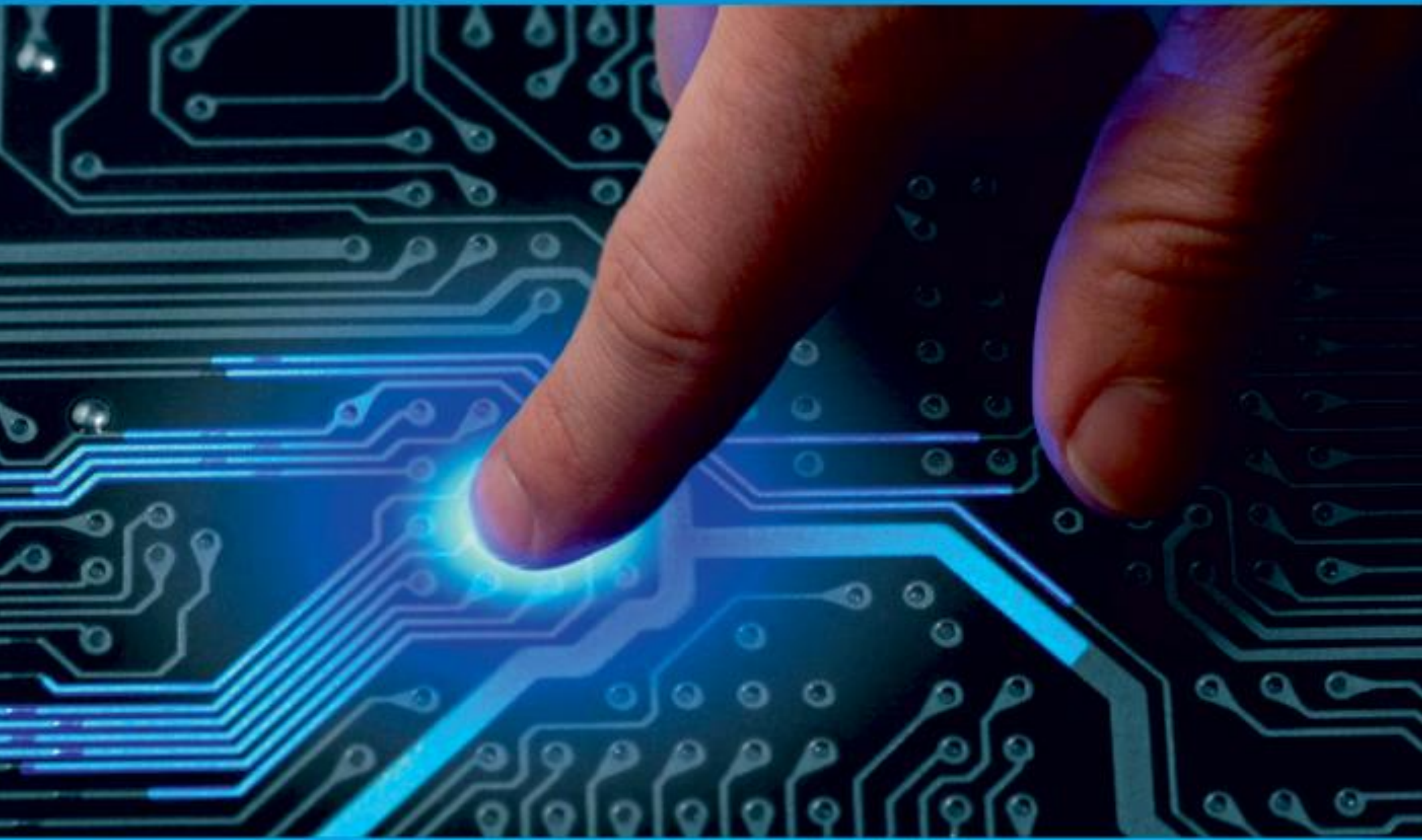
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 12, Issue 5, May 2024

INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 8.379

9940 572 462 6381 907 438 ijircce@gmail.com www.ijircce.com

Gesture-Enhanced Sterile Image Viewer through Deep Learning

Yuheshwar K S, Suriya Khailesh R V, Varun Prasadd S D and V Vidhya

UG Student, Department of IT, Sri Venkateswara College of Engineering, Pennalur, Sriperumbudur, India

UG Student, Department of IT, Sri Venkateswara College of Engineering, Pennalur, Sriperumbudur, India

UG Student, Department of IT, Sri Venkateswara College of Engineering, Pennalur, Sriperumbudur, India

HOD & Professor, Department of IT Sri Venkateswara College of Engineering, Pennalur, Sriperumbudur, India

ABSTRACT: In medical settings, minimizing the risk of contamination is paramount to ensuring patient safety and reducing the spread of infections. Traditional methods of accessing and manipulating digital information, such as touching screens or using keyboards, pose a significant risk of cross-contamination. To address this challenge, we propose a novel Gesture-Controlled Image Transformation System (GCITS) designed specifically for medical environments. GCITS utilizes advanced gesture recognition technology to enable healthcare professionals, particularly doctors and surgeons, to manipulate images and other viewing-type information in operation theaters and clinical cabins without physical contact. By employing a combination of computer vision algorithms and machine learning techniques, the system accurately interprets hand gestures to perform a variety of functions, including zooming, rotating, panning, and switching between images.The system's interface is intuitive and user-friendly, allowing medical practitioners to seamlessly navigate through medical images, X-rays, scans, and other visual data during surgical procedures or clinical consultations.

KEYWORDS: You Only Look Once (YOLO), Convolutional Neural Network (CNN), Grid Celling, Image Upscaling

I. INTRODUCTION

In the ever-evolving landscape of healthcare, where precision and hygiene are paramount, the integration of cutting-edge technologies becomes imperative. Addressing the critical need for minimizing contamination risks in medical environments, this project introduces a groundbreaking approach leveraging gesture-controlled image transformation. This project harnesses the power of the You Only Look Once (YOLO) algorithm, a state-of-the-art object detection system, to revolutionize the way medical professionals interact with digital information within operating theaters and clinical settings. By employing intuitive hand gestures, doctors and surgeons can seamlessly manipulate and navigate through vital medical images and viewing information without physical contact, ensuring a contaminant-free environment.

Traditional methods of interacting with digital displays often involve direct physical contact, posing a significant risk of contamination, especially in sterile surgical environments. This innovative approach eliminates the need for physical touch altogether, thereby mitigating potential sources of infection and enhancing overall safety protocols.

By integrating gesture recognition technology with the YOLO algorithm, the system enables practitioners to perform a myriad of tasks with simple hand movements, including zooming, rotating, panning, and selecting specific regions of interest within medical images or other viewing information. This intuitive control mechanism not only enhances workflow efficiency but also minimizes distractions during critical procedures. Furthermore, the solution is designed to be highly adaptable, seamlessly integrating with existing medical imaging systems and display interfaces commonly found in operating theaters and clinical environments. This ensures compatibility and ease of implementation without requiring extensive infrastructure modifications. this project represents a pioneering advancement in healthcare technology, offering a transformative solution to mitigate contamination risks and enhance safety standards in medical environments. By empowering doctors and surgeons with gesture-controlled image transformation capabilities, it aims to revolutionize the way medical professionals interact with digital information, ultimately contributing to improved patient outcomes and elevated standards of care.

II. RELATED WORK

In [1] the authors present a lightweight model based on YOLOv3 and DarkNet-53 for real-time hand gesture recognition, achieving high accuracy even in complex environments without additional preprocessing. The model was tested on both static images and dynamic gestures, demonstrating its robustness and effectiveness.

In [3] the authors provide an extensive review of hand gesture recognition methods, focusing on image processing techniques and their applications in human-computer interaction. They categorize different approaches and analyze their performance and applicability in various contexts.

In [5] the authors present a CNN-based approach for static hand gesture recognition, highlighting data augmentation techniques to improve model accuracy. The study showcases the model's high performance in recognizing various hand gestures, making it suitable for real-time applications.

In [6] the authors describe a real-time hand gesture recognition system using the YOLO object detection algorithm. They focus on optimizing the model for speed and accuracy, making it suitable for applications requiring immediate feedback, such as interactive displays and control systems.

In [8] the authors focus on recognizing dynamic hand gestures using CNNs. They develop a system that captures and processes video sequences of gestures, demonstrating its effectiveness in various applications, including sign language recognition and human-computer interaction.

In [10] the authors present a robust hand gesture recognition system using the Kinect sensor. They focus on enhancing the accuracy and reliability of gesture recognition in noisy environments, making it suitable for real-world applications like interactive displays and gaming.

In [12] the authors propose a gesture recognition system based on fuzzy logic. They develop a fuzzy inference system to handle the variability and ambiguity in hand gestures, achieving high accuracy in recognition tasks and demonstrating its potential in human-computer interaction.

In [14] the authors present a machine learning-based approach to hand gesture recognition, focusing on feature extraction and classification techniques. They compare different machine learning algorithms and highlight their performance in recognizing a variety of hand gestures.

In [15] the authors discuss the application of hand gesture recognition in human-computer interaction, using various recognition techniques to enable intuitive control of computer systems. They demonstrate the system's effectiveness in enhancing user experience and interaction efficiency.

III. PROPOSED ALGORITHM

The algorithm employs the YOLO (You Only Look Once) framework to recognize hand gestures demonstrated by medical professionals within digital images or video streams, subsequently performing corresponding transformations on the displayed content. The algorithm outline is explained below:

Image Capturing: The images are captured using a webcam module connected to the device, which allows for real-time monitoring. This setup captures images at specified time intervals from the live feed, providing a continuous stream of visual data. The time interval between each captured image can be adjusted according to the specific requirements of the task at hand, offering flexibility in how frequently the images are recorded. This adjustable timing feature ensures that the system can be tailored to various monitoring needs, whether for detailed, high-frequency image capture or less frequent snapshots.

Feature Extraction: To detect objects, the input image is processed through a deep CNN that extracts rich feature representations. These features are crucial for accurately detecting and classifying objects. YOLOv3 uses a backbone network like DarkNet-53 for feature learning. YOLOv3 employs a modified version of the DarkNet architecture, specifically DarkNet-53, which has 53 convolutional layers. This network serves as the backbone, extracting hierarchical features at different levels of abstraction from the input image.

Grid Celling: By utilizing grid celling in the YOLO model, the system efficiently scans the image, localizes objects, predicts bounding boxes, and classifies objects with speed and reliability. For simpler gesture patterns where gestures are small and uniformly distributed, a regular grid celling approach is suitable. This method divides the input image into cells, simplifying the architecture and ensuring uniform predictions.

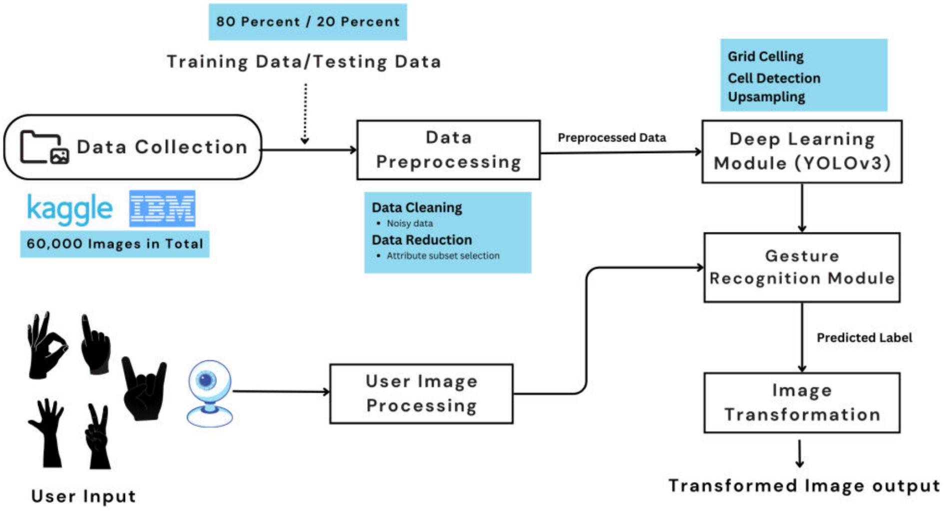
Class Prediction: For each predicted bounding box, the model outputs class probabilities. This involves determining the likelihood of the object in the bounding box belonging to each of the predefined classes.
YOLOv3 uses softmax activation to predict class probabilities for each box.

Image Upscaling: Upscaling images in the YOLOv3 algorithm using techniques like resizing, maintaining aspect ratio, bilinear interpolation, padding, and normalization, the input images are prepared in a consistent and optimized format for effective object detection and recognition tasks. This preprocessing step plays a crucial role in ensuring the accuracy and efficiency of the YOLOv3 model in detecting objects, including hand gestures, with high precision

Gesture recognition Module: The gesture recognition module in the context of the YOLOv3 model for hand gesture recognition plays a crucial role in identifying and classifying hand gestures in real-time.Using the extracted features, the module classifies the hand gesture into predefined categories or classes. This classification is based on comparing the features to patterns learned during the training phase of the neural network.

User Image Processing: The module acquires images from the user, either through a camera feed or from stored image files. These images serve as input data for the hand gesture recognition system. The user image processing module may enhance the quality of the input images to improve the performance of the gesture recognition system. This can involve adjusting brightness, contrast, or sharpness to make gestures more distinguishable.

Image Transformation Module: The image transformation module facilitates the adjustment of image orientation and aspect ratio in accordance with the predicted gesture. These transformations are mapped to their respective gestures, rendering the system easily customizable.



IV. PSEUDO CODE

```
Step 1: Importing necessary packages
import operator
import cv2 # opencv library
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model#to load our trained model
import os
from werkzeug.utils import secure_filename
Step 2: Loading YOLO model:
```

```
app = Flask(__name__,template_folder="templates") # initializing a flask app
model=load_model('gesture.h5')
print("Loaded model from disk")
```

Step 3: Navigating to user interface through routing:

```
@app.route('/')# route to display the home page
def home():
    return render_template('home.html')#rendering the home page
@app.route('/intro') # routes to the intro page
def intro():
    return render_template('intro.html')#rendering the intro page
@app.route('/image1',methods=['GET','POST'])# routes to the index html
def image1():
    return render_template("launch.html")
```

Step 4: Prediction on user input Gesture:

```
@app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
def launch():
    if request.method == 'POST':
        print("inside image")
        f = request.files['image']
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        print(file_path)
        cap = cv2.VideoCapture(0)
        while True:
            _, frame = cap.read()
            frame = cv2.flip(frame, 1)
            x1 = int(0.5*frame.shape[1])
            y1 = 10
            x2 = frame.shape[1]-10
            y2 = int(0.5*frame.shape[1])
            cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
            roi = frame[y1:y2, x1:x2]
            roi = cv2.resize(roi, (64, 64))
            roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
            cv2.imshow("test", test_image)
```

Step 5: Classification of predicted result:

```
result = model.predict(test_image.reshape(1, 64, 64, 1))
              prediction = {'ZERO': result[0][0],
                          'ONE': result[0][1],
                          'TWO': result[0][2],
                          'THREE': result[0][3],
                          'FOUR': result[0][4],
                          'FIVE': result[0][5]}
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
cv2.imshow("Frame", frame)
```

Step 6: Transformation on required image:

```
image1=cv2.imread(file_path)
              if prediction[0][0]=='ONE':
                resized = cv2.resize(image1, (200, 200))
                cv2.imshow("Fixed Resizing", resized)
                key=cv2.waitKey(3000)
                if (key & 0xFF) == ord("1"):
                    cv2.destroyWindow("Fixed Resizing")
              elif prediction[0][0]=='ZERO':
```

```
            cv2.rectangle(image1, (480, 170), (650, 420), (0, 0, 255), 2)
            cv2.imshow("Rectangle", image1)
            cv2.waitKey(0)
            key=cv2.waitKey(3000)
            if (key & 0xFF) == ord("0"):
               cv2.destroyWindow("Rectangle")
          elif prediction[0][0]=='TWO':
            (h, w, d) = image1.shape
            center = (w // 2, h // 2)
            M = cv2.getRotationMatrix2D(center, -45, 1.0)
            rotated = cv2.warpAffine(image1, M, (w, h))
            cv2.imshow("OpenCV Rotation", rotated)
            key=cv2.waitKey(3000)
            if (key & 0xFF) == ord("2"):
               cv2.destroyWindow("OpenCV Rotation")
          elif prediction[0][0]=='THREE':
            blurred = cv2.GaussianBlur(image1, (21, 21), 0)
            cv2.imshow("Blurred", blurred)
            key=cv2.waitKey(3000)
            if (key & 0xFF) == ord("3"):
               cv2.destroyWindow("Blurred")
          elif prediction[0][0]=='FOUR':
            resized = cv2.resize(image1, (400, 400))
            cv2.imshow("Fixed Resizing", resized)
            key=cv2.waitKey(3000)
            if (key & 0xFF) == ord("4"):
               cv2.destroyWindow("Fixed Resizing")
Step 7: Interrupt Mechanism:
interrupt = cv2.waitKey(10)
          if interrupt & 0xFF == 27: # esc key
             break
          cv2.destroyAllWindows()
          cap.release()
          cv2.destroyAllWindows()
          return render_template("home.html")
```

V. SIMULATION RESULTS

When the application is initiated, it becomes accessible through the local server at the following link: http://127.0.0.1:5000. Upon accessing this link, users are directed to the homepage of the application. Upon interaction, typically by clicking a designated "launch" button, a new page is presented to the user. Here, the user is prompted to provide an image for transformation. The application facilitates this process by allowing users to upload their desired image. Subsequently, users are presented with an option to proceed with the transformation, typically through a button labeled "predict." Upon clicking this button, the uploaded image is accepted for processing. At this point, a new window emerges, granting the user access to their camera. Additionally, within this window, a designated region of interest is displayed. This region serves as the canvas for users to perform gestures, which will subsequently trigger specific transformations on the initially uploaded image. For clarity and guidance, examples of the gestures and their respective image transformations are visually showcased within the interface. These examples aid users in understanding the range of gestures supported by the application and the corresponding transformations they can expect to witness. Through this intuitive and interactive process, users can seamlessly engage with the application's transformative capabilities.

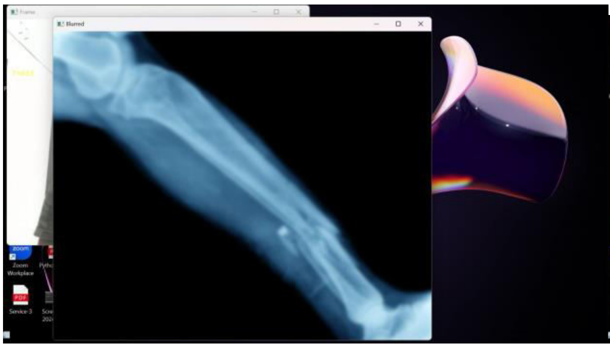


Figure 1

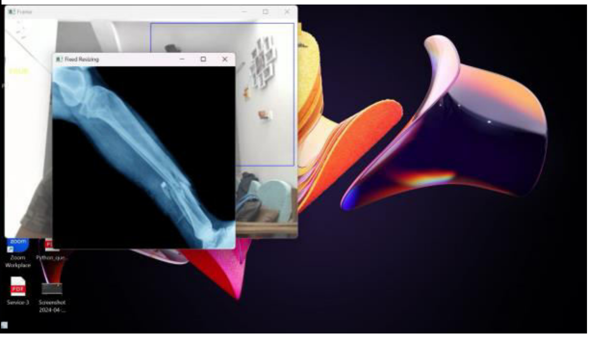


Figure 2

Figures 1 and 2 represent the image transformation for hand gestures three and Four respectively.

VI. CONCLUSION AND FUTURE WORK

The developed gesture-enabled image viewer marks a significant advancement in medical technology, offering doctors a novel way to interact with radiology images using hand gestures. This tool enhances workflow efficiency and reduces contamination risks in sterile environments, crucial for patient safety during surgeries. Its intuitive, hands-free interface, powered by YOLOv3's advanced techniques, ensures accurate and real-time gesture recognition, making it user-friendly and effective. Beyond healthcare, its potential extends to education and presentations, where it can enhance visual learning and seamless content navigation. Future enhancements could focus on improving gesture recognition speed, expanding gesture range, and integrating voice commands, further augmenting its functionality and indispensability in medical settings.

REFERENCES

1. "Hand Gesture Recognition Using Deep Learning YOLOv3 Model" - Rytis Maskeliūnas, Karrar Hameed Abdulkareem, et al., MDPI Applied Sciences, 2021.
2. "A Survey on Hand Gesture Recognition" - J. Wachs, M. Kölsch, H. Stern, Y. Edan, Communications in Computer and Information Science, 2011.
3. "Hand Gesture Recognition: A Literature Review" - P. Garg, N. Aggarwal, S. Sofat, International Journal of Computer Applications, 2009.
4. "Vision-based Hand Gesture Recognition Using Deep Learning" - S. Ge, Z. Luo, D. Wang, IEEE Transactions on Multimedia, 2018.
5. "Hand Gesture Recognition Using Convolutional Neural Networks" - S. K. Jain, M. Sharma, A. Mittal, Springer Advances in Intelligent Systems and Computing, 2018.
6. "Real-time Hand Gesture Recognition Using YOLO" - N. A. Siddique, M. F. B. Saleh, IEEE Access, 2020.
7. "Hand Gesture Recognition Using Depth Data" - J. Wan, Q. Ruan, W. Li, P. Ogunbona, IEEE Signal Processing Letters, 2014.
8. "Dynamic Hand Gesture Recognition Using CNNs" - Y. Xia, C. Zhao, Y. Shao, IEEE International Conference on Image Processing, 2019.
9. "A Framework for Hand Gesture Recognition Using RGB-D Data" - M. Marin, F. Dominio, P. Zanuttigh, Journal of Visual Communication and Image Representation, 2014.
10. "Robust Hand Gesture Recognition with Kinect Sensor" - R. G. Liu, Z. Q. He, Z. Wang, IEEE Transactions on Systems, Man, and Cybernetics, 2013.
11. "Hand Gesture Recognition Using Hidden Markov Models" - X. Zhang, Y. Zhu, Y. Tian, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.
12. "Gesture Recognition Based on Fuzzy Logic" - D. Kim, S. K. Kim, J. H. Lee, IEEE Transactions on Industrial Electronics, 2009.
13. "Hand Gesture Recognition with Leap Motion Controller" - C. Li, Y. Chen, M. Chen, IEEE International Conference on Robotics and Automation, 2015.
14. "Hand Gesture Recognition Using Machine Learning" - V. Pansare, A. Mishra, R. Kothari, International Journal of Advanced Research in Computer Science, 2014.
15. "Hand Gesture Recognition for Human-Computer Interaction" - C. Keskin, A. Erkan, L. Akarun, IEEE Transactions on Human-Machine Systems, 2013.

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

📱 9940 572 462 ☎ 6381 907 438 ✉ ijircce@gmail.com

www.ijircce.com

Scan to save the contact details