A Learning Camera Platform for Remote Operations with Industrial Manipulators [⋆]

Sigurd A. Fjerdingen [*] **Magnus Bjerkeng** [**]
Aksel A. Transeth [*] **Erik Kyrkjebø** [*] **Anders Røyrøy** [***]

[*] *SINTEF ICT Dept. of Applied Cybernetics, NO-7465 Trondheim, Norway (e-mail: sigurd.fjerdingen@sintef.no).*
[**] *Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*
[***] *TPD RD New Development Solutions, Statoil ASA, P.O. 7200, NO-5020 Bergen, Norway*

Abstract: Robot manipulators may be used as flexible camera platforms by mounting cameras on the wrist of the robot. In this paper we present a new way of interaction between an operator and the camera platform, where the operator wants to get a visual overview of a remote operation. The goal is to relieve the operator from controlling both the operation and the camera platform simultaneously, and allow the operator to focus only on the operation while the camera plaform is automatically controlled based on learned operator preferences. We describe an architecture for learning from operator inputs, and use an active camera control algorithm as a base for learning. An M-RAN sequential function approximator is used as memory function. Experimental results on a demonstration case indicate that the camera platform responds to and remembers differences in operator preference.

Keywords: Robotic manipulators, industrial robots, learning systems, radial basis function network, remote control, offshore operations.

1. INTRODUCTION

Offshore oil and gas platforms pose challenging environments for personnel owing to their remote, isolated places, harsh maritime environment, and often explosive, toxic and/or corrosive atmosphere. Remote control of offshore operations from onshore control centres may reduce both costs and health and safety concerns. Fig. 1 shows such a platform concept for remote operations. There are, however, several challenges involving remote operations, some of which may be alleviated by using robot learning methods. This is a topic of the current paper. A concept laboratory for remote operations with robots has been built and presented by Kyrkjebø et al. (2009). In the laboratory, robot manipulators can be remotely controlled by a human operator. Sufficient overview of the process and environment is needed to perform remote controlled offshore operations, and key to this is live video monitoring. The authors have previously presented a robot manipulator with a wrist mounted camera, constituting a *movable camera platform* (see Fig. 2 and Bjerkeng et al. (2011a,b)). The camera platform includes obstacle avoidance and can be joystick-controlled by an operator.

An important issue is the fact that during remote operations the operator is tasked with controlling the camera platform *simultaneously* with controlling the operation manipulator itself, which creates potentially stressful and information loaded situations. Ideally, the camera platform should position itself at good viewing angles with as little interference from the operator as possible.

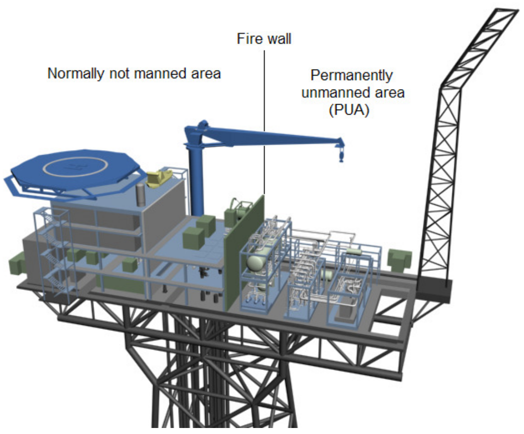
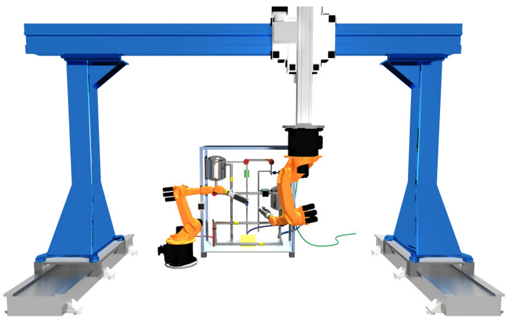


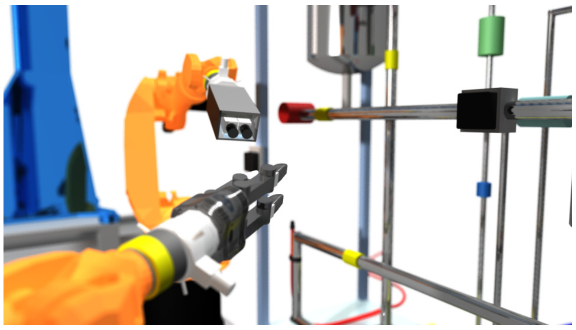
Fig. 1. The Mesa Verde platform concept. The robot system is to be mounted in the PUA region.

In this paper we propose a new method of interaction between the remote operator and the camera platform with the goal of reducing the need to control the platform. The camera platform tries to learn where the operator wants it for any given operation by learning from interactions the operator has with the platform. We describe a framework for learning from operator inputs using a hierarchical

[⋆] Contribution to invited session Robot Automation in Explosive Oil & Gas Environments. This work is part of the research project *Next Generation Robotics for Norwegian Industry*. The project partners are The Research Council of Norway, SINTEF, NTNU, Statoil, Hydro, Tronrud Engineering, SbSeating, Glen Dimplex Nordic, and RobotNorge.



(a) Overview of the two manipulators and process structure.



(b) Camera platform seen from the operation manipulator viewpoint.

Fig. 2. 3D models of the laboratory facility.

composition of the state space for learning and a modified minimum resource-allocating-network (M-RAN) function approximator to incrementally approximate operator preferences (Yingwei et al., 1997).

A plethora of different learning strategies exist for robot-centric learning. Types of robot learning where a human is included as an expert in some form includes interactive learning (Thomaz et al., 2005), imitation learning or learning from demonstration (Argall et al., 2009), and apprenticeship learning (Abbeel and Ng, 2005). All these methodologies share some common ground, and as such we choose to follow the terminology used by e.g. Argall et al. (2009) and use the phrasing *learning from demonstration* for denoting learning not only based on interactions with the environment, but also on feedback from an external human operator as this general term seems to encompass more specialized versions. We extend and adapt central ideas of these methodologies to the case of a camera platform able to learn preferences of an operator.

The paper is organized as follows. Section 2 presents the remote operations concept and laboratory facility, as well as describes the active camera control algorithm. We develop an approach to learning for the active camera control system, and describe mapping functions and correspondence issues in Section 3. In Section 4 the actual implementation of the learning algorithms are described, and in Section 5 experimental setup and results displaying the concept during a select operation is presented. Conclusions and suggestions to further work are given in Section 6.

2. BACKGROUND

In this section we first give a short overview of the concept for normally-unmanned oil and gas platforms. Then, a laboratory facility for testing and development of methods for remote inspection and maintenance (I&M) on such platforms is presented. Finally, we give a summary of a recently published control approach for robot-based real-time camera monitoring which is used as a basis for the learning algorithms developed in this paper.

2.1 Remote Operations Concept

A novel concept for remote controlled I&M on offshore oil and gas platforms was presented by Kyrkjebø et al. (2009). The platform concept separates the work area accessible by human operators, and a closed permanently unmanned area (PUA) that is only serviced by robots (see Fig. 1). The remotely operated platform concept is designed on the premise that robots may replace humans for the most important scheduled I&M operations inside the PUA such as gauge readings, valve and lever operations and monitoring leakages (Graf and Pfeiffer, 2008).

Remote offshore I&M operations pose many significant challenges (Anisi et al., 2010; Kyrkjebø et al., 2009). Onshore operators must for instance be able to monitor and control I&M operations with a large range of level of detail. This requires new and versatile monitoring approaches with active camera control, which is a topic of this paper.

2.2 Lab Facility

A lab facility has been built in Trondheim, Norway, in order to develop, test, and demonstrate solutions for next-generation I&M operations for normally-unmanned oil platforms. A brief overview of the facility is given in the following.

The lab facility is composed as follows. A process structure simulates parts of a production process on a real oil-platform and two robot manipulators are used for I&M tasks on the process structure (see Fig. 2).

Both robots are standard 6-axes manipulators (Kuka KR-16). One is mounted on a 3-axes gantry. The gantry-mounted robot (GR) performs the main I&M operations on the process equipment, while the floor-mounted robot (FR) is used for monitoring and assisting the gantry-mounted robot with e.g. a stereo vision camera.

The lab facility can be remotely controlled from any location via the Internet. Live video streams and continuously updated 3D models of the facility provide a remote operator with awareness of the robot operations. The remote operator can initiate high-level commands for automatic I&M routines from a graphical representation of the process equipments, or control the robots with off-the-shelf joysticks either directly or via 3D models. See Kyrkjebø et al. (2009) for further details.

2.3 Active Camera Control

Bjerkeng et al. (2011b) developed and tested a new position based approach for real-time monitoring of an operation using an automated camera platform. The solution is based on the weighted pseudoinverse redundancy resolution method and provides globally stable camera tracking.

Secure operations are achieved using three independent levels of collision avoidance, and robustness to kinematic and representational singularities is shown to be an inherent feature of the specific task formulation. Joint velocity weighing is used to gracefully achieve joint limitations present in industrial manipulators. This approach is simpler than the standard visual servoing technique, and does not require image processing or modifications to processing equipment such as the addition of visual markers. The position based method does however not eliminate the possibility of occlusion.

The operator is allowed to vary a desired zoom distance $y = ||\mathbf{p} - \mathbf{x}_r(\mathbf{q_r})||$ on-line since stereoscopic cameras typically do not support zooming. The position of the camera lens is given by $\mathbf{x}_r(\mathbf{q_r})$ and $\mathbf{p}(t)$ is the point the camera should follow.

This distance control input is imposed as a proportional gain controller along the desired view vector $\mathbf{p} - \mathbf{x}_r$ as

$$\mathbf{F}_d = -k\mathbf{J}_v^T \frac{\mathbf{p} - \mathbf{x}_r}{||\mathbf{p} - \mathbf{x}_r||}(y_{\text{desired}} - ||\mathbf{p} - \mathbf{x}_r||), \qquad (1)$$

and is imposed in the task nullspace. In (1), $\mathbf{J}_v$ is the linear velocity Jacobian of the end effector, and $k > 0$ is the control gain. The camera task is achieved by using the industrial manipulator's on-board joint controller to comply with $\mathbf{q}_r$ given by integrating

$$\dot{\mathbf{q}}_r = -k_p \mathbf{J}_w^+ \mathbf{\Sigma}(\mathbf{q}_r) + (\mathbf{I} - \mathbf{J}_w^+ \mathbf{J}_t)\mathbf{W}^{-1}\mathbf{F}_d, \qquad (2)$$

where $k_p{>}0$ is a proportional gain and $\mathbf{\Sigma}(\mathbf{q}_r) : \mathbb{R}^n \mapsto \mathbb{R}^2$ is a minimal camera task description. The matrix $\mathbf{J}_t = \frac{\partial \mathbf{\Sigma}(\mathbf{q}_r)}{\partial \mathbf{q}_r}$ is the task Jacobian, and $\mathbf{J}_w^+$ is its weighted Moore-Penrose pseudoinverse. The projection matrix $(\mathbf{I} - \mathbf{J}_w^+ \mathbf{J}_t)\mathbf{W}^{-1}$ maps the joint velocity $\mathbf{F}_d$ to the task nullspace to eliminate task interference.

3. CONTROL STRATEGY

This section describes how an external operator is able to adjust the zoom parameter of the active camera control system and how the camera platform over time adapts to and remembers the correct parameter in the given situation.

We base our control strategy on the active camera control algorithm of Bjerkeng et al. (2011b). This algorithm makes the camera platform tool point to the wrist of the operation manipulator at a set distance which we denote the *zoom distance*, as shown in Fig. 3. In the current implementation of the active camera control algorithm, the zoom distance is the only adjustable parameter. It is straightforward to extend both the active camera control algorithm as well as the learning algorithms presented in this paper to include other parameters such as adjusting spatial focus points. This extension will also alleviate the issue of occlusions as mentioned in section 2.3, as the operator will be able to move the camera platform away from occluded positions. This is also mentioned as future work in section 6.1. In this paper we base the presentation on the published version of the algorithm which is limited to adjustable zoom distance.

The control objective for the learning system is to generate the correct zoom distance in real time according to data previously received from an operator which should relate to the viewing preferences of that particular operator.

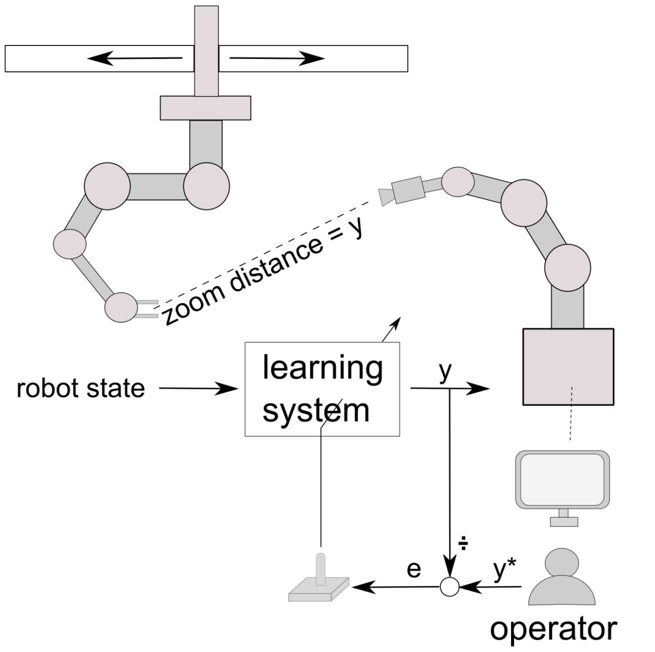


Fig. 3. Illustration of the learning system feedback loop and the laboratory setup consisting of operation manipulator and camera platform.

The section is divided into three parts, where the first part gives an overview of the learning process while the latter two parts cover two central aspects of the learning algorithm.

3.1 Overview of the Learning Process

An illustration of the learning process is shown in Fig. 3. The operator observes the current state of the operation manipulator through a monitor streaming video from the camera platform. When the zoom distance y of the camera platform deviates from the preferred zoom distance y^* of the operator, the operator may use a joystick to inform the learning system both that the current zoom distance is incorrect as well as in which direction to experiment with a new zoom distance. When the learning system receives $e = y^* - y \neq 0$ from the operator, it is allowed to change its guess of the optimal zoom distance based on this e.

The learning process itself is quite simple in our context. While being inspired by reinforcement learning (Sutton and Barto, 1998) and interactive reinforcement learning (Thomaz et al., 2005) in particular, the underlying Markov decision process (MDP) of this particular problem can be said to be stateless in the sense that for each state of the operation manipulator we are interested in the optimal zoom distance, i.e. there is no need to traverse a sequence of states in order to reach the goal state. The truth of the preceeding sentence is naturally dependent on the choice of state space, which is one of two key issues of our approach. The second key issue is the creation and maintaining of a memory system capable of storing and updating experiences in such a way as to gain a *memory map* of the operator's preferred views in any situation.

3.2 Correspondence: On the Selection of State Space for Learning

The form of learning discussed in this paper can be viewed as a method of mapping situations to actions, i.e. to answer the question of which action is best in a given situation (Argall et al., 2009) – in our case the action maps to the correct zoom distance. In order to recognize and isolate one situation from another, the key components of that situation need to be identified and quantified. We refer to this quantified situation description as the *state space* of our learning problem.

In a general setting the issue of correspondence deals with identifying a mapping between the teacher and learner which allows transfer of information from one entity to another (Argall et al., 2009). For a camera platform that should learn a behavior based on the percieved intentions of an external operator, this question becomes complex. We can at most anticipate what the operator's preferences for camera viewing are based on. We propose the following three structural components of the state space:

(1) The spatial position of the currently used tool,
(2) Which tool is currently in use, and
(3) Which operator is currently controlling the operation manipulator.

These situation identifiers are based purely on heuristic experience gathered from users of the remote inspection and maintenance concept laboratory. For the operations carried out in our laboratory, each operation needs a particular tool, so we are able to identify which operation the operator is executing by examining which tool has been selected (structural component 2). This means that we in our case can identify each operation uniquely without needing the operator to input the operation at hand. In a general setting, structural component 2 could read which *operation* is currently executing. Ideally, automated techniques based on observing operators' usage patterns over time could be employed for pruning and selecting more optimal state spaces, but this has not been adressed in this paper.

Further, we propose a hierarchical composition of this state space. Fig. 4 illustrates this composition. Structural components 2 and 3 (tool type and operator ID) are discrete and finite state dimensions of limited size, whereas the spatial position of the currently used tool (i.e. the operation manipulator wrist position) is a vector $\mathbf{p} \in \mathbb{R}^3$ of three continuous variables contained within a closed set $\mathbf{p} \in \mathcal{P}$ governed by the manipulation envelope. In our composition, component 2 and 3 can be seen as indices into a table where each table entry in itself contains a separate memory function for that specific case. This memory function is needed in order to represent a continuous set through a finite set of variables.

Earlier works have described efficient memory functions for continuous state spaces, and the following section describes the internals of the memory function used in the current research.

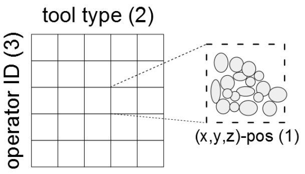


Fig. 4. Graphical illustration of the composition of the state space for learning. *Tool type* and *operator ID* are used as discrete indices into a table consisting of memory functions based on the M-RAN function approximator.

3.3 The M-RAN Function Approximator

Handling continuous state spaces through function approximation has been extensively studied both outside (Bishop, 2007) and in the robot learning literature. One particular advantage of function approximators is the inherent ability to generalize experience from a finite set of samples to a possibly infinite space. Commonly used function approximators in robotics include connectionist structures such as artificial neural networks (ANN), radial basis function networks (RBFN) (Fjerdingen et al., 2010), and cerebellar model arithmetic computers or tile coding (CMAC) (Sutton and Barto, 1998).

The memory function of the current control system implements an adaptation of a sequential learning scheme using minimal radial basis function networks, denoted *Minimal Resource Allocation Network* or M-RAN (Yingwei et al., 1997). Radial basis function networks are well suited for function approximation due to a simple topological structure and local approximation validity. A sequential version of the RBFN algorithm, adding units to the network based on the novelty of new data, is called a resource-allocating network (RAN). The M-RAN algorithm additionaly allows for pruning units shown to be inactive over time, thereby going by the name *minimal.* These two properties are important for our case as the hierarchical composition of memory may lead to a large amount of memory maps (i.e. function approximators), and as such each memory map should be kept as lean and fast computing as possible. The following discussion summarizes the work of Platt (1991); Kadirkamanathan and Niranjan (1993); Yingwei et al. (1997).

The output of a RAN algorithm can be given as

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^{K} \alpha_k \phi_k(\mathbf{x}), \tag{3}$$

where $\phi_k(\mathbf{x})$ is the response of the kth unit to input vector $\mathbf{x}$. α_k is the corresponding weight parameter and α_0 a bias term. For radial basis functions $\phi_k(\mathbf{x})$ is given by a Gaussian function:

$$\phi_k(\mathbf{x}) = \exp\left(-\frac{1}{\sigma_k^2}||\mathbf{x} - \mu_k||^2\right), \tag{4}$$

where μ_k and σ_k denote the center vector and width of each Gaussian function. The input vector in this particular case is a vector containing the subset of the state space variables previously mentioned pertaining to structural component 1, i.e. $\mathbf{x} = \mathbf{p}$. As (3) shows, when summing all Gaussian functions multiplied with their weight factors, an arbitrary function may be approximated. The question then is how to add and prune units as well as adjust currently existing units in order to approximate the function in question.

In the classical RAN algorithm, isotropic Gaussian functions are employed. This implies that the width parameter σ_k is a scalar, which means that in a multivariate Gaussian function all dimensions will have equal width. We elect to use a more general multivariate radial basis function in our approach where we use a diagonal covariance matrix Σ, as this in practice allows for tuning the width of the basis functions on a per-dimension basis. Such a basis function is given by

$$\phi'_k(\mathbf{x}) = \exp\left(-(\mathbf{x}-\mu_k)\Sigma_k^{-2}(\mathbf{x}-\mu_k)\right), \tag{5}$$

where, if $\mathbf{x} \in \mathbb{R}^N$,

$$\Sigma_k = \begin{bmatrix} \sigma_{k,1} & 0 & \cdots & 0 \\ 0 & \sigma_{k,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{k,N} \end{bmatrix} \in \mathbb{R}^{N\times N} \tag{6}$$

denotes the width in each dimension.

Growth Strategy Learning involves both adding units and adjusting network parameters. A new unit is added to the network if the observation $(\mathbf{x}_n, y_n)$ adheres to the following constraints:

$$||\mathbf{x}_n - \mu_{nr}|| > \quad \epsilon_n, \tag{7}$$

$$e_n = y_n - f(\mathbf{x}_n) > \quad e_{\min}, \tag{8}$$

$$e_{\text{rms}n} = \sqrt{\frac{\sum_{i=n-(M-1)}^{n}(y_i - f(\mathbf{x}_i))^2}{M}} > e_{\text{rms,min}}, \tag{9}$$

where ϵ_n and $e_{\min}$ are tunable thresholds for minimum input state distance and minimum output difference, respectively. μ_{nr} is the center of the unit closest to $\mathbf{x}_n$. $e_{\text{rms}n}$ describes the RMS value of the output error over a sliding window of size M. This value should be greater than the tunable threshold $e_{\text{rms,min}}$. The RMS criterion is there to filter out spurious occurances where a wildpoint observation erroneously creates a new unit.

Parameters associated with the newly created unit are:

$$\alpha_{K+1} = e_n, \tag{10}$$

$$\mu_{K+1} = \mathbf{x}_n, \tag{11}$$

$$\text{diag}(\Sigma_{K+1}) = \kappa(\mathbf{x}_n - \mu_{nr}). \tag{12}$$

κ is a tunable factor determining how much the initial responses of units should overlap in the input space. When using an N-dimensional Σ, κ could easily be extended to control the initiation width of each dimension independently. This was, however, not necessary for our case where all dimensions were of equal magnitude and significance.

Adjustment Strategy When an observation $(\mathbf{x}_n, y_n)$ does not meet the criteria in (7)-(9) the currently existing network parameters $\mathbf{w}$ are adjusted instead in order to adjust the existing network to more closely approximate the new value. An EKF algorithm (see Kadirkamanathan and Niranjan (1993)) is used for this purpose:

$$\mathbf{w} = [\alpha_0,\ \alpha_1,\ \mu_1^T,\ \sigma_1, \ldots,\ \alpha_K,\ \mu_K^T,\ \text{diag}(\Sigma_K)^T]^T \tag{13}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + e_n\mathbf{k}_n, \tag{14}$$

where $\mathbf{k}_n$ is the Kalman gain vector given by

$$\mathbf{k}_n = (R_n + \mathbf{a}_n^T P_{n-1}\mathbf{a}_n)^{-1} P_{n-1}\mathbf{a}_n, \tag{15}$$

and $\mathbf{a}_n = \nabla_w f(\mathbf{x})$ is the gradient vector of $f(\mathbf{x})$ with respect to $\mathbf{w}$. R_n is the measurement noise variance and P_n is the error covariance matrix

$$P_n = (I - \mathbf{k}_n\mathbf{a}_n^T)P_{n-1} + QI, \tag{16}$$

where Q denotes a scalar which determines an allowed random step in the gradient vector direction, introduced to prevent the model from converging too rapidly to adapt to new data. P_n must be increased in dimensionality each time a new unit is added. See Kadirkamanathan and Niranjan (1993) for more details on implementing this EKF algorithm.

Pruning Strategy If the output of a unit $\alpha_k\phi(\mathbf{x}_k)$ is less than a threshold over a number of M consecutive inputs, Yingwei et al. (1997) removes this unit from the network. This makes sense in many applications, but for our application even though current observations do not visit some part of our state space in a while it does not necessarily mean they should be forgotten. It may just mean the operator is currently occupying a separate part of the system's state space during an operation.

By the above argument, we choose not to implement a pruning strategy. This makes the implemented function approximator more akin to the RANEKF algorithm of Kadirkamanathan and Niranjan (1993), but criterium 3 of the growth strategy, the noise filtering criterium (see (9)), is exclusive to the M-RAN algorithm.

4. IMPLEMENTATION

This section describes how the learning camera control system is implemented in software in the lab. The camera control system is based on a laboratory architecture allowing an operator to control two industrial robot manipulators. Matlab interfaces for communicating with both manipulators are also given, able to communicate commands in either joint perturbations in the configuration space ($\delta\mathbf{q}$) or wrist perturbations in the operational space ($\delta\mathbf{p}$). Both manipulators' joint configuration states $\mathbf{q}$, or respective wrist operational space states $\mathbf{p}$, may be retrieved. We refer the reader to Bjerkeng et al. (2011b) for more specifics on this architecture.

The algorithm of Bjerkeng et al. (2011b), summarized in section 2.3, is implemented as a Matlab-based controller on top of this architecture. The controller is interfaced through setting the current operation manipulator wrist coordinates, the requested zoom distance, as well as a maximum increment parameter controlling the dynamic properties of the camera platform manipulator.

The implemented algorithm is given in pseudo-code as the Algorithm 1-listing states. The variable p in line 2 refers to the wrist coordinates of the operation manipulator in operational space ($\mathbf{p}$). The zoom distance *zoomDist* (line 3) is the currently estimated best zoom distance of the camera platform by the learning system, while *desiredDist* (line 5) includes possible perturbations given by the operator through the joystick interface. It is assumed here that the nominal joystick position is 0 and that joystick movements j (line 4) in either direction (closer or further

Algorithm 1. Learning camera platform controller loop

```
1:  while controller running do
2:      p = GetRobotState()
3:      zoomDist = policy.Calc(p)              ▷ Eqs. (3-4)
4:      j = GetJoystick()
5:      desiredDist = zoomDist + j*maxChange
6:      err = desiredDist - zoomDist
7:      if |err| > errThreshold then
8:          policy.Train(p, err)               ▷ Section 3.3
9:      end if
10:     CameraCtrl(p, zoomDist)                ▷ Section 2.3
11: end while
```

Table 1. Control parameters of the implemented M-RAN algoritm.

Parameter	Value
ϵ_n	0.4
$e_{\min}$	0.008
$e_{\text{rms,min}}$	0.008
Q	0.4
κ	0.1

away) is given by a scalar normalized to the interval $[-1, 1]$. The parameter *maxChange* may then be used to scale the responsiveness of the joystick.

If the error has a large enough magnitude, governed by *errThreshold*, the algorithm will add a node, or train the existing network of nodes, according to the rules given in Section 3.3. One important note here regards the update rule for operator inputs in line 5; if the operator keeps pushing the joystick in one direction or the other, the controller loop ensures that the desired distance is continuously decreased or increased until the operator is satisfied since the M-RAN policy, and hence the zoom distance, will be updated at each controller step.

5. EXPERIMENTAL RESULTS

5.1 Experimental Setup

The experimental setup consists of the operation manipulator, denoted the Gantry Robot (GR), and the camera platform, denoted the Floor Robot (FR). These are illustrated in Fig. 3. The FR holds a stereo vision camera as illustrated in Fig. 2. The GR is controlled by joystick commands and the FR by the algorithms described in this paper.

The modified M-RAN function approximator is governed by some control parameters. These parameters are reproduced in Table 1 for completeness.

5.2 The Experimentation Case

We choose to look at a specific operation case in our experiments, as learning from an operator is heavily dependent on the operator and operation in question. A given operator and tool is assumed, effectively locking structural component 2 and 3 of the state space. These two components act simply as indices into a two-dimensional array of

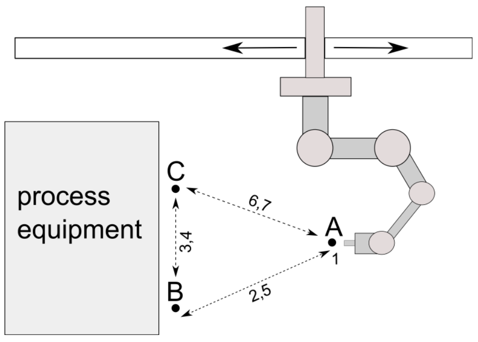


Fig. 5. Illustration of operation robot (GR) movements during experimentation case. Numberings correspond to the transitions and locations described in Section 5.2.

memory functions, and it is straightforward to check the implementation for correctness of this table lookup. We thereby exclude this from the current experiments.

The selected tool for the operation is a laser vibrometer; a tool able to measure vibrations by pointing a red laser beam at any physical location (see Kyrkjebø et al. (2009)). This tool is particularly interesting because it requires the operator to switch between close-up views for aiming the laser at certain locations and zoomed-out overviews for finding the next location to measure.

We emulate an operator by creating a sample course to follow for the GR. The procedure is outlined below, and Fig. 5 illustrates the described experimentation case.

(1) The case starts at location A.
(2) Operator moves to location B for measurement. Uses joystick to get better camera position.
(3) Operator moves to location C for another measurement. Uses joystick to get better camera position.
(4) Operator moves near location B again. Camera position and zoom should be retrieved from memory.
(5) Operator moves near location A for overview. Uses joystick to get better camera position.
(6) Operator moves near location C for measurement. Camera position and zoom should be retrieved from memory.
(7) Operator moves near location A for overview. Camera position and zoom should be retrieved from memory.

5.3 The 0-State Experiment

The goal of this experiment is to validate the algorithm when no learning inputs are given by the operator. In this case the algorithm should behave exactly like the active camera control algorithm of Bjerkeng et al. (2011b). We follow the outline of the experimentation case given in Section 5.2, but without any operator feedback. The results of the active camera control algorithm are compared with the output of the learning system, and should be exactly identical. Table 2 shows the results for the experimentation case. The results are rather trivial, but clearly show that in case of no feedback from the operator the learning algorithm does not influence the original algorithm at all.

Table 2. Results of experiment from Section 5.3.

Transition	Location	Zoom
1	A	0
2	A → B	0
3	B → C	0
4	C → B̃	0
5	B̃ → Ã	0
6	Ã → C̃	0
7	C̃ → Ã	0

5.4 *Learning Operator Preferences*

The goal of this experiment is to show whether operator input has the desired effect on the camera zoom factor. The outline of the experimentation case given in Section 5.2 is followed. Table 3 summarizes the results. As indicated by the table, after feedback has been given by the joystick (transition 2, 3, and 5) the learning system adjusts the preferred camera zoom factor for the current state in the state space (transition 4, 6, and 7). The new zoom distances are now stored by the memory function, and each time the operation robot nears the approximate same area of operation, independent of the route the operator selects for getting to that location, the camera platform will adjust its zoom accordingly.

There is an interpolated transition between the stored state and neighbouring states owing to the generalizing nature of the function approximator. This is illustrated in Fig. 6, where the results of transition 6 are given. The graph shows the time development on the x-axis and the corresponding zoom distance parameter at each time instant on the y-axis. The stair-like look of the graph is due to operator pausing when moving the operation robot (GR).

If the operator were to adjust the zoom factor 'in between' for instance the B and C points, the learning system would, based on the selected parameters for learning, add a new point of interest at this location. This is exactly what happened when point B and C were identified as points of interest that deviated from what the learning algorithm so far believed had the intended zoom factor (transition 2 and 3).

Table 3. Results of the experiment from Section 5.4.

Transition	Location	Zoom
1	A	0
2	A → B	-0.999
3	B → C	-0.999
4	C → B̃	-0.973
5	B̃ → A	0.531
6	A → C̃	-0.974
7	C̃ → Ã	0.529

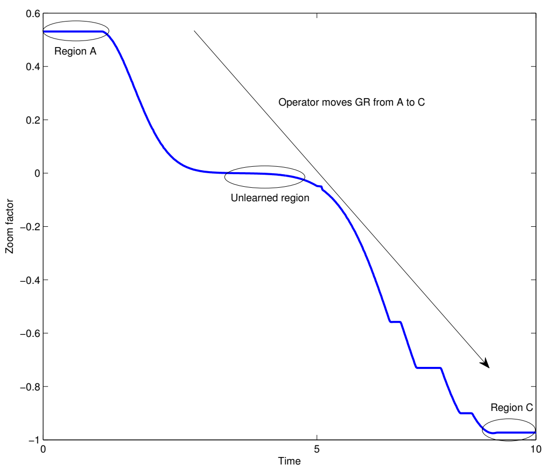


Fig. 6. Result of transition 6 in the experiment from Section 5.4. The x-axis shows the time development when moving from A to C, while the y-axis shows the corresponding zoom distance at each time instant.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a new method of interaction between an operator and remote camera platform for monitoring of manual robot manipulator operations. The method is founded on the idea that the operator should spend as little time and effort as possible on controlling the camera platform to be able to focus on the task at hand instead of positioning the monitoring system. To this end, we have shown the feasability of a learning camera platform by using a modified M-RAN function approximator as memory function in a system architecture for learning.

To demonstrate our approach we have equipped an industrial manipulator with a stereo vision camera and designated it as a follower robot for another joystick controlled industrial manipulator. Experimental results on a demonstration case indicate that the camera platform responds to and remembers differences in operator preference.

6.1 *Future Work*

In the current implementation of the active camera control algorithm, the learnable parameters are limited to the zoom factor. This limitation will be alleviated in future works, so that more freedom is given to the operator. We are currently looking at learning positioning of the camera platform on a virtual manifold around the TCP of the operation manipulator in order to learn the camera platform to avoid situations of occlusion.

The camera platform's ability to differentiate situations an operator sees as distinctive is heavily dependent on the choice of learning state space, as previously mentioned. Looking at automated techniques for selecting important state dimensions from a potentially larger set of candidate dimensions, using dimensionality reduction techniques (see e.g. Fu and Wang (2003)), may yield greater applicability of the algorithm for more cases, as well as relieve some of the guesswork of the designer. In cases where latent state space features are present, more advanced techniques than

RBF networks are needed. We are currently looking into the use of LO-nets (Ray and Oates, 2011) for predicting operator behaviour considering latent variables.

A more complete experiment suite with several operators and a set of different operations is also in order to better verify if the method is tractable in a real-life setting.

REFERENCES

Abbeel, P. and Ng, A.Y. (2005). Exploration and apprenticeship learning in reinforcement learning. In *Proc. 22nd international conference on machine learning*.

Anisi, D., Gunnar, J., Lillehagen, T., and Skourup, C. (2010). Robot automation in oil and gas facilities: Indoor and onsite demonstrations. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.

Argall, B.D., Chernova, S., Veloso, M., and Browning, B. (2009). A survey of robot leaning from demonstration. *Robotics and Autonomous Systems*, 57(5), 469–483.

Bishop, C.M. (2007). *Pattern Recognition and Machine Learning*. Springer.

Bjerkeng, M., Pettersen, K.Y., and Kyrkjebø, E. (2011a). Stereographic projection for industrial manipulator tasks: Theory and experiments. In *Proc. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Bjerkeng, M., Transeth, A.A., Pettersen, K.Y., Kyrkjebø, E., and Fjerdingen, S.A. (2011b). Active camera control with obstacle avoidance for remote operations with industrial manipulators: Implementation and experimental results. In *Proc. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Fjerdingen, S.A., Kyrkjebø, E., and Transeth, A.A. (2010). AUV pipeline following using reinforcement learning. In *Proceedings for the joint conference of ISR 2010 and ROBOTIK 2010*.

Fu, X. and Wang, L. (2003). Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 33(3), 399–409.

Graf, B. and Pfeiffer, K. (2008). Mobile robotics for offshore automation. In *Proc. IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance*.

Kadirkamanathan, V. and Niranjan, M. (1993). A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6), 954–975.

Kyrkjebø, E., Liljebäck, P., and Transeth, A.A. (2009). A robotic concept for remote inspection and maintenance on oil platforms. In *Proceedings of the ASME 2009 28th International Conference on Ocean, Offshore and Arctic Engineering*.

Platt, J. (1991). A resource allocating network for function interpolation. *Neural Computation*, 3, 213–225.

Ray, S. and Oates, T. (2011). Discovering and characterizing hidden variables using a novel neural network architecture: LO-Net. *Journal of Robotics*, 2011, 1–16.

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press, London, England.

Thomaz, A.L., Hoffman, G., and Breazeal, C. (2005). Real-time interactive reinforcement learning for robots. In *AAAI 2005 Workshop on Human Comprehensible Machine Learning*.

Yingwei, L., Sundararajan, N., and Saratchandran, P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation*, 9(2), 461–478.