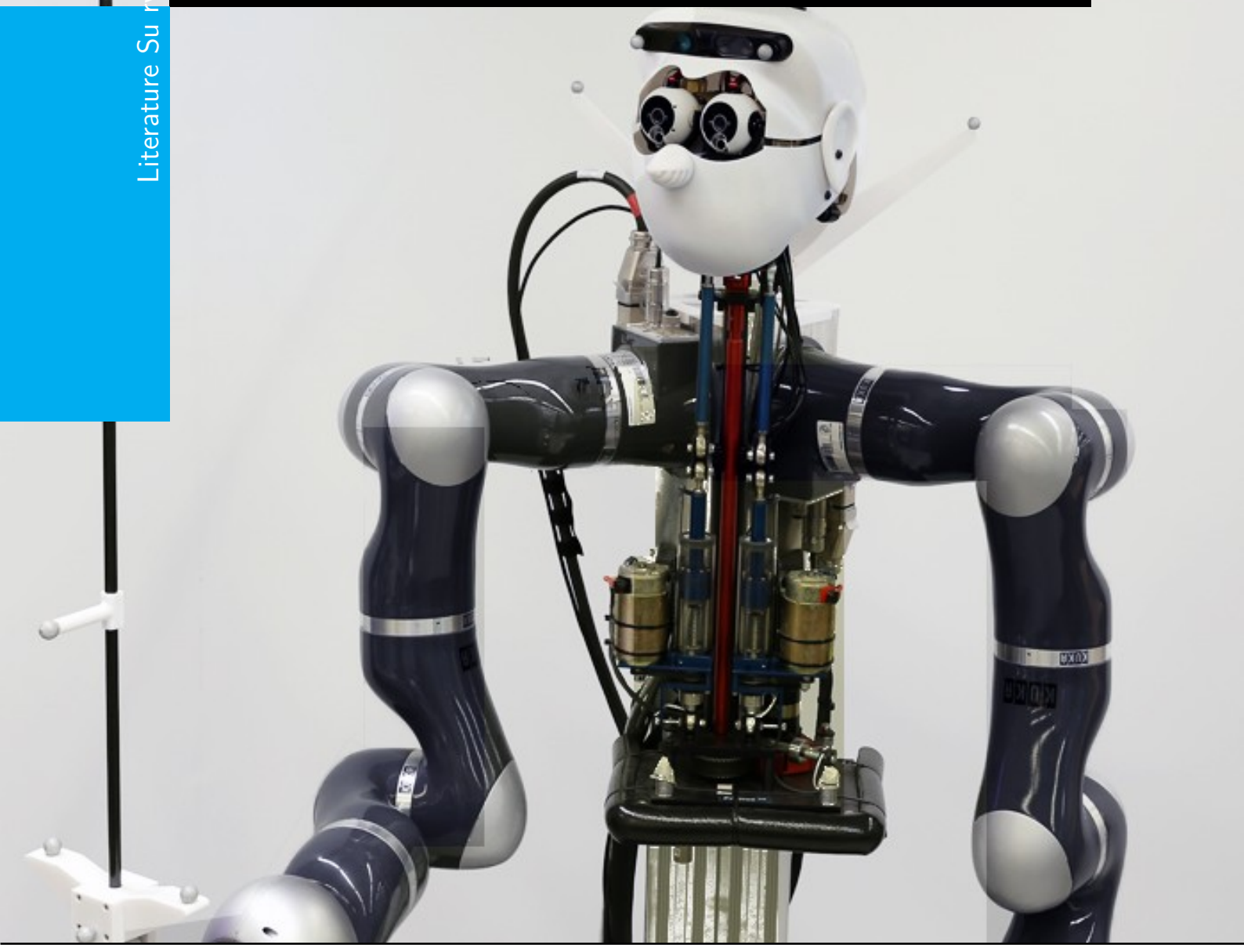
# Deep Lear ning of Vision-based Pole Balancing

A. Deichler

Literature Su rvey



TU Delft Delft University of Technology

DelftCenterforSystems andControl

# Deep Learning of Vision-based Pole Balancing

Literature Survey

A. Deichler

April 5, 2017

Faculty of Mechanical,Maritimeand Materials Engineering (3mE) ˙ Delft Univers ity of Technology

MAX-PLANCK-GESELLSCHAFT

TU Delft Delft University of Technology

DCSC

# Table of Contents

# Chapter 1

# Introduction

For humans vision is the primary sensory system to obtain information about the world. The field of computer vision aims at translating capabilities of the human vision system to algorithms through mathematical techniques. In recent years there has been a lot of research conducted in visual-guided robotics. Although robots usually have multi-modal sensing systems, visual sensors play a crucial role in gathering information about the world around them. This is especially true for mobile robots in human environments, which in contrast to the repetitive tasks of industrial robots, are often faced with dynamic and unstructured environments. Visual-control has been applied for various robotics tasks, e.g.,mobile robot navigation [1], vision-based lateral control [2], automated warehouse applications [3], UAV applications [4].

Early visual-servoing approaches relied heavily on prior knowledge of the robot-camera system and used hand-crafted visual features with a fixed hand-tuned controller. It was assumed that the camera calibration parameters are correct and the robot forward kinematics model is known. However these assumptions do not hold in most cases.

In recent years there has been an increasing trend of incorporating robot learning techniques in visual servoing in order to increase flexibility and decrease the need for hand-crafted features. Learning techniques can be applied in several areas within visual servoing e.g., learning internal models, for automatic feature extraction and for obtaining a control strategy. Deep reinforcement learning (DRL) algorithms have been proven a promising approach to achieve pixel-to-torque control. DRL learning made its first widely successful appearance in 2012, when DeepMind used it for automating the Atari game [5]. It was demonstrated that control policies can be obtained directly from high dimensional input data by combining deep learning and reinforcement learning. Since then it has been shown to provide an improved performance in a number of different fields, including natural language processing [6], self-driving cars [7], and robotics [8], [9].

Figure 1-1 depicts the structure of document. This literature survey focuses on the approaches of incorporating learning in the vision-based control. For this it is necessary to discuss the visual servoing framework. The first part of this literature survey aims at giving an overview of

vision-based control strategies in robotics (Chapter 2). In the following chapters possible ways of including learning techniques in the visual servoing framework are introduced along with their related theoretical concepts and examples of applications. Chapter 3 focuses on learning control, with an emphasis on reinforcement learning algorithms. Chapter 4 discusses learning in perception, with a focus on deep learning techniques. In the last chapter vision-based Deep Reinforcement Learning is discussed, where learning is incorporated in both perception and control.

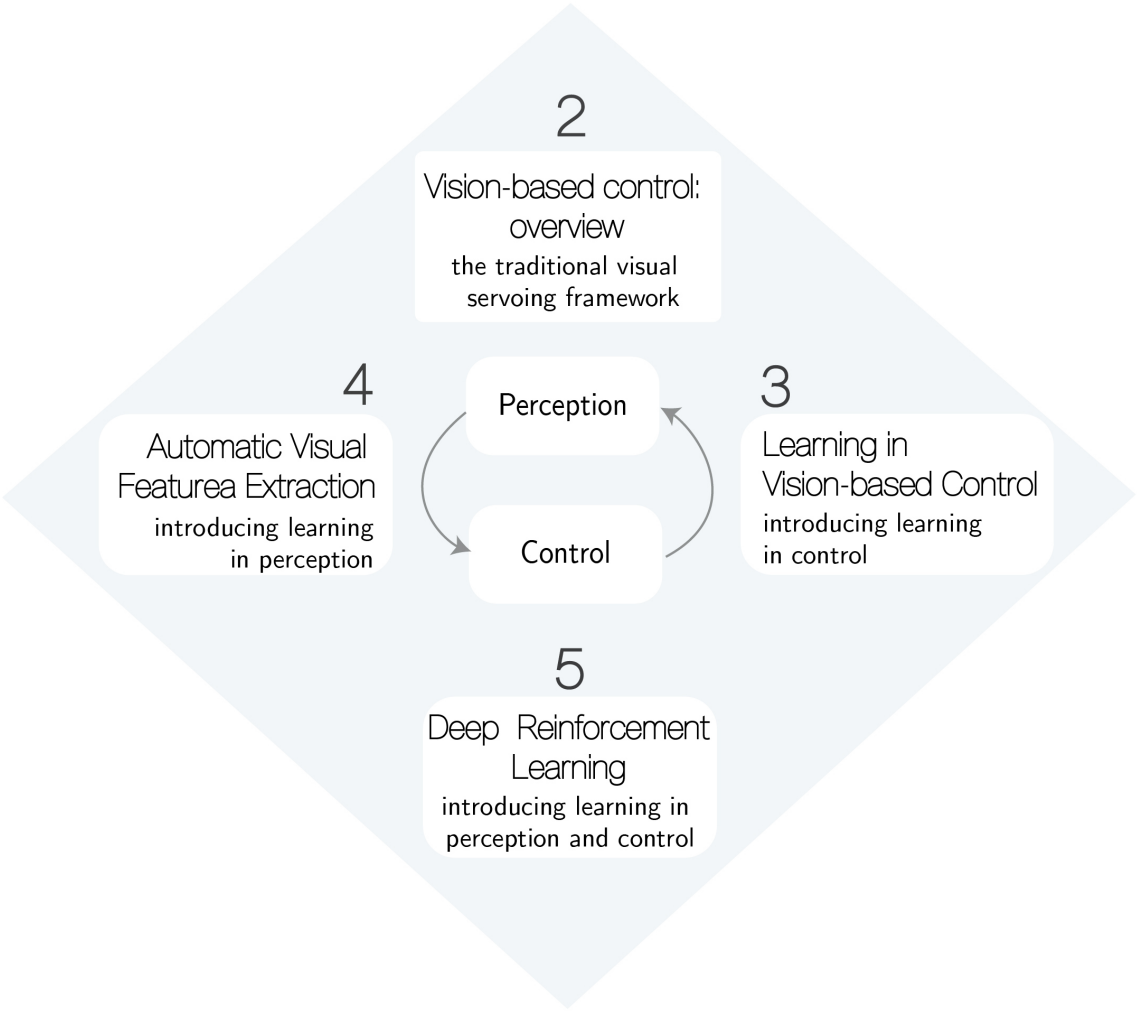



**Figure 1-1:** Document structure

# Chapter 2

# Vision-based control: overview

In visual servoing computer vision data is used to control the motion of the robot. Using visual information for control applications dates back to the 1970s, and since then it has been applied to a variety of problems [10] [11]. An important transition in visual servoing for robotics is moving from structured industrial environment to unstructured dynamic environments, where the robot has to explore its surroundings via sensory readings and adapt its behaviour based on these observations [12].

In the first visual servoing approaches the vision information was used in a complementary fashion to a low-level control system (e.g., joint-angle servo control system relying on encoder feedback information). The motivation for incorporating vision in the control system was to monitor the final position error, which is caused by parameter inaccuracies in the forward kinematic system [13]. Since then the focus in visual servoing research has shifted towards direct visual servoing, where visual information is used as the primary feedback information in the control system. When using directly the visual data, uncertainties from the camera system have to be accounted for in addition to the uncertainties from the robot kinematic model . Various approaches have been proposed to overcome problems with parameter uncertainties in the robot-camera system. e.g., Kalman-filter or neural network based parameter estimation algorithms.

There are several advantages of using only visual information as feedback in the control loop. Cameras are easily available and inexpensive sensors, which is an important aspect for mobile robots. They also provide rich information about the environment. On the other hand, extracting relevant information from high-dimensional observations is challenging, especially in real-time applications.

This chapter introduces the basic components and the terminology of the visual servoing framework, together with the basic control approaches and camera configurations. Then the main components and challenges in visual servoing are introduced. An overview of learning-free vision-based control attempts to the inverted pendulum problem is also given. Lastly, the chapter opens the question, where can improvements be made by using machine learning techniques.

## 2-1 The visual servoing framework

### 2-1-1 Coordinate frames and camera configurations

Robotic tasks are often defined with respect to one or more coordinate frames. Typically there are four coordinate frames distinguished in the visual servoing framework: the coordinate frame attached to the camera $\Phi_c$, the coordinate frame attached to the object (target) $\Phi_o$, the coordinate frame attached to the robot and effector $\Phi_e$ and the base frame of the robot $\Phi_0$.

The two main basic camera-robot configurations are the eye-in-hand and the eye-to-hand configurations. In the eye-in-hand configuration the camera is fixed on the robot, a constant relationship exists between the pose of the camera and the pose of the end-effector. In the eye-to-hand configuration the camera is in a fixed location in the world frame. The camera is often related to the robot base frame.

The eye-in-hand configuration (Figure 2-1) is usually used in mobile robot navigation, or in precision positioning tasks. In case of the eye-to-hand (Figure 2-2) configuration the geometric relationship between the workspace and the camera is fix, therefore offline calibration is possible.

Camera calibration is necessary in both cases of camera configurations in order to determine the intrinsic parameters of the camera, such as focal length, pixel pitch and principal point. The fixed camera pose with respect to the world coordinate system is also stored as the extrinsic parameters of the camera. In case of the eye-in-hand configuration, the calibration between end-effector and camera is also stored.

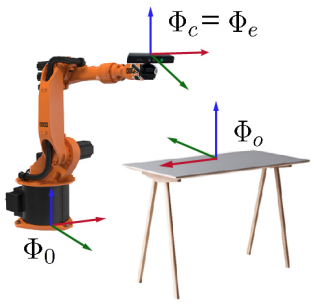



**Figure 2-1:** Eye-in-hand visual servoing configuration

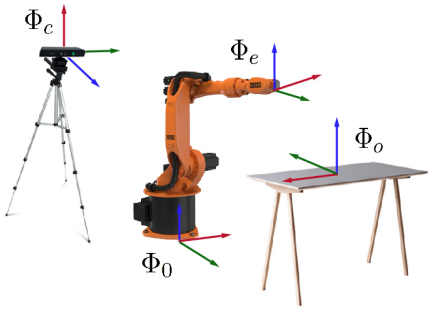



**Figure 2-2:** Eye-to-hand visual servoing configuration

### 2-1-2 The error function

In traditional visual servoing approaches, the aim is to actuate the robot to minimize the error $\mathbf{e}$ defined between a current and a goal configuration expressed in features ($\mathbf{f}$) derived from the image information. The error is typically defined as:

$$\boldsymbol{e}(t) = \boldsymbol{f}(\boldsymbol{m}(t), \boldsymbol{a}) - \boldsymbol{f}^*(t) \tag{2-1}$$

The $\mathbf{m}$(t) vector contains the image measurements for each time instant $t$, $\mathbf{a}$ is a set of parameters representing additional knowledge (e.g., 3D object model, camera intrinsic parameters). The visual features used for control is represented by $\boldsymbol{f}$, the desired target value of this features by $\boldsymbol{f*}$ [14].

### 2-1-3 Main approaches in visual servo control

Based on the used image information for control ($\boldsymbol{f}$), two main approaches are distinguished within visual servo control.

**Image-based visual servoing (IBVS)** uses 2D information directly for calculating the next best robot motion. The features used for control are defined in the 2D image plane in image features $\boldsymbol{s}$. This set of image features usually consists of image coordinates of points belonging to the target object, a projection of a physical feature of the target object to the camera plane.The parameters $\mathbf{a}$ are the camera intrinsic parameters, used for calculating features from image measurements. Image-based visual servoing control laws are based on the image Jacobian (interaction matrix), which represents the differential relationship between the camera frame motion and motion of features on the image plane. The image Jacobian can be derived analytically by the perspective projection model [15].

$$\dot{\boldsymbol{s}} = \Big(\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\Big)\frac{d\boldsymbol{r}}{dt} = J_s\dot{\boldsymbol{r}} \tag{2-2}$$

where $\boldsymbol{r} \in \mathbb{R}^3$ represents the camera's pose in Cartesian coordinates and $\boldsymbol{s} \in \mathbb{R}^n$ are the image features, and $\boldsymbol{n}$ is the number of used features. Most IBVS approaches compute the camera velocity sent to the controller [16]. This can be calculated as in 2-3.

$$\dot{\boldsymbol{r}} = \lambda(\hat{J_s^+}(\boldsymbol{s} - \boldsymbol{s}^*)) \tag{2-3}$$

Where $\hat{J_s^+}$ is an estimation of the pseudo-inverse of the image Jacobian $J_s$, and $\lambda$ is a proportional gain. The IBVS control scheme is depicted on Figure 2-3.

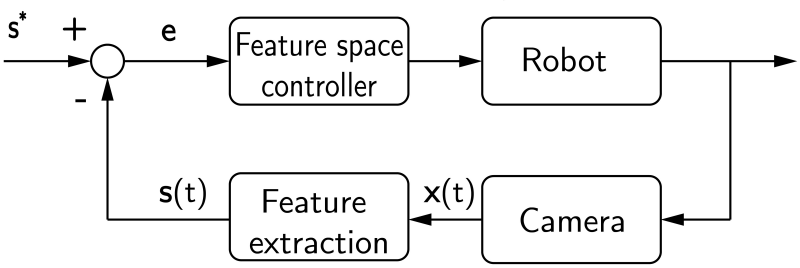



**Figure 2-3:** Image-based visual servoing

**Position-based visual servoing (PBVS)** uses 3D image information for control. Image measurements are used to estimate the object pose $\hat{\boldsymbol{r}}$ in Cartesian coordinates. In PBVS the task Jacobian is defined as:

$$J_{\hat{r}} = \left(\frac{\partial \hat{\boldsymbol{r}}}{\partial \boldsymbol{s}}\right)\left(\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{r}}\right) \tag{2-4}$$

The second term is the same image Jacobian as in 2-3. The first term represents the variation of the estimated pose as a function of the feature variation, which is unknown and depends on the visual features, shape of the object, camera calibration parameters and the used pose estimation algorithm. In this case the additional knowledge assumed to be known (**a**) are the camera intrinsic parameters, the 3D model of the object and the robot kinematic model. [16]. The PBVS control scheme is depicted on Figure 2-4.

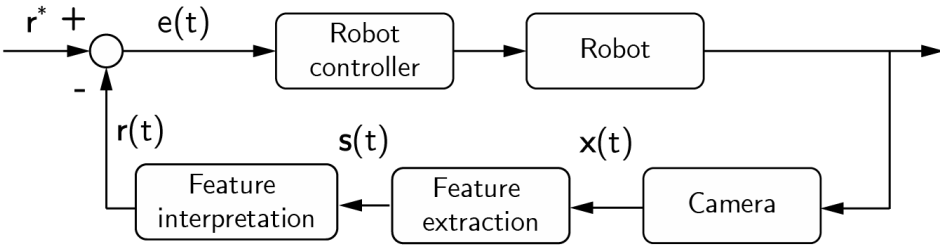



**Figure 2-4:** Position-based visual servoing (from [12])

The preferred visual sevoing approach used to be IBVS due to its better robustness properties [12]. Kinematic model uncertainties of the robot and calibration can be avoided since it does not require 3D reconstruction. However, sensitivity to lighting conditions and occlusion cannot be eliminated. In the past, PBVS methods had to rely on stereo triangulation to obtain depth information for pose estimation. This limited the applicability of PBVS methods, since feature extraction and correspondence matching require long processing times and do not work well in low-textured environments. Recently RGB-D cameras have become a popular alternative to obtain depth information in PBVS methods. These sensors are now available at low prices and provide fast depth information. Compared to stereo camera systems, RGB-D sensor are also better at handling occlusion and less sensitive to lighting conditions.

### 2-1-4 Tracking and control

The two subsystems of visual servo control are tracking (perception) and control (Figure 2-5). Earliest approaches in vision-based control used the "look-then-move" scheme, where the subtasks of visual information extraction and control are executed in sequence. In real-time applications it is necessary that the two subsystems are intertwined, executed continuously [11].

**Tracking** The tracking/perception subsystem extracts and tracks visual features. In real-time visual servo systems it is important to know the expected location of the target object. An overview of recent approaches is given in [17] about approaches for feature tracking in visual servoing.

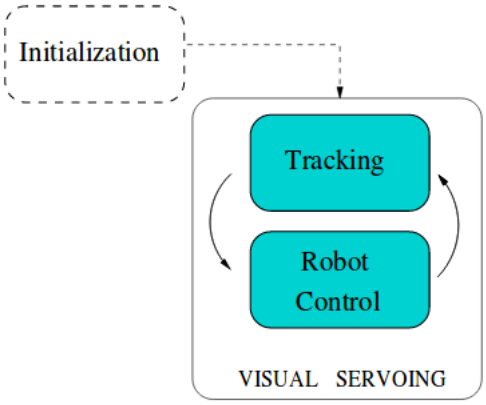



**Figure 2-5:** Main componentes of the visual servo system (from [11])

The two main approaches in tracking are 2D and model-based 3D tracking. 2D approaches are either feature-based, which use geometric features (pints, contours, shapes) or intensity-based, which use histograms. In 3D tracking all 6 Degrees of freedom (DoF) of the target object (position and orientation) are estimated continuously using a 3D model of the object.

In visual servoing the main requirements for the tracking algorithm are robustness, accuracy, speed and generality. 3D methods are in general more robust, since they can handle partial occlusion better and are less sensitive to lighting conditions than 2D methods. Incorporating a motion model can make the tracking more accurate in visual servoing applications. In Bayesian tracking the target location is estimated through a filtering process and the target movement is predicted using noisy observations (images) of the target's position and a dynamic model of the target's motion. Bayesian trackers are often used in visual servoing applications [18].

**Control** Traditional visual servoing control systems assume knowledge about kinematic parameters of the robot and the camera intrinsic and extrinsic parameters are known. The control strategies usually involve calculating the task Jacobian based on these parameters and deriving a control law for the end-effector velocity [19].

## 2-2 Literature overview on vision-based control of the inverted pendulum

With regards to the topic of this thesis, this section of the literature review gives an overview of vision-based control approaches for the inverted pendulum problem. Herein, approaches that do not or employ learning techniques are presented. A common factor amongst these publications is the use of an ad hoc experiment setting, which is designed to facilitate the recognition of the pendulum on the input images (e.g. homogenous background, using landmarks).

#### Fuzzy-logic Control of an Inverted Pendulum with Vision Feedback

In [20] a fuzzy-logic based controller is proposed for the cart-pole system. The tracking/perception system is highly dependent on the experiment set-up. The pendulum is painted black, while the background is white, the recognition of the pendulum is based on the contrast change in the image pixels using simple trigonometric functions. The fuzzy-logic controller uses a triangular membership function to fuzzify the measured angles. Fuzzy rules for the angle and angular velocity are based on heuristics, on the knowledge of the experimental setup. This vision based controller is limited in time to a few seconds of keeping the pendulum upright. The pendulum is managed to be balanced near the unstable equilibrium zone not exceeding 5 °, the control system is too slow to compensate larger deviations.

#### Vision-Based Control of an Inverted Pendulum Using Cascaded Particle Filters

In [17] a vision-based feedback controller for the cart-pole system, which uses Bayesian trackers to keep track of the position of the cart an the angle of the pendulum. Specifically, a cascaded set of particle filters are used, there are separate cameras for the cart location and the pole angle. Similarly to the previous approach, gradient based edge detection is used to locate the pendulum on the images. The measurement updates are provided at $30Hz$. Swing-up and balancing of the pole is managed in a real setting. However, on-site calibration of the camera system is needed to make it robust against distortions and the experiment set-up is designed to facilitate recognition.

## 2-3 Potential problems and uncertainties in visual servo control systems

The main problems related to tracking are the limits of the camera field of view and feature extraction from the images. Effective and accurate feature extraction can be impeded by illumination of the image scene, occluded target objects, or camera sensor noise. In IBVS approaches the features are limited to geometrical primitives (segments, contours), while the analytical form of the interaction has to be derived for them separately. This limits the applicability of these approaches to structured settings [16].

The main problems related to the control are inaccurate camera calibration, feedback delays from image processing and sensor measurement noise. As a result of these inaccuracies the image Jacobian is also inaccurate, leading to stability problems in IBVS control. Parameter inaccuracies also cause the camera pose estimate in the PBVS methods to be non-robust. Stability problems in PBVS and IBVS are discussed in [16].

## 2-4 Learning in the visual servoing framework

In the traditional vision-based robotic control pipeline (depicted on Figure 2-6) four main components can be distinguished: feature design, feature detection, feature tracking and control. All these elements required human engineering and domain knowledge in the past. In recent years many different approaches have been taken to incorporate machine learning into these steps in order to reduce the need for human-crafted designs. Learning approaches can also be used to overcome problems mentioned in the previous section.

Machine learning studies algorithms that can learn and make predictions from data. Machine learning can be applied to multiple problems within robotics. The intersection of robotics and machine learning is called robot learning, in which a robot has to acquire new skills, perform tasks autonomously in an unstructured environment through learning algorithms. In robot learning approaches relevant information from high dimensional sensory readings is automatically extracted to solve the robotic task [21].

The two sub-fields of robot learning, which are relevant for visual servoing are learning to perceive through visual sensory readings and learning control.

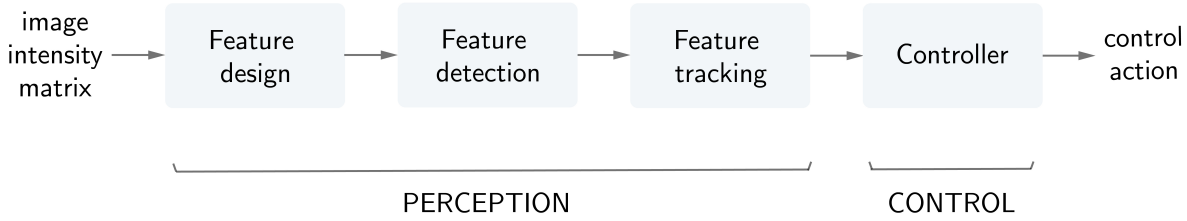



**Figure 2-6:** The traditional visual servoing pipeline

The following chapters investigate existing approaches of incorporating learning in the visual serving framework and introduces the necessary background information for these methods. Chapter 3 discusses learning control for vision-based robot control applications by introducing the reinforcement learning (RL) robotics framework, the main challenges related and the basics of RL algorithms which have been used in recent vision-based control publications. Chapter 4 discusses methods of incorporating learning in the perception part of the traditional vision-based robot control pipeline (feature extraction, detection, tracking). Deep learning is introduced as a form of representation learning and control relevant deep neural network architectures are introduced. State representation learning approaches within the RL are also discussed. In Chapter 5 recent publications incorporating learning in perception and control are presented.

# Chapter 3

# Learning control in visual servoing

One of the main drawbacks of traditional visual servoing control methods is that they assume prior knowledge about the robot forward kinematics, the camera model and use fixed controllers with hand-tuned gains. These robot and camera models however are not always available, inaccurate and can vary over time. There have been different approaches proposed in the past to overcome these problems, such as uncalibrated visual servoing methods and adaptive controller methods. Learning control frameworks provide a way to increase the flexibility of the control system. In learning robot control the aim is to acquire a sensory-motor control strategy for a particular task and system by trial and error [21].

In the visual servoing framework teacher-based and critic-based learning algorithms have been both applied. Teacher based learning methods include learning by demonstration and supervised learning. In the former approach the teacher provides a model of 'good behaviour', which usually means a video sequence of the movement to be learned [22]. In supervised learning methods the provided training data consists of input-output pairs of the function to be learned [23]. Reinforcement learning (RL) algorithms on the other hand involve a critic, and no idealized behaviour is available. The control policy has to be learned by trial and error based on a scalar reward.

RL is an area of machine learning where an agent learns how to behave from interacting with the environment in order to achieve a desired goal [24]. Traditionally RL algorithms have been applied and benchmarked on discrete, small scale problems. The application of RL to robotics problems poses several challenges e.g., the high dimensionality, uncertainty of real physical systems, sensory delays [25].

This chapter focuses on the application of RL to visual servoing problems. First the theoretical framework of RL is introduced and a basic taxonomy of the RL approaches is provided. Then the challenge of the high-dimensionality of vision-based robotics problems is discussed with a brief overview of recently proposed solutions, which is followed by a more detailed description of these algorithms. Finally, application examples are given in a literature overview of approaches in the visual servoing context, which use learning in control, but not in perception.

## 3-1 The RL problem robotics formulation

Formally the RL problem can be introduced in the framework of a Markov decision process (MDP). An MDP is a mathematical framework for decision making problems and can be described by a tuple of five elements: $M = \langle X, U, P(\cdot,\cdot), \rho(\cdot,\cdot), T \rangle$ , where

- $X$ is a set of states, called the state space
- $U$ is a set of actions, called the action space
- $P$ is the state transition probability function, it maps each state-action pair to a distribution over successor states: $P : (X \times U \times X) \to [0, \infty)$
- $\rho$ is the reward function $r : X \times U \times X \to \mathbb{R}$
- $T$ is the time horizon

State transitions of an MDP have to satisfy the Markov property, the next state depends on the current state and action.

The two main components of the RL problem are the agent and the environment (Figure 3-1). The agent is the decision-maker, the environment is everything that is external to the agent. The agent senses the environment states ($\mathbf{x}$) and takes actions ($\mathbf{u}$), which modify the environment states. The actions taken by the agent in a given state are evaluated by the task-specific scalar reward function (reinforcement) of the environment. The reward function can defined in different ways, e.g., $r_{t+1} = \rho(\boldsymbol{x}_t, \boldsymbol{u}_t, \boldsymbol{x}_{t+1})$, including the transitions, or as $r_{t+1} = \rho(\boldsymbol{x}_t, \boldsymbol{u}_t)$, depending on only the current state and action [25].

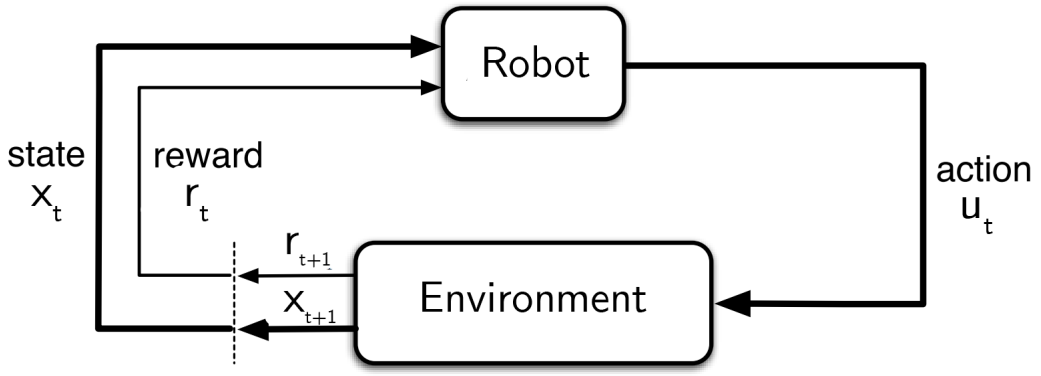



**Figure 3-1:** The RL problem, from [25]

In the visual servoing setting the agent is the robot, which has no access to the true state. It obtains only incomplete and noisy information about the world through high-dimensional sensory readings in the form of images. The learning algorithm has to with this incomplete state information. Memory is needed to be incorporated into the algorithm in order to determine states that are partially observable from successive states. These types of problems

can be modelled by a Partially Observable Markov Decision Process (POMDP). POMDPs are commonly used in robotics problems, since they allow reasoning about uncertainty, through the introduction of the belief state. Another option is to keep the MDP formulation and extend the state with the previous state(s). This leads to a higher dimensional state space, but algorithms for the MDP formulation can be applied, since the Markov property is retained.

The actions are motor commands ($\mathbf{u}$) and together with the states they form a trajectory $\boldsymbol{\tau}$=($\mathbf{x}_0, \mathbf{u}_0..$). Robotic tasks are often episodic, with an episode ending after $T$ time steps. The accumulative reward (return) collected along a trajectory $\boldsymbol{\tau}$ can be written in the following way:

$$R(\boldsymbol{\tau}) = \rho_T(\boldsymbol{x}_T) + \sum_{t=0}^{T-1} \gamma^t \rho(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{3-1}$$

, where $\gamma \in (0, 1)$ is the discount factor, rating recent rewards higher and $\rho_T$ is a final reward. The goal of the robot is to maximize the expected cumulative reward, which is given for the quality of the trajectory. This can be achieved by learning a mapping from states to actions. The mapping is called policy function $\pi$. In most robotic applications the policy $\pi$ is time-dependent and can be stochastic $\pi(\boldsymbol{u}_t|\boldsymbol{x}_t, \boldsymbol{u}_t)$. The policy is said to be optimal $\pi^*$ if it maximizes the expected accumulated reward for every initial state $\boldsymbol{x}_0$.

The expected return is denoted by $J_\pi$ and can be expressed in the following way:

$$J_\pi = \mathbb{E}[R(\boldsymbol{\tau}|\pi)] = \int R(\boldsymbol{\tau}) P_\pi(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{3-2}$$

,where $P_\pi(\boldsymbol{\tau})$ is the probability of a trajectory under the policy $\pi$ and state transition dynamics.

Every policy can be characterized by its unique state-action value function $Q^\pi : X \times U \to \mathbb{R}$. Computing the state-action value function for a policy is called policy evaluation.

$$Q^\pi(\boldsymbol{x}, \boldsymbol{u}) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} ... | \boldsymbol{x}, \boldsymbol{u}] \tag{3-3}$$

The state-action function gives the return for following the policy after an initial state-action pair. Under the optimal policy, the value of a state-action pair must be equal to the expected return for the best action from that state (This is expressed by the Bellman optimality equation).

In general the optimal policy $\pi^*$ can be found by solving the following optimization problem:

$$\pi^*(\boldsymbol{x}) \in \underset{\boldsymbol{u}}{\operatorname{argmax}} J_\pi \tag{3-4}$$

## 3-2 RL algorithm taxonomy

This section gives a brief taxonomy of the existing RL approaches in order to discuss algorithms in later sections. A more extensive overview is provided in [25], [26].

### 3-2-1 Model-based and model-free algorithms

Reinforcement learning algorithms can be classified as model-free (direct) or model-based (indirect) based on the steps of obtaining the policy [25].

**Model-free algorithms** In these algorithms the agent does not know the transition probabilities and reward function. The controller is obtained from directly interacting with the environment through learning the value function or the policy. One drawback of model-free learning algorithms is that that they require a large number of training episodes to find a solution [27].

**Model-based algorithms** assume an explicit model of the environment. The system model is jointly learned with the value-function/policy based on collected data. This method is also called indirect learning, because of the intermediate step of model learning (e.g. environment model). For robotics applications model-based algorithms have the advantage of requiring less interactions with the environment by learning from simulations in addition to learning from real interactions. The main drawback is that the unknown dynamics cannot be accurately modeled for effective policies [25],[21].

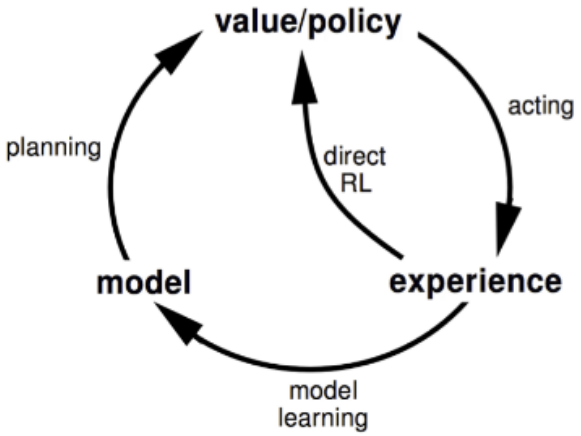



**Figure 3-2:** RL algorithms (from [24])

### 3-2-2 Policy-based and Value function-based algorithms

Three different groups of algorithms can be distinguished based on the data structure they maintain and update: value-function based algorithms, policy-based algorithms and actor-critic algorithms [25].

**Value-based algorithms** are also called critic-only algorithms. These algorithms first estimate the value function (3-3), the long term expected reward of a policy for each state and time step, then derive the controller (mapping from states to actions) based on the estimated value function. Value-function based RL algorithms can be classified into three categories: Monte Carlo methods, temporal difference (TD) methods and dynamic programming bases optimal control approaches.

**Policy-based algorithms** maintain a parametrized policy (actor) $\pi_{\boldsymbol{\theta}}$ and search for the optimal policy in the neighbourhood of the current policy. The mapping from observations to actions is learned directly, without estimating a value function. In robotic applications policy search methods have several other advantages over the value function methods, e.g. it allows the incorporation of expert knowledge through policy parametrization and initialization [25]. The existing policy search algorithms can be grouped into three categories: policy gradient, expectation-maximization and information theoretic [26].

**Actor-critic algorithms** incorporate the advantages of both value function-based and policy-based methods by maintaining and updating both the value function and the policy. The policy is explicitly maintained (e.g. function approximator) in addition to the value function for the current policy. The critic estimates the action-value function using a policy evaluation algorithm (e.g. Temporal-difference learning). The actor adjusts the policy parameters $\boldsymbol{\theta}$ by gradient ascent [25],[26].

### 3-2-3 Online and batch learning

Online RL algorithms continuously improve the policy as the environment is explored, the solution is estimated and the process is controlled simultaneously, updates are made after each new experience.

In batch algorithms at each iteration $i$ trajectories are generated and the data collected along the trajectory is stored $\boldsymbol{\tau}_t = \{\boldsymbol{x}_t^i, \boldsymbol{u}_t^i, r_t^i\}_{t=0}^T$ . Updates occur synchronously in the whole batch of transitions (e.g. value-fitted policy iteration [28]).

### 3-2-4 On-policy and off-policy learning

On-policy explore the environment using the current policy, therefore an exploration component has to be built in the policy in order to ensure proper exploration of the state space. This can be achieved by executing actions using soft policies. The value function can only be updated based on experience.

Off-policy methods use an exploration strategy that is independent of the current policy to learn about the environment. Different policies can be learned for exploration behaviour and value function estimation, which makes it possible to separate estimation from control. Off-policy algorithms can update the estimated value functions using hypothetical actions, not only based on experience as in the on-policy case [25].

## 3-3 The problem of high-dimensional state and action spaces

In vision-based control both action and state spaces are high-dimensional. Information about the environment is provided by images, making the observation/state space high-dimensional. The control action is also real valued and high-dimensional. Applying RL algorithms in large scale problems, such as games or real-world problems with continuous variables brings up several challenges and have been studied for a long time [25].

Value approximation based RL approaches for large observation spaces have been proposed. In case of value-function approaches with small action and state spaces the approximate value function can be represented by a table. For complex environments with high dimensional/continuous state and action spaces the table size can become very large, resulting in tabular representation being computationally intractable. There are two approaches to solve continuous-state RL problems: discretizing the state space with grids or using a function approximator. Discretizing the state/action space leads to computational intractability due to the curse of dimensionality. For higher-dimensional robotic problems function approximations (e.g., wire-fitting [29], neural networks) provide a more suitable alternative. Neural network based function approximations have the advantage of being differentiable. In neural network based value function approximations the tabular representation is replaced by a neural network, whose inputs are states and actions and the output is the utility of the state-action pair. This makes generalization from seen states to unseen states possible [25].

However by using neural networks as function approximators for the value-function, convergence and stability guarantees do not hold anymore [25]. It has been shown that NN based value-function approximation RL algorithms using online gradient descent do not converge [30]. Two different approaches have been proposed to resolve the stability problems in NN based value-function approximation RL algorithms. One of them is the Neural Fitted Q Iteration (NFQI) [28] algorithm, which will be presented in the next section along with Q learning. The other is the Deep Q Network (DQN) [5] algorithm using a deep neural network. This algorithm and its applications in robotics will be presented in Chapter 5, after discussing the basic ideas of deep learning in Chapter 4.

In the NFQI and DQN algorithms the action space is assumed to be discrete. Both value function based and policy based approaches have been proposed for dealing with continuous action spaces in robotics. The Neural Fitted Q-iteration with Continuous Actions (NFQCA) [31] is a value approximation RL algorithm, a variant of the NFQ algorithm for continuous action spaces. The double-DQN algorithm [32] is a variant of the original DQN network for continuous action spaces. The Deterministic Policy Gradient (DPG) algorithm [33] is a policy based algorithm, also for dealing with high-dimensional action spaces. The Deep Deterministic Policy Gradient (DDPG) [27] is and actor-critic RL algorithm with neural network function approximators, which is based on the DPG network. The Deep Deterministic Policy Gradient algorithm (DDPG) [27] has been applied successfully to real world robotics problems [9].

The DQN, NFQI or DPG/DDPG algorithms will be discussed in more detail, since many recent publications in vision-based robotics build on the variants of these algorithms. The NFQI and DPG algorithms do not use deep neural networks as function approximators, therefore there are presented in this chapter. The DDPG and DQN algorithms, which employ deep neural networks will be presented in Chapter 5, after the introduction to deep neural networks in Chapter 4.

## 3-4 Value function based algorithms in vision-based RL

### Q learning

Q learning is an off-policy temporal-difference (TD) value-function based algorithm, where the state-action values are incrementally estimated at each time step based on reward received from the environment and the agent's previous Q-value estimates. In tabular methods direct Q value assignment can be made, a table is maintained to store the Q-values for the state and action set pairs. In neural network value function approximations an error function is introduced, which measures the difference between the current Q value and the target Q value. The Q network is the neural network based approximation of the value-function:

$$Q(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w}) \approx Q_{\pi}(\boldsymbol{x}, \boldsymbol{u})$$

The mean-squared error in Q-values can be written as:

$$L(\boldsymbol{w}) = \mathbb{E}\left[\left(r + \max_{\boldsymbol{u}'} Q(\boldsymbol{x}', \boldsymbol{u}', \boldsymbol{w}) - Q(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})\right)^2\right] \tag{3-5}$$

### Neural-Fitted Q Iteration

The Neural Fitted Q Iteration (NFQ) algorithm [28] has been proposed to confront the stability problems arising from using gradient descent for weight parameter adjustment in 3-5. Classical Q learning estimates the Q function online, only the current environment feedback is used for weight adjustment. The NFQ algorithm uses a memory of a set of experiences, collected in tuples $\{\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}'\}$ to train the neural network instead of online updates. As a result batch NN learning techniques can also be applied [28].

The NFQ algorithm is one approach to confront the stability problems of the naive Q learning algorithm. The DQN algorithm is a different approach, which will be presented in Chapter 5.

## 3-5 Policy based algorithms and actor-critic algorithms in vision-based RL

Policy search methods use parametrized policies $\pi_{\boldsymbol{\theta}}$ and typically avoid learning a value function. In contrast to most value-function based RL algorithms they do not bootstrap, the experienced reward is directly used for quality assessment of state-action pairs. The three classes of algorithms are policy gradient (PG), expectation maximization (EM) and information theoretic [26].

### 3-5-1 The Deterministic Policy Gradient algorithm

Policy gradients have been proven to be a promising approach to confront the high/continuous dimensionality of large scale/robotics problems, since they do not suffer from the lack of guarantees of a value function, the intractability problem resulting from uncertain state

information and the complexity arising from continuous states and actions. In case of stochastic policy $\pi(\boldsymbol{u}|\boldsymbol{x})$ gradients, the policy gradient is estimated from data generated during the execution of a task [34]. Policy gradient methods are based on the policy gradient theorem [24]. The general goal in policy gradient algorithms is to optimize the policy parameters $\boldsymbol{\theta}$, so that the expected return is optimized:

$$J(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t r_t | \pi_{\boldsymbol{\theta}}\right] \tag{3-6}$$

The Deterministic Policy Gradient (DPG) algorithm [33] maintains a deterministic policy $\boldsymbol{u} = \mu_{\boldsymbol{\theta}}(\boldsymbol{x})$, which specifies the current policy by deterministically mapping states to a specific action and has been shown to outperform their stochastic counterparts in case of high-dimensional action spaces [33].

### 3-5-2 Guided Policy Search algorithm

The Guided Policy Search [35] (GPS) trains complex policies, such as deep neural network using supervised learning. Trajectory optimization is solved with RL separately from a fixed number of initial states. The outcomes of these separate trajectory optimization problems are linear Gaussian controllers, optimized for the given initial state and goal position. The final policy unifying the pre-trained controllers is obtained with supervised learning, which allows the use of a high-dimensional policy representation, such as a multilayer neural network. As a result complex behaviours can be obtained with good generalization.

## 3-6 Literature overview of RL based approaches in visual servoing without automatic feature extraction

This section provides a brief overview of approaches for visual servoing which use learning in control, but not in the perception part of the visual servoing pipeline (Figure 2-6). Up to recently, perception and control were not integrated tightly due to the lack of tools to deal with high dimensional, ambiguous an noisy sensory data. The perception systems in these approaches are simplified and use heavily preprocessed data [25].

Reinforcement learning has been used in many vision-based robotic applications: learning to shoot a ball ([36], target tracking with mobile robot[29] , for docking tasks [37], mobile robot obstacle avoidance [38] and for visual set-point regulation with an industrial robotic arm [39].

Among the chosen examples most use a variant of the Q learning algorithm:[36] uses the standard Q learning algorithm, [29] uses model-free NN based Q learning, [39] uses model-based an model-free RFQI algorithm, [37] uses model-based Q learning with the adjoining property from optimal control , with the exception of [38] which used a policy search algorithm. One common point of these RL based approaches is that they extract some high-level information about the observed scene that is relevant to the task, which is fed to the RL based controller.

#### Reinforcement Learning for Visual Servoing of a Mobile Robot

In [29] multi-layer perceptrons are used to learn wandering behaviour and visual servoing. A mobile robot is trained to wander and track a target in real time. The approach uses reinforcement learning to learn a direct mapping from image space and other state information to the actuator. The proposed algorithm uses Q learning with artificial neural network and wire-fitting for interpolation to face the increased dimensionality of the continuous state and action spaces. The NN input is the state, the output is set of actions and their values, which is a sample of the Q function. The wire-fitting interpolation method generalizes between these states. They achieve inferior performance compared to calibrated visual systems, but gain flexibility, since the algorithm still works if the camera is moved, or the system changes.

#### Fast Reinforcement Learning for Vision-guided mobile robots

In [37] a vision-based RL control solutions is presented for the problem of navigating a non-holonomic mobile robot to a table. State space discretization is proposed to confront the continuous state space problem in real world robotics application. To overcome discretization problems, the adjoining property is introduced to Q learning. The standard Q-learning assumes unordered states (nominal numbers), their approach exploits the natural ordering of sensory state spaces by only allowing transitions between neighbouring states. By enforcing the adjoining property and by applying memory-based sweeping they acquire faster Q learning . The approach requires no calibration, nor geometric models and is robust to perturbations and noise.

#### Model-based and Model-free Reinforcement Learning in Visual Servoing

In [39] an approach to uncalibrated visual servoing is presented for visual set-point regulation with a robotic Puma arm. First the visual-motor forward kinematics model is estimated using Gaussian kernel-based locally linear regression. Then the regularized version of fitted Q-iteration is used to find the optimal control policy. Both model-based and model-free algorithms are proposed. The model-based approach first estimates the visual-motor model and uses it to generate samples for the RFQI algorithm, while the model-free approach uses samples from directly interacting with the robot. Without tuning the regularization parameter in the RFQI algorithm and the kernel width parameter the RL based algorithms underperform the linear controller, which has access to the exact visual-motor model. Model selection needs to be applied to select the best model out of different regularization and kernel width parameters, then the RL based approaches outperform the linear controller.

### 3-6-1 Relevant learning control approaches to the project

This section presents two examples of learning control. These are relevant to the project, since they use the same robotic platform. Both employ automatic controller tuning for the inverted pendulum problem, learning approaches have been used to automatically tune controllers. [40] uses a model-free approach, while [41] uses a model-based approach for the controller tuning problem.

In [40] an automatic controller tuning framework for tuning state feedback controllers on a real a robot is proposed. The authors propose a parametrized LQR realization of an state feedback controller. An initial set of parameters is automatically tuned by solving an optimization problem addressed with Entropy Search (ES), an information-efficient Bayesian optimization method, that leverages the acquired data when only a limited number of experiments re available. The authors demonstrate the effectiveness of the method on a robotic platform, that consist on a humanoid robot balancing an inverted pole. The pole states are estimated using a motion capture system, which provides low-latency data at $200Hz$. This approach can be classified as model-free. An initial linearized model of the plant is used, but this model is not updated during the controller learning process.

In [41] a model-based policy search framework (PILCO)[42] is used to automatically tune multivariate PID controllers. PILCO is a model-based RL framework for data-efficient learning of control policies int continuous state-control spaces. Based on collected data, a probabilistic model of the system dynamics is learned. The proposed method is flexible with respect to different PID structures.

# Chapter 4

# Automatic visual feature extraction

Introducing learning in perception can alleviate problems in the vision-based robot control system listed in Chapter 2. In visual servoing applications extracting control relevant information from the high-dimensional input data has a crucial role in the success of the control algorithm. Extracting relevant information of the high-dimensional input data in visual servoing application poses a serious challenge. A lot of work has been carried out for designing appropriate features for robotic manipulation tasks.

In general it is also important for learning robots in unstructured and dynamic environments to be able to extract control-relevant information from sensory readings. In order to learn by trial and error visual sensory readings have to be processed automatically. Various approaches have been taken to introduce learning in the perception part of the vision-based robot control pipeline: feature extraction, feature detection and tracking (Figure 4-9). Learning the visual features from real world data can give flexibility to the method on how visual input can be used, what tasks it can learn to perform [8].

In recent years results from computer vision research have transformed robot learning. These algorithms deal with the high-dimensional image data by learning a lower dimensional feature representation. The results of deep architectures for passive computer vision tasks such as object recognition and scene segmentation have been transferred to robotic applications. Deep neural networks made RL algorithms able to cope with the high-dimensionality of real world robotic applications.

This chapter first introduces representation learning techniques within computer vision with a focus on deep neural networks. First the basic concepts of neural networks and deep learning are presented. This is followed by the distinction of supervised and unsupervised learning, with the introduction to relevant models (CNN, autoencoders) to the visual servoing framework. Lastly the concept of a special instance of representation learning, state representation learning is discussed.

## 4-1 Representation learning in computer vision

In machine learning representations are used to transform the input data to a mathematical form, which can be then effectively used by a learning algorithm (e.g classifier). The performance of the machine learning algorithm relies heavily on the representation, finding a good representation can facilitate discovering structure in the input data by the learning algorithm [43].

The first representations were based on feature engineering, which required careful handcrafted design. In contrast representation learning aims at extracting useful information from the input for a given task automatically, with minimal human engineering. Features are learned directly from data. The two main approaches to representation learning are supervised and unsupervised learning. Supervised learning is a teacher-based learning method: manually defined labelled examples are provided to learn the mapping between the input and the output. In contrast in unsupervised learning no labels are provided, learning is self-taught, the algorithm has to discover statistical structure in observed data . The objective function is defined as the reconstruction loss. Unsupervised representation methods include: sparse coding, autoencoders, restricted Boltzmann machines [44]. Figure 4-2 compares representations in the traditional and the novel computer vision pipeline.

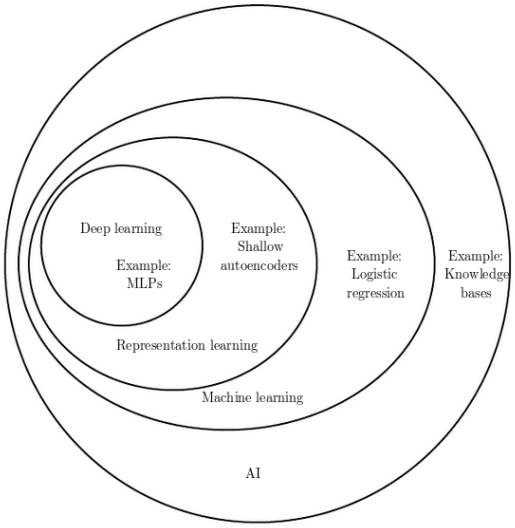



**Figure 4-1:** Deep learning as a form or representation learning (from [45])

In the last two decades multi-layer neural network based feature extraction has achieved unforeseen performance in various computer vision applications (e.g.,object detection, scene segmentation). The main key-points in the success of deep neural network based learning algorithms applied to computer vision tasks are the works of LeCun et al.,[46] and Hinton et al., [47]. In [46] it was shown that stochastic gradient descent can be used in supervised learning algorithms to discriminatively train large convolutional neural networks. In [47] it was demonstrated that unsupervised pre-training can accelerate the subsequent supervised learning. The increased computational power from GPUs compared to CPUs has also contributed greatly to the success of today's algorithms. The major breakthrough of using deep

neural networks was computer vision was in 2012, when a CNN outperformed traditional classification methods by a large margin on the ImageNet image classification benchmark [48]. Since then it has been shown that using visual features extracted from convolutional neural networks trained on large training data sets can lead to an increased performance in several computer vision tasks, such as segmentation and detection. Deep learning approaches to representation learning have been studied intensively in the last few years. An extensive overview of DL different DL approaches and applications is given in [45]. Figure 4-1 illustrates how deep learning is related to other machine learning methods.

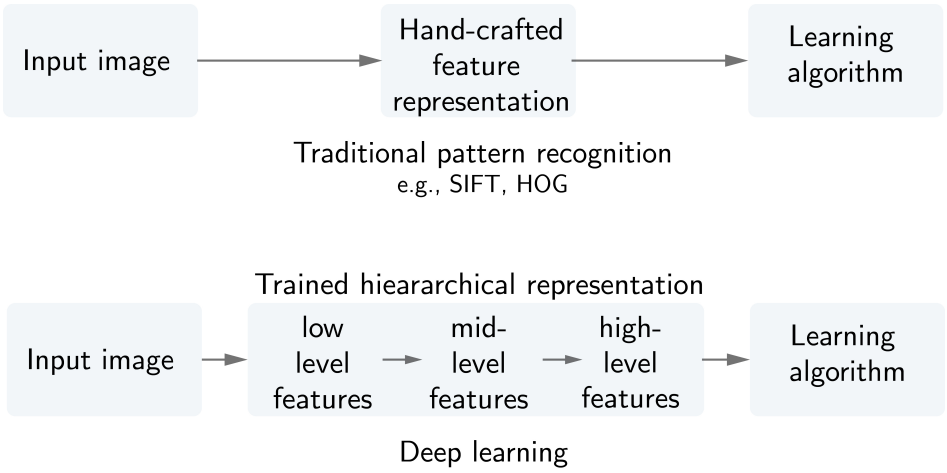



**Figure 4-2:** Representations in the traditional and in the novel computer vision algorithms

## 4-2 Neural networks

Neural Networks (NN) have been initially proposed as a way of modelling biological neural systems, but has found its applications within machine learning tasks. Generally an artificial neural network is a programming paradigm that is inspired by information processing in the biological nervous system, applied for learning from data [49].

The basic computational unit of neural networks are artificial neurons (depicted on Figure 4-3), which are based on the neurons of the brain. An artificial neuron takes several weighted inputs ($x_i$), and computes a function $g$ ($g$) of the weighted sum of the received inputs. The weights are denoted by $\omega_i$. This function is called the activation function or non-linearity. By using an appropriate activation function at the output of the neuron, a single neuron can be turned into a linear classifier. Commonly used activation function include the $Sigmoid$, the $Tanh$, $Softmax$ and the $ReLu$ functions. Until recently the $Sigmoid$ function was the most popular choice for an activation function. In recent years the $ReLu$ activation function have become a popular alternative, due to its improvement in convergence [48].

Neural networks are modeled as a collection of neurons, organized layer-wise. Different NN architectures exist based on the connections between the neurons of different layers. The only restriction on the NN architecture is that cycles are not allowed, because that would lead to an infinite loop during computation.

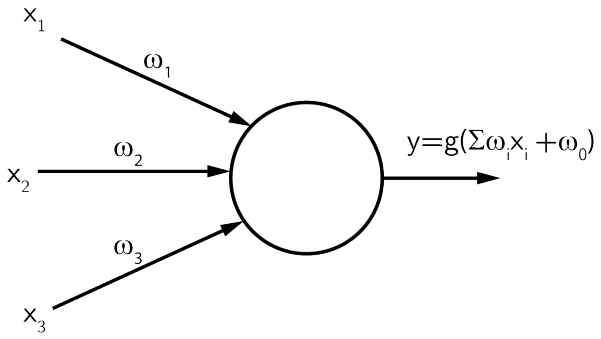



**Figure 4-3:** The artificial neuron model

The most common layer architecture type is the fully-connected neural network (depicted on Figure 4-4). In a fully connected neural networks the adjacent layer neurons are all pairwise connected, but neurons within the same layer have no connections. Other neural network architectures include networks with feedback connections (recurrent neural networks) and stochastic networks (restricted Boltzmann machines).The input to the neural network is transformed through a series of hidden layers during the forward pass. The last layer is called the output layer. The output of unit $i$ within layer $l$ can be computed as the following:

$$y_i^l = g(z_i^{(l)}) \qquad z_i(l) = \sum_{k=1}^{m^{(l-1)}} w_{i,k}^{(l)} y_k^{l-1} + w_{i,0}^l \tag{4-1}$$

where $g$ is the activation function, $w_{i,k}^l$ is the weighted connection between neurons, $w_{i,0}^l$ is the bias term, $l$ and $l-1$ are connecting layers.

The adjacent layers form a network, and can be represented by a composition of functions, which form a chain. The overall length of the this chain is called the depth of the network.

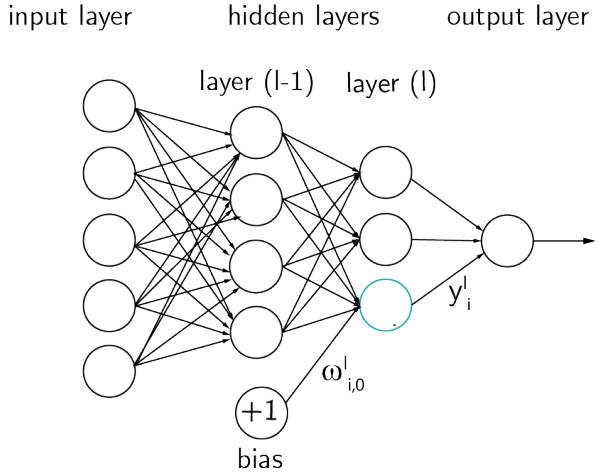



**Figure 4-4:** Fully-connected neural network

## 4-3 Motivation for Deep Neural Networks

Increasing the size of the neural network (number of layers, neurons) is motivated by the gain in capacity, the number of representable functions increases with the size of the network. The increased number of parameters also comes with increased expressiveness, by using a lot of parameters in the neural network any decision function in data space can be approximated.

A deep neural network (DNN) is a feed-forward artificial neural network that has more than one layer of hidden units between its inputs and ints outputs. Increasing depth in neural networks was inspired by how humans perceive the world, representing the world in terms of hierarchical structure: concepts at one level of abstraction as composition of concepts at lower levels. Deep architectures were motivated by the goal to automatically discover abstractions from lowest to highest level concepts.

Deep architectures are well-suited to represent higher-level abstractions, because of the reusability and composition of features.

In machine learning applications typically the number of possible configurations of variables is much larger than the number of training examples (curse of dimensionality). Compared to classical machine learning approaches the deep learning uses two prior assumptions about the network architecture to confront the this problem: distributed representation and deep architecture [45] .

In recent year deep learning has become a popular method for various machine learning applications replacing traditional methods due to the increased achieved performance (e.g., computer vision applications, natural language processing, time series forecasting).

## 4-4 Supervised Neural Networks

In case of supervised learning, a training set of input-output pairs is provided to the network $\left\{(x^{(1)}, y^{(1)}, (x^{(2)}, y^{(2)}), .., (x^{(m)}, y^{(m)})\right\}$ ,where $m$ is the number of examples, $x$ is the input and $y$ is the labelled output. During the training the network weights are adjusted using the backpropagation algorithm [45].

### 4-4-1 Supervised Convolutional Neural Networks

Convolutional neural networks (CNN) are a type of feedforward neural networks. In recent years deep convolutional neural networks (DCNN) have become the most popular approach to extract features from images, since they allow learning rich feature representation of input data with little preprocessing. CNNs have a wide application range (e.g., image and video recognition systems, natural language processing, recommender systems and deep reinforcement learning for robotics).

CNNs are inspired by the localized receptive field in the visual cortex. The basic concept behind CNNs is describing the image in a hierarchical structure, bottom-up connections can be used to infer high-level concepts (objects on image) from low-level features (edges). CNNs have two main properties: weight sharing and local connectivity.

**Local connectivity** CNNs exploit the topological structure of images inspired by the human vision system. Based on the 2D (in case of RGB-D 3D) structure of the input images, local receptive fields are defined, which are only sensitive to smaller regions in the visual field, reflecting the spatial dependencies between neighbouring pixels. This locally connected structure is also motivated by the curse of dimensionality, which comes with the high dimensional image data. By restricting unit connections between hidden layers, the number of trainable parameters in the network compared to the fully-connected network. In the network structure this means that the inputs in the units of a hidden layer (l) are restricted to a subset (receptive field) of the units of the previous layer (l-1) (Figure 4-5). This is the local connectivity property [50].

**Weight sharing** CNNs also exploit the stationary property of natural images, meaning that the statistics of one part of the image are the same as any other part. As a result the learned feature detectors (filters) can be applied anywhere on the input image. Due to weight sharing, relatively few parameters may be necessary to describe the behaviour of such a convolutional layer [50].

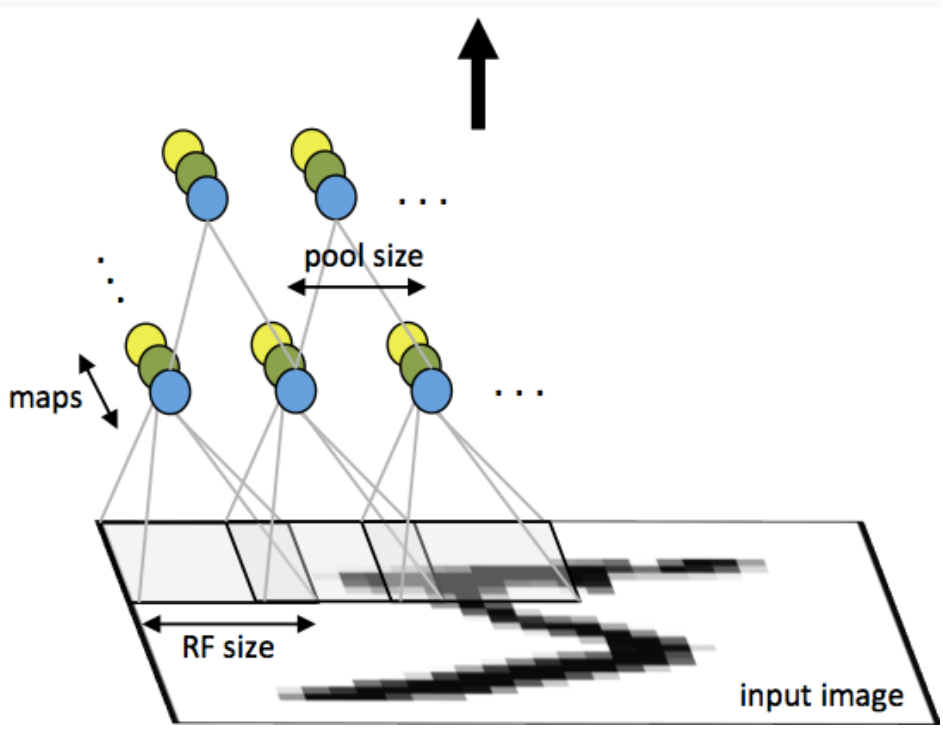



**Figure 4-5:** Convolutionial and pooling layers in the CNN architecture (from [50])

### CNN architecture

Convolutional neural networks consist of tree main types of layer: the convolutional layers, the pooling layers and optionally fully-connected layers. Complex architectures can be built by stacking multiple layers of these together [51].

In the simplest case a convolutional neural network consists of a convolutional layer, an activation function (ReLu), a pooling layer and a fully-connected layer.

**The convolutional layer** the input image is repeated convolved with a linear filter, a bias term is added and a non-linear function is applied. The output of the convolutional layer is the feature map. Each hidden layer is composed of multiple feature maps $h^k, k = 0, ..K$.

**The pooling layer** is a form of down-sampling. Each feature map is subsampled typically with mean or max operation into a set of non-overlapping rectangles. The pooling layers outputs the mean/maximum value for each sub-region. Sub-sampling decreases the computational need for upper layers and provides a form of translational invariance.

Since its introduction multiple convolutional neural network architectures have been introduced. The AlexNet is a CNN architecture consisting of 5 layers, the VGG is is an extended architecture, consisting of 16 convolutional layers.

By removing the classification and pooling layers deriving generic features across datasets becomes possible, which can be applied in robotics. It has been shown that CNNs trained on natural images have roughly the same Gabor filters and color blobs on the first layer for various datasets. It has been established that earlier layers in the hierarchy more general features are learned while later layers learn more specific ones. Removing the aggregations (pooling) and the classification layers results in convolutional filters which can be used to derive generic features across datasets [52].

#### Application of supervised deep neural networks in visual servoing

Most deep learning research has been carried out for recognition/classification tasks, where regions of interest (bounding box) are first extracted, then fed into a neural network for determining the object category. In robotic manipulation object detection and object pose estimation are needed, where a specific image has to be localized on the input image and then the orientation of the object is also often needed to be determined (e.g. for grasping). In object detection the biggest improvement came from the fusion of deep learning and classical detection methods (R-CNN algorithm)[53] ,

Perception learning based visual servoing requires the integration of subcomponents of passive visual tasks (e.g.,segmentation, recognition, pose estimation) into one pipeline. In this case the feature extraction, detection and tracking subproblems of the perception part are integrated in one learning algorithm.

This poses several challenges. One of these is the scarcity of labelled data and processing of RGB-D data. Pose estimation from RGB-D data using CNN structures have been proposed by several authors. Notable recent approaches of learned 6D pose include the self-learn RCNN network. [54], [53]

In [54] the self-learn RCNN algorithm is presented for $6D$ pose estimation. The algorithm achieves pose estimation with a rotational error of 8.5 deg and a translational error of $0.03m$. The scarcity of training data is resolved by an automated system for generating and labelling

datasets for training CNNs. This is achieved by proposing a phsysics simulator that uses the information from camera, calibration, shelf or table localization to setup an environment and performs physics simulation to place objects at realistic configurations. Then images of scenes are rendered to generate a synthetic dataset for training an object detector. This approach also opens up the possibility of a lifelong self-learning system that uses the object detector trained with the simulator to perform a robust multi-view pose estimation. The pipeline taken from the paper is depicted on Figure 4-6.

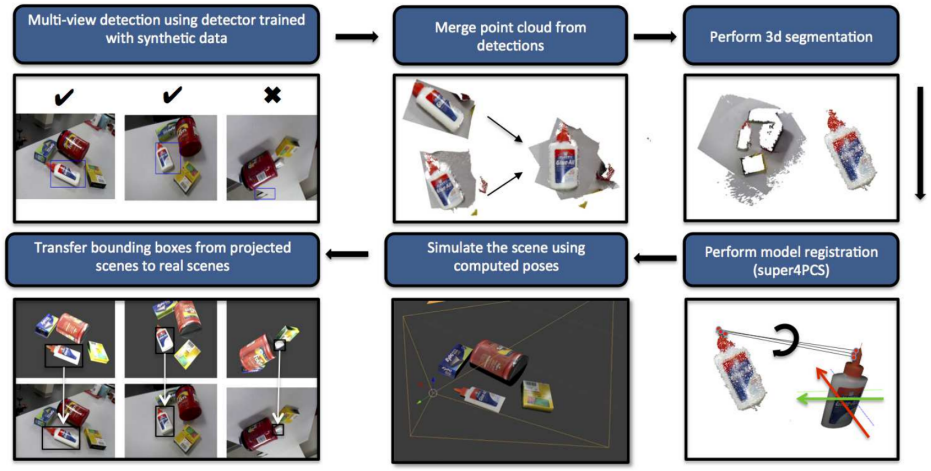



**Figure 4-6:** The self-learn RCNN algorithm (from [54])

In [53] the SegICP algorithm is presented for $6D$ pose estimation. The algorithm achieves estimation errors on the order of $5\,\text{deg}$ and $0.01m$ and the integrated semantic segmentation and object pose estimation operates at $4-14Hz$. Their approach leverages multi-view RGB-D data and self-supervised, data-driven learning. The proposed system can reliably estimate the $6D$ pose of objects under a variety of scenarios. The algorithm first segments the object from a scene by feeding multiple-view images to a deep neural network and then fits a 3D model to a a segmented point cloud and outputs the object's $6D$ pose. This is the registration step. Each color image is then fed into a convolutional network for $2D$ object segmentation. The point cloud then goes through background removal and then it is aligned with a pre-scanned $3D$ model to obtain its $6D$ pose. The algorithm pipeline taken from the paper is depicted on Figure 4-7.

## 4-5 Unsupervised Neural Networks

In unsupervised feature learning no target values are provided, the algorithm has to discover structure in the data itself. The motivation behind using unsupervised learning in robotics is that once a good high-level representation is learned, it could facilitate other learning tasks (e.g., supervised or reinforcement learning). It can be also used to initialize or regularize in the context of supervised learning systems. By using unsupervised learning as a pre-training step for processing the input data, the parameters of a supervised or reinforcement learning algorithm in a region from where local optimization (e.g. gradient descent) would yield good solutions [44].

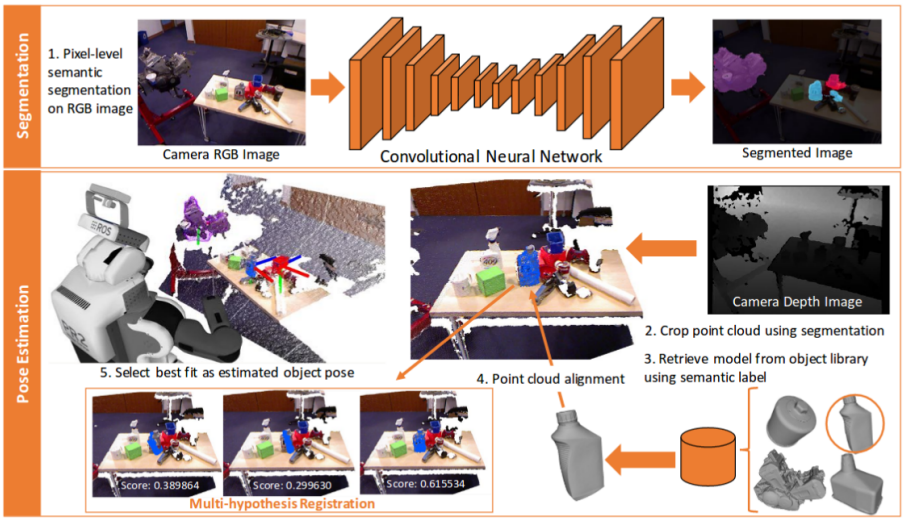



**Figure 4-7:** The SegICP algorithm (from [53])

The aim of unsupervised representation learning algorithms is to disentangle unknown factors of variations automatically. Factors of variations are different aspects of the data that can vary separately and often independently [43].

In unsupervised representation learning there is a distinguishment between sparse overcomplete and dense undercomplete representations. PCA is a dense undercomplete representation. The motivation for sparse representation is that it leads to an increases generalization power, the number of possible configurations increase and the effective dimensionality can vary in input region. A sparse representation means that only small subset of neurons are active at once, which leads to computational efficiency. It has been shown also that sparse autoencoders gave rise to more invariant representations, meaning that features are less sensitive to input transformations [43].

#### Principle Component Analysis

Principle component analysis (PCA) is an unsupervised learning technique for dimensionality reduction, and has been applied to a broad class of computer vision problems. PCA projects each input vector to low-dimensional coordinate vector defining low dimensional hyperplane in input space, where density is assumed concentrated.

### 4-5-1 Autoencoders

An autoencoder is a neural network that is trained to attempt to copy its input $\mathbf{x}$ to its output $\mathbf{r}$. It has a hidden layer $\mathbf{h}$ that describes a code used to represent the input. The network consists of two parts: and encoder function $\boldsymbol{h} = f(\boldsymbol{x})$ and a decoder function $\boldsymbol{r} = g(\boldsymbol{h})$. The training objective can be expressed as minimizing the loss penalizing the output being dissimilar from the input:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) \tag{4-2}$$

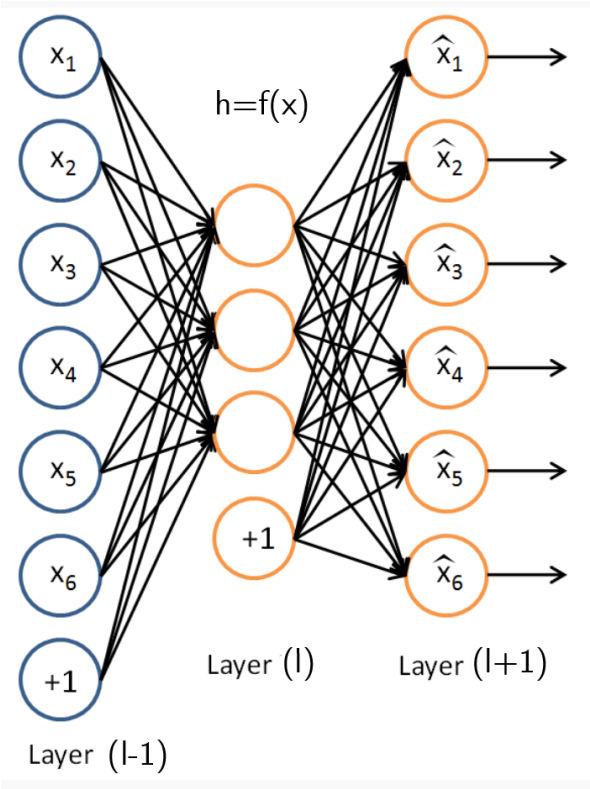



**Figure 4-8:** The autoencoder structure (from [50])

The interest is not in the output of the encoder, but on the code, the internal representation **h**. By placing constraints on the network, such as by limiting the number of hidden units, the aim is to discover interesting structure about the input data. An autoencoder is called undercomplete, if its code dimension is less than the input dimension.

Various methods have been proposed to prevent learning identity, these include explicit and implicit weight regularization, adding noise to the encoding (denoising autoencoder) and adding sparsity constraint (sparse autoencoder) on the learned code [44].

Desirable properties of the representation learned by the autencoder include sparsity, smallness of derivative of the representation and robustness to noise and to missing inputs. Different types of regularized autoencoders have been proposed to achieve this properties [45].

In deep autoencoders, multi-layer networks are used for the encoder or decoder functions. For example [8] uses a CNN architecture for the encoder function.

**The sparse autoencoder (SAE)** is an autoencoder whose training criterion involves a sparsity penalty $\Omega(\boldsymbol{h})$ on the code layer in addition to the reconstruction error:

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h}) \tag{4-3}$$

Sparse autoencoders are typically used to learn features for classification. [45].

**The denoising autoencoder (DAE)** has a modified reconstruction error term. The input is corrupted $\tilde{x}$. The denoising autoencoder must undo this corruption to reconstruct the original input. This denoising training forces the encoder and decoder functions to implicitly learn the structure of the input data [45]. Denoising autoencoders can be used to robustify the learning algorithm. In case of the denoising autoencoder the loss function takes the following form:

$$L(\boldsymbol{x}, g(f(\tilde{\boldsymbol{x}}))) \tag{4-4}$$

**The Variational Autoencoder (VAE)** combines deep learning with Bayesian inference. The motivation behind VAEs is to confront the scarcity of high dimensional real-world data. In contrast to other encoders, which are deterministic, the VAE is stochastic. It consists of a probabilistic encoder NN and a generative decoder NN.

## 4-6 State representation learning in robotics

Previous RL based visual servoing approaches relied on hand-crafted features for extracting control relevant information from the high dimensional input data. This provided them with only limited capabilities (3-6).

In recent years various methods have been proposed to extract lower dimensional, control-relevant information from images. State representation learning (SRL) is an instance of representation learning concerned with finding an appropriate state representation for control design in the RL problem. The motivation behind these algorithms is to extract a lower dimensional, compact state representation from the high dimensional sensory input without relying on specific human expert knowledge.

In these approaches the perception part of the visual servoing pipeline is replaced by a state representation learning algorithm , which extracts control relevant information from raw visual data. This is then fed directly to the control algorithm. Figure 5-1 depicts how state representation learning fits in the traditional visual servoing pipeline (Figure 2-6).

In contrast to CNN based supervised learning approaches, [53],[54] the aim of state representation learning is not the extraction of 6D pose information, which is more closely related to the tradition visual servoing pipeline.

In [55] explicit knowledge about physics is given to the learning algorithm through a set of robotic priors. This makes the learning problem tractable by reducing its dimensionality .

Many approaches use state dimensionality reduction, exploiting the assumption that in a lot of cases the high dimensional sensory data is intrinsically low dimensional. Autoencoders are often employed to obtain a lower dimensional representation from the raw sensory readings [56], [8], [57]. The lower dimensional state representation is obtained by training the autoencoder to reconstruct the sensor input. A control appropriate state representation can be obtained by specifying constraints on the reconstruction. In [8] a smoothness penalty is added to the reconstruction loss following [55] to constrain the features. In [57] variational and denoising autoencoders are used to reduce the dimension of input data and to automatically extract relevant state representation. The main motivation for using autoencoders to

compress the sensory readings is that high dimensional input domain (vision, tactile) contains correlated, non-relevant dimensions, where error metrics are hard to define.

Another approach is to use non-parametric kernel-based algorithms, which perform operations on kernel values. This makes them invariant to the high dimensionality, without the separate step of dimensionality reduction [58].

.

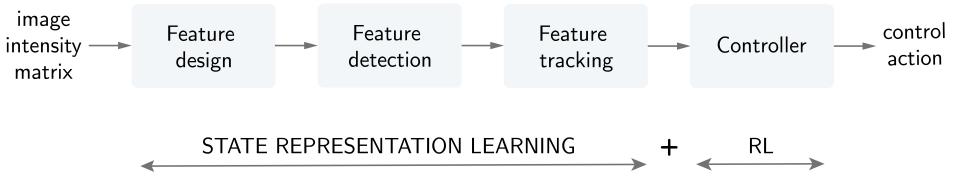



**Figure 4-9:** State representation learning

All approaches within state representation learning aim at constructing a lower dimensional representation from the high dimensional observations space with a non-linear mapping $\phi : Z \rightarrow X$. Requirements for a good state representations $X$ are being Markov, being able to represent the true value of the current policy well enough for policy improvement, generalize to unseen states with similar features and low dimensionality [30].

In the next chapter some examples of state representation learning will be presented under model-based deep reinforcement learning section.

# Chapter 5

# Deep reinforcement learning

For autonomous robots the aim is to find optimal closed-loop policies from raw visual information in an end-to-end fashion. So far, Deep Reinforcement Learning (DRL) has been the most promising approach to achieve this.

DRL combines deep neural networks with reinforcement learning. In image based DRL methods learning techniques are applied in both the perception and the control. Neural networks can be for representing the state-action value functions [5],[56], the policy [27] or for the forward model of the dynamics [59].

The first DRL algorithm was the DQN network [5], which combined a deep network with a value function based RL method. It was developed by DeepMind for the Atari 2600 games. They demonstrated that a single model can be used to automatically learn policies for different games, with different environment states and action sets. However translating the success of the DQN algorithm to real world problems is not straight-forward. The real-world applications DQN is constrained by the sample-inefficiency of model-free algorithms. Model-free DRL algorithms do not learn a predictive model of state transition dynamics, therefore they have the inherent problem of data-inefficiency of model-free RL algorithms [25].

Model-based DRL approaches have also been proposed [60], [61], [62]. In contrast to model-free approaches, these learn the underlying dynamics that govern the interaction between the robot and the environment. Model-based DRL approaches attempt to learn a dynamical model solely on pixel information by employing a deep neural network structure. Subsequently a control policy is optimized for this learned dynamics network. This is also the source for the main drawback of these methods. The quality of the policy depends on the quality of the learned model [25].

This chapter presents recent image based closed-loop DRL algorithms, where a control policy is obtained from raw visual data. In the first section model-free DRL approaches are collected and the DQN algorithm is presented, along with the DDPG algorithm. In the second section some examples of recent model-based DRL approaches are presented.

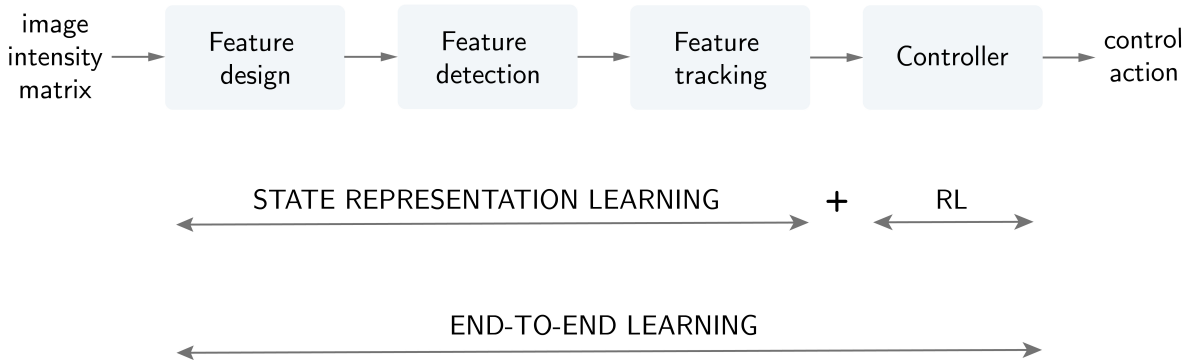



**Figure 5-1:** Comparison of end-to-end vs state representation learning algorithms based on the traditional visual servoing pipeline

## 5-1 Model-free deep reinforcement learning approaches

Deep reinforcement learning on image data has first been applied in a model-free context. Most of these approaches are value-function based, they first estimate the value function and the policy is derived from the estimated value function [56], [5].

The DQN network was he first RL algorithm to tackle the high-dimensional state space problem. However it did not solve the problem of high-dimensional action spaces in RL problems, the action set in these games was quite limited. In [56] the action set is also discrete. A algorithm learns to control slot cart with a discrete action set based on raw images.

Various extensions of the original DQN algorithm have been proposed. A large part of these extensions concentrate on translating the success of the DQN algorithm to high-dimensional action spaces (DDPG algorithm [27], DDAC algorithm, Double DQN algorithm).

### The DQN algorithm

In the DQN network [5] value-function based RL is combined with a deep neural network. The DQN algorithm trains a CNN to approximate the highly non-linear Q function with online gradient descent approach based on a large number of interactions with system. A GPU based CNN is employed to train a RL algorithm to play ATARI games directly from $84x84$ pixel $60Hz$ video input (5-2).

The input to the network is stack of raw pixels from last 4 frames, the output is $Q(\boldsymbol{x}, \boldsymbol{u})$ for 18 joystick/button positions. The reward is defined as the change in the score for the given step. The environment states in the games are represented by four subsequent game images. The subsequent images are needed for velocity information. By using the image data, rather than a smaller set of game-specific state environments as states, a general framework for learning can be obtained. This allows generalization across different tasks, allows the same network to be used for different games. On the other hand this makes the dimensionality of the problem very large. A CNN is used to confront the high dimensionality of the state space in the value function approximation method.

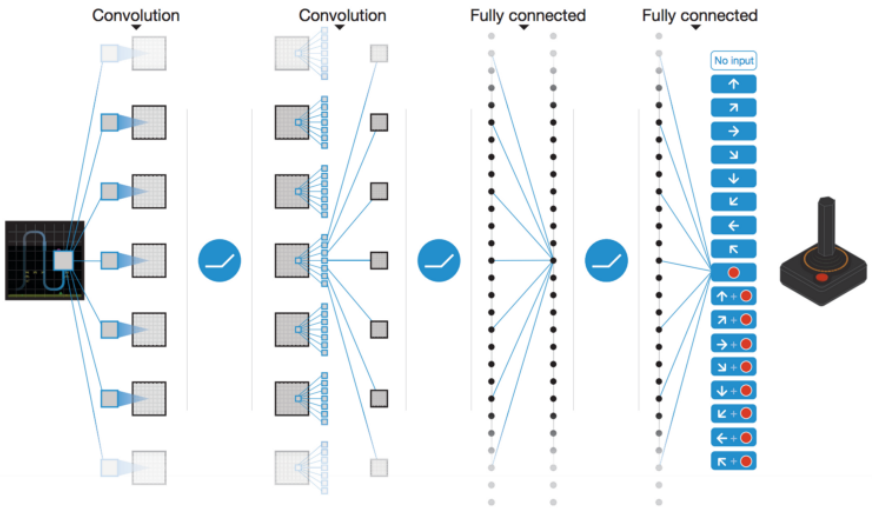



**Figure 5-2:** The CNN structure in the DQN algorithm (from [5])

The CNN architecture contains three convolutional layers, followed by two fully connected layers. The pooling layers are removed,since they are responsible for translation invariance, which would make the network becomes insensitive to the location of an object in the image.

Previous attempts using neural networks had been unsuccessful due to convergence problems. The DQN algorithm provides a stable solution to deep neural network based value function based RL. The key ideas to the success of the DQN algorithm are the use of experience replay and using a a target network. Based on experience replay approach in the RL framework, state transitions are stored in the replay memory and from this memory random mini-batch transitions are sampled. This ensures learning from past policies, compared to online learning. A separate target network is used for providing target values to the main estimator network during learning. The reason for this is to prevent stability problems caused by coupling between the target and estimated Q values. The Q learning targets are calculated with respect to the old, fixed parameters, independently of the estimated Q values.

### The DDPG algorithm

The DDPG algorithm is a model-free, actor-critic algorithm, which is an extension to the value-based DQN algorithm. The motivation behind it was to create an algorithm that can be used in high dimensional action spaces.

The DDPG algorithm is an actor-critic algorithm, a separate parametrized action policy (actor) is learned in addition to the Q function (critic). Two separate neural networks are used for the policy and the value-function. The input to the policy network is the current state, the output is the action, represented by single real value. The value-function network calculates the value function based on the current state and the action provided by the policy network. A TD update rule is used to update the value-function network.

## 5-2 Model-based deep reinforcement learning approaches

When RL is applied to real world robotic problems, on of the main challenges that has to solved is to make the algorithm sample effectively, since real world experiments are costly. The main motivation behind learning a forward model of the dynamical system is that it provides a way to increase data efficiency, making the algorithm be able to learn from a small number of trials [25]. The learned model can be used for internal simulations and policy learning, decreasing the need for rollouts, which are costly in real robotics problems. Learning forward models have been applied previously in robotics problems [22]. In the past few years many model-based reinforcement learning algorithms using deep learning have been proposed [61], [59], [63], [62].

Most of these approaches use deep architectures to learn a compact lower dimensional state representation from the high dimensional visual input information as in Section 4-6, and learn a predictive model of the forward dynamics. In contrast to the model-free DRL algorithms, these algorithms have a separate component for state representation learning and policy learning.

### Deep spatial autoencoders for visuomotor learning

In [8] a method is presented to directly learn state representations from raw sensory data. The motivation behind the proposed method is learning a task-appropriate state representations for control without human supervision. These task-appropriate state representations are the poses of the relevant objects in the world using Cartesian coordinates.

The proposed method uses a deep autoencoder to acquire a set of feature points describing the locations of the objects relevant to the current task, which are then used to learn the motor skills for the given task. The encoder is constrained to learn a lower dimensional state representation and by performing a specific operation (spatial soft arg max) on the output of the last convolutional layer, the pixel locations with the maximum activations are returned. Figure 5-3 represents the spatial autoencoder structure, taken from the paper.

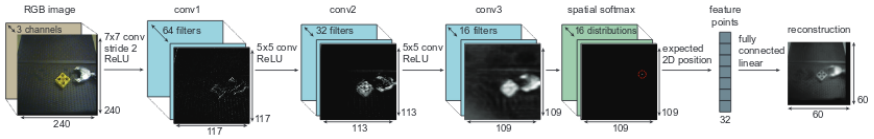



**Figure 5-3:** Deep spatial autoencoder structure used in [8]

Additionally, a smoothness penalty is added to the reconstruction loss of the deep spatial autoencoder, in order to make the learning algorithm able to predict the dynamics of the state representation.

For the control part, the algorithm builds on previous results [60]. The Guided Policy Search approach is used. First local linear-Gaussian controllers are trained for multiple initial positions, generating multiple solutions. Then the final policy for a specific task is obtained by using supervised algorithm that uses data from the linear controllers. This approach makes the algorithm be able to generalize to new end goal positions.

The set-up is similar to traditional visual servoing methods, where features are extracted and fed into a controller, with the main difference that here the features are learned from real-world data and in contrast to most visual servoing approaches, camera calibration is not required.

### Learning deep dynamical models from image pixels

In [61] a model-based DRL algorithm is proposed, which allows learning closed-loop policies in continuous state and action spaces directly from pixel information. The two main components of the method are a deep dynamical model (DDM) and an Model Predictive (MPC) controller. The DDM is used to make long-term predictions using a learned forward model, which are fed into the MPC controller, which determines the optimal action.

In the DDM approach the feature mapping from high dimensional image input to a lower dimensional representation is jointly learned with a forward model in this lower dimensional feature space. A deep auto-encoder is used to learn the lower dimensional, compact representation. For learning the predictive model in the lower dimensional features space, a non-linear autoregressive feed-forward neural network is used. It is assumed that future features depend on an n-step history of past features and control inputs. Therefore the DDM has three sets of parameter vectors the encoder, the decoder and the predictive model parameters, which are found by jointly optimizing the squared reconstruction error of the deep auto-encoder and the prediction errors of the forward model.

### Deep Visual Foresight for Planning Robot Motion

In [59] the motivation behind their approach is to replace the standard robotic manipulation pipeline with learning algorithm, based on the success of learning algorithms for passive perception tasks. The setup is similar to [61], the main components of the learning algorithm are a predictive model of raw sensory observations and MPC control. The learning algorithm is completely self-supervised: no reward function, no image of goal, nor the ground-truth object pose information is defined beforehand. The main contribution of the paper is a deep video-predictive model, that can be used in real robotic system to manipulate previously unseen objects.

# Bibliography


[1] J. Biswas and M. Veloso , "Depth camera based indoor mobile robot localization and navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 1697–1702, IEEE, May 2012.

[2] J. Kosecka, R. Blasi, C. J. Taylor, and J. Malik, "A comparative study of vision-based lateral control strategies for autonomous highway driving," *International Journal of Robotics Research*, vol. 18, pp. 442–453, 1999.

[3] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Lessons from the amazon picking challenge," *arXiv preprint arXiv:1601.05484*, 2016.

[4] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1765–1772, IEEE, 2013.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 02 2015.

[6] K. Narasimhan, T. D. Kulkarni, and R. Barzilay, "Language understanding for textbased games using deep reinforcement learning," in *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Citeseer, 2015.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[8] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 512–519, IEEE, 2016.

[9] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," *arXiv preprint arXiv:1610.00633*, 2016.

[10] P. I. Corke, "Visual control of robot manipulators – a Review," in *Visual Servoing*, pp. 1–31, World Scientific Pub Co Pte Lt, oct 1993.

[11] D. Kragic, H. I. Christensen, *et al.*, "Survey on visual servoing for manipulation," *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, vol. 15, 2002.

[12] K. Hashimoto, "A review on vision-based control of robot manipulators," *Advanced Robotics*, vol. 17, no. 10, pp. 969–991, 2003.

[13] L. Weiss, A. Sanderson, and C. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal on Robotics and Automation*, vol. 3, no. 5, pp. 404–417, 1987.

[14] S. Hutchinson and F. Chaumette, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[15] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *ieee Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 2002.

[16] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The confluence of vision and control*, pp. 66–78, Springer, 1998.

[17] M. Stuflesser and M. Brandner, "Vision-based control of an inverted pendulum using cascaded particle filters," in *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE*, pp. 2097–2102, IEEE, 2008.

[18] C. G. Cifuentes, J. Issac, M. Wuthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robotics and Automation Letters*, 2016.

[19] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 82–90, December 2006.

[20] M. E. Magana and F. Holzapfel, "Fuzzy-logic control of an inverted pendulum with vision feedback," *IEEE transactions on education*, vol. 41, no. 2, pp. 165–170, 1998.

[21] S. Schaal and C. G. Atkeson, "Learning control in robotics," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 20–29, 2010.

[22] C. G. Atkeson and S. Schaal, "Robot learning from demonstration," in *ICML*, vol. 97, pp. 12–20, 1997.

[23] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *NIPS*, pp. 739–746, 2005.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[25] J. Kober, J. A. D. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, July 2013.

[26] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Found. Trends Robot*, vol. 2, pp. 1–142, Aug. 2013.

[27] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[28] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*, pp. 317–328, Springer, 2005.

[29] C. Gaskett, L. Fletcher, A. Zelinsky, *et al.*, "Reinforcement learning for visual servoing of a mobile robot," in *Proc. Australian Conf. on Robotics and Automation (ACRA2000)*, 2000.

[30] W. Böhmer, J. T. Springenberg, J. Boedecker, M. Riedmiller, and K. Obermayer, "Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations," *KI-Künstliche Intelligenz*, vol. 29, no. 4, pp. 353–362, 2015.

[31] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control," *Machine learning*, vol. 84, no. 1-2, pp. 137–169, 2011.

[32] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2094–2100, AAAI Press, 2016.

[33] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning (ICML)*, 2014.

[34] J. Peters, "Policy gradient methods," *Scholarpedia*, vol. 5, p. 3698, Nov. 2010.

[35] S. Levine and V. Koltun, "Guided policy search.," in *ICML (3)*, pp. 1–9, 2013.

[36] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," in *Recent Advances in Robot Learning*, pp. 163–187, Springer, 1996.

[37] T. Martínez-Marín and T. Duckett, "Fast reinforcement learning for vision-guided mobile robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pp. 4170–4175, IEEE, 2005.

[38] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning (ICML)*, pp. 593–600, ACM, 2005.

[39] A. massoud Farahmand, A. Shademan, M. Jagersand, and C. Szepesvári, "Model-based and model-free reinforcement learning for visual servoing," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 2917–2924, IEEE, 2009.

[40] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic lqr tuning based on gaussian process global optimization," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 270–277, IEEE, 2016.

[41] A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe, "Model-based policy search for automatic tuning of multivariate pid controllers," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.

[42] M. P. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *International Conference on Machine Learning (ICML)*, pp. 465–472, 2011.

[43] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning.," *ICML Unsupervised and Transfer Learning*, vol. 27, pp. 17–36, 2012.

[44] Y. Bengio *et al.*, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[46] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[47] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[49] M. A. Nielsen, "Neural networks and deep learning," *Determination Press*, 2015.

[50] A. Ng, "UFLDL Tutorial." http://ufldl.stanford.edu/tutorial/, 2008. [Online; accessed 20-February-2017].

[51] K. Jarrett, K. Kavukcuoglu, Y. LeCun, *et al.*, "What is the best multi-stage architecture for object recognition?," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153, IEEE, 2009.

[52] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[53] J. M. Wong, V. Kee, T. Le, S. Wagner, G.-L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. Johnson, *et al.*, "Segicp: Integrated deep semantic segmentation and pose estimation," *arXiv preprint arXiv:1703.01661*, 2017.

[54] C. Mitash, K. E. Bekris, and A. Boularias, "A self-supervised learning system for object detection using physics simulation and multi-view pose estimation," *arXiv preprint arXiv:1703.03347*, 2017.

[55] R. Jonschkowski and O. Brock, "Learning state representations with robotic priors," *Autonomous Robots*, vol. 39, no. 3, pp. 407–428, 2015.

[56] S. Lange, M. Riedmiller, and A. Voigtlander, "Autonomous reinforcement learning on raw visual input data in a real world application," in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–8, IEEE, 2012.

[57] H. van Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, "Stable reinforcement learning with autoencoders for tactile and visual data," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 3928–3934, IEEE, 2016.

[58] H. van Hoof, J. Peters, and G. Neumann, "Learning of non-parametric control policies with high-dimensional state features," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, vol. 38, p. 995âĂŞ1003, JMLR, 2015.

[59] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," *arXiv preprint arXiv:1610.00696*, 2016.

[60] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[61] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "Learning deep dynamical models from image pixels," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1059–1064, 2015.

[62] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: a locally linear latent dynamics model for control from raw images," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pp. 2746–2754, MIT Press, 2015.

[63] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman, "Deep predictive policy training using reinforcement learning," *arXiv preprint arXiv:1703.00727*, 2017.