Making Robots Learn to See

Gunther Heidemann and Helge Ritter

1 Introduction

Machine vision will be a key technique in future robotics. Though we are still far from the construction of robots with human capabilities, machine vision already plays an increasingly important role in many robot applications. In this contribution, we will outline a man machine interaction scenario as an example what can be achieved by now using adaptive methods in machine vision. We will discuss the problems of current vision systems and outline possible future developments.

The first question to be discussed is why robots have to see at all, or, in other words, what are objects or actions that might be interesting to robot vision. Part of the answer is straightforward: a robot needs vision to analyze its surroundings, i.e. recognize objects, find landmarks for navigation, detect movements and so on. Besides this, however, a robot will have to carry out actions using its end effectors (e.g. grippers or anthropomorphic hands) and has to control these actions. This can be achieved only if feedback on the actions is available. Humans use a sophisticated combination of visual and haptic feedback for the manipulation of objects. As humans can handle objects skillfully even without any vision using haptics only, it can be concluded how powerful human haptics really is. This points out a major shortcoming in current robotics: still, haptic sensors are crude in comparison with natural systems. However, in technical systems vision can replace haptic capabilities to a certain extent: Fig. 1 shows a three-fingered robot hand with a camera mounted on top which gives a detailed view of the fingertips and the manipulated object. Hence, in a way the range of visual sensing is even larger in robotics than in nature.

Another field of machine vision applications is the interaction between robots and humans. To facilitate natural instruction of robots, the visual recognition of hand gestures, gaze direction and facial expression is essential. In this contribution, all these three fields of robot vision will be addressed:

- Scene analysis
- Visual feedback for the control of a robot end effector
- Interaction with humans

Today’s machine vision is essentially computer vision (CV), i.e. most of the visual processing is performed on standard von Neumann-like computers, whereas specialized vision hardware is more exceptional (at least for research). Therefore, we will first describe a human-machine interaction scenario which illustrates several CV applications. Then, some of the basic problems in CV will be outlined and solutions presented. Finally, possible future directions of development will be discussed.
2 Human–Machine Interaction in the Scenario of SFB 360

In the research project Sonderforschungsbereich “Situated Artificial Communicators” (SFB 360) of the Deutsche Forschungsgemeinschaft (DFG) we use a setup which allows to explore multiple aspects of robotics, vision, haptics, speech recognition and -production as well as human-machine interaction. As a sample task a human instructor tries to tell a robot constructor to build an airplane from wooden toy pieces\(^1\) (Fig. 2). The instructor uses speech and hand gestures to communicate with the robot, but is not allowed to manipulate the toy pieces. In the version shown in Fig. 2, the robot construc-

\(^1\) Baufix system. Baufix is a registered trademark of the Lorenz company.
tor uses a three-fingered hand to manipulate the objects. Both instructor and constructor are allowed to inspect the scene visually, in an advanced version the robot will also use speech production for further inquiries about the instructors intentions. We will outline as much of the current setup as necessary in more detail, an overview is given in [28].

2.1 Hardware Setup

The instructor is standing in front of a table on which toy pieces of the Baulix system are presented. The robot arm with the hand is on the opposite side of the table. To the right of the instructor a binocular active camera head is mounted on a tripod. The AV-head was constructed by Christensen et al. at Aalborg University [3, 4]. It has four degrees of freedom: pan, tilt and 2× vergence. The camera signals are digitized and processed by a Datacube MaxVideo 200 with Digicolor² which is coupled via a VME-bus with a Sun Sparc Station as host computer which does also part of the low level processing. For later processing steps, the images are sent to a standard Linux-PC with Intel Pentium III processor. In addition, an overhead camera is mounted at the ceiling looking downwards at the scene for classification of single objects and parts of objects. The images are digitized by a standard frame grabber card on another PC.

We use a standard industrial robot arm Puma 560 with six degrees of freedom operated in real time in a Unix environment by means of the RCCL-software package [13]. The end-effector is an oil-hydraulic robot hand developed by Pfeiffer et al. [16]. It has three equal fingers mounted in an equilateral triangle, pointing in parallel directions (Fig. 1). The fingers are approximately human-sized, each finger has three degrees of freedom: bending the first joint sideways and inward at the wrist, and bending the coupled second and third joint. The oil pressure in the hydraulic system serves as a sensory

² Datacube, MaxVideo and Digicolor are trademarks of Datacube Inc.
feedback. Additionally, there are position sensors on the hydraulic motor pistons, however, due to hysteresis effects these measurements allow only a qualitative estimation of the joint angles.

To compensate for this lack of accuracy, a fingertip force feedback system was developed by Jockusch [10]. The sensors measure the vertical force and two horizontal torque components sampled at about 500 Hz. The sensors, robot hand and arm are operated by a Sun-Solaris workstation.

To guide the grasping procedure, at the end effector a camera is mounted which can see the object and all fingertips using a wide-angle lens (Fig. 1). Additionally, there is also a force/torque wrist sensor which is not used in the current setup.

2.2 Current Functionality

Our setup covers the three aspects of vision named in Sect. 1. Scene analysis is carried out by the AV-head and the overhead camera. The AV-head shifts its fixation point automatically from one object to the next as long as there is no action. As soon as a hand appears, it will be tracked by the AV-head and analyzed for a pointing gesture (human-machine interaction). If such a gesture occurs, the pointing direction will be evaluated and the best match object will be grasped by the robot arm. To guide the grasping the hand camera will be used (control of end effector). A second hand gesture indicates where to put down the object. The overhead camera is used to classify the objects using the NESSY software, this will be the basis for future extensions when speech will be incorporated in guiding the robot ("take the red bolt").

In the following, we will first try to give insight in the problems of CV, then the SFB-setup will serve as an example for contemporary solutions.

3 Object Recognition — a Basic Problem of Computer Vision

The scenario outlined in the last section clearly demonstrates the importance of computer vision (CV) for robotics, not only as the "eyes" but also to guide a grasping process. However, up to now the capabilities of vision systems are limited, sometimes in a way that is not transparent at first glance. We will discuss some of the limitations of the vision modules and point out the principal problems behind. Figures 3–5 give an impression of the task of CV. You can see the image grey values as a matrix, which is the digitized camera output and as a mesh. Both presentations can hardly be interpreted by humans, only the printed image makes sense to us. The challenge of CV is to bring such "sense" into the grey value matrix.

3.1 Visual Degrees of Freedom

Recognition of objects is a key ability in CV. As the human visual system solves this task perfectly, we usually don’t realize its full complexity. In CV, "recognition" means comparing the pixel appearance of an object with a memorized representation. The

\[^3\text{NEural viSion SYstem, developed in research project A1 of SFB 360, [9].}\]
trouble is, that the pixel appearance of an object is subject to considerable changes depending on the exact conditions under which the image is taken. As an example, Fig. 7 shows the upper part of an object (hole of a Baufix cube) under changing lighting direction. As a light source we used a single halogen lamp with a sharp beam, which was moved in a half circle around the object by a Puma 360 robot arm (Fig. 6). It becomes clear that the pixel appearance of the object changes substantially, even though it is in exactly the same location in all images.

In other words, the pixel-appearance of an object has several "visual degrees of freedom" (DOF): Three for translation, i.e. two for translation of the object in the image plane plus scale (distance), another three for rotations, and countless more for lighting conditions. The latter cannot be counted in the way of the first six DOF since the number, dimensions and positions of the light sources have to be taken into account as well as brightness and spectra.

Even if the lighting conditions are fixed for simplicity, the variability of the pixel appearance caused by translation and rotation is a formidable problem: In a straight

---

A good visualization of the rotational degree of freedom can be found in [18] by projection of images of objects at different angles from the Columbia Object Image Library [20] onto the first eigenvectors.
Fig. 4. The image of Fig. 3 as a mesh . . .

Fig. 5. . . and as grey value image.

forward matching approach, enough appearances of all objects would have to be memorized to fill the “product space” of all DOF sufficiently. Recognition would mean the comparison between the input image and all memorized appearances; so the “curse of dimensionality” strikes. This is also known as the what-where problem: The question, where (and in which pose) an object is located has to be answered simultaneously with the question for the object type.

To reduce the immense computational effort of such a simple matching procedure, mainly two methods can be applied: (1) separation of the “what” and the “where” problem, (2) improving the efficiency of the matching itself by means of projecting the
image patch with the object to a lower dimensional feature space, thus exploiting redundancy in the image. We will deal with the separation problem first.

3.2 Separation of the “What-Where” Problem

Finding an object’s identity, location and pose is not a separable problem in a strict sense. In principle, the whole product space has to be taken into account, but in many cases there are cues like color which allow to find the location first. So in a first processing stage the where-problem can be solved and further processing steps can find out the object’s identity and pose. However, here we encounter a new problem: If the hunt for an object’s location concentrates on highly specific features — like a special color — then only very few objects (and poses) can be found, the rest won’t pass the color filter. In other words, a great part of the entire recognition has been done already by the color filter at the price of a very small object domain that can be recognized at all. By contrast, less specific features will allow the location of more — and more different —
objects, but in this case the background must be simple enough not to be confused with the objects. In practice, this dilemma leads to an unsatisfying procedure of hand-tuning e.g. color parameters, trying to catch the objects but not too much background.

For these reasons, we do not try to solve the where-problem entirely in a single step in our approach. Instead, we concentrate on features which are common to many objects and help to detect image regions which might be interesting in the sense that there might be relevant objects. An example for such a feature is symmetry (see e.g. [32]). Symmetry is not too specific a feature but nevertheless a good indication for regions of interest (ROI) since most man made and also many natural objects (like faces) are highly symmetric. Nature has made use of this fact in guiding human attention by symmetry as shown e.g. in [2, 14]. For these reasons, we made symmetry detection an essential component of the approach for ROI detection described below.

The benefit of methods like symmetry detection in solving the what-where problem is their universality. The independence of a special object domain makes it possible to cover a wide range of applications and to adapt to a new domain by means of relatively simple to adjust parameters. The price, however, is that not only relevant objects will be found by such methods but also others, hence, the task of the classifier becomes more difficult because it will involve the rejection of non-domain objects.

In our approach, in a first processing step so-called focus points (FPs) are generated on image locations that might be interesting, then in a second step image patches around the FPs are the input to a neural classifier which identifies the object or rejects the patch as “unknown”. The concept of focus points helps to solve part of the localization-identification problem by domain independent methods and postpone a “hard” decision to later processing which can be trained more specifically to the image domain.

A FP can be considered as the “condensation” of an ROI. That means, the FP is located within an ROI — which was detected as such by means of low level features — at an outstanding position. To be more precise, we will give some requirements that FP should fulfill:

- **Specificity.**
  - All objects of the domain have to be detected by the FP-generation module.
  - By contrast, we do not require FPs to be located only on objects of the domain.

- **Stability.**
  - FPs should facilitate reliable classification. Consequently, they have to be robust against noise, lighting and object rotation. The latter means that FP should not vanish or move under rotations within a reasonable angular range.
  - A specific position of FPs on an object is not required (only a stable one).
  - If an FP disappears due to object rotation, there should be a stable new one.
  - It is not required that there is only one FP on one object or that all FPs are stable. It is enough if there is one stable FP on each relevant object in all its poses.

These criteria ensure that the task of the classification system becomes much simpler than memorizing the entire six DOF of each object because every object appears only in a few constant (though arbitrary) poses. As an example we will outline in the next section an algorithm which fulfills the named criteria.
4 Generating Focus Points by the ECS-Algorithm

The proposed method for the generation of FPs combines the three image features entropy, color and symmetry (ECS-algorithm) [9]. The task of the first processing stage is to find ROI within which the computationally more costly algorithms will be applied. An overview is shown in Fig. 8.

4.1 Entropy Map

For the purpose of fast ROI-segmentation the local Shannon-Entropy appears to be well suited under the assumption that relevant image parts also have a high information content in the sense of information theory. This is demonstrated e.g. in [11] for traffic scenes. Calculation of an entropy map is based on an — usually low resolution — intensity image $I(x, y)$:

$$B_E(x, y) = -\sum_q \tilde{H}(x, y, q) \cdot \log \tilde{H}(x, y, q), \quad \tilde{H}(x, y, q) = \frac{H(x, y, q)}{\sum_{q'} H(x, y, q')}$$

(1)

where $H$ denotes the histogram within a $n_E \times n_E$-window ($n_E > 3$ and odd) around the pixel $(x, y)$:

$$H(x, y, q) = \sum_{y'=y-\tilde{n}}^{y+\tilde{n}} \sum_{x'=x-\tilde{n}}^{x+\tilde{n}} \delta_{I(p(x', y'))}, q,$$

(2)

with $\tilde{n} = (n_E - 1)/2$ and $q = 0 \ldots 2^{Q_E} - 1$ ($Q_E$ denotes the quantization of the histogram which should be about $2^{Q_E}/n_E^2 \approx 10 - 20$.

Figure 9 shows an intensity image, its entropy map and the original multiplied by the binarized entropy map. Obviously, the relevant parts of the image could be segmented. The crucial parameter in entropy calculation is the window size $n_E$ in combination with the resolution of the intensity image. It determines the scale on which structures are evaluated. A window which is too small to capture object structure is mainly working as an edge detector, by contrast, too large a window leads to a blurred segmentation.

4.2 Color Segmentation

To generate FPs within the segmented regions, two further algorithms are applied to the high resolution color image: color segmentation and symmetry detection. In color segmentation, there are two major classes of algorithms, which are either based on color classification or on local color comparison. Using the Color Structure Code (CSC) developed by Priese and Rehrmann [22, 24] we apply a method of the latter type since it avoids the adaptation to colors of a special domain. The version used here is an implementation of Guido Menkhaus and Tim W. Nattkemper [15, 19].

The CSC uses hierarchical region growing to pool neighboring pixels of similar colors within one region. Region growing, however, suffers from the problem of chaining:
Fig. 8. Focus point generation by the ECS-method. The image is segmented for regions of interest (ROI) by an entropy map calculated at a coarse resolution. Within the ROI, focus points (FPs) are generated using the Color Structure Code (CSC) and a symmetry map. FPs from both modules are integrated in the last processing stage.

As it is possible that a “chain” of pixels which are locally similar in color connects two distant pixels of entirely different colors, inhomogeneous regions might emerge as long as the segmentation relies on local methods alone. Consequently, the CSC checks all regions emerged from the growing procedure and splits inhomogeneous regions top down along the hierarchy. For details of the rather complicated method see [22, 24].
Fig. 9. *Left:* Original grey value image, *middle:* Entropy filtered image (bright — high local entropy), *right:* regions selected by entropy filtering.

Fig. 10. Segmentation into regions of homogeneous color using the Color Structure Code (CSC). FPs are generated at the centers of image regions. *Left:* Without color contrast filter, FPs are in all regions. *Right:* Using the filter, FPs are assigned only to regions with significant color contrast compared to the surrounding regions.

The result is a complete segmentation of the image into regions within which color varies only within a predefined tolerance $\theta_{CSC}$. To improve results, CSC-segmentation is usually preceded by image enhancement with a symmetric nearest neighbor filter [6, 31] which smooths noisy colors but preserves edges.

What good are homogeneous color regions for FP-generation? Color plays an important role in human perception, attention is caught by regions of colors with high contrast to their surroundings. Therefore, the color regions of a CSC-segmented image could be used for FP-generation. An FP can be set to the center of gravity of a color region. However, not all regions are suited equally well to provide stable FPs. As demonstrated in Fig. 10, left, there are regions which have hardly any color contrast compared to the surrounding regions, especially in the lower part of the image in the shadow of the toy airplane. Such low contrast regions are bound to appear if there is a continuous transition between different colors (or intensities): The top down splitting of regions with chaining will enforce borders to keep the tolerance $\theta_{CSC}$, even if these
borders are not motivated by a strong local change of color. Such borders are highly unstable against slight changes in the scene or lighting. Hence, the regions will not provide stable FPs.

To filter “salient” color regions from the resulting CSC-segmentation, we calculate the average local color contrast on each region border. To be more precise, a region \( R \) is specified by the \( n_R \) pixels \( p_1 \ldots p_{n_R} \) which form its border. For each of these pixels \( p_i \) the local color contrast \( F(p_i) \) is calculated from the RGB-image by a convolution of each of the color channels with a \( 5 \times 5 \) Laplace filter. The result are edge strength values \( R_K(p), G_K(p), B_K(p) \) for red, green and blue channel. The average local color contrast along the region border \( F(R) \) is

\[
F(R) = \frac{1}{n_R} \sum_{i=1}^{n_R} F(p_i), \quad F(p) = \sqrt{R_K^2(p) + G_K^2(p) + B_K^2(p)},
\]

which can be used as a measure for the “saliency” of the region. FPs are generated only from regions with \( F > \theta_{\text{contrast}} \).

### 4.3 Symmetry Map

The third component in the ECS-Algorithm is calculation of the local symmetry. While CSC looks for homogeneous, prominent colors, symmetry detection is based on low level form features. Using symmetry is motivated by insights into human attention in which symmetry plays an important role [2, 14]. We adopted a method developed by Reisfeld et al. [25] for the calculation of a symmetry based saliency map \( B_S \) from grey value images and enhanced performance by evaluation of color information.

The saliency map \( B_S(p) \) contains a real valued symmetry measure for each pixel the calculation of which can be outlined only in a simplified version:

\[
B_S(p) = \sum_{(p_i, p_j) \in U_S(p,r)} \text{PW} F(p_i, p_j) \cdot \text{GW} F(p_i) \cdot \text{GW} F(p_j) \cdot \text{CW} F(p_i, p_j).
\]

Fig. 11. Focus points generated from local symmetry detection.
The calculations are carried out within a circle $U_c$ with $p$ in the center, thereby all pairs of opposing pixels are evaluated. The Phase Weight Function $PW F$ gives a judgement on the relative angles of the grey value gradients at $p_i$ and $p_j$. Gradients that might belong to a symmetric object contribute a high symmetry rating to the central pixel. To enhance the contribution of pixels on intensity edges, the Gradient weight function $GW F$ weights the contributions of the gradients logarithmically. Reisfeld et al. use some more grey value based factors which proved, however, not to yield contributions which are relevant for FP-generation. In Eq. (4) there is another factor, the Color Weight Function $CW F$ as introduced by Nattkemper [19]:

$$CW F(p_i, p_j, \theta_{sym}^{Euc}) = \begin{cases} 0 & \text{for } \Delta_{Euc}(p_i, p_j) > \theta_{sym}^{color} \\ 1 & \text{else} \end{cases},$$

where $\Delta_{Euc}$ denotes the Euclidean color distance. Due to the factor $CW F$, only pixels of similar colors can contribute to the symmetry judgement. That way FPs which are e.g. caused by the symmetry of an object and its own shadow can be avoided.

The symmetry map is binarized by an appropriate threshold, then, FPs are generated as the centers of blobs (above the threshold) which have a certain minimum size. Figure 11 shows a symmetry map and the corresponding FPs.

### 4.4 Integration of FPs

As shown in Fig. 8, the FPs of both the CSC- and the symmetry branch are merged in a separate module. As prominent color regions and symmetry are substantially different image features, collisions of FPs are exceptional in most domains. If, however, FPs from both modules come too close together, one of them will be suppressed. Therefore, either CSC or symmetry must be defined to dominate (symmetry in the application described here).

Stability of the FPs was tested using the robot setup described in Sect. 3.1. Two types of test image sequences were used: Images from a camera moved by the robot around the object to test for rotational stability and images from a still camera while the robot moved the halogen lamp around the object. For the example of the Baufix cube shown in Fig. 7 we obtained about four to six FPs on the object in each image. For each angle (either of the camera or the lamp) there was at least one FP stable over three shots. In most cases the stable FP was in one of the holes, originating from symmetry detection, whereas the CSC did not contribute much to stable FPs due to inhomogeneous lighting. By contrast, when the experiments were repeated with the toy airplane shown in Fig. 10, the CSC contributed stable FPs located on the colored bolts. From systematic evaluation could be concluded, that the FPs solve the task of a stable localization, however, at the cost of several additional unstable FPs. To sort out such FPs is the task of the classifier.

### 5 An Approach to Data Driven Filter Design and Classification

The recognition system proposed here consists of three processing stages: (1) FP-generation, (2) extraction of features in a window around each FP and (3) classific-
Fig. 12. Architecture of the VPL-classifier. To classify an input window, first the best match reference vector is determined. Then features are extracted from the input window by a PCA-net attached to the best match reference vector. The resulting feature vector is classified by an (expert) LLM-net.

tion of these features by an artificial neural network [9]. The latter two points will be outlined in this section.

Artificial neural networks (ANNs) are well-suited for classification tasks in computer vision. Their adaptivity has two major advantages: (a), ANNs acquire the necessary knowledge from examples and therefore can be adapted to different object domains, (b), because of the implicit representation ANNs can store even knowledge that can hardly be modelled explicitly by a human designer. Since for the most tasks ANNs cannot be applied directly to the raw pixel data, suitable features have to be extracted before the classification itself. However, the benefit of adaptivity gained by the application of ANNs may vanish if the required feature extraction has to be designed from scratch for a new vision problem. Hence, feature extraction should have the same adaptivity as the neural classifier.
In our approach, feature extraction and classification are joined in a three stage architecture. All of these processing stages can be adapted to the recognition task by examples. In the first level, the input data are pre-structured by a vector quantization. Then, features are extracted by local principal component analysis which are classified by expert ANNs in the last processing level.

Principal component analysis (PCA) is a standard method in computer vision to project an image patch (window) into a lower dimensional feature space. As PCA captures the directions of greatest variance in the data space, the resulting feature vectors are still highly specific but reduce redundancy for the benefit of lower computational effort and better stability. A well known example is face classification using so called “eigenfaces”, i.e. filters originating by PCA from face sample images [30]. Normal PCA is the optimal linear method to approximate a data set in the least squares sense. Local PCA can be viewed as a nonlinear extension of normal (global) PCA [29]. A related example is given in [17].

Figure 12 shows the VPL-architecture (VPL – Vector quantization, PCA, LLM, see Sect. 5.3). In the following, these processing levels will be described in more detail.

### 5.1 Vector Quantization

The input data are roughly approximated by $N_{vq}$ reference vectors $W^{vq}_{ij}$, $i \in [1, N_{vq}]$, which are positioned by the vector quantization learning rule

$$
\Delta W^{vq}_{n_{n}} = \epsilon (x - W^{vq}_{n_{n}}), \quad \epsilon \in [0, 1],
$$

(6)

where $n_{n}$ is the index of the best match reference vector. Since application of Eq. (6) alone tends to be caught in local minima of the mean square error function for the approximation error, we enhanced the algorithm by a so called “Activity Equalization”. In short, this method re-initializes the nodes if they seldom or never become the “winner”, so the average node activities will be equalized during training. For details see [7, 9].

An input vector $x \in \mathbb{R}^{w^2}$ represents an image window of size $w \times w$, or, alternatively, $x \in \mathbb{R}^{3 \times w^2}$ if three color channels are evaluated. $x$ is mapped by the VQ to an integer number $n_{R} = 1 \ldots N_{vq}$ ($n_{R}$ denotes the best match).

### 5.2 Local PCA

Sanger [27] proposed a single layer feed forward network for the successive calculation of the principal components (PCs) of training vectors. The nodes have a linear activation function

$$
V_{i} = \sum_{j=1}^{d} W_{ij}^{pca} x_{j}, \quad i = 1 \ldots N_{pca},
$$

(7)

where $W_{ij}^{pca}$ are the input weight vectors of the nodes and $x$ the input (see last section). After training by Sanger’s rule

$$
\Delta W_{ij}^{pca} = \epsilon V_{i} \left[ x_{j} - \sum_{k=1}^{i-1} V_{k} W_{kj}^{pca} \right] - V_{i} W_{ij}^{pca}
$$

(8)
the weight vectors approximate the PCs in the order of their eigenvalues, beginning with the largest. The output $\mathbf{V} \in \mathbb{R}^{N_{\text{PCA}}}$ of the network is the projection of the input $\mathbf{x}$ to the first $N_{\text{PCA}}$ PCs with the largest eigenvalues.

5.3 LLM-Nets

The Local Linear Map is related to the self-organizing map [12] and the GRBF approach (e.g., [21]). It can be trained to approximate a nonlinear function by a set of locally valid linear mappings, for details see e.g., [26]. For the classification task, a mapping from the $N_{\text{PCA}}$-dimensional input vector $\mathbf{V}$ to an $N_{\text{cl}}$-dimensional output vector $\mathbf{y}$ is required ($N_{\text{cl}}$ = number of classes). The target training vector for class $c$ has the form $y_j^{(c)} = \delta_{jc}, j = 1 \ldots N_{\text{cl}}$, where $c$ denotes the number of the training example.

![Image](image.png)

**Fig. 13.** The local linear map (LLM) approximates a nonlinear mapping by several locally valid linear mappings.

An LLM-node $i$ has an input weight vector $W_{i,\text{in}}^{\text{llm}} \in \mathbb{R}^{N_{\text{PCA}}}$ and a linear function to compute the output, which consists of the output weight vector $W_{i,\text{out}}^{\text{llm}} \in \mathbb{R}^{N_{\text{cl}}}$ and a matrix $A_i \in \mathbb{R}^{N_{\text{cl}} \times N_{\text{PCA}}}$. The network output is calculated from the input vector $\mathbf{V} \in \mathbb{R}^{N_{\text{PCA}}}$ by a search for the best match node $k$, which is determined by

$$k = \arg \min_{i=1 \ldots N_{\text{llm}}} (||\mathbf{V} - W_{i,\text{in}}^{\text{llm}}||),$$  

(9)

where $N_{\text{llm}}$ is the number of LLM-nodes. The output vector $\mathbf{y} \in \mathbb{R}^{N_{\text{cl}}}$ is then

$$\mathbf{y} = W_{k,\text{out}}^{\text{llm}} + A_k (\mathbf{V} - W_{k,\text{in}}^{\text{llm}}).$$

(10)
Given correct input-output pairs of the form \((V^{(a)}, y^{(a)})\), the best match node \(k\) of the network is adapted supervised with the adaptation step sizes \(\epsilon_{in}, \epsilon_{out}, \epsilon_{A} \in [0, 1]\):

\[
\Delta W_{k}^{lin, in} = \epsilon_{in}(V^{(a)} - W_{k}^{lin, in}),
\]

\[
\Delta W_{k}^{lin, out} = \epsilon_{out}(y^{(a)} - y(V^{(a)})) + A_{k} \Delta W_{k}^{lin, in},
\]

\[
\Delta A_{k} = \epsilon_{A}(y^{(a)} - y(V^{(a)})) \frac{(V^{(a)} - W_{k}^{lin, in})^{T}}{\|V^{(a)} - W_{k}^{lin, in}\|^2}.
\]

5.4 Training Procedure

Given a training set \(T\), the neural VPL-architecture shown in Fig. 12 is adapted in three stages:

1. VQ of the input space, then divide \(T\) into subsets \(T_{1}, \ldots, T_{N_{ca}}\) which contain the best match examples for the obtained reference vectors \(W^{(eq)}\).
2. Train one PCA-net for each subset \(T_{i}\). Compute the sets \(T_{i}^{e}\) of the (input-) projections of all examples in \(T_{i}\) to the first \(N_{pca}\) PCs of the corresponding PCA-nets.
3. Train one LLM-net for each training set \(T_{i}^{e}\).

The combination of the VQ with subsequent PCA-nets leads to a local PCA within the Voronoi tessellation cells in input space.

6 Applications of NESSY

In Sects. 4 and 5 we outlined the two major components of the vision system NESSY. For the application to a specific domain, first the ECS-module (Fig. 8) has to be parameterized appropriately, which means mainly specification of the “scale of interest” and the color parameters. Once the ECS-module yields enough stable FPs, windows centered at the FPs on training images are used to adapt the VPL-module (Fig. 12).

In the human-machine interaction scenario outlined in Sect. 2, NESSY is used for the classification of the toy objects. Fig. 14 shows an example. As the objects are not isolated in most cases but bolted together, it would not be useful to train the system for the recognition of complete objects. Instead, parts of objects that are likely to be visible even in aggregates were trained, like heads of bolts, holes of bars or cubes and corners of cubes.

Another application of NESSY is demonstrated in Fig. 15. Here, the images are captured by the hand camera (see Fig. 1) which is looking downwards on the fingers and the grasped object. In order to check the grasp on the object, the haptic sensory system of the hand (fingertip sensors and oil pressure, see Sect. 2.1) is supplemented by evaluation of the camera image. There are mainly two questions which have to be answered by vision: (1) type of the object and (2) stability of the grasp.

To solve this task, several different VPL-classifiers are employed. The input to all of them is the entire image (as shown in the lower row of Fig. 15), there is no need to generate FPs as not details are to be classified but the whole situation has to be judged. Object recognition is carried out by a \(VPL_{O}\) which triggers the use of the other VPLs:
For each object type $i$, there is one $V \mathcal{L}_i$ which is trained to judge the grasp on this object. Figure 15 shows different situations for a Baufix cube: Above, hand and object are shown from the side, below, as seen from the hand camera. In the left images, the grasp is optimal as the fingers are in three of the holes of the cube. By contrast, the grasp in the right images is suboptimal as one of the fingers has lost contact. This difference can be seen clearly from the side but the difference is small from the position of the hand camera. However, it turns out that the individual training enables the $V \mathcal{L}_i$ to judge the grasping positions reliably in the three grades perfect – suboptimal but still stable – unstable.

So far, the first two categories of CV outlined in Sect. 1 have been dealt with: scene analysis and visual control of the end effector. In the following, we will describe the interaction with humans.

7 Vision for Human-Machine Interaction

In the setup outlined in Sect. 2, the active camera head is used for scene surveillance, hand tracking and the evaluation of pointing gestures. Due to its complexity, only the basics of this system can be outlined in this contribution, for details see [5,23,28].

As long as there is no action, the AV-head surveys the scene and locates the objects on the table using the “saccade subsystem”. If a hand appears, attention shifts to the
hand and the “pursuit subsystem” is activated. A moving hand will be tracked, but as soon as the movement ceases, the hand posture is evaluated.

The active scene exploration is controlled by the saccade subsystem. It is based on a layered architecture (Fig. 16) which transforms the stereo input to several “feature maps” indicating edges, color saturation and intensity, motion (difference map) and skin color. As the system is to recognize human hands, another feature map is calculated by multiplication of the skin segmentation map (indicating skin) and the difference map (indicating motion). The final attentional map is formed by a weighted sum of the feature maps which is multiplied with a manipulation map and a fade out map.

The maximum of the resulting attentional map is used to determine the next fixation point. The fade out map indicates regions that were “visited” by the movements of the AV-head, so “saccades” are carried out moving the gaze from one interesting point to the next. As the most important goal is hand tracking, the “moving skin” map is superimposed to the fade out map to keep interest in the hand. Tracking of the hand itself is performed by the pursuit subsystem which is realized by zero disparity filtering implemented on the Datacube. Using vertical edges, regions of zero disparity can be detected which are used by the pursuit subsystem to estimate the depth movement for the next fixation.

The tracked “hand-region” is analysed by a multi-layer perceptron to decide whether a pointing gesture is carried out. If so, from the geometric form of the segmented hand
region the pointing direction is computed. To shift attention in the indicated direction, the “manipulation map” is used which is overlaid to the attentional map. By these means a saccade in the indicated direction is carried out towards the “best matching” object in the attentional map. From the fixation of the object, its 3D-coordinates can be estimated for the control of the robot, which uses the hand camera during the approach movement.

8 Conclusion and Outlook

8.1 Conclusion

The aim of this contribution is to clarify some of the problems of robot vision, to give an impression of the state of the art of CV and discuss possible future solutions. We named three major goals of robot vision (scene analysis, control of end effectors and interaction with humans) and exemplified these goals using a human-machine interaction scenario.

One of the major problems present in all CV applications is the great gap between the pixel appearance of objects and the high level interpretations needed by robots to carry out actions. Dealing with the pixel appearance, all problems have to be solved “simultaneously”: What object, where, in what pose? We confronted this problem in all of the outlined applications and found answers for some aspects. The NeSSY recognition system reduces the complexity of the task by restrictions to the scene like fixed lighting, a fixed camera position, a limited set of objects and a fixed scale (i.e. size of the objects). With these restrictions, however, notable results could be achieved like the classification of parts of objects in relatively complex situations and the judgement of quite abstract situations like the “quality” of a robot grasp on an object.

The system for hand tracking and pose evaluation described in Sect. 7 solves an even more difficult task. This can be achieved by some further restrictions: Attention of the system has to be caught first by slow movements and the pointing gesture must be precise and held for some seconds to give the system time for the processing.
Fig. 17. Hand tracking and gesture evaluation. Above: Attention of the AV-system is caught by moving the hand (left), then a pointing gesture towards one of the objects is carried out (right). Below: Left: From the color images of both cameras the skin colored regions are segmented and the pointing direction is determined. An “activation cone” in the pointing direction is used in the manipulation map to shift attention to the object. Right: The AV-head has fixated the indicated object. From [25].

These examples show, that although CV cannot be compared to human vision by far, already solutions to single, clearly restricted problems are possible. Location and recognition of slightly occluded objects and even difficult tasks like shifting attention by pointing become possible if constant image acquisition conditions are provided. All of the outlined applications are based on adaptive system which “learn” the task from training images, however, the learning process still requires the assistance of the human “knowledge designer” who selects the learned primitives by parameterization of the modules as well as by the design of the entire setup.

As the importance of adaptivity and learning becomes clear, the question arises what role they will play in future vision systems. So what challenges of CV are ahead? Figure 18 shows some directions of development arising from current problems. In the following, we will discuss these directions with respect to adaptivity and learning.
8.2 Future Directions of Development

On the face of it, three of the developments indicated by Fig. 18 seem to be straight forward extensions of contemporary systems: Invariance against illumination, recognition of partial occlusion and a general rejection class. The latter denotes the ability to decide whether an object belongs to the domain or not. However, as pointed out in Sect. 3.1, lighting and partial occlusion mean an enormous increase of the “visual complexity” of objects, or in other words, of the manifold an object covers in pixel space.

It would mean an “abuse” of learning mechanisms if the additional visual degrees of freedom were just covered by enlarging the training sets. With larger computers and robotized image acquisition the current methods of memorizing as many views as possible could be stretched to some extent, but the complexity of the real world would be inaccessible. Therefore, new concepts have to be found which facilitate a separation of lighting- or occlusion-type degrees of freedom in a similar way like the what-where problem can be (partly) separated. Once such a separation is possible, learning the single sub-problems becomes feasible.

Development of mechanisms that allow the rejection of objects or, more general, of visual stimuli that are not known to the system is often misjudged to be a minor point. In fact, it is seldom realized to be a problem at all. However, most current adaptive vision systems suffer from the inability to refuse the processing or classification of images for which they are not prepared. This is due to the fact that learning is feasible only with positive examples, i.e. samples of the visual input that is to be processed later. By contrast, it is impossible to feed negative examples (that should be rejected) as this would mean “the whole world of visual stimuli except for a small domain”. In other words, the – vanishingly small – manifold within the pixel space that corresponds to the
domain a recognition system is built for cannot be defined clearly enough to exclude
the rest. It must be doubted whether this problem can be solved on the low level at
all – the human concept of “unknown” is obviously quite different from a low level
separation of pixel space into known and unknown regions. Instead, when confronted
with an unknown object, a human is able to perceive a lot of features like size, shape,
color, texture or even material. The classification “unknown” is the result of the lack of
a high level concept what or for what purpose the object might be.

This leads to another important direction of development: The incorporation of high
level or world knowledge into the recognition process. Vision is clearly not purely data
-driven but involves e.g. the goal oriented verification of expectations, hence, a mere
bottom up processing architecture is not appropriate. First attempts are made to join
adaptive systems for the low level processing with semantic networks as models for
a certain domain [1]. Up to now, the structure of semantic nets is mostly determined
by the experience of the human designer. It remains to be seen if concepts of learning
can be extended to this level. Especially interesting is the question for the “borderline”
between holistic and symbolic processing, or in other words, the question up to which
level of abstraction learning systems as outlined in this contribution can be applied and
where symbolic processing must set in.

The last (but not least) important line of development addressed here is image se-
quence processing. Currently, most CV systems rely on the evaluation of single im-
ages. Thus, recognition is restricted to static scenes. To capture the world of motion,
sequences have to be evaluated. However, motion of objects in front of the camera
is only part of the story as humans also use ego-motion to actively explore a scene.
Adaptive methods like the local PCA outlined in Sect. 5 can be extended to image se-
quences when the computational power becomes available, though they will meet the
same problems like in the static case. Apart from such “classification like” solutions,
however, new methods are required for the active exploration of scenes.

As a conclusion, it can be stated that adaptivity is surely a key feature future vi-
sion systems must possess as the world’s complexity cannot be captured by explicit
knowledge base design. However, there remain still many fundamental problems to be
solved like the separation problem. Though the increasing computational power of the
rapidly developing hardware is tempting, these problems should not be ignored since
real solutions cannot be replaced by the development of brute force methods.

Acknowledgement

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) in research
project A1 “Hybrid Knowledge Representation” within SFB 360, Bielefeld.

We would like to thank our former group members Robert Rae, Jan Jockusch, Nils
Jungclaus and Christoph Dücker for their contributions in the construction of the current
demonstration setup.

References

1. E. Braun, G. Heidemann, H. Ritter, and G. Sagerer. A multi-directional multiple path recog-
nition scheme for complex objects. Pattern Recognition Letters, special issue on Pattern