Dynamic Tactile Sensing for Object Identification

Gunther Heidemann and Matthias Schöpfer
Neuroinformatics Group
Faculty of Technology
Bielefeld University, Germany
Email: gheidema@techfak.uni-bielefeld.de

***Abstract*— We propose a neural architecture for the recognition of objects by haptics. We demonstrate its performance for a set of household objects and toys using a low cost 2D pressure sensor of coarse resolution, which is moved by a robot arm guided by contact points. The approach transfers the well known view-based method from computer vision to the domain of tactile sensing. However, in contrast to computer vision, not static frames but entire time series of 2D pressure profiles are evaluated.**

I. Introduction

The human skill in handling objects relies mostly on haptics. However, in robotics approaches for haptics-based control are still rare. This is partly due to the lack of sensors: Though high quality sensors exist, e.g. based on piezo-resistive [1] or piezo-electrical [2] pressure sensitive foils, sensors sufficiently small and flexible to be mounted on an artificial hand are hardly available. Another important problem is that pattern recognition from the rather coarse resolution of pressure sensors is still difficult. Significant progress in this field has been made with the introduction of *dynamic tactile sensing* [3], [4], which allows in a natural way the gathering of much more information than static sensing. However, dynamic sensing has so far mostly been applied either to reconstruct shape as a such [5], [6] and fitting object models to these shapes [7], or to explore surface features [8].

However, humans do not recognize objects by explicit shape reconstruction and model matching, but rather in a holistic manner. An attempt to avoid geometric modelling and to deal with uncertainty using fuzzy sets and neural nets was made in [9] within the context of contact formation. For an overview on tactile sensing see [10].

In this paper, we show that the *view based* approach of visual object recognition can be successfully transferred to tactile sensing of object identity. For view based recognition, sample views of objects are collected and used to build an implicit representation, instead of explicitly modelling object geometry. The idea is to memorize low level signal properties rather than high level models. In this field, *eigenspace representations* have become popular, e.g. for object- and face recognition [11], [12], and robot vision [13]. For the present work, local principal component analysis (local PCA) [14], [15] is applied for a holistic object representation.

The view based approach has two major benefits which make it a useful technique also for tactile processing: (*i*) The difficult step from the signal to a memorized model is unnecessary, i.e., there is no explicit shape reconstruction. (*ii*) The neural recognition system applied here can be trained from examples, including feature extraction.

We use a low cost 2D sensor array of only coarse resolution to explore and recognize natural objects by their shape. In the following section, the hardware setup for data acquisition will be outlined. Section III describes the neural processing architecture, section IV the results for a set of household and toy objects. The final section V discusses benefits and problems of the approach and the consequences for future research.

II. Experimental setup

A. Hardware

Fig. 1 shows the used sensor, which has the shape of a square matrix (Type DSA100-256, Weiss Robotics). It has a resolution of 16×16 pressure sensitive points with a distance of 6 mm. Pressure on a sensor point is measured by the electric current which goes from the center (small round circle) through a pressure sensitive foil (black material). The resistance of the foil decreases with increasing pressure.

Each sensor point can be sampled at 25 Hz with 12 bit pressure resolution. For the following, however, a reduced frame rate of 10 Hz and 8 bit for pressure are used to keep the amount of data manageable. The sensor array is mounted to the

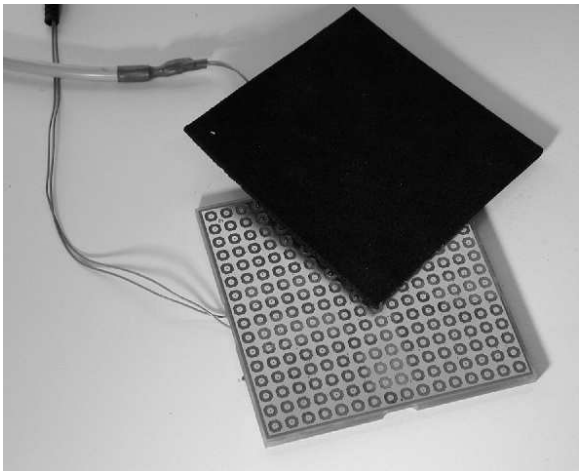

Fig. 1. DSA100-256 pressure sensor. The pressure sensitive black foil is partly removed to show the 16×16 grid of measuring points.

end-effector of a PUMA-200 industrial robot arm, controlled under RCCL [16].

B. Data acquisition

To gather tactile data from an object, it is positioned on an elastic pad (Fig. 2). Since the pressure imposed by the end-effector is low, fixation of the object is not necessary. The robot arm is moved in three phases to gather tactile data:

1) Approach: The robot lowers the sensor array towards the object in a horizontal position (first image of Fig. 2). When the object is touched and a certain pressure detected, the approach stops (position 1 in Fig. 3).
2) Tilt: The robot rotates the sensor to the right (upper right in Fig. 2), the rotation axis goes through the rightmost contact point. As soon as a new contact point (to the right) is detected, it becomes the new center of rotation (position 2 in Fig. 3). The movement stops when a maximum tilt angle is reached or the sensor has contact to the elastic pad. The latter case is assumed when the border points of the sensor detect pressure. This is a reasonable working solution as long as the object size is sufficiently small.
3) Move around: Steadily keeping contact, the robot arm now moves the sensor around the object, i.e. first forward while reducing the tilt to the right (lower left of Fig. 2), then to the left (lower right of Fig. 2), and so on.

Frames are recorded beginning with the first contact. The length of the sample recording is fixed to 16 seconds, so in total $N_F = 160$ frames are sampled. If the actual time needed to go around the object is less, the last frames remain blank.

Fig. 4 shows examples of the sensor profiles for two different objects, the frames are arranged in temporal order (from left to right and top to bottom). For each of the objects, only 60 out of 160 frames are shown for simplicity (every second frame is omitted plus the last 40 frames). Sensor points detecting high pressure are bright. The first few frames correspond to the touchdown onto the object, as visible best for the second object (a cup), for which the border is clearly visible before the tilt and rotation phases begin.

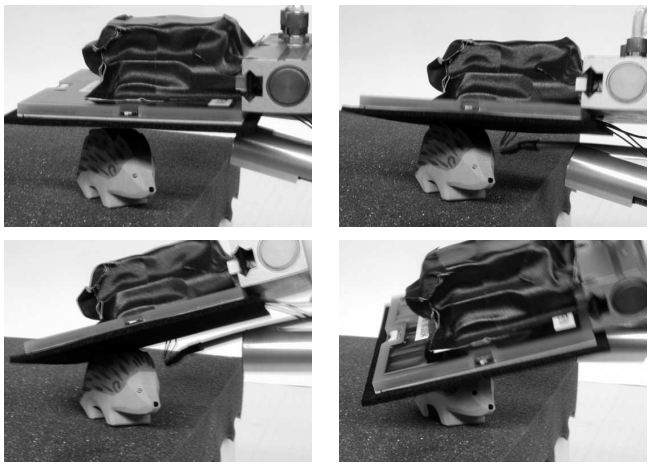

Fig. 2. Procedure to take a sequence of pressure profiles. The sensor is mounted to a PUMA-200 robot arm and lowered to the object until contact is sensed (upper left). Then the sensor records a sequence of 160 samples while being moved around the object, keeping contact steadily. The camera also visible at the end effector is not used in the current experiments.

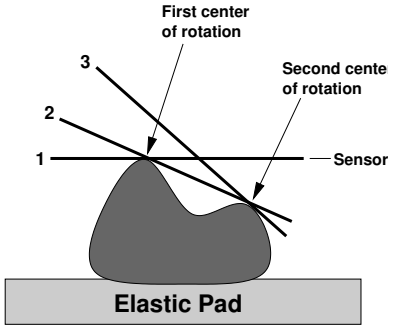

Fig. 3. 2D illustration of tilting the sensor: From the horizontal position of first contact (1), the sensor is tilted to the right until a new contact point is detected (2), then it is rotated around the new point (3).

The sparse pressure profiles of Fig. 4 illustrate that the information provided in a single frame $F(t)$ is poor: Due to the coarse spatial resolution, the objects leave only quite unspecific marks, which are in addition subject to noise. Here, $F(t) \in \mathbb{R}^{16\times 16}$ denotes the 16×16 matrix of pressure values at time t.

However, the complete trajectory over all frames is unique for each object. Therefore, to gather more information and achieve stability by redundancy, the sampled pressure values $F(t)$ are combined to form a single vector $\vec{x} \in \mathbb{R}^d$ characterizing the object. Its dimension is $d = N_F \cdot 16^2 = 40960$ since there are 16×16 sensor points. In the following, it will be described how the object can be classified from $\vec{x}$ using a neural approach.

III. Tactile signal processing

To classify objects from their sequence of pressure profiles $\vec{x} \in \mathbb{R}^d$, a neural architecture is applied that allows (*i*) to deal with high input dimensionality d and (*ii*) to form features automatically. By this means the difficult step of "designing" a feature extraction based on heuristics can be omitted. Instead, a three-stage processing architecture is applied (Fig. 5), which basically consists of a feature extraction based on local principal component analysis (local PCA) [14], [15] with a subsequent feature classification by several neural networks. This system has been successfully applied so far for several computer vision tasks [17], [13].

The neural system as a whole performs a mapping $\vec{x} \to \vec{y}, \vec{x} \in \mathbb{R}^d, \vec{y} \in \mathbb{R}^{N_C}$, where $\vec{y}$ is the output vector. The dimension of $\vec{y}$ is here set to the number of different object classes N_C. So the discrete valued result (the class number) is represented by an N_C-dimensional, continuous valued vector to avoid an artificial introduction of "neighbourhoods" within the class system. Training is carried out using samples $(\vec{x}_i^{Tr}, \vec{y}_i^{Tr})$, where the target output vectors are a binary encoding of the correct class number $c = 1 \ldots N_C$: $(\vec{y}_i^{Tr})_j = \delta_{jc}, j = 1 \ldots N_C$. Once the training is complete, the system classifies an unknown input $\vec{x}$

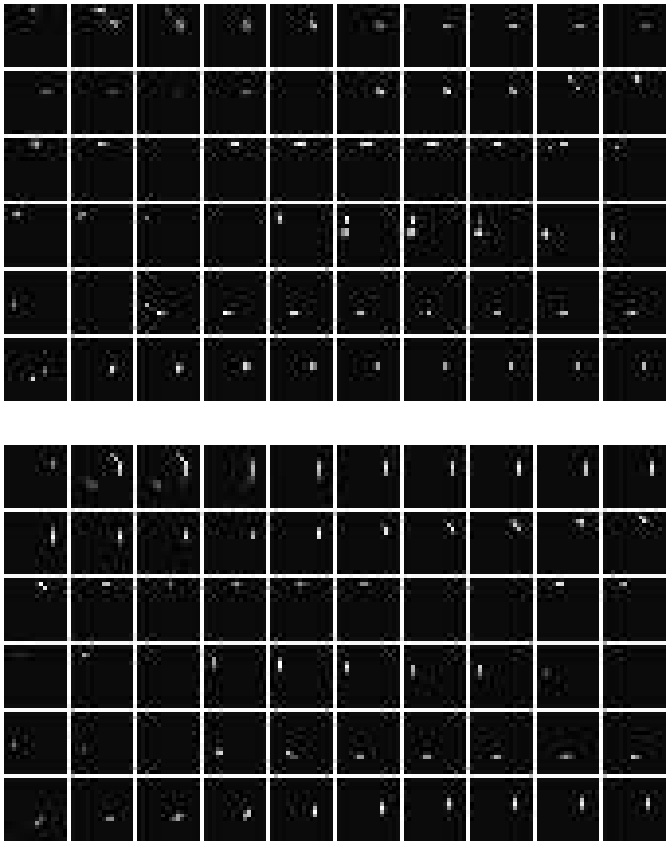

Fig. 4. Sample sequences of sensor profiles for two different objects. Each of the small squares shows the pressure on the 16×16 sensor at a certain time while the sensor is moved around the object (bright = high pressure). Only 60 out of 160 frames are shown for simplicity. The first few frames show the first contact with the object when the sensor is still in a horizontal position. Evidently, the information conveyed by a single frame is rather small and unreliable, but the entire time sequence gives a highly object specific signature.

as the class c' for which the corresponding output component has the maximal value: $c' = \arg\max_j (\vec{y}(\vec{x}))_j$.

In the following, both training and application of the three processing stages are described.

A. Stage 1: Data partitioning

The first stage partitions the input space $\mathbb{R}^d$ using vector quantization (VQ) (see e.g. [18]). A number N_{VQ} of reference vectors $\vec{r}_i \in \mathbb{R}^d, i = 1 \ldots N_{VQ}$, is positioned in input space by *Activity Equalization* VQ (AEV) [19]. A detailed description of this algorithm is beyond the scope of this paper, in short, AEV is a method particularly designed for high dimensionality and a sparsely "filled" input space. AEV avoids the problem of codeword under-utilization [20], i.e. the problem that in a high dimensional space many reference vectors are likely to remain outliers. This is achieved by an explicit count of codeword access frequencies.

For classification, for an input $\vec{x}$ the best match reference vector $\vec{r}_{n^*(\vec{x})}$ is searched for which minimizes $\|\vec{x} - \vec{r}_i\|, i = 1 \ldots N_{VQ}$. The result of the first level is the number $n^*(\vec{x})$.

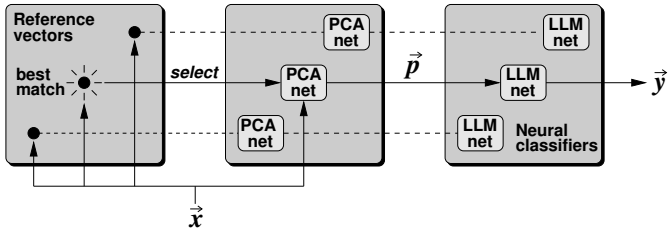

Fig. 5. Architecture of the neural feature extraction and classification: For the input $\vec{x}$ which consists of the pressure values of the entire sequence, first the best match reference vector is found (left). This selects a PCA-net specialized to the data in the corresponding Voronoi tessellation cell (middle). Projection of $\vec{x}$ onto the PCs yields the feature vector $\vec{p}$, which is classified by an attached LLM-net to obtain the output vector $\vec{y}$ (right). The maximum component of $\vec{y}$ denotes the "winner" class.

B. Stage 2: Local PCA

The second level performs PCA for each of the reference vectors for the training samples $\vec{x}$ which fall into the corresponding Voronoi tessellation cell. For calculation of the principal components (PCs), to each reference vector $\vec{r}_i$ a single layer feed forward neural network is attached. The N_{PC} PCs with the largest eigenvalues are computed successively using the training rule proposed by Sanger [21]. This iterative procedure is necessary because a direct computation of the eigenvectors of the input autocorrelation matrix is impossible due to the high data dimensionality $d = 40960$. After training, the weight vectors $\vec{w}_{ij} \in \mathbb{R}^d$, $i = 1 \ldots N_{VQ}$, $j = 1 \ldots N_{PC}$ of the neural nets represent the local PCs.

For application, an input $\vec{x}$ is projected onto the weight vectors $\vec{w}_{n^*(\vec{x}),j}$ of net number $n^*(\vec{x})$: $\vec{x} \rightarrow \vec{p}_{n^*(\vec{x})}(\vec{x})$. The projection $\vec{p} \in \mathbb{R}^{N_{PC}}$ can be regarded as the feature vector of input $\vec{x}$.

C. Stage 3: Feature classification

While stage 1 and 2 are trained unsupervised to form features (the local PCs $\vec{w}_{ij}$), the third stage classifies the feature vector $\vec{p}_{n^*(\vec{x})}(\vec{x})$ to obtain $\vec{y} \in \mathbb{R}^{N_C}$ and thus requires supervised training. To each of the PCA-nets an individual neural classifier is attached as a "local expert". These neural nets are of the *Local Linear Map* – type (LLM), for details see [22]. In short, the LLM is related to the self-organising map [23] and the GRBF-approach [24]. It performs first VQ using N_{LLM} reference vectors (or "nodes") in its input space, which has here dimensionality N_{PC}. The VQ is again carried out using the AEV-algorithm [19]. To each of the nodes a trainable, locally valid linear mapping is attached. In combination, these N_{LLM} linear mappings approximate the nonlinear training function. The only parameter of the LLM is its number of linear mappings, N_{LLM}.

To obtain the final output for $\vec{x}$, after finding the best match reference vector number $n^*(\vec{x})$ and the projection $\vec{p}_{n^*(\vec{x})}(\vec{x})$, the LLM number $n^*(\vec{x})$ maps $\vec{p}_{n^*(\vec{x})}(\vec{x}) \rightarrow \vec{y}$.

D. Discussion of the classification architecture

Local PCA is used for feature extraction because it allows to obtain efficient features in a purely unsupervised way. While

normal PCA [25] is the optimal linear method to capture the variance of a data distribution, local PCA is a nonlinear extension which allows a better approximation of nonlinear and in particular clustered distributions [14], [15]. It has been successfully applied in a number of applications for dimension reduction [26] and pattern recognition [27], [28], [29].

Among the various methods to calculate a local PCA representation, the algorithm outlined above is a simple one, since it decouples clustering and PCA. Other approaches improve this procedure by repeated iterations of clustering and PCA to minimize the least square reconstruction error [26]. Even further advanced is the probabilistic method of [30]. The decoupled approach is used here because computational efficiency of the training is more important than reconstruction accuracy, the latter can be achieved also by an increase of N_{VQ}. Decoupled adaptation is computationally cheaper, because the "product space" of possible VQ- and PCA-solutions does not have to be searched.

The separation into isolated VQ and PCA has the additional advantage that no extensive parameter tuning is required because the algorithms for VQ and PCA are well known and can be treated as "black boxes" without adaptation of any training parameters. The remaining parameters are N_{VQ}, N_{PC} and N_{LLM}, which are decisive for the achievable system performance and have to be customized to the complexity of the problem. Too small values result in poor recognition rates since the networks do not have sufficient capacity to memorize the variety of sensor profiles. However, the process of finding suitable values for N_{VQ}, N_{PC} and N_{LLM} is easy, because in previous work [31] it could be shown that classification performance rises smoothly up to saturation with all three parameters. Overfitting phenomena were not observed, a too large classifier loses only in classification speed but not in accuracy. Thus, the effort to find suitable values can be kept manageable.

IV. Results

To test the approach, data were sampled from seven objects (toy car, pencil sharpener, cup, toy hedgehog, marble, candle holder and a piece of foam plastic). Each object was scanned 20 times under slightly different conditions, i.e. minor translations and rotations of the object. To achieve a common translational alignment, the 2D center of gravity in the sensor plane was computed over each complete sequence $F(0) \ldots F(159)$ (not over single frames). So a common 2D shift of all frames $F(t) \rightarrow F'(t)$ of a sequence could be performed to move the center of gravity to the middle of the sensor plane. Then the input vector $\vec{x}$ was formed from the shifted pressure values $F'(t)$.

For the neural feature extraction and classifier, a choice of $N_{VQ} = 6$ reference vectors, $N_{PC} = 8$ local PCs and $N_{LLM} = 20$ nodes proved to provide sufficient capacity to memorize the sequences. The parameters were found by starting with small values and increasing them in turn as long as classification performance could be improved. The procedure stops when no further performance increase can be achieved.

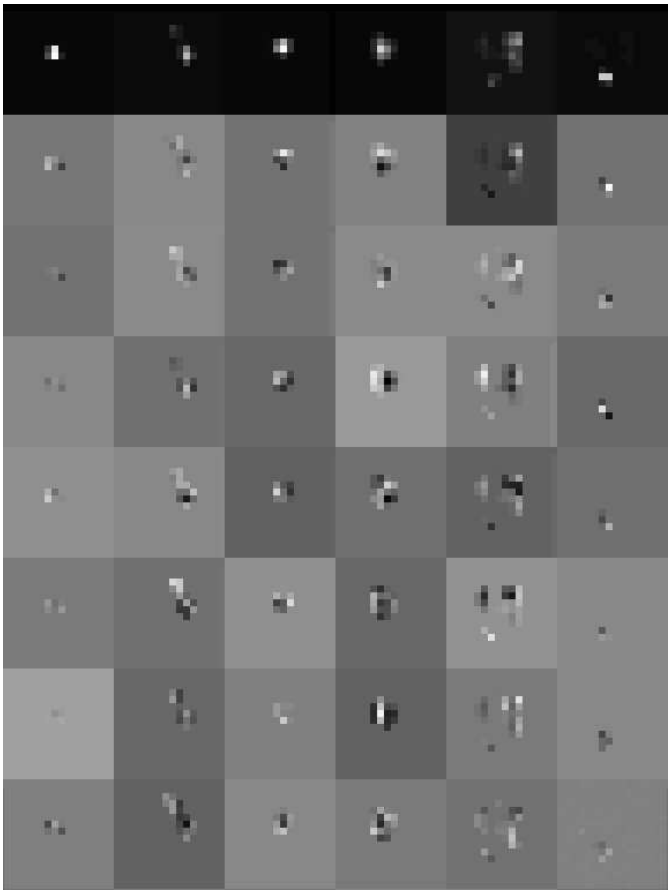

Fig. 6. Local PCA – representation for a fixed time slice t. The first row shows the reference vectors, below each are the corresponding local PCs in a column. Most of the reference vectors are not specialized to a single object, instead, the information of the training set is "spread out" over the whole representation. As a time series is represented, the complete reference vectors and eigenvectors would show a moving pressure profile each.

Fig. 6 shows a "slice" of the obtained reference vectors $\vec{r}_i$ and the attached local PCs at a certain time t. It has to be kept in mind that each reference vector as well as each PC is not a single prototypic pressure profile but an entire time sequence of profiles. The first row in Fig. 6 shows the reference vectors, below each are the corresponding PCs in a column. A close inspection of the $\vec{r}_i$ and their PCs reveals that most of them are not specialized to a particular object. Instead, the information is spread out over the representation.

For evaluation, the set of 20 sequences of each object was divided into a training set of 16 sequences and a test set of 4 sequences. So for seven objects the entire training set comprises 112 sequences, the test set 28. The system was then trained to the classification of the seven objects using the training set, and evaluated on the test set. The particular partitioning into training- and test set can slightly influence the measured performance: If, loosely speaking, the test set lies completely "within" the variety of sensor profiles covered by the training set, better classifications are achieved than in the case that the test set comprises sequences which are untypical (compared to the training set). To reduce

the influence of the random partitioning, computations were repeated ten times with different training / testing partitioning and results averaged.

The achieved average rate of correct classifications is 81%, where results of the single runs range between 74% and 83%, chance level is $\approx 14\%$ (since "correct" means the system selects the right one of seven objects). So far, the system has no rejection class, i.e. there is no way to indicate an unknown object. The system is restricted to the trained object set, if an unknown one is presented, it will inevitably be sorted into one of the trained categories. This problem is familiar from other pattern recognition tasks: Adaptive systems can be trained to a certain domain, but not to ignore the "rest of the world". The difficulty stems from the complexity of the volume which the sensor data $\vec{x} \in \mathbb{R}^d$ cover within $\mathbb{R}^d$. Establishing a general rejection class would require to define a hull around this volume, which separates signals of the domain from unknown ones. Such a border, however, is difficult to establish only from positive samples.

An important source of information for the system are located peaks of pressure. Since these are detected by few sensor points, translations or rotations of an object change appearance on the signal level drastically and are the main source of errors — however, mostly when the test sample shows an object translation not covered by training samples. Therefore, errors can be reduced by enlarging the training set accordingly.

V. Discussion and future prospects

We have presented a system for object recognition based on haptics as a step towards a "view based", holistic representation of haptic sensor data. The main benefit of the approach is that a suitable and robust data representation can be achieved by training from examples, without the need to design a geometrical object model or to select features based on heuristics. The system works even with a low cost sensor with coarse spatial resolution.

Still, the method is confined to the identification of isolated objects presented in a restricted set of poses. However, several directions of future development appear promising:

1) Temporal alignment: So far it is provided that the object is positioned in or close to a pose covered by the training set. If it is rotated to an un-trained pose, it will not be recognized though the sampled sequence in principle holds the information. The problem is that the novel sequence cannot be temporarily aligned with the training sequences. This problem resembles the "what-where" problem known from computer vision, though now in the time domain. Possibly, the method of *interest point* detection can be transferred from vision to haptics. If outstanding ("salient") points can be detected in the spatio-temporal representation, they could be used to achieve alignment without the need of an "exhaustive matching" to find a specific (start-) frame.

2) Sliding: In the present scenario, the sensor is rolled around the object such that a point of the object always appears at the same sensor location, which yields a poor representation due to the coarse resolution. Using a more natural manner of sensing where the sensor is sliding over the object surface, an improved *effective resolution* could be realized. Again, a holistic representation appears to be feasible, so deconvolution techniques as in [3] would not be necessary.

3) Active exploration: The strategy for object exploration could be modified online based on partial results for mainly three different reasons:

- If the up side of an object does not allow a decision, it could be turned by a gripper to scan the bottom side.
- Still, only convex surface points can be exploited. Detected concavities, however, could be scanned by a smaller sensor.
- Noisy or incomplete data could be improved by repeated sampling.

All of the above points are aimed at improving object exploration by the sensor alone. The question arises whether results could be improved by exploiting geometric information, e.g. the position of the contact point in 3D space, which is easily available from the joint angles. Explicit geometric information would be mostly redundant, because the sequence of pressure profiles implicitly represents not only the objects surface properties in the sense of texture, but also part of the overall object geometry. So what can be gained from 3D information is mainly the object's height above the elastic pad (thickness). The use of 3D information, however, will become necessary for objects too large to be explored by the rolling sensor without interruption. For a piecewise sampling, computation of 3D contact points will be required.

Acknowledgment

The authors would like to thank Helge Ritter, head of the Neuroinformatics Group at Bielefeld University, for the support of their work. This work was funded by the Deutsche Forschungsgemeinschaft (DFG).

References

[1] *The force sensing resistor*, Interlink Electronics, Europe, Echternach, G. D. de Luxemburg, 1990.

[2] *Piezo Film Sensors Technical Manual*, AMP Incorporated, Valley Forge, PA 19482, 1993.

[3] R. D. Howe and M. R. Cutkosky, "Dynamic tactile sensing: Perception of fine surface features with stress rate sensing," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 2, 1993.

[4] G. Canepa, M. Campanella, and D. de Rossi, "Slip detection by a tactile neural network," in *Proc. ICIROS 94*, vol. 1, 1994, pp. 224–231.

[5] N. Chen, R. Rink, and H. Zhang, "Local Object Shape from Tactile Sensing," in *Proc. IEEE Int'l Conf. on Robotics and Automation 1996*, 1996.

[6] M. Moll and M. A. Erdmann, "Dynamic shape reconstruction using tactile sensors," in *ICRA-02*, 2002, pp. 1636–1641.

[7] P. K. Allen and P. Michelman, "Acquisition and interpretation of 3-d sensor data from touch," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 4, pp. 397–404, 1990.

[8] A. M. Okamura and M. R. Cutkosky, "Feature detection for haptic exploration with robotic fingers," *Int'l J. of Robotics Research*, vol. 20, no. 12, 2001.

[9] M. Skubic and R. A. Volz, "Identifying single ended contact formations from force sensor patterns," *IEEE Trans. on Robotics and Automation*, vol. 16, no. 5, pp. 597–603, 2000.

[10] M. H. Lee, "Tactile sensing: New directions, new challenges," *Int'l Journal of Robotics Research*, vol. 19, no. 7, 2000.

[11] H. Murase and S. K. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int'l J. of Computer Vision*, vol. 14, pp. 5–24, 1995.

[12] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.

[13] G. Heidemann and H. Ritter, "Learning to Recognise Objects and Situations to Control a Robot End-Effector," *KI (Artificial Intelligence), special issue on Vision, Learning, Robotics*, no. 2, pp. 24–29, 2003.

[14] N. Kambhatla and T. K. Leen, "Fast Non-Linear Dimension Reduction," in *Advances in Neural Information Processing Systems 1993*, J. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan Kaufmann Publishers, 1994, pp. 152–159.

[15] C. Bregler and S. M. Omohundro, "Surface Learning with Applications to Lipreading," in *Advances in Neural Information Processing Systems 1993*, J. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan-Kaufmann Publishers, 1994, pp. 43–50.

[16] J. Lloyd and M. Parker, "Real time control under unix for rccl," in *Robotics and Manufacturing ISRAM 90*, vol. 3, 1990, pp. 237–242.

[17] G. Heidemann and H. Ritter, "Visual Checking of Grasping Positions of a Three-Fingered Robot Hand," in *Proc. ICANN 2001*, G. Dorffner, H. Bischof, and K. Hornik, Eds. Springer-Verlag, 2001, pp. 891–898.

[18] J. Buhmann and H. Kühnel, "Vector quantization with complexity costs," *IEEE Trans. on Information Theory*, vol. 39, no. 4, pp. 1133–1145, 1993.

[19] G. Heidemann and H. Ritter, "Efficient Vector Quantization Using the WTA-rule with Activity Equalization," *Neural Processing Letters*, vol. 13, no. 1, pp. 17–30, 2001.

[20] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Sci.*, vol. 11, pp. 23–63, 1987.

[21] T. D. Sanger, "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, vol. 2, pp. 459–473, 1989.

[22] H. J. Ritter, T. M. Martinetz, and K. J. Schulten, *Neuronale Netze*. München: Addison-Wesley, 1992.

[23] T. Kohonen, *Self-Organizing Maps*. Springer Verlag, 1995.

[24] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. of the 1988 Connectionist Models Summer School*. San Mateo, CA: Morgan Kaufman Publishers, 1988, pp. 133–143.

[25] I. Jolliffe, *Principal Component Analysis*. New York: Springer Verlag, 1986.

[26] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1493–1516, 1997.

[27] T. Hastie, P. Simard, and E. Sackinger, "Learning Prototype Models For Tangent Distance," in *Advances in Neural Information Processing Systems 1994*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. MIT Press, 1995, pp. 999–1006.

[28] G. E. Hinton, P. Dayan, and M. Revow, "Modelling the Manifolds of Images of Handwritten Digits," *IEEE Trans. on Neural Networks*, vol. 8, no. 1, pp. 65–74, 1997.

[29] B. Moghaddam and A. Pentland, "Probabilistic Visual Learning for Object Representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696–710, 1997.

[30] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

[31] G. Heidemann and H. Ritter, "Combining Multiple Neural Nets for Visual Feature Selection and Classification," in *ICANN 99, Ninth Int'l Conf. on Artificial Neural Networks*. IEE, London, 1999, pp. 365–370.