Face drawing by KUKA 6 axis robot manipulator

Maxim Pichkalev
Laboratory of Intelligent Robotic Systems
Intelligent Robotics Department, ITIS
Kazan Federal University
maxvseman@gmail.com

Roman Lavrenov
Laboratory of Intelligent Robotic Systems
Intelligent Robotics Department, ITIS
Kazan Federal University
lavrenov@it.kfu.ru

Ramil Safin
Laboratory of Intelligent Robotic Systems
Intelligent Robotics Department, ITIS
Kazan Federal University
safin.ramil@it.kfu.ru

Kuo-Hsien Hsia
National Yunlin University
of Science and Techology
Yunlin City, Taiwan
khhsia@yuntech.edu.tw

***Abstract*—This article describes the algorithm of points extraction from the picture processed by a canny edge detector that will be turned in the curves using the cubic spline interpolation. Here will be presented several examples, including code. Furthermore, this paper will describe steps of creating the system, which will depict on a sheet of paper a picture taken from a web-camera. The system enables a webcam from which will be taken snapshot, the convenient program with the graphical user interface, including algorithms, whose parameters can be changed, TCP server, webcam control, observation of the drawing process and logging. The main drawing component of the system is KUKA KR3 R540. KR3 R540 is robotic system with a manipulator that will take commands to draw a result image.**

Keywords—image processing, robotic system, computer vision algorithm, industrial robot, draftsman robot, OpenCV, KUKA

I. INTRODUCTION

Robots are actively used today. Mobile robots are autonomously planning route and mapping the surrounding space. Robot-manipulators are widely used in industrial problems and they placed on mobile wheeled and tracked robots to manipulate various objects [1]. However, they are also used for scientific purposes due to their excellent accuracy and repeatability. For example, you can calibrate cameras using robotic manipulators KUKA [2]. Alternatively, you can explore the algorithms for the visual recognition of fiducial markers [3], if you attach the marker to the robot's flange [4]. In our work, we will talk about our robotic system based on the KUKA robotic arm and the OpenCV program, with which the robot will draw the taken image.

The tasks of computer vision are being solved today in various systems. In mobile devices, in street video surveillance systems. In addition, computer vision is important in solving the problem of self-calibration of cameras for autonomous robots equipped with manipulators. [5]. We decided to combine face recognition and programming a robot manipulator in a drawing task.

There are not a lot of picture drawing systems and corresponding algorithms that get data in the online regime described. Therefore, this is an attempt to describe such kind of system created from scratch. Existing drawing robotics systems were taken into account. Humanoid artists [6] - [8] and different types of manipulators [9] – [13]. The first humanoid [6] – Fujitsu robot HOAP2 was used to create the human-like process. Firstly, the robot draws contours, and then fill another area. To make the process much easier OpenCV library was implemented. There was used a Canny edge detector [14] to extract face contours, the image was binarized, unwanted contours and noise were removed applying different filters and thresholds. The closed contours were coded as a Freeman chain code. The second humanoid [7] is Pica; here are used the Center-off algorithm to find the main contours of the face. After removing the noises, the next step is to identify all face features. Another step is to prepare features for drawing. The third humanoid is Betty [8]. Here also, the OpenCV Canny edge detection algorithm is implemented. For finding main features for drawing Furthest Neighbor Theta-graphs is used.

Delta manipulator [9] also uses the OpenCV with the Canny edge detector, it draws every pixel. Then when one pixel is drawn, the manipulator moves a pan to nearby point using some rules.

The robot Kuka KR6 [10] has similar steps to the Canny algorithm's steps in the image processing. The way the calculation of the segment of a face is not properly shared. However, it takes only five minutes to draw a realistic portrait and more attention should be spent here. We can also find an interesting robotic system on the YouTube channel [15]. Here is used a sort of halftone technique - error diffusion to get image data and then a division of an image into parts to avoid ignored points that will cause drawing ruining lines through the entire image after using the nearest neighbor search algorithm to find the point with the smallest Euclidean distance to find lines to draw.

Our research is divided into three subtasks:

1. Creating an algorithm, which will translate an image's pixels, taken using web-camera, into the set of pair of points which will represent an image's contour's splines.
2. Creating a server with protocol, which will translate achieved set from step above using TCP channel to the controller of the robotic system. The KRL program for Kuka.
3. Combining image process algorithm and server with protocol into one application.

978-1-7281-3021-7/19/$31.00 ©2019 IEEE
DOI 10.1109/DeSE.2019.00132

II. USED TECHOLOGIES

1. *Opencv[16]*

   Popular open-source computer vision library.

2. *Qt[17]*

   Qt is a framework for convenient developing of cross-platform application with a graphical interface, also have many useful classes, e.g. for network, database, string processing, multithreading, etc.

3. *Work Visual 4.0[18]*

   The software package WorkVisual is the developing environment for KR C4 controlled robotic cells. It is able to program robots offline, editing TOOL and BASE coordinate system and etc.

4. *KR3 R540 [19]*

   KR3 R540 is a robotic system that includes all components of an industrial robot: manipulator (mechanical system and electrical installations, see figure 1); controller; connecting cables; end effector; SmartPAD; software; other equipment.

Manipulator consist of following principal components:

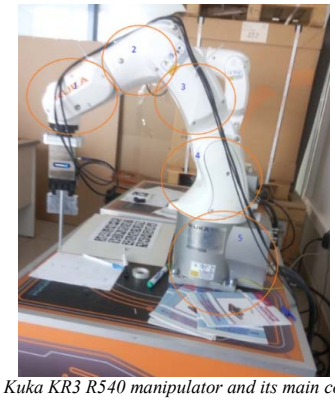

Figure 1. Kuka KR3 R540 manipulator and its main components.

Where:

1. in-line wrist
2. arm
3. link arm
4. rotating column
5. base frame and electrical installations

Axis	Beginning position	Area of axis motions
A1	0°	±170°
A2	-90°	-170 & 50°
A3	90°	-110 & 155°
A4	80°	±175°
A5	0°	±120°
A6	0°	±350°

The programming language of the Kuka controller is KRL – Kuka Robot Language (similar to Pascal language). Code located in two files: one with the dat file extension (here you can find variables), second with the src extension (there are commands).

You can set a different kind of motions in KRL:

- PTP (Point to Point)
- LIN(Linear)
- SPLINE (linear moves and another splines can be inside)
- CIRC(circle)

You can declare variables, arrays and structures and use many other things like loops, logical and arithmetic operations.

III. CREATING AN ALGORITHM, WHICH WILL TRANSLATE IMAGE'S PIXELS, TAKEN USING WEBCAM, INTO THE SET OF PAIR OF POINTS WHICH WILL REPRESENT IMAGE'S CONTOUR'S SPLINES.

Firstly, using canny algorithms we will get contours of the picture with width equals one pixel. Short explanation:

- Prepare image (see fig. 2) using openCV library:

```
cv::Mat imgOrig=cv::imread(absolutePath+"image.jpg");
```

Figure 2. Initial image

- convert RGB image into grayscale

all methods:

I. ((R + G + B) / 3

II. (max (R, G, B) + min (R, G, B)) / 2

III. 0.21 R + 0.72 G + 0.07 B)

In openCV:

```
cv::cvtColor(imgOriginal, imgGrayscale,
COLOR_BGR2GRAY);                    // convert to grayscale
```

Use a Gaussian filter to get rid of the noise. The image will be smoothed. Complexity is of image*kernel equal O(W*H*K^2), where K is a number of rows and column of the kernel. If divide kernel into two one dimensional vectors, then complexity will be O(H*W*K) + O(H*W*K) = 2* O(H*W*K). Also, using Fast Fourier Transform we will get O(W*H*log(W*H)).

In OpenCV:

```
cv::GaussianBlur(imgGrayscale, imgBlurred,
cv::Size(5, 5), 1.8);
```

- Four convolutions of the image with edge detector kernels and computation of the gradient direction
- Non-maximum suppression (choose local minimum)
- Thresholding with hysteresis (remove values which below the low threshold, which between the threshold and are not connected with values above high threshold)

From the canny result we extract a set of points pair using this way (let's call this way "feature extraction algorithm"): Divide image 512*512 pixel into cells.

Go through every cell:

```
for (int i = 0; i <8; i++)
 for (int j = 0; j < 8; j++)
```

E.g., take cell where i equals zero, and j equals three, go through every pixel of the cell:

```
for (int y = i * 64; y < (i + 1) * 64; y++) {
 for (int x = j * 64; x < (j + 1) *64; x++) {
```

Going through every y under current x, if value is not zero, then put it into variable y2;

```
if (imgCanny.at<uchar>(x, y) > 0) {
    imgCanny.at<uchar>(x, y) = 255;
    y2 = cv::Point(x, y); }
```

After passing all y with a certain x, y2 put into currentX vector, checking if it equals to nought.

```
if (y2.x == 0 && y2.y == 0) {}
else { curSixX.push_back(y2); }
y2.x = 0; y2.y = 0;
```

After this, do same operation with all x. Find cubic splines [1] for whole array to depict it. Cubic spline finds function between two points. Main idea of spline is to find unknown parameter taking in our case two derivative.

This way we depict achieved image (fig.3).

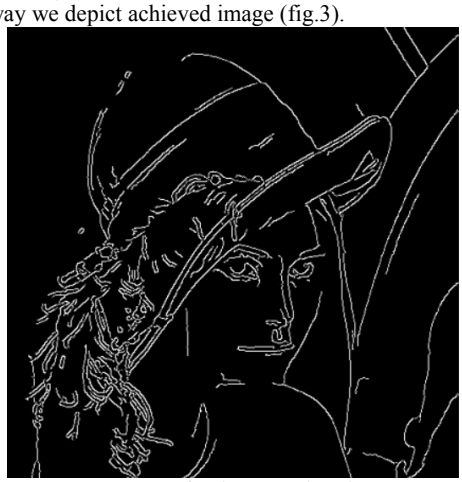

Figure 3. After the Canny algorithm

Submit to the algorithm separated currentX vector - X and Y vectors, based on them calculate function unknown parameters. Set some x, function give us corresponding y. Draw point (fig. 4-5).

```
void opencv::drawSplines (
      cv::Mat &im, vector<double> &vecX,
      vector<double> &vecY, bool isClosed ,
      bool yOrX ) {
   tk::spline s;
   s.set_points(vecX, vecY);
   double beg = vecX.at(0);
   double en = vecX.at(vecX.size() - 1);
   double dif = en - beg;
   double step_size = 0.05;
   int step_number = dif/step_size ;
   double cur_y = 0;
   while (beg < en) {
      beg += step_size;
      cur_y = s(beg);
      if(yOrX == false)
         circle(im, cv::Point(beg, cur_y), 0.5, cv::Scalar(255, 200, 0), 1, 8);
      else circle(im, cv::Point( cur_y, beg), 0.5, cv::Scalar(255, 200, 0), 1, 8);
  } }
```

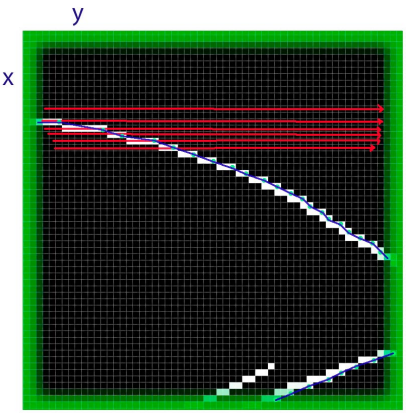

Figure 4. Cell. First for with X. For every x chosen one point. Points put in the currentX vector. After passing all x, draw splines.

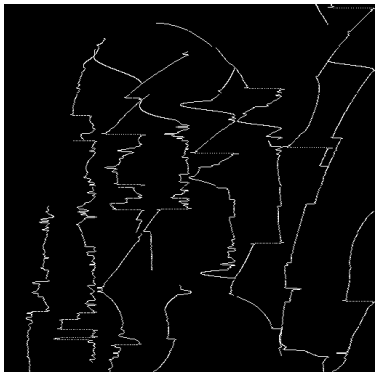

*Figure 5. 8*8 cells with 64*64 pixels.*

The result with 64 cells enough inaccurate depicts an initial image. For more accuracy, we should increase the amount of cells (fig. 6). Also, for simplicity pull out some currentX vectors, setting the threshold of currentX size:

```
if (curSixX.size()>4) {
```

*Figure 6. 50*50 cells with 10*10 pixels and pull out currentX vectors with size <4, in sum we get 682 cells (first for with X);*

We may increase the result image quality, at first computing algorithm with first for with Y (fig. 7), and sum this with the result of the algorithm there first for with X.

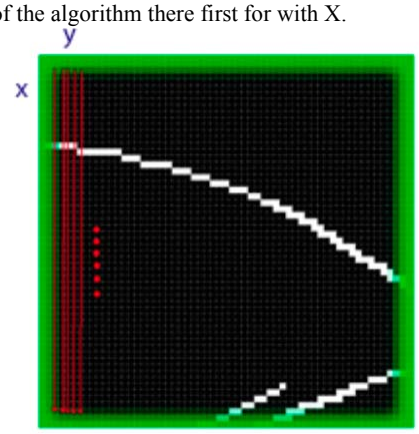

Figure 7. Cell. First for with Y.

For the simplicity of data transfer protocol in every currentX vectors, leave only first and last values. The message consisting of two points is easier transfer than the message with a dynamically changing number of points (fig. 8).

For the reading of the image from the web camera, we use openCV:

cv::VideoCapture capWebcam(0);

OpenCV and opencv_contib libraries were built using CMake, MinGW generator.

IV. REATING A SERVER WITH PROTOCOL, WHICH WILL TRANSLATE ACHIEVED SET FROM STEP ABOVE USING TCP CHANNEL TO THE CONTROLLER OF THE ROBOTIC SYSTEM. THE KRL PROGRAM FOR KUKA.

The protocol:

The server part. There is the integer variable - counter that is assigned a value of zero.

1) waiting for the client (for the robotic system). The connection.

2) the server transfers a message, the counter is incremented by one, waits for the client reply.

3) repeat step two.

4) If the counter equals two, the server waits for the control message, after getting which the server goes to the first point, again the counter is assigned zero.

The client part:

The variable counter is assigned one (because in the Kuka array the first index is one).

1) the connection to the server.

2) the waiting message from the server, after getting which sends an echo message, the counter is incremented, now it is two, moves the manipulator.

3) the counter equals two, gets the message, moves the manipulator, sends the echo message, the counter is incremented;

4) the counter equals three, the client sends the control flag message and goes to the second point making the counter equals one.

Classes for the work with TCP in the test client, for the creating socket, were taken from the c++ library <WS2tcpip.h> and for the server application was used the Qt network model.

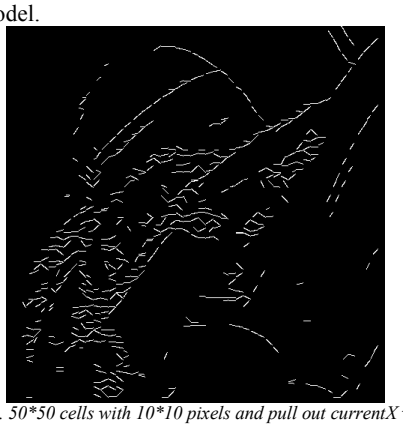

*Figure 8. 50*50 cells with 10*10 pixels and pull out currentX vectors with size <4, in currentX vectors leave first and last points, in sum 541 currentX vectors (first for with Y);*

What is about the robotic system, for data exchanging we used the EthernetKRL package of functions [18], letting:

- take XML or binary information
- send XML or binary information
- set controller like client or server
- connection configuration using XML file

There could be also used Kuka RSI (Robot Sensor Interface) [20] for synchronous hard real-time communication, where you should provide command updates every 12ms, or your robot will stop. C

The program for manipulator control was written in KRL [21]. We used a structure where x, y, z is a position, a, b, c is an orientation in space; S (status) and T (Turn) were used to eliminate the ambiguity of a position (it can be achieved with different axis position).

Calibrated tool and base that are shown in Figure 9. Base's origin is on the corner of the box, the tool's origin is at the end of the pen.

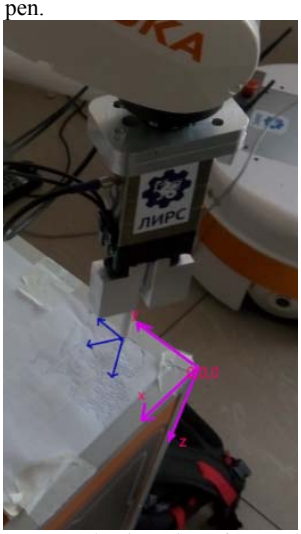

Figure 9. Base and tool coordinate frames. Blue is the tool. Purple is the base.

XML message format:
"<Server><Pos2><X>SomeVariableX</X><Y>SomeVariableY</Y><Z>0.15</Z><A>65.75</A><B>-81.44</B><C>150.0</C><S>2</S><T>3</T></Pos2></Server>"

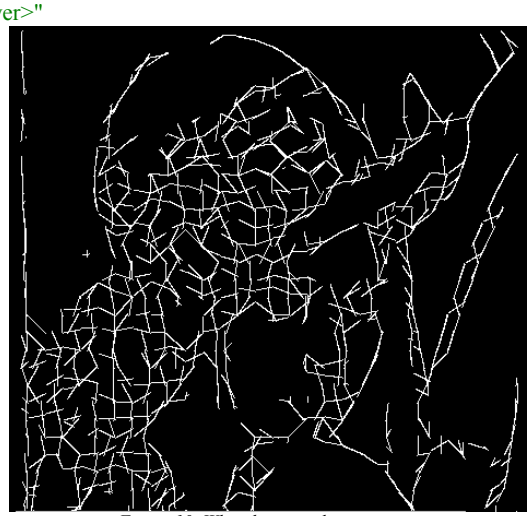

Figure 10. What depicts robotic system.

The drawing of one line takes about 3.5 seconds, in sum 1050 lines. (fig.10) It takes about one hour to draw the picture above. Results you can see in figures 11-13. The velocity of movements equals twenty percent of the maximum of the KR3 R540.

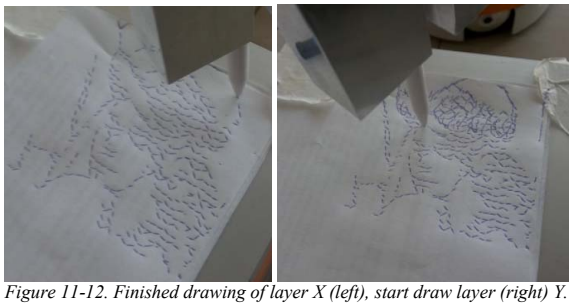

Figure 11-12. Finished drawing of layer X (left), start draw layer (right) Y.

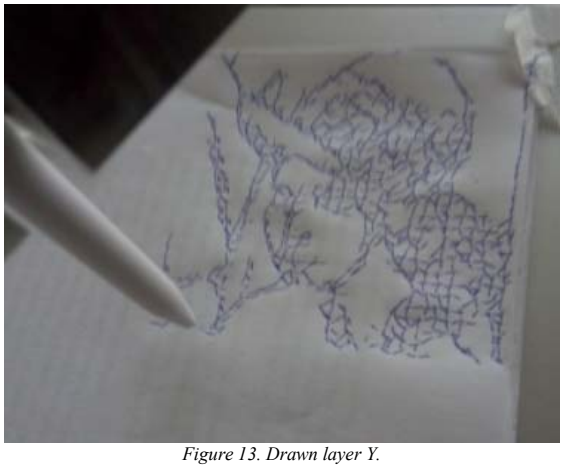

Figure 13. Drawn layer Y.

V. Combining Image Process Algorithm and Server with Protocol into One Application

In the beginning, we can see the window with such elements:

- createServer – create a server and wait for clients.
- startCam – start camera, add new functions.
- downloadImg – download a picture, instead of getting a picture from the camera.
- rebootS – reboot the server.
- exit – close the application

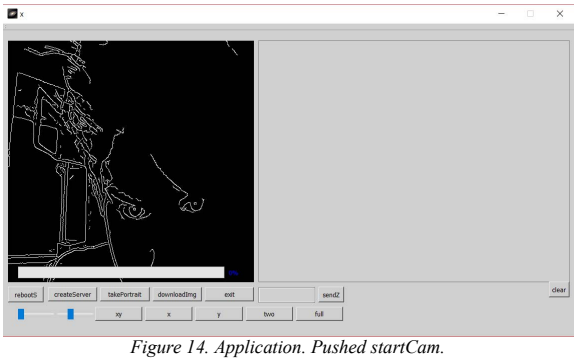

Figure 14. Application. Pushed startCam.

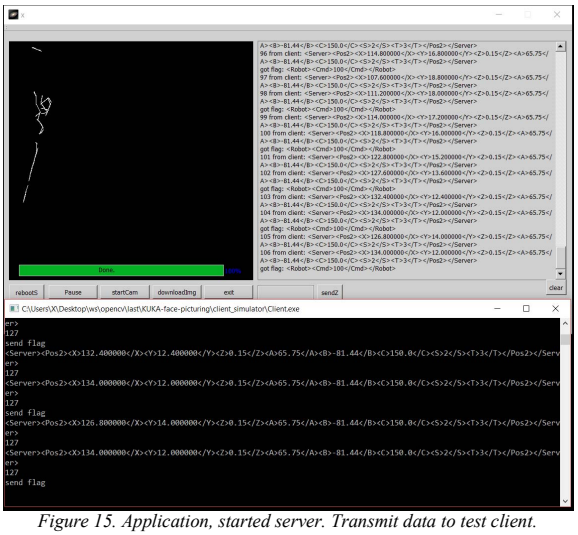

Figure 15. Application, started server. Transmit data to test client.

Shortly about the application (fig. 14-15): the first group of regimes: xy, x, y. XY handles image with two passes (one has main for X, another Y). The second regime group: full and two. Two takes only the first and last value of the currentX vector. In full, we take all currentX vector's values. The Button takePortrait puts the current camera's frame in the algorithm, takes a result, saves it with a name saved.jpg and attaches it to display window, also saves pairs of points in the file points.txt for server needs.

To launch the application on your computer, you can download the release folder with the .exe file, .dll files and image.jpg picture (512x512 pixels). All files must be placed in the same directory with a .exe application file. In addition to the main application, there are plain algorithm code, test TCP client, KRL code in the repository.

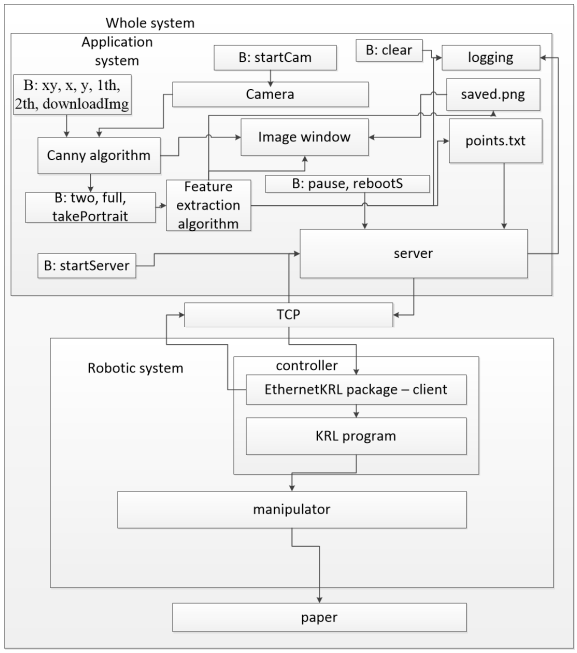

Figure 16. Whole system.

VI. CONCLUSIONS

The resulting system is shown in figure 16. On the top is the application system and the bottom is the robotic system. Arrows show dependencies between elements.

"B: name" means buttons and their names.

The camera can be turned on by pushing the button startCam. The canny algorithm gets input values depending on the number of pushed buttons (XY, X, Y, th1, th2, downloadImg). Button downloadImg downloads an image, which placed in the same package with name image.jpg. This image will be used by Canny algorithm instead of the camera's images. The image window displays the Canny algorithm's or Feature extraction algorithm's results. The image window gets images from the Canny algorithm or the Feature extraction algorithm. The Feature extraction algorithm is an algorithm for extracting the image`s main feature's points described above. The Feature extraction algorithm saves result image with the name saved.png and points to the file points.txt, also it sends logs to the logging element. The server can be started by startServer button then it will wait for the client connection using the specified IP address and the concrete port 59152. After receiving a connection, the server starts to transmit messages. Buttons pause and rebootS can stop or reboot the server. The server also depends on the points.txt file that it will send to the client through TCP. The EthernetKRL package is deployed as the client and it connects to the server and gets messages, puts them into the KRL program, and sends commands to the manipulator. The manipulator draws an image.

The next step in this research is to draw the picture with splines as in the fig. 10 that improves the time parameter at least twice by using segments calculation, like a KR 6 robot [19].

Repository: https://gitlab.com/LIRS_Projects/KUKA-face-picturing

ACKNOWLEDGEMENTS

The reported study was funded by the Russian Foundation for Basic Research (RFBR) according to the research project No. 18-58-45017.

REFERENCES

[1] Mavrin I., Lavrenov R., Magid E. Development of a Graphical User Interface for a Crawler Mobile Robot Servosila Engineer //2018 11th International Conference on Developments in eSystems Engineering (DeSE). – IEEE, 2018. – pp. 192-197.

[2] Khusainov R., Klimchik A., Magid E. Humanoid robot kinematic calibration using industrial manipulator //2017 International Conference on Mechanical, System and Control Engineering (ICMSC). – IEEE, 2017. – pp. 184-189.

[3] Sagitov, A., Shabalina, K., Lavrenov, R., Magid, E. Comparing fiducial marker systems in the presence of occlusion //2017 International Conference on Mechanical, System and Control Engineering (ICMSC). – IEEE, 2017. – pp. 377-382.

[4] Shabalina, K., Sagitov, A., Svinin, M., Magid, E. Comparing Fiducial Markers Performance for a Task of a Humanoid Robot Self-calibration of Manipulators: A Pilot Experimental Study //International Conference on Interactive Collaborative Robotics. – Springer, Cham, 2018. – pp. 249-258.

[5] Meng Y., Zhuang H. Self-calibration of camera-equipped robot manipulators //The International Journal of Robotics Research. – 2001. – V. 20. – №. 11. – pp. 909-921..

[6] Calinon S., Epiney J., Billard A. A humanoid robot drawing human portraits //5th IEEE-RAS International Conference on Humanoid Robots, 2005. – IEEE, 2005. – pp. 161-166.

[7] Lin C. Y., Chuang L. W., Mac T. T. Human portrait generation system for robot arm drawing //2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. – IEEE, 2009. – pp. 1757-1762.

[8] Lau M. C. et al. A portrait drawing robot using a geometric graph approach: Furthest Neighbour Theta-graphs //2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). – IEEE, 2012. – pp. 75-79.

[9] Hsu C. F. et al. Motion planning and control of a picture-based drawing robot system //2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS). – IEEE, 2017. – pp. 1-5.

[10] Jean-Pierre G., Saïd Z. The artist robot: A robot drawing like a human artist //2012 IEEE International Conference on Industrial Technology. – IEEE, 2012. – pp. 486-491.

[11] Ye X. et al. Deep learning-based human head detection and extraction for robotic portrait drawing //2017 IEEE International Conference on Robotics and Biomimetics (ROBIO). – IEEE, 2017. – pp. 1282-1287.

[12] Tresset P., Leymarie F. F. Portrait drawing by Paul the robot //Computers & Graphics. – 2013. – V. 37. – №. 5. – pp. 348-363.

[13] Jain S. et al. A force-controlled portrait drawing robot //2015 IEEE International Conference on Industrial Technology (ICIT). – IEEE, 2015. – pp. 3160-3165.

[14] Canny J. A computational approach to edge detection //Readings in computer vision. – Morgan Kaufmann, 1987. – pp. 184-203.

[15] Super Make Something Robotic Drawing Machine. [Online video]. URL: https://www.youtube.com/watch?v=OuCiHp43q20&t=370s

[16] Opencv. URL: https://docs.opencv.org/3.2.0/index.html

[17] Qt. URL: https://www.qt.io/

[18] KUKA Roboter GmbH, WorkVisual 4.0, Zugspitzstraße 140, D-86165 Augsburg, Germany

[19] KUKA Deutschland GmbH, Spez KR 3 AGILUS, Zugspitzstraße 140, D-86165 Augsburg, Germany

[20] KUKA Roboter GmbH, KUKA.RobotSensorInterface 2.3, Zugspitzstraße 140, D-86165 Augsburg, Germany

[21] KUKA Roboter GmbH, Kuka.Ethernet KRL 2.1, Zugspitzstraße 140, D-86165 Augsburg, German