Codes
main.py

```
import sys
sys.path.remove('/opt/ros/kinetic/lib/python2.7/dist-packages')
import cv2
filepath = "../catkin_ws/photo.jpeg"
#filepath = "../hhh.jpg"

maxheight = 0.4
minheight = 0.1

class face:
      def detect(self):
            img = cv2.imread(filepath) # 读取图片
            if img is None:
                  #print('no face')
                  return "noface"
            size = img.shape
            print(size)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转换灰色
            # OpenCV人脸识别分类器
            classifier = cv2.CascadeClassifier(
"haarcascade_frontalface_default.xml" )
            color = (0, 255, 0) # 定义绘制颜色
            # 调用识别人脸
            faceRects = classifier.detectMultiScale( gray,
scaleFactor=1.3, minNeighbors=5, minSize=(32, 32))
            L = []
            R = []
            T = []
            B = []
            if len(faceRects): # 大于0则检测到人脸
                  for faceRect in faceRects: # 单独框出每一张人 脸
                        x, y, w, h = faceRect
                        # 框出人脸
                        L.append(x)
                        R.append(x+w)
                        B.append(y)
                        T.append(y+h)
                        cv2.rectangle(img, (x, y), (x + h, y + w), color,
2)
            else :
                  print('no face...')
                  return "noface"
            print(min(L), max(R), min(B), max(T))
            range = [min(L)/size[1], max(R)/size[1], min(B)/size[0],
max(T)/size[0]]
            print(range)
            cv2.rectangle(img, (min(L), min(B)), (max(R), max(T)),
color, 2)

            cv2.imshow("image", img) # 显示图像
            #c = cv2.waitKey(10)
            #cv2.waitKey(0)
            #cv2.destroyAllWindows()
            if (range[3] - range[2]) > maxheight:
                  return "back"
            if (range[3] - range[2]) < minheight:
                  return "forward"
            if range[0] > ((1-range[1]) +0.3):
                  return "left"
            if range[0] < ((1 - range[1]) -0.3):
                  return "right"
            if range[3] > 0.5:
                  return "down"
            if range[2] < 0.1:
                  return "up"
            return "ok"

if __name__ == '__main__':
    a = face()
    a.detect()
```

face.py

```
import sys
sys.path.remove('/opt/ros/kinetic/lib/python2.7/dist-packages')
import cv2
filepath = "../catkin_ws/photo.jpeg"
#filepath = "../hhh.jpg"

maxheight = 0.4
minheight = 0.1

class face:
      def detect(self):
            img = cv2.imread(filepath) # 读取图片
            if img is None:
```

```
            #print('no face')
            return "noface"
        size = img.shape
        print(size)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转换灰色
        # OpenCV人脸识别分类器
        classifier = cv2.CascadeClassifier(
"haarcascade_frontalface_default.xml" )
        color = (0, 255, 0) # 定义绘制颜色
        # 调用识别人脸
        faceRects = classifier.detectMultiScale( gray,
scaleFactor=1.3, minNeighbors=5, minSize=(32, 32))
        L = []
        R = []
        T = []
        B = []
        if len(faceRects): # 大于0则检测到人脸
            for faceRect in faceRects: # 单独框出每一张人 脸
                x, y, w, h = faceRect
                # 框出人脸
                L.append(x)
                R.append(x+w)
                B.append(y)
                T.append(y+h)
                cv2.rectangle(img, (x, y), (x + h, y + w), color,
2)
        else :
            print('no face...')
            return "noface"
        print(min(L), max(R), min(B), max(T))
        range = [min(L)/size[1], max(R)/size[1], min(B)/size[0],
max(T)/size[0]]
        print(range)
        cv2.rectangle(img, (min(L), min(B)), (max(R), max(T)),
color, 2)

        cv2.imshow("image", img) # 显示图像
        #c = cv2.waitKey(10)
        #cv2.waitKey(0)
        #cv2.destroyAllWindows()
        if (range[3] - range[2]) > maxheight:
            return "back"
        if (range[3] - range[2]) < minheight:
            return "forward"
        if range[0] > ((1-range[1]) +0.3):
            return "left"
        if range[0] < ((1 - range[1]) -0.3):
            return "right"
        if range[3] > 0.5:
            return "down"
        if range[2] < 0.1:
            return "up"
        return "ok"

if __name__ == '__main__':
    a = face()
    a.detect()
```

image_fetch.py

```
import rospy
import os
from geometry_msgs.msg import Twist
from tf2_msgs.msg import TFMessage
from sensor_msgs.msg import LaserScan, Image
from nav_msgs.msg import Odometry
import roslib
from cv_bridge import CvBridge, CvBridgeError
import actionlib
from actionlib_msgs.msg import GoalStatus, GoalStatusArray
from math import acos, asin, sin, cos, pi
import cv2


class Image_fetch():
    def __init__(self):
        rospy.init_node('imagefetch')
        self.img_sub = rospy.Subscriber('/usb_cam/image_raw', Image,
self.image_callback)
      self.myphoto = 0
        rospy.spin()

    def image_callback(self, msg):
      try:
            image = CvBridge().imgmsg_to_cv2(msg, "bgr8")
          if not os.path.exists('photo.jpeg'):
            cv2.imwrite('photo.jpeg', image)
          elif not os.path.exists('photo2.jpeg'):
            cv2.imwrite('photo2.jpeg', image)
          else:
            os.remove('photo.jpeg')
            os.rename('photo2.jpeg', 'photo.jpeg')
            cv2.imwrite('photo2.jpeg', image)
          if os.path.exists('../ui/data/photo1') and
os.path.exists('../ui/data/photo2') and self.myphoto == 0:
            self.myphoto = 1
            cv2.imwrite('../ui/data/media/myphoto.jpeg', image)
          if not os.path.exists('../ui/data/photo1') and not
os.path.exists('../ui/data/photo2'):
            self.myphoto = 0
          cv2.waitKey(50)
      except CvBridgeError, e:
          print(e)
```

Image_fetch()

voice.py

```
import time
import urllib
import json
import hashlib
import base64
import urllib.request
import urllib.parse

class voice:
    def main(self):
        f = open("test.wav", 'rb')
        file_content = f.read()
        base64_audio = base64.b64encode(file_content)
        body = urllib.parse.urlencode({'audio': base64_audio})

        url = 'http://api.xfyun.cn/v1/service/v1/iat'
        api_key = '7a3f2f91e4dd5cf3ba5b93402519403d'
        param = {"engine_type": "sms16k", "aue": "raw"}

        x_appid = '5b21d666'
        x_param = base64.b64encode(json.dumps(param).replace(' ',
'').encode('utf-8'))
        x_param = str(x_param, 'utf-8')

        x_time = int(int(round(time.time() * 1000)) / 1000)
        x_checksum = hashlib.md5((api_key + str(x_time) +
x_param).encode('utf-8')).hexdigest()
        x_header = {'X-Appid': x_appid,
                    'X-CurTime': x_time,
                    'X-Param': x_param,
                    'X-CheckSum': x_checksum}
        req = urllib.request.Request(url=url, data=body.encode('utf-8'),
headers=x_header, method='POST')
        result = urllib.request.urlopen(req)
        result = result.read().decode('utf-8')
        print(result)
        return json.loads(result)['data']


if __name__ == '__main__':
    a = voice()
    a.main()
```

my_lidar.py

```
import rospy
import roslib
import serial
#from get_signal import get_signal, detect
from sensor_msgs.msg import LaserScan
from playsound import playsound
import time

class GoForwardAvoid():
    def __init__(self):
        self.state_ = 0 #0 stop, 1 go
        self.safe_distance = 0.25
        rospy.loginfo("Scan")
        rospy.init_node('hello_world')
        self.scan_sub = rospy.Subscriber('scan', LaserScan,
self.laser_callback, queue_size=1)
        #rate = rospy.Rate(10)
        #rospy.spin()
        '''msg = rospy.wait_for_message('scan', LaserScan)
        ranges = list(msg.ranges)
        ranges = [x for x in ranges if x >= 0.2]
        d = min(ranges)
        print(d)
        if d>self.safe_distance:
            rospy.loginfo("Move forward")
            self.state_= 1
        elif d<=self.safe_distance:
            rospy.loginfo("Stop")
            self.state_= 0
            if int(time.time())%5:
                playsound('recorder/tooclose.mp3')'''

    def laser_callback(self, msg):
        rate = rospy.Rate(10)
        ranges = list(msg.ranges)
        ranges = [x for x in ranges if x >= 0.2]
        # ranges.remove(min(ranges))
        d = min(ranges)
        #print(d)
        if d>self.safe_distance:
            #rospy.loginfo("Move forward")
            self.state_= 1
        elif d<=self.safe_distance:
            #rospy.loginfo("Stop")
            self.state_= 0
            #if int(time.time())%5:
                #playsound('recorder/tooclose.mp3')
        #if self.state_ == 1:
            #rospy.signal_shutdown("shutdown")
```

record.py

```
import wave
from pyaudio import PyAudio,paInt16
class record:
    framerate=16000
    NUM_SAMPLES=2000
    channels=1
    sampwidth=2
    TIME=2
    def save_wave_file(self, filename,data):
        '''save the date to the wavfile'''
        wf=wave.open(filename,'wb')
        wf.setnchannels(self.channels)
        wf.setsampwidth(self.sampwidth)
        wf.setframerate(self.framerate)
        wf.writeframes(b"".join(data))
        wf.close()

    def my_record(self):
```

```python
        pa=PyAudio()
        stream=pa.open(format = paInt16,channels=1,
                       rate=self.framerate,input=True,
                       frames_per_buffer=self.NUM_SAMPLES)
        my_buf=[]
        count=0
        print(")
        while count<self.TIME*10:
            string_audio_data = stream.read(self.NUM_SAMPLES)
            my_buf.append(string_audio_data)
            count+=1
            #print('.')
        self.save_wave_file('test.wav',my_buf)
        print('')
        stream.close()

    chunk=2014
    def play(self):
        wf=wave.open(r"test.wav",'rb')
        p=PyAudio()

stream=p.open(format=p.get_format_from_width(wf.getsampwidth()),channels
=
        wf.getnchannels(),rate=wf.getframerate(),output=True)
        while True:
            data=wf.readframes(self.chunk)
            if data=="":break
            stream.write(data)
        stream.close()
        p.terminate()

if __name__ == '__main__':
    a = record()
    a.my_record()
    a.play()
    print('')
```