UNIVERSITY OF APPLIED SCIENCES

FACHHOCHSCHULE WEDEL

MASTER IT-ENGINEERING

ROBOTICS

Car puzzle

KUKA finds a match

Jakob Brüst (ite103511)   *Piet Gömpel* (ite102987)   *Niels Röschmann* (ite103747)   *Anurag Tyagi* (ite103217)

supervised by
Prof. Dr. Ulrich HOFFMANN and Hermann HÖHNE

submitted on
July 02, 2019

Acknowledgement

We would like to express our gratitude to Prof. Dr. Ulrich Hoffmann for helping us understand the fundamentals of robotics, their inner workings and applications. Also, we are grateful for the constant support and discussions carried out by him, that helped move the progress forward.

We would also like to thank Mr. Hermann Höhne for letting us access the laboratory, operate on the robot with no interruptions and explore the various aspects of its programming through a detailed introductory lesson. His suggestions were crucial in helping us maintain safety in the laboratory and motivated us to find more creative solutions to our problems.

The vast knowledge and effective communication from the two advisors helped to make this project a success within the limited time frame.

Contents

List of Figures

List of Tables

1 Initial Project proposal

Document:	MITE_Robotics_FinalReport.tex
Version:	3.0
Date:	2019–07–03
Status:	Document State: final
Group:	Brüst, Jakob - Gömpel, Piet - Tyagi, Anurag - Röschmann, Niels

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 to be able to place them to the correct places by using the Kuka robot.

1.1 Goals

The goal of this group project is to enable an industrial robot to solve a car puzzle intelligently. The project will finish with the final presentation at calendar week 28. To achieve the project goal the required tasks are split in multiple sub tasks. One of the two main tasks will be to detect the puzzle objects by an image processing software and the second main task will be to align the robot arm according to the coordinates of the puzzle pieces and to grab and place the objects into the correct cardboard positions/slots.
The secondary goals of the project will be to make the movement as dynamic/fluent as possible and to be able to adapt the software and the robot to different situations like changing the puzzle object or changing positions in real time.

1.2 Current Situation

For the implementation of the project, an industrial robot from Kuka with an air pressure operated gripper is used. The robot will interact with a camera of the Type uEye UI-1540 at a certain height of 1.475 meter above the workplace. By connecting the robot and the camera through computer vision capabilities we can improve and extend the applications of the Kuka Robot and solve the given task.

1.3 Project Approach

The project approach is to distribute the tasks into two main tasks. Each of the tasks will be worked upon by two of the four group members. The following is a short abstract of the two main tasks with the corresponding subtasks:

- Movement of the Kuka Robot according to the coordinates with the subtasks. Responsible for this tasks: Brüst, Jakob and Röschmann, Niels
  - Finding the initial origins
  - Movement of the robot to the grip holder
  - Distance calculation of the objects to the cardboard
  - Alignments and transformations for picking up and dropping the puzzle pieces
  - Machine code for the Kuka robot
  - Development of the gripper control
- Camera Vision to detect the edges from the pictures to export the important data for the robot movement. Responsible for this task: Gömpel, Piet and Tyagi, Anurag
  - Detection of the cardboard
  - Detection of all contours of puzzlepieces and slots
  - Determination of the related coordinates
  - Determination of the related angles
  - Find matches between slots and puzzle pieces

By implementing and demonstrating this project on a smaller scale, it can be used in industries in the future on much bigger scales e.g. container docks or different use cases in industrial production chains. By a dynamic implementation of the project, the combination of the robot and computer vision can be used in many situations with different conditions and various objects.
To successfully implement the project some requirements are needed. To achieve the project goals a reliable and dynamic software for image processing and for controlling the robot is required. Our group chose the programming language Python as the main software tool for mostly all software parts of the project. Python and OpenCV will be used for image processing and contour detection. Further Python is used to remote control the robot with the python entry hack and it will be the interface between the image processing and controlling of the robot.

1.4 Deliverables

Another project requirement will be the project documentation. The documentation will contain a broad overview about the project planning and the initial documentation. Furthermore it will contain a detailed description about the project and the development process timeline. An important part will be the report of the software implementation. The report will also contain the inline documentation of the code and the user manual for the car puzzle project. The end of the report will include the concluded results, the insights the group members acquired during the project and a demonstration accompanied by a video footage.

1.5 Material and Costs

The components required for the car puzzle can be subdivided into hard- and software parts. The software side factors are:

- Python3
- OpenCV Library
- Kuka Software IDE

The hardware components for the task are:

- Kuka Robot
- Camera
- Puzzle
- Black backgorund including underlaying foam
- Kuka Human Interface Device
- Optional:
  - LED spotlight
  - Additional Kuka pressure pipes

Future or additional costs are not need for any further purchase.

1.6 Timing

The Timing schedule dictates that the group will meet once a week to discuss the current state of the Project and the corresponding subtasks via Scrum meetings. One of the main topics in the meetings will be the discussion about the difficulties that appear. Furthermore, there will be a comparison between the actual and the target performance and also the next steps the group wants to take.

Schedule

Both tasks, the Kuka controlling and image processing work in parallel for each milestone. The specific division into subtasks will be decided at the beginning of each milestone. In general the focus is to work especially for the Kuka movement as close as possible. On the one hand this is due to the reason that there is only one robot at the university where both group members have to work on. On the other hand this may harm the robot or puzzle. For better control we have decided to work always together.
Also for the image processing it is helpful working close together. The resulting code is better readable and easy to understand if both group members working on it.
In the final report we will unambiguously mark the specific tasks each group member has worked on.

Table 1.1: Initial schedule containing all milestones

Week (CW)	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Milestone 1	Start													
Milestone 2				Basic										
Milestone 3							Test							
Milestone 4										Prototype				
Error fixing														
Milestone 5														Pres.

1.6.1 Milestones

Milestone 1

Name:	Initial project presentation
Due Date:	2019-04-30
Accomplished Goals:	Hand in initial project describtion and presentation
Acceptance Criteria:	Project proposal passed Prof. Hoffmanns criteria

Milestone 2

Name:	Define basic principles
Due Date:	2019-05-14
Accomplished Goals:	Define coordinate system Via openCV it is possible to detect and differentiate the puzzle pieces and destinations. Kuka robot operation space is defined and can't violated, controlled movements via "python interface"
Acceptance Criteria:	All goal should be archived to the end of week 20

Milestone 3

Name:	Implement and test basic principles
Due Date:	2019-06-04
Accomplished Goals:	Image processing is able to calculate the coordinate of the puzzle piece handles Kuka robot is able to pick rotate and place a puzzle piece
Acceptance Criteria:	All goal should be archived to the end of week 23

Milestone 4

Name:	Working prototype
Due Date:	2019-06-25
Accomplished Goals:	Marriage of image processing and robot control software
Acceptance Criteria:	All goal should be archived to the end of week 26

Milestone 5

Name:	Final presentation
Due Date:	2019-07-09
Accomplished Goals:	Final documentation and presentation
Acceptance Criteria:	Finished in time and with the acceptance of Mr. Höhne and Prof. Hoffmann

1.7 Risks

One of the main risks in this group project could be a superficial consideration while dividing the project into subtasks. It may be more effort needed for a subtask then in this stage excepted - this would greatly delay the project. Another risk could be that focus will be shifted to secondary goals that are not needed to solve the car puzzle. This could eventually, but not necessarily, lead to a delay. Furthermore a (temporary) absence of a group member could also influence the project schedule.

2 Final Report

Robotics deals with the construction, design, operation, and use of robots for moving large objects, image processing, and exploring areas that are otherwise unreachable. Furthermore, robots paired with Computer Systems are being incorporated in all fields of technology for carrying out tasks efficiently, quickly and with precision.

These robots have been deployed in numerous areas including assembly lines (manufacturing processes), bomb detection and deactivation, and places impossible for humans to access (radiation zones, deep sea, etc.).

This project focuses on assembly line robots that play a crucial role in the manufacturing process, more specifically an Articulated Arm Robot, the KUKA KR 6 900 SIXX (nicknamed KR AGILUS). These type of robots are used to move large payloads around the assembly line, perform quick and complex actions automatically and perform product inspection and packaging. Such robots accompanied by sensors open up a wider range of applications and programmability. These sensors collect data and help the robot understand the environment and adapt to changes in it. Most common of these sensors include cameras, temperature or light detectors.

Additionally, the setup includes actors (motors, servos, hydraulic pumps, etc.) and effectors (grippers, probes, soldering equipment, etc.) in tandem to perform the desired tasks. The sensors provide environmental data to the computing system, which in turns activates and moves the actors and/or effectors.

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.

2.1 Hardware

2.1.1 KUKA

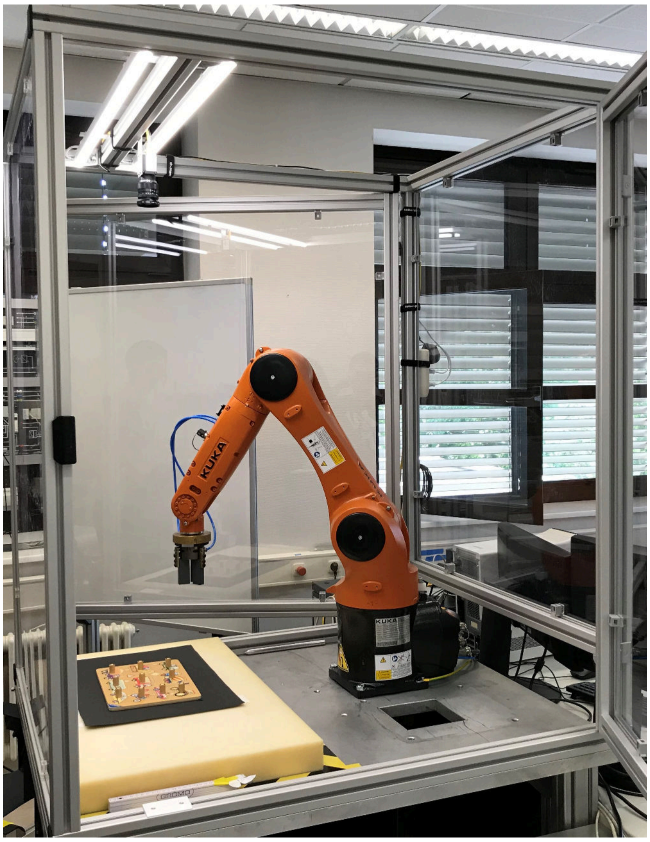

Figure 2.1: Laboratory with KUKA robot

KUKA robot

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 901mm. 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.

KUKA smartPAD

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 operators control to 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 operated remote by VNCViewer.

2.1.2 uEye camera

A wide-lens high resolution camera to take pictures with sufficient clarity and light.

2.1.3 Car puzzle

To carry out the sorting and placement task, a generic wood-carved puzzle that includes a main board with distinctly defined slots was used. It also comes with a number of puzzle pieces that fit in their respective slots. The pieces are attached to handles that allow the arm to pick up the pieces with the gripper.

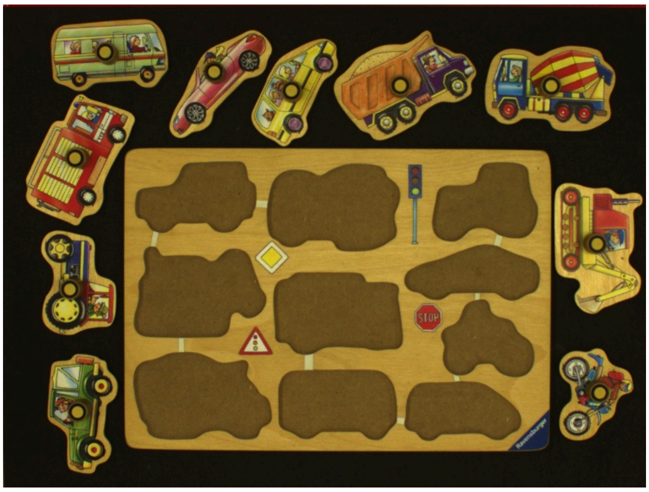

Figure 2.2: Picture from uEye camera showing the Car Puzzle

2.2 Software

2.2.1 Computer vision

Computer Vision enables the robot to “see” details about the environment. This is done by accumulating images from the camera and feeding them to the computing system for extraction of knowledge. The computing system, with the help of a computer vision software, does the calculations by manipulating and processing the image. In this instance, OpenCV is used to perform these tasks and calculate coordinates for the movement of the robot arm.

Image processing using openCV

A picture is taken every time the robot returns to its resting position. The process of image processing with OpenCV is executed every time after a puzzlepiece was picked and placed. This way it is possible to make changes to the puzzle while the execution of the program. It makes it possible to remove or add puzzlepieces during execution. The robot will pick and place puzzlepieces until the puzzle is solved completely. The detailed process of image processing in steps is explained in the following sections.

1. Taking a picture: The process begins with the camera taking a picture of the puzzle with the arm at resting position. This is done everytime before the robot picks and places a puzzlepiece. The image is processed through the OpenCV library.

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.

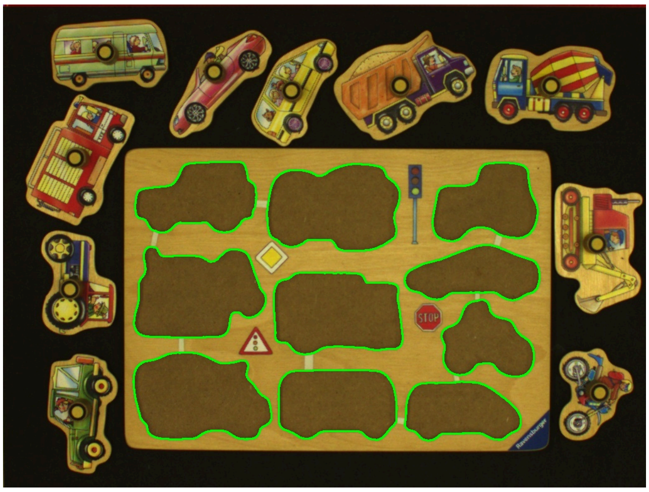

Figure 2.3: Contours of the slots

3. Detection of piece contours: This step is done quite similar to the detection of slot contours. The image is grayed and we apply a threshold to be able to detect contours. The 'findContours' method is used in a way so that it only detects external contours, so that the slot contours are not detected and the board contour is filtered out by checking for contour length and area.

4. Finding matches: The next step is to find the matches between puzzlepieces and slots. OpenCV provides the method 'matchShapes', which is very suitable for this case. 'matchShapes' returns a value that is an indicator for similarity. The lower the value, the better the match is. This way it is possible to compare every puzzlepiece and slot contour with each other and determine the appropriate match.

5. 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.

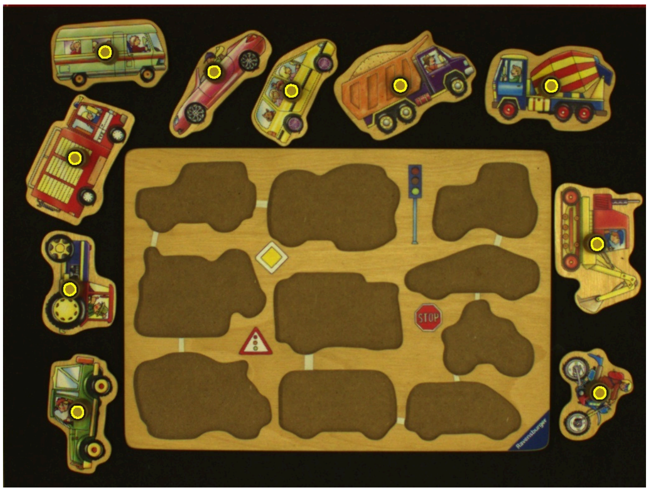

Figure 2.4: Contours of the handle circles

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.

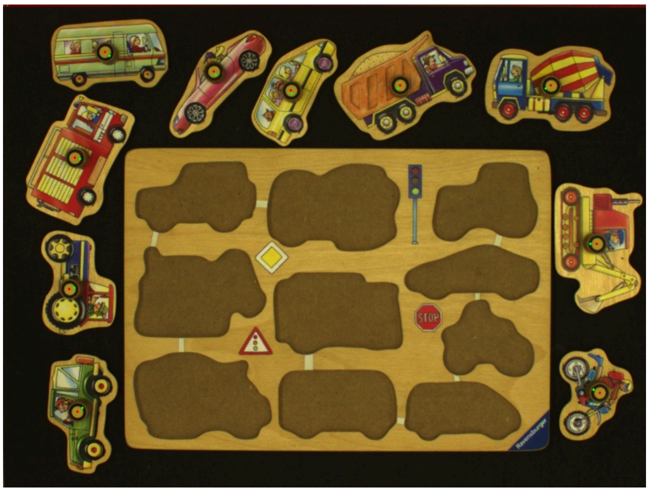

Figure 2.5: Parallax deviation

6. 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.

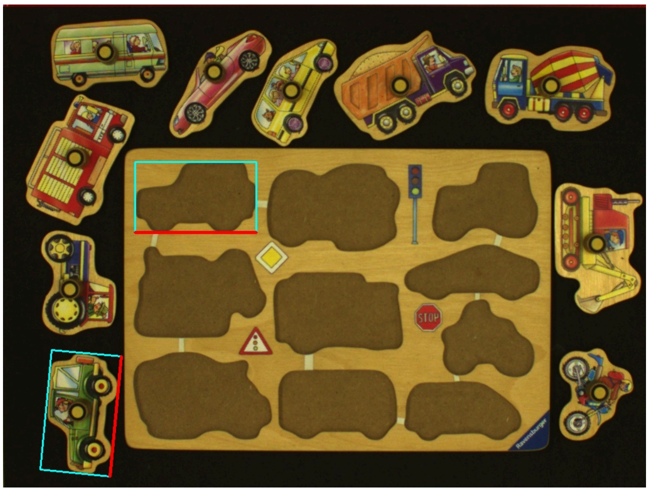

Figure 2.6: Determination of relative angle between piece and slot

The approach to use the minimum rectangles turned out to be really reliable, but it is possible that the resulting angle is flipped by 180 degrees. To determine if the resulting angle is correct or not we used the following approach: We overlap the contour of the slot and the contour of puzzlepiece rotated by the calculated angle and rotated by the calculated angle adding 180 degrees. For both overlapping images, we determine the external contour and we compare the area of this overlapping contour. The contour with the smaller area is the one with the correct angle. The following images visualizes this approach.

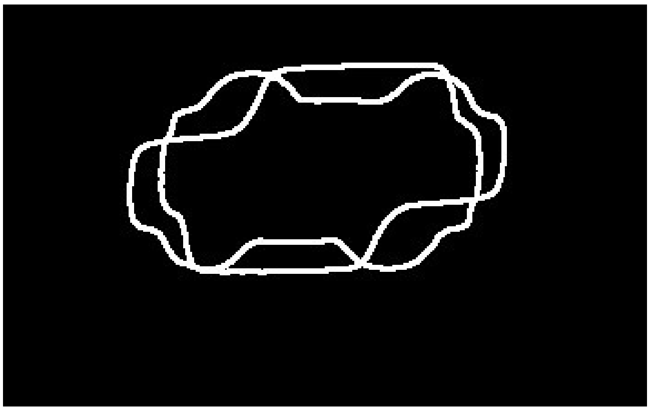

Figure 2.7: Overlay image of flipped angle by 180 degrees

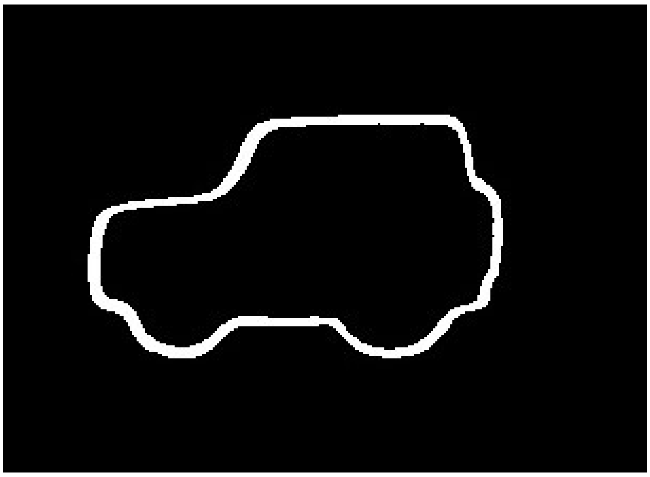

Figure 2.8: Overlay image of correct angle

7. Finding the slot center: The second important coordinate to detect is the coordinate of the slot, where the robot should release the puzzlepiece. It should be the same point like the handle of the puzzlepiece, relative to its contours. Using this approach we calculate the vector between the center of the puzzlepiece contour and the correct center of the handle. We can then apply this vector to the center of the slot contour (which is the same as the piece contour) to determine the correct coordinate of the slot.

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.

$$1\,\mathrm{mm} = 2.3022\,\mathrm{p}ixel \tag{2.1}$$

2.2.2 KUKA controlling

To be able to solve the puzzle, the pieces have to be moved into the correct spot. Where to find these pieces and slots is the task of the above described image processing. The actual movement of the pieces is done by a KUKA Robot (see figure 2.1). How this is done is described in the following.

KUKA safety configurations

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:

Table 2.1: KUKA Workspace parameter

Origin	Value	Distance to origin	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

This area is completely inside of the operational box around the KUKA (see figure 2.1). 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:

Table 2.2: KUKA gripper tool parameter

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

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:

Table 2.3: KUKA base parameter

Position	Value
X	429.837
Y	-258.221
Z	60
A	90
B	0
C	180

Due to some inaccuracies during the measurement this base is unfortunately not exactly the same as the coordinate system from the image processing. To fix this problem there is a fixed offset added to the coordinates provided by the image processing. With this configurations the KUKA is ready to be used for solving the puzzle. The next steps are the software based automated execution of movements that are described in the next part.

KUKA remote controlling

To be able to control the KUKA from a remote source, there has to be a special program running on the KUKA. This program is a simple infinity loop, which checks a status variable in each iteration. Is this variable set to a different value, the KUKA starts to operate this command. The value of this status variable can be changed during runtime from an other computer. Via this "back door" it is possible to control the KUKA remotely. The KUKA can execute the following commands:

Table 2.4: KUKA variable and according command

Value of status variable	**Command**
COM_ACTION = 1	Do nothing
COM_ACTION = 11	Do a point-to-point movement
COM_ACTION = 12	Do a linear movement
COM_ACTION = 13	Open or close the gripper, using green and read air-pressure valve

For executing COM_ACTION = 13 the state of the gripper has to be set before. To open the gripper the red valve has to be closed and green valve opened. For closing the gripper, it is correspondingly the other way round.

All the remote controlling computer has to do for moving the KUKA is to define a destination coordinate and set the status variable to the according value. This is done by the class "PuzzleSolver" that is a wrapper class between our project and the provided "RemoteControlKUKA" class. RemoteControlKUKA class is setting values within the "openshowvar" module. This module is actually changing the values of the KUKA variables during runtime. The class PuzzleSolver provides the following methods:

Table 2.5: Methods from python to control KUKA at runtime

Method	**Description**
go2Origin()	Move the KUKA back into the origin/resting position

shake_X()
shake_O()
	to improve the result, these methods will shake the tool-center-point after placing a piece into the puzzle in X direction or in O direction (further description see point 11 in 2.2.2)

pick()	Move KUKA to a coordinate and pick the puzzle
place()	Move the KUKA to the destination and align the puzzle piece to the slot
draw_Points4Offset()	For calculating the offset this method draws point on a sheet of paper inside the workspace. This points can be evaluated by the image processing

We decided to use only linear movements to move the KUKA from one to another position, because this movement is predictable. Point-to-point movements might be a bit faster, but it could happen that the robot is leaving the save workspace. Due to the fact that the solving of a puzzle is not a time critical operation, we only use linear movements. Using the above described methods the KUKA will solve the puzzle in the following way:

1. Go to resting position. This position is far above the surface, so that it not disturbs the image taken by the camera. Has the KUKA arrived this position, the image processing is doing its work (see 2.2.1)

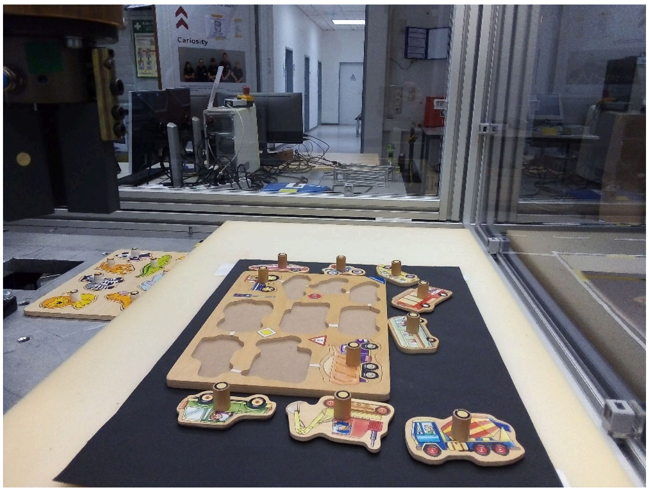

Figure 2.9: KUKA in resting position

2. KUKA goes down to a height, where it does not touch anything, the so called "traveling-height"

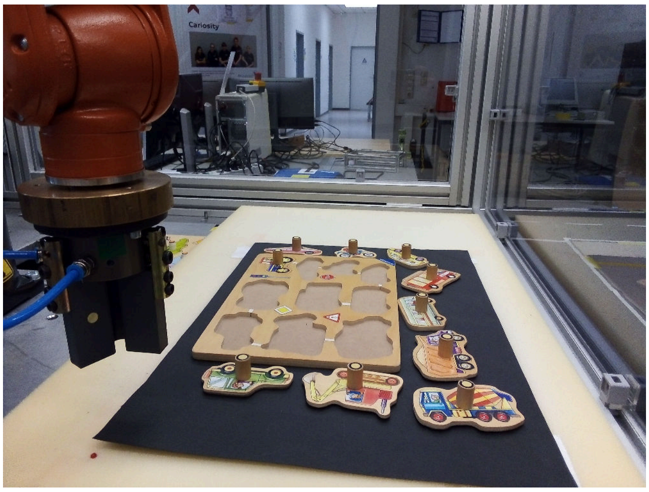

Figure 2.10: KUKA at travel-height

3. On this level it moves to the coordinate of the puzzle piece that is at first detected by the image processing

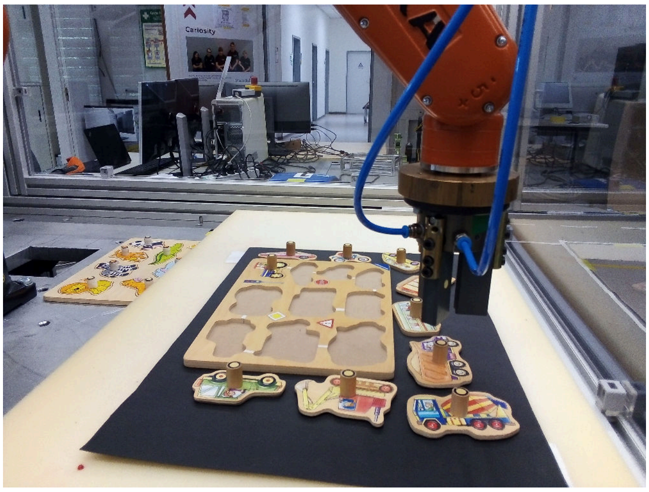

Figure 2.11: KUKA above piece

4. Is this position reached, the gripper is opened
5. The KUKA goes down to grap the piece and closes the gripper

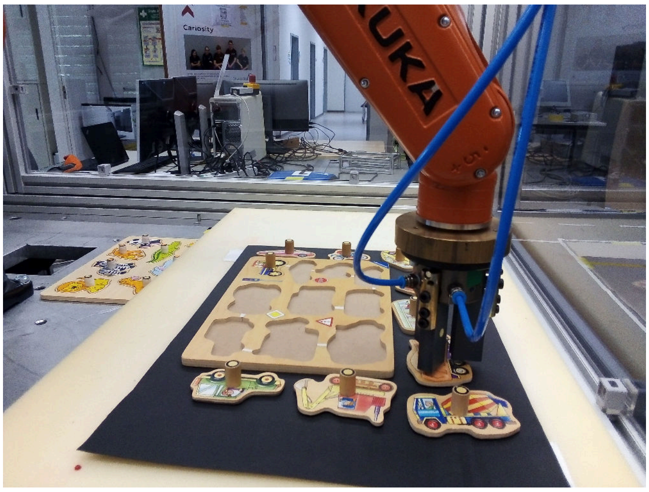

Figure 2.12: KUKA picks a puzzle piece

6. After closing the gripper the KUKA moves back to the "traveling-height"

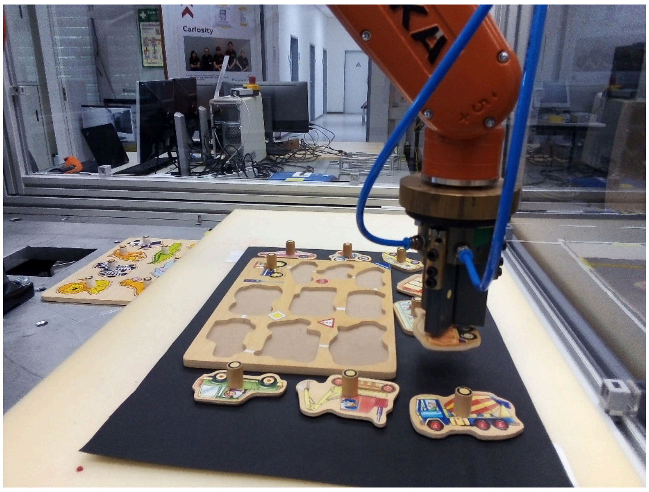

Figure 2.13: KUKA has picked a piece and moved up

7. Now it is moving towards the destination of the respective piece
8. While moving, the piece is aligned to the destination, so that it fits into the slot

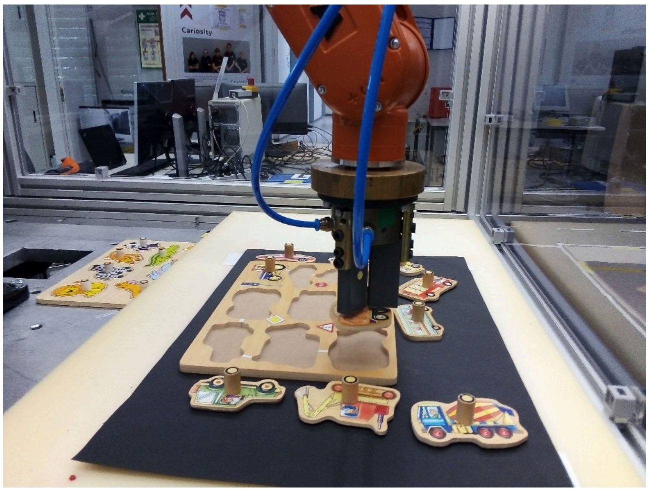

Figure 2.14: KUKA in placing position

9. When the KUKA arrives at the destination, it goes down to place the piece inside of the slot

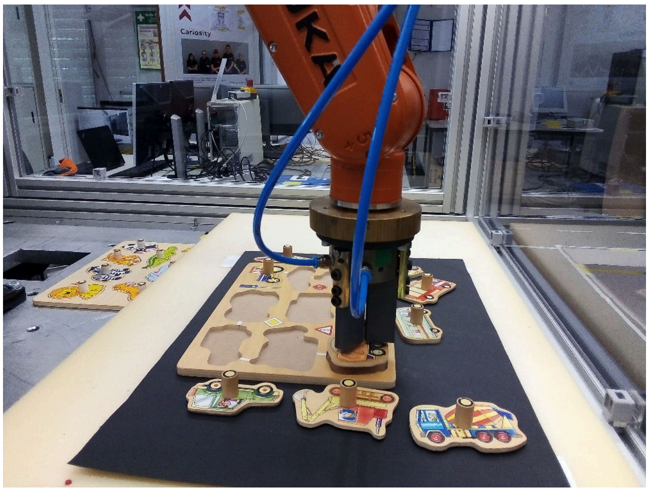

Figure 2.15: KUKA at place height

10. Once the KUKA has reached the lower level it opens the gripper and the puzzle piece falls into the slot

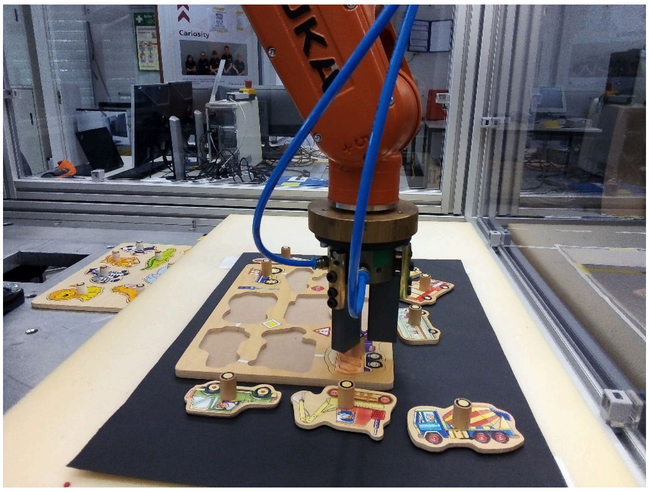

Figure 2.16: KUKA placed a puzzle piece

11. OPTIONAL: Now a shake can be executed. This means, that the KUKA moves slightly around the coordinates of the destination to make sure that the piece has really placed correctly
12. Afterwards the KUKA goes straight up to "traveling-height", so that it doesn't touch other pieces

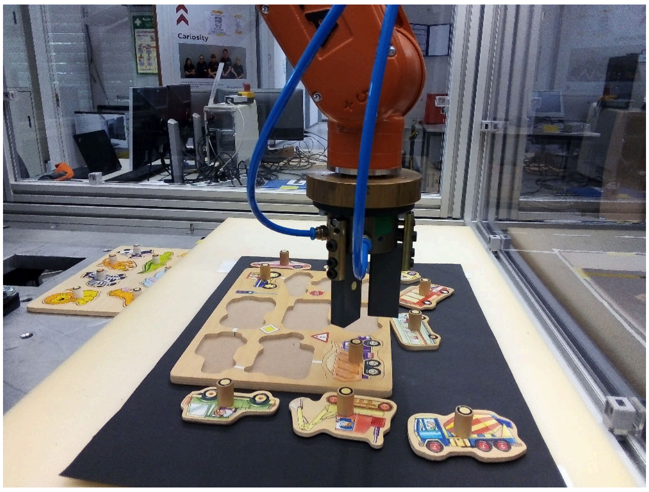

Figure 2.17: KUKA again at travel-height

13. The KUKA goes back into the resting position (see figure 2.9) and an other cycle of picking and placing is started

14. If all pieces are within the puzzle, or the image processing is not executing correctly, the KUKA will stay in the resting position

This is a dynamic approach, so every time the KUKA reaches the resting position (see figure 2.9) a new picture is taken and a new image processing will provide new coordinates and angles. Therefore this allows the user to replace the pieces while the KUKA is operation. Be aware that the pieces have to be completely captured by the camera. Otherwise the KUKA will not pick these pieces (see chapter 2.2.1). Also should the user be cautious because he is entering the workspace of the robot.
If the user trys to replace a piece while the image processing is taking a new picture, this will lead to faulty results. The KUKA waits in the resting position until the image processing has got a valid picture again. Afterwards the normal procedure starts again.

2.3 User manual

The following steps will provide a manual how to use the KUKA to solve the car puzzle using the above described approach:

1. Mount the air pressure gripper on top of the KUKA

2. Start the KUKA robot and operational computer (and connect the computer to the robotic lab network)

3. Switch on all lights in the room and close the curtains

4. Open the central valve for air pressure

5. Deploy the Carpuzzle_withWS.wvs from the KUKA operational PC to the robot

6. Start the program CARPUZZLE_KLR.src using the HMI of the KUKA robot in automatic mode

7. RECOMMENED: Start the ueye_camera.py program to see if all pieces are completely captured by the camera. Also check if the foam is completely covered by the black cardboard. Close uieye_camera.py afterwards

8. Start main.py from the module "KUKA car puzzle"

9. The KUKA starts with solving the puzzle, like described above

10. If all pieces are within the puzzle, the program stops executing

11. After running the program and no further actions, be aware to remove the gripper from the KUKA, shut down the robot, operational computer and all lights

2.4 Observations

This chapter deals with the problem we had to face during this project and also the solutions we found to overcome those.

Project successes

Table 2.6: Observations

Objective	Description
Found perfect matches every-time	Through image transformation and contour detections, matching slot for every piece was found
Smooth Movement of arm	The arm moved and rotated with precision and sufficient speeds
Gripper control	The gripper opened and closed at the desired times
Defined workspace and base.	Workspace was limited to the visible frame of the camera, with one corner used as the base position
Software Coupling	Both the KUKA motion part and the Computer Vision part were integrated successfully
Dynamic functionality	Robot reacts to change in pieces and continues solving

Unexpected Events

Table 2.7: Unexpected events

Description	Impact	Actions taken
The base of the arm had to be defined	Wrong coordinate transformation and origin	Base was set to top corner of set workspace, within the camera view
Rotating a Piece	Pieces were not turned correctly for placement	Pieces did not always fit target slots
Parallax caused the calculated centers of the handles to be quite off from the actual centers	Pieces were not perfectly grabbed and picked up	Adjustment in coordinate system for parallax effect
Contour and shape detection	Pieces and slots were sometimes not detected	Thresholding was calibrated

Lessons Learned

Table 2.8: Lessons learned

Description	Recommendation
Hard to read KUKA manuals	Refer to official webpages/forums
Learning by doing	Preferred to have some prior experience or external help
Not always simple and quick to work in parallel (scheduling issues)	Need to have clearly defined time plans and goals, but prepare for uncertainties

2.5 Conclusion

The project demonstrated the efficiency and ease with which robots perform their programmed tasks. These tasks could range from sorting and placing to carrying huge loads. Through this project, it can be seen how powerful computer vision or machine learning tools could prove to be on large scale and real world problems/use cases.
We also proved, that our preparation has paid off. We have finished each milestone in time, and we were able to finish the project and the to reach the project goals as planned.

2.6 Scope for future improvements

In terms of detection of a certain piece, different methods could be employed than contour detection. A few examples include detecting various parts/entities off an image through color

or pattern classification. Computer vision could also be used to find complex shapes in the image.
Moreover, parameters could be made more dynamic to work with different puzzles and different pieces. For reasons of time we didnt have the chance to test another puzzle and to increase the abstraction of the implementation. Still we think that those adjustments can be done in a short time. For the movement part, the arm could be programmed to move point-to-point instead of linearly to mimic a more free motion.