Provide a Global Tracking Feature for Person-Following Robot based on the Kinect Sensor

Zaid A. Mundher
Department of Computer Science, University of Mosul, Iraq
Email: zaidabdulelah@gmail.com

Jiaofei Zhong
Department of Mathematics and Computer Science, University of Central Missouri Warrensburg, USA
Email: zhong@ucmo.edu

***Abstract*—Person-tracking with mobile robots is a very active research area which has found success in areas such as museum guidance and hospital assistance. The ability to track only the target user and ignore other people in the robot's field of view is one of the main issues associated with the person tracking. This problem can be divided into two sub-problems: local tracking and global tracking. This paper introduces a solution to these problems by developing a body measurement identification system using the skeletal data from the Kinect sensor. The proposed solution to this problem is combined with a simple person-following method that enables the robot to follow the target person.**

***Index Terms*—global-tracking, kinect sensor, person-following, person-identification.**

I. Introduction

Person-tracking with a mobile robot is a very active research topic in the robotics field. Different approaches have been introduced to solve the person-tracking problem. One of the more common methods of tracking people is a vision-based approach which uses a camera as the main sensor for detecting and tracking people. According to [1], the tracking problem could be divided into two sub-problems: local tracking and global tracking. The tracking problem is considered local as long as the target person is in the robot's field of view. In contrast, the global tracking is initialized when the target person walks out and re-enters the robot's field of view. The aim behind the local and global tracking is to make the robot able to follow only the target person and ignore the other people in its field of view.

Recently, the Kinect sensor is being used to accomplish the person tracking task. Since the Kinect Windows SDK can recognize up to six users simultaneously, it is important to identify which person is being tracked and followed by the robot. Using the Kinect Windows SDK tracking engine, local tracking problem could be easily solved using the TrackingId property. The Kinect Windows SDK tracking engine assigns each detected user (skeleton) a unique integer identifier (TrackingId) which could be used to identify the tracked person [2]. Unfortunately, the TrackingId cannot be used to solve the global tracking problem because when the skeleton-tracking engine loses the ability to track the user, the TrackingId for that skeleton is retired. In other words, when the target leaves the scene and comes back, it will receive a new TrackingId.

To conquer the global tracking problem, a biometric person identification system based on human body measurements is introduced by this work. The proposed solution to this problem is combined with a simple person-following method that enables the robot to follow the target person. The major contribution of this work is to address the global person tracking problem based on the Kinect for Windows SDK to overcome the limitation of the Kinect's field of view.

II. Global Tracking Problem

The goal behind the global tracking is to re-identify a target when it leaves and re-enters the robot's field of view. The Kinect for Windows SDK has a reliable skeleton-tracking engine. However, if the target user is lost from sight during tracking, the skeleton-tracking engine cannot recover and correctly resume tracking that user [3]. The loss of tracking may easily happen since the Kinect has limited angles for its field of view, which are 57 º horizontally and 43 º vertically. Moreover, the robot may also lose tracking ability of the target when an obstacle appears between the target and the robot. As a part of obstacle avoidance, the robot needs to rotate far enough to avoid the obstacle. This means that the target will be out of the robot's field of view, leading to a loss of the target. When the robot once again faces the target, it will have to re-detect the target [4]. Re-detecting the same target will be a problem if there is more than one person in the field of view because the robot must continue following the first detected target and ignore the other. In general, in the global tracking, the person has to be identified after re-entering the robot's field of view.

III. Person Identification Systems

Manuscript received October 9, 2013; revised December 2, 2013.

doi: 10.12720/joace.2.3.322-326

The main idea behind the biometric systems is to measure one or more behavioral or physical characteristics such as voice, gait, or face. In general, any biometric system has two main phases: enrollment and matching. In the enrollment phase, the data is acquired from the individual and stored; in contrast, in the matching phase the data is re-acquired from the individual and compared against the stored data to determine the user's identity [5].

IV. Methodology

To address the global tracking problem for the person-following robot, a long distance person recognition system needs to be implemented. In long distance person recognition systems, it is not easy to recognize people with their exact biometric characteristics such as iris and face [6], [7]. Therefore, using soft biometric features that could be measured with low resolution images has great advantages over other biometric techniques. The proposed algorithm uses the skeleton joint points to define three semi-biometric features, as listed in Table I, to identify the target person.

TABLE I. Person Characteristics

Number	Characteristic Name
#1	Height
#2	Shoulder width
#3	Arm length

A. *Person Detection and Tracking*

The Kinect provides a skeleton-tracking feature that allows developers to recognize people and track their actions. Using depth sensors, the Kinect can recognize up to six users who are standing between 0.8 to 4.0 meters away (2.6 to 13.1 feet). Two of the detected users can be tracked in detail with 20 joint points as shown in Fig. 1.

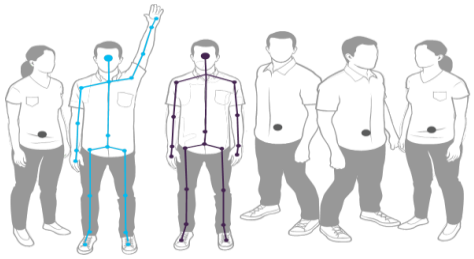

Figure 1. Kinect skeleton tracking [8]

Each skeleton joint is measured in a three dimensional plane (X, Y, Z). The X and Y coordinates indicate the joint location in the plane, while Z indicates how far the joint is from the Kinect sensor.

B. *Data Capturing and Processing*

The Kinect for Windows SDK provides a set of APIs to access to the skeletal data. Moreover, the Kinect for Windows SDK provides a SkeletonFrameReady event which fires each time new skeletal data becomes available for the Kinect sensor. Using this event, skeletal data can be retrieved from the SkeletonStream. Once the skeletal data is captured, the proposed system starts calculating the required characteristics (height, shoulder width, and arm length) based on the coordinates that are provided by the Kinect. First of all, the proposed system calculates the target's height. In order to measure the person's height, the sum of the lengths of each body part listed in Table II must first be calculated as illustrated in Algorithm 1 (where Length method implements the Euclidean distance formula as explained in Section C).

TABLE II. Required Body Parts to Calculte the Person Height

Number	Body Part
#1	Head → Spine
#2	Spine → HipCenter
#3	HipCenter → HipRight
#4	HipRight → KneeRight
#5	KneeRight → AnkleRight

Algorithm 1: GetHeight

```
declare double Head = skeleton.Joints[JointType.Head]
declare double Spine = skeleton.Joints[JointType.Spine]
declare double HipCenter =
        skeleton.Joints[JointType.HipCenter]
declare double HipRight =
        skeleton.Joints[JointType.HipRight]
declare double KneeRight =
        skeleton.Joints[JointType.KneeRight]
declare double AngleRight =
        skeleton.Joints[JointType.AngleRight]
declare double Head_Spine = Length(Head, Spine)
declare double Spine_HipCenter = Length(Spine, HipCenter)
declare double HipCenter_HipRight =
                Length(HipCenter, HipRight)
declare double HipRight_KneeRight =
                Length(HipRight, KneeRight)
declare double KneeRight_AngleRight =
                Length(KneeRight, AngleRight)
declare double OrigHeight = Head_Spine + Spine_HipCenter
 + HipCenter_HipRight + HipRight_KneeRight +
   KneeRight_AngleRight
```

Second, the proposed system calculates arm length. Arm length is calculated by summing the length between the joint points that are listed in Table III. The algorithm of measuring arm length is illustrated in Algorithm 2.

TABLE III. Required Body Parts to Calculate the Arm Length

Number	Body Part
#1	ShoulderRight → ElbowRight
#2	ElbowRight → WristRight

Algorithm 2: GetArmLength

```
declare double ShoulderRight =
        skeleton.Joints[JointType.ShoulderRight]
declare double ElbowRight =
        skeleton.Joints[JointType.ElbowRight]
declare double WristRight =
        skeleton.Joints[JointType.WristRight]
declare double ShoulderRight_ElbowRight =
        Length(ShoulderRight, ElbowRight)
declare double ElbowRight_WristRight =
        Length(ElbowRight, WristRight)
declare double OrigArmLength =
        ShoulderRight_ElbowRight +
        ElbowRight_WristRight
```

Finally, the proposed system calculates shoulders width. Shoulders width is calculated by measuring the distance between the right shoulder joint point and the left shoulder joint point as Algorithm 3 shows.

Algorithm 3: GetShoulderWidth

```
declare double ShoulderRight = s
        keleton.Joints[JointType.ShoulderRight]
declare double ShoulderLeft =
        skeleton.Joints[JointType.ShoulderLeft]
declare double OrigShoulderWidth = Length(ShoulderRight,
        ShoulderLeft)
```

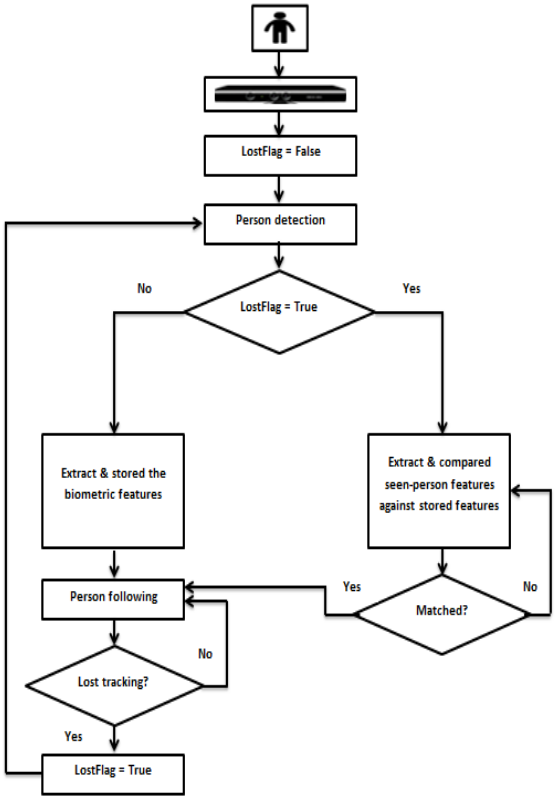

Figure 2. An overview of the proposed system

According to the proposed algorithm, the target person needs to stand 2m away facing the robot in order to be measured and tracked. Once the system calculates and stores the required biometric features, the robot starts following the target person based on the proposed person-following algorithm, as explained in Section V. Meanwhile, the algorithm keeps checking the number of detected people in the robot's field of view. When the number of detected people is zero, it means there is no person in the field of view, and the robot has lost the ability to track of the target. When the number of detected people becomes larger than zero, it means that there is one person (or more) in the field of view. The robot then measures the required biometric features for all detected persons and compares the result against the stored features values. Depending on the result, the robot will follow the closest matched person if the differences of the required biometric features with 3% of the stored data. These steps are explained in Fig. 2 and Algorithm 4.

C. *Measure the Distance Between Two Joints*

The distance between any two joint points can be measured using the formula of the Euclidean distance. According to the Euclidean distance formula, the distance between any two points in three dimensional planes with coordinate (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) is given in (1)

$$dist((X_1,Y_1,Z_1),(X_1,Y_1,Z_1)) = \sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2+(Z_2-Z_1)^2} \quad (1)$$

Algorithm 4: Target Re-identification

```
if(lostTracking = true)
   loop ( skeletons )
      HeightList[index] = GetHeight(skeleton)
      ShoulderWidthList[index] = GetShWidth(skeleton)
      ArmLengthList[index] = GetArmLength(skeleton)

      DiffHeightsList[index] = Abs(HeightList[index] –
                               OrigHeight)
      DiffShoulderWidthList[index] = Abs(
                  ShoulderWidthList [index] –
                  OrigShoulderWidthList)
      DiffArmLengthList[index] = Abs(
      ArmLengthList[index] – OrigArmLength)
      index++
   end loop
   loop(DiffHeightsList)
      diff.Add (DiffHeightsList[index] +
      DiffShoulderWidthList[index] +
      DiffArmLengthList[index])
       index++
    end loop
    declare MinimumDiff = Min(diff)
     if (MinimumDiff < thr)
       Start Following
```

V. Person Following

The proposed person following system is designed based on the distance-based control loop approach [9]. The proposed method calculates the distance between the robot and the target person every second. The robot attempts to remain at a distance that does not make the target person uncomfortable (which is 2m in this work). Therefore, the robot moves toward the target person when the distance is greater than 2m. Besides moving towards the target person, the robot should also be able to turn in order to follow the target person. Therefore when the target person steps to the right side or left side, the robot needs to change its direction based on the right and left motion parameters in order to keep the target person in the center of the robot's view. To calculate the right and left motion parameters, the algorithm keeps tracking the right and left shoulders in order to determine the direction of the target person. Based on the shoulders' values, the robot will determine if a turn is necessary. As explained in Algorithm 5, if the right shoulder is closer to the robot than the left shoulder, the robot executes a rotate left command. If the left shoulder is closer to the robot from the right shoulder, the robot executes a rotate right command. Otherwise, a forward command is executed.

Algorithm 5: Person Following Procedure

```
function PersonFollowing (distance , Threshold,
          rightShoulderPosition, leftShoulderPosition)
   if (distance > 2 meter)   then
      if ( abs (rightShoulder - leftShoulder) < Threshold)
          MoveForward
      elseif (leftShoulder > rightShoulder)
          TurnLeft
      else
          TurnRight
      end if

   else
        StopTheRobot
   end if

end function
```

VI. Implementation and Results

In the experiment, a LEGO Mindstorm robot is used. As shown in Fig. 3, the robot is equipped with a Kinect sensor.

Figure. 3. The robot

At the beginning, the target person needs to introduce himself/herself to the robot so the robot can calculate all the required biometric features. The robot starts following the target once the distance is greater than 2m. The target person was asked to move to the side fast enough to be lost by the robot. When the target re-enters the robot's field of view, the robot was able to re-detect and re-identified him. As a second step, another person was asked to enter the robot's field of view with the target person after lose tracking. The robot was successfully able to re-identify and keep following the target person, while ignoring the second person. The experiments have shown that the best threshold value is 0.03 where the false rejection rate is zero.

The experiment and all the proposed algorithm's steps are illustrated in Table IV.

TABLE IV: Experiment Result

Feature	Enrolled	Lived		Differences btw stored and lived features	
	P 0	P 1	P 2	P 1	P 2
Height	1.56	1.56	1.60	0	0.04
Shoulder	0.40	0.41	0.43	0.01	0.03
Arm	0.54	0.53	0.58	0.01	0.04
Diff.				0.02	0.11
Min.				0.02 < thr	
Result				followed	ignored

VII. Conclusion

One of the main issues associated with the person tracking is guaranteeing the robot is able to follow only the target person and ignore all the other people in the field of its view. Although the Kinect Windows SDK provides a reliable human tracking solution, it has some limitations. This paper focuses on the development of a person tracking system that supports global tracking based on the Kinect Windows SDK to develop a body measurement identification system. Although utilizing semi-biometric features, such as body measurement, is not a 100% accurate way to identify people, it is still a powerful tool especially in long-distance identification.

Acknowledgment

This work was financially supported in part by the Higher Committee for Education Development in Iraq (HCED).

References

[1] W. Zajdel, Z. Zivkovic, and B. J. A. Krose, "Keeping track of humans: Have I seen this person before?" in *Proc. IEEE International Conference on Robotics and Automation*, 2005, pp. 2081–2086.

[2] A. Jana, *Kinect for Windows SDK Programming Guide*, 1st ed. UK: Packt Publishing, ch. 6, 2012, pp. 183

[3] B. J. Southwell and G. Fang, "Human object recognition using colour and depth information from an RGB-D kinect sensor," *International Journal of Advanced Robotic Systems*, vol. 10, 2013.

[4] W. R. Kulp, "Robotic person-following in cluttered environments," M. S. thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, 2012.

[5] A. K. Jain, A. A. Ross, and K. Nandakumar, *Introduction to Biometrics*, Michigan: Springer, 2011, ch. 1, pp. 4.

[6] A. Sinha, K. Chakravarty, and B. Bhowmick "Person identification using skeleton information from kinect," presented at ACHI 2013: The Sixth International Conference on Advances in Computer- Human Interactions, Nice, France, February 24, 2013.

[7] K. Kim, M. Siddiqui, A. Fran ̧ois, G. Medioni, and Y. Cho, "Robust real-time vision modules for a personal service robot," presented at the 3rd International Conference on Ubiquitous Robots and Ambient Intelligence, 2006

[8] Microsoft. (2012). Skeletal tracking. [Online] Available: http://msdn.microsoft.com/en-us/library/hh973074.aspx.

[9] E. A. Topp and H. I. Christensen, "Tracking for following and passing persons," presented at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Alberta, Canada, Aug 2-6, 2005.

Zaid A. Mundher is currently a master's student in the Department of Mathematics and Computer Science, University of Central Missouri. He received his B.S. in Computer Science from the University of Mosul, Iraq in 2007. He was a Teaching Assistant at the Department of Computer Science, University of Mosul, between 2008 and 2011.

Jiaofei Zhong is currently an Assistant Professor of Computer Science at University of Central Missouri. She received her PhD in Computer Science in 2012 and M.S. degree in 2010, both from the University of Texas at Dallas. Dr. Zhong has served as a peer reviewer for a number of international conferences and journals, and has been the Publicity Chair, Financial Chair, and OCS co-Chair in the organizing committees of several international conferences. Her research interests are in the areas of data engineering and information management, especially in Wireless Communication Environment, including Data Broadcasting, Vehicle Ad hoc Networks, and Sensor Database.