Dexterous Manipulation Using Hierarchical Reinforcement Learning

Stephan Schädle[1] and Wolfgang Ertel[2]

***Abstract*— On a novel pneumatic four-finger gripper with three degrees of freedom per finger we apply reinforcement learning to learn dexterous manipulation of objects. In order to reduce the search space, we implemented hierarchical learning on two levels. Low-level learning is used for basic movement primitives like grabbing, lifting or rotation of an object around three cartesian axes, whereas in high-level learning we use the already learned low-level actions to find a policy that enables the gripper to move a target point on the surface of a sphere to the top position in a few seconds. It turns out that Q-learning with a finite state- and action space solves the learning task very well. Additional videos are available at [1].**

Note to the reviewers: The password for the videos is: DMUHRL (initials of the paper title). Access to the videos will be opened prior to the workshop.

I. Introduction

Flexible and dexterous manipulation of objects is getting more and more important for industrial applications. Motivated by this goal, the FESTO company is doing research on pneumatic gripper technology. Among others, a four-finger gripper with three pneumatic actors per finger has been developed. The gripper shown in Fig. 1 is inspired by an elephant's trunk [2]. In contrast to classical pneumatic cylinders, the blue pneumatic actors shown in Fig. 1 can provide soft forces. The gripper comprises four crosswise-mounted pneumatic fingers and has twelve degrees of freedom leading to a high-dimensional state and action space. The skills and versatility of the human hand served as a guidance during the development of the gripper.

Possible applications of the gripper are the classical bin-picking task where the gripper picks a detected object with arbitrary orientation from a bin, grasps it, rotates it without releasing into the desired orientation and then puts it to the desired location on a conveyor belt. A similar application is the placement of a piece of fruit, e.g. an apple, with its nicest-looking side on top into a fruit box pallet.

As a demonstration of such dexterous skills, our goal was to show that the new pneumatic gripper is able to learn the following task: Initially, a sphere lying on the bottom of the gripper has to be lifted and manipulated, i.e. rotated, until a defined spot on the surface of the sphere (the FESTO logo in Fig. 1) has been moved to the top in a defined orientation. After the initial lift-up, during the whole task, the sphere is no longer allowed to touch the bottom.

Solving such tasks involves, even for humans, nontrivial sensory motor skills. Thus, programming a robot gripper manually for such tasks is a challenging and time-consuming task. We show how to successfully apply reinforcement learning on the pneumatic gripper to the above defined sphere orientation task. In order to reduce the 12-dimensional state- and action-spaces, we adopt a hierarchical approach with two levels. We use low-level learning for basic movement primitives like lifting the sphere or rotation around three cartesian axes. In high-level learning we use the already learned low-level actions to find a policy that enables the gripper to move the target point on the surface of the sphere to the top position.

[1,2]Institute for Artificial Intelligence, University of Applied Sciences Ravensburg-Weingarten, Germany, schaedst@hs-weingarten.de, ertel@hs-weingarten.de

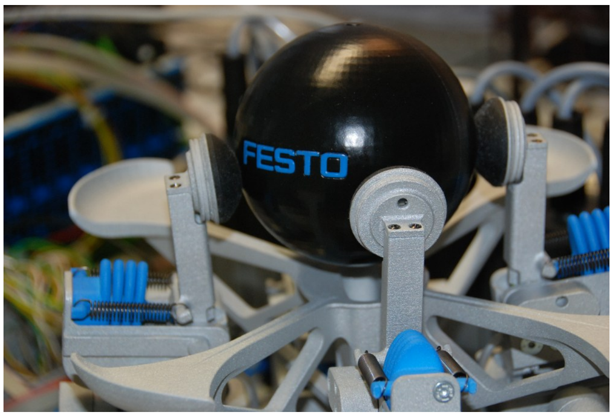

Fig. 1 – FESTO Learning Gripper

II. Related Work

Current industrial applications use multi-finger grippers for grasping predefined objects in a well-defined orientation. Object orientation and manipulation are not realized with these grippers. The gripper is usually mounted to an external robotic arm and the object pose and position relative to the gripper are fixed.

Many different multi-finger grippers have been developed since the late 70's [3]. Only few of these grippers have been equipped with learning algorithms yet. In [4] a simple two-fingered gripper learns to grasp objects with appropriate force without slip. The authors compare different methods with the result that a hybrid combination of learning from demonstration and reinforcement learning works best. In [5] a quite general approach using support vector machines is used to learn optimal grasps of complex objects with the GraspIt! simulator.

The new task presented here requires simultaneous grasping and manipulation requiring more than two fingers [6]. The Fraunhofer Vision Institute developed a flexible three-finger gripper [7] grasping objects of different sizes by using tactile sensors on the fingertips and image processing. This approach and the three-finger approach in general do not allow all manipulation actions [8].

Thus, grasping and manipulation of objects on a four-finger gripper is a new and challenging area. Due to the complexity of the possible motion patterns of such a quite complex gripper, the authors believe that machine learning is among the most interesting approaches to intelligent object manipulation.

Hierarchical reinforcement learning is the key approach for handling the huge state-spaces of the gripper by splitting up the task into subtasks[9], [10].

III. Hardware and Position Control

Only on appropriate hardware good policies can be learned. Thus, for the mentioned dexterous skills the gripper hardware had to be optimized in various ways. Particularly important was the parallel development of hardware while the learning experiments were already running. This lead for example to better finger tip surfaces for tighter object grasping. It also lead to more reliable and durable hardware which is crucial for reinforcement learning because learning on real hardware takes its time and thus stresses the hardware.

Finally we achieved the following hardware properties: The finger tips establish a form-closure grasp due to a soft silicone surface on the finger tips (black in Fig. 1). The gripper construction is made of polyamide allowing rapid engineering and construction by selective laser sintering. The height information of the sphere above the bottom of the gripper is provided by an infrared distance unit in the socket of the gripper. The orientation of the sphere is tracked with an integrated inertial measurement unit (IMU) which is mounted inside the sphere and communicates the orientation of the sphere via bluetooth to the learning algorithm on the PC.

The pneumatic actors of the fingers are position controlled using a magnetic encoder in each axis. Underlying pressure is controled using a programmable logic controller (FESTO CECX) with ethernet and CANbus interface. Every actor can thus be manipulated into any position using a single command.

IV. Manipulation learning

Our approach to hierarchical reinforcement learning is divided into low-level and high-level learning. Low-level learning learns a set of basic manipulation strategies. Each one of these strategies defines a high-level action. The high-level learning process learns to achieve a target orientation using the pre-learned high-level actions. In contrast to other approaches such as [11] the previously learned high-level actions are no longer modified in high-level learning. We use model free temporal difference tabular learning agents. The popular Q-learning and SARSA algorithms with eligibility traces [12] have been applied. The results with discrete and continuous state space are compared in Section VI.

A. Low-level learning

1) State and action space: Our intuitive approach establishes a state discretization of the twelve-dimensional gripper. The number of states $n = 5$ for each actor defines the accuracy of the discrete state space with 5^{12} states. The action space is discretized, too. Using three primitive movements (positive, neutral, negative) for each actor, we end up with 3^{12} possible actions.

The discrete action space can be reduced using the opponent fingers in a mirrored way (here called finger pair). This pre-knowledge is considering human object manipulation tactics. Therefore the number of actions can be reduced dramatically using actions where on each finger-pair on both fingers one actor moves simultaneously. The basic movements of a finger pair are:

$$\left.\begin{array}{r}1 = \text{left}\\ 2 = \text{right}\end{array}\right\}\text{ actor 1}\qquad \left.\begin{array}{r}3 = \text{forward}\\ 4 = \text{backward}\end{array}\right\}\text{ actor 2}$$

$$\left.\begin{array}{r}5 = \text{bend}\\ 6 = \text{stretch}\end{array}\right\}\text{ actor 3}\qquad 7 = \text{pause}$$

An action of the system with action i for pair 1 and j for pair 2 is denoted (i, j). The action space thus involves $7 \times 7 = 49$ actions. For example action $(1, 4)$ moves finger-pair one left and finger-pair two backward. All other axes stay fixed. An example visualization of low-level actions can be found in Fig. 3.

The goal is to learn subpolicies for basic manipulation skills (e.g. lift-up, rotate) used as high-level actions.

2) Reward: The reward is defined by the movement of the object and its manipulation height. A rotation in positive z-axis direction has the following reward function. Initially the reward for each step is negative. If the object is lifted higher than γ an additional reward ϵ is generated. The lifted reward also includes the angle change Δ_z of the performed action.

$$r = \begin{cases} -1 & \text{if } height < \gamma \\ r + \epsilon + \Delta_z & \text{if } height >= \gamma. \end{cases} \tag{1}$$

To ensure manipulations with no contact to the bottom, only states higher than γ are rewarded. The reward is dependent on the number of actions in a learning episode. Therefore the reward is divided by the number of steps in each episode in order to obtain a measure independent of the episode length. The episodes ends when the object has been dropped. In Fig. 5 and 6 the resulting reward per step is calculated every 25th episode showing the learning process. In these experiements a rotation skill around the z-axis is learned.

B. High-level manipulation learning

High-level learning finds an optimal policy in order to implement the lift-up task, the balancing above the socket and the correct manipulation for moving the target point on the surface of the sphere up to the top.

1) State and action space: Using the height information as an additional state information, the learning process can distinguish between three possible target object height states (down, medium, high). Different states will lead to different actions selected in high-level learning. If the object is on the bottom, an action leading to height gain is performed. On the other hand, the object can be too high and the grasp therefore be unstable. The grasp has to be readjusted to lose height. Only if the object is on the desired height, the manipulation is performed.

The high-level agent has to choose the correct manipulation direction for moving the given surface point to the top. Here the orientation of the sphere is required. In order to keep the state space for RL small, we reduce the infinite number of threedimensional orientations given by the IMU to four quadrants as shown in Fig. 2. The actual quadrant is given by the direction the target point faces. A quadrant is calculated using a vector being transformed form the Euler angles into the cartesian xyz system [3], whereby $c_\theta = \cos(\theta)$ and $s_\theta = \sin(\theta)$

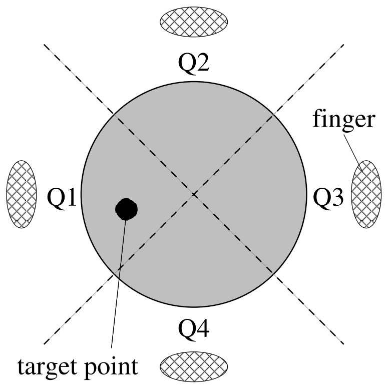

Fig. 2 – Definition of the four quadrants for the location of the target point to be rotated upwards.

$$\begin{pmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{pmatrix} \quad (2)$$

The following manually selected high-level actions are suitable for a high-level object orientation task.

1) lift-up
2) move first quadrant up = pos. rotation around x-axis
3) move second quadrant up = pos. rotation around y-axis
4) move third quadrant up = neg. rotation around x-axis
5) move fourth quadrant up = neg. rotation around y-axis
6) rotate right
7) rotate left
8) open grasp = lowering

Low-level actors learned to rotate the object around the z-axis or move a point in a given quadrant upwards. The rotation strategy of high-level action number seven can be seen in Fig 3.

V. Simulation

In order to save time, to preserve the hardware and for a better understanding of the physics, a gripper simulation was developed. Another advantage is the parallel evaluation of multiple learning algorithms. For this task we evaluated the simulation tools OPENRave [13], Gazebo [14] and GraspIt! [15]. Most tools have their benefits in

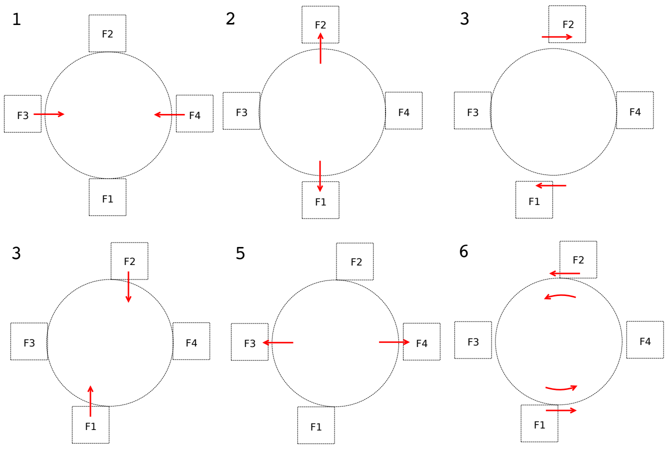

Fig. 3 – High-level rotation (e.g. move the object around the z-axis). A rotation strategy found by the agent repeats the following low-level actions: (3,7), (7,4), (7,1), (7,3), (4,7), (7,2).

high-level planing or inverse kinematics not required here. Direct physics simulation is the best way of manipulating physics configurations to achieve a plausible behavior with soft contact fingertips. Therefore for the gripper simulation we used the ODE physics engine. A drawback of rigid body physics is the collision behavior of solid objects. This problem can be solved using contact joints with a spring-damper system. The simulation uses the pyOpenGL and the pyODE wrapper for rapid prototyping and visualizing the manipulation results. The simulation is used in an RL_glue [16] environment. The target object and the 3D gripper model can be seen in Fig. 4. Any hardware changes can be simulated (e.g. top mounted gripper) before the hardware is actually changed.

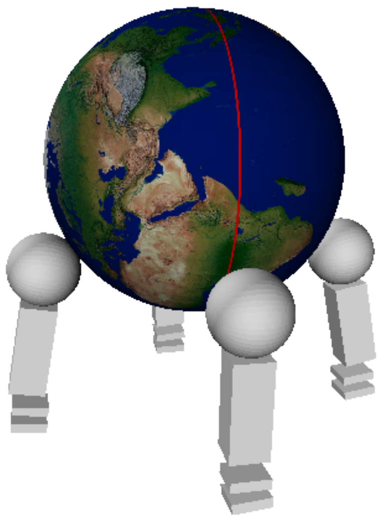

Fig. 4 – Gripper simulation.

VI. Experimental Results

Due to the dynamics of the pneumatic system, the real control is 20 times slower than the simulation.

A. *low-level manipulation experiment*

The learning result of a z-axis rotation can be found in Fig. 5 using different model-free temporal-difference tabular learning agents.

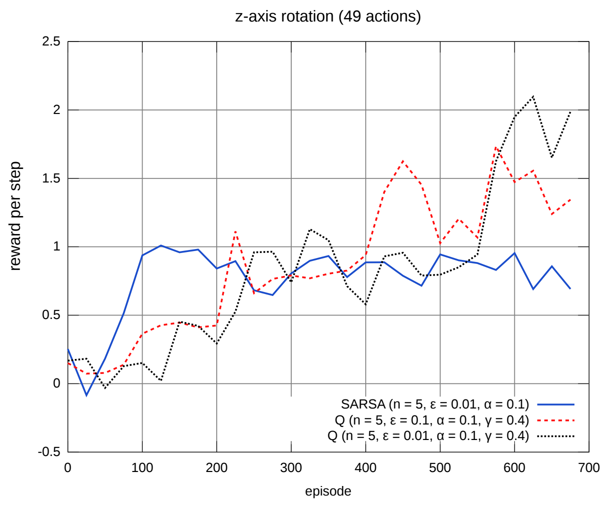

Fig. 5 – Learning high-level action (discrete state space). The reward per step shows the amount of rotation a single action has performed in the goal direction. The curves show single execution of different discrete learning agents with different learning parameters.

1) Continuous vs. discrete state space: By using a state discretization, the tabular learning algorithm leads to a memory consumption problem of storing a $Q(s, a)$-table of size $5^{12} \times 49$ (45634 MB). Moreover, additional information is not yet included (object height, force sensor of fingertips). Thus we compared the following methods for approximating the Q-table:

- TileCoding
- RBFs [17]
- LSPI (Least-Squares Policy Iteration [18])

We applied TileCoding[12] with 8 partitions and 16 tilings. The compared RBF learning agent uses $4^{12} \times 49$ radial basis functions to learn value function approximations. The simulation results are shown in Fig 6. The TileCoding-based agent has robust learning results with the average reward per step being comparable to the tabular learning algorithms (see Fig. 5). Therefore a tabular agent using a discrete state space was selected for low-level learning.

B. *High-level manipulation experiment*

As described earlier, high-level learning uses the previously learned low-level actions. This is an episodic discrete task. After 50 learning episodes the agent converges towards a robust policy (see Fig. 7). The robustness is quite high. Once the object is lifted initially, the gripper may drop the

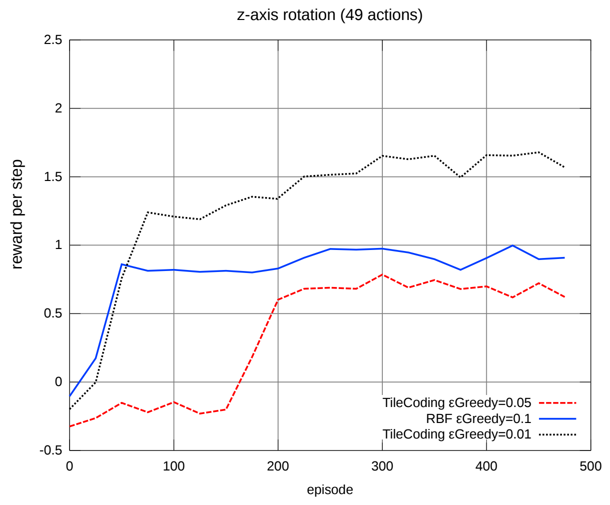

Fig. 6 – Learning high-level action (continuous state space). The reward per steps shows the amount of rotation a single action has performed in the goal direction. The curves show different continuous learning agent with different learning parameters.

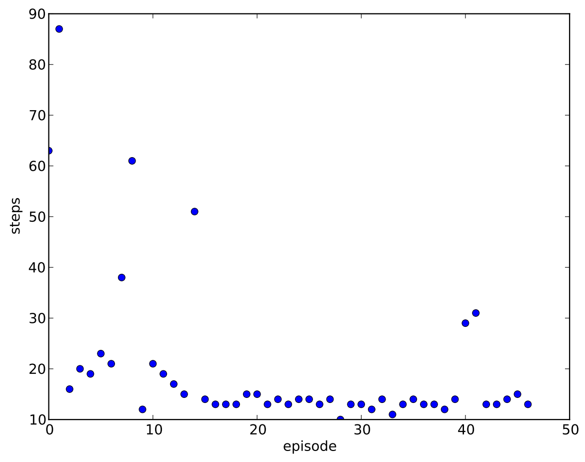

Fig. 7 – Learning high-level manipulation for a given goal orientation. The blue dots represent the required number of high-level actions to achieve the goal orientation. Outliers are due to the sphere being dropped during manipulation. After about 15 episodes a robust strategy has been found.

sphere in one of about 200 attempts, what can be seen in Fig. 7 after 40 episodes. When the object hits the bottom, the agent needs more steps to lift the object again to the required manipulation height.

C. *Learning time*

Manipulation of the sphere to the desired orientation with the target point on top requires about 12 high-level actions and about 60 low-level actions. Due to hardware speed limitations this results in a time of about 30 seconds. The complete learning phase takes around 9 hours considering that all eight high-level actions are learned separately. The learning process of the high-level manipulation requires only one hour.

VII. Conclusion

This challenging project involved development of the gripper hardware by the FESTO company in parallel to policy learning by the university partner. For a task as difficult as object manipulation with four fingers, learning has to be done on the real hardware. Simulation can help for tuning the algorithms, but the final policy has to be trained on the real hardware, even though learning times on hardware are much longer than in the simulation.

Thus, no good policy can be learned without good hardware. The gripper hardware has to run stable during the whole learning phase which may take many hours.

Hardware optimization and policy learning had to be done in an efficient alternating process. During the learning phases hardware bugs and deficiencies became obvious. In order to save long idle times in the project, new hardware versions were needed quickly. In the ten month project we worked with about nine different major hardware releases and many minor modifications of the hardware. The problem was that good, successful policies could be learned only on the last two hardware releases, whereas on the previous releases learning only served as a means to uncover the bugs and weaknesses of the hardware. This project was successful in such a short time because of two reasons:

1) After any new hardware release (almost) no software development was necessary. Only the training with RL had to be rerun which led to new hardware improvements.
2) Due to a very efficient hardware development process involving a tight coupling of CAD development and production of new parts via selective laser sintering, new major releases of the gripper hardware could often be built within one week.

In the future we want to learn initial policies in the simulation which are then transferred to the hardware and improved by further RL. We may also provide hand-coded initial policies to be improved by RL on the hardware. Our current policy learning approach works with an underlying classical state control. As an alternative, we want to use RL for learning basic actions which directly control the pressure levels on the valves.

Acknowledgments

The authors would like to thank the FESTO Bionic Learning Network contributing the gripper hardware and its team Nadine Kärcher, Elias Knubben, Arne Rost and Andreas Gause. Special thanks to Vien Ngo, Michel Tokic and Benjamin Stähle for fruitful discussions as well as Tobias Fromm for his generous support.

References

[1] S. Schädle and W. Ertel, "Videos of the learning gripper." February 2013, http://kids.hs-weingarten.de/learning-gripper/videos/.

[2] W. Stoll, *Bionik: Impulsgeber der Technik*, 1st ed. Schmidt, 10 2012.

[3] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer, 2008.

[4] J. Dominguez-Lopez, R. Damper, R. Crowder, and C. Harris, "Adaptive neurofuzzy control of a robotic gripper with on-line machine learning," *Robotics and Autonomous Systems*, vol. 48, no. 23, pp. 93–110, Sep. 2004.

[5] R. Pelossof, A. Miller, P. Allen, and T. Jebara, "An SVM learning approach to robotic grasping," in *IEEE International Conference on Robotics and Automation*, vol. 4, 26-may 1, 2004, pp. 3512 – 3518 Vol.4.

[6] N. Daoud, J. P. Gazeau, S. Zeghloul, and M. Arsicault, "A real-time strategy for dexterous manipulation: Fingertips motion planning, force sensing and grasp stability," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 377–386, Mar. 2012.

[7] W. Weller, "Intelligenter, flexibler Drei-Finger-Greifer mit integrierter Sensorik," Stand: 07.10.2012, http://www.vision.fraunhofer.de/de/presse/72.html.

[8] C. Ferrari and J. Canny, "Planning optimal grasps," in *IEEE International Conference on Robotics and Automation*. IEEE, 1992, pp. 2290–2295.

[9] R. S. Sutton, D. Precup, S. Singh *et al.*, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[10] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 4, pp. 341–379, 2003.

[11] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of Artificial Intelligence Research*, no. 13, pp. 227–303, 2000.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, Mass.: MIT Press, 1998.

[13] R. Diankov and J. Kuffner, "OpenRave: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, 2008.

[14] B. Calli, M. Wisse, and P. Jonker, "Grasping of unknown objects via curvature maximization using active vision," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 995–1001.

[15] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics & Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, 2004.

[16] B. Tanner and A. White, "Rl-glue: Language-independent software for reinforcement-learning experiments," *The Journal of Machine Learning Research*, vol. 10, pp. 2133–2136, 2009.

[17] M. Schneider, "Reinforcement learning with RBF-networks," *Scientific Project, University of Applied Sciences Weingarten*, 2006. [Online]. Available: http://amser.hs-weingarten.de/dokumente/project_schneider_rlwithrbf.pdf

[18] M. G. Lagoudakis and R. Parr, "Model-free least-squares policy iteration," in *Neural Information Processing Systems*, 2001, pp. 1547–1554.