Trixi the Librarian

Fabian Wieczorek
dept. informatik, TAMS
University of Hamburg
Hamburg, Germany
fabian.wieczorek@uni-hamburg.de

Shang-Ching Liu
dept. informatik, TAMS
University of Hamburg
Hamburg, Germany
shang-ching.liu@studium.uni-hamburg.de

Björn Sygo
dept. informatik, TAMS
University of Hamburg
Hamburg, Germany
bjoern.sygo@uni-hamburg.de

Mykhailo Koshil
dept. informatik, TAMS
University of Hamburg
Hamburg, Germany
mykhailo.koshil@studium.uni-hamburg.de

***Abstract*—(Fabian) In this work, we present a three-part system that automatically sorts books on a shelf using the PR-2 platform. The paper describes a methodology to sufficiently detect and recognize books using a multistep vision pipeline based on deep learning models as well as conventional computer vision. Furthermore, the difficulties of relocating books using a bi-manual robot along with solutions based on MoveIt and BioIK are being addressed. Experiments show that the performance is overall good enough to repeatedly sort three books on a shelf. Nevertheless, further improvements are being discussed, potentially leading to a more robust book recognition and more versatile manipulation techniques.**

***Index Terms*—Librarian, Robot, Book grasping, Object detection, Implementation**

I. Introduction (Mykhailo)

While the use of industrial robots is already widespread, the use of service robots remains limited [32]. There are a few reasons for this. First, implementing the use of the service robot in the business requires not only the acquisition of the robot itself but also leads to changes in the organization like training the personnel, adapting the environment for the robot, etc. Second, while the industrial robot is used in a highly structured environment, the use case of the service robot implies working alongside humans, thus having less structure and more unseen situations. This leads to the high complexity of design, and as a result service robots often fail and require human intervention, which puts their commercial use in question.

Therefore the main purpose of this work is rather showcasing and testing the feasibility of automating the work of a librarian on the task of manipulating the books on the shelf, rather than creating a commercially viable product. As the original plan included interaction with the visitors, this puts our system in the category of 'Professional Social Service Robots' according to the [16]. In this work, we aim at creating a service robot that will work in the library, so it can be deployed on the site without major changes to accommodate the robot.

The project is based on the PR-2 platform that is available in our department Fig. 2, and which is suitable for bi-manual manipulation. To tackle the task of book manipulation was divided into two sub-tasks: manipulation and perception. And while both of these tasks were solved to some extent, the main contribution of this work is combining them in form of a librarian robot that can operate in the library environment with as few modifications as possible.

II. Related Works

A. Book manipulation (Björn / Mykhailo)

There already exist different concepts on librarian robots. In example UJI librarian robot [26], [25]. It utilizes a single Mitsubishi PA-10 arm mounted on a mobile base. It can move around the library, locate the wanted shelf and book, and retrieve it using a specially customized two-finger gripper. Also, experiments have been made using the UJI robot equipped with a three-finger gripper to grasp books, using tactile sensors [18]. The motion planning for the book tilting is investigated further in [24]. Here, the authors develop a probabilistic motion planning algorithm that allows for planning in low-dimensional sub-manifolds, created by the constraints in the planning space. The developed planner was then tested in a scenario similar to ours and using MoveIt [5] framework.

A lot of approaches rely on environment modification in order to facilitate the robot's functioning, mostly to solve the navigation and object detection. For example, a robot that was designed to work in a highly structured environment, where it would pick and arrange books [15]. It utilizes a two-finger gripper to pick the books. The robot relies heavily on landmarks in the environment, utilizing a floor with radio-frequency identification (RFID) tags to navigate and an intelligent bookshelf to locate the books. Recent work [37] explores the librarian scenario using the robot similar to [25], but focuses on navigation and position using QR code and binocular vision, rather than manipulation. So the book manipulation is done by using a special parallel gripper and a predefined position for placing.

Some works focus primarily on manipulation. In [19], the authors investigate the use of a bi-manual setup with a suction gripper, in a setting similar to ours and train the fully connected network to predict which object to support with a non-suction gripper for the safe extraction of the selected object. The network is trained to perform in an environment similar to ours, i.e. bookshelf. Other works solve the book grasping outside of the library environment. One example would be [20], which utilized a combination between suction and a two-finger gripper for grasping books in different configurations.

There is no readily available solution to our knowledge, that would automate book manipulation in the library environment similar to ours.

B. Perception (Shang-Ching Liu / Fabian)

In the perception related tasks, we try to model the real-time books in the scene and furthermore matching individual book to the known book database. Thus, we dig into each part for previous achievement.

For detection method like YOLO [27] or Fast r-cnn [9] are two main method directions in state-of-art, the YOLO is more efficient with bounding box output and Fast r-cnn gives precise segmentation result. The evolution model of YOLO — YOLO-v5 [14] have well documentation and robust pipeline and utilities such as Roboflow [28] to proceed fine-tuning technique to extract the book spine, which we choose as the approach for book spine detection.

For book matching there are SIFT [23] to find the key-point of the picture, HSV (for hue, saturation, value) [36] histogram to understand the color encoding, fuzzywoozy to measure the text similarity between detection text with book title in database.

Inventory Management in a library is a tedious task that has been tried to automize in the past decade. Book spines standing on the shelf were attempted to be detected and recognized using different computer vision methods without the aid of special markers. A frequently seen approach to detect book spines is to use edge detection along with further line segment processing [3], [6], [22], [34]. Often, an orthographic representation of the book spines is required. To detect the spines independently from the viewpoint, Talker et al. used a constrained active contour model allowing the spines to be non-parallel to the image axis [35].

The detection part is crucial to find book spine candidates, but recognizing them correctly plays an evenly important role in inventory management. While many approaches focus on text recognition to identify the book spines [3], [6], [22], [34], Fowers et al. made use of *difference of gaussian* (DOG) over the YCbCr color space to extract features [7]. In combination with SIFT, this approach does not depend on an OCR engine (e.g. Tesseract [31]) while yielding robust performance.

Comparing the results of the mentioned work meaningfully is hard as no standardized way in the field of book spine recognition exists. However, in the domain of scene text recognition (STR), which can be utilized for text-based book spine recognition, a framework was developed by Baek et al. to allow comparison of different model architectures [1]. Since deep learning methods have generally not been widely applied in book spine recognition, this work tries to incorporate such an STR model to perform text matching.

III. System Overview (Shang-Ching Liu)

The central system can be separated into three parts, including Manipulation, vision pipeline, and task planning, as shown in figure 1. Task planning module controlling vision pipeline module and Manipulation module. The Vision pipeline takes the RGBD camera (Azure Kinect) scene as input and matches the books in the scene to the books database, and finally creates a MoveIt Planning Scene in visualization. The Manipulation has a controller to control two hands of the Robot (PR-2). One is combined with Shadow hands, and another is combined with a two fingers gripper, as shown in the figure 2j.

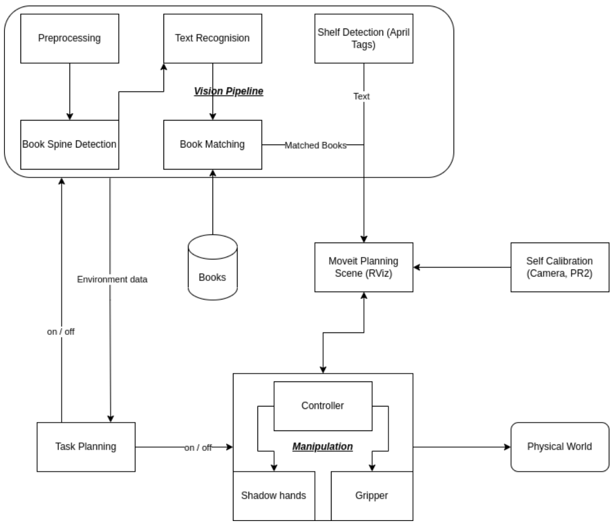


Fig. 1: System Overview

A. Setup

The robot used for this project was a Willow Garage's PR2, which was partly customized. The lower right arm and its gripper was replaced with a Shadow Dexterous Hand and on its head, a Microsoft Azure Kinect was mounted. Roughly 0.3 m in front of the robot, a shelf was placed. The shelf had two compartments and dimensions of 0.52 m (depth) x 0.797 m (width) x 1.25 m (height). The lower compartment had a size of 0.5 m x 0.765 m x 0.335 m and started at a height of 0.597 m while the upper compartment had a size of 0.5 m x 0.765 m x 0.287 m, starting at height 0.948 m. It has an April tag mounted at the top left corner, from the robots perspective, but no further markings.

IV. Perception (Fabian / Shang-Ching Liu)

A. Preprocessing (Fabian)

The camera of the PR-2 is located on top of its head. This will create a perspective projection of the shelf (fig. 3, left;

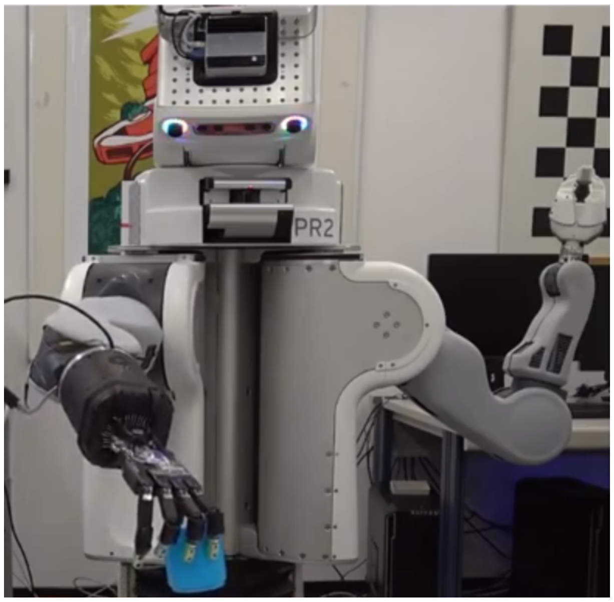


Fig. 2: PR-2 with shadow hands on the right-hand side and normal two finger grippers on the left-hand side.

fig. 5, right) making book spine detection more challenging since the edges tend to be not aligned with the image axis. To automatically mitigate this problem without rearranging the hardware, a perspective transformation is applied to the image twice as shown in fig. 3. For each shelf level, the corners (red/blue dots) are determined using the AprilTags known pose along with offsets matching the shelfs dimensions. The 3D points are then projected onto the image to be used as anchor points for the transformation. The results are two images with the book spine edges being aligned with the image axes (fig. 3, right).

To project back from the corrected images to the original image, the inverse projection matrix is also computed and will be used later on.

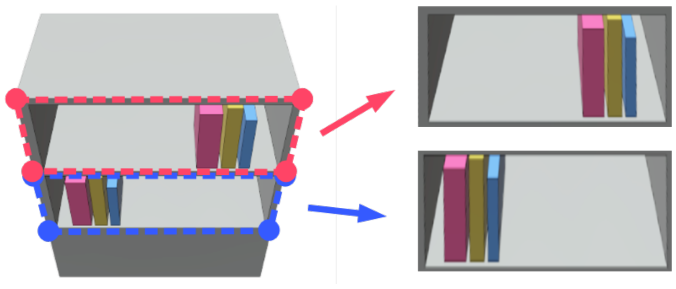
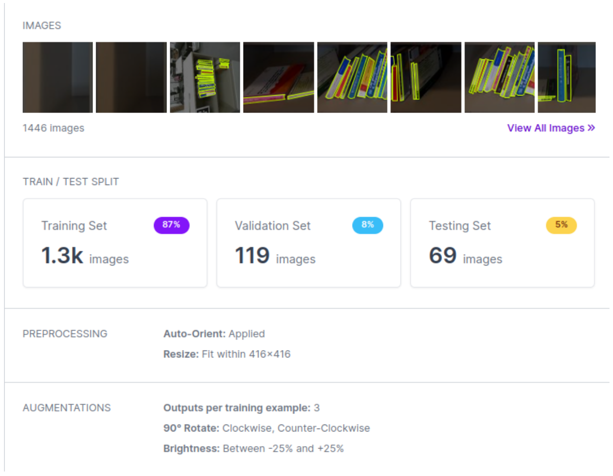


Fig. 3: Visualization of the perspective transformation. The corners of each shelf level (red/blue dots) are used as anchors for the transformation. The result is an image of each shelf level having the book spines aligned with the image axis (right).



IMAGES

1446 images

View All Images »

TRAIN / TEST SPLIT

Training Set 87%
1.3k images

Validation Set 8%
119 images

Testing Set 5%
69 images

PREPROCESSING

Auto-Orient: Applied

Resize: Fit within 416×416

AUGMENTATIONS

Outputs per training example: 3

90° Rotate: Clockwise, Counter-Clockwise

Brightness: Between -25% and +25%

Fig. 4: The book spine dataset, we have already split it to training, validation, and testing set. You can access by link [30]

.

B. Book Matching

The section below describing the approach we have used to tackle down the book matching tasks.

1) Book database (Shang-Ching Liu): We collect the following information from each books as shown in table I.

column	note
id	incremental number of unique id
title	book title
height	book height
width	book width
depth	book depth
author	book author
cover_type	hard/soft cover of book
count	Number of specific book

TABLE I: Books Database

2) Book Spine Recognition (Shang-Ching Liu): Turning now to book recognition. The book spine is intuitively the hint from the front side perspective. In the beginning, we tried object detection for the book, which is one of the initial labels in the COCO dataset [17]. However, it failed to recognize the book spine and had a critical issue for recognizing multiple books spine together as one book spine. So that we start to collect our own dataset for book spine data to fine-tuning the model. We have created the dataset [30] which labeled book spine for 611 images and done the data augmentation as describe in figure 4.

3) Text Detection (Fabian): In order to make use of the scene text recognition model proposed in [1], the image needs to be cropped so that only the text of interest is visible. To get the bounding boxes of each text segment, the CRAFT model [2] is being used. Running the model on the two preprocessed input images results in the desired bounding boxes for each detected text pieces in the form of polygons (see fig. 5, magenta boxes). However, the vertices of the polygons are still aligned with the transformed images and not the source image. To reverse the perspective transformation applied during preprocessing, each point is transformed using the inverse matrix (fig. 5).

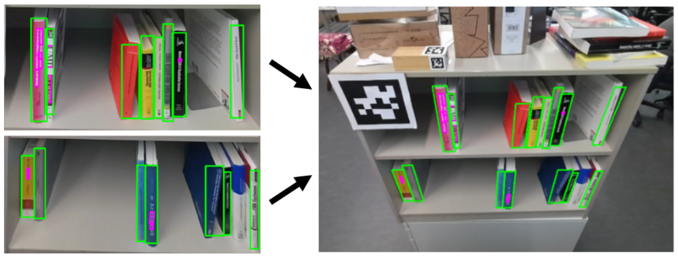


Fig. 5: Visualization of reversing the perspective transformation along with the bounding boxes of detected spines (green) and text (magenta). The vertices of the polygons were transformed back so they line up with the source image (right).

4) Text Recognition (Fabian / Shang-Ching Liu):

a) DTRB Model (Fabian): To recognize the text coming from the text detection, the proposed model from the deep-text-recognition-benchmark (DTRB) [1] is used. The authors compared different combinations of model components against each other and existing STR models. They showed that a combination of thin-plate splines (TPS) [13], ResNet [11], Bidirectional LSTM (BiLSTM) [10], and attention-based sequence prediction (ATTN) [4] shows the best performance while also taking the most time to compute. With this combination, each text detection is turned into actual text and can be used as a feature for the final classification.

b) Google Vision API (Shang-Ching Liu): In addition, we integrate Google Vision API to conquer the difficulty of text recognition issues due to size, light, font, or multi-language in the scene. The pros would be Track changes is off Everyone Track changes for everyone You Track changes for You fabian.wiecz Track changes for fabian.wiecz goerner Track changes for goerner 6sygo Track changes for 6sygo mishamail7 Track changes for mishamail7 Guests Track changes for guests Added iz panel as figure 13… (show all) Sep 30, 2022 10:07 PM • Yo•••••••••••u Current file Overview Trixi the Librarian Fabian Wieczorek dept. informatik, TAMS University of Hamburg Hamburg, Germany fabian.wieczorek@uni-hamburg.de generalized text recognition and able to generate quite a robust result. The drawback, in reality, is that most of the time, the camera still fails to give a clear enough image to do the text recognition. On the other hand, the Vision API from Google is limited to free usage 1000 times a month.

5) HSV histograms of book spines (Fabian): While text recognition can serve as a distinct feature for matching, it suffers as soon as the text becomes illegible e.g. due to being too small. To make the classification more robust, the histogram over the HSV color space is used as another feature similar to [7]. For each detected book spine, the histogram over each channel is computed with a resolution of 20 bins. Then, a score on how much a spine detection resembles to a book from the database is determined using the cosine similarity of both histograms.

6) SIFT Matching (Shang-Ching Liu): On the other hand, we try out the pure SIFT Matching for book matching. It works pretty well on some examples, as shown in figure 6, but some similar books are hard to tackle, shown in figure 7. Thus, we do not use SIFT as our implementation.

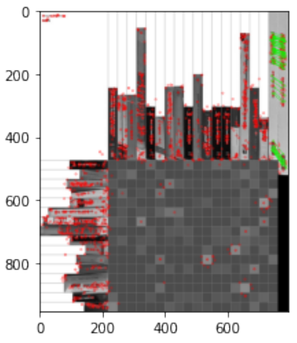


Fig. 6: Successful to match the book but not for the vertical one, on the left side is all the book spine in the databases in vertical and horizontal view.

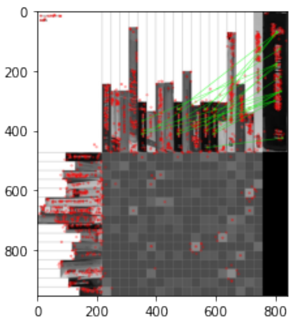


Fig. 7: Failed to match the book with multiple similar book in the database, on the left side is all the book spine in the databases in vertical and horizontal view.

7) Final Classification (Fabian): Since neither the text-based matching nor the SIFT matching showed sufficient performance, the final score is only computed using the HSV-histogram feature with the value-channel being ignored as well. The score was empirically defined as $x_{score} = sim_{hue} + sim_{sat} * 0.2$. Figure 8 shows each score from a set of detections against each book from the database.

To determine which detection is assigned which book ID from the database, algorithm 1 is being used with the score matrix as input.

This ensures that each detection is assigned an ID while each ID is only being assigned once. Furthermore, the score of each book associated with the assigned ID defines the confidence of this book recognition.

Algorithm 1 Assign book ID to each detection

```
while rows left do
    $row, col \leftarrow$ findHighestScore()
    assign ID from book at $col$ to detection at $row$
    removeRow($row$)
    removeColumn($col$)
end while
```

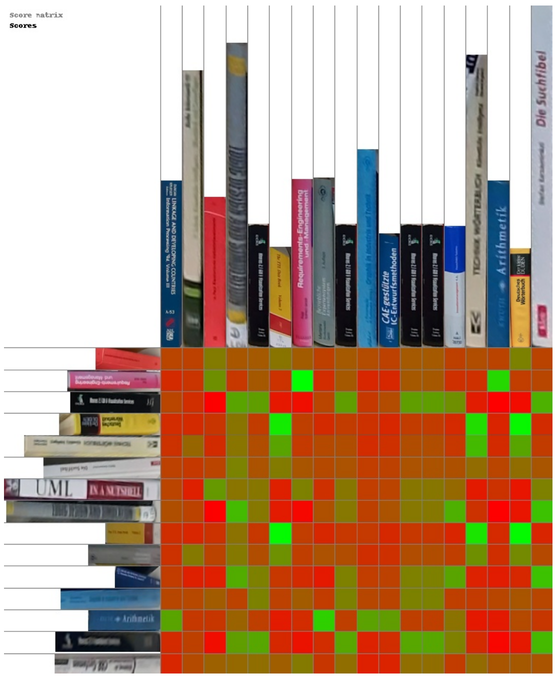


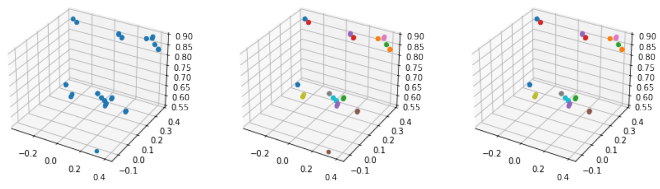
Fig. 8: The final score of each detected book (rows) against the ones registered in the database (columns). Green means higher score indicating a higher similarity.

C. Physical World Simulation (Fabian)

To make the vision less prone to noise, the final belief about the environment is calculated with 10 sequential observations. The environment consists of a set of recognized books including their pose similar to one observation. After accumulating the book sets, it can neither be assumed that each observation contains the same amount of books nor that the same books were recognized at the same position. To decide which book recognitions belong to the same book instance, the recognitions are spatially clustered using the position and the k-means clustering algorithm (fig. 9).

The parameter k is chosen to be the maximum number of book recognitions in one observation which ensures a sufficient number of centroids when clustering (fig. 9, b). Since miss detections should be excluded from the environment, clusters that have less than 4 recognitions are being pruned (fig. 9, c).

For each cluster, the position can be determined by averaging the positions from the associated book recognitions. To assign an ID, algorithm 1 is used again, except the rows are substituted with clusters rather than book recognitions



(a) 140 Detections (b) After clustering (c) After pruning

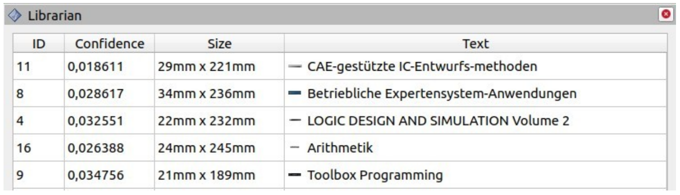
Fig. 9: Visualization of book clustering and pruning.

and the score for each book is computed as the number of ID occurrences inside the respective cluster. Furthermore, a confidence c for each cluster is defined as

$$c = \frac{\sum_{i=0}^{n} \text{conf}(b_i)}{n}$$

where b is the list of book recognitions and n is the size of the respective cluster. The function *conf* returns the confidence for the given book recognition computed in section IV-B7. With each cluster having a book ID, confidence, and a pose, the environment is complete and is then used for task planning and to create the collision objects in the planning scene.

1) RViz Panel (Fabian): To monitor the perceived environment described in the previous subsection, a special RViz panel is used. The panel titled *Librarian* displays each book recognition as a row of a table showing the book ID, confidence, estimated dimension, and title from the database (see fig. 10).



Librarian

ID	Confidence	Size	Text
11	0,018611	29mm x 221mm	CAE-gestützte IC-Entwurfs-methoden
8	0,028617	34mm x 236mm	Betriebliche Expertensystem-Anwendungen
4	0,032551	22mm x 232mm	LOGIC DESIGN AND SIMULATION Volume 2
16	0,026388	24mm x 245mm	Arithmetik
9	0,034756	21mm x 189mm	Toolbox Programming

Fig. 10: The RViz panel can be used to monitor the environment as it was percieved.

V. Task Planning (Mykhailo)

A separate task planning module is developed to process the environment state received from the perception module and issue commands to the manipulation module.

The backbone of the planning module is ROS [33], and all the communications between the modules are done in form of the ROS messages.

As can be seen in Fig. 1, the task planning module receives a world state representation from the perception module creates a plan, and sends it for execution to the manipulation module. In order to facilitate the perception module, the task planning module issues a command to stop or start the perception. This is done to avoid possible in-scene interference caused by the grippers executing motions.

The world state consists of the set of the book entities with the information regarding the pose, and properties retrieved from the database. The set of possible actions is:

1) perception start/stop
2) "pick+place", pick the specified book and place it against a vertical object in the specified place and from the specified direction.

We use a descriptive model of the world [8], that is we assume that outcome of the action is determined. The desired behavior can be input by sending a message to the module with the specified command. Currently, the implemented behavior is to arrange book by the desired property on the lower shelf starting from the left corner.

Due to the limited set of action (a book can be picked only when standing vertical) and strong assumptions on the environment (finite static environment, no explicit time, no concurrency, determinism of action success), as well as almost no precondition on the allowed actions, we have resorted to the use of the procedurally generated plans: the books on the scene are sorted regarding the desired criteria, and then moved to the lower shelf in the sorted order. This can be justified as: the cost of placing a book on the lower shelf is equal for all books (no search needed); all books should be sorted eventually; there is no action that allows for the fallback recovery.

For future work, we plan on expanding the planning module to include more complex behavior such as recovery from the fallback scenarios and adding new desired behaviors. This can be done in the current framework by extending the state space and removing the assumption that books are only standing, adding manipulation for new states of the book. This will allow us to use *integration of task and motion planning* like [21] or [12] which are suitable for our hierarchical architecture.

VI. Manipulation (Björn)

The book manipulation process is divided into two main procedures. First is grasping and extracting the book from the shelf. Second is placing the book into a designated position in the shelf again. The platform used is our custom fit PR-2, which has both a two finger gripper and a shadow hand on its two arms. For controlling the robot and planning, we utilized the MoveIt [5] framework as well as the BioIK solver [29]. Specifically, we use the OMPL planner for planning the motions. For motions, which require one of the grippers to a position further away, we don't just give a pose to the OMPL planner, but instead get a joint configuration for the corresponding arm from the BioIK solver and then let the OMPL planner move to that configuration. This has proven to be more reliable to result in configurations, which can execute the following steps, through the use of more specific constraints, that can be given to the BioIK solver.

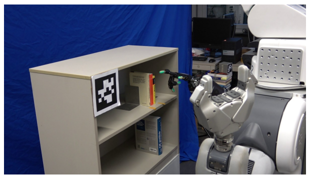
A. *Grasping and extracting*

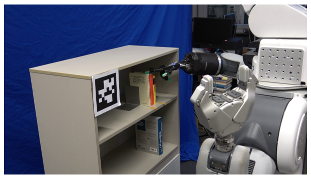
The grasping process is implemented as a bi-manual procedure utilizing both arms of the PR-2. The bi-manual approach for grasping and extracting the book was only our second plan. First, we wanted to expand on the approach in [18] and only utilize the shadow hand. During experimentation, we wanted the shadow hand to hold the book in the tilted position with its index finger and use the other fingers to grasp the book. However, due to the limited workspace of the shadow hand, we didn't manage to manually find a position, which allowed the planning, both FK and IK planners, to find a path to grasp the book. With additional quantitative testing, we also came to the conclusion, that the workspace of the shadow hand might not allow such and operation. Another option would be to use reinforcement learning to find such a position, but with doubt, that such a position even exists, and our limited time schedule, we decided not to invest more time into it and decided on a bi-manual approach instead.

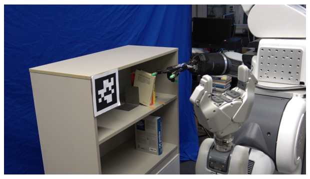
Now, the shadow hand is first used to bring the book into as position, which allows the other gripper to grasp the book around its spine. Since usually this isn't possible from the start because the book spines align, we implemented the previously mentioned process. It takes one finger of the shadow hand, places it on top of the book and starts tilting it by pulling on it. We settled on an approach, where the finger of the shadow hand is stretched out, while the other fingers are curled up. Originally, we had the finger in a hook like pose, so it could exert force more easily on the book. However, this approach ran into problems with the limited space between the book and the shelf above, making it necessary to switch to the stretched out finger.

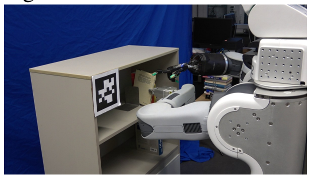
This tilting allows the gripper then to grasp the book on the newly exposed part of the book and later extract it. For this procedure to work, we need a few assumptions for the books to be true. First, their book spines need to align with each other. Second, the books need to stand upright in the shelf. Both these assumptions seem reasonable in a library environment. Further, the books are limited by weight constraints of the robot, and they need to be able to stand upright on their own consistently. After the gripper has grasped the book, the shadow hand then removes itself from the book and the gripper extracts it. It does so by first pulling the book pack further and then letting it go to fall back. This process lets the book fall back in an upright position again with a still exposed spine, allowing the gripper to again grasp the book, but with a more stable grasp, having a larger exposed area for grasping. Then the book gets pulled out from its position. The full process can be seen in 11.

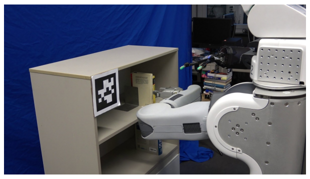
We chose to use the two general purpose manipulators already mounted on our PR-2, instead of building a custom gripper, suited for book manipulation, as in [25] or a more specialized gripper, for example a suction gripper [20]. Our reason for this is, that in the future, a librarian robot should also be able to perform other tasks than just manipulating books, for example interact with customers or other objects in the library. Therefore, a specialized gripper for just the task of manipulating books could hinder the robot to fulfill these tasks, which is why we wanted to try to allow the robot to complete this task without special customization.


(a) Moving before the book to prepare tilting.


(b) Moving the finger on the book to make contact for tilting.


(c) Pulling back the shadow hand to tilt the book.


(d) Use the gripper to grasp the book while it is tilted by the shadow hand.


(e) Gripper after pulling the book out a bit and releasing it.

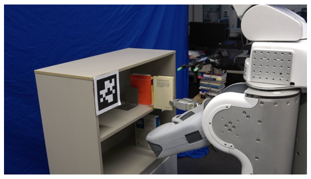
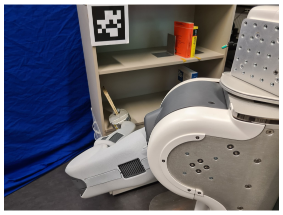
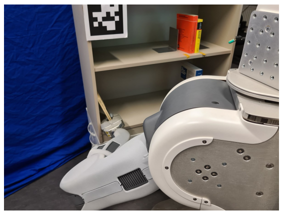
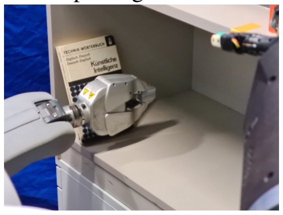

(f) Pulled out book after the gripper has grasped it again.

Fig. 11: The process of extracting the book.


(a) Moving the book into its initial position in the shelf before placing.


(b) Leaning the book against the shelf to the left.


(c) Positioning next to the book to prepare pushing the book.

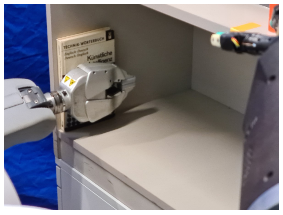

(d) Pushing the book against the shelf to let it stand upright.

Fig. 12: The process of placing the book.

B. *Placing*

For the placing procedure, we rely on the two finger gripper and its force-torque sensor. First, the book is moved next to the designated position and tilted slightly, so in the next steps it can be leaned against either the shelf wall or another book. This limits our implementation to not be able to place the book on its own or between other books. However, placing the book on its own can be easily implemented, if a way to stabilize the book during the placing can be found. Placing the book between other books, however, is similar to the famous peck insertion task, and solving this was outside the scope of our project. Therefore, we have decided not to try to implement any way to solve this problem. The leaning against an object process is then done using the force-torque sensor. At the start, the book is moved as far as possible against the object. For the stopping point, a threshold for the change in force on the gripper between holding the book and touching the object is considered. Similarly, the next step is moving as far down as possible to touch the book to the shelf. Then, the gripper can then release the book, and the book is leaning against the object. Finally, the gripper can move again from the side against the book, pushing it upright against the object. This movement is also controlled using the force-torque sensor. Now, the book stands upright next to the designated object. The full process can be seen in 12.

While this process has proven its feasibility in our experiments, it also has its drawbacks. First, it requires a surface against which the books have to be leaned against, making it impossible to place the books in the middle of an empty shelf. However, when sorting books, the usual approach would be to place them from left to right anyway, making this limitation acceptable. The second, more limiting factor, is the problem, that placing a larger against a smaller book can result in failure, if the height difference during leaning the second book against the first is too great. This limits our process to only sort the books consistently by height, from highest to lowest. In the future, this issue might be able to be solved by implementing a different procedure, if the height difference is too great. One example would be to grasp the second book on the top of its spine with the two finger gripper and already placing it vertically. While this is not done, we have a working procedure for placing the picked books, with some limitations.

VII. Results (Shang-Ching Liu)

The system has been organized into three modules, including librarian_vision, librarian_manipulation, and librarian_common. In the final demo, we demonstrate sorting three books on the upper shelf and placing them on the lower shelf, as shown in figure 11. You can see the book matching result easily in the RViz panel as figure 13. After the robot recognizes the book, we turn off the perception pipeline and start the grasping task. The grasping performs cooperation from both hands and notices we have intentionally placed the front and back sides of the book in front of the camera vision focus area to be further processed for text reading or to eliminate the error in the perception area. Finally, place the sorted book on the lower shelf.

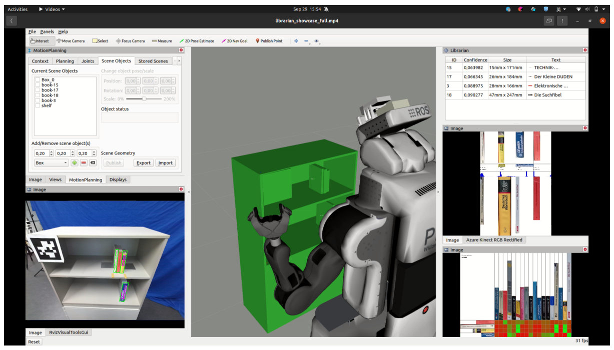


Fig. 13: Demo result of RViz

VIII. Conclusion (Mykhailo / Björn)

In this paper, we have successfully implemented a pipeline, which enables the robot to locate, pick and place different books within a bookshelf. By first using a bounding box approach to detect the book spines, we managed to locate the different books at previously unknown positions on the shelf. Through HSV matching, we then managed to identify the individual books, using the available database. Further, we utilized the database to find the correct order in which to grasp and sort the books, which is then accomplished by grasping them and placing them, sorted, in another area of the bookshelf. The grasping process utilized both different grippers from the PR-2 to be able to even pick a book from between other books. While our pipeline is not the most robust and has some limitations, our results show the feasibility of our approach and with additional work in the future, it should be possible to remove many of the current limitations and make the pipeline more robust. Some possible improvements for the vision part of the pipeline would include a larger training set for our YOLO network to prevent overfitting, making it more robust and allowing it to generalize better, or more fine-tuned metrics for our matching, making this part more robust as well. For the grasping part, having the robot move to be able to reach more areas of the shelf or modify the placing strategy as already described, to account for larger books getting placed against smaller ones would remove a lot of the current limitations. Afterward, the pipeline could be upgraded with more interaction between the different parts or new strategies for grasping, which would all be interesting improvements in the future.

Acknowledgment

We would like to express our special thanks of gratitude to both supervisor Michael Görner and Dr. Norman Hendrich as well, who provide fully support to all issues during the project. Also, the research group Technical Aspects of Multimodal Systems (TAMS) giving rich experiment robot environment that make us came to know about so many new things.

References

[1] Jeonghun Baek, Geewook Kim, Junyeop Lee, Sungrae Park, Dongyoon Han, Sangdoo Yun, Seong Joon Oh, and Hwalsuk Lee. What is wrong with scene text recognition model comparisons? dataset and model analysis. In *International Conference on Computer Vision (ICCV)*, 2019.

[2] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. Character region awareness for text detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9365–9374, 2019.

[3] Lina Cao, Mengdi Liu, Zhuqing Dong, and Hua Yang. Book spine recognition based on opencv and tesseract. *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 1:332–336, 2019.

[4] Zhanzhan Cheng, Fan Bai, Yunlu Xu, Gang Zheng, Shiliang Pu, and Shuigeng Zhou. Focusing attention: Towards accurate text recognition in natural images. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5086–5094, 2017.

[5] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.

[6] Xunrui Duan, Qingjie Zhao, and Shahzad Anwar. Identifying books in library using line segment detector and contour clustering. *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, pages 998–1003, 2012.

[7] Spencer G. Fowers, Dah-Jye Lee, and Guang ming Xiong. Improved library shelf reading using color feature matching of book-spine images. *2010 11th International Conference on Control Automation Robotics & Vision*, pages 2160–2165, 2010.

[8] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[10] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society*, 18 5-6:602–10, 2005.

[11] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[12] Keliang He, Morteza Lahijanian, Lydia E. Kavraki, and Moshe Y. Vardi. Towards manipulation planning with temporal logic specifications. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 346–352, 2015.

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015.

[14] Glenn Jocher. yolov5. https://github.com/ultralytics/yolov5, 2022.

[15] Bong Keun Kim, Takayuki Sugawara, Kenichi Ohara, Kosei Kitagaki, and Kohtaro Ohba. Design and control of the librarian robot system in the ubiquitous robot technology space. In *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 616–621, 2008.

[16] In Lee. Service robots: a systematic literature review. *Electronics*, 10(21):2658, 2021.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] Antonio Morales, Mario Prats, Pedro Sanz, and Angel P Pobil. An experiment in the use of manipulation primitives and tactile perception for reactive grasping. In *Robotics: Science and Systems, Workshop on Robot Manipulation: Sensing and Adapting to the Real World, Atlanta, USA*, 2007.

[19] Tomohiro Motoda, Damien Petit, Weiwei Wan, and Kensuke Harada. Bimanual shelf picking planner based on collapse prediction. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 510–515, 2021.

[20] Kazuyuki Nagata and Takao Nishi. Modeling object arrangement patterns and picking arranged objects. *Advanced Robotics*, 35(16):981–994, 2021.

[21] Srinivas Nedunuri, Sailesh Prabhu, Mark Moll, Swarat Chaudhuri, and Lydia E. Kavraki. Smt-based synthesis of integrated task and motion

plans from plan outlines. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 655–662, 2014.

[22] M. P. Nevetha and Aravind Baskar. Automatic book spine extraction and recognition for library inventory management. In *WCI '15*, 2015.

[23] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[24] George Jose Pollayil, Giorgio Grioli, M. Bonilla, and Antonio Bicchi. Planning robotic manipulation with tight environment constraints. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9385–9392, 2021.

[25] Mario Prats, Ester Martínez, Pedro J Sanz, and Angel P Del Pobil. The uji librarian robot. *Intelligent Service Robotics*, 1(4):321–335, 2008.

[26] Rafael Ramos-Garijo, Mario Prats, Pedro J Sanz, and Angel Pasqual Del Pobil. An autonomous assistant robot for book manipulation in a library. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, volume 4, pages 3912–3917. IEEE, 2003.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[28] Roboflow Inc. Roboflow. https://roboflow.com/, 2022. [Online; accessed 28-September-2022].

[29] Philipp Ruppel. *Performance optimization and implementation of evolutionary inverse kinematics in ROS*. PhD thesis, Master Thesis, University of Hamburg, 2017.

[30] Shang-Ching Liu. book spine dataset. https://app.roboflow.com/ds/NC9bTX2P8g?key=FKRtGVDdhx, 2022. [Online; accessed 28-September-2022].

[31] R. Smith. An overview of the tesseract ocr engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2:629–633, 2007.

[32] Matteo Sostero. Automation and robots in services: review of data and taxonomy. *JRC Working Papers Series on Labour, Education and Technology No. 2020/14*, 2020.

[33] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. [Online; accessed 28-September-2022].

[34] Nuzhat Tabassum, Sujan Chowdhury, Muhammad Kamal Hossen, and Salah Uddin Mondal. An approach to recognize book title from multi-cell bookshelf images. *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pages 1–6, 2017.

[35] Lior Talker and Yael Moses. Viewpoint-independent book spine segmentation. *IEEE Winter Conference on Applications of Computer Vision*, pages 453–460, 2014.

[36] Wikipedia contributors. Hsl and hsv — Wikipedia, the free encyclopedia, 2022. [Online; accessed 28-September-2022].

[37] Xiaojun Yu, Zeming Fan, Hao Wan, Yuye He, Junye Du, Nan Li, Zhaohui Yuan, and Gaoxi Xiao. Positioning, navigation, and book accessing/returning in an autonomous library robot using integrated binocular vision and qr code identification systems. *Sensors*, 19(4):783, 2019.

APPENDIX (SHANG-CHING LIU)

Following is the progress of the Trixi through developing:

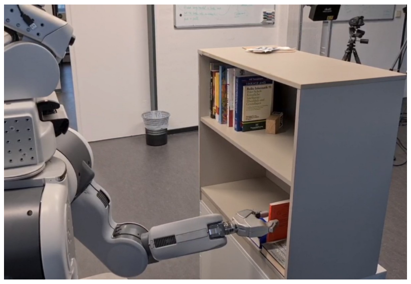


Fig. 14: Begin from one book being grasped by one gripper.

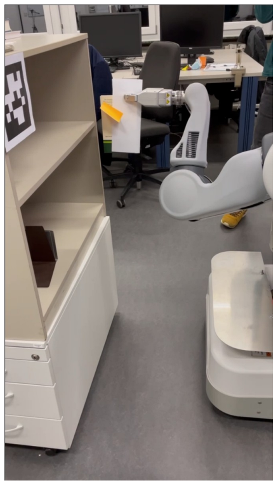


Fig. 15: Recognize the text and grasped specific book, however the text size matter a lot conflict with the book weight that robot gripper can offer, so we create a fake book cover here.

Fig. 16: Collecting the book spine database image.

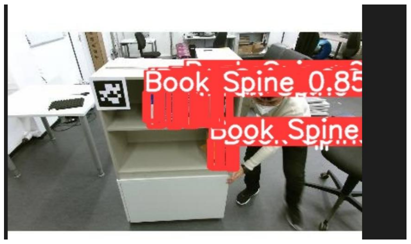


Fig. 17: Get quite good result from YOLOv5 to detect the book spine.

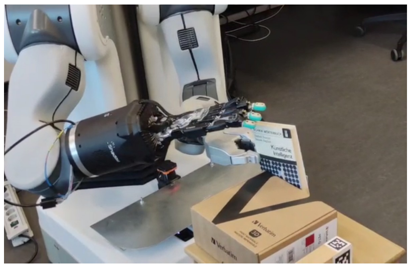


Fig. 18: First time pick the book by shadow-hand success.

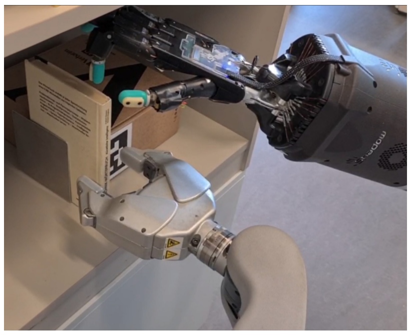


Fig. 19: First time, the cooperation between shadow hand and gripper success.

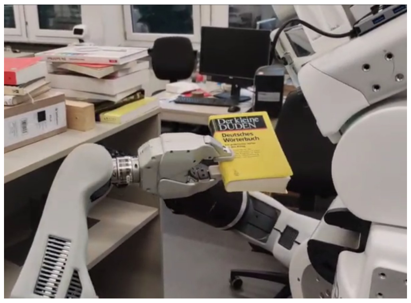


Fig. 20: First motion to read the book implement in Trixi.