Gesture-Based Robot Programming Using Microsoft Kinect

Sebastian Blankemeyer(✉), Joshua Göke, Tobias Grimm, Benedikt Meier, and Annika Raatz

Institute of Assembly Technology, Leibniz Universität Hannover, An der Universität 2, 30823 Garbsen, Germany
{blankemeyer, raatz}@match.uni-hannover.de

Abstract. Companies have to face constantly changing framework conditions. Product lifecycles are reduced and the number of variants increase as a result of increasing product individualization. To ensure that production systems such as robots can achieve this flexibility, more and more programming procedures are being developed that do not require a high level of qualification. This thesis deals with an intuitive approach that can be used for programming adhesive joints. With the help of the simple pointing to surfaces, a path corresponding to the intersection line of the two planes is calculated. The required algorithm for fingertip recognition is implemented with the help of Voronoi diagrams. The recognition of the planes is based on a region growing algorithm. The accuracy of the whole process is limited by the current technical state of the art and does not yet meet the necessary requirements.

Keywords: Robot programming · Intuitive human machine interface
Gesture recognition

1 Introduction

Nowadays a high degree of automated processes exist, especially in productions with high unit numbers. For small and medium-sized companies however, fully automated production processes are not cost-efficient due to the lower number of units and the highly specialized processes that are tailored to the customer requirements. Human robot collaboration (HRC) is an approach trying to increase productivity especially in small to medium-sized production quantities. Here the advantages of a robot, such as its precision and repeatability, are combined with the problem-solving and adaptability of humans. On the one hand it is important to ensure the safety of the human worker, who must work with the robot in its immediate environment and, on the other hand, to find approaches that enable people to interact quickly and intuitively with the robot, thus to guarantee the optimal workflow. In the research area of HRC different types of programming methods are developed. One approach is learning from demonstration, where the user programs the robot by showing it successful examples. This can happen through guiding the robot through its path (also called kinestic teaching) or by capturing the movements of the user that the robot has to imitate [1].

Published by Springer Nature Switzerland AG 2019
S. Ratchev (Ed.): IPAS 2018, IFIP AICT 530, pp. 72–81, 2019.
https://doi.org/10.1007/978-3-030-05931-6_7

In addition, some work has already been done developing gesture based programming methods. To implement the programming methods, many of the projects use the Kinect v2 as camera system. The Kinect v2 has been analyzed, exploited and continuously optimized, since its first release [2, 3]. Additionally, an accompanying Software Development Kit (SDK) exists that comes with a lot of practical features like an implemented body tracking function, different types of images and depth measurement [4].

In [5], an approach has been presented that allows the user to teach in points to a welding robot over gestures. For saving start or end points the position of the right hand is saved when the user's left arm is raised. This approach allows even inexperienced users to communicate with a robot. For industrial usage, however, the result might be too inaccurate due to the usage of the system-integrated hand joint tracking of the Kinect, as the position detection of the hand was fluctuating [5]. One way to improve accuracy of the process could be implementing a more precise finger detection. In Canal et al. [6] a programming method for assistive robotics is described in which a user points to objects that is detected by the robot. Then the robot will move to the object pointed to. In order to determine the resulting pointer direction the software calculates the angles between the separate arm bones and the body.

In this paper we present the description of a new approach to an intuitive programming and interaction method for industrial robots, especially for adhesive application. The approach is based on gesture control using a Kinect v2. By pointing on planes, the operator can show the objects that need to be glued together. After that the robot path is determined by calculating the cutting edge. The needed and implemented fingertip recognition algorithm is described as well as the required plane detection and the corresponding precision of the approach. On the basis of the conclusions, the future work is elaborated and described.

2 Concept of Programming

The goal of this approach is to develop and implement an intuitive operating concept for industrial robots in human-robot collaboration. For this purpose, a gesture control system is developed which enables interaction with the robot only with the help of hands and does not require any other input devices. To do this, the operator points to two adjacent objects one after another (Fig. 1). By specifying the areas, the program then calculates the cutting line so that it can be transferred to the robot. This can be used in particular for the production of glued joints especially when the position is moved from one object to the next. However, the process can also be used to produce welded joints or carry out press processes. In order to ensure that the robot can also be moved intuitively and without the aid of a hand-held user panel, an additional mode for moving the robot with the aid of a hand movement is implemented. This mode includes both positioning and orientation of the end effector.

A Kinect v2 is used to implement the gesture control. These time-of-flight cameras can generate 3D data of the environment and are often used in research and development. Using the supplied API, people can be easily recognized and transferred to a skeleton model. The Kinect is fixed to the workstation so that the operator is always in view.

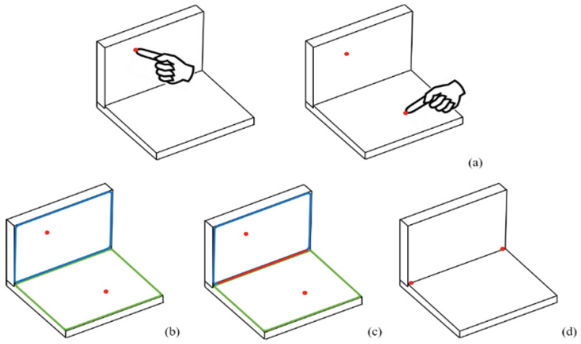


Fig. 1. Principle of corner point detection, (a) selection of planes, (b) plane detection, (c) receiving cutting line, (d) corner points

In the "teach in points" mode, the layers are defined by simply pointing to the surfaces (Fig. 2). These planes are saved and then the common intersection line is determined. Finally, this line is translated into a movement command for the robot.

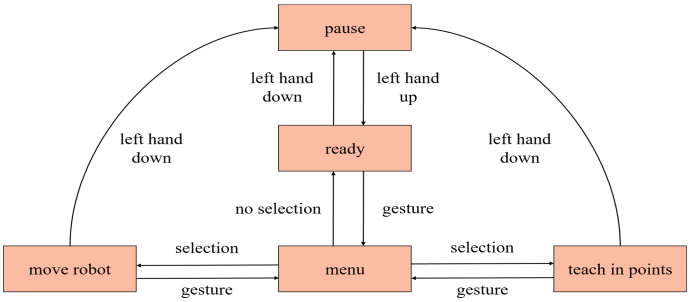


Fig. 2. Concept of interaction

To switch between modes, an intuitive interface is provided in the form of a menu. The inputs can also be made here without any further operator panels.

To ensure safety for humans at all times, an additional safety function is added. Thus, entries can only be made if the user's hands are both above the navel. This makes it possible to distinguish between intentional and unintentional entries. In addition, the program is paused and the movement commands are not forwarded to the robot.

The challenges in creating the programming procedure are, on the one hand, the recognition of the fingertip to show the layers. This must be precise and reliable in order to select the correct plane. On the other hand, a surface detection is required so that the surroundings can be detected. Finally, the cutting edge must be reliably detected and calculated. The corresponding explanations for the fingertip recognition and the surface recognition will follow in the next chapters.

3 Recognition of Fingertips

The recognition of fingers and fingertips belongs to the research area of computer vision. Traditional algorithms are looking for skin-colored areas within the images. This is a challenging task due to changing lighting conditions and different skin colors [6]. Since the introduction of consumer depth camera like the Kinect a lot of research has been done to use depth images and the identified skeleton of the Kinect for recognition algorithms [7].

The Kinect v2 can track up to six persons and calculates a skeleton with 26 joint positions in 3D coordinates for every person. The accuracy of the recognized hand position is unfortunately not very precise. However, we are using it as a first estimate for our algorithm to set the region of interest and to create a field of view (FOV) of the depth image.

The size of a hand in an image is dependent on the distance to the camera. Therefore, we are using the recognized hand position of the Kinect as the center of the image and calculate the image size dynamically to get the best FOV. Afterwards we are using a foreground segmentation which uses the depth information of the hand position to remove unwanted parts of the image. Following this, we implemented a finger detection algorithm which is based on the hand recognition algorithm of Yeo [8]. We have adapted the algorithm in order to obtain a higher reliability and accuracy, as this is necessary for our approach.

Many feature-based algorithms are using the palm of the hand to determine the size of the hand in the image. Yeo calculates the largest empty circle (LEC) within the hand contour to approximate the palm of the hand. This is a computation-intensive task, so Yeo is sampling down the image and checks every N-point instead of all points in the hand contour. For these points the shortest distance to the contour is calculated to get the radius of the LEC.

The problem of finding the LEC is analyzed in Gesing [9] and a solution is presented using a Voronoi-Diagram. Gesing shows that the center of the LEC can be either on a Voronoi node or on an intersection between a Voronoi edge and the convex hull of the points. For our application we are approximating the palm of the hand by the LEC, so the center must be inside the contour on a Voronoi node. Thus, we are calculating the convex hull of the contour and create a Voronoi-Diagram. Afterwards we examine every Voronoi node and calculate the shortest distance to the contour to get the radius of the LEC.

The feature-based recognition of fingertips utilizes several criteria to filter possible fingertips. Yeo uses a circle with 3.5 times the radius of the LEC to remove unwanted points in the arm area (outer red circle in Fig. 3 left). In our use case it turned out that we had too many false detections for a reliable execution of the algorithm. We solved this by using additional information from the Kinect to filter possible fingertips. The identified joints of the hands include the hand wrist (blue point in Fig. 3) as well as the recognized center of the hand by the Kinect (pink point in Fig. 3). Our algorithm creates a line through both points and calculates the angle to the potential fingertips. We are limiting the angle to 110° because it is anatomically hard to create a greater angle. Figure 3 (right) shows the line through the points as well as the resulting angle limit (dark area). By using this method, we are independent of the hand orientation and we can filter more potential fingertips.

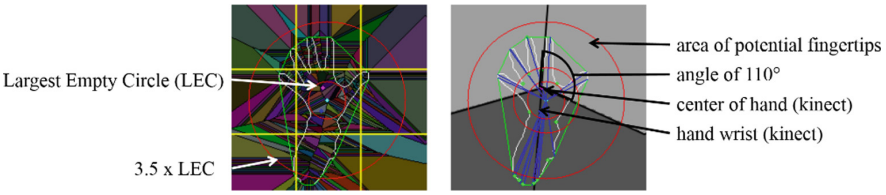


Fig. 3. Voronoi-Diagramm of the hand contour (left) and filtering points in the arm area (right) (Color figure online)

Sometimes there are multiple detections on the same finger, so Yeo is checking the surrounding area of each detection and deletes points nearby. Our approach uses k-curvature to find the fingertip as well, but we are creating a region of interest (ROI) around every possible fingertip. This is done by adding the surrounding points on the contour in forward and backward direction to a list. After creating individual lists for every possible fingertip, we are checking the lists for overlaps and combine them if necessary. This approach eliminates multiple detections for one fingertip and we can reduce the calculation of the k-curvature because every point is checked only once.

Afterwards we calculate the k-curvature to find the exact position of the fingertip. The last step of the finger recognition algorithm is the identification of a gesture. The possible gestures can be seen in Fig. 4. For our application we are distinguishing between a fist (zero fingers), a pointing finger (one finger), a pistol (two fingers) and a rock gesture (three fingers). The open hand belongs to four to six detected fingers to react to one false detection or nearby fingers. For safety reasons, the gesture must be recognized in several consecutive images until it is published.

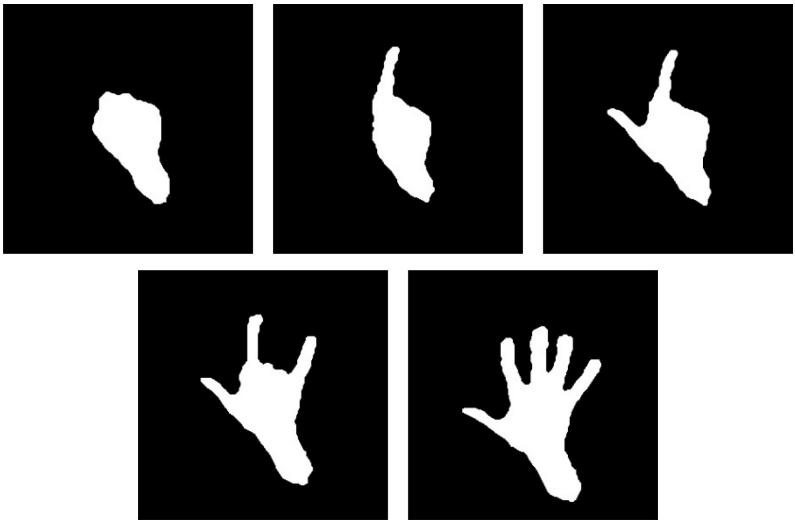


Fig. 4. Recognizable gestures: fist (top-left), pointing finger (top-mid), pistol (top-right), rock (bottom-left), full hand (bottom right)

In our program, we need all gestures except “pistol”. The most important gestures to interact with the program are the “pointing finger” and “fist” to mark or save the planes. With the help of the described finger detection we are able to reliably interact with the program. Furthermore, we detect the fingertip to recognize the correct surface.

4 Recognition of Planes

The next part of this work deals with the question of how the starting and end points of linear movements can be defined. In this section we present an algorithm which identifies and locates the edge, so the robot can move along it, as it is usually used in welding or gluing processes. To identify the edge, a recognition algorithm of two adjacent planes is presented, which are selected through the finger recognition already presented.

4.1 Related Work

Past research projects have been concerned with the optimal recognition of planes from images and three-dimensional point clouds for some time now. In principle, the methods for the identification of planes in 3D can be categorized into three classes according to Holz et al. [10]: Methods representing the RANSAC algorithm that iteratively try to represent a plane in space as best as possible by a random choice of points. In order to describe a plane the number of points needed, three points in total, is taken randomly. Furthermore the number of points that are within a certain tolerance band around the newly created plane are considered. The plane regression, which describes the real plane the best after a predetermined number of iterations is stored. In [11] a RANSAC algorithm is used for identifying the plane of a table in order to distinguish objects laying on the table of the same. Advantages of this approach are the robustness and accuracy even in data sets with a large proportion of outliers. However, the random selection of samples makes them slower than procedures that can handle ordered records. A further disadvantage is the tendency of the algorithms to simplify complex structures. For example, two parallel planes with only a small distance between them can be recognized as one [12]. Furthermore, there are methods of 3D-Hough-transformation. The Hough transformation is the de facto standard for line and circle recognition in images. There are different extensions existing for three-dimensional space. For an overview, reference is made to [13] and [14].

The third category of methods for the identification of planes in 3D point data is Region-Growing, in which the image structure is used for orderly search for geometries within the image. Starting from a point, the neighborhood is examined for previously defined properties. In [10] the authors present their method for plane recognition. In doing so, they derive a grid of 3D points from the image structure and use the neighborhood to create a local surface normal and estimate curvatures. In order to get noise reduction, the points are filtered through a bilateral filter and segmented based on the region growth.

4.2 Plane and Edge Detection

After testing the procedures, our approach is based on the region growing algorithm as well. We decided to take this method, because in our program only two separate planes need to be detected and thus the computing effort is practicable. To recognize the surfaces, we compare the surface normal vector with the ones of the neighborhood. The algorithm associates the pixels to one surface as long as the normal vectors are in a defined range. This will be done until all neighbors of the surface are examined and declared as outside.

At first in order to be able to compare surface vectors a normal frame is produced every time the user starts to communicate with the robot. The frame contains a surface vector for every pixel in the image. To receive a better normal frame, five depth frames are saved and the average of them is computed. According to Lachat et al. [15] a high number of averaged frames does not correlate with increased accuracy, which is why a small number of frames is sufficient. They tested the accuracy of the depth image with different numbers of frames from 20 to 200. Results for fewer frames may be less smooth, but above 50 frames no major changes could be observed. Trying to keep the computing time low, we decided to average only five frames. In addition to this the frame is edited with a bilateral filter to smooth the 3D points even further. The starting point of the region growing algorithm is defined as the point where the fingertip is detected in. In order to determine a first provisional plane, the average normal vector is then formed from all normal vectors belonging to a 10×10 quadrate of points around the starting point. Starting from this square, the neighbors are examined for the orientation of their normal vectors and their distance from the previously defined plane. For each point newly added to the plane, the surface vector is recalculated as the mean value of all normal vectors whose points have been added to the plane so far.

If a point cannot be assigned to the plane due to the fact that the normal vector is differently orientated, it will be saved as an "edge point". Its surrounding will not be examined any further. The plane detection is finished as soon as no further neighborhood points can be added to the plane, so that the plane is surrounded by edge points completely. The surface detection is executed for both planes that are needed. In Fig. 5 an example is shown, where two planes of an object have been identified.

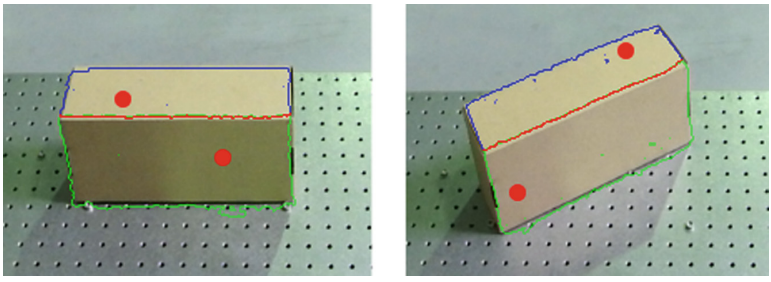


Fig. 5. Detected edge points of the layers on a carton. The red dots represent the starting points of the region-growing algorithm. (Color figure online)

In order to get the intersection line that the two planes have in common, every surrounding point of the first plane is examined whether it is also part of the surrounding of the other plane. If a surrounding point of the first plane is also a surrounding point of the second plane, or at least in the near of it, the point will be marked as edge point. After all points have been examined, the result is only approximately a line, which is caused by not accurate measurements (see red "line" in Fig. 5). A linear regression is applied to get a real line for a straight robot movement.

After determining the edging line by the use of the linear regression, the start and endpoint are transferred to the robot. Then, the robot can execute the task of applying the glue. In order to determine the precision of the process and thus of the product quality, we conduct some tests that are described in the next chapter.

5 Precision of Surface Recognition

Since in production environments varying light conditions can be assumed, we use 3D information for our plane detection instead of color differences in an image that can be distorted heavily by light reflection. Nevertheless, in order to investigate the accuracy, experiments were first carried out on a matte object as well as later on a highly reflective object. It has been assumed that the positioning accuracy of the robot is two decimal potencies better than the accuracy of the depth information of the Kinect, as the Kinect's accuracy was extensively examined by [4] and [15]. With this assumption, the measurement of the robot was set to be accurate. The mean value, standard deviation and repeatability out of 20 attempts for each corner point were examined to measure the accuracy of the corner points. In every attempt the surfaces were selected, the intersection line calculated and then the outermost corner points computed. After that the robot moved to these two points and the distance between the edge points of the real object and the robot tool was measured. The results are listed in Table 1.

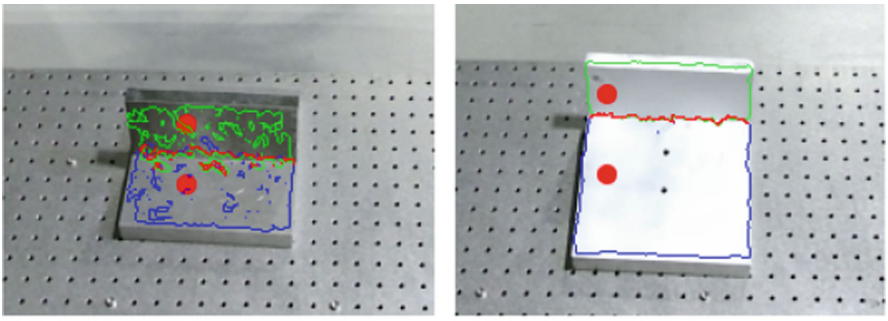
Table 1. Accuracy edge point detection

Test object	Mean value in [mm]	Standard deviation [mm]	Repeatability [mm]
Cardboard	5.14	1.24	8.85
Aluminum	6.99	4.44	20.3
Aluminum treated	7.17	1.7	12.68

One of the points that have been noticed is that the calculated corner points are placed inwards along the edge compared to their real counterpart. The reason for this is the use of the surface normal also at the surrounding of the planes. For the calculation, vectors of points are needed which do not belong to the object and thus lead to inaccurate normal vectors in the area. This results in rounded edges and also in the high mean value stated in Table 1.

Looking at the second examined object, an angle of aluminium, it has been found that the reflective properties of the aluminium significantly impairs the recognition of

the planes. Unfortunately, the 3D information are influenced by the surface of the object, since the Kinect v2 operates with infrared radiation to maintain the depth information. Thereupon, the object was treated with a developer, as found, for example, in material testing applications to minimize the reflection characteristic. Figure 6 shows the difference.



(a) (b)

Fig. 6. Layer detection on metal. (a) untreated, highly restricted detection; (b) treated, better but not optimal

Our approach allows users to easily teach in points for the robot that would be otherwise difficult to reach. It enables movements along edges of workpieces that can be used for example for welding or gluing processes. In tests we demonstrated that our work can be used for processes that do not have high precision requirements. For matte objects such as cardboards or treated metals, the trajectory planning works reliably, however, especially for welding applications the distortion of Kinect's depth information through reflective surfaces can be seen as a challenge, as it leads to false surface detections and thus to unusable paths.

6 Conclusion and Outlook

This paper provides an approach to simplified programming of industrial robots. The main focus of the applications is on adhesive application and welding processes. The programming process is designed in such a way that it is very intuitive and does not require any specialist knowledge. With the help of a menu, it is possible to switch between the two modes "Move robot" and "Teaching points".

The implemented algorithm for fingertip recognition is necessary, as this is not provided by the API of Kinect. Recognition is based on the use of the human skeleton and the determination of the region of interest. The fingertips are determined with the help of Voronoi diagrams and appropriate filters.

The subsequent detection of planes is based on a region growing algorithm. Since the edge of the planes is not recognized by the algorithm as a straight line, a linear regression is applied through the point cloud. In this way, the start and end points of the

robot movement are determined in addition to the path. The results show that the available accuracy is not yet sufficient to use in production environments. Particularly in the case of reflecting surfaces, very high deviations are still achieved in some cases, so that these materials cannot be processed.

In future research, the use of other camera systems will be tested in order to be more independent of the material properties and thus increase system accuracy. Furthermore, the detection of more types of edges (circles, ovals etc.) will be investigated in order to expand the application areas.

References

1. Akgun, B., et al.: Trajectories for kinesthetic teaching: a human-robot interaction perspective. In: Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, vol. 12, pp. 391–398 (2012)
2. Lun, R., Zhao, W.: A survey of applications and human motion recognition with microsoft kinect. Int. J. Pattern Recogn. Artif. Intell. **29**(5), 1555008 (2015)
3. Yang, L., et al.: Evaluating and improving the depth accuracy of kinect for windows v2. IEEE Sens. J. **15**(8), 4275–4285 (2015)
4. Windows Dev Center. https://developer.microsoft.com/de-de/windows/kinect/hardware. Accessed 08 Nov 2017
5. Landa-Huartado, L., et al.: Kinect-based trajectory teaching for industrial robots. In: Pan-American Congress of Applied Mechanics, vol. 14 (2016)
6. Canal, G.: A real-time human-robot interaction system based on gestures for assistive scenarios. Comput. Vis. Image Underst. **149**, 65–77 (2016)
7. Taylor, J., et al.: Efficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences. ACM Trans. Graph. (TOG) **35**(4), 143 (2016)
8. Yeo, H., et al.: Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware. Multimed. Tools Appl. **74**(8), 2687–2715 (2015)
9. Gesing, S.: Approximation von Punktmengen durch Kreise (2005)
10. Holz, D., Behnke, S.: Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. Intell. Auton. Syst. **12**, 61–73 (2012)
11. Stueckler, J., et al.: Real-time 3D perception and efficient grasp planning for everyday manipulation tasks. In: European Conference on Mobile Robots, vol. 5 (2011)
12. Zhang, L., et al.: Fast plane segmentation with line primitives for RGB-D sensor. Int. J. Adv. Robot. Syst. **6**, 1–8 (2016)
13. Vosselmann, G., et al.: Recognising structure in laser scanner point clouds. Inf. Sci. **46**(8), 1–6 (2004)
14. Borrmann, D., et al.: The 3D Hough transform for plane detection in point clouds: a review and a new accumulator design. 3D. Research **2**, 1–13 (2011)
15. Lachat, E., et al.: First experiences with kinect V2 sensor for close range 3D modelling. In: International Workshop 3D-ARCH, vol. 6, pp. 93–100 (2015)