Kin2. User Guide

Juan R. Terven, Diana M. Cordova

1 Introduction

The Kin2 Toolbox for Matlab is an easy to use set of classes and functions that encapsulate the Microsoft Kinect 2 SDK. It is mostly based on C++ through mex files. The current version contains two classes and 30 functions divided in different features such as coordinate mapping, skeleton tracking, 3D reconstruction, and face gestures recognition.

2 Image Acquisition

Kinect three provides three sources of video images: RGB, depth, and infrared. The resolution of the RGB images is 1920 × 1080, the depth and infrared have resolution of 512 × 424.

The first step is to generate a `Kin2` object specifying the image sources we wish to obtain from the Kinect2. The available sources are `color, depth, infrared` and `body`. For example:

`k2 = Kin2('color', 'depht', 'infrared')`

generates a Kin2 object named `k2` that we can use to gather color, depth and/or infrared frames from the Kinect2.

To extract frames from the Kinect cameras we call the `updateKin2` function on the *Kin2* object. For example:

`validData = k2.updateKin2`

Note that this function gets data from the Kinect and save it in an internal buffer in Matlab. If there was a valid frame, the function returns 1 if not, returns 0.

To actually get the frames in Matlab variables we call the corresponding functions on the `Kin2` object: `getDepth`, `getColor`, and `getInfrared`.

Listing 1.1 shows how to create a Kin2 object to gather and display video from these three sources. In this example, we created we three figures, one for each source and update the content each valid frame.

Listing 1.1. Kin2 initialization and data acquisition

```
clear all
% Create Kinect 2 object and initialize it
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared' and 'body'
k2 = Kin2('color','depth','infrared');

% Create the figures for the images
```

```
figure, h1 = imshow(zeros(424,512,'uint16'),[0 4000]);
colormap('Jet'), colorbar;
figure, h2 = imshow(zeros(1080,1920,3,'uint8'),[]);
figure, h3 = imshow(zeros(424,512,'uint16'),[0 8000]);
set(gcf,'keypress','k=get(gcf,''currentchar'');');

% Loop until pressing 'q' on infrared figure
k=[];
while true
    % Read frames from Kinect
    validData = k2.updateData;

    % Before accessing the data,
    % make sure that a valid frame was acquired.
    if validData
        depth = k2.getDepth;
        color = k2.getColor;
        infrared = k2.getInfrared;

        set(h1,'CData',depth);
        set(h2,'CData',color);
        set(h3,'CData',infrared);
    end

     % If user presses 'q', on the infrared figure exit loop
    if ¬isempty(k)
        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

close all

% Close kinect object
k2.delete;
```

3 Coordinate Mapping

Coordinate Mapping is used to perform two tasks [1]: (1) map between locations on the depth image and their corresponding locations on the color image and viceversa and (2) project and unproject from 2D image space to 3D camera space. 2D image space refers to the depth or color image coordinates.

In practice, to perform any mapping, you should always update the Kin2 data calling `updateKin2` on the Kin2 object and see if there are valid data as shown in listing 1.

3.1 Depth Space

In depth space, $x = 1, y = 1$ corresponds to the top left corner of the image and $x = 512, y = 424$ is the bottom right corner of the image. When we need the depth value or z, simply sample the depth image at the row/column in question, and use that value as z (in millimeters). For example: if `depth` is a depth image obtained with `getDepth`, we get the depth as `z = depth(row,col)`.

Map Depth to Color The color sensor and depth sensor have a small offset between each other and different resolutions. To map between locations on the depth image and their corresponding locations on the color image Kin2 provides the function

```
ptColor = k2.mapDepthPoints2Color(ptDepth)
```

where `ptColor` and `ptDepth` are a $n \times 2$ matrix of image coordinates.

For a complete code example see the mapping demo in section 7.1.

Map Depth to Camera and Point Cloud A common operation on the depth image is to generate a 3D point cloud of the scene. For this, Kin2 provides a couple of methods. The first one consists of mapping a set of coordinates from depth space to camera space for example. This method is useful if you only need to map few points from depth space to camera space:

```
ptCam = k2.mapDepthPoints2Camera(ptDepth)
```

where `ptDepth` is a $n \times 2$ matrix (n x,y coordinates), and `ptCam` is $n \times 3$ matrix (n x,y,z coordinates).

If you want to get the complete point cloud, *i.e.* to map all the points from depth space to camera space, Kin2 provides:

```
pointCloud = k2.getPointCloud
```

where `pointcloud` is a $N \times 3$ camera space values. And $N = 512 \times 424 = 217088$. Note that coordinates with no depth information can not be mapped to camera space, and these coordinates are return as `inf` or `-inf` in camera space.

For a complete code example see the camera mapping demo in section 7.3 and the point cloud demo in section 7.2.

Map Depth to Infrared Depth and infrared images come from the same sensor so there is no offset and no change in resolution. To map a point from depth to infrared just sample the same coordinates.

3.2 Color Space

In color space, $x = 1, y = 1$ corresponds to the top left corner of the image and $x = 1920, y = 1080$ is the bottom right corner of the image. A common practice in RGBD processing is to obtain the depth or world coordinates of specific color space coordinates. For example if we find features in the color image and we want to know their depth or location in the world.

Map Color to Depth Kin2 provides the following function to map from color space to depth space.

```
ptDepth = k2.mapColorPoints2Depth(ptColor)
```

where `ptColor` and `ptDepth` are a $n \times 2$ matrix of image coordinates.

For a complete code example see the mapping demo in section 7.1.

Map Color to Camera Kin2 also provides a function to map from color space to camera space.

```
ptCam = k2.mapColorPoints2Camera(ptColor)
```

where `ptColor` is a $n \times 2$ matrix (n x,y coordinates), and `ptCam` is $n \times 3$ matrix (n x,y,z coordinates).

For a complete code example see the camera mapping demo in section 7.3.

3.3 Camera Space

Camera space refers to the 3D coordinate system (right-handed) used by Kinect. The coordinate system is defined as follows [1]:

- The origin (x=0, y=0, z=0) is located at the center of the IR sensor on Kinect.
- X grows to the sensors left
- Y grows up (note that this direction is based on the sensors tilt)
- Z grows out in the direction the sensor is facing
- 1 unit = 1 meter

Kinect2 SDK provides mapping capabilities between camera space and depth or color space that is 2D projections.

Map Camera to Depth Kin2 provides a function to map from 3D camera coordinates to depth image coordinates:

```
ptDepth = k2.mapCameraPoints2Depth(ptCam)
```

where `ptCam` is $n \times 3$ matrix (n x,y,z coordinates) and `ptDepth` is a $n \times 2$ matrix (n x,y coordinates).

For a complete code example see the camera mapping demo in section 7.3.

Map Camera to Color Kin2 also provides a function to map from 3D camera coordinates to color image coordinates.

```
ptColor = k2.mapCameraPoints2Color(ptCam)
```

where `ptCam` is $n \times 3$ matrix (n x,y,z coordinates) and `ptColor` is a $n \times 2$ matrix (n x,y coordinates).

For a complete code example see the camera mapping demo in section 7.3.

4 Body Tracking

Kin2 provides easy access to Kinect 2 body tracking capabilities.

To enable body tracking, you must indicate the *body* source when creating the Kin2 object. For example:

```
k2 = Kin2(’color’,’depth’,’body’)
```

creates a Kin2 object capable of fetching color and depth frames and also body tracking information.

Then to get the body data Kin2 provides the function `getBodies` that you can call after updating the Kin2 data. For example:

```
bodies = k2.getBodies
```

where `bodies` is a structure array with one element for each body (6 bodies maximum). Each element contains the following information:

– Position: a 3x25 matrix containing the x,y,z of the 25 joints in camera space coordinates
– TrackingState: state of each joint. These can be: NotTracked=0, Inferred=1, or Tracked=2
– LeftHandState: state of the left hand
– RightHandState: state of the right hand

The RightHandState and LeftHandState properties provide information about the state of each of the player’s hands. You can use this information to determine if a player is interacting with an object in the title’s world [2]. The states returned are: Open, Closed, Lasso, NotTracked, Unknown.

Once you have the joints position in 3D space you can map them to depth or color space for visualization using the `mapCameraPoints2Depth` or `mapCameraPoints2Color` functions described in section 3.3, for example:

```
posDepth = k2.mapCameraPoints2Depth(bodies(1).Position’)
posColor = k2.mapCameraPoints2Color(bodies(1).Position’)
```

For a complete example of body tracking see section 7.4.

4.1 Drawing Bodies

Kin2 provides functions to draw the bodies on the depth image or the color image freeing the developer from this tedious task.

To draw bodies on depth or color image you can use the `drawBodies` function as follows:

```
k2.drawBodies(d.ax,bodies,’depth’,5,3,15)
```

where the six parameters are the following:

1. Image axes. Figure axes obtained with Matlab `axes` function.
2. Bodies structure. Bodies returned by the `getBodies` function.
3. Destination image. ’depth’ or ’color’.
4. Joints’ size. Circle raddii.
5. Bones’ Thickness.
6. Hands’ Size.

For a complete example of body tracking and drawing see section 7.4.

5 Face Processing

Kin 2 provides two levels of face processing: simple face processing and HD face.

5.1 Simple Face Processing

With this functionality We can detect and track the face and recognize eight facial properties listed in table 1. The `getFaces` method returns a MATLAB structure array containing the following face data for each detected face:

- `FaceBox`: rectangle containing the user's face.
- `FacePoints`: five alignment points located on the user's face.
- `FaceRotation`: face orientation expressed as Euler angles: *pitch*, *yaw*, *roll*.
- `FaceProperties`: read-only key/value pairs. See Table 1 for a description of each property. Each property can have the following values: `Unknown`, `No`, `Maybe`, or `Yes`.

Table 1. Face Properties contained in the `FaceProperties` field of the structure array returned with `getFaces`.

Name	Description
`Happy`	The user is showing a smile.
`Engaged`	Combines results from Looking Away and Eye Closed to determine if user is engaged with content.
`WearingGlasses`	The user is wearing glasses.
`LeftEyeClosed`	The user's left eye is closed.
`RightEyeClosed`	The user's right eye is closed.
`MouthOpen`	The user's mouth is open.
`MouthMoved`	The user's mouth moved
`LookingAway`	Determines if the user is looking away from the content

5.2 HD Face Processing

HD Face provides amazing face processing capabilities: (1) face capture with 94 shape units and a high definition face model with 1347 mesh vertices, (2) face tracking of 17 animation units (AUs) expressed as a numeric weight varying between 0 and 1. *Kin2* provides access to these HD face capabilities with the `getHDFaces` method. This method returns a structure array containing the following fields for each detected face:

- `FaceBox`: rectangle containing the user's face.
- `FaceRotation`: face orientation expressed as Euler angles: *pitch*, *yaw*, *roll*.
- `HeadPivot`: center of the head, which the face may be rotated around. The origin is located at the Kinect's optical center, the Z axis is pointing towards a user, the Y axis is pointing up and the X axis is pointing to the right. The units are in meters.
- `AnimationUnits`: 17 animation units (AUs) expressed as a numeric weight varying between 0 and 1. Refer to [3] for a list of these animation units.
- `ShapeUnits`: 94 shape units (SUs) expressed as a numeric weight varying between -2 and +2. Refer to [4] for a list of these shape units, also called *shape deformations*.
- `FaceModel`: high definition face model with 1347 mesh vertices. Refer to [5] for a list of these high detail face points.

6 3D Reconstruction

The *Kin2* toolbox includes a version of *Kinect Fusion* taken from the Kinect for Windows SDK 2.0. To initialize the 3D reconstruction engine, *Kin2* provides the method `KF_init` that can be configured for different reconstruction's resolution and size. The method prototype is the following:

`KF_init(voxelsPerMeter,voxelsX,voxelsY,voxelsZ,gpu)`

For example if we set `voxelsPerMeter` to 256 we will have a resolution of $1000mm \div 256vpm = 3.9mm/voxel$, then if we set `voxelsX`, `voxelsY`, and `voxelsZ` to 384 we will have a reconstruction of $384voxels \div 256vpm = 1.5m$ wide reconstruction and we will require at least $384 \times 384 \times 384 \times 4$ bytes per voxel $= 227MB$ of memory. The final parameter `gpu` if *true*, the algorithm will use the GPU otherwise, it will use the CPU.

Once initialized, each call to `KF_update` updates the volume reconstruction with new views.

7 Complete Code examples

7.1 Mapping Demo

```
% MAPPINGDEMO Illustrates how to map points between depth ...
    and color images
%
% Usage:
%   - Press 'd' to select 5 points on the depth image. ...
    The selected points
%     will be mapped from depth to color and will be ...
    displayed on both
%     images in red.
%   - Press 'c' to select 5 point on the color image. The ...
    selected points
%     will be mapped from color to depth and will be ...
    displayed on both
%     images in green.
%   - Press 'q' to exit.
%
% Juan R. Terven, October 2015.
% jrterven@hotmail.com

addpath('Mex');
clear all
close all

% Create a Kin2 object and initialize it
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared' and 'body'
k2 = Kin2('color','depth');

% images sizes
d_width = 512; d_height = 424; outOfRange = 4000;
c_width = 1920; c_height = 1080;

% Color image is to big, let's scale it down
COL_SCALE = 0.5;

% Create matrices for the images
depth = zeros(d_height,d_width,'uint16');
color = ...
    zeros(c_height*COL_SCALE,c_width*COL_SCALE,3,'uint8');

% Images used to draw the markers
depthAdditions = zeros(d_height,d_width,3,'uint8');
colorAdditions = ...
    zeros(c_height*COL_SCALE,c_width*COL_SCALE,3,'uint8');

% depth stream figure
d.h = figure;
d.ax = axes('units','pixels','drawmode','fast');
d.im = imshow(depth,[0 255]);
title('Depth Source (press q to exit)')
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% color stream figure
c.h = figure;
c.im = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress
```

```


% Loop until pressing 'q' on any figure
k=[];

disp('Instructions:')
disp('Press d to select a point on the depth image')
disp('Press c to select a point on the color image')
disp('Press q on any figure to exit')

while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Copy data to Matlab matrices
        depth = k2.getDepth;
        color = k2.getColor;

        % update depth figure
        depth8u = uint8(depth*(255/outOfRange));
        depth8uc3 = repmat(depth8u,[1 1 3]);
        set(d.im,'CData',depth8uc3 + depthAdditions);

        % update color figure
        color = imresize(color,COL_SCALE);
        set(c.im,'CData',color + colorAdditions);
    end

    % If user presses 'd' enter to points selection mode ...
        on the depth image
    % If user presses 'c' enter to points selection mode ...
        on the color image
    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'d')
            figure(d.h);
            title('Clic the image to sample 5 points');

            % Grab 5 points
            [x,y] = ginput(5);
            disp('Input depth coordinates');
            disp([x y])
            % Draw the selected points in the depth image
            depthAdditions = ...
                insertMarker(depthAdditions,[x ...
                y],'Color','red');

            % Using the mapping, map the points from ...
                depth coordinates to color coordinates
            % Input and output: n x 2 matrix (n points)
            colorCoords = k2.mapDepthPoints2Color([x y]);
            colorCoords = colorCoords * COL_SCALE; % ...
                scale the color coordinates

            disp('Output color coordinates');
            disp(colorCoords);

            % Draw the output coordinates on the color image
            colorAdditions = insertMarker(colorAdditions, ...
                colorCoords,'Color','red','Size',10);

           k = [];
        elseif strcmp(k,'c')
            figure(c.h);
            title('Clic the image to sample 5 points');

            % Grab 5 points
            [x,y] = ginput(5);
            disp('Input color coordinates');
            disp([x y]);

            % Draw the selected points in the color image
            colorAdditions = ...
                insertMarker(colorAdditions,[x ...
                y],'Color','green','Size',5);

            % Using the mapping, map the points from ...
                color coordinates to depth coordinates
            % Input and output: n x 2 matrix (n points)
            depthCoords = ...
                k2.mapColorPoints2Depth([x/COL_SCALE ...
                y/COL_SCALE]);

            disp('Output depth coordinates')
            disp(depthCoords);

            % Drae the output coordinates on the depth image
            depthAdditions = ...
                insertMarker(depthAdditions,depthCoords,'Color','green');


            k = [];
        end

        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all
```

7.2 Point Cloud Demo

```
% POINTCLOUDDEMO Illustrates how to use the Kin2 class to ...
    get the
% pointcloud in camera space
%
% Juan R. Terven, January 2016.
% jrterven@hotmail.com

addpath('Mex');
clear all
close all

% Create Kinect 2 object and initialize it
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared' and 'body'
k2 = Kin2('depth');

% images sizes
depth_width = 512; depth_height = 424; outOfRange = 4000;


% Create matrices for the images
depth = zeros(depth_height,depth_width,'uint16');
pointCloud = zeros(depth_height*depth_width,3);

% depth stream figure
figure, h1 = imshow(depth,[0 outOfRange]);
title('Depth Source (press q to exit)')
colormap('Jet')
colorbar
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% point cloud figure
```

```
figure, hpc = ...
    plot3(pointCloud(:,1),pointCloud(:,2),pointCloud(:,3),'.');
title('Point Cloud (press q to exit)')
axis([-3 3 -3 3 0 4])
xlabel('X'), ylabel('Y'), zlabel('Z');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% Loop until pressing 'q' on any figure
k=[];

disp('Press q on any figure to exit')
while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Copy data to Matlab matrices
        depth = k2.getDepth;

        % update depth figure
        depth(depth>outOfRange) = outOfRange; % truncate ...
            depht
        set(h1,'CData',depth);

        pointCloud = k2.getPointCloud;
        set(hpc,'XData',pointCloud(:,1),'YData',pointCloud(:,2),'ZData',pointCloud(:,3

    end

    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'q');
            break;
        elseif strcmp(k,'p');
            pause;
        end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all;
```

7.3 Camera Mapping Demo

```
% MAPPINGTOCAMDEMO Illustrates how to map points between ...
    depth and color images
%
% Usage:
%   - Press 'd' to select a point on the depth image. The ...
    selected point
%     will be mapped from depth to camera and the ...
    resulting coordinates are
%     printed on command window. Then the camera ...
    coordinates are mapped
%     back to depth space and printed to command window.
%   - Press 'c' to select a point on the color image. The ...
    selected point
%     will be mapped from color to camera and the ...
    resulting coordinates are
%     printed on command window.  Then the camera ...
    coordinates are mapped
%     back to color space and printed to command window.
%   - Press 'q' to exit.
%
% Juan R. Terven, October 2015.
% jrterven@hotmail.com

addpath('Mex');
clear all
close all

% Create Kinect 2 object and initialize it
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared' and 'body'
k2 = Kin2('color','depth');

% images sizes
depth_width = 512; depth_height = 424; outOfRange = 4000;
color_width = 1920; color_height = 1080;

% Color image is to big, let's scale it down
COL_SCALE = 0.5;

% Create matrices for the images
depth = zeros(depth_height,depth_width,'uint16');
color = ...
    zeros(color_height*COL_SCALE,color_width*COL_SCALE,3,'uint8');
```

```

% Images used to draw the markers
depthAdditions = zeros(depth_height,depth_width,3,'uint8');
colorAdditions = ...
    zeros(color_height*COL_SCALE,color_width*COL_SCALE,3,'uint8');

% depth stream figure
h1 = figure;
hdepth = imshow(depth,[0 255]);
title('Depth Source (press q to exit)')
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% color stream figure
h2 = figure;
hcolor = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress


% Loop until pressing 'q' on any figure
k=[];

disp('Instructions:')
disp('Press d to select a point on the depth image')
disp('Press c to select a point on the color image')
disp('Press q on any figure to exit')

while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Copy data to Matlab matrices
        depth = k2.getDepth;
        color = k2.getColor;

        % update depth figure
        depth8u = uint8(depth*(255/outOfRange));
        depth8uc3 = repmat(depth8u,[1 1 3]);
        set(hdepth,'CData',depth8uc3 + depthAdditions);

        % update color figure
        color = imresize(color,COL_SCALE);
        set(hcolor,'CData',color + colorAdditions);
    end

    % If user presses 'd' enter to points selection mode ...
        on the depth image
    % If user presses 'c' enter to points selection mode ...
        on the color image
    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'d')
            figure(h1);
            title('Clic the image to sample a point');

            % Grab 1 points
            [x,y] = ginput(1);
            disp('Input depth coordinates');
            disp([x y])
            % Draw the selected points in the depth image
            depthAdditions = ...
                insertMarker(depthAdditions,[x ...
                y],'Color','red');

            % Map the point from depth coordinates to ...
                camera coordinates
            % Input: 1 x 2 matrix (1 points, x,y)
            % Output: 1 x 3 matrix (1 point, x,y,z)
            camCoords = k2.mapDepthPoints2Camera([x y]);

            disp('Mapped camera coordinates');
            disp(camCoords);

            % Map the resulting camera point back to ...
                depth space
            depthCoords = ...
                k2.mapCameraPoints2Depth(camCoords);
            disp('Mapped depth coordinates');
            disp(depthCoords);

           k = [];
        elseif strcmp(k,'c')
            figure(h2);
            title('Clic the image to sample 5 points');

            % Grab 1 point
            [x,y] = ginput(1);
            disp('Input color coordinates');
            disp([x/COL_SCALE y/COL_SCALE]);

            % Draw the selected point in the color image
            colorAdditions = ...
                insertMarker(colorAdditions,[x ...
                y],'Color','green','Size',5);

            % Map the points from color coordinates to ...
                camera coordinates
            % Input: 1 x 2 matrix (1 points, x,y)
            % Output: 1 x 3 matrix (1 point, x,y,z)
            camCoords = ...
                k2.mapColorPoints2Camera([x/COL_SCALE ...
                y/COL_SCALE]);

            disp('Mapped camera coordinates')
            disp(camCoords);

            % Map the resulting camera point back to ...
                color space
            colorCoords = ...
                k2.mapCameraPoints2Color(camCoords);
            disp('Mapped color coordinates');
            disp(colorCoords);

            k = [];
        end

        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all
```

7.4 Body Tracking Demo

```
% BODYDEMO Illustrates how to use the Kin2 object to get ...
    and draw the
% Skeleton data
%
% Juan R. Terven, October 31 2015.
% jrterven@hotmail.com

addpath('Mex');
clear all
```

```matlab
close all

% Create Kinect 2 object and initialize it
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared' and 'body'
k2 = Kin2('color','depth','body');

% images sizes
d_width = 512; d_height = 424; outOfRange = 4000;
c_width = 1920; c_height = 1080;

% Color image is to big, let's scale it down
COL_SCALE = 1.0;

% Create matrices for the images
depth = zeros(d_height,d_width,'uint16');
color = ...
    zeros(c_height*COL_SCALE,c_width*COL_SCALE,3,'uint8');

% depth stream figure
d.h = figure;
d.ax = axes('drawmode','fast');
d.im = imshow(zeros(d_height,d_width,'uint8'));
%hold on;

title('Depth Source (press q to exit)')
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% color stream figure
c.h = figure;
c.ax = axes;
c.im = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress
%hold on

% Loop until pressing 'q' on any figure
k=[];

disp('Press q on any figure to exit')
while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Copy data to Matlab matrices
        depth = k2.getDepth;
        color = k2.getColor;

        % update depth figure
        depth8u = uint8(depth*(255/outOfRange));
        depth8uc3 = repmat(depth8u,[1 1 3]);
        d.im = imshow(depth8uc3, 'Parent', d.ax);

        %set(d.im,'CData',depth8uc3);

        % update color figure
        color = imresize(color,COL_SCALE);
        c.im = imshow(color, 'Parent', c.ax);

        %set(c.im,'CData',color);

        % Get 3D bodies joints
        % getBodies returns a structure array.
        % The structure array (bodies) contains 6 bodies ...
            at most
        % Each body has:
        % -Position: 3x25 matrix containing the x,y,z of ...
            the 25 joints in
        %   camera space coordinates
        % -TrackingState: state of each joint. These can be:
        %   NotTracked=0, Inferred=1, or Tracked=2
        % -LeftHandState: state of the left hand
        % -RightHandState: state of the right hand
        bodies = k2.getBodies;

        % Number of bodies detected
        numBodies = size(bodies,2);
        disp(['Bodies Detected: ' num2str(numBodies)])

        % first body info:
        %disp(bodies(1).TrackingState)
        %disp(bodies(1).RightHandState)
        %disp(bodies(1).LeftHandState)

        % To get the joints on depth image space, you can ...
            use:
        %pos2D = ...
            k2.mapCameraPoints2Depth(bodies(1).Position');

        %To get the joints on color image space, you can use:
        %pos2D = ...
            k2.mapCameraPoints2Color(bodies(1).Position');

```

```matlab
        % Draw bodies on depth image
        % Parameters:
        % 1) image axes
        % 2) bodies structure
        % 3) Destination image (depth or color)
        % 4) Joints' size (circle raddii)
        % 5) Bones' Thickness
        % 6) Hands' Size
        k2.drawBodies(d.ax,bodies,'depth',5,3,15);

        % Draw bodies on color image
        k2.drawBodies(c.ax,bodies,'color',10,6,30);

    end

    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all;
```

7.5 Face Processing Demo

```matlab
% FACEDEMO Illustrates how to use the Kin2 object to get ...
    and draw the
% face data
%
% Note: You must add to the windows path the bin ...
    directory containing the
%       Kinect20.Face.dll.
%       For example: C:\Program Files\Microsoft ...
    SDKs\Kinect\v2.0_1409\bin
%
% Juan R. Terven, January 2016.
% jrterven@hotmail.com

addpath('Mex');
clear all
close all

```

```
% Create Kinect 2 object and initialize it
% Available sources: 'color', 'depth', 'infrared', ...
    'body_index', 'body',
% 'face' and 'HDface'
k2 = Kin2('color','face');

% images sizes
c_width = 1920; c_height = 1080;

% Color image is to big, let's scale it down
COL_SCALE = 1.0;

% Create matrices for the images
color = ...
    zeros(c_height*COL_SCALE,c_width*COL_SCALE,3,'uint8');

% color stream figure
c.h = figure;
c.ax = axes;
c.im = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% Loop until pressing 'q' on any figure
k=[];

disp('Press q on any figure to exit')
while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Get color frame
        color = k2.getColor;

        % Get the faces data
        % faces is a structure array with at most 6 ...
            faces. Each face has
        % the following fields:
        % - FaceBox: rectangle coordinates representing ...
            the face position in
        %   color space. [left, top, right, bottom].
        % - FacePoints: 2 x 5 matrix representing 5 face ...
            landmarks:
```

```
        %   left eye, right eye, nose, right and left ...
            mouth corners.
        % - FaceRotation: 1 x 3 vector containing: pitch, ...
            yaw, roll angles
        % - FaceProperties: 1 x 8 vector containing the ...
            detection result of
        %   each of the face properties.
        %   The face properties are:
        %   Happy, Engaged, WearingGlasses, ...
            LeftEyeClosed, RightEyeClosed,
        %   MouthOpen, MouthMoved, LookingAway
        %   The detection results are:
        %   Unknown = 0, No = 1, Maybe = 2, Yes = 3;
        faces = k2.getFaces;

        % update color figure
        color = imresize(color,COL_SCALE);
        c.im = imshow(color, 'Parent', c.ax);

        % Display the faces data:
        % Parameters:
        % 1) image axes
        % 2) faces structure obtained with getFaces
        % 3) face landmarks size (radius)
        % 4) display text information?
        % 5) information font size in pixels
        k2.drawFaces(c.ax,faces,5,true,20);

    end

    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all;
```

7.6 HD Face Processing Demo

```
% FACEHDDEMO Illustrates how to use the Kin2 object to ...
    get and display the
```

```
% HD face data
%
% Note: You must add to the windows path the bin ...
    directory containing the
%       Kinect20.Face.dll.
%       For example: C:\Program Files\Microsoft ...
    SDKs\Kinect\v2.0_1409\bin
%
% Juan R. Terven, January 2016.
% jrterven@hotmail.com

addpath('Mex');
clear all
close all

% Create Kinect 2 object and initialize it
% Available sources: 'color', 'depth', 'infrared', ...
    'body_index', 'body',
% 'face' and 'HDface'
k2 = Kin2('color','HDface');

% images sizes
c_width = 1920; c_height = 1080;

% Color image is to big, let's scale it down
COL_SCALE = 1.0;

% Create matrices for the images
color = ...
    zeros(c_height*COL_SCALE,c_width*COL_SCALE,3,'uint8');

% color stream figure
c.h = figure;
c.ax = axes;
c.im = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

model = zeros(3,1347);
figure, hmodel = plot3(model(1,:),model(2,:),model(3,:),'.');
%axis([-1 1 -1 1 -1 1])
title('HD Face Model (press q to exit)')
xlabel('X'), ylabel('Y'), zlabel('Z');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% Loop until pressing 'q' on any figure
k=[];

```

```
disp('Press q on any figure to exit')
while true
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Get color frame
        color = k2.getColor;

        % update color figure
        color = imresize(color,COL_SCALE);
        c.im = imshow(color, 'Parent', c.ax);

        % Get the HDfaces data
        % the output faces is a structure array with at ...
            most 6 faces. Each face has
        % the following fields:
        % - FaceBox: rectangle coordinates representing ...
            the face position in
        %   color space. [left, top, right, bottom].
        % - FaceRotation: 1 x 3 vector containing: pitch, ...
            yaw, roll angles
        % - HeadPivot: 1 x 3 vector, computed center of ...
            the head,
        %   which the face may be rotated around.
        %   This point is defined in the Kinect body ...
            coordinate system.
        % - AnimationUnits: 17 animation units (AUs). ...
            Most of the AUs are
        %   expressed as a numeric weight varying between ...
            0 and 1.
        %   For details see ...
            https://msdn.microsoft.com/en-us/library/microsoft.kinect.face.faceshapean
        % - ShapeUnits: 94 hape units (SUs). Each SU is ...
            expressed as a
        %   numeric weight that typically varies between ...
            -2 and +2.
        %   For details see ...
            https://msdn.microsoft.com/en-us/library/microsoft.kinect.face.faceshapede
        % - FaceModel: 3 x 1347 points of a 3D face model ...
            computed by face capture
        faces = k2.getHDFaces;

        % Display the HD faces data and face model(1347 ...
            points):
        % Parameters:
```

```
        % 1) image axes
        % 2) faces structure obtained with getFaces
        % 3) display HD face model vertices(1347 points)?
        % 4) display text information (animation units)?
        % 5) text font size in pixels


        % Plot face model points
        if size(faces,2) > 0
             model = faces(1).FaceModel;
             set(hmodel,'XData',model(1,:),'YData',model(2,:),'ZData',model(3,:));
         end
    end

    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'q'); break; end;
    end

    pause(0.02)
end

% Close kinect object
k2.delete;

close all;
```

7.7 Kinect Fusion Demo

```
% KINECTFUSIONDEMO Illustrates how to use the Kin2 to ...
    perform 3D
% reconstruction
%
% Note: You must add to the windows path the bin ...
    directory containing the
%       Kinect20.Fusion.dll
%       For example: C:\Program Files\Microsoft ...
    SDKs\Kinect\v2.0_1409\bin
%
% Juan R. Terven, January 2016.
% jrterven@hotmail.com

addpath('../Mex');
clear all
close all

% Create Kinect 2 object and initialize it
```

```matlab
% Select sources as input parameters.
% Available sources: 'color', 'depth', 'infrared', ...
    'body_index', 'body',
% 'face' and 'HDface'
k2 = Kin2('color','depth');

k2.KF_init;

% images sizes
depth_width = 512; depth_height = 424; outOfRange = 4000;
color_width = 1920; color_height = 1080;

% Color image is to big, let's scale it down
colorScale = 0.4;

% Create matrices for the images
depth = zeros(depth_height,depth_width,'uint16');
volume = zeros(depth_height,depth_width,3,'uint8');
color = ...
    zeros(color_height*colorScale,color_width*colorScale,3,'uint8');

% depth stream figure
figure, h1 = imshow(depth,[0 outOfRange]);
title('Depth Source (press q to exit)')
colormap('Jet')
colorbar
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% color stream figure
figure, h2 = imshow(color,[]);
title('Color Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% volume stream figure
figure, h3 = imshow(volume,[]);
title('Volume Source (press q to exit)');
set(gcf,'keypress','k=get(gcf,''currentchar'');'); % ...
    listen keypress

% Loop until pressing 'q' on any figure
k=[];
timedFrames = zeros(1,100);
disp('Press q on any figure to exit')
for i=1:100
    tic
    % Get frames from Kinect and save them on underlying ...
        buffer
    validData = k2.updateData;

    % Before processing the data, we need to make sure ...
        that a valid
    % frame was acquired.
    if validData
        % Copy data to Matlab matrices
        depth = k2.getDepth;
        color = k2.getColor;

        k2.KF_update;
        volume = k2.KF_getVolumeImage;

        % update depth figure
        depth(depth>outOfRange) = outOfRange; % truncate ...
            depht
        set(h1,'CData',depth);

        % update color figure
        color = imresize(color,colorScale);
        set(h2,'CData',color);

        % update infrared figure
        set(h3,'CData',volume);
    end

    % If user presses 'q', exit loop
    if ¬isempty(k)
        if strcmp(k,'q'); break; end;
        if strcmp(k,'m');
            mesh = k2.KF_getMesh;
            k=[];
        end;
    end

    pause(0.02)
    timedFrames(i) = toc;
end

% Close kinect object
k2.delete;

close all;
```

References

1. Microsoft: Coordinate mapping. https://msdn.microsoft.com/en-us/library/dn785530.aspx (January 2016)
2. Microsoft: Body tracking. https://msdn.microsoft.com/en-us/library/dn799273.aspx (January 2016)

3. Network, M.D.: Faceshapeanimations enumeration. https://msdn.microsoft.com/en-us/library/ microsoft.kinect.face.faceshapeanimations.aspx (January 2016)
4. Network, M.D.: Faceshapedeformations enumeration. https://msdn.microsoft.com/en-us/library/ microsoft.kinect.face.faceshapedeformations.aspx (January 2016)
5. Microsoft: Highdetailfacepoints enumeration. https://msdn.microsoft.com/en-us/library/ microsoft.kinect.face.highdetailfacepoints.aspx (January 2016)