Linköping University Post Print

Real-Time View-Based Pose Recognition and Interpolation for Tracking Initialization

Michael Felsberg and Johan Hedborg

N.B.: When citing this work, cite the original article.

Original Publication:

Michael Felsberg and Johan Hedborg, Real-Time View-Based Pose Recognition and Interpolation for Tracking Initialization, 2007, Journal of real-time image processing, (2), 2-3, 103-115.

<http://dx.doi.org/10.1007/s11554-007-0044-y>

Copyright: Springer Science Business Media

Postprint available at: Linköping University Electronic Press

<http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-39505>

Michael Felsberg · Johan Hedborg

Real-time view-based pose recognition and interpolation for tracking initialization

Received: date / Revised: date

Abstract In this paper we propose a new approach to real-time view-based pose recognition and interpolation. Pose recognition is particularly useful for identifying camera views in databases, video sequences, video streams, and live recordings. All of these applications require a fast pose recognition process, in many cases video real-time. It should further be possible to extend the database with new material, i.e., to update the recognition system online.

The method that we propose is based on P-channels, a special kind of information representation which combines advantages of histograms and local linear models. Our approach is motivated by its similarity to information representation in biological systems but its main advantage is its robustness against common distortions such as clutter and occlusion. The recognition algorithm consists of three steps:

1. low-level image features for color and local orientation are extracted in each point of the image
2. these features are encoded into P-channels by combining similar features within local image regions
3. the query P-channels are compared to a set of prototype P-channels in a database using a least-squares approach.

The algorithm is applied in two scene registration experiments with fish-eye camera data, one for pose interpolation from synthetic images and one for finding the nearest view in a set of real images. The method compares favorably to SIFT-based methods, in particular concerning interpolation.

The method can be used for initializing pose-tracking systems, either when starting the tracking or when the tracking has failed and the system needs to re-initialize. Due to its

real-time performance, the method can also be embedded directly into the tracking system, allowing a sensor fusion unit choosing dynamically between the frame-by-frame tracking and the pose recognition.

Keywords: pose recognition, pose interpolation, P-channels, real-time processing, view-based computer vision

1 Introduction

The proposed method has been developed in context of the MATRIS project¹ aimed at markerless real-time tracking of camera poses [40]. The MATRIS system applies a hybrid technique for estimating the pose dynamically by fusing position estimates of tracked image features² [38,37] and inertia measurements [24]. For this process to start or to restart in case of failure, an initial registration of the current view in a database is required to acquire a coarse pose estimate. The proposed method allows the registration to be achieved by pose recognition³.

Pose recognition is a special case of image recognition. Most image recognition approaches are based on either of the following two paradigms: Model-based recognition or view-based recognition. It is propounded in the literature that view-centered recognition has significant advantages compared to object-centered models (see e.g. [18], Sect. 5). Following this line of arguments, we focus on view-based representations. View-based representations can be *global*, i.e., the elements of the representation depend on the whole image, or they can be *local*, where the elements of the representation only depend on a sub-region of the image. Both kinds of representation can be either *dense*, i.e., all image pixels contribute to the representation, or *sparse*, i.e., only selected image pixels contribute.

The problem that we consider in this paper is the following: Interpolate the camera pose for a previously unseen

This work has been supported by the CENIT project CAIRIS (<http://www.cvl.isy.liu.se/Research/Object/CAIRIS>), EC Grants IST-2003-004176 COPSAL and IST-2002-002013 MATRIS. This paper does not represent the opinion of the European Community, and the European Community is not responsible for any use which may be made of its contents.

M. Felsberg · J. Hedborg
Computer Vision Laboratory
Linköping University, Sweden
Tel.: +46-13-282460
Fax: +46-13-138526
E-mail: mfe@isy.liu.se

¹ <http://www.ist-matris.org>

² By *features* we denote "a numerical property" [14].

³ By *recognition* we denote "identification" aka "the process of associating some observations with a particular instance [...] that is already known" [14].

view with a system which has seen the scene from different poses before. Interpolation of camera poses results in a non-linear mapping on the space of previously seen images. The difference to view based object recognition, see e.g. [12], is that object recognition aims at producing the most probable object class independent of the view angle, whereas in the present paper, we try to estimate the camera pose as accurate as possible. The different views might originate from a single physical scene ('object class') or from several ones.

The solution that we propose belongs to the category of dense, local methods, i.e., all image pixels contribute to the view representation, but each element has a local region of support. Algorithms that work with local image regions are popular, see e.g. [1,2], due to their "robustness to geometric distortion" [2]. The proposed method uses P-channels as a view representation, a relatively new technique [11] which is related to local histograms and is introduced in Sect. 2.3. The proposed recognition algorithm consists of three steps:

1. low-level image features for color and local orientation are extracted in each point of the image
2. these features are encoded into P-channels by combining similar features within local image regions
3. the query P-channels are compared to a set of prototype P-channels in a database using a least-squares approach.

The last step is related to comparing local density estimates of the low-level features, as a local density estimate can be obtained from a linear combination of P-channels [12]. It has been shown in [12] that using least-squares techniques works slightly better than the (regularized) Kullback-Leibler divergence [7] of the kernel density estimate.

The least-squares method for estimating the camera pose from P-channels is the main contribution of this paper, i.e., the paper focuses on the view representation and its matching. In order to build a system including a large database of stored view representations, the data has to be organized cleverly, e.g. in tree structures, for achieving fast access, but this is considered as a separate problem not treated in this paper. The interested reader is referred to prominent papers as, e.g., [30,34,33].

If pose recognition shall be applied for tracking re-initialization of video streams and live recordings, the method must work in video real-time, i.e., recognition in images with PAL resolution at 25 Hz, in order to avoid frames without pose estimates. For databases and video sequences, even higher *average* recognition speed might be desirable in order to analyze the data in sub-real-time. Real-time processing on up-to-date computer systems or on future wearable systems rules out current implementations of many powerful approaches for recognition, e.g., learning-based methods as [35, 36,29], image matching [6], group-theoretic global methods [13], and eigen-space projections [41]. The proposed method for pose recognition runs in real-time on ordinary personal computers.

As we cannot guaranty that all views required for successful recognition of all poses are available prior to the tracking, it must be possible to add new views, e.g. from

successful tracking, dynamically to the learning set. Requiring that *learning has to be done on the fly* excludes batch-learning methods as, e.g., [22,32]. Our proposed method allows to add new view representations at any time.

For a method to be applicable in practice, it needs to be robust and stable. By *robustness* we refer to the property of a system or method to handle quantitative and qualitative deviations from the assumed model, e.g., occlusion or specular highlights. If an input to the system lies outside the modeled space, it should lead to some standard output, e.g. suppression of an input or re-initialization of a tracking system in case of lost tracking. To differentiate from general robustness, *statistical robustness* denotes the limited quantitative influence of input outliers to the output of an estimation process. We consider statistical robustness as a special case of general robustness, starting from the assumption of a linear model for the estimator which is then modified to achieve saturation of the estimate error for outliers.

Robustness is often mixed up with the term *stability*, by which we denote bounded output error for a bounded input error (bibo stability), or equivalently a finite operator norm. The latter is trivially fulfilled for finite linear operators. The proposed method is robust against intensity changes, i.e., we do not model intensity invariance, but the system's output is not influenced by changes caused by opening / closing blinds, changing the aperture or shutter time. The pose estimates are robust against occlusion and specular highlights, which is shown in the experimental section. Finally, the method behaves stable under geometric distortions, i.e., introducing small geometric distortions leads to small additional errors in the pose estimate.

In the experiments section of the paper it is shown that P-channel representations perform well compared to state-of-the-art descriptors. Many recently published visual recognition methods make use of SIFT features [30] in one or the other way. The SIFT feature is computed by extracting the SIFT (keypoint) descriptor at SIFT keypoints with appropriate orientation and scale. P-channels can likewise be computed in a local frame [27] and therefore it makes sense to compare the SIFT *descriptor* (cf. Sect. 6 of [30]) to the P-channel representation. The SIFT descriptor is similar to a linear-spline channel representation (cf. [10]) of the gradient orientation weighted by the saturated gradient magnitude. The main differences of the here applied P-channels to SIFT descriptors are: disregard for the gradient magnitude, no spatial weighting, inclusion of hue and saturation, and finally using piecewise polynomials of order zero and one instead of linear interpolation. SIFT descriptors of the same dimensionality are significantly slower to compute than P-channels. Comparing the descriptive power of both image descriptors shows that P-channels perform equally well for nearest-neighbor matching and perform superior for the least-squares interpolation of poses, i.e., non-linear mappings on the image space.

2 Density Estimation by Channel Representations

In this section, we recapitulate those definitions and properties of channel representations which are relevant for the application of density estimation. Within the field of density estimation, we only consider the simplest case where we estimate the pdf from samples ξ_j drawn from the distribution $p(\xi)$ using the kernel B as

$$\hat{p}(\xi) = \frac{1}{J} \sum_{j=1}^J B(\xi - \xi_j) . \quad (1)$$

It is well known that the expectation value of the estimate is given as the convolution of the true density with the kernel function [3]:

$$E\{\hat{p}\} = B * p . \quad (2)$$

For further details and properties the interested reader is referred to standard text books as e.g. [3,4]. In this section we introduce a method for non-parametric density estimation, which is significantly faster than standard kernel density estimators.

2.1 Channel Representations

The approach for density estimation that we will apply in what follows, is based on the channel representation [19, 39]. The channel representation is related to observations from biological systems, where receptive fields are spatially located and are only sensitive to certain value ranges, e.g. colors or orientations. The spatial support and value sensitivities are overlapping and establish a smooth transition between neighbored fields. Each receptor thus counts incoming stimuli weighted by the spatial distance to its center and weighted by the distance in value domain, cf. [16], Chapt. 5.

In the channel representation, compact, smooth, non-negative basis functions B , the *channels*, are placed regularly⁴ in spatial and value domain (see Fig. 1 (b) for channels in the value domain). A single measurement is defined by its position and value, i.e., it is considered as a point in the joint space of position and value, and thus activates all channels that have non-zero support at this point. This process is called *channel encoding*. Hence, a single measurement is represented by a tuple of numbers, the *channel vector*. Each component of this vector, i.e., each *channel value*, corresponds to the activation of the respective channel.

This is a highly redundant representation unless several measurements drawn from the same distribution (see Fig. 1 (a)) have to be represented. In that case the channel vectors of all measurements are averaged, thus leading to a fuzzy voting scheme [8] for the different channels (see Fig. 1 (c)). Picking local maxima of the channel vector and applying

some local interpolation scheme allows to extract cluster centers of the data. This process is called *channel decoding*, but it is not considered in more detail here. For further details on the issue of channel decoding see [15].

In this paper we make direct use of the channel representation in terms of a kernel density estimate. This relation is formalized below. In order to compensate for the convolution with the kernel function, one can imply constraints, e.g., on entropy maximization [26] (see Fig. 1 (d)). The most trivial case of (non-smooth) channels representations are histograms, but they have the drawback of losing sub-bin accuracy compared to smooth channel representations.

Channel vectors (denoted as boldface letters) do not form a vector space as no additive inverse elements exist. This results from the non-negativity of channels. The value zero (in each component) has a special meaning: *no information*. Most channel value are equal to zero in practice and need not be stored in the memory if a sparse data type is used. Note that channel representations are not just a way to re-represent data, but they allow advanced non-linear processing by means of linear operators, see Sect. 2.2.

Formally, the channel vector is obtained from a finite set of *channel projection operators* $\{B_n\}_n$, where n is the channel index. For a 1D measurement f_j , each operator B_n is applied to f_j resulting in the channel value p_{nj} :

$$p_{nj} = B_n(f_j) \quad n = 1, \dots, N . \quad (3)$$

The measurement f_j is thus mapped to the channel vector $\mathbf{p}_j = (p_{1j}, \dots, p_{Nj})^T$. The average over j , i.e., over all instances, results in the channel representation of the set of measurements:

$$\mathbf{P} = \frac{1}{J} \sum_{j=1}^J \mathbf{p}_j . \quad (4)$$

If the considered measurements are vector-valued, i.e., we would like to encode K -D measurement vectors $\mathbf{f}_j = (f_{j1}, \dots, f_{jK})^T$, we first apply projection operators to each of the K components according to (3), resulting in channel vectors $\mathbf{p}_{jk} = (p_{1jk}, \dots, p_{Njk})^T$. Before summation over j , i.e., all instances, we compute the outer product (tensor product) of the respective channel vectors $\mathbf{p}_{jk}$:

$$\mathbf{P}_j = \bigotimes_{k=1}^K \mathbf{p}_{jk} \quad \text{i.e.} \quad P_{i_1, i_2, \dots, i_K, j} = p_{i_1, j, 1} \cdot p_{i_2, j, 2} \cdots p_{i_K, j, K} . \quad (5)$$

The final K -D channel representation is given by the average over j :

$$\mathbf{P} = \frac{1}{J} \sum_{j=1}^J \mathbf{P}_j . \quad (6)$$

As pointed out above, the channels are smooth and overlapping, hence, each measurement activates several channels. If we assume that each measurement component results in a non-zero values in (3), each $\mathbf{P}_j$ contains obviously a^K non-zero values. Thus, the previously described method is only feasible to apply for small K .

⁴ There is no formal requirement for having regularly spaced channels or spatially invariant basis functions. For the purpose of density estimation, these restrictions make however sense and simplify the understanding.

Fig. 1 Example for channel representation of 1D data. (a) noisy 1D signal. (b) quadratic spline channels. (c) mean values of channel representation of 1D signal. (d) reconstructed probability density function [26].

2.2 Relation to Density Estimation

The projection operators can be of various form, e.g., $\cos^2$ functions, B-splines, or Gaussian functions [15]. The channel representation has previously been applied for associative learning [25] or statistically robust smoothing [10].

In context of statistically robust smoothing it has been shown that averaging B-spline channel vectors (i.e., channels with B-splines as basis functions B) of samples from a stochastic variable ξ results in a sampled⁵ kernel density estimate of the underlying distribution $p(\xi)$, see Fig. 1 (d):

$$E\{\mathbf{p}\} = E\{[B_n(\xi)]\} = (B * p)(n) . \quad (7)$$

The global maximum of p is the most probable value for ξ and for locally symmetric distributions, it is equivalent to the maximum of $B * p$. The latter can be approximately extracted from the channel vector $\mathbf{p}$ using an implicit B-spline interpolation [10] with prefiltering according to [42].

⁵ Sampled in a signal processing sense, not in statistical sense.

2.3 P-Channels

As pointed out at the end of Sect. 2.1, channel representations are not feasible to compute for higher-dimensional measurements. The reason for this is the overlapping of channels, which can only be avoided if basis function with disjoint support are used. This leads however to ordinary rectangular bins which do not allow for accurate decoding. One way out of this dilemma is to use non-overlapping channels with an algebraic structure richer than positive real numbers. An obvious first choice are complex numbers, where the measurement is represented by the argument [20]. The drawback of this method is the non-linearity when averaging complex channels.

This is different for the projective line $\mathbb{P}^1$ respectively projective space $\mathbb{P}^K$, i.e., homogeneous coordinates: the embedding remains linear. Representing measurements as homogeneous channels leads to the approach of P-channels [11]. The P-channels are obtained by using two respectively $K + 1$ basis functions for each channel: a rectangular basis function, i.e., an ordinary histogram bin, and one respectively

Fig. 2 P-channel representation for the example from Fig. 1. (a) P-channels at original scaling. (b) channels at integer positions. (c) mean values of P-channel value vectors.

K components which store the offset from the channel center. As a consequence, the components of the channel vector become vector-valued (boldface letters) and are denoted as *channel value vectors*. The channel vector becomes an array of channel value vectors, in other words: a matrix (boldface capital).

A set of 1D measurements $\{f_j\}_j$ is encoded into P-channels as follows. Without loss of generality the channels are located at integer positions (see Fig. 2 (b)). The values f_j are accounted respectively to the channels with the center $[f_j]$, where $[f_j]$ is the closest integer to f_j :

$$\mathbf{p}_i = \frac{1}{J} \sum_{j=1, [f_j]=i}^J \begin{pmatrix} f_j - [f_j] \\ 1 \end{pmatrix}. \quad (8)$$

Hence, the second component of the channel value vector is an ordinary histogram estimate of the average number of occurrences within the channel bounds over the whole set of measurements. The first component of the channel value vector contains the average linear offset from the channel center over the whole set.

Let $\mathbf{f}_j$ be a K -dimensional feature vector. The P-channel representation of a set of vectors $\{\mathbf{f}_j\}_j$ is defined as

$$\mathbf{P}_i = \frac{1}{J} \sum_{j=1, [\mathbf{f}_j]=\mathbf{i}}^J \begin{pmatrix} \mathbf{f}_j - [\mathbf{f}_j] \\ 1 \end{pmatrix}, \quad (9)$$

where $\mathbf{i}$ is a multi-index, i.e., a vector of indices $(i_1, \dots, i_K)^T$, and $[\mathbf{f}]$ is defined component-wise as $([f_1], \dots, [f_K])^T$. The algorithmic equivalent to (9) is summarized in Alg. 1. An example for the P-channel representation is given in Fig. 2 (c).

The main advantage of P-channels opposed to overlapping channels is the linear increase of time and space complexity with growing number of measurement dimensions K . Whereas each input sample affects a^K channels if the

Algorithm 1 P-channel encoding algorithm.

Require: $\mathbf{f}$ is scaled correctly for integer channel centers

```

1: for all  $j$  do
2:   for  $k = 1$  to  $K$  do
3:      $i_k \leftarrow \text{round}(f_{j,k})$ 
4:      $o_k \leftarrow f_{j,k} - i_k$ 
5:   end for
6:    $l \leftarrow \text{linear\_index}(\mathbf{i})$ 
7:   for  $k = 1$  to  $K$  do
8:      $p_{l,k} \leftarrow p_{l,k} + o_k$ 
9:   end for
10:   $p_{l,K+1} \leftarrow p_{l,K+1} + 1$ 
11: end for
```

channels overlap with a neighbors, it affects only $K + 1$ P-channels. Hence, P-channel representations have a tendency to be extremely sparse, i.e., only very few values are non-zero. In our implementation we exploit the sparseness by storing only non-zero values together with their index. This results in a significant speed-up of the channel value determination as well as the further processing.

3 Pose Estimation Based on P-Channels Matching

This section explains the details of the applied technique for pose estimation based on P-channels. We define the feature space for which the P-channel representations are computed, we define the learning problem for the mapping from measured features to pose vectors, and we apply P-channels and SIFT descriptors to solve this learning problem.

3.1 Feature Space

The image features that we will use for the further processing in the P-channel framework are the same as introduced

by Felsberg and Hedborg for the purpose of object recognition [12]. The feature space is chosen in an ad-hoc manner. The choice is motivated by including color information, commonly available fast implementation⁶, robustness against intensity variations, inclusion of geometric information, and stability against geometric distortions.

The first three points motivate the use of hue h and saturation s instead of RGB channels or some advanced color model. The value component v is not included in the feature vector, as absolute intensity is not relevant for the registration. Instead, we use it to compute a simple measure of geometric information. Stability under geometric distortions is trivially fulfilled by a finite linear operator (cf. Sect.1). Therefore, we use an ordinary gradient estimate to determine the local orientation in double-angle representation, cf. [21]:

$$\theta = \arg((\partial_x v + i\partial_y v)^2) . \quad (10)$$

The feature vector is complemented with explicit spatial coordinates, such that the final vector to be encoded is five-dimensional: hue, saturation, orientation, x-, and y-coordinate. For each image point, such a vector is encoded into P-channels, where we used up to 8 channels per dimension. Note in this context that periodic entities (e.g. orientation) can be encoded in exactly the same way as linear ones (e.g. x-coordinate).

3.2 Least-Squares Learning of Pose Estimation

Let $\mathbf{d}$ be an l -dimensional image descriptor and $\mathbf{c}$ an n -dimensional camera pose vector. The actual choice for the descriptor and the pose representation is not relevant in this section and concrete examples will be given in Sect. 3.3. We formulate our pose estimation problem as follows: By supervised learning we want to find the optimal mapping from $\mathbf{d}$ to $\mathbf{c}$, represented by the $n \times l$ matrix $\mathbf{M}$:

$$\mathbf{c} = \mathbf{M}\mathbf{d} . \quad (11)$$

Note that this equation is related to a linear mapping of density estimates if the latter can be expressed as a linear function of $\mathbf{d}$, which is the case if $\mathbf{d}$ is a P-channel representation. The explicit formula for converting P-channels into linear spline density estimates is given in [12]. Optimality is defined in terms of least-squares, mostly due to practical considerations. For least-squares efficient solutions applying the SVD (singular value decomposition) are available.

For the supervised learning, a set of inputs and outputs of the mapping are given: m pairs of vectors $(\mathbf{d}_j, \mathbf{c}_j)$, $j = 1, \dots, m$. The training set is rewritten as $\mathbf{D} = [\mathbf{d}_1 \dots \mathbf{d}_m] \in \mathbb{R}^{l \times m}$ and $\mathbf{C} = [\mathbf{c}_1 \dots \mathbf{c}_m] \in \mathbb{R}^{n \times m}$. In our learning problem, the number of degrees of freedom (unknowns) in the mapping is by orders larger than number of learning pairs (constraint equations). We hence obtain an underdetermined sys-

tem of equations. For solving the system, we apply a minimum norm approach to our optimization problem:

$$\mathbf{M} = \arg \min_{\mathbf{M} \in \mathcal{S}} \|\mathbf{M}\|_F^2, \quad \mathcal{S} = \{\mathbf{M} \in \mathbb{R}^{n \times l}; \mathbf{M}\mathbf{D} = \mathbf{C}\} , \quad (12)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Minimizing the Frobenius norm of $\mathbf{M}$ guarantees stability by giving an upper bound for the energy of the output [17]. The unique solution is obtained as (see e.g. introduction to [9], using an addition transpose operation)

$$\mathbf{M} = \mathbf{C}\mathbf{D}^\dagger , \quad (13)$$

where $\mathbf{D}^\dagger \in \mathbb{R}^{m \times l}$ is the pseudo-inverse of $\mathbf{D}$. For numerical reasons, it is favorable to compute the pseudo inverse by the SVD of $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ [31]:

$$\mathbf{D}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T . \quad (14)$$

The pseudo-inverse of $\mathbf{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_r, 0, \dots, 0)$ reads

$$\mathbf{\Sigma}^\dagger = \text{diag}(\sigma_1^{-1}, \dots, \sigma_r^{-1}, 0, \dots, 0) , \quad (15)$$

where $r \leq m$ is the rank of $\mathbf{D}$.

Note that weighting the respective dimensions of $\mathbf{c}_j$ differently does not affect the result. This can be seen by replacing $\mathbf{C}' = \text{diag}(w_1, \dots, w_n)\mathbf{C}$ and $\mathbf{M}' = \text{diag}(w_1, \dots, w_n)\mathbf{M}$ assuming $\prod_{k=1}^m w_k \neq 0$.

In operation mode, the mapping is applied to query descriptors, i.e., for a given descriptor $\mathbf{d}_q$ we search for the corresponding camera pose $\mathbf{c}_q$. The solution is given in terms of the pseudo inverse computed above:

$$\mathbf{c}_q = \mathbf{M}\mathbf{d}_q = \mathbf{C}\mathbf{D}^\dagger\mathbf{d}_q . \quad (16)$$

Note that due to associativity we can split the RHS into two parts: The second one, $\mathbf{D}^\dagger\mathbf{d}_q$, which is only depending on the selection of descriptors, samples, and the query, results in an index function. This index function indicates the similarity of the query view and the respective sample views and can be used for matching, c.f. [12]. It is independent of the output space, i.e., the pose vector, but the matching scores can be used for interpolation in an arbitrary output matrix / pose matrix $\mathbf{C}$. The latter is the first part of the RHS and its purpose is to map views to poses. Multiplying the scores from the right part with $\mathbf{C}$ results in a linear superposition of pose vectors for the different views.

In [12], it has been shown that the index function defined above results in slightly better ROC (receiver operating characteristic) curves for pose-invariant object recognition, i.e., better recognition rates, than the criterion based on an approximation of the Kullback-Leibler divergence. One explanation for the better performance of (16) is the higher number of degrees of freedom in the P-channel vector compared to the converted P-channel vector used in the KLD.

In the same reference, it was proposed not to use the SVD since it cannot be used for live update of learning problems, i.e., it has to be recomputed for every newly added

⁶ We based our implementation on the OpenCV library <http://sourceforge.net/projects/opencvlibrary/>.

input-output sample. According to [5] it is actually possible to compute the SVD incrementally, and thus, the advantages of linear superposition and better performance can be combined with incremental learning. All experiments below were however based on the ordinary (batch) SVD.

3.3 Pose Estimation by P-Channels and SIFT Descriptors

In order to obtain a proper pose-estimation algorithm from (16), the pose vector space and the image descriptor have to be established. Under the assumption of mild variations of the camera pose, the actual representation of the pose is not crucial and we chose a non-redundant representation of the tangent plane in terms of euclidean position vector and a rotation vector:

$$\mathbf{c} = [p_1 \ p_2 \ p_3 \ r_1 \ r_2 \ r_3]^T. \quad (17)$$

Note that for scenarios with large variations of the pose angle and the camera position, the proper topology of the pose space have to be taken into account and robust representations have to be used. Channel representations of the camera pose seem to be a good choice for achieving this [22].

On the image descriptor side, i.e., for the $\mathbf{d}$ -vector, we propose two solutions: the first one based on P-channels and the second one, for reference, based on SIFT descriptors.

For the first alternative, we encode the image features chosen in Sect. 3.1 with four channels for h , s , and θ , respectively, and eight channels for both spatial coordinates. This results in 24.576 components per view. The channel vectors are then combined in the matrix $\mathbf{D}$.

For the second alternative, we process each RGB channel separately, as SIFT descriptors are defined for gray-scale images. In our experiments this resulted in a significant improvement compared to first converting the image to gray-scales and also to using other color models, as e.g. YCbCr. We extract the standard descriptors of length 128 at 64 regularly spaced positions in the image. This results in the same number of components as the P-channel representation and also assures repeatability. The latter was lost extracting SIFT features at SIFT points, as the order of descriptors becomes permuted. Even with additional SIFT matching step (implementation from [43]), the estimates with regularly spaced SIFT descriptors was always preferable to the one with descriptors at SIFT points.

Therefore, we are using the SIFT descriptors and not the complete SIFT features in this comparison, as already pointed out in Sect. 1. SIFT features are mostly applied in sparse feature matching for estimating epipolar geometry or finding 2D-3D correspondences, whereas the here considered problem is a model-free 2D-2D matching approach. Hence, the results below should not be interpreted as P-channels being superior to SIFT features, but as P-channels performing equally well as or superior to SIFT descriptors when it comes to 2D-2D matching and interpolation. The implementation for the SIFT descriptor extraction is taken from [43], as the original implementation does not exist for all platforms.

Fig. 3 Top: perspective view on the synthetic scene (40 cm deep, 50 cm wide, and 25 cm high). Bottom: example for synthetic fisheye image with noise.

4 Recognition Experiments

In this section we present two experiments, one quantitative using synthetic images from a fisheye camera, and one qualitative experiment with images from a real fisheye camera. As pointed out above, the proposed method is a 2D-2D matching method and therefore we compare our method to a SIFT descriptor based 2D-2D matching method. We make no efforts to compare our method to model-based 2D-3D pose estimation method, since those solve a different problem, where a model of the scene is available.

4.1 Pose Estimation Experiment with Synthetic Data

As it is difficult to acquire test data with noise-free pose vectors from a real fisheye camera, the quantitative experiment has been done on synthetic data with known pose angles. We excluded noisy pose vectors, as the primary focus is on the robust processing of scene views, and not on the regression of pose vectors. The latter task can be accomplished with state of the art methods from control theory [23].

The virtual scene and one instance of the generated synthetic views are depicted in Fig. 3. The images were rendered (linear in angle, 180° field of view) and noise (Gaussian noise with 10% standard deviation of the intensity range [0...255]) was added afterwards. The virtual camera was always located inside the scene area somewhere over the "Vi-

Table 1 Estimation errors for the different tested methods.

method	position RMSE [cm]	rotation error [°]
SIFT/D	0.51	1.1
SIFT/C	1.37	4.3
Channel/D	0.52	1.1
Channel/C	0.21	0.3

talis” package in Fig. 3, pointing roughly downwards. The depth range was thus about 8 cm to about 30 cm.

The views for training were selected by picking all views where the pose vector either has a local extremum or an inflection point. Altogether, we considered 252 views and selected 72 for training. Note that the minimum number of views to span a six-dimensional space is $2^6 = 64$, i.e., we have only selected eight redundant views by the method described above. Choosing views by a different scheme, leads to a slight degradation of results. For practical applications, we recommend to use at least the extrema. If too few extrema are available (less than 64, as it is the case here), one can choose further views pretty arbitrarily, unless they are too close to the already existing views.

The results are summarized in Fig. 4, displaying the estimates of all six pose vector components for four different methods. Tab. 1 lists the errors separately for the position part (root mean square error) and rotation part (angular error according to [28]) of the pose vector. The first position coordinate corresponds to the horizontal position in Fig. 3, the second to the vertical position, and the third to the depth position. The methods considered were: SIFT descriptor based nearest-neighbor estimate (SIFT/D), SIFT based continuous interpolation (SIFT/C), channel based nearest-neighbor estimate (Channel/D), and channel based continuous interpolation (Channel/C). Here, continuous interpolation refers to the superposition according to (16) and nearest-neighbor estimate refers to

$$\mathbf{c}_q = \mathbf{c}_{j_0} \text{ where } j_0 = \arg \max_j (\mathbf{D}^\dagger \mathbf{d}_q)_j . \quad (18)$$

From the numerical results in Tab. 1 it is evident that for the nearest-neighbor case, SIFT and channel based descriptors do not significantly differ. For the case of interpolation, the SIFT based method becomes worse by factor three to four, whereas the channel based method improves by factor three to four. This can be explained by considering the graphs in Fig. 4.

When looking at the different results from linear superposition of the pose vectors, the SIFT based interpolation has a tendency to flatten extrema and to introduce oscillations instead. This is a typical sign of over-fitting with alternating signs, which often results from non-compact basis functions. Linear superposition of P-channels, however, corresponds to a superposition of local linear models, thus avoiding oscillations by locality. The oscillations are the main reason why performance degrades for SIFT descriptors, but improves for channel representations.

Fig. 5 Example for real fisheye image.

4.2 Scene Recognition Experiment with Real Data

The task considered in this experiment is to find, for a given view, the stored view from a catalogue of views which is most similar to the given input view. In this case, the views were obtained with a fisheye lens. As we use real data without known pose angles, see Fig. 5 for an example, the evaluation had to be done manually, based on subjective criteria. This is done in the following way.

In the training, a number of views is selected by a strategy based on visual dissimilarity instead of extremal pose vectors or inflections points. We initialize the recognition system with P-channel representations of two different views, i.e., the initial matrix $\mathbf{D}$ is of size $l \times 2$. For all vectors from the training set $\{\mathbf{d}_j\}_j$, it is now checked whether the index function that responds best delivers a matching score above a certain threshold:

$$\min_j \max_i (\mathbf{D}^\dagger \mathbf{d}_j)_i > \theta , \quad (19)$$

where θ is empirically chosen as 0.48 in order to give a reasonable number of columns in the final $\mathbf{D}$. If and only if (19) is not fulfilled, the vector $\mathbf{d}_j$ with the worst match is added to $\mathbf{D}$, and (19) is re-evaluated.

In operation mode, the views represented in $\mathbf{D}$ are displayed together with a query view and the best match is indicated, see Fig. 6. If and only if the best match is inspected to be the correct view, the recognition task is considered successful.

In our experiment with 167 views, divided into 38 trained views and 129 test views, only two views are not correctly matched. In both cases this happens due to missing fitting views in $\mathbf{D}$, and a better fitting view is scored second. Matching scores are low in both cases.

Interestingly, no views have been selected for $\mathbf{D}$ that contain the person visible in the query view in Fig. 6. The correctness of the matching is not influenced by the partial occlusion caused by the person who appeared in seven views. The matching was not distorted by specular highlights or

Fig. 4 Results from pose estimation. Abbreviations: gt – ground truth, sd – SIFT/D, sc – SIFT/C, cd – Channel/D, and cc – Channel/C.

Fig. 6 Window displaying all views represented by $\mathbf{D}$, one query view (lower right), and the best match indicated in red. The first number at each view indicates the index in the data set. The second number indicates the matching score in %. As it can be seen in this example, the matching is robust to partial occlusion of the scene.

other distortions as comatic and chromatic aberration, or vertical smear (CCD artifact caused by local over-exposure, occurred in 42 instances). The robustness against occlusion and lighting distortions is a particularly important property in context of markerless tracking systems, since the system is initialized, i.e., trained, in front of an empty scene that is later partially occluded by various objects and exposed to other lighting conditions.

Performing the same experiment using SIFT descriptors was somehow problematic, since using a fixed threshold in (19) resulted in much more training views due to a very slowly increasing minimum score. The minimum score decreases in the beginning and it takes more than 38 views before the minimum score is larger than the initial minimum score. Instead, we fixed the number of views to 38, which however leads to lower matching scores in general.

The matching works similar well as the one based on P-channels. Three non-correct matches were found, but this is a non-significant difference for a test set of 129 instances. The matching scores were however less distinct, i.e., it is not possible to predict false matches from the matching score. The SIFT-based method showed the same robustness against occlusion and image distortions as our proposed method.

Overall, the two methods perform much more similarly on the real data than in the synthetic interpolation experiment. This was however expected as no interpolation could be tested on the real data.

4.3 Efficient Implementation

So far we have shown that P-channel based pose estimation works reasonable well, in particular it works better for interpolation than the SIFT-based method. What remains to show is that P-channel encoding can be achieved in video real-time.

First of all, P-channel encoding is fast because only one loop through the image is required [11], see also Alg. 1. For each image point, each image feature hue, saturation, orientation, and position, i.e., five dimensions altogether, is added to the appropriate set of bins. Under appropriate scaling of the feature vector, the multi-index of the bins is computed by rounding the feature vector to the next integer in each component. The five non-integer parts define the respective offsets and the sixth component of the P-channel is increased by one.

After computing the image features only 24 operations per image pixel are required for the encoding. For PAL resolution (720 by 576, 25Hz), this results in less than 250 million operations per second. The initial feature extraction⁷ requires about 32 operations per pixel, including the rescaling. Hence, less than 600 million operations are required for the complete P-channel extraction, and thus, real time performance is realistic on current systems (2.4GHz dual core), even if adding overhead for memory accesses, operative system, etc. This is significantly faster than the extraction of SIFT descriptors, which has also been verified in the implementation (SIFT implementation [43]).

For the matching of descriptors, the computational costs are given by the matrix multiplication (16), i.e., $2 \cdot l \cdot n$ operations are required, which is neglectable compared to the P-channel encoding if the average number of channel per dimension is below twelve. In the described experiments the number of required operations for the matching was below 10 million per second. If the number of channels increases significantly, the P-channels must be implemented as sparse matrices. This limits the effort independent of the number of channels, as an upper bound is obtained that is linear with the number of pixels [11].

Measurements of our C++ implementation confirmed the theoretic considerations above. Without tweaking and special programming (e.g. GPU, using dual core, MMX, etc.), the whole scene registration process takes 0.039 seconds per PAL image, i.e., full framerate at full resolution. This corresponds roughly to about 600 million operations which is reasonable since many operations need more than one clock cycle and some type conversions are required. The main bottleneck of the implementation is the atan2 in (10), which originally took 0.043 seconds (including the necessary type conversions) and which has been replaced by a look-up table to achieve full framerate.

5 Conclusion

We have presented a novel approach to pose recognition and interpolation suitable for real-time applications. The method is based on a sampled kernel density estimate, computed from the P-channel representation of the considered image features. The density estimates from the test set are classified using a least-squares learning technique implemented using the SVD. It has been shown in two experiments with fisheye data that

- the method performs equally well as SIFT descriptor based recognition for finding the closest view despite its real-time capability,
- the method outperforms SIFT descriptor if the pose angle is interpolated using the matching function (SVD),
- the method is robust against occlusion, changes of lighting conditions, and common distortions.

⁷ Note that the feature extraction needs to be done on the whole image in general, as the localization of relevant areas is unknown at this stage.

This makes the method a suitable tool for the initialization of pose tracking in the MATRIS system and similar applications as, e.g., visual navigation.

Acknowledgements We thank our project partners for providing the test data used in the experiments. We thank in particular Graham Thomas, Jigna Chandaria, Gabriele Bleser, Reinhard Koch, and Kevin Koeser.

References

1. S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, 2004.
2. A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondence. In *IEEE Computer Vision and Pattern Recognition*, 2005.
3. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.
4. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
5. M. Brand. Incremental singular value decomposition of uncertain data with missing values. Technical Report TR-2002-24, Mitsubishi Electric Research Laboratory, May 2002.
6. Q. Chen, M. Defrise, and F. Deconinck. Symmetric phase-only matched filtering of Fourier-Mellin transforms for image registration and recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1156–1168, December 1994.
7. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, New York, 1991.
8. E. Dimitriadou, A. Weingessel, and K. Hornik. Fuzzy voting in clustering. In *Fuzzy-Neuro Systems*, pages 63–75, 1999.
9. G. Farneback. Spatial domain methods for orientation and velocity estimation. Lic. Thesis LiU-Tek-Lic-1999:13, Dept. EE, Linköping University, March 1999.
10. M. Felsberg, P.-E. Forssén, and H. Scharf. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):209–222, 2006.
11. M. Felsberg and G. Granlund. P-channels: Robust multivariate m-estimation of large datasets. In *International Conference on Pattern Recognition*, Hong Kong, August 2006.
12. M. Felsberg and J. Hedborg. Real-time visual recognition of objects and scenes using p-channel matching. In *Proc. 15th Scandinavian Conference on Image Analysis*, volume 4522 of LNCS, pages 908–917, 2007.
13. M. Ferraro and T. M. Caelli. Lie transformation groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8(4):33–44, 1994.
14. R. B. Fisher, K. Dawson-Howe, A. Fitzgibbon, C. Robertson, and E. Trucco. *Dictionary of Computer Vision and Image Processing*. Wiley & Sons, 2005.
15. P.-E. Forssén. *Low and Medium Level Vision using Channel Representations*. PhD thesis, Linköping University, Sweden, 2004.
16. M. S. Gazzaniga, R. B. Ivry, and G. R. Mangun. *Cognitive Neuroscience, Second Edition*. W. W. Norton & Company, March 2002.
17. K. Gopalsamy. Stability of artificial neural networks with impulses. *Applied Mathematics and Computation*, 154(3):783–813, 2004.
18. G. H. Granlund. The complexity of vision. *Signal Processing*, 74(1):101–126, April 1999. Invited paper.
19. G. H. Granlund. An Associative Perception-Action Structure Using a Localized Space Variant Information Representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.
20. G. H. Granlund. Personal communication, 2006.
21. G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, Dordrecht, 1995.

22. G. H. Granlund and A. Moe. Unrestricted recognition of 3-d objects for robotics using multi-level triplet invariants. *Artificial Intelligence Magazine*, To appear 2004.
23. F. Gustafsson. *Adaptive filtering and change detection*. John Wiley & Sons, Ltd., 2000.
24. J. Hol, T. B. Schön, H. Luinge, P. Slycke, and F. Gustafsson. Enabling real-time tracking by fusing measurements from inertial and vision sensors. *submitted to the same issue*, 2007.
25. B. Johansson, T. Elfving, V. Kozlov, Y. Censor, P.-E. Forssén, and G. Granlund. The application of an oblique-projected landweber method to a model of supervised learning. *Mathematical and Computer Modelling*, 43:892–909, 2006.
26. E. Jonsson and M. Felsberg. Reconstruction of probability density functions from channel representations. In *Proc. 14th Scandinavian Conference on Image Analysis*, 2005.
27. E. Jonsson and M. Felsberg. Accurate interpolation in appearance-based pose estimation. In *Proc. 15th Scandinavian Conference on Image Analysis*, volume 4522 of *LNCS*, pages 1–10, 2007.
28. H. Knutsson and M. Andersson. Robust N-dimensional orientation estimation using quadrature filters and tensor whitening. In *Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing*, Adelaide, Australia, April 1994. IEEE.
29. N. Krüger. Learning object representations using a priori constraints within ORASSYLL. *Neural Computation*, 13(2):389–410, 2001.
30. D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
31. M. Mühlich and R. Mester. A considerable improvement in non-iterative homography estimation using TLS and equilibration. *Pattern Recognition Letters*, 22:1181–1189, 2001.
32. E. Murphy-Chutorian, S. Aboutalib, and J. Triesch. Analysis of a biologically-inspired system for real-time object recognition. *Cognitive Science Online*, 3:1–14, 2005.
33. D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *IEEE Computer Vision and Pattern Recognition*, 2006.
34. Š. Obdržálek and J. Matas. Sub-linear indexing for large scale object recognition. In W. F. Clocksin, A. W. Fitzgibbon, and P. H. S. Torr, editors, *BMVC 2005: Proceedings of the 16th British Machine Vision Conference*, volume 1, pages 1–10, London, UK, September 2005. BMVA.
35. M. Pontil and A. Verri. Support vector machines for 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):637–646, 1998.
36. D. Roobaert, M. Zillich, and J.-O. Eklundh. A pure learning approach to background-invariant object recognition using pedagogical support vector learning. In *IEEE Computer Vision and Pattern Recognition*, volume 2, pages 351–357, 2001.
37. J. Skoglund and M. Felsberg. Evaluation of subpixel tracking algorithms. In *International Symposium on Visual Computing*, volume 4292 of *LNCS*, pages 374–382, 2006.
38. J. Skoglund and M. Felsberg. Covariance estimation for sad block matching. In *Proc. 15th Scandinavian Conference on Image Analysis*, volume 4522 of *LNCS*, pages 372–382, 2007.
39. H. P. Snippe and J. J. Koenderink. Discrimination thresholds for channel-coded systems. *Biological Cybernetics*, 66:543–551, 1992.
40. D. Stricker, G. Thomas, and J. Chandaria. The matrix project: real-time markerless camera tracking for ar and broadcast applications. *submitted to the same issue*, 2007.
41. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.
42. M. Unser. Splines – a perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16:22–38, November 1999.
43. A. Vedaldi. An open implementation of SIFT. <http://vision.ucla.edu/~vedaldi/code/sift/sift.html>, last accessed 23/05/2007, 2007.