Walking Aid System for Blind people In Smart House(WASBISH)

Zhe Sun, S2029081   Divya S. Avalur, s2082330

April 21, 2011

Contents

1 Introduction

The WASBISH has been developed for the blind people living in the smart homes. The main goal of WASBISH is to provide a walking aid for the blind. We all know that blind people face a lot of difficulty to search for the correct path inside the house. In any place which just place different markers, the blind people who hold the Kinect with their hands can use WASBISH system. The most important aspect of WASBISH is "Path Detection". User inputs the start position and destination, then the system can tell the blind people which direction should they go and the distance between the target point and the user. The system can get a shortest path automatically and notify the information to the user in real time. The other important aspect is the "Obstacle Detection". This is basically developed for the safety of the blind persons. If any obstacle is encountered in the path it alerts the person by making vibrations to his mobile phone. In this project we use Android phones. This will prevent the blind person from falling, colliding with the wall and other dangerous accidents.

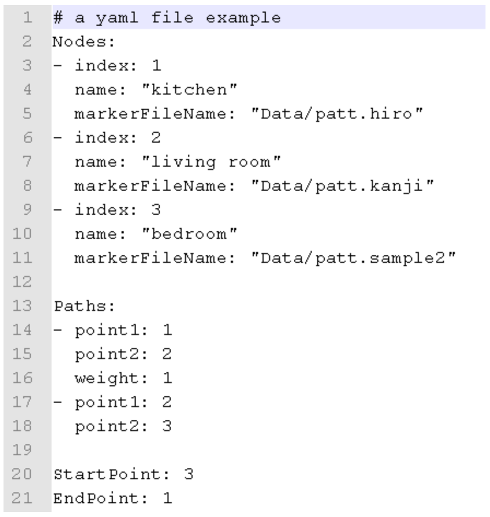
2 Terminology

- WASBISH
  Short for Walking Aid System for Blind people In Smart House
- Marker
  Markers are the squares that the WASBISH can recognise and tracks in the image stream captured by Kinect. Markers are the physical patterns that can be created or printed. They must be square, have a continuous border (generally either full black or pure white) and they must sit on a background of contrasting colour (generally the opposite of the border colour). By default, the border thickness is 25% of the length of an edge of the marker. Figure 1 is an example of marker which will be used in the demonstration.
- Node
  A node is a specific spot which is tagged by a marker. User can assign some attributions of a node.
- Path
  A path is two direction way to connect two nodes. It has the same meaning with graph in graph theory.
- Smart House
  In this report, Any space which is represented by the Node and Path and tagged by the marker is called Smart House.
- YAML
  YAML is a human friendly data serialization standard for all programming languages. In this project YAML file is used to describe a Smart House.

3 Use method

1. Construct Smart House
   (a) Define, generate and print marker by ARToolKit
   (b) Paste the markers where the user expect in the house
   (c) Edit YAML file: nodes, paths
2. Run Android phone side WASBISH application, don't forget open Wifi of Android phone.
3. Run PC side WASBISH application.
   (a) Input start point and destination by edit YAML file.
   (b) Read YAML file. If user does not input YAML file name, "house.yaml" is the default file.

An example of Marker



```
# a yaml file example
Nodes:
- index: 1
  name: "kitchen"
  markerFileName: "Data/patt.hiro"
- index: 2
  name: "living room"
  markerFileName: "Data/patt.kanji"
- index: 3
  name: "bedroom"
  markerFileName: "Data/patt.sample2"

Paths:
- point1: 1
  point2: 2
  weight: 1
- point1: 2
  point2: 3

StartPoint: 3
EndPoint: 1
```

An example of YAML file

Figure 1

(c) Generate the graph and shortest path

(d) Kinect start up

(e) Application state machine initial, speak “Begin working”.

(f) Set next node, application speak “Next is (the name of the node)”
- If the node is the destination, jump to step 3j.

(g) Blind people begin walking

(h) Obstacle detection
- If obstacle object is less than a pre-configure distance(means obstacle is near by the user), mobile phone make a short vibration.
- If obstacle object is less than a pre-configure distance(means very close to the obstacle), mobile phone make a long vibration.

(i) Detect the expected node
- If node is not detected by Kinect, speak “(the name of the node) is not in the view”, jump to step 3g.
- If node is detected by Kinect, speak “(the name of the node) detected”
  - If the distance is less than an pre-configure distance, speak “(the name of the node) arrive”, jump to step 3f
  - If the distance is greater than an pre-configure distance, speak the distance and the angle, jump to step 3g.

(j) End the application, release all the resource

4 Features

4.1 User-define house by YAML file

- User input YAML file
  In WASBISH, YAML file is used to represent a house, start/destination points. YAML file name is an input parameter for the application. If no file input, “house.yaml” is the default YAML file. Figure 1 is sample YAML file, self description, easy syntax and nice portability.

- User-defined Node
  User can define the name and marker pattern for each node. The name will be speak when system is running. The marker pattern which can be recognized by the application should be printed and pasted in the house.
- User-define path
  User can define the path which connect two nodes. “weight” is an optional attribution. Sometimes user has a preferent path for some reason, “weight” solves this kind of requirement.
  The application has a maximal distinguish distance for marker recognition. This distance depends on the size of the marker. It requires the path length shorter than the nearest maximal distinguish distance.
- User-defined start node and destination node
  User can define the start node and destination node freely, but the two point should be connected in the graph.

4.2 Shortest walking path

By parsing the user-defined YAML file, the application generates a graph of the smart house and the Dijkstra algorithm is used to get the shortest walking part.

4.3 Node recognition, distance and orientation detection, and voice notification

- The application can recognize the node in the view of Kinect. The node distinguishing distance depend on the size of the marker. A 20cm $\times$ 20cm can be recognized in 3.5 meters and a 10cm $\times$ 10cm can be recognized in about 2 meters.
- When a node is captured by the application, the distance and orientation(angle between Kinect and node) can be calculated using the raw depth data from Kinect in the same time. In this way, the user get to know where the target node is clearly.
- Application can read aloud the distance and orientation to the blind people.

4.4 State machine drives the aid walking procedure

A finite state machine maintains and drives the the aid walking procedure. The advantages of using FSM is:

- Filter unstable marker in/out the view event
- Program is easy to maintain
- Clear software architecture

4.5 Obstacle detection and notification

Because of The sensor of Kinect has an angular field of view of 57 degree horizontally and 43 degree vertically, the application only choose a 1m $\times$ 1m and distance less than 1.2m as the detection area.
The application can detect if there is obstacle object in the detection area per 600ms. The raw depth is captured by Kinect, then use some image processing method(threshold, dilation, erosion, edge, etc) to recognize the obstacle object and get the position. If an obstacle object do exist, the application send message to Android phone side application and let the mobile phone vibration. The vibration pattern is configurable. If the obstacle is near, the vibration is long; if far the vibration is short.

4.6 Real-time vidoe show for the view, marker and obstacle

The application can show the image captured by Kinect. When marker is detected, marker is tagged as blue quad in the video. The same as obstacle just it is tagged as red color. In this way, software developer can debug and give the demonstration more easily.

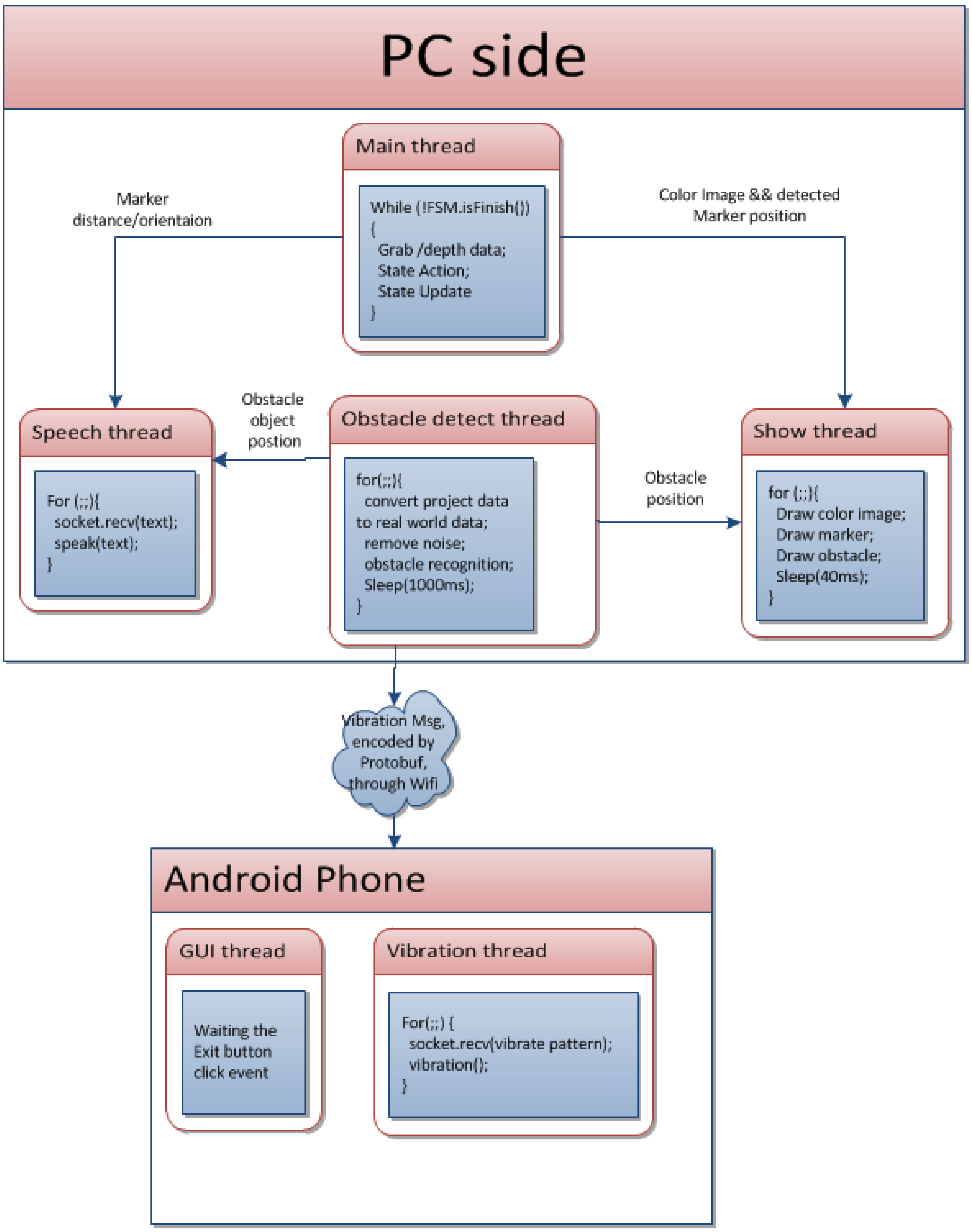
5 Developing environment

5.1 Hardware and software

- Windows XP Service Package 3
- Microsoft Visual C++ 2008
- Eclipse v3.5
- Android SDK v2.1
- Kinect and third party driver SensorKinect-Win32-5.0.0
- Android phone
- Wifi wireless router

5.2 Third party tool/library

- Yaml-cpp v1.2
  C++ YAML parser and emitter. We use it to parse the YAML file and construct the House class.
  `http://code.google.com/p/yaml-cpp/`
- ARToolKit v2.72
  The Augmented Reality library ARToolKit is a software library for building Augmented Reality applications. We use it to pre-define the markers and detect markers in the image scene in real-time.
  `http://www.hitl.washington.edu/artoolkit/`
- Microsoft Speech Application Interface(SAPI) v5
  Microsoft text-to-speech API. We use it to notify the user by sound
- OpenNI API v1.1.0
  OpenNI is a middleware for kinect development. We use the APIs to get image and depth data. The depth raw data is a array of depth data[0..2047] with the size of 640×480.
  `http://www.openni.org/`
- Protobuf v2.3
  Protocol Buffers(Protobuf) are a way of encoding structured data in an efficient yet extensible format. We use it to define, encode and decode message between PC and android side. The most key role is serialize the message to avoid some data transmission problem between different architecture CPU, like big/little endian. The reason we choose Protobuf is its light-weight and stability.Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats.
  `http://code.google.com/p/protobuf/`
- OpenCV v2.1
  Open Source Computer Vision(OpenCV) is a library of programming functions for real time computer vision. We use it to do some image processing operation like dilation, erosion and find contours.
  `http://opencv.willowgarage.com/wiki/`
- OpenGL
  We use OpenGL API show the scene, marker and obstacle video.


PC side
Main thread
Marker distance/orientaion
Color Image && detected Marker position
Speech thread
Obstacle object postion
Obstacle detect thread
Show thread
Obstacle position
Vibration Msg, encoded by Protobuf, through Wifi
Android Phone
GUI thread
Waiting the Exit button click event
Vibration thread

```
While (!FSM.isFinish())
{
  Grab /depth data;
  State Action;
  State Update
}
```

```
For (;;){
  socket.recv(text);
  speak(text);
}
```

```
for(;;){
  convert project data
to real world data;
  remove noise;
  obstacle recognition;
  Sleep(1000ms);
}
```

```
for (;;){
  Draw color image;
  Draw marker;
  Draw obstacle;
  Sleep(40ms);
}
```

```
For(;;) {
  socket.recv(vibrate pattern);
  vibration();
}
```

Figure 2: System overview

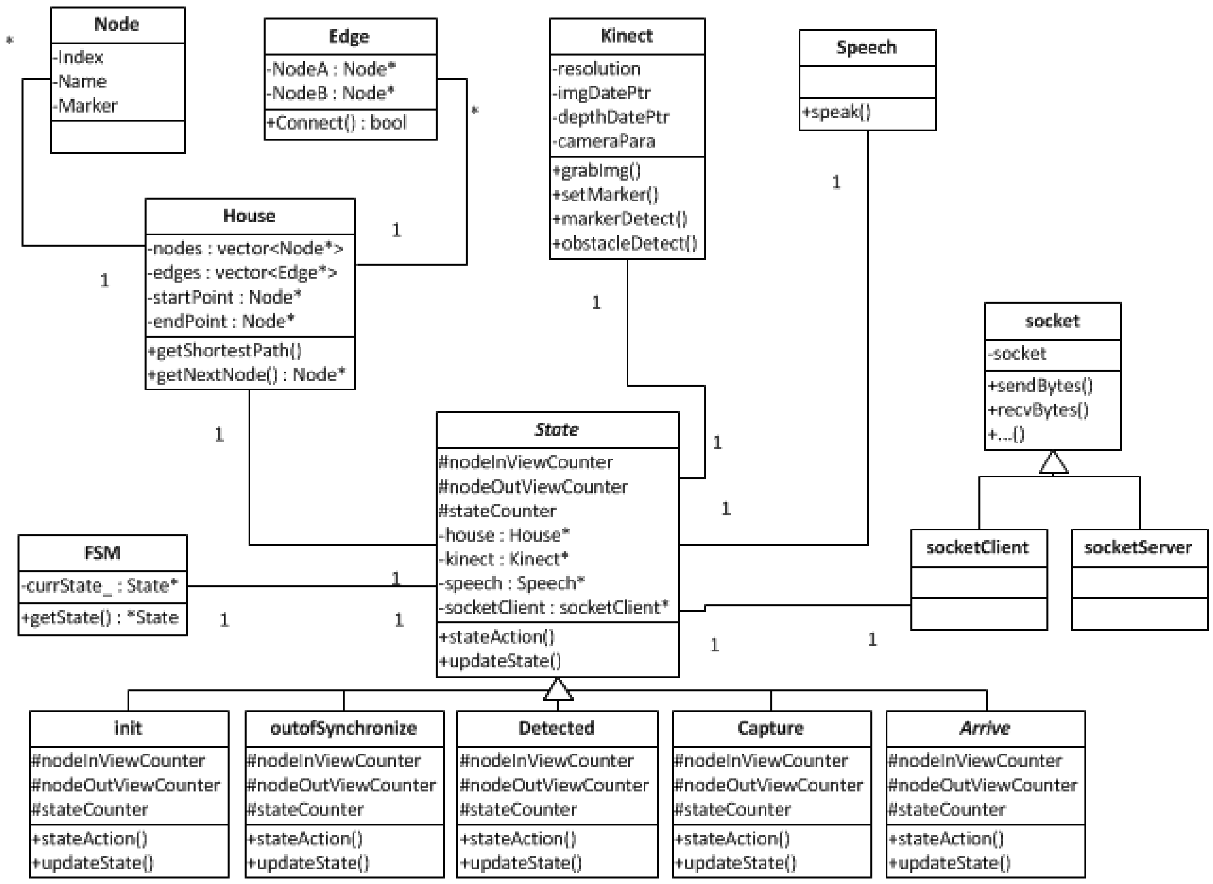


Figure 3: UML of main classes

6 System architecture

The system diagram is figure 2. The pc side application is in charge of the main function of walking aid and the android side are responsible for obstacle notification using vibration.
In the PC side application, there are 4 threads. The main thread control work flow by the state machine, meanwhile other 3 threads execute some time-cost task. Share memory, Semaphore and socket are used as thread synchronization and inter-thread communication, mutex-exclusive semaphores are used to protect the critical section. The functions of each thread are listed below:

- Main thread
  - Initial all the data and instance of class
  - Create other three thread
  - Sample color/depth image per 40ms
  - Maintain the finite state machine
  - Tell Speech thread the text which it want to speak like state updating and marker position
  - When the state machine finish, kill all the thread and release all the resource
- Speech thread
  - Create a UDP socket server
  - Suspend the thread until receive UDP datagram which contain the text
  - speak the received text
- Obstacle detection thread
  - Poll the depth data in the share memory
  - Convert the depth data from project view to real world view
  - Filter the data, only the detection area depth data is left
  - Use image processing method to remove noise and detect the remarkable obstacle
  - Detect the obstacle distance, encode vibration message by protobuf and send to Android phone through Wifi.
  - Approximate the obstacle object by polygon, and update the share memory which is used in show thread.
- Show thread
  - Poll the share memory
  - When the image data update, show in the screen
  - When the marker detected, show in the screen
  - When obstacle is detected, show in the screen

In the Android side application, there are 2 threads. The main thread initial the application, show the GUI, create the vibration thread and waiting the Exit button click event. The vibration thread is a socket server, it listen and receive the vibration message from PC side application, decode the message use protobuf and vibrate as the instruction from PC side application.

7 Some technical detail

7.1 Project scale and the number of classes

The application in PC side includes 24 files, 2600 lines of code, 18 classes. The application in Android side includes 2 files, 550 lines of code, 4 classes.

7.2 UML of the mainly class

Figure 3 shows UML of the mainly class. The State Pattern and Singleton Pattern in Design Pattern are used.

- Singleton Pattern
  Class House is a class which describe the smart house. Class Kinect supply a group of method to visit Kinect like get color/depth image, get distances and detect obstacle. Class Speech supply a group of method to make PC speak. These classes can be regarded as an group of utilities. In the project, Singleton pattern is used to get the facility: only one instance per class and only one visit point globally.

- State Pattern
  The state machine is as Figure 4. Five states are used to represent different stage when aid walking. They are ***Init***, ***OutofSychronize***, ***Detect***, ***Capture***, and ***Reach***. State pattern is used to implement the state machine and it make the code have clear structure and easy to maintain.

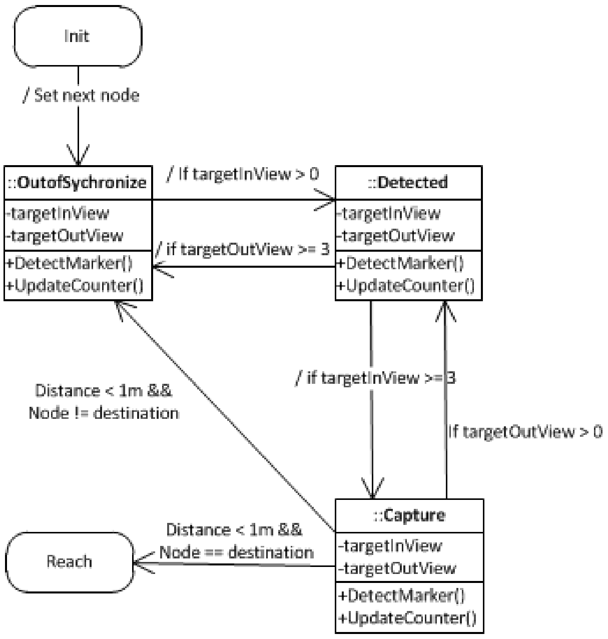


Figure 4: State transition condition

7.3 Message between PC and Android

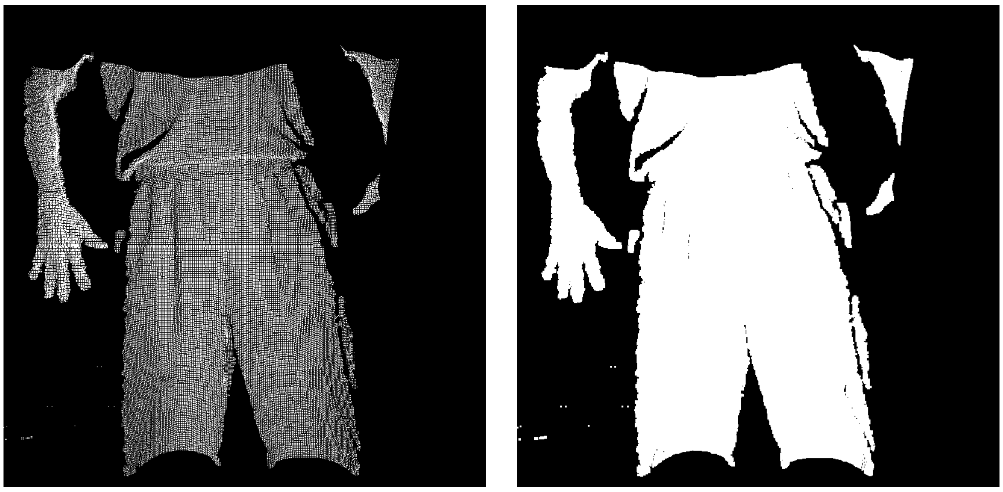
The message between PC and Android is define like below. The android phone only support vibration function, so only vibration and vibration pattern are defined in the message. But it is very easy to expend the function.

```
package mobileProtocol;

option java_package = "rug.ucProject";
option java_outer_classname = "mobileProtocol";

enum CmdType {
  VIBRATION = 0;
}

message ucMsg {
  required CmdType type = 1;
  repeated  int64 vPattern = 2;
}
```

Message can be compiled by Protobuf v2.3 and two new class files which supply encode, decode, serialize and deserialize methods are generated. One class file is C++ class for PC side and another class file is Java class for Android side.

7.4 Depth image noise remove

When convert the depth date from project view to real world view, a lot of texture/noise is generated. Here an erosion and a dilation is use to remove the noise. Figure 5 show this. About the isolate noise in the left bottom side, we can set a threshold of the area of the connects components. If less than 36 pixel, for example, we think the connect component is noise and remove them.



real world depth image   After noise remove

Figure 5

8 Future work

8.1 Start/destination point input by voice

Obviously, blind people can not update the start/destination point by editing YAML file. Voice input is apparently an essential component. An approach is using Speech Application Programming Interface.

8.2 Android text-to-speech

Now the Android phone only vibrate when obstacle object is detected. By using Android text-to-speech API, the Android phone can also speak the position of obstacle object. Due to using the Google Protobuf as transmission layer, this feature can be expanded easily.

8.3 Mobile Kinect

Besides the power from USB, a extra power supply is essential for Kinect working. It limits the mobility of WASBISH – Just think of the long power wire behind Kinect. Since the output of the power supply is 12V and 1.08A, using battery is possible. Now there is already some tutorial in the Internet like `http://www.ros.org/wiki/kinect/Tutorials/Adding%20a%20Kinect%20to%20an%20iRobot%20Create`. But due to this way will damage/modify the splitter permanently, we give up this method. In the demonstration we will use cart to move the kinect with long power wire behind or just move the maker and obstacle to simulate the walking.

8.4 Outlook if using 3D image sensor

Using Kinect as WASBISH is just a lab toy but not a production – huge volume, high power consumer, poor mobility. If the 3D image sensor can be purchased in an acceptable price in the market, 3D image sensor will substitute for Kinect in WASBISH system. All the kernel components will be integrated in one PCB board, 3D image sensor chips array, CPU, memory, control chips and electronic circuit, bluetooth/wifi interface, battery etc. Plus the 3D image/depth camera, wireless earphone and vibration equipment(these two can be substitute by mobile phone) the whole hardware system will be no larger than a pair of glass. Then, it is a tiny, mobile and easy-to-use system.

9 Conclusion

The main aim of our project is to develop an application which can be helpful for the disabled(in our case blind) people. We chose the context of our project as SMART homes because they make use of the latest technologies like 3D image sensors and other devices. This walking aid is very user-friendly and efficient. It can be used in all environments – home, workplace and public place – as long as the place is tagged and pasted by the markers and the nodes and paths are configured by YAML file. Hence it is robust as well. The operation of this application is very simple and easy to use. It does not require any basic training. It also makes use of very less hardware. This project when implemented in SMART homes will definitely prove very beneficial for the blind.