# A Real-Time Fall Detection System in Elderly Care Using Mobile Robot and Kinect Sensor

Zaid A. Mundher and Jiaofei Zhong

***Abstract*—The growing population of elderly people especially in developed countries motivates the researchers to develop healthcare systems to ensure the safety of elderly people at home. On the other hand, mobile robots can provide an efficient solution to healthcare problem. Moreover, using new technologies such as the Kinect sensor with robotics could bring new ways to build intelligent systems that could use to monitor the elderly people, and raise an alarm in case of dangerous events, such as falling down, are detected. Falls and their consequences are a major risk especially for elderly people who live alone where immediate assistance is needed. In this work, the Kinect sensor is used to introduce a mobile robot system to follow a person and detect when the target person has fallen. In addition, the mobile robot is provided with a cell phone that is used to send an SMS message notification and make an emergency call when a fall is detected.**

***Index Terms*—Fall detection, healthcare, Kinect sensor, person-following.**

## I. Introduction

Falls and their consequences, such as bone fractures, are a major risk especially for elderly people who live alone where immediate assistance is needed. According to [1], falls are the sixth most common cause of death for people over the age of 65. Moreover, it is the second most common cause of death for people between the age of 65 and 75, and the most common cause for people over 75. Therefore, automatically detecting falls at home has become a major interest in research.

Fall detection systems are used to monitor the place where elderly people live and send a notification to an emergency center or caregivers once a fall is detected. Many approaches and algorithms have been developed to detect falls. According to [1], fall detection approaches are divided into three main categories: vision-based, environmental (ambient devices), and wearable. Since vision-based systems are able to overcome the limitations of other sensor types [2], a vision-based approach which utilizes the Kinect depth camera is used in this work. The main goal of this work is to accomplish a reliable fall detection system using a low-cost mobile robot platform.

Manuscript received November 21, 2013; revised February 12, 2014. This work was financially supported by the Higher Committee for Education Development in Iraq (HCED).

Z. A. Mundher is with the Department of Computer Science, University of Mosul, Mosul, Iraq (e-mail: zaidabdulelah@gmail.com).

J. Zhong is with the Department of Mathematics and Computer Science, University of Central Missouri Warrensburg, MO 64093, USA (e-mail: zhong@ucmo.edu).

## II. Literature Review

Most of the previous works related to vision-based method used fixed regular RGB cameras as the main sensor [3], [4]. However, in recent years, researchers started using depth cameras, such as the Kinect sensor, to detect falls. Compared to classical methods, which use regular RGB cameras, the depth cameras improve performance with lower costs and more functionality. Moreover, changing lighting conditions does not affect the result of the depth camera. The use of the Kinect sensor as a fixed camera has been proposed in the literature [5]-[7]. However, the use of fixed Kinect sensors can be unreliable because of occlusions by furniture items and limitations of the Kinect viewpoint. In addition, the field of view for the Kinect is limited, so the Kinect sensor cannot cover the entire surveillance area. Therefore, in order to overcome the previous drawbacks, a mobile robot is used in this work. The proposed fall detection system is combined with a proposed person following system, so the robot is able to track and follow elderly people in order to monitor their activities. As a result, the coverage limitation of using a fixed Kinect sensor to detect falls is solved in this work.

## III. Human-Robot Interaction

Gesture recognition and speech recognition are the most important research areas in the field of the Human-Robot-Interaction. Since the proposed system is designed to work with elderly people who prefer a simple and natural way to interact with robots, it was important to create a natural user interface to interact with the robot. Therefore, in this work, the Kinect sensor is used to develop and implement a gesture recognition system and a speech recognition system to provide a simple and natural interaction method through which elderly people can send commands to the robot without the need for physical contact with the robot.

## IV. The Proposed System

The main components of the proposed system are shown in Fig. 1.

### *A. Hardware Layer*

#### *1) The Kinect sensor*

The Microsoft Kinect, which was released in November 2010, is a set of sensors that was originally developed as an input device for the Xbox 360 gaming console (Fig. 2). The Kinect can track the human body movements, so the player

DOI: 10.7763/IJMMM.2014.V2.115

can interact with the game without using a controller; the body is the controller.

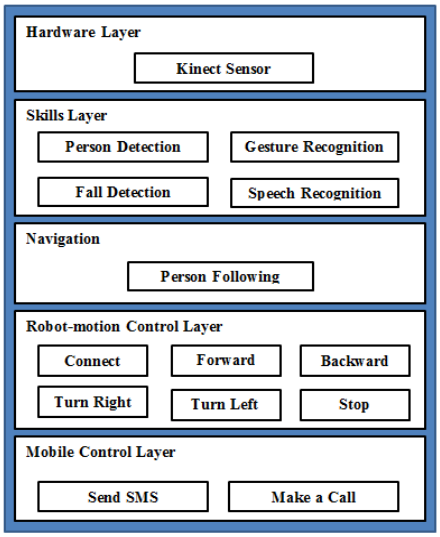



Fig. 1. Main components of the proposed system architecture.

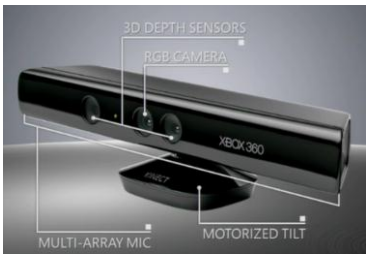



Fig. 2. The Kinect device.

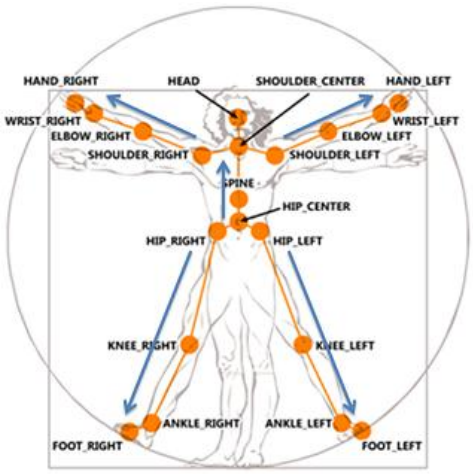



Fig. 3. Skeleton joint points.

In June 2011, Microsoft released a software development kit (SDK) for the Kinect. The Kinect for Windows SDK is a toolkit that has a set of libraries to develop different applications for Kinect devices. Besides providing depth images, another feature of the Kinect sensor is a skeletal tracker. The Kinect for Windows SDK 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 twenty joint positions. Each skeleton joint is measured in a three dimensional (X, Y, Z) plane. The X axis of the joint will change when the joints move from the right side to the left side or vice versa, while the Y axis will change when the joints move in the upwards or downwards direction. Similarly, Z axis will change when the joints move forwards or backwards in relation to the Kinect sensor. The twenty joint points are shown in Fig. 3.

### *B. Skills Layer*

#### *1) Person detection*

To detect and track a person, the skeleton stream from the Kinect is used. An event is registered to listen and track the skeleton frame. Once a person is detected, the proposed algorithm starts processing the skeletal information.

#### *2) Fall detection*

The proposed method to detect a fall is based on the skeleton joint positions relative to the ground. Since the proposed fall detection algorithm calculates the distance between the body joint points and the floor-plane, the floor-plane must be detected first. The Kinect for Windows SDK provides a floor-clipping-plane vector, which contains the coefficients of an estimated floor-plane equation. To calculate the distance between a point and the plane, when the point is $(x_0, y_0, z_0)$ and the equation of the plane is $(ax + by + cz + d)$, Eq. (1) is used, where $D$ is the distance between the point and the plane.

$$D = (ax_0 + by_0 + cz_0 + d) / \sqrt{a^2 + b^2 + c^2} \quad (1)$$

Using this relation, the distance from the floor to each body joint point can be calculated. A fall is detected by thresholding the distance between the joint points that are shown in Fig. 4 and Table I and the ground.

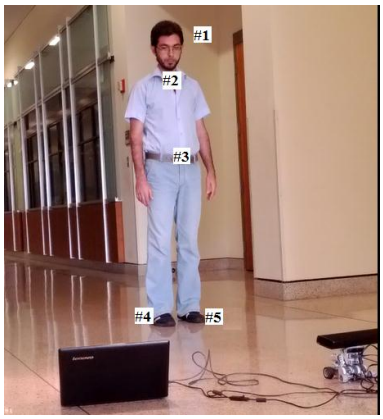



Fig. 4. Fall detection joint points.

TABLE I: FALL DETECTION JOINT POINTS

| Number | Joint Point |
|---|---|
| #1 | Head |
| #2 | ShoulderCenter |
| #3 | HipCenter |
| #4 | AnkleRight |
| #5 | AnkleLeft |

The procedure of the fall detection system based on the floor-plane is shown in Algorithm 1.

When the floor is not visible or detectable, the proposed algorithm depends on the skeleton space coordinate system to detect falls. As shown in Fig. 5, the origin of the skeleton space coordinate is placed at the Kinect sensor, and it is a

right-handed coordinate system, which means y-axis extends upwards. According to the proposed method that is shown in Algorithm 2, if the Y coordinate of the points that are listed in Table I is less than a given threshold, a fall is detected.

**Algorithm 1**: Fall Detection – Method1

```
function PersonFallingDown1( JointPoint )
  if  floor_plane is detected
      declare float A ← FllorClipPlane.Item1
      declare float B ← FllorClipPlane.Item2
      declare float C ← FllorClipPlane.Item3
      declare float D ← FllorClipPlane.Item4
      declare float x ← A * JointPoint.X
      declare float y ← B* JointPoint.Y
      declare float z ← C * JointPoint.Z
      declare float distance ← (x+y+z+D)/
                               sqrt((A*A)+(B*B)+(C*C))
      if ( distance <= 0.3)
        return "Fall detected"
      else return "Safe"
    end if
  end function
```

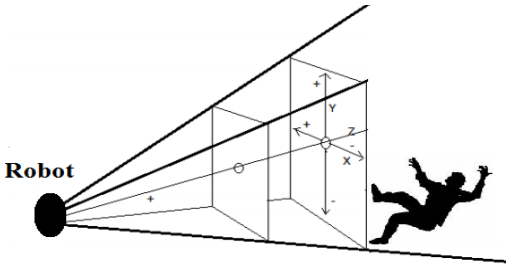



Fig. 5. Skeleton space coordinate system.

**Algorithm 2**: Fall Detection – Method2

```
function PersonFallingDown( skeleton)
   declare double yHead, yShoulderCenter, ySpine, yHipCenter,
                  yAnkleLeft, yAnkleRight, Threshold
   declare  double Head_Ankle1 =0, Head_Ankle2 =0, thr
   declare Boolean flag = true
   yHead ← Y coordinate of the skeleton.Head point
   yShoulderCenter ← Y coordinate of the skeleton.ShoulderCenter
point
   yHipCenter ← Y coordinate of the skeleton.HipCenter point
   yAnkleLeft ← Y coordinate of the skeleton.AnkleLeft point
   yAnkleRight ← Y coordinate of the skeleton.AnkleRight point

   if Head_Ankle1 ==0
      Head_Ankle1 = yHead – yAnkleRight
      thr = Head_Ankle1 * 0.7
   else
      Head_Ankle2 = yHead-yAnkleRight
      if (abs(Head_Ankle1- Head_Ankle2) > thr ) and
      if (yHead < Threshold) and
      if ((yShoulderCenter < Threshold) and

      if (yHipCenter < Threshold)   and
      if (yAnkleLeft < Threshold)       and
      if (yAnkleRight < Threshold))     then
        return "Fall is detected "

       else return "Safe"
   end if
 end function
```

*3) Gesture recognition*

The proposed gesture-recognition engine is developed based on comparing the joints' positions and the deviation between the joints' positions. In this work, two types of gesture movements are recognized, as shown in Table II.

TABLE II: GESTURE PATTERNS

| Action | Gesture Pattern |
|---|---|
| Start following | Right-hand is above the right shoulder and the left hand is below the left hip |
| Stop following | Left-hand is above the left shoulder and the right-hand is below the right hip |

The *StartFollowing* gesture is identified when the right hand is raised above the right shoulder and the left hand is below the left hip, as shown in Fig. 6. To recognize this gesture type, the right shoulder joint position is defined as the reference point, while the right hand joint position is defined as the target point.

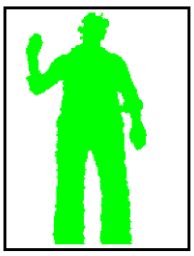


Fig. 6. The StartFollowing gesture.

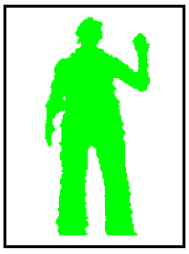


Fig. 7. The StopFollowing gesture.

In contrast, the *StopFollowing* gesture is identified when the left hand is raised above the left shoulder while the right hand is below the right hip as shown in Fig. 7. To recognize this gesture type, the left shoulder joint position is defined as the reference point, while the left hand joint position is defined as the target point.

The algorithm of the gesture recognition is shown in Algorithm 3.

**Algorithm 3**: Gesture Recognition Procedure

```
function (skeleton, threshold)

   if (Abs(skeleton.HandRight.X – skeleton.ElbowRight.X)) <
threshold) and (Abs(skeleton.HandRight.Z -
skeleton.ElbowRight.Z) < threshold) and
(Abs(skeleton.ElbowRight.Z - skeleton.ShoulderRight.Z) <
threshold)      and (Abs(skeleton.HandRight.Y -
skeleton.ShoulderRight.Y) < threshold)    and
(Abs(skeleton.HandRight.Z - skeleton.ShoulderRight.Z) <
threshold)    and  (skeleton.HandLeft.Y < skeleton.HipLeft.Y))
then

        cmd ← "Start"

  if (Abs(skeleton.HandLeft.X – skeleton.ElbowLeft.X)) <
threshold) and   (Abs(skeleton.HandLeft.Z -
skeleton.ElbowLeft.Z) < threshold) and
(Abs(skeleton.ElbowLeft.Z - skeleton.ShoulderLeft.Z) <
threshold) and   (Abs(skeleton.HandLeft.Y -
skeleton.ShoulderLeft.Y) < threshold) and
(Abs(skeleton.HandLeft.Z - skeleton.ShoulderLeft.Z) <
threshold) and   (skeleton.HandRight.Y < skeleton.HipRight.Y))
then

        cmd ← "Stop"

 return cmd

end function
```

#### 4) Speech recognition

In this part, the audio stream of the Kinect for Windows SDK is used to capture the Kinect audio data. Moreover, Microsoft Speech Library is used to build the speech recognition engine. The introduced speech recognition engine is able to recognize three commands that are listed in Table III.

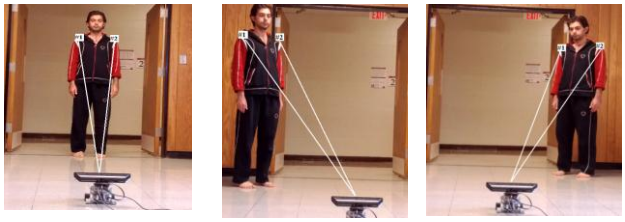
TABLE III: VOICE COMMANDS

| Command | Description |
|---|---|
| Run | To start the fall monitoring system |
| Stop | To stop the fall monitoring system |
| Call | To make a call |

### C. Navigation Layer

#### 1) Person following

The distance-based control loop approach [8] is used to develop the proposed person-following subsystem. According to the proposed algorithm, the robot moves toward the target person when the distance is greater than 2m. Besides moving towards the target person, the robot needs to change its direction when the target person steps to the right side or left side. Therefore, the robot calculates the right and left motion parameters in order to keep the target person in the center of the robot's view. The proposed 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 shown in Fig. 8(A), if the right and left shoulders are at the same distance from the robot (given a threshold), the forward command is executed. In addition, if the right shoulder is closer to the robot than the left shoulder, as shown in Fig. 8(B), the robot executes a rotate left command. If the left shoulder is closer to the robot from the right shoulder, as shown in Fig. 8(C), the robot executes a rotate right command.



(A): Forward (B): Turn Left (C): Turn Right

Fig. 8. Person-following commands.

The algorithm of the introduced person-following method is given is Algorithm 4.

**Algorithm 4**: Person Following Procedure

```
function PersonFollowing (distance , Threshold,
                          rightShoulderPosition,
                          leftShoulderPosition)

  if (distance > 2m)   then
    if ( abs (rightShoulder - leftShoulder) < Threshold)  then
          MoveForward
    elseif (leftShoulder > rightShoulder)   then
          TurnLeft
    else
          TurnRight
    end if
  else
     StopTheRobot
  end if

end function
```

### D. Robot-Motion Control Layer

To interact with the robot's motors, the RobotControl class was built based on the MindSqualls library, which is free and it can be downloaded from http://www.mindsqualls.net. The RobotControl class provides six functions to control the robot as shown in Fig. 9 and Table IV.

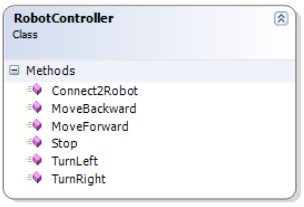



Fig. 9. The RobotControl class.

TABLE IV: THE ROBOTCONTROL CLASS METHODS

| Method | Purpose |
|---|---|
| Connect2Robot | Initialize the connection to the robot |
| MoveForward | Drive the robot forward |
| MoveBackward | Drive the robot backward |
| TurnRight | Turn the robot to the right |
| TurnLeft | Turn the robot to the left |
| Stop | Stop the robot |

### E. Mobile Control Layer

To implement the fall detection alarm system, the robot is provided with a mobile phone device to send an SMS message notification and to make an emergency call when the robot detects a fall. To reduce false positive alarms, after a fall is detected, the system waits 5 seconds to confirm that the fall is followed by an inactivity period, which means that the fallen person needs help with getting up again. Once the five seconds end and the user is still lying on the ground, an SMS message notification is sent to a caregiver phone number. Moreover, the robot, based on the skeletal data, will move toward the head of the fallen person to be close enough to enable the fallen person to use the voice command to make a call. AT commands are used to control the mobile phone device. More specifically, AT+CMGS command is used to send the SMS, while ATD command is used to make a call.

## V. IMPLEMENTATION

The LEGO Mindstorm robot and the Kinect sensor are used to implement the proposed system. To implement the Kinect processing, the Kinect Windows SDK with C# is adopted. Moreover, a computer laptop is used as the processor (robot computer), and the Nokia 6200 mobile phone is used to raise an alarm (send an SMS or make a call).

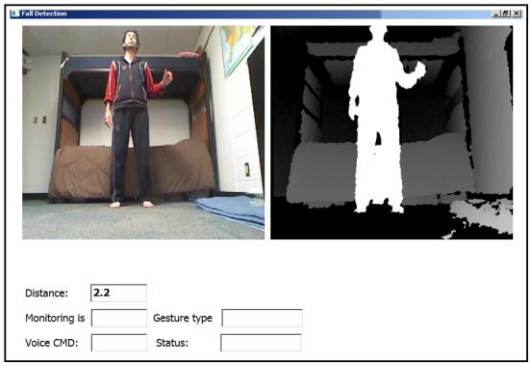


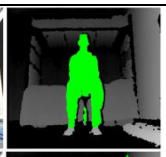
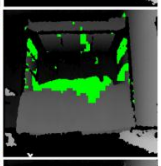
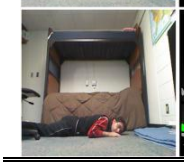
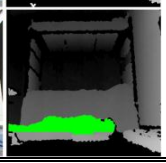

Fig. 10. Main user interface.

Windows Presentation Foundation (WPF) is used to design the main user interface as shown in Fig. 10.

## VI. Experiments and Results

Experiments were conducted in a real indoor environment under different light conditions. First, the system was successfully tested to verify that there is no false alarm will be raised while the target person is sitting on a chair or lying on a bed, in contrast, a fall is detected once the target person is lying on the ground, as shown in Table V.

TABLE V: Different Situation of the Target Person

| Target Statue | System Status |
|---|---|
|   | Safe |
|   | Safe |
|   | Fall is detected |

Second, the fall detection system was tested on different fall scenarios (fall to the right side, fall to the left side, fall to the front, and fall to the back). Fall-detection scenarios are shown in Fig. 11, and the performance of the fall-detection system is shown in Table VI and Fig. 12.

Fig. 11. Different falls scenarios.

TABLE VI: Fall Detection Results

| Distance | No. of simulated falls scenarios | No. of detected falls scenarios | No. of undetected falls scenarios | Accuracy |
|---|---|---|---|---|
| 2.0m | 4 | 2 | 2 | 50% |
| 2.5m | 4 | 3 | 1 | 75% |
| 3.0m | 4 | 4 | 0 | 100% |
| 3.5m | 4 | 4 | 0 | 100% |

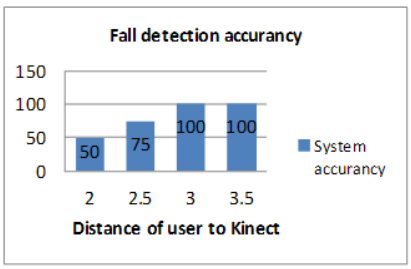



Fig. 12. Fall-detection accuracy.

In addition, the results have shown that the proposed gesture recognition system works perfectly when the distance is between 2 and 3.5m as shown in Fig. 13.

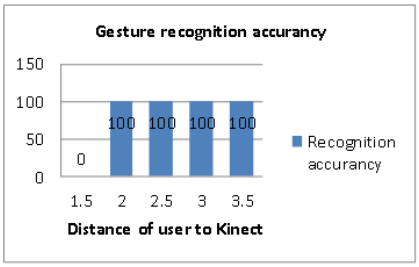



Fig. 13. Gesture recognition accuracy.

Moreover, the results have also shown that the speech recognition system works well. On the other hand, the person-following results have shown that the robot is able to move directly toward the target when the distance between the robot and the target person is larger than 2m. Meanwhile, the robot adjusts itself to keep the target in the middle of the robot's field of view. The snapshots of the indoor person-following experiment are shown in Fig. 14.

Fig. 14. Sequence of frames of following the target.

## VII. Conclusion

Today, the Kinect device is one of the major input devices that are used in the robotics field. The use of the Kinect device in conjunction with robotics provides several opportunities and promising possibilities. In this work, the Kinect device is used to accomplish a fall detected system with mobile robot. To achieve real-time detection and tracking of a user, the skeletal tracking feature of the Kinect for Windows SDK is used. The results revealed the use of the Kinect device to implement the proposed fall-detection system is efficient with a low computational complexity.

## References

[1] S. Abbate, M. Avvenuti, P. Corsini, A. Vecchio, and J. Light, "Monitoring of human movements for fall detection and activities recognition in elderly care using wireless sensor network: a survey," in *Wireless sensor networks: application-centric design*, Y. K. Tan, Ed. China: InTech, 2010, pp. 1-20.

[2] A. Mihailidis, B. Carmichael, and J. Boger, "The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, issue 3, pp. 238–247, Sept. 2004.

[3] Y. Huang, S. Miaou, and T. Liao, "A human fall detection system using an omni-directional camera in practical environments for health care applications," presented at the MVA2009 IAPR Conference on Machine Vision Applications, Yokohama, Japan, May 20-22, 2009.

[4] V. Rudakova, "Probabilistic framework for multi-target tracking using multi-camera: applied to fall detection," M.S. thesis, Color in Informatics and Media Technology, Gjovik university college, Norway, 2010.

[5] C. Kawatsu, J. Li, and C. Chung, "Development of a fall detection system with Microsoft Kinect," in *Robot Intelligence Technology and Applications 2012*, vol. 208, J. Kim, E. T. Matson, H. Myung, and P. Xu, Eds. Springer Berlin Heidelberg, 2012, pp. 623-630.
[6] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier, "Fall detection from depth map video sequences," in *Toward Useful Services for Elderly and People with Disabilities*, vol. 6719, B. Abdulrazak *et al*. Eds. Montreal, Canada: Springer Berlin Heidelberg, 2011, pp. 121-128.
[7] R. Planinc and M. Kampel, "Detecting falls by using depth cameras," presented at the 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012.
[8] 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.