In this work, I focus on recognizing hand-object interactions in wearable videos. Recognizing hand and object interactions while preparing foods can be useful for monitoring the wearer's diet. Also, with limited movement and context information gained from first-person frames, I reconstruct the kitchen background and build up a body structure model to illustrate the detailed movement of the wearer and interaction between the camera-man and his background.

First-person-view Body Pose & Context 3D Reconstruction

Summer Internship Report

Yuwei Qiu

Contents

Acknowledgement

Thanks Prof. Jianbo Shi, Prof. Kostas Daniilidis, Prof. Yuanqi Su, and Prof. Xiaowei Zou for kind suggestions.

Thanks Shan Su, Shiwei Yuan, Xin Yuan and Xiao Zhang for assistance.

Thanks Jianyu Wang for love and accompanies.

Bird-View

Computer Vision

Computer Graphics

Robotics

1. Computer vision

Computer Vision is an interdisciplinary field that deals with how computers can be made for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.
Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions.

Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that can interface with other thought processes and elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

First-person-view   Body Pose & Context 3D Reconstruction

As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

Sub-domains of computer vision include scene reconstruction, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, and image restoration.

Applications range from tasks such as industrial machine vision systems which, say, inspect bottles speeding by on a production line, to research into artificial intelligence and computers or robots that can comprehend the world around them. The computer vision and machine vision fields have significant overlap. Computer vision covers the core technology of automated image analysis which is used in many fields. Machine vision usually refers to a process of combining automated image analysis with other methods and technologies to provide automated inspection and robot guidance in industrial applications. In many computer vision applications, the computers are pre-programmed to solve a particular task, but methods based on learning are now becoming increasingly common. Examples of applications of computer vision include systems for:

- Automatic inspection, *e.g.*, in manufacturing applications;
- Assisting humans in identification tasks, e.g., a species identification system;[19]
- Controlling processes, *e.g.*, an industrial robot;
- Detecting events, *e.g.*, for visual surveillance or people counting;
- Interaction, *e.g.*, as the input to a device for computer-human interaction;
- Modeling objects or environments, *e.g.*, medical image analysis or topographical modeling;
- Navigation, *e.g.*, by an autonomous vehicle or mobile robot; and

Organizing information, *e.g.*, for indexing databases of images and image sequences.

2. Computer Graphics

Computer graphics are pictures and films created using computers. Usually, the term refers to computer-generated image data created with help from specialized graphical hardware and software. It is a vast and recent area in computer science. The phrase was coined in 1960, by computer graphics researchers Verne Hudson and William Fetter of Boeing. It is often abbreviated as CG, though sometimes erroneously referred to as computer-generated imagery (CGI).

Some topics in computer graphics include user interface design, sprite graphics, vector graphics, 3D modeling, shaders, GPU design, implicit surface visualization with ray tracing, and computer vision, among others. The overall methodology depends heavily on the underlying sciences of geometry, optics, and physics.

Computer graphics is responsible for displaying art and image data effectively and meaningfully to the user. It is also used for processing image data received from the physical world. Computer graphic development has had a significant impact on many types of media and has revolutionized animation, movies, advertising, video games, and graphic design generally.

3. Robotics

Robotics is an interdisciplinary branch of engineering and science that includes mechanical engineering, electrical engineering, computer science, and others. Robotics deals with the design, construction, operation, and use of robots,[1] as well as computer systems for their control, sensory feedback, and information processing.

These technologies are used to develop machines that can substitute for humans. Robots can be used in any situation and for any purpose, but today many are used in dangerous environments (including bomb detection and de-activation), manufacturing processes, or where humans cannot survive. Robots can take on any form but some are made to resemble humans in appearance. This is said to help in the acceptance of a robot in certain replicative behaviors usually performed by people. Such robots attempt to replicate walking, lifting, speech, cognition, and basically anything a human can do. Many of today's robots are inspired by nature, contributing to the field of bio-inspired robotics.

The concept of creating machines that can operate autonomously dates back to classical times, but research into the functionality and potential uses of robots did not grow substantially until the 20th century.[2] Throughout history, it has been frequently assumed that robots will one day be able to mimic human behavior and manage tasks in a human-like fashion. Today, robotics is a rapidly growing field, as technological advances continue; researching, designing, and building new robots serve various practical purposes, whether domestically, commercially, or militarily. Many robots are built to do jobs that are hazardous to people such as defusing bombs, finding survivors in unstable ruins, and exploring mines and shipwrecks. Robotics is also used in STEM (Science, Technology, Engineering, and Mathematics) as a teaching aid.

As more and more robots are designed for specific tasks this method of classification becomes more relevant. For example, many robots are designed for assembly work, which may not be readily adaptable for other applications. They are termed as "assembly robots". For seam welding, some suppliers provide complete welding systems with the robot i.e. the welding equipment along with other material handling facilities like turntables etc. as an integrated unit. Such an integrated robotic system is called a "welding robot" even though its discrete manipulator unit could be adapted to a variety of tasks. Some robots are specifically designed for heavy load manipulation, and are labelled as "heavy duty robots".
Current and potential applications include:

- Military robots
- Caterpillar plans to develop remote controlled machines and expects to develop fully autonomous heavy robots by 2021.[20] Some cranes already are remote controlled.
- It was demonstrated that a robot can perform a herding[21] task.
- Robots are increasingly used in manufacturing (since the 1960s). In the auto industry, they can amount for more than half of the "labor". There are even "lights off" factories such as an IBM keyboard manufacturing factory in Texas that is 100% automated.[22]
- Robots such as HOSPI[23] are used as couriers in hospitals (hospital robot). Other hospital tasks performed by robots are receptionists, guides and porters helpers.[24]

- Robots can serve as waiters[25][26] and cooks,[27] also at home. Boris is a robot that can load a dishwasher.[28]
- Robot combat for sport – hobby or sport event where two or more robots fight in an arena to disable each other. This has developed from a hobby in the 1990s to several TV series worldwide.
- Cleanup of contaminated areas, such as toxic waste or nuclear facilities.[29]
- Agricultural robots (AgRobots[30][31]).
- Domestic robots, cleaning and caring for the elderly
- Medical robots performing low-invasive surgery
- Household robots with full use.
- Nanorobots
- Swarm robotics

Background and Literature

First-Person-Videos and Ego-Surfing

First-person camera directly supplies the source of person-object interaction, is a significant departure from the traditional passive observation from a third person camera. A first-person camera provides information about what the person sees in terms of object appearance and his/her specific action, which may function well in object detection and skill assessment.

A single first-person image may tell us about human visual attention/action with objects appearing in the scene without an eye-tracking device.

Action-object
Action-objects are introduced as the trigger of person's visual and motor signals. An action-object is associated with specific actions such as seeing (i.e visual appearance) or touching (which can be detected armed with hand detector). It possesses distance and orientation (i.e 3D formation) to the person.

EgoNet
An newly designed network called EgoNet is built up to detect

action-objects in first-person images. Object's visual appearance, first person's gazes, and 3D spatial cues are learnt to give prediction of a per-pixel action-object likelihood map. To fully generalize these information, two pathways (\emph{Semantic Gaze Pathway} and \emph{3D Spatial Pathway}) are built up for the learning of 1) object's visual appearance and first person's gaze; 2) 3D spatial cues. Additional convolutional layer is used to integrate these outputs of two pathways. The entire network is jointly optimized by minimizing the per-pixel softmax loss function with respect to the binary ground truth action-object labels.

Comparison with third-view-videos

In direction of human visual or motor behaviors, EgoNet for action-object prediction performs better than other saliency methods because the letter ones do not incorporate object level cues. Also, traditional object detectors (e.g. hand or head detector) fail to identify visual interactions (the mirror) whereas a classic object detector incorrectly treats every object as an action-object. In comparison, EgoNet accurately predicts action-objects in both cases.

First-person in skill assessment

First-person cameras provide a high resolution view of a player's actions, and also eliminate tracking of the player in third-person cameras.

Basketball elements

Basketball elements including the player 1) has the ball, 2) is doing a shot and 3) is in the layup area are easy to be understand. Therefore, these elements can be easily labeled in the data even by non-experts. Timely sequences of basketball elements are defined as basketball activities including 1) making jumpshot, 2) missing jumpshot, 3) making a layup, 4) missing a layup and 5) getting a rebound by asking binary yes/no questions. Based on these basketball activities, individual performance index rating is established. Detection FCNs are used for basketball elements detection in the first-person images.

Multi-path element CNN

To locate basketball elements, detection FCNs with separate pathways. For different elements prediction, different Zooming-in images are used as inputs. Then a 1024 dimensional LSTM layer and 2 dimensional fully connected layers are attached to optimize each pathway separately to minimize a softmax loss.

Introduction and Topic

First-Person Body Pose Reconstruction and Motion Recognition

Wearable cameras do not restrict the user's activity and can easily record daily activities from a first-person viewpoint. Analysis of these videos has been actively explored for different applications, such as recognizing events [1][2], interactions [3][4], ego-actions [5] and handled objects [6][7]. In this work, I focus on recognizing hand-object interactions in wearable videos. Recognizing hand and object interactions while preparing foods can be useful for monitoring the wearer's diet. Also, with limited movement and context information gained from first-person frames, I reconstruct the kitchen background and build up a body structure model to illustrate the detailed movement of the wearer and interaction between the camera-man and his background.

Main challenges are three-hold: 1) the reconstruction of both the context and camera-man extracted from limited first-person videos information, 2) the objection and segmentation of camera-man's moving hand and 3) the reasonable prediction for body parts without recording in the first person videos.

First-person-view Body Pose & Context 3D Reconstruction

Reconstruction of both context, camera-man and interactive objects from first-person videos is fairly challenging. Difficulties from both the lack of body motion information and ego-centric videos have to be overcome. A challenge of using first person cameras is that they often produce highly jittery, blurry, and narrow view, unlike third person videos captured from mostly static and omniscient views. To overcome this, we need to first build up 3D models of the whole environment including the camera-man, the interactive people and objects and the context the same time. However, since the information is limited, it is ultimate-challenging to build up the structure from motion. In achieving this, a structure from motion framework is built up for context recovering.

Hand activity recognition from videos recorded in third-person viewpoints has been extensively studied [8], with particular focus given to features based on optical flows such as HOF [9] and MBH [10]. These features can extract local motion features from keypoints, and previous research has shown that these features are effective for recognizing whole-body activities. Wang et al. [11] proposed dense trajectories to effectively sample the keypoints for activity recognition. The ad- vantage of their approach is that it can extract statistically reliable features by densely sampling keypoints. They calculated the motion features at each keypoint by using dense optical flows [12] and tracked the keypoints for fixed time periods to avoid drifting. From each keypoint, the HOF and MBH are extracted to represent the local motion features and the HOG [13] is extracted to represent the local shape features. They also introduced trajectory features that represent the relative movement of each keypoint. In the recent activity recognition challenge [14], Peng et al. [15] showed that dense trajectories gave the best results. Although there have been significant improvements in whole body activity recognition, hand activity recognition is still challenging because local features do not contain enough information and information about hand configuration is needed to disambiguate similar actions.

However, hand motion recognition and extraction in ego-centric videos is much more difficult than third-angle sequences. With limited parts of both the arms and fingers recorded in the sequences, all segmentation work should be set in pixel-level. To object hand and track its motion and movements, I use KLT detector for hands tracking and super-pixel level segmentation, also region-growing algorithms.

To solve the ambiguous problem of classifying human legs from range data, the adaboost learning approach [14] is applied to create a strong classifier from a weak classifier that depends on the properties of the pre-defined geometric constraints. The authors employ Kalman filter-based leg tracking by using Multiple Hypothesis Tracking (MHT) for solving the leg occlusion state problem. This method is extended from a linear motion model to a social force model [15] for the purpose of human motion prediction. Single- layer LRF tracking can track only one part of the human body. Tracked parts are easily hidden from the view and the system can not recover the occluded state. To solve this problem, the multi-layer LRFs tracking system [16], [17] can be used for extracting sufficient features at different levels of the human body. An optional method would be to place many external LRFs [18] in the environment. Lately, 3D LRFs techniques [19], [20] for human detection and tracking using bottom-up and top-down approaches are also proposed but require a lot of training data. In parallel, Vision-based human tracking has been developed in recent years. The Histogram of Oriented Gradients (HOG) [21] is a widely used method for robust visual people detection in nowadays. The depth image from a stereo camera is used to obtain 3D information of the environment, which make foreground- background segmentation very easy. In [22], the authors propose a stereo vision system on a mobile platform using a combination of visual odometry, HOG pedestrian detection and Depth estimation and tracking. The objective is to detect multiple people and roughly track their body pose.

In this work, I introduce general human body proportion to generate the general structure of human body. Also, I build up an intelligence system for body pose prediction and recognition with limited input including feet position and head orientation.

To sum up, the main goal of this work is to build up most reasonable body structure and pose from limited first-person videos. Secondly, pixel-level 3D reconstruction of the context, people and objects are also generated. Computer graphic frameworks are also introduced for multi-view geometry reconstruction of the whole environment. Thirdly, sequence processing is one of the most important part of this work.

Proposed Methods and Models

1 Structure From Motion

Humans perceive a lot of information about the three-dimensional structure in their environment by moving through it. When the observer moves and the objects around the observer move, information is obtained from images sensed over time.

Finding structure from motion presents a similar problem to finding structure from stereo vision. In both instances, the correspondence between images and the reconstruction of 3D object needs to be found.

Generally, structure from motion generate 3D coordinates from two-views frames and pictures, which shows a relatively great result just gained from two frames timely neighboring with each other in a piece of first person video. In the first person video, a go-pro camera is wearing on a camera-man's head and every movement of his body (more specifically his hands, arms, fingers and elbows are recorded without his notice). To attain the position of a person's head, only the orientation and position of the camera are required.

To find correspondence between images, features such as corner points (edges with gradients in multiple directions) are tracked from one image to the next. One of the most widely used feature detectors is the scale-invariant feature transform (SIFT). It uses the maxima from a difference-of-Gaussians (DOG) pyramid as features. The first step in SIFT is finding a dominant gradient direction. To make it rotation-invariant, the descriptor is rotated to fit this orientation. Another common feature detector is the SURF (Speeded Up Robust Features). In SURF, the DOG is replaced with a Hessian matrix-based blob detector. Also, instead of evaluating the gradient histograms, SURF computes for the sums of gradient components and the sums of their absolute values. The features detected from all the images will then be matched. One of the matching algorithms that track features from one image to another is the Lukas–Kanade tracker (KLT tracker).

Suppose a bunch of featured corresponding points, $\mathbf{x_1}$ and $\mathbf{x_2}$ and camera calibration matrix $\mathbf{K}$, are extracted from the images already, a trangulation fundamental matrix $\mathbf{F}$ will be generated as follows with least square method:

$$\mathbf{x_2}^{\mathbf{T}}\mathbf{F}\mathbf{x_1} = \mathbf{0} \tag{1}$$

Where F should be a normalized matrix.

With F, camera pose and 3D triangulation can be generated as follows

$$\begin{aligned} &\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}} \\ &\begin{cases} \mathbf{R}_c = \mathbf{U}\mathbf{V}^{\mathsf{T}}, \quad \mathbf{t}_c = \mathbf{t}/\mathbf{D}_{1,1} & \text{if } \det(\mathbf{U}\mathbf{V}^{\mathsf{T}}) = 1 \\ \mathbf{R}_c = -\mathbf{U}\mathbf{V}^{\mathsf{T}}, \quad \mathbf{t}_c = -\mathbf{t}/\mathbf{D}_{1,1} & \text{if } \det(\mathbf{U}\mathbf{V}^{\mathsf{T}}) = -1 \end{cases} \end{aligned} \tag{2}$$

Where R and t stand for camera position's rotation and translation.
For 3D points triangulation, non-linear least square methods are formulated. The loss function is as follows.

$$\underset{\mathbf{X}}{\text{minimize}} \quad \sum_{j=1}^{3} (\tilde{x}_i - u_i/w_i)^2 + (\tilde{y}_i - v_i/w_i)^2 , \tag{3}$$

where

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \mathbf{K}\mathbf{R}_i (\mathbf{X} - \mathbf{C}_i) . \tag{4}$$

$\mathbf{C}_i$ and $\mathbf{R}_i$ are the i^{th} camera optical center and orientation. We can solve this nonlinear optimization by finding $\Delta\mathbf{X}$:

$$\begin{aligned} \Delta\mathbf{X} &= (\mathbf{J}^\mathsf{T}\mathbf{J})^{-1}\mathbf{J}^\mathsf{T}(\mathbf{b} - \mathbf{f}(\mathbf{X})) && (5) \\ \mathbf{b} &= \begin{bmatrix} \tilde{x}_1 & \tilde{y}_1 & \tilde{x}_2 & \tilde{y}_2 & \tilde{x}_3 & \tilde{y}_3 \end{bmatrix}^\mathsf{T} && (6) \\ \mathbf{f}(\mathbf{X}) &= \begin{bmatrix} u_1/w_1 & v_1/w_1 & u_2/w_2 & v_2/w_2 & u_3/w_3 & v_3/w_3 \end{bmatrix}^\mathsf{T} && (7) \end{aligned}$$

where $\mathbf{J}$ is the Jacobian for $\mathbf{X}$:

$$\begin{aligned} \mathbf{J} &= \begin{bmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{X}}^\mathsf{T} & \frac{\partial \mathbf{f}_2}{\partial \mathbf{X}}^\mathsf{T} & \frac{\partial \mathbf{f}_3}{\partial \mathbf{X}}^\mathsf{T} \end{bmatrix}^\mathsf{T} && (8) \\ \frac{\partial \mathbf{f}_i}{\partial \mathbf{X}} &= \begin{bmatrix} \frac{w_i \partial u_i/\partial \mathbf{X} - u_i \partial w_i/\partial \mathbf{X}}{w^2} \\ \frac{w_i \partial v_i/\partial \mathbf{X} - v_i \partial w_i/\partial \mathbf{X}}{w^2} \end{bmatrix} && (9) \\ \frac{\partial u_i}{\partial \mathbf{X}} &= \begin{bmatrix} fr_{11} + p_x r_{31} & fr_{12} + p_x r_{32} & fr_{13} + p_x r_{33} \end{bmatrix} && (10) \\ \frac{\partial v_i}{\partial \mathbf{X}} &= \begin{bmatrix} fr_{21} + p_y r_{31} & fr_{22} + p_y r_{32} & fr_{23} + p_y r_{33} \end{bmatrix} && (11) \\ \frac{\partial w_i}{\partial \mathbf{X}} &= \begin{bmatrix} r_{31} & r_{32} & r_{33} \end{bmatrix} . && (12) \end{aligned}$$

Sometimes some of the matched features are incorrectly matched. This is why the matches should also be filtered. RANSAC (Random Sample Consensus) is the algorithm that is usually used to remove the outlier correspondences. In the paper of Fischler and Bolles, RANSAC is used to solve the Location Determination Problem (LDP), where the objective is to determine the points in space that project onto an image into a set of landmarks with known locations.

The feature trajectories over time are then used to reconstruct their 3D positions and the camera's motion. An alternative is given by so-called direct approaches, where geometric information (3D structure and camera motion) is directly estimated from the images, without intermediate abstraction to features or corners. One illustrated figure is shown as Figure 1.

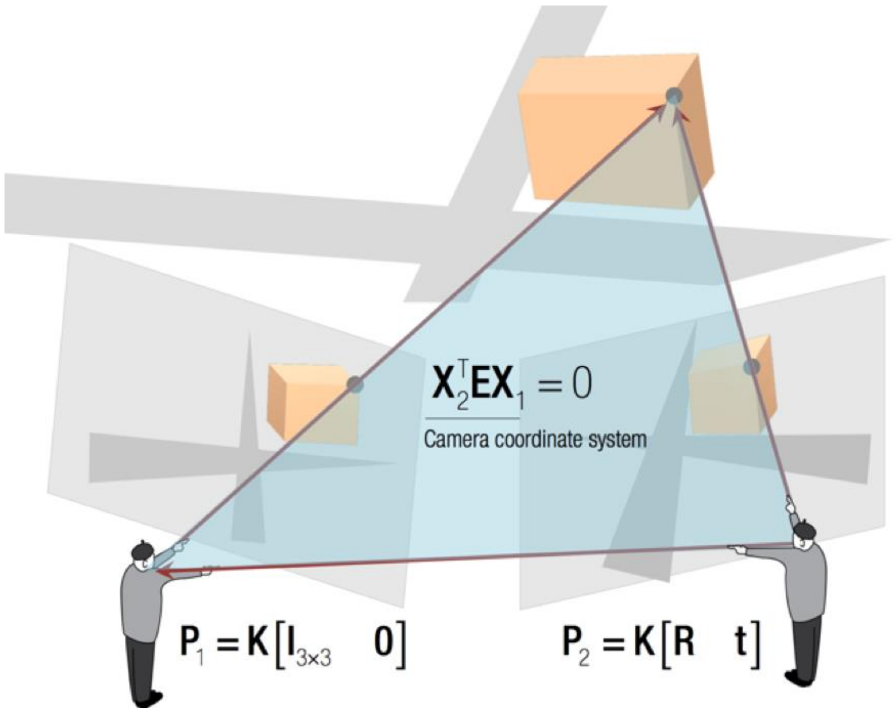


Figure 1

Camera coordinate system illustration and Structure from Motion.

There are several approaches to structure from motion. In incremental SFM, camera poses are solved for and added one by one to the collection. In global SFM, the poses of all cameras are solved for at the same time. A somewhat intermediate approach is out-of-core SFM, where several partial reconstructions are computed that are then integrated into a global solution.

2. Multi-View Reconstruction and Bundle Adjustment

Two-view reconstruction is far from satisfying. The main goal of this work is to build up an intelligence system to build up reconstruction from first person videos, namely images sequences. I introduce multi-view reconstruction and multi-view geometry for context reconstruction in this session. To achieve this, first I introduce bundle adjustment towards the two-view reconstruction.

Bundle adjustment is almost always used as the last step of every feature-based 3D reconstruction algorithm. It amounts to an optimization problem on the 3D structure and viewing parameters (i.e., camera pose and possibly intrinsic calibration and radial distortion), to obtain a reconstruction which is optimal under certain assumptions regarding the noise pertaining to the observed[1] image features: If the image error is zero-mean Gaussian, then bundle adjustment is the Maximum Likelihood Estimator. Its name refers to the bundles of light rays originating from each 3D feature and converging on each camera's optical center, which are adjusted optimally with respect to both the structure and viewing parameters (similarity in meaning to categorical bundle seems a pure coincidence).

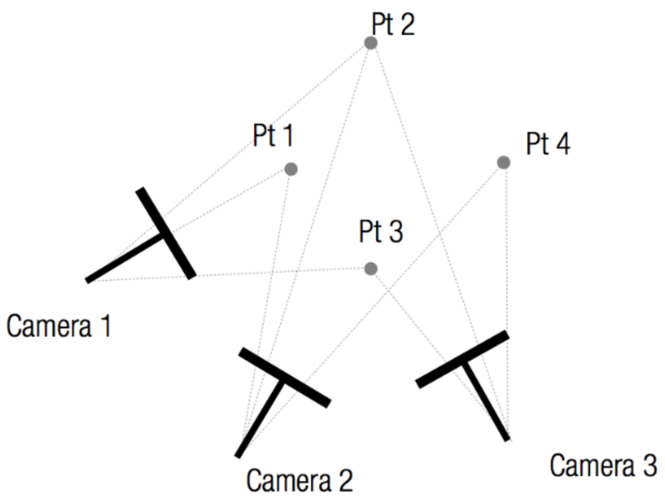


Figure 2

Bundle adjustment

Given a set of images depicting a number of 3D points from different viewpoints, bundle adjustment can be defined as the problem of simultaneously refining the 3D coordinates describing the scene geometry, the parameters of the relative motion, and the optical characteristics of the camera(s) employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all points as it is shown in Figure 2.

The loss function is shown as follows:

$$e = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u/w \\ v/w \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P}\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \mathbf{KR}\left[\mathbf{I}_{3\times3} \quad -\mathbf{C}\right]\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$\underset{\mathbf{R},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u(\mathbf{R},\mathbf{C},\mathbf{X}) / w(\mathbf{R},\mathbf{C},\mathbf{X}) \\ v(\mathbf{R},\mathbf{C},\mathbf{X}) / w(\mathbf{R},\mathbf{C},\mathbf{X}) \end{bmatrix} \right\|^2$$

$$= \underset{\mathbf{q},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) / w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \\ v(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) / w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \end{bmatrix} \right\|^2 \tag{13}$$

3 Hand Tracking and Segmentation

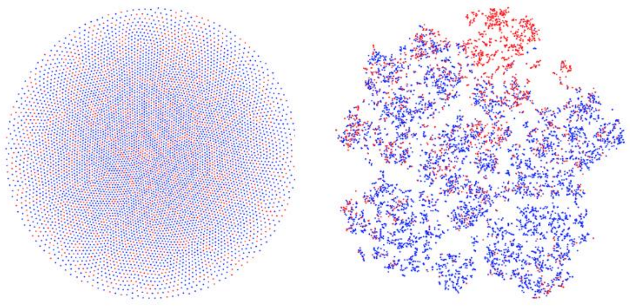


Figure 3

(left) color feature. (right) color and texture feature.

I use the response of a bank of 48 Gabor filters (8 orientations, 3 scales, both real and imaginary components) to examine how local texture affects the discriminability of skin color regions. One of the typical limitations of color- based skin detection approaches is the difficulties encountered with attempting to discriminate against objects that share a similar color distribution to skin. Pixel features extracted from a portion of the hand (marked in red) and a portion of the desk (marked in blue) are visualized in 2D. Notice how the pixel features extracted from the hand and context are completely overlapped in the color space (Figure 3a). However, by concatenating the response of 32 Gabor filters (4 orientations, 4 scales) to the color feature, we can see that the visualization of the feature space shows better separation between pixels of the hand and pixels of the desk (Figure 3b). This visualization suggests that low-level texture can help to disambiguate between hands and other similar colored objects. For pixel connection, a region growing filter is added. Some of the results are shown in Figure 4.

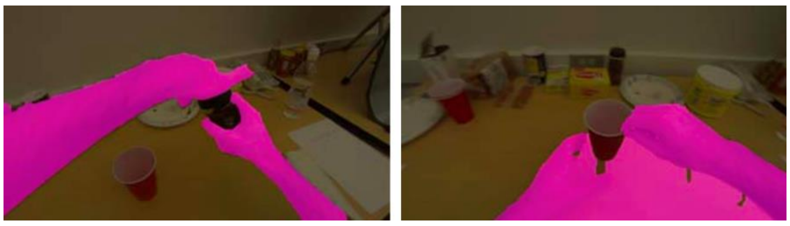


Figure 4

(left) color feature. (right) color and texture feature.

4 Body Pose Prediction and Reconstruction

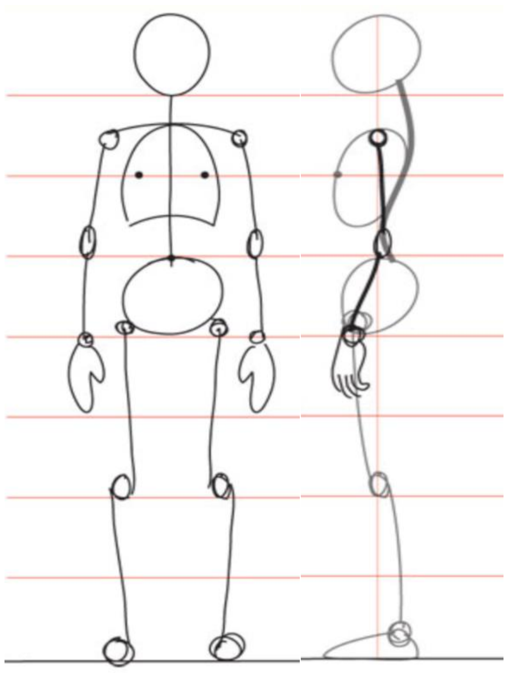


Figure 5

General human body proportion model

With camera position (head position and orientation), hand and arms segmentation and attained, a human body can be built up. Generally, the human body proportion model generated from “eight-head theory” is shown in Figure 5.

Main body parts and parameter required are connected through a mathematical model. The correspondence of human body parts and the positions or coordinates I attained from the previous sessions is shown in Table 1.

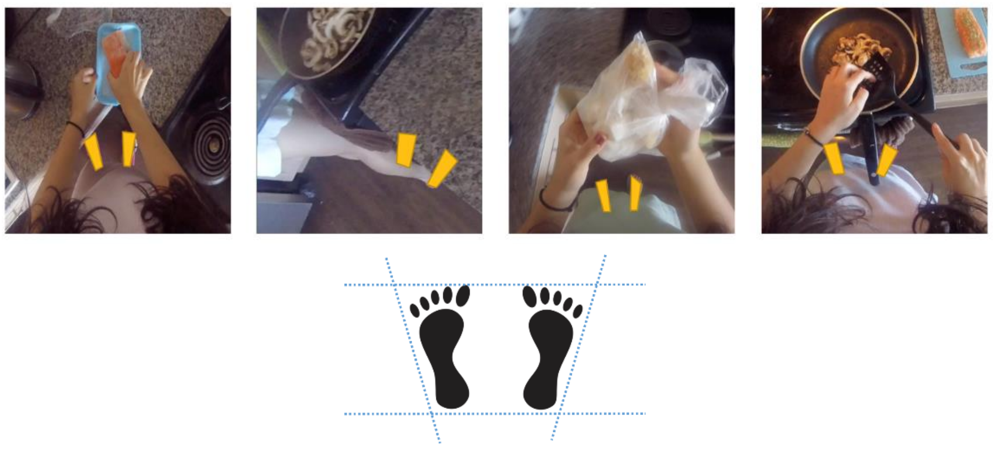
Body parts	Head	Finger ends	Elbows	Gravity center
Positions	Camera position	Hand segmentation	Cross points	Camera projection

Table 1

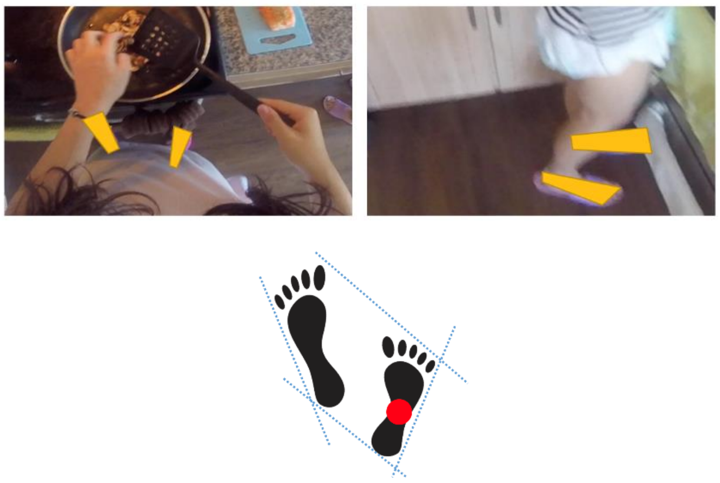
Body parts and positions, coordinates correspondence

Other parts, which I cannot generate exactly and accurately from the information of first person videos, required reasonable prediction.

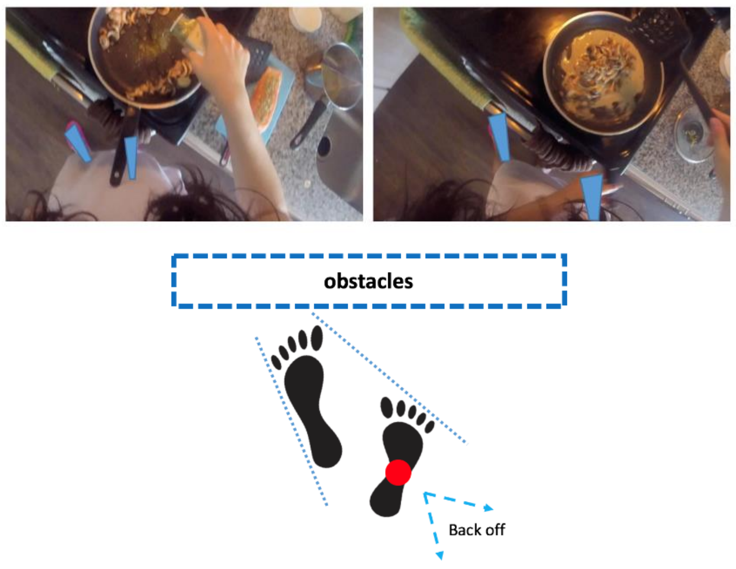
As for feet action connection, I generate different types of first-person feet pattern from the first-person videos, which has been shown in Figure 6.



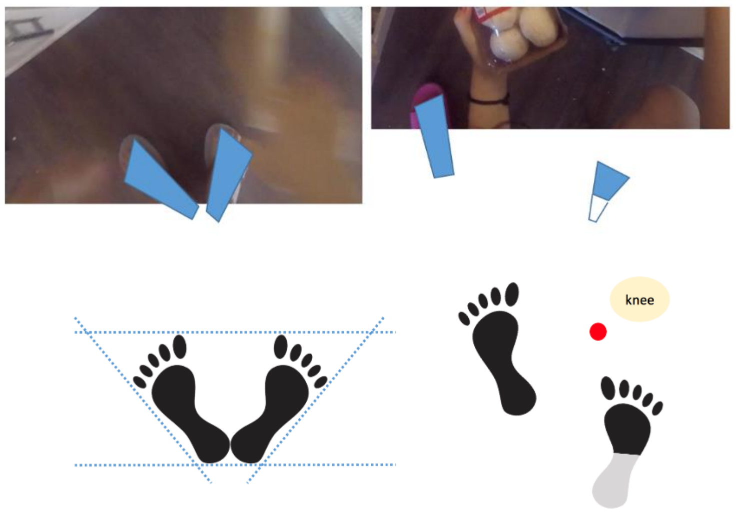
(a)



(b)



(c)



(d)

Figure 6

Feet patterns during cooking

Dataset and Training

First-Person Video Dataset

I collected and present a first person cooking-and-preparing-food dataset composed of 6 hours of videos with 3 different cooks. Two of the cooks are familiar with the environment. Each video is 12 minutes long captured by GoPro Hero 5 (Figure 8) mounted with a head strip. It is recorded at 1280x960 with 100 fps. We record 48 videos during the two days, with a different group of people playing each day. We use 24 videos from the first day for training and 24 videos from the second day for testing. I extract the video frames at 5 fps to get 98, 452.

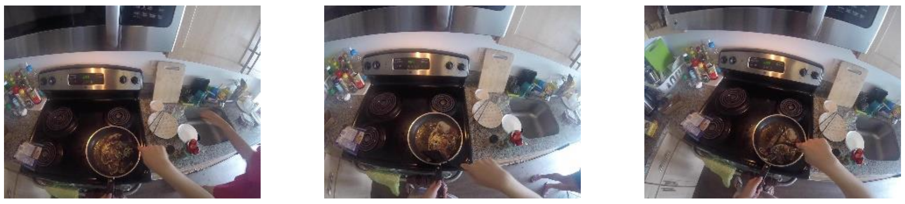


Figure 7

Examples from first person videos

Figure 8

Go-pro Hero series used for data collection

Results

Context 3D Reconstruction

Hand Structure Extraction

Body Pose Structure Sequences

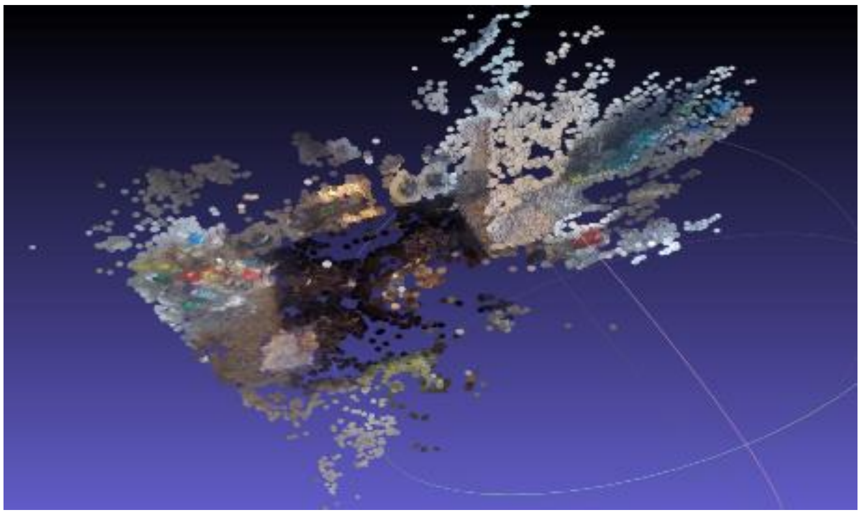
1 Context 3D Reconstruction

In this section, I present and discuss the results of the reconstruction from two-view sequences and multi-view sequences.

Two-view structure from motion lacks a re-adjustment for both the camera poses and 3D coordinates of the triangulated points. Since the information from two neighboring frames is limited, only a small part of the environment is built up as shown in Figure 9(a). After introducing the multi-view stereotype and geometry, more information is added to the reconstruction work therefore a far more evident and apparent environment can be reconstructed as shown in Figure 9(b). With multi-view structure, a much more abstract structure is extracted from the 3d points reconstruction of the environment as it has been shown in Figure 10.

2 Hand Structure Extraction

The results of my proposed method are shown in Figure 11, with all complex context information added into the environment. Every fundamental key points of the body (elbows, arms and finger ends) are accurately detected and tracked.


(a)

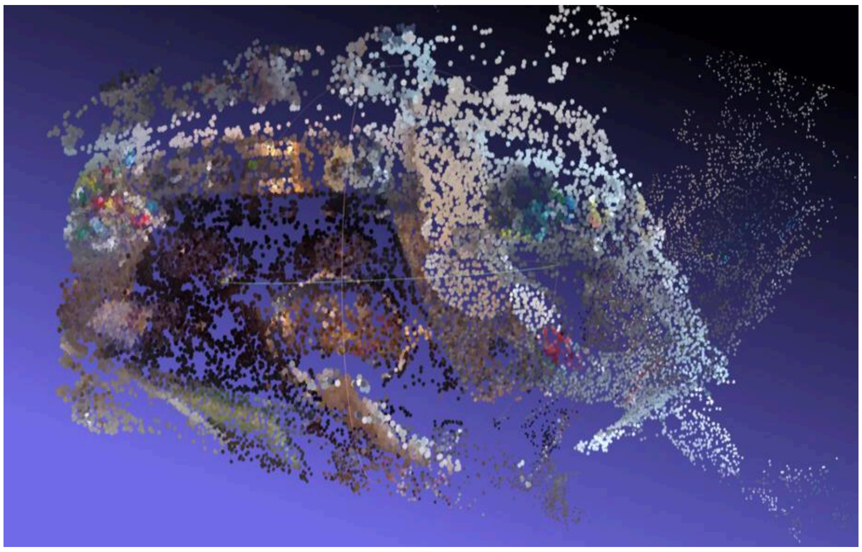

(b)

Figure 9
3D reconstruction of the kitchen

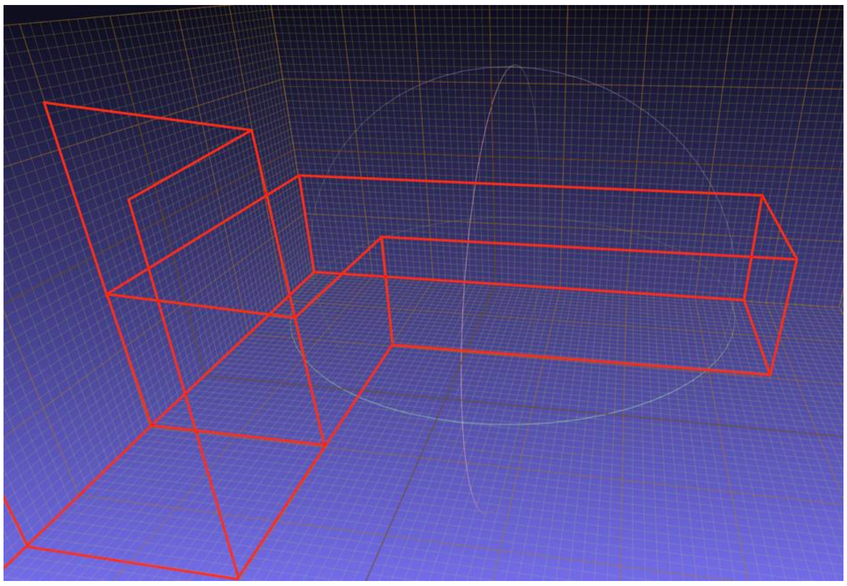
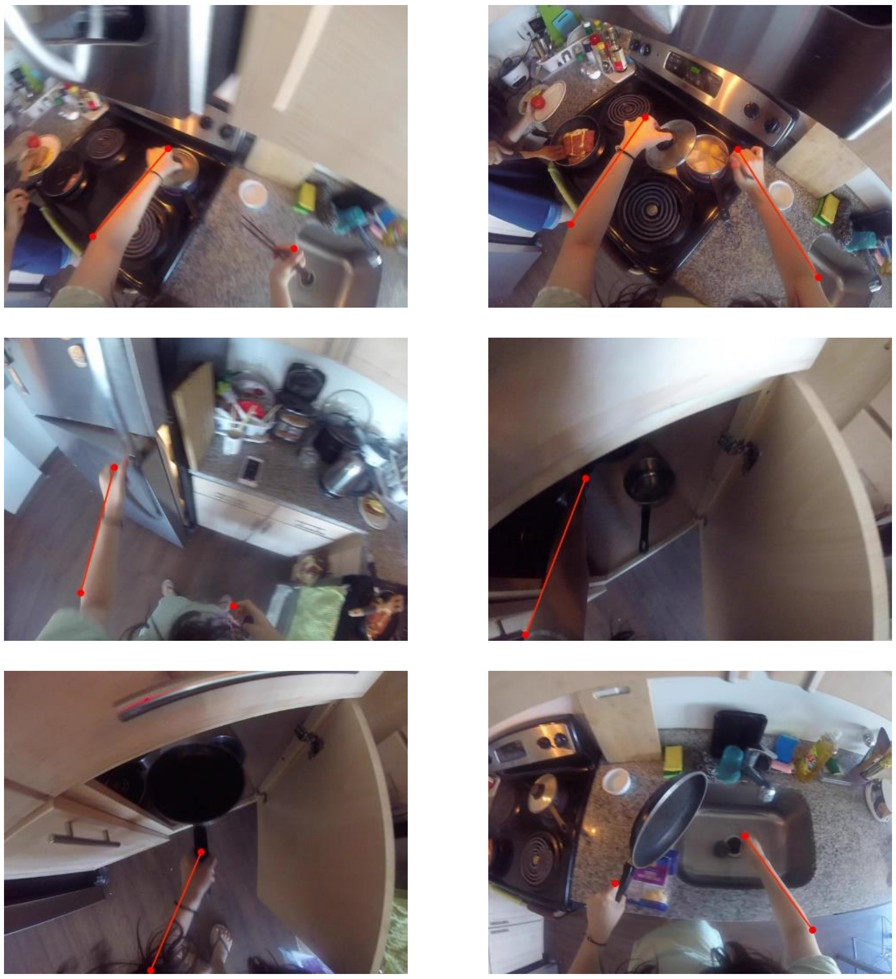


Figure 10
Abstract 3D structure of the kitchen



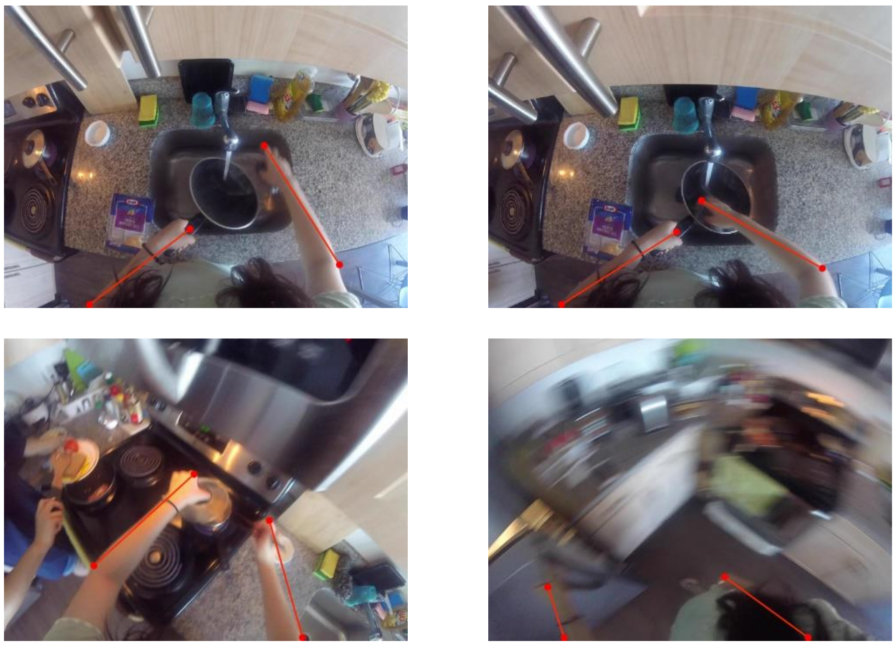


Figure 11

3 Body Pose Structure Sequences

With elbows, arms and finger ends detected and environment reconstructed, I also built up the camera poses, which serve as the position and orientation of the camera-man. Given the general body proportion of human-beings, body pose sequences are generated as Figure 12.

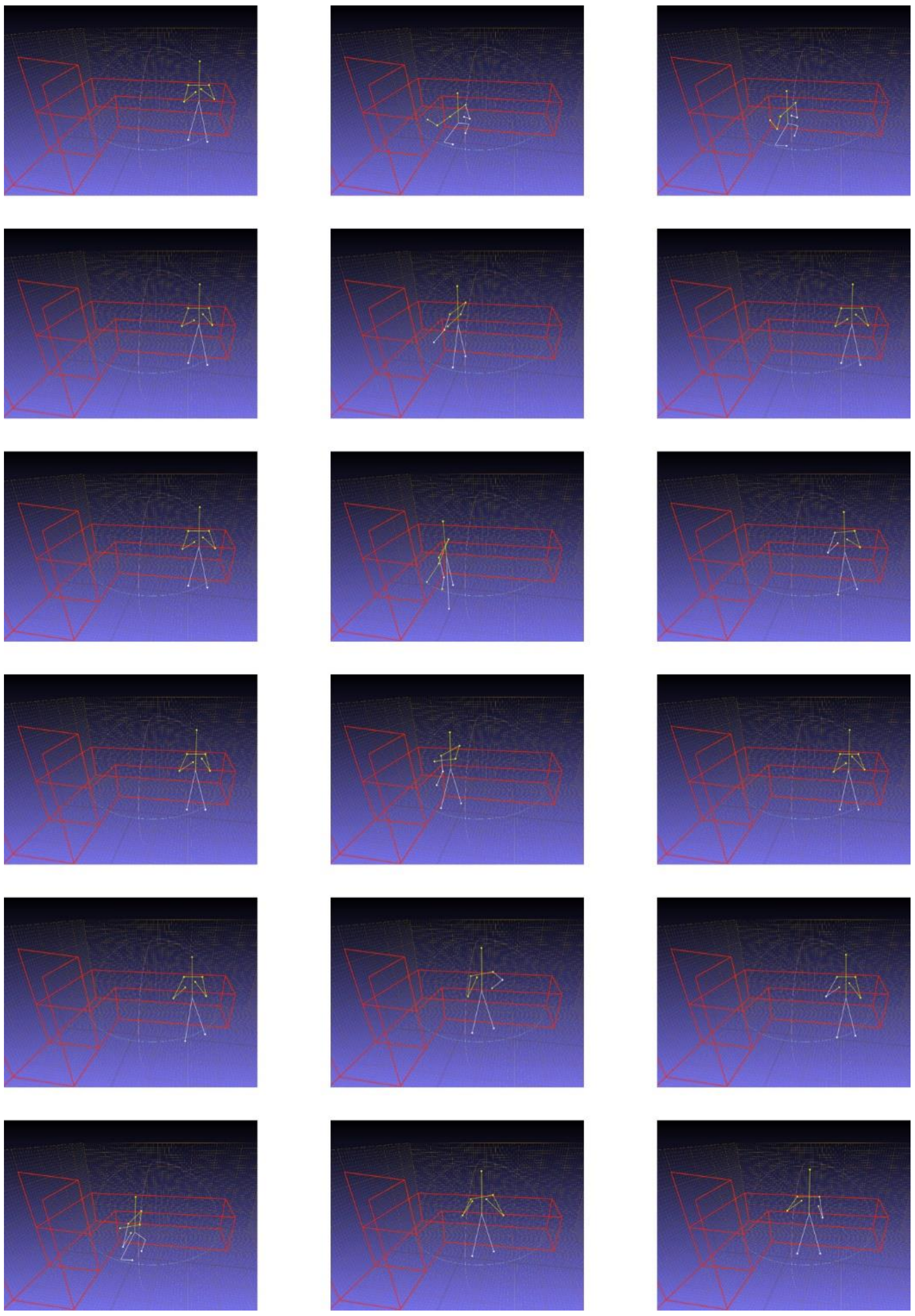
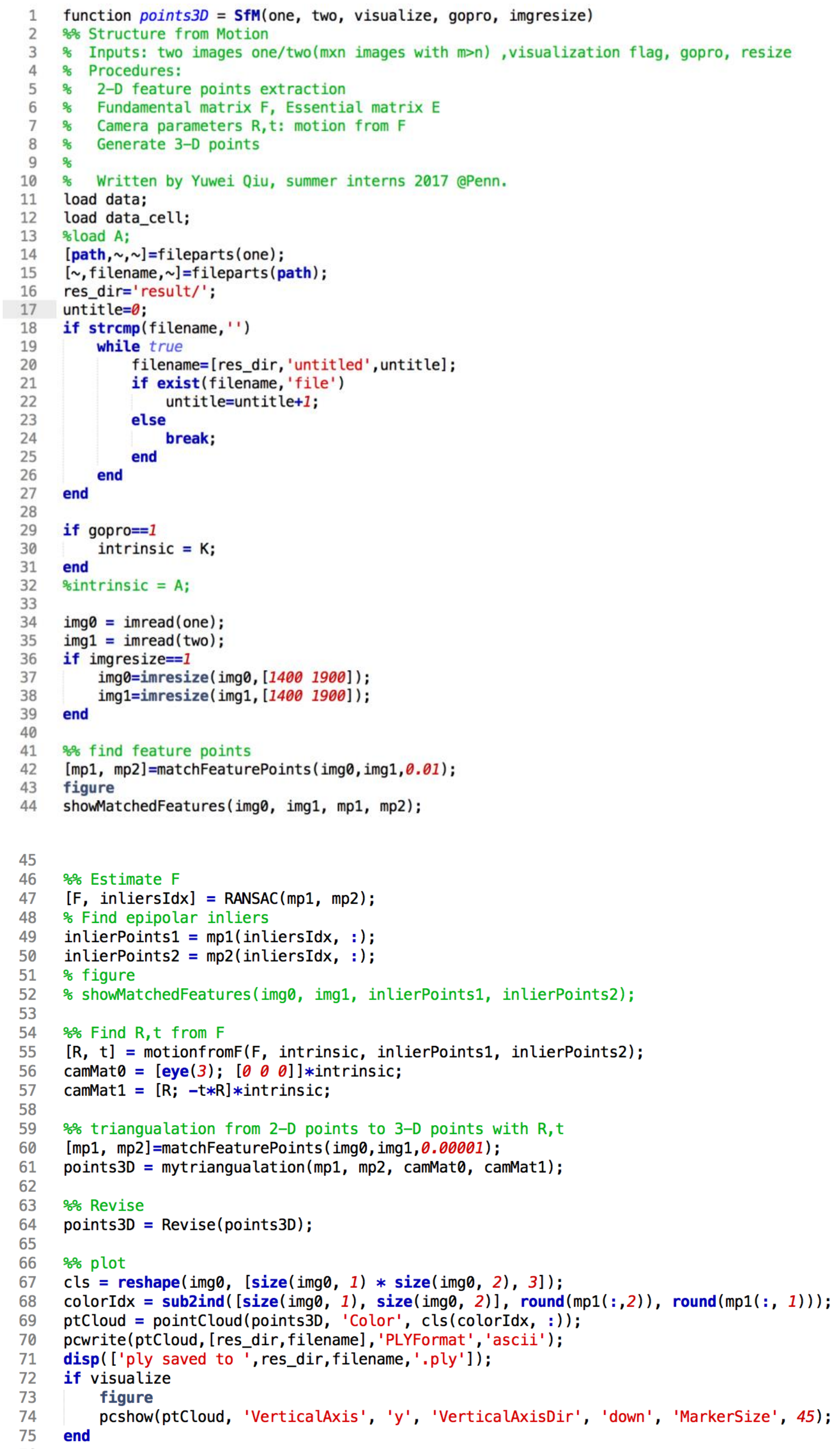


Figure 12
3D body poses structure

Conclusions

In this work, I built up 3D body poses from first-person videos, overcoming three main challenges including 1) the reconstruction of both the context and camera-man extracted from limited first-person videos information, 2) the objection and segmentation of camera-man's moving hand and 3) the reasonable prediction for body parts without recording in the first person videos. Multi-view structure from motion algorithms and bundle adjustments are conducted toward the context, camera man and interactive objects from the first-person sequences. A CNN to generate hand segmentations and color features also Gabor filters. My framework is much simpler but achieve a relatively accurate results as the state-of-the-art algorithms. A connection between human body parts and the position and orientation of the cameras as well as the coordinates of finger ends and elbows is built up for 3D human body proportional model reconstruction.

Appendix: Implementation details and fundamental codes



```
function points3D = SfM(one, two, visualize, gopro, imgresize)
%% Structure from Motion
%  Inputs: two images one/two(mxn images with m>n) ,visualization flag, gopro, resize
%  Procedures:
%   2-D feature points extraction
%   Fundamental matrix F, Essential matrix E
%   Camera parameters R,t: motion from F
%   Generate 3-D points
%
%   Written by Yuwei Qiu, summer interns 2017 @Penn.
load data;
load data_cell;
%load A;
[path,~,~]=fileparts(one);
[~,filename,~]=fileparts(path);
res_dir='result/';
untitle=0;
if strcmp(filename,'')
    while true
        filename=[res_dir,'untitled',untitle];
        if exist(filename,'file')
            untitle=untitle+1;
        else
            break;
        end
    end
end

if gopro==1
    intrinsic = K;
end
%intrinsic = A;

img0 = imread(one);
img1 = imread(two);
if imgresize==1
    img0=imresize(img0,[1400 1900]);
    img1=imresize(img1,[1400 1900]);
end

%% find feature points
[mp1, mp2]=matchFeaturePoints(img0,img1,0.01);
figure
showMatchedFeatures(img0, img1, mp1, mp2);


%% Estimate F
[F, inliersIdx] = RANSAC(mp1, mp2);
% Find epipolar inliers
inlierPoints1 = mp1(inliersIdx, :);
inlierPoints2 = mp2(inliersIdx, :);
% figure
% showMatchedFeatures(img0, img1, inlierPoints1, inlierPoints2);

%% Find R,t from F
[R, t] = motionfromF(F, intrinsic, inlierPoints1, inlierPoints2);
camMat0 = [eye(3); [0 0 0]]*intrinsic;
camMat1 = [R; -t*R]*intrinsic;

%% triangualation from 2-D points to 3-D points with R,t
[mp1, mp2]=matchFeaturePoints(img0,img1,0.00001);
points3D = mytriangualation(mp1, mp2, camMat0, camMat1);

%% Revise
points3D = Revise(points3D);

%% plot
cls = reshape(img0, [size(img0, 1) * size(img0, 2), 3]);
colorIdx = sub2ind([size(img0, 1), size(img0, 2)], round(mp1(:,2)), round(mp1(:, 1)));
ptCloud = pointCloud(points3D, 'Color', cls(colorIdx, :));
pcwrite(ptCloud,[res_dir,filename],'PLYFormat','ascii');
disp(['ply saved to ',res_dir,filename,'.ply']);
if visualize
    figure
    pcshow(ptCloud, 'VerticalAxis', 'y', 'VerticalAxisDir', 'down', 'MarkerSize', 45);
end
```

Structure from Motion

```matlab
function points3d = mytriangulation(matchedPoints1, matchedPoints2, cam1, cam2)
%% triangulation
%  Inputs: homo coords of 2-D points matchedPoints1/2, R,ts of camera parameters
%  Outputs: homo coords of 3-D points

points2d(:,:,1)=matchedPoints1;
points2d(:,:,2)=matchedPoints2;
cam(:,:,1)=cam1;
cam(:,:,2)=cam2;
nPoints = size(points2d, 1);
points3d = zeros(nPoints, 3, 'like', points2d);

for i = 1:nPoints
    pairs=squeeze(points2d(i, :, :))';
    A = zeros(4, 4);
    for j = 1:2
        P = cam(:,:,j)';
        A(2*j-1,:)=pairs(j, 1)*P(3,:)-P(1,:);
        A(2*j,:)=pairs(j, 2)*P(3,:)-P(2,:);
    end
    [~,~,V] = svd(A);
    X = V(:, end);
    X = X/X(end);
    points3d(i, :) = X(1:3)';
end
end
```

Triangulation

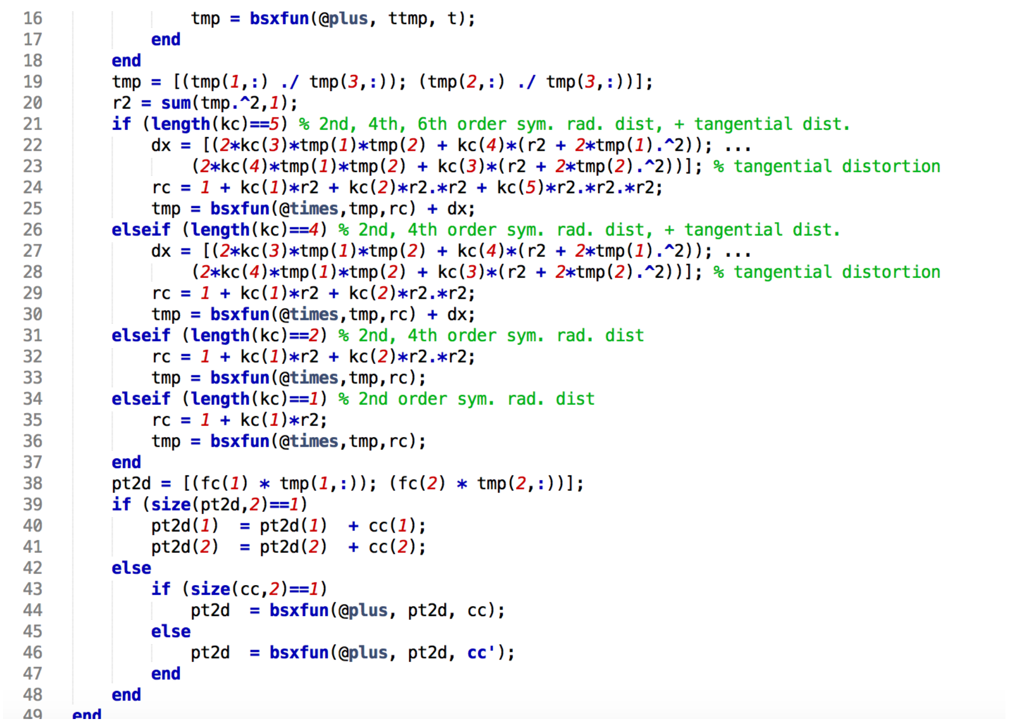
```matlab
% Segment based on area, Region Growing
% Input: fileName: image path
% Output: segmentated super pixels cross points.
%
% Written by Yuwei Qiu, summer interns @Upenn.
fileName='1.jpg';
I = imread(fileName);
if( ~( size(I,3)-3 ))
    I = rgb2gray(I);
end
I = im2double(I);
Ireshape = imresize(I,[600,800]);
I = Ireshape(51:475,200:699);
gausFilter = fspecial('gaussian',[5 5],0.5);
I = imfilter(I,gausFilter,'replicate');


J = zeros(size(I));
Isizes = size(I);
reg_mean = I(x,y);
reg_size = 1;
neg_free = 10000;
neg_list = zeros(neg_free,3);
neg_pos = 0;
pixdist = 0;
while (pixdist < 0.06 && reg_size < numel(I))

     for j=1:4
         xn = x + neigb(j,1);
         yn = y + neigb(j,2);

         ins = (xn>=1)&&(yn>=1)&&(xn<=Isizes(1))&&(yn<=Isizes(1));

         if( ins && J(xn,yn)==0)
             neg_pos = neg_pos+1;
             neg_list(neg_pos,:) =[ xn, yn, I(xn,yn)];
             J(xn,yn) = 1;
         end
     end
    if (neg_pos+10>neg_free)
        neg_free = neg_free + 100000;
        neg_list((neg_pos +1):neg_free,:) = 0;
    end
    dist = abs(neg_list(1:neg_pos,3)-reg_mean);
    [pixdist,index] = min(dist);

    reg_mean = (reg_mean * reg_size +neg_list(index,3))/(reg_size + 1);
    reg_size = reg_size + 1;
    J(x,y)=2;
    x = neg_list(index,1);
    y = neg_list(index,2);
%     if(J(x,y)==2)
%     plot(x,y,'r.');
%     end
    neg_list(index,:) = neg_list(neg_pos,:);
    neg_pos = neg_pos -1;
end
```

Region Growing and Hand Segmentation

```
function [f,bestInliers] = RANSAC(points1, points2)
%% RANSAC: find the most suitable f by RANSAC and 8-points method
%  points1homo and points2homo are 3xn matrix of homo feature points.

nPoints = size(points1, 1); % number of feature points

% homogenious points
points1homo=points1';
points1homo(3,:)=1.0;
points2homo=points2';
points2homo(3,:)=1.0;

%% initialization
threshold=0.01;
maxtrails = 20000;
bestDist = realmax('double'); %bestDist = 1.7977e+308: start from inf

%% iteration
for trails=1:maxtrails
    % estimate f using random 8 points
    sampleIndicies = randperm(nPoints, 8); % randomly select 8 points
    % eightpoint method to find the fundatmental matrix for each groups of points
    f = eightPoint(points1homo(:, sampleIndicies), points2homo(:, sampleIndicies));

    % reprojection error: SHOULD be x2^T * F * x1 = 0
    pfp = (points2homo' * f)';
    pfp = pfp .* points1homo;
    d = sum(pfp, 1) .^ 2; % square error

    % find inliers
    inliers = coder.nullcopy(false(1, nPoints));
    inliers(d<=threshold)=true;
    nInliers=sum(inliers);

    % MSAC metric
    Dist = cast(sum(d(inliers)),'double') + threshold*(nPoints - nInliers);
    if bestDist > Dist
      bestDist = Dist;
      bestInliers = inliers;
    end
end

f = eightPoint(points1homo(:, bestInliers), points2homo(:, bestInliers));
end
```

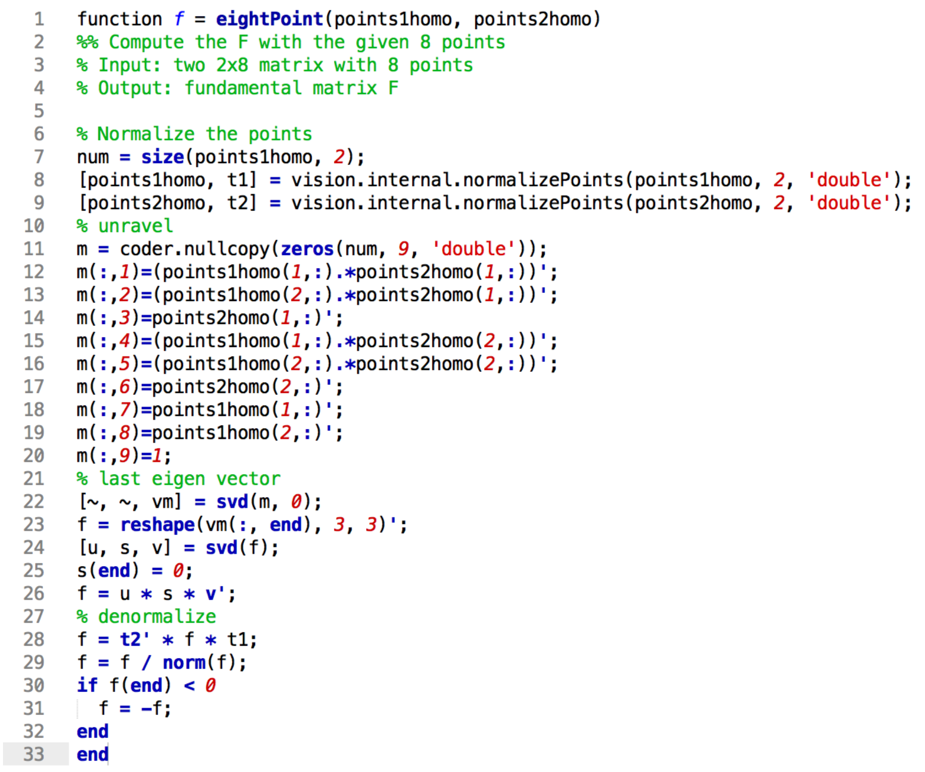
RANSAC

```
function pt2d = BundleAdjustment(pt3d, R,t,fc,cc,kc, camcenter)
    if (camcenter==1)  % t is set in world frame
        if (size(pt3d,2)==1)
            tmp = (R*(pt3d -t));
        else
            ttmp = bsxfun(@minus, pt3d, t');
            tmp = arrayfun(@(x) R*ttmp(x,:)', 1:size(ttmp,1), 'UniformOutput',0);
            tmp = cat(2,tmp{:});
        end
    else  % t is set in camera reference frame
        if (size(pt3d,2)==1)
            tmp = (R*pt3d + t);
        else
            ttmp = arrayfun(@(x) R*pt3d(x,:)', 1:size(pt3d,1), 'UniformOutput',0);
            ttmp = cat(2,ttmp{:});
```



```
            tmp = bsxfun(@plus, ttmp, t);
        end
    end
    tmp = [(tmp(1,:) ./ tmp(3,:)); (tmp(2,:) ./ tmp(3,:))];
    r2 = sum(tmp.^2,1);
    if (length(kc)==5) % 2nd, 4th, 6th order sym. rad. dist, + tangential dist.
        dx = [(2*kc(3)*tmp(1)*tmp(2) + kc(4)*(r2 + 2*tmp(1).^2)); ...
            (2*kc(4)*tmp(1)*tmp(2) + kc(3)*(r2 + 2*tmp(2).^2))]; % tangential distortion
        rc = 1 + kc(1)*r2 + kc(2)*r2.*r2 + kc(5)*r2.*r2.*r2;
        tmp = bsxfun(@times,tmp,rc) + dx;
    elseif (length(kc)==4) % 2nd, 4th order sym. rad. dist, + tangential dist.
        dx = [(2*kc(3)*tmp(1)*tmp(2) + kc(4)*(r2 + 2*tmp(1).^2)); ...
            (2*kc(4)*tmp(1)*tmp(2) + kc(3)*(r2 + 2*tmp(2).^2))]; % tangential distortion
        rc = 1 + kc(1)*r2 + kc(2)*r2.*r2;
        tmp = bsxfun(@times,tmp,rc) + dx;
    elseif (length(kc)==2) % 2nd, 4th order sym. rad. dist
        rc = 1 + kc(1)*r2 + kc(2)*r2.*r2;
        tmp = bsxfun(@times,tmp,rc);
    elseif (length(kc)==1) % 2nd order sym. rad. dist
        rc = 1 + kc(1)*r2;
        tmp = bsxfun(@times,tmp,rc);
    end
    pt2d = [(fc(1) * tmp(1,:)); (fc(2) * tmp(2,:))];
    if (size(pt2d,2)==1)
        pt2d(1)  = pt2d(1)  + cc(1);
        pt2d(2)  = pt2d(2)  + cc(2);
    else
        if (size(cc,2)==1)
            pt2d  = bsxfun(@plus, pt2d, cc);
        else
            pt2d  = bsxfun(@plus, pt2d, cc');
        end
    end
end
```

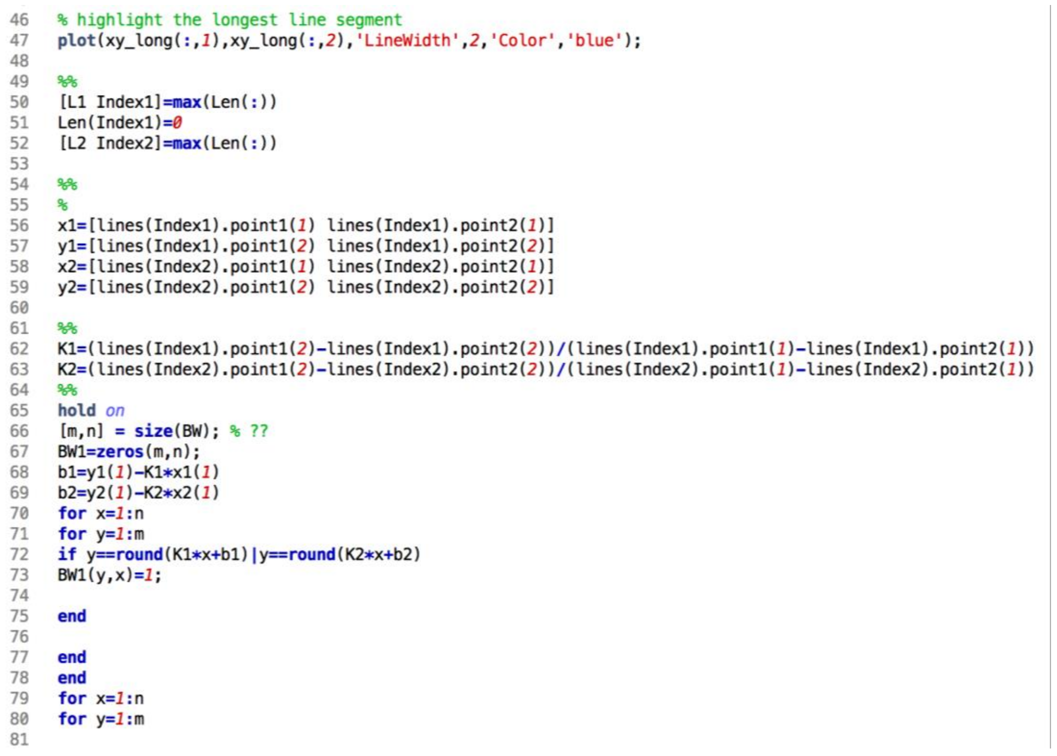
Bundle Adjustment



```
function f = eightPoint(points1homo, points2homo)
%% Compute the F with the given 8 points
% Input: two 2x8 matrix with 8 points
% Output: fundamental matrix F

% Normalize the points
num = size(points1homo, 2);
[points1homo, t1] = vision.internal.normalizePoints(points1homo, 2, 'double');
[points2homo, t2] = vision.internal.normalizePoints(points2homo, 2, 'double');
% unravel
m = coder.nullcopy(zeros(num, 9, 'double'));
m(:,1)=(points1homo(1,:).*points2homo(1,:))';
m(:,2)=(points1homo(2,:).*points2homo(1,:))';
m(:,3)=points2homo(1,:)';
m(:,4)=(points1homo(1,:).*points2homo(2,:))';
m(:,5)=(points1homo(2,:).*points2homo(2,:))';
m(:,6)=points2homo(2,:)';
m(:,7)=points1homo(1,:)';
m(:,8)=points1homo(2,:)';
m(:,9)=1;
% last eigen vector
[~, ~, vm] = svd(m, 0);
f = reshape(vm(:, end), 3, 3)';
[u, s, v] = svd(f);
s(end) = 0;
f = u * s * v';
% denormalize
f = t2' * f * t1;
f = f / norm(f);
if f(end) < 0
    f = -f;
end
end
```

Eightpoints Random

```
%% Hough Transform and elbows detection
% Input: images
% Output: elbow position(2D coords)
%
% Written by Yuwei Qiu, summer interns @Upenn.


I=imread('2220.jpg');
BW=im2bw(I);
BW=edge(BW,'canny');
% I=imdilate(I,strel('disk',8));
% I=imerode(I,strel('disk',8));
% BW=bwmorph(I,'thin',20);
%%

[H,T,R] = hough(BW);
imshow(H,[],'XData',T,'YData',R,...
'InitialMagnification','fit');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
P = houghpeaks(H,10,'threshold',ceil(0.3*max(H(:))));
x = T(P(:,2)); y = R(P(:,1));
plot(x,y,'s','color','white');
lines = houghlines(BW,T,R,P,'FillGap',5,'MinLength',7);
figure,imshow(BW), hold on
max_len = 0;
%%

for k = 1:length(lines)
xy = [lines(k).point1; lines(k).point2];
plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

% Plot beginnings and ends of lines
plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

% Determine the endpoints of the longest line segment
len = norm(lines(k).point1 - lines(k).point2);
Len(k)=len
if ( len > max_len)
max_len = len
xy_long = xy
end
end
```



```
% highlight the longest line segment
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','blue');

%%
[L1 Index1]=max(Len(:))
Len(Index1)=0
[L2 Index2]=max(Len(:))

%%
%
x1=[lines(Index1).point1(1) lines(Index1).point2(1)]
y1=[lines(Index1).point1(2) lines(Index1).point2(2)]
x2=[lines(Index2).point1(1) lines(Index2).point2(1)]
y2=[lines(Index2).point1(2) lines(Index2).point2(2)]

%%
K1=(lines(Index1).point1(2)-lines(Index1).point2(2))/(lines(Index1).point1(1)-lines(Index1).point2(1))
K2=(lines(Index2).point1(2)-lines(Index2).point2(2))/(lines(Index2).point1(1)-lines(Index2).point2(1))
%%
hold on
[m,n] = size(BW); % ??
BW1=zeros(m,n);
b1=y1(1)-K1*x1(1)
b2=y2(1)-K2*x2(1)
for x=1:n
for y=1:m
if y==round(K1*x+b1)|y==round(K2*x+b2)
BW1(y,x)=1;

end

end
end
for x=1:n
for y=1:m

```

```
if ceil(K1*x+b1)==ceil(K2*x+b2)
y1=round(K1*x+b1)
BW1(1:y1-1,:)=0;
end
end
end
```

Hough

```
function [R, location] = motionfromF(F, intrinsic,inliers1, inliers2)
%% Linear PnP: generate R,t from images
%  Inputs: fundamental matrix F, 2-D homo coords of two images captured from two cameras
%  Outputs:
E = intrinsic * F * intrinsic';
%% decompose E
[U, D, V] = svd(E);
e = (D(1,1) + D(2,2)) / 2;
D(1,1) = e;
D(2,2) = e;
D(3,3) = 0;
E = U * D * V';
[U, ~, V] = svd(E);
W=[0 -1 0;
    1 0 0;
    0 0 1];
Z = [0 1 0;
    -1 0 0;
     0 0 0];
R1 = U * W * V';
R2 = U * W' * V';
if det(R1) < 0
    R1 = -R1;
end
if det(R2) < 0
    R2 = -R2;
end
Tx = U * Z * U';
t = -[Tx(3, 2), Tx(1, 3), Tx(2, 1)];
R=R1';
%% choose solution
negs = zeros(1, 4);
nInliers=size(inliers1, 1);
camMat0 = ([eye(3);[0 0 0]]*intrinsic)';
M1 = camMat0(1:3, 1:3);
c1 = -M1 \ camMat0(:,4);
for i = 1:4
    if i>2
        R=R2';
    end
    t=-t;
    camMat1 = ([R; t]*intrinsic)';
    M2 = camMat1(1:3, 1:3);
    c2 = -M2 \ camMat1(:,4);
    for j = 1:nInliers
        a1 = M1 \ [inliers1(j, :), 1]';
        a2 = M2 \ [inliers2(j, :), 1]';
        A = [a1, -a2];
        alpha = (A' * A) \ A' * (c2 - c1);
        p = (c1 + alpha(1) * a1 + c2 + alpha(2) * a2) / 2;
        m1(j, :) = p';
    end
    m2 = bsxfun(@plus, m1 * R, t);
    negs(i) = sum((m1(:,3) < 0) | (m2(:,3) < 0));
end
[~, idx] = min(negs);
if idx<3
    R=R1';
end
if idx==1 || idx ==3
    t=-t;
end
t = t ./ norm(t);

location = -t * R';
end
```

MotionfromFundamentalMatrix

References

[1] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[2] X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang. Multi-context attention for human pose estimation. In *CVPR*, 2017.

[3] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool. Dynamic filter networks. In *NIPS*, 2016.

[4] J. Deng, Y. Kaiyu, and A. Newell. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[5] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015.

[6] C. Doersch. *Supervision Beyond Manual Annotations for Learning Visual Representations*. PhD thesis, Carnegie Mellon University, 2016.

[7] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. Training generative neural networks via maximum mead discrepancy optimization. *UAI*, 2015.

[8] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[9] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.

[10] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrents network models for human dynamics. In *ICCV*, 2015.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Corville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[12] A. Gorban, H. Idrees, Y.-G. Jiang, A. Roshan Zamir, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://www.thumos.info/`, 2015.

[13] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Scholkopf, and A. Smola. A kernel two-sample test. *JMLR*, 2012.

[14] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 107(2):191–202, 2014.

[15] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *PAMI*, 2014.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004v1*, 2016.

[17] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, 2016.

[18] K. Kalchbrenner, A. van den Oord, K. Simonyan, I. Dnaihelka, O. Vinyals, A. Graves, and K. Kavikcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

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

[20] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *ICLR*, 2014.

[21] K. Kitani, B. Ziebart, D. Bagnell, and M. Hebert. Activity forecasting. In *ECCV*, 2012.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[23] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011.

[24] T. Lan, T.-C. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *ECCV*. 2014.

[25] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. *JMLR*, 2015.

[26] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, 2016.

[27] M. Mathieu, C. Couprie, and Y. Lecun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016.

[28] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[29] V. Pătrăucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *ICLR Workshop*, 2016.

[30] S. L. Pintea, J. C. van Gemert, and A. W. Smeulders. Déjà vu: Motion prediction in static images. In *ECCV*. 2014.

[31] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[32] V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *CVPR*, 2016.

[33] M. Ranzato, A. Szlam, J. Bruna, M. Mathieu, R. Collobert, and S. Chopra. Video (language) modeling: a baseline for generative models of natural videos. *arXiv preprint arXiv:1412.6604*, 2014.

[34] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *NIPS*, 2016.

[35] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

[36] N. Rhinehart and K. M. Kitani. Online semantic activity forecasting with DARKO. *arXiv preprint arXiv:1612.07796*, 2016.

[37] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese. Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV*, 2016.

[38] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *NIPS*, 2016.

[39] T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. *ICML*, 2015.

[40] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016.

[41] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

End

Aug.17th, 2017
@GRASP lab, Levine Hall, University of Pennsylvania