Using Off-the-Shelf Deep Neural Networks for Position-Based Visual Servoing

Matteo Tschesche[1][0009−0006−0872−2835], Till Hofmann[2][0000−0002−8621−5939], Alexander Ferrein[1][0000−0002−0643−5422], and Gerhard Lakemeyer[2][0000−0002−7363−7593]

[1] Mobile Autonomous Systems and Cognitive Robotics Institute, FH Aachen University of Applied Sciences, Aachen Germany
{tschesche,ferrein}@fh-aachen.de
[2] Knowledge-Based Systems Group, RWTH Aachen University, Aachen, Germany
{hofmann,lakemeyer}@kbsg.rwth-aachen.de

Abstract. Visual servoing is a well-established technique for object grasping and controls the robot in a closed-loop fashion. It typically uses hand-crafted features or a neural network that directly learns the control output. We propose an alternative approach that uses an off-the-shelf neural network object classifier and can therefore compute target poses without manually selected features while also not requiring training from scratch. Instead, the object classifier only needs to be fine-tuned on a domain-specific dataset, significantly reducing the amount of required training data. We describe a task sequencing approach that can control a robot with a mobile base and an additional gripper, which is a typical setup in many robotics applications. We evaluate the approach in the RoboCup Logistics League and demonstrate the reliability and speed of the proposed approach.

Keywords: sensory-motor control · visual servoing · 3D perception.

1 Introduction

Grasping objects is an essential part of almost every robotics application. While motion planning allows for computing trajectories to reach a goal pose, a motion planner typically operates in an open-loop fashion, i.e., it senses the target and computes the path once and then executes it until completion. In contrast, in *visual servoing* [7], a closed-loop controller updates its target and adapts its movements in every iteration and can therefore react to moving targets and sensor errors, but typically does not plan long trajectories. Usually, visual servoing uses manually selected features, which are often based on simple shapes such as lines, or circles [9]. While this has been used successfully in many applications, it requires careful fine-tuning of the selected features and does not allow for complex objects or shapes. Alternatively, some approaches directly learn a closed-loop controller, i.e., a neural network determines the next control output given an input image from a camera [25]. However, this typically requires

training a neural network from scratch and therefore requires large amounts of training data.

In this paper, we propose a visual servoing architecture that uses a neural network as object detector in order to determine the pose of the target object but then uses a classical control law to determine the control outputs. This allows us to use an off-the-shelf object detector such as YOLOv4 [5] for object detection and therefore alleviates the need for expensive training. Instead, a pre-trained network can be used that only requires fine-tuning on a domain-specific dataset.

The proposed framework supports kinematically redundant robots that have a wide-range but inaccurate mobile base and an additional close-range but accurate gripper. The two components are combined with *task sequencing*, which first searches the target object using the robot's navigation stack and switches to a closed-loop controller once the target object has been detected. It then moves the mobile base to a target region near the target object, before it adjusts its pose by moving the gripper until the target pose has been reached with high accuracy. This allows wide-range and precise movements while also being fast.

We evaluate the approach in several tasks from the RoboCup Logistics League (RCLL) [30]. In the RCLL, a team of robots needs to transport workpieces between machines in order to execute a sequence of manufacturing steps. Hence, each robot needs to pick workpieces from and put workpieces on the machine's conveyor belt, grab workpieces from a shelf, and feed workpieces onto a slide. As each assembly process involves numerous grasping tasks, speed and reliability of grasping is a crucial factor in the design of the robots. As we will see, the proposed approach is able to execute grasping tasks more reliably but also faster than a baseline approach using the iterative closest point (ICP) algorithm, which has been successfully used in previous iterations of the competition.

The remainder of the paper is structured as follows. In Section 2, we discuss related work, before we describe our approach in greater detail in Section 3. We evaluate the proposed visual servoing approach and compare it against the baseline in Section 4 and finally conclude in Section 5.

2 Related Work

Visual servoing [14,6,7] is the task of moving a camera or robot to match the current configuration of visual features with a target configuration. In contrast to motion planning [18], visual servoing uses a closed control loop and typically does not involve collision avoidance, but uses real-time visual information to compute the next control inputs until the goal is reached. The controller may directly determine joint inputs or provide set-point inputs to a joint-level controller, which is sometimes referred as a *dynamic look-and-move* controller [14]. Typically, the camera is moving with the robot (*eye-in-hand*), but other configurations such as a static camera observing the robot workspace are also possible [7].

Visual servoing can be formulated as an optimization problem of minimizing the error $\mathbf{e}(t) = \mathbf{f}(t) - \mathbf{f}^*$ between the target features $\mathbf{f}^*$ and the the current features $\mathbf{f}(t)$ at timestep t. Depending on the choice of features $\mathbf{f}$, we can dis-

tinguish *image-based visual servoing* and *position-based visual servoing* [7,15], where the former directly uses the image feature measurements in the image plane to compute the error, while the latter estimates the camera pose and compares the estimated camera pose to the target pose. Position-based visual servoing requires a geometric model and a calibrated camera system, while mapping the image space to the workspace in image-based visual servoing sometimes results in control problems and non-convergent behavior [7]. Hybrid approaches try to avoid these drawbacks [9], e.g., 2-$\frac{1}{2}$-D visual servoing [26] combines image features with 3D data by determining a partial estimate for the camera pose without requiring a camera model, and partitioned visual servoing [10] decouples the z-axis motions from the other degrees of freedom.

Typically, visual servoing requires carefully designed visual features which are usually based on detecting simple shapes such as circles as lines or identifying predefined tags [9]. More recently, visual servoing has been combined with various learning approaches to allow for more complex features [25]. Several approaches learn the relative pose of a pair of images, e.g., based on optical flow estimation [32], convolutional neural networks (CNNs) [4,17], or siamese neural networks [38]. Alternatively, a policy that directly determines the control inputs can be learned, e.g. with CNN-based regression [34], unsupervised reinforcement learning [20,31,21,16], or one-shot imitation learning [1]. While these approaches can be used without fine-tuning visual features, they typically require large amounts of training data and sometimes suffer from the reality gap, that is, while they work well in simulation, reproducing similar precision in real-world domains is challenging. As an alternative and similar to the approach proposed in this paper, pre-trained neural networks can be used to detect objects, which then serves as input for visual servoing. This has been used successfully with stereo cameras for navigating drones towards wind turbines [12] and only using a RGB data for aligning bolster springs in railway maintenance application [23]. In contrast to the former approach, our approach does not require a stereo camera. The latter approach requires the image plane to be parallel to the object plane, while our approach allows arbitrary initial yaw rotations.

A robot is said to be *kinematically redundant* if it has more degrees of freedom than required for the given task [8]. This can be exploited, for, say, enlarging the stable area of the task [29], avoiding joint limits [11], or for achieving secondary goals [28]. *Task sequencing* [27] combines several subtasks by executing each task in a pre-defined order, e.g., by first constraining only a few degrees of freedom while the robot is far away from the goal and then adding constraints as the robot gets closer to the goal [27]. A *task priority strategy* [8,11] assigns a priority to each task and only enables a lower priority task while the higher priority task has reached its goal. A similar approach can also be used to combine a mobile base with a robotic arm [24], where the mobile base is prioritized while the robot far away from the goal, while the arm is prioritized for grasping tasks.

For the RCLL, several approaches have been described to solve manipulation tasks. The team GRIPS uses a closed-loop approach using a LiDAR as sensor [35]. As the shape of the machines used in the RCLL is known, a cluster-

ing algorithm can be used to compute the pose of the machine. From this pose, the goal pose for grasping can be derived. A closed-loop sliding mode controller is used to reach the computed goal pose. Before adopting the approach proposed in this paper, the team Carologistics used an open loop approach with multiple stages [13]. In a first stage, similar to [35], the machine's pose is estimated from the LiDAR. Next, a plane fitting algorithm based on RANSAC is used to detect the front of the conveyor belt, using point cloud data from an RGB-D camera. Finally, the exact pose of the conveyor belt is computed with an ICP algorithm. In every stage, the robot moves to the computed pose without additional sensor feedback. This approach serves as baseline for the evaluation in Section 4.

3 Visual Servoing with an Off-the-Shelf Object Detector

In the following, we describe our position-based visual servoing approach consisting of 3D object localization, triangulation, and a control approach based on task sequencing. In order to determine the goal pose, we use the pre-trained and fine-tuned object detector YOLOv4 [5]. This allows us to use learned features rather than manually defined hard-coded features while avoiding the need of large amounts of training data, as we only need to fine-tune the object detector to the domain at hand. Furthermore, assuming geometrical constraints given by the domain, determining the 3D position of the target location is sufficient, as roll, pitch, and yaw can be inferred, for instance, because the target object is known to be placed on a flat surface. Given the location of the target object in the 2D image and the known shape of the target object, the 3D goal position is computed by means of triangulation. Finally, as the robot is kinematically redundant, we use task sequencing to first move the mobile base to the vicinity of the goal and then use the robot's gripper to reach the goal region.

3.1 3D Object Localization

The controller needs a 6D pose as target input. However, it is often sufficient to determine the 3D position of the target and then infer the rotation from the environment. This is the case in the RCLL: the conveyor belt and slide are known to be placed level on the machines, i.e., there is no rotation relative to the machine, and the machine itself stands flat on the ground. Similarly, workpieces are always placed flat on the conveyor belt and are rotationally symmetric about the yaw axis, hence we can assume that its yaw is zero. This allows us to use an off-the-shelf object detector that only uses RGB input, which computes a bounding box of the target object in the image. As the shape of the object is known, triangulation can be used to determine the 3D position. In this paper, we use YOLOv4 [5] and YOLOv4-tiny [37] to compute bounding boxes of the target objects. YOLOv4 uses a CNN architecture and is fast enough to be used in a closed-loop approach while also being precise enough to determine the goal pose with sufficient accuracy. It is pre-trained on the COCO dataset [22] and we fine-tuned the detection layers on a domain-specific training set.

Triangulation Assuming the shape of the target object is known, we can raycast through bounding box corners and apply triangulation to project the detected 2D bounding boxes from image space to 3D Euclidean space, as shown in Figure 1. We use the lower bounding box corners for the projection $\mathbf{i_1} = \begin{pmatrix} u_1 & v_1 \end{pmatrix}^T$ and $\mathbf{i_2} = \begin{pmatrix} u_2 & v_2 \end{pmatrix}^T$ where $v_1 = v_2 = v$, shown as blue dots in image and Euclidean space. They are used to determine the raycast direction for p_1 given by (Δ_{x_1}, Δ_y) and for p_2 by (Δ_{x_2}, Δ_y) normalized for $z = 1$. Using the known object width, w_o, and a calibrated camera with focal lengths f_x in pixel-width and f_y in pixel-height, the object positions $\mathbf{p_m} = \begin{pmatrix} x_m & y_m & z_m \end{pmatrix}^T$ in the middle between $\mathbf{p_1} = \begin{pmatrix} x_1 & y_1 & z_1 \end{pmatrix}^T$ and $\mathbf{p_2} = \begin{pmatrix} x_2 & y_2 & z_2 \end{pmatrix}^T$ of the projection of bounding box corners to the 3D Euclidean space can be computed using the angle between the raycast of p_m and the z axis defined by α as follows:

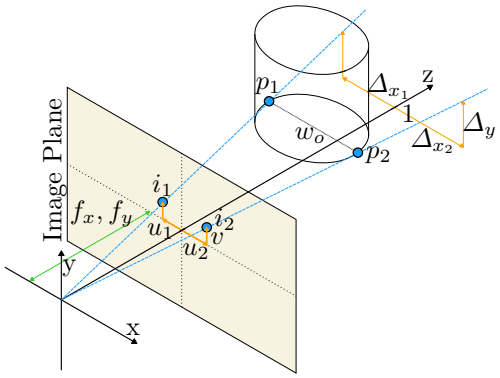

Fig. 1: 3D projection of a workpiece.

$$p_m = \begin{pmatrix} \Delta_{x_1} \left(\frac{\cos\alpha\, w_o - \sin\alpha\, w_o\, \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin\alpha\, w_o}{2} \right) + \frac{\cos\alpha\, w_o}{2} \\ \Delta_y \left(\frac{\cos\alpha\, w_o - \sin\alpha\, w_o\, \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin\alpha\, w_o}{2} \right) \\ \frac{\cos\alpha\, w_o - \sin\alpha\, w_o\, \Delta_{x_2}}{\Delta_{x_2} - \Delta_{x_1}} + \frac{\sin\alpha\, w_o}{2} \end{pmatrix}$$

Target Poses Using p_m and the known height h of the object, we can compute the center and upper edge of the target object. Since kinematically redundant robots typically have a mobile base that is less accurate than the gripper, it is often useful to define two different goal poses, one for the mobile base and one for the gripper. The target pose for the gripper is set to be just above the target object such that the robot can move down the gripper to reach the target. The target pose for the mobile base is set further away from the actual goal to avoid collisions.

3.2 Controllers

Once we have determined the target pose for the mobile base $\begin{pmatrix} x_b^* & y_b^* & \psi_b^* \end{pmatrix}^T$ and the gripper $\begin{pmatrix} x_g^* & y_{g^*}^* & z_g^* \end{pmatrix}^T$, we can use simple linear controllers to reach each from its current pose $\begin{pmatrix} x_b & y_b & \psi_b \end{pmatrix}^T$ and $\begin{pmatrix} x_g & y_g & z_g \end{pmatrix}^T$ for base and gripper respectively. As the mobile base is omni-directional, we can compute control outputs for translational speeds $v_{b,x}$, $v_{b,y}$, and the rotational speed $v_{b,\psi}$ separately. We

use a P controller [3] to determine the desired speed $\mathbf{v}_b = \begin{pmatrix} v_{b,x} & v_{b,y} & v_{b,\psi} \end{pmatrix}^T$ at timestep t:

$$\begin{pmatrix} v_{b,x} \\ v_{b,y} \\ v_{b,\psi} \end{pmatrix} = \begin{pmatrix} \max\left(v_{b,x}^{\min}, \min\left(v_{b,x}^{\max}, a_{b,x} \cdot t, |x_b^* - x_b| \cdot \frac{v_{b,x}^{\max}}{a_{b,x}^{\text{dec}}\, s_{b,x}}\right)\right) \\ \max\left(v_{b,y}^{\min}, \min\left(v_{b,y}^{\max}, a_{b,y} \cdot t, |y_b^* - y_b| \cdot \frac{v_{b,y}^{\max}}{a_{b,y}^{\text{dec}}\, s_{b,y}}\right)\right) \\ \max\left(v_{b,\psi}^{\min}, \min\left(v_{b,\psi}^{\max}, a_{b,\psi} \cdot t, |\psi_b^* - \psi_b| \cdot \frac{v_{b,\psi}^{\max}}{a_{b,\psi}^{\text{dec}}\, s_{b,\psi}}\right)\right) \end{pmatrix}$$

Here, $\mathbf{v}_b^{\min} = \begin{pmatrix} v_{b,x}^{\min} & v_{b,y}^{\min} & v_{b,\psi}^{\min} \end{pmatrix}^T$ and $\mathbf{v}_b^{\max} = \begin{pmatrix} v_{b,x}^{\max} & v_{b,y}^{\max} & v_{b,\psi}^{\max} \end{pmatrix}^T$ are the minimal and maximal possible velocities respectively. The acceleration is constant and defined by the factors $\mathbf{a}_b = \begin{pmatrix} a_{b,x} & a_{b,y} & a_{b,\psi} \end{pmatrix}^T$. The velocity required while decelerating is computed based on the required distance $\mathbf{s}_b = \begin{pmatrix} s_{b,x} & s_{b,y} & s_{b,\psi} \end{pmatrix}^T$ to stop at the target, the difference between target and current pose, the current velocity, and the constant deceleration $\mathbf{a}_b^{\text{dec}} = \begin{pmatrix} a_{b,x}^{\text{dec}} & a_{b,y}^{\text{dec}} & a_{b,\psi}^{\text{dec}} \end{pmatrix}^T$.

The gripper uses three separate stepper motors for the x, y, and z axis, each being controlled independently using a P controller. We define control output for the x axis, the other axes are controlled analogously. First, the maximal possible speed $v_{g,x}^p(i)$, that can be decelerated before reaching the target, at timestep i is computed by:

$$v_{g,x}^p(i) = \sqrt{2\, a_{g,x}\ \left|x_g^* - x_g(i)\right|}$$

Here, the constant acceleration $a_{g,x}$ and error in x direction between the x position $x_g(i)$ at timestep i and target x position x_g^* are used.

The velocity $v_{g,x}(i)$ in x direction at timestep i is computed using the maximum speed of the gripper's x axis $v_{g,x}^{\max}$ as follows:

$$v_{g,x}(i) = \begin{cases} \min(v_{g,x}^{\max}, v_{g,x}(i-1) + \frac{a_{g,x}}{v_{g,x}(i-1)} & \text{if } v_{g,x}^p(i) > v_{g,x}(i-1) \\ v_{g,x}(i-1) - \frac{a_{g,x}}{v_{g,x}(i-1)} & \text{else} \end{cases}$$

The velocities are applied by computing the time between each motor step using the ramp algorithm [2].

3.3 Task Sequencing

As the robot's three degrees of freedom (DOF) mobile base and three DOF gripper are kinematically redundant, we use task sequencing as shown in Figure 2 to coordinate the subtasks.

Search In the first subtask, the robot is far away from the goal and may possibly not sense the target. Instead, it uses the expected pose (e.g., based on the known machine position) to instruct the robot's navigation stack in order to reach a position from which the target object is visible. As the robot gets closer to the machine and the machine is detected, the expected position is computed

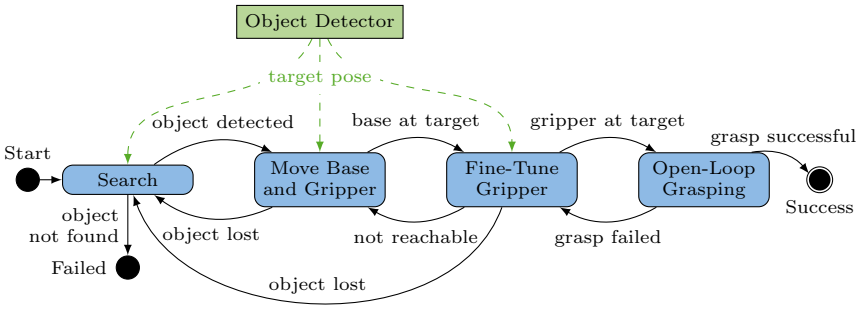

Fig. 2: Task sequencing.

relative to the observed position of the machine, which improves the accuracy and increases the robustness. Once the object is detected reliably and the robot is within a threshold to the target pose, the next subtask is activated. If the navigation fails to reach the target region the complete task fails.

Move Base and Gripper Once the target object is reliably detected and the robot is close to the target, the first visual servoing task is activated, which moves the mobile base and gripper simultaneously. The base velocity is reduced as the distance towards the machine decreases. If vision of the object is lost, the controller switches back to *Search*. The goal of this subtask is to move the mobile base to its target pose within a threshold, using the target pose and control law as described above. At the same time, the gripper is moved to its expected target pose assuming the robot's mobile base precisely reaches its target pose.

Fine-Tune Gripper Once the mobile base has reached its target, the gripper needs to be moved to reach its target pose. As the mobile base can only move imprecisely, the gripper pose typically needs to be readjusted even if the gripper already reached its target pose in the previous subtask. This subtask succeeds if the gripper reaches its target pose within a small threshold. In case the object moved and is no longer in range of the gripper, the previous subtask is reactivated to re-adjust its base. If the object is no longer visible, the procedure switches back to *Search*.

Open-Loop Grasping After the gripper has reached its target pose, the object is grasped in an open loop fashion, assuming that the workpiece will not move. This subtask consists of three poses which need to be reached consecutively. It starts from the visual servoing target pose, moves down the gripper, and either picks the workpiece up or puts it down. Afterwards, the gripper is moved back in its default pose.

Table 1: Performance and speed of YOLOv4 and YOLOv4-tiny with a minimum IoU of 50% and 75%, showing the F1 score, average precision for workpieces (AP_{wp}), conveyor belts (AP_c), and slides (AP_s), mean average precision (mAP), and inferences per second (IPS).

Object Classifier	min. IoU	F1	AP_{wp}	AP_c	AP_s	mAP	IPS (s^{-1})
YOLOv4	50	**0.97**	**0.9966**	**0.9888**	**0.9804**	**0.9886**	0.904
	75	0.92	0.9741	0.9147	0.8217	0.9035	0.904
YOLOv4-tiny	50	0.91	0.9720	0.9526	0.8688	0.9311	**18.904**
	75	0.74	0.7976	0.7553	0.4263	0.6597	**18.904**

4 Evaluation

We evaluate our visual servoing framework[3] and its components in several scenarios of the RCLL. First, we analyze whether YOLOv4 and YOLOv4-tiny are performant enough to be used for real-time object detection on a robot. As a robot typically has no dedicated GPU, all inferences needs to be done on a CPU. Hence, inference time on a CPU is crucial. Next, we compare the novel visual servoing approach against our previous approach, which was based on ICP.

4.1 Object Detection

We compare the `darknet`[4] implementations of YOLOv4 and YOLOv4-tiny with 3 detection layers, which were fine-tuned on object classes from the RCLL.

Training We used 4000 labeled images for training and used several augmentation techniques to improve robustness. Each image was rotated by 0°, 90°, 180°, or 270°. Additionally, the dataset was augmented by applying cropping and padding, hide and seek [33], and CutMix [39], using random images from the COCO dataset [22] as replacement.

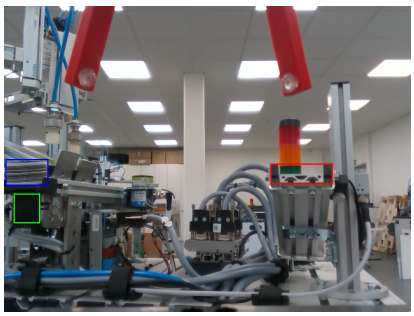

Fig. 3: Object detection with bounding boxes for a workpiece (blue), conveyor belt (green), and slide (red).

Results Table 1 shows a comparison of YOLOv4 and YOLOv4-tiny with a minimum Intersection over Union (IoU) of 50% and 75% on 300 test images using an AMD Ryzen 5 1600 CPU. We can see that YOLOv4 performs better

[3] The code including weights of the neural network is available at https://zenodo.org/records/11032321.

[4] https://github.com/kiyoshiiriemon/yolov4_darknet

than YOLOv4-tiny, but accomplishes less than one inference per second, which is too slow for closed-loop visual servoing, typically requiring 5 to 10 inferences per second to work reliably. On the other hand, for YOLOv4-tiny, we can see that a minimum IoU of 75% does not perform well with a mean average precision (mAP) of 0.6597. However, with a minimum IoU of 50%, performance is significantly better while also faster with approximately 19 inferences per second, thus being both performant enough and fast enough to be used for visual servoing.

4.2 Comparison of VS and ICP

We compare our visual servoing approach with the ICP approach previously used by Carologistics [13]. We do so by benchmarking several grasping tasks with varying machine (BS, CS, RS) and initial robot poses (A, B, C, D, and E with different orientations), as shown in Figure 4. We evaluate picking tasks of a randomly chosen workpiece for each machine's output side of the conveyor belt as well as each cap station's shelf, as well as putting tasks for each machine's input side of the conveyor belt. This results in seven pick and three put tasks for each starting pose and hence 100 different tasks overall. The workpiece configuration was chosen randomly. This setup covers every possible manipulation task in the RCLL from multiple starting poses.

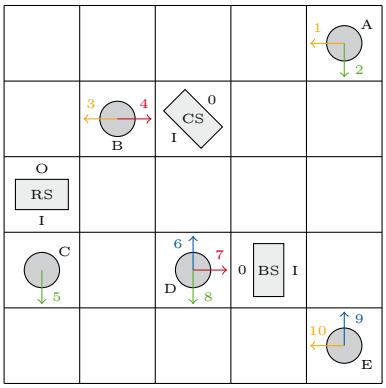

Fig. 4: Field configuration for the evaluation tasks with a base station (BS), ring station (RS), and cap station (CS) with input (I) and output (O), and 10 different initial robot poses (A-E).

The results are shown in Figure 5. We can see that for almost all tasks, the VS approach was faster than the ICP approach. On average, the ICP approach required 26.1 s, while the VS approach required 21.1 s. Furthermore, the VS approach was also more reliable, as it succeeded in 99 out of 100 tasks, while the ICP approach was only successful in 92 out of 100 tasks. On the robot's Intel Core i5-8400H CPU, the VS approach updates the target pose with approximately 19 Hz and thus is sufficiently fast to run on the robot.

Grasping Challenge As an additional benchmark, we evaluated the VS approach in the RCLL Grasping Challenge, where the task of a team of three robots is to move a workpiece from a machine's output to its input, three times for each of the three machines [19]. The team Carologistics won this challenge in 2022 using the ICP approach described above, requiring 225 s to complete the task. In comparison, the VS approach only required 123 s, decreasing the total execution time by 45%.

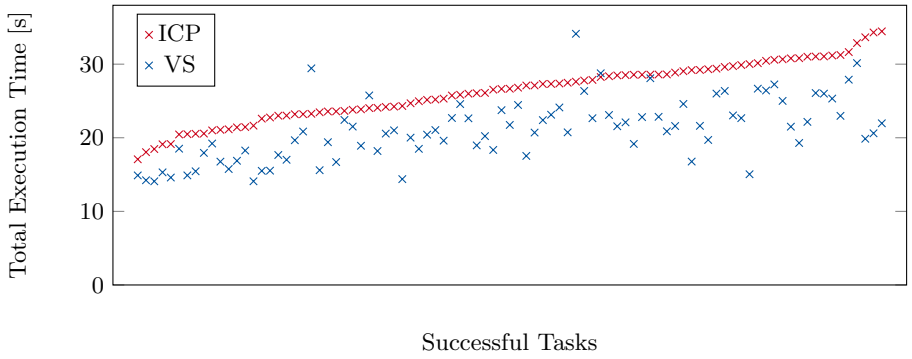

Fig. 5: Total execution time of each attempt, showing only tasks where both the ICP approach and the VS approach were successful. The tasks are ordered by the execution time needed by the ICP approach.

In conclusion, the proposed approach is more reliable and faster than the previous approach based on ICP, which was the best approach in the 2022 grasping challenge.

5 Conclusion

In this paper, we have described a visual servoing framework for kinematically redundant robots with a mobile base with a wide range of motion but low accuracy and a limited-range but highly accurate gripper. Using an off-the-shelf neural network object classifier such as YOLO, the framework is capable of precise grasping without fine-tuning manually selected features but also with only little requirement for training. As we can use a pre-trained neural network, we only need to apply domain-specific fine-tuning to obtain a suitable object detector. The detected object position in image space is then used to compute the target pose of the robot by means of triangulation. A task sequencing approach exploits the kinematic redundancy by first aligning the mobile base to the target with a high tolerance and then moving the gripper closely to the target pose.

We demonstrated the effectiveness of the proposed visual servoing approach by comparing it to a previous approach based on ICP in several scenarios of the RCLL. The visual servoing approach was not only more reliable but also executed the tasks significantly faster. The proposed visual servoing approach contributed to the success of the team Carologistics at RoboCup 2022 [36].

Acknowledgements This work was supported by the Federal Ministry of Education and Research (BMBF) under grant No. 02L19C602 and the EU ICT-48 2020 project TAILOR (No. 952215).

We thank the reviewers for their feedback and suggestions, in particular for a suggestion that may improve and simplify the pose computation.

References

1. Argus, M., Hermann, L., Long, J., Brox, T.: FlowControl: Optical Flow Based Visual Servoing. In: IROS. pp. 7534–7541 (Oct 2020)
2. Austin, D.: Generate stepper-motor speed profiles in real time. EMSP **1** (2005)
3. Bajpai, P.: Chapter 24 - process control. In: Biermann's Handbook of Pulp and Paper, pp. 483–492. Elsevier, 3rd edn. (2018)
4. Bateux, Q., Marchand, E., Leitner, J., Chaumette, F., Corke, P.: Training Deep Neural Networks for Visual Servoing. In: ICRA. pp. 3307–3314 (May 2018)
5. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection (2020)
6. Chaumette, F., Hutchinson, S.: Visual servo control. I. Basic approaches. IEEE Robotics Automation Magazine **13**(4), 82–90 (Dec 2006)
7. Chaumette, F., Hutchinson, S., Corke, P.: Visual Servoing. In: Springer Handbook of Robotics, pp. 841–866. Springer (2016)
8. Chiacchio, P., Chiaverini, S., Sciavicco, L., Siciliano, B.: Closed-Loop Inverse Kinematics Schemes for Constrained Redundant Manipulators with Task Space Augmentation and Task Priority Strategy. IJRR **10**(4), 410–425 (Aug 1991)
9. Cong, V.D., Hanh, L.D.: A review and performance comparison of visual servoing controls. IJIRA **7**(1), 65–90 (Mar 2023)
10. Corke, P., Hutchinson, S.: A new partitioned approach to image-based visual servo control. IEEE Transactions on Robotics and Automation **17**(4), 507–515 (Aug 2001)
11. De Luca, A., Ferri, M., Oriolo, G., Giordano, P.R.: Visual Servoing with Exploitation of Redundancy: An Experimental Study. In: 2008 IEEE International Conference on Robotics and Automation. pp. 3231–3237 (May 2008)
12. Durdevic, P., Ortiz-Arroyo, D.: A Deep Neural Network Sensor for Visual Servoing in 3D Spaces. Sensors **20**(5), 1437 (Jan 2020)
13. Hofmann, T., Limpert, N., Mataré, V., Ferrein, A., Lakemeyer, G.: Winning the RoboCup Logistics League with fast navigation, precise manipulation, and robust goal reasoning. In: RoboCup 2019. pp. 504–516. Springer (2019)
14. Hutchinson, S., Hager, G., Corke, P.: A tutorial on visual servo control. IEEE Transactions on Robotics and Automation **12**(5), 651–670 (Oct 1996)
15. Janabi-Sharifi, F., Deng, L., Wilson, W.J.: Comparison of Basic Visual Servoing Methods. IEEE/ASME Transactions on Mechatronics **16**(5), 967–983 (Oct 2011)
16. Jin, Z., Wu, J., Liu, A., Zhang, W.A., Yu, L.: Policy-based deep reinforcement learning for visual servoing control of mobile robots with visibility constraints. IEEE Transactions on Industrial Electronics **69**(2), 1898–1908 (2022)
17. Jokić, A., Petrović, M., Miljković, Z.: Semantic segmentation based stereo visual servoing of nonholonomic mobile robot in intelligent manufacturing environment. Expert Systems with Applications **190**, 116203 (2022)
18. Kavraki, L.E., LaValle, S.M.: Motion Planning. In: Springer Handbook of Robotics, pp. 139–162. Springer (2016)
19. Knoflach, L., Kohout, P., Imhof, S., Rohr, A., Swoboda, D., Viehmann, T.: RoboCup Logistics League: Rules and Regulations 2022 (2022)
20. Lampe, T., Riedmiller, M.: Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In: IJCNN. pp. 1–8 (Aug 2013)
21. Lee, A.X., Levine, S., Abbeel, P.: Learning Visual Servoing with Deep Features and Fitted Q-Iteration. In: ICLR (Jul 2022)

22. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: ECCV (2014)
23. Liu, H., Li, D., Jiang, B., Zhou, J., Wei, T., Yao, X.: MGBM-YOLO: a Faster Light-Weight Object Detection Model for Robotic Grasping of Bolster Spring Based on Image-Based Visual Servoing. JINT **104**(4), 77 (Apr 2022)
24. Liu, J., Balatti, P., Ellis, K., Hadjivelichkov, D., Stoyanov, D., Ajoudani, A., Kanoulas, D.: Garbage Collection and Sorting with a Mobile Manipulator using Deep Learning and Whole-Body Control. In: Humanoids. pp. 408–414 (Jul 2021)
25. Machkour, Z., Ortiz-Arroyo, D., Durdevic, P.: Classical and Deep Learning based Visual Servoing Systems: a Survey on State of the Art. JINT **104**(1) (Dec 2021)
26. Malis, E., Chaumette, F., Boudet, S.: 2 1/2 D visual servoing. IEEE Transactions on Robotics and Automation **15**(2), 238–250 (Apr 1999)
27. Mansard, N., Chaumette, F.: Tasks sequencing for visual servoing. In: IROS (2004)
28. Mansard, N., Chaumette, F.: Directional Redundancy: a New Approach of the Redundancy Formalism. In: IEEE CDC. pp. 5366–5371. Seville, Spain (2005)
29. Mansard, N., Chaumette, F.: Task Sequencing for High-Level Sensor-Based Control. IEEE Transactions on Robotics **23**(1), 60–72 (Feb 2007)
30. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed. In: Automation, Communication and Cybernetics in Science and Engineering 2015/2016, pp. 605–618. Springer International Publishing (2016)
31. Sadeghzadeh, M., Calvert, D., Abdullah, H.A.: Self-Learning Visual Servoing of Robot Manipulator Using Explanation-Based Fuzzy Neural Networks and Q-Learning. Journal of Intelligent & Robotic Systems **78**(1), 83–104 (Apr 2015)
32. Saxena, A., Pandya, H., Kumar, G., Gaud, A., Krishna, K.M.: Exploring convolutional networks for end-to-end visual servoing. In: ICRA. pp. 3817–3823 (2017)
33. Singh, K.K., Lee, Y.J.: Hide-and-Seek: Forcing a Network to be Meticulous for Weakly-Supervised Object and Action Localization. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 3544–3553 (Oct 2017)
34. Tokuda, F., Arai, S., Kosuge, K.: Convolutional neural network-based visual servoing for eye-to-hand manipulator. IEEE Access **9**, 91820–91835 (2021)
35. Ulz, T., Ludwiger, J., Steinbauer, G.: A robust and flexible system architecture for facing the robocup logistics league challenge. In: RoboCup 2018. pp. 488–499. Springer (2018)
36. Viehmann, T., Limpert, N., Hofmann, T., Henning, M., Ferrein, A., Lakemeyer, G.: Winning the RoboCup Logistics League with Visual Servoing and Centralized Goal Reasoning. In: RoboCup 2022:. pp. 300–312. Springer (2023)
37. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Scaled-yolov4: Scaling cross stage partial network (2021)
38. Yu, C., Cai, Z., Pham, H., Pham, Q.C.: Siamese Convolutional Neural Network for Sub-millimeter-accurate Camera Pose Estimation and Visual Servoing. In: IROS. pp. 935–941 (Nov 2019)
39. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: CutMix: Regularization Strategy to Train Strong Classifiers With Localizable Features. In: ICCV. pp. 6023–6032 (2019)