \\ &= \frac{1}{\omega_i h_i} \int_{u \epsilon \Omega} Z(y-u) du \end{aligned}$$

$$= F_{w_j,sh_j}(y) \otimes Z(y) = x_j(y)$$

Where $\Omega = \{(u_1, u_2) | 1 \le u_1 \le \omega_i, 1 \le u_2 \le h_i\}$ and $\Omega_s = \{(u_1, u_2) | 1 \le u_1 \le s\omega_i, 1 \le u_2 \le sh_i\}$

We assume all elements in v are independently distributed and model them with a naive Bayes classifier

$$\mathrm{H(v)} = \log\left(\frac{\prod_{i=1}^{n} p(v_i|y=1)p(y=1)}{\prod_{i=1}^{n} p(v_i|y=0)p(y=0)}\right) = \sum_{i=1}^{n} \log\left(\frac{p(v_i|y=1)}{p(v_i|y=0)}\right)$$

Where we assume uniform prior, $p(y=1) = p(y=0)$, and $y \epsilon \{0,1\}$ is a binary variable which represents the sample label.

The random projections of high dimensional random vectors are almost always Gaussian. Thus the conditional distributions $p(v_i|y=1)$ and $p(v_i|y=0)$ in the classifier H(v) are assumed to be Gaussian distributed with four parameters $(\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0)$,

$$p(v_i|y=1) \sim N(\mu_i^1, \sigma_i^1), p(v_i|y=0) \sim N(\mu_i^0, \sigma_i^0)$$

Where $\mu_i^1(\mu_i^0)$ and $\sigma_i^1(\sigma_i^0)$ are mean and standard deviation of the positive (negative) class. The scalar parameters in last equation are incrementally updated by

$$\mu_i^1 \leftarrow \lambda\mu_i^1 + (1-\lambda)\mu^1$$

$$\sigma_i^1 \leftarrow \sqrt{\lambda(\sigma_i^1)^2 + +(1-\lambda)(\sigma^1)^2 + \lambda(1-\lambda)(\mu_i^1 - \mu^1)^2}$$

Where $\lambda > 0$ is a learning parameter,

$$\sigma^1 = \sqrt{\frac{1}{n}\sum_{k=0|y=1}^{n-1}(v_i(k) - \mu^1)^2}$$

And $\mu_1 = \frac{1}{n}\sum_{k=0|y=1}^{n-1} v_i(k)$. Parameters μ_i^0 and σ_i^0 are updated with similar rules. The above equations can be easily derived by maximum likelihood estimation.

Because the variables are assumed to be independent in our classifier, the n-dimensional multivariate problem is reduced to the n univariate estimation problem. Thus, it requires fewer training samples to obtain accurate estimation than estimating the covariance matrix in the multivariate estimation. Furthermore, several densely sampled positive samples surrounding the current tracking result are used to update the distribution parameters, which is able to obtain robust estimation even when the tracking result has some drift. In addition, the useful information from the former accurate samples is also used to update the parameter distributions, thereby facilitating the proposed algorithm to be robust to misaligned samples. Thus, our classifier performs robustly even when the misaligned or the insufficient number of training samples is used.

Results

Some of the experiment result will be shown in Figure 7 and Figure 8 with red rectangle to compare some of the state-of-the-art method. These pictures show that the tracking result is satisfied even some challenges occur. In addition, the success rate is evaluated through the overlap rate which is defined by the PASCAL VOC [11] criterion if given the tracking result of each frame R_T, the corresponding ground truth R_G and $score = \frac{area(R_T \cap R_G)}{area(R_T \cup R_G)}$. The tracking results are regarded as being valid when the score is over 0.5. The average overlap rate of our tracker is 0.82 while the highest is 0.61 at present.

Figure 7. The tracking result of David

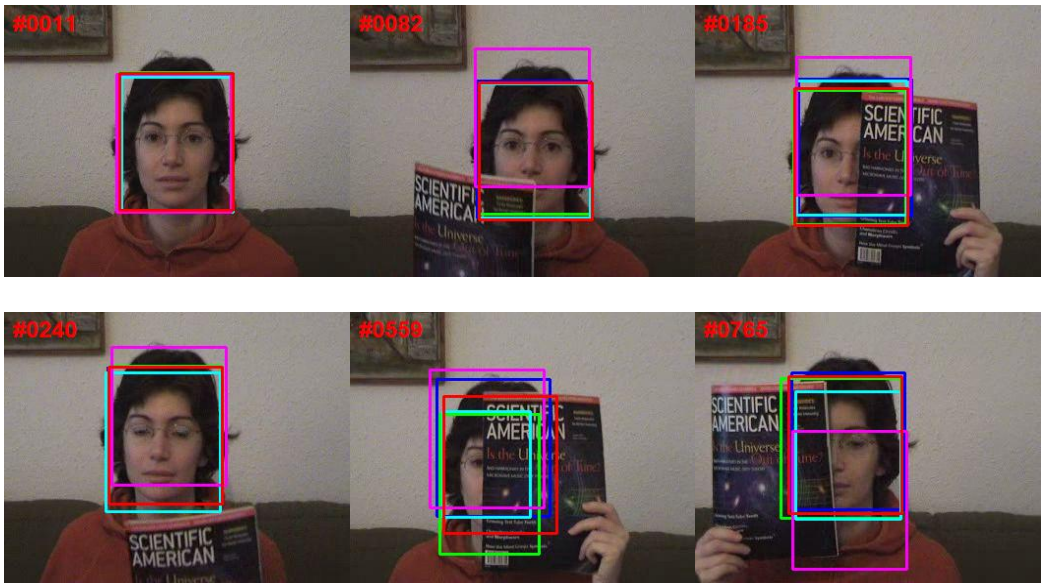
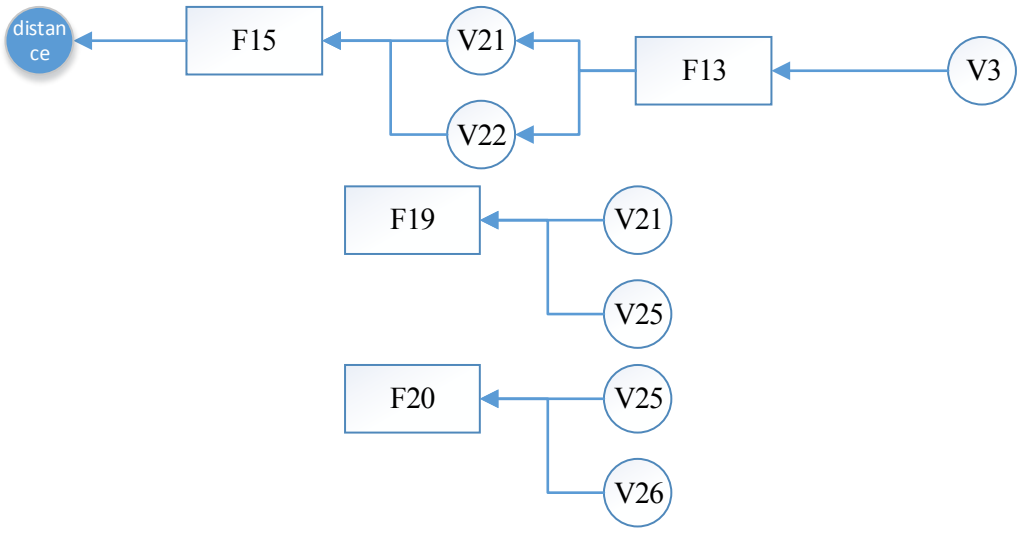


Fig.8 The tracking result of occlusion

Functions

Related functions are F13, F14, F15, F19, F20 and F25.

Relation of above functions and global variables are shown as follows.



Speech Recognition

Method

Our implementation is based on Microsoft Kinect SDK.

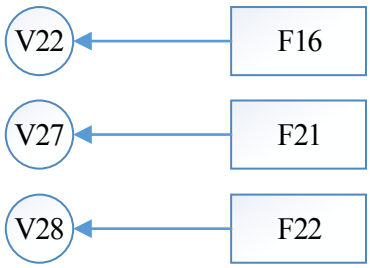
Results

We can recognize isolated words or continuous sentences, which were built on top of word recognition technology. Our speech recognition system is speaker independent. It means we do not need to train a system that recognizes each of the words uttered single or multiple times by specific set of speakers. Our speech recognition system is based on a speech dictionary, which includes important words and sentences for speech recognition.

Functions

Function: Related functions are F16, F21 and F22.

Relation of above functions and global variables are shown as follows.



Support File: This function needn't the Entrance parameters. This function will starts to recognize the speech and returns a textual representation on the screen when the subject speaks.

References:

1. Lepetit, V., F. Moreno-Noguer, and P. Fua, *Epnp: An accurate o (n) solution to the pnp problem.* International journal of computer vision, 2009. **81**(2): p. 155-166.
2. Viola, P. and M.J. Jones, *Robust real-time face detection.* International Journal of Computer Vision, 2004. **57**(2): p. 137-154.
3. Daugman, J.G., *High Confidence Visual Recognition of Persons by a Test of Statistical Independence.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993. **15**(11): p. 1148-1161.
4. Xiong, X.H. and F. De la Torre, *Supervised Descent Method and its Applications to Face Alignment.* 2013 Ieee Conference on Computer Vision and Pattern Recognition (Cvpr), 2013: p. 532-539.
5. Dementhon, D.F. and L.S. Davis, *Model-Based Object Pose in 25 Lines of Code.* International Journal of Computer Vision, 1995. **15**(1-2): p. 123-141.
6. Valenti, R. and T. Gevers, *Accurate Eye Center Location through Invariant Isocentric Patterns.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012. **34**(9): p. 1785-1798.
7. Markus, N., et al., *Eye pupil localization with an ensemble of randomized trees.* Pattern Recognition, 2014. **47**(2): p. 578-587.
8. Jan, F., I. Usman, and S. Agha, *A non-circular iris localization algorithm using image projection function and gray level statistics.* Optik, 2013. **124**(18): p. 3187-3193.
9. Huang, D., et al., *Local binary patterns and its application to facial image analysis: a survey.* Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 2011. **41**(6): p. 765-781.
10. Zhang, K.H., L. Zhang, and M.H. Yang, *Fast Compressive Tracking.* Ieee Transactions on Pattern Analysis and Machine Intelligence, 2014. **36**(10): p. 2002-2015.
11. Everingham, M., et al., *The Pascal Visual Object Classes (VOC) Challenge.* International Journal of Computer Vision, 2010. **88**(2): p. 303-338.