Master IT – Engineering
Module: Robotics
Supervised by:
Prof. Dr. Ulrich Hoffmann and Hermann Höhne

Car Puzzle
KUKA finds a match

Jakob Brüst, Piet Gömpel, Niels Röschmann, Anurag Tyagi

fhwedel
UNIVERSITY OF APPLIED SCIENCES

KUKA finds a match

Introduction

For the scope of this project, the robotic setup has been limited to a 6-axis articulated robot arm, equipped with a high power camera and a gripper powered by hydraulic cylinders. The goal of the project is to implement and apply a robot arm to perform a moderate task, ideally faster than a human would. Particularly, the robot is programmed to solve a puzzle on its own that contains various pieces of different shapes and the respective slots. The computing system applies Computer Vision to provide data to the robot and supplement the movement. In an ideal case, the robot places all the pieces in their correct slots, with sufficient speed and agility. A successful implementation of this project will reinforce the undeniable fact that industrial robots provide a wide and dynamic array of applications and uses which can benefit industries and improve their productivity. A specific real-world extension of this project could be the employment of robots in container docking and shipping, rendering the need of a human `mover' useless while increasing the output and decreasing the error rate.
Within this project a combined software of python openCV and the Kuka-software will be implemented. Main purpose of this new software is to solve a puzzle. Image processing will be used to find the pieces and place them to the correct places by using the Kuka robot.

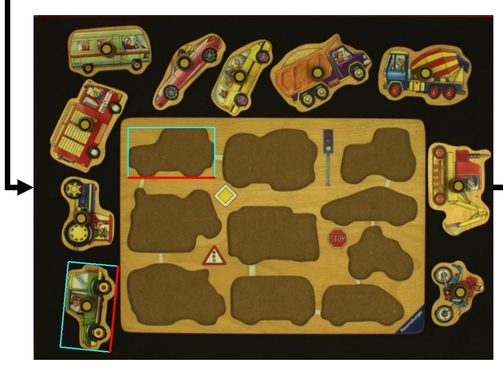
Computer Vision using python openCV

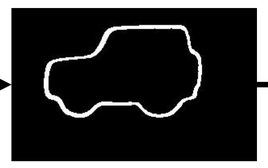
1. Take a picture

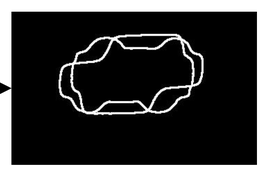
2. Find puzzle slots

3. Find handle centers

4. Calculate paralax effect







5. Calculate angle between piece and slot

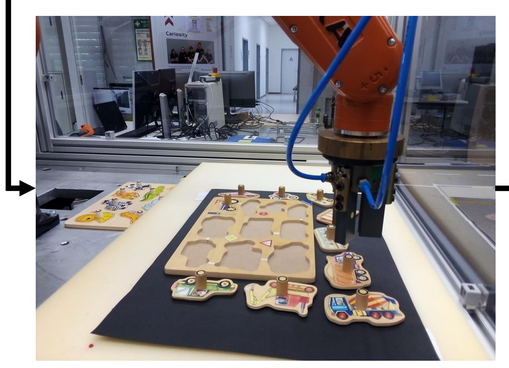
6. All relevant information is calculated

2. Detection of slots contours: The detection of the slot contours is the first step of the image processing process. The inital picture is turned to a gray image and then a threshold is used, to be able to detect contours by using the OpenCV 'findContours' method. This method returns an list of detected contours. To be able to filter out inappropriate contours like the contour of the board, or the contours of the puzzlepieces, we process the image so that everything except the inside of the board is blacked out. This way we are able to end up with a list of the contours of the slots.
3. Finding the puzzlepiece handles: The coordinate that has to be detected for picking the puzzlepiece is the exact coordinate of the handle. Each puzzlepiece is cut out using the contour so that it is possible to only process each single puzzlepiece. The surface of each handle is covered by a black circle which is centered. This makes it possible to process the cut out puzzlepiece in a way the only the contour of this circle is detected.
4. Calculating parallax effect: By calculating the center of the contours, a coordinate for further processing has been calculated. This coordinate is not accurate though. Because of the height and the projection of the handles to a two dimensional images, the parallax error has to be taken into consideration. The further the piece is away from the center of the image, the bigger is the deviation to the correct center. By using the intercept theorems, the coordinate can be adjusted. The following images shows the deviation of the parallax error. The red point show the coordinate without considering the parallax error. The green point shows the correct coordinate.
5. Calculation of the angle: The puzzlepieces will be most likely not aligned to the board of the puzzle. It is therefore necessary to detect the relative angle between the slot and the according puzzlepiece, to be able to rotate the puzzlepiece before dropping it into the slot. OpenCV provides the method 'minAreaRect', which can create the minimum rectangle around a contour. By applying this method to the contours of the puzzlepieces and the slots it is possible to calculate the angle between the baseline segments of the minimum rectangles.
6. The calculated values for handle center, slot center and the angle are used by the kuka control after each pick and place of the robot. The coordinates of the image processing (pixels) are transformed to the coordinates of the robot (mm). The relation between the image processing coordinates and the robot coordinates has been figured out by taking and processing a picture of a big ruler. 1mm is equivalent to 2.3022 pixel

Controling the KUKA robot for solving the puzzle

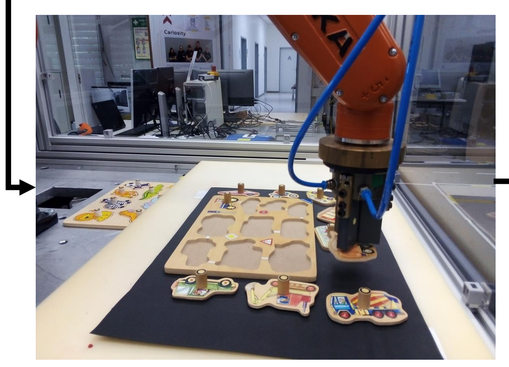
1. Go to resting position

2. Lower arm to travel-height



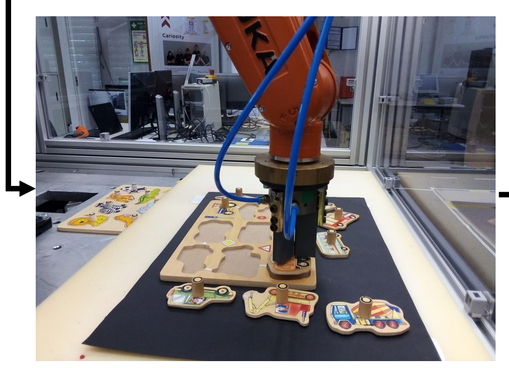
3. Go to puzzle piece

4. Lower arm for picking



5. Pick puzzle piece

6. Go to destination and align piece



7. Lower arm for placing

8. Drop piece into puzzle

9. Go back to travel height

Parameters of the KUKA

The KUKA KR 6 900 SIXX is a versatile 6-axes robot that offers precision and quick cycle times. It supports a payload of 6kg and is able to reach unto 901 mm. The 6-axes flexibility provides a rather fluid and agile movement of the robot arm. The articulated arm also offers particularly high working speeds in tasks including loading/unloading, packaging, and processing parts etc. The robot is paired with a hydraulically powered gripper as an effector for handling objects.

The Kuka is supplemented by a hot-pluggable smartPAD which functions as the smart Human Machine interface (HMI). It contains safety switches and responsive jog keys that give operator control the arm. It also provides an 8.4" capacitive touch display that also supports debugging mode. For the scope of this project, this HMI was used to mirror and operate via VNCViewer.
For safety reasons it is recommended to define a workspace, in which the robot is allowed to operate. If the robot is out of this workspace it will automatically stop its execution. In this case it is possible to drive the robot back into the allowed area only manually. This workspace is defined by the following parameters (see table "Workspace definition")
This area is completely inside of the operational box around the KUKA (see figure top left). For all following movements it is not able to move the robot out of this area. As a consequence we saved not only the robot itself, but also the surrounding box from damage. The next step is to define the tool that is attached to the robot. This tool moves the tool-center-point away from the robot. Therefore this has to be considered separately. The attached tool is a gripper operated by air pressure. It has the following properties.
The next step is to define a working base. This enables the robot to use the same coordinate system as the image processing. For this reason we implemented a work base with the following properties

Origin	Value	Delta	Value
X	424	X1	340
Y	-275	X2	0
Z	55	Y1	480
A	0	Y2	0
B	0	Z1	500
C	0	Z2	0

Workspace definition

Direction	Value
X	429.84
Y	-258.22
Z	60
A	90
B	0
C	180

Base definition

Direction	Value
X	0
Y	0
Z	135.3
A	0
B	0
C	0

Gripper tool definition