Egocentric Visual Event Classification with Location-Based Priors

Sudeep Sundaram and Walterio W. Mayol-Cuevas

Department of Computer Science, University of Bristol

Abstract. We present a method for visual classification of actions and events captured from an egocentric point of view. The method tackles the challenge of a moving camera by creating deformable graph models for classification of actions. Action models are learned from low resolution, roughly stabilized difference images acquired using a single monocular camera. In parallel, raw images from the camera are used to estimate the user's location using a visual Simultaneous Localization and Mapping (SLAM) system. Action-location priors, learned using a labeled set of locations, further aid action classification and bring events into context. We present results on a dataset collected within a cluttered environment, consisting of routine manipulations performed on objects without tags. [1]

1 Introduction and Related Work

Visual event and activity classification has been mostly studied for cases when the camera is static and/or where the action is well centered and localized in the image [1–4]. Less work has been concerned with the case of a moving camera, which is the situation in systems that are observing inside-out e.g. a wearable system.

Detecting events and activity on the move can lead to assistive devices and this is indeed one of the primary goals for work carried out in this area. Examples of this are applications ranging from monitoring systems for the elderly and disabled [5–7], to systems that ”watch and learn” how to carry out complex tasks [8, 9].

When used for sequential activity recognition, knowledge of *where* the user is in each time step, can play a vital role in ensuring robustness of the system. Location also brings about the all-important element of context in terms of the user's interaction with the immediate environment. Benefits of using location for recognition of user activity have previously been demonstrated. In [10] and [6], *only* location is used, while in [11], location is combined with signals obtained from a microphone to recognize activity. To the best of our knowledge, location has yet to be combined with human actions to recognize events and activity on the move. Several recent systems have demonstrated significant interest in human action recognition using cameras, although again a majority of the methods deal with appropriately placed static cameras.

Feature descriptors used for representing actions, can be broadly classified based on sparse or dense sampling of feature points from space-time representations of actions.

[1] The authors are deeply grateful to the British Council for the PhD studentship granted to SS, and to the EUFP7 COGNITO project for partially funding WMC.

Space-time interest points for action recognition were used in [12–14] and yielded considerably good results. However, densely sampled features have been shown to generally perform better [15]. In particular, Histograms of Oriented Gradients [1] has been a popular choice of feature descriptor for actions [2–4].

Visual sensing for user location meanwhile, has recently seen important advances and of particular importance here, are those methods amenable to real-time performance. Specifically, some works related to localization and mapping [16, 17] have developed fast methods for re-location.

2 Motivation and Contributions

Any method for recognition of egocentric manipulations must address problems arising due to - (1) camera motion, (2) changes in camera vantage point, (3) variations in the way a manipulation is performed, and (4) computational efficiency, to enable real-time performance.

In order to address the first problem, we carry out coarse stabilization of the input sequence to compensate for camera motion, as described in Section 3.1. The presented method learns *translation-invariant, deformable* graph models (covered in Section 5) for each manipulation class, thus addressing the second and third issues. Computational efficiency is ensured by classifying actions using low resolution images.

In order to build on the use of location for activity recognition, we use the same wearable camera to estimate the user's location within a labeled (not tagged) environment using a Simultaneous Localization and Mapping system. Learned prior distributions of manipulations performed in known locations are used to enhance the classifier's performance.

The contributions of this paper are two-fold - (1) the use of deformable graph-based action models for classification of egocentric visual events, and (2) the estimation of user location from the same sensor to bring events into context, resulting in improved classification.

3 Action Cell

This section describes the steps leading up to feature-based representations of action sequences. Our approach for event classification aims to avoid the recognition of individual objects that are being manipulated, and concentrates instead on the more generic hand and arm motion and the general working scene detection. This demands careful extraction of manipulation data with reduced background noise.

3.1 2-D Affine Image Registration

Given our use of a wearable camera, we first attempt to roughly compensate for camera motion relative to the background. Let I_k be the current frame being processed, and I_{k+1} be the next incoming frame. Let $\phi(I, \mu)$ denote the result obtained when an image I is affine transformed by parameter vector μ. We approximate camera motion compensation by finding the affine transformation parameter vector ζ that minimizes the absolute intensity difference between $\phi(I_k, \zeta)$, and I_{k+1}. Estimation of ζ and the corresponding stabilized difference image D_k is given by:

$$\zeta = argmin_{\mu}(I_{k+1} - \phi(I_k, \mu)) \tag{1}$$

$$D_k = I_{k+1} - \phi(I_k, \zeta) \tag{2}$$

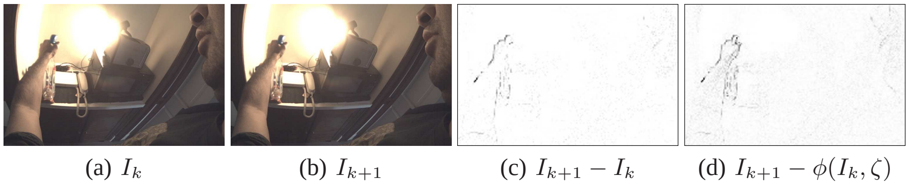

(a) I_k (b) I_{k+1} (c) $I_{k+1} - I_k$ (d) $I_{k+1} - \phi(I_k, \zeta)$

Fig. 1. Consecutive frames (*a, b*) from a "Spray" action sample, with contrast normalized difference images (*c, d*). Once camera motion is compensated, foreground motion of the hand and object is more clearly visible

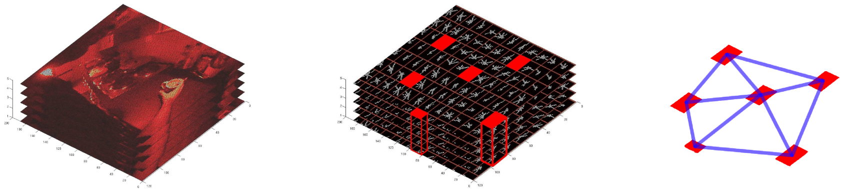
3.2 Action Cell Extraction and Matching

The volume of stabilized difference images over the action sequence is then split into 16x16xt spatio-temporal blocks, where t is the temporal length of the sequence. Histograms of oriented gradients are computed from the image contained within each time step in each spatio-temporal block, as described in [1]. The histograms are then concatenated over the entire temporal length to obtain a feature vector of size $n_b \times t$, where n_b is the number of histogram bins. We term this feature vector, along with information about its spatial location in the image, as an *action cell*.

In order to match any two given action cells a_1 and a_2, respectively of lengths t_1 and t_2, a distance matrix $\varDelta$ of size $t_1 \times t_2$ is constructed such that cell $\varDelta_{xy}$ contains the L_2 distance between histograms $x \in a_1$ and $y \in a_2$. Normalized dynamic time warping is used to compute a matching cost between a_1 and a_2 by finding the shortest path through $\varDelta$.

4 Action Fragment

A number of action cells belonging to a single action sequence may be modeled as vertices of a graph, which we term as an *action fragment*. This section deals with the extraction and matching of action fragments. Formally, any given action sequence A can be converted to a sequence of roughly stabilized difference images, which in turn is used to generate a set of action cells α i.e. $A := \{\alpha_i \ : \ i = 1, 2, ..., |\alpha|\}$. An action fragment is located on the action sequence as a set of unique action cells, and is modeled as graph $\varUpsilon_A = \langle \upsilon_A, \varepsilon_A \rangle$, whose vertices $\upsilon_A \subseteq \alpha$ are the action cells, and edges ε_A may be found using one of various methods such as Delaunay Triangulation.



(a) (b) (c)

Fig. 2. Example of an action sequence (*a*) represented as a set of action cells (*b*). Randomly picked action cells form the vertices of a graph that represents a seed fragment (*c*)

4.1 Seed Fragment

The first step in every iteration of the learning process is to randomly pick an action fragment in some given action sequence. In the above example, this is done by specifying $|\upsilon_A| : 1 \leq |\upsilon_A| \leq |\alpha|$, and populating υ_A with action cells randomly picked from α. ε_A is determined by performing a Delaunay Triangulation on the spatial centroids of υ_A. An example of this process is shown in Figure 2.

4.2 MRF-Based Fragment Localization

Consider any two action sequences A and B. Let $\Upsilon_A = \langle \upsilon_A, \varepsilon_A \rangle$ be an action fragment computed from A. This section deals with the localization of Υ_A in B. In other words, we attempt to find the action fragment $\Upsilon_B = \langle \upsilon_B, \varepsilon_B \rangle$ such that the cost $C(\Upsilon_A, \Upsilon_B)$ of matching Υ_A to Υ_B is minimized:

$$\Upsilon_B = argmin_{\Upsilon_b} \left(C(\Upsilon_A, \Upsilon_b) \right) \; \forall \, \Upsilon_b \in B \tag{3}$$

Finding an exact solution for Υ_B is clearly NP-hard. Instead, we use the MAX-SUM approach [18] to find an approximation, in a manner previously adopted in [19].

The matching cost can be measured as:

$$C(\Upsilon_A, \Upsilon_B) = C(\upsilon_A, \upsilon_B) + C(\varepsilon_A, \varepsilon_B) \tag{4}$$

where $C(\upsilon_A, \upsilon_B)$ measures the cost of matching corresponding action cells between the two fragments, and $C(\varepsilon_A, \varepsilon_B)$ measures the cost of matching structures of the two graphs. The dual of Equation 4 would be to maximize the *qualities* of υ_B and ε_B. The overall quality of the localized fragment Υ_B, to be maximized, is given by:

$$Q(\Upsilon_B) = \sum_{\upsilon \in \upsilon_B} Q(\upsilon) + \sum_{\varepsilon \in \varepsilon_B} Q(\varepsilon) \tag{5}$$

Υ_B is computed using a Markov Random Field, which represents a graph consisting of $M = |\upsilon_A|$ nodes. The adjacency of the nodes is maintained as in Υ_A. Each node, called an object, consists of N fields or labels, with associated qualities. The labels of two adjacent nodes are fully connected by N^2 edges. An example of such a graph is shown in

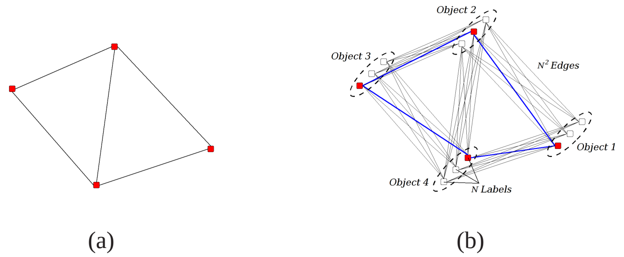


Fig. 3. Fragment Localization using a Markov Random Field. The model graph (action fragment) in (*a*) is localized on the MRF (*b*). The localized action fragment is highlighted by shaded (red) vertices and thick (blue) edges

Figure 3. Maximizing Equation 5 is equivalent to finding a maximum posterior configuration of the MRF shown. In the current problem, labels corresponding to some Object i represent the N action cells in υ_B, that are most similar to $\upsilon_A(i)$. Labels are found using an exhaustive search through the set υ_B, using the matching technique described in Section 3.2. The quality of a label is inversely proportional to the cost for matching the label to its corresponding object. Dummy labels, with relatively low qualities, are added to each object, to facilitate localization where one or more objects remain unmatched. Label qualities for a single object are normalized to have a maximum value of 0 and a median of -1, and therefore lie in the range $[-\infty, 0]$. The quality of an edge is computed as the weighted sum of its length and orientation similarities to the corresponding edge in the model graph. Edge qualities are normalized to lie in the range $[-1, 0]$.

Let the MxN matrix L represent the label qualities for each of the objects, and the $|\varepsilon_A|$xN^2 matrix E represent edge qualities between pairs of labels. The total quality of the labeling $S = \{n_1, ..., n_M\}$ with $n_i \in \{1, ..., N\}$ is given by

$$Q(S) = \sum_{m=1}^{M} L(m, S(m)) + \sum_{e=1}^{|\varepsilon_A|} E(e, \beta(E, S, \varepsilon_A)) \qquad (6)$$

where $\beta(E, S, \varepsilon_A)$ denotes the column representing the edge between the labels chosen to represent the edge $\varepsilon_A(e)$. Rewriting Equations 3, 4 and 5 in terms of the MAX-SUM problem, Υ_B can be computed by finding the set $S^* = argmax_S(Q(S))$.

5 Learning Action Models

Consider a dataset of manipulations $\Lambda = \{\Lambda_i \ : \ i = 1, ..., Z\}$ labeled class-wise, consisting of Z classes. Let any class i be represented by the set $\Lambda_i = \{\Lambda_{ij} \ : \ class(\Lambda_{ij}) = i \ ; \ j = 1, ..., |\Lambda_i|\}$ of manipulations Λ_{ij}.

5.1 Fragment Models

Consider a seed fragment Υ_m chosen from a randomly picked manipulation sample Λ_{im}, as described in Section 4.1. In order to build a fragment model, we first localize

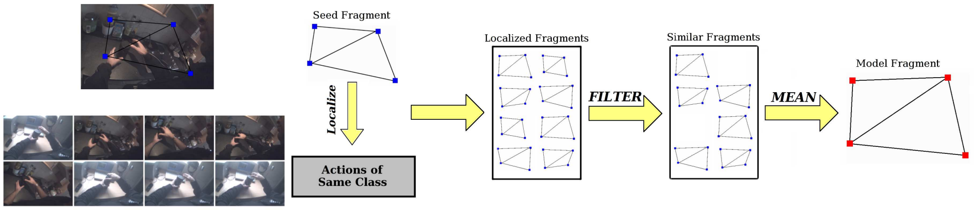


Fig. 4. From seed fragment $\varUpsilon_m$ to similar fragments κ to model fragment $\varUpsilon_f$

$\varUpsilon_m$ in all samples $\varLambda_{ij} \in \varLambda_i$, resulting in a set of similar action fragments κ.

$$\kappa = \{\varUpsilon_j \ : \ j = 1, ..., |\varLambda_i| \ ; \ \varUpsilon_j = argmin_{\varUpsilon} \left(C(\varUpsilon_m, \varUpsilon) \right) \ \forall \, \varUpsilon \in \varLambda_{ij} \} \tag{7}$$

Each $\varUpsilon_j$ is evaluated using the quality of localizing $\varUpsilon_m$ on $\varLambda_{ij}$, obtained using Equation 6. If this quality is found to be low, then $\varUpsilon_j$ is discarded from κ. Once all fragments are validated, if the size of κ is too small, then a new seed $\varUpsilon_m$ is found, and the process repeats. This filtering step ensures *consistency* across learned action fragments of the same class.

If the size of κ is large enough, a fragment model can be created. Using each action cell $\upsilon_m \in \varUpsilon_m$ as a reference, each of the corresponding action cells $\upsilon_j \in \varUpsilon_j$ are re-aligned using dynamic programming as described in Section 3.2 so that they are of the same length as υ_m. Mean feature vectors are then computed at each time step, resulting in a learned set of action cells. These action cells form the vertices of the learned fragment model $\varUpsilon_f$.

We now run a filtering step to check whether or not to retain $\varUpsilon_f$, by attempting to localize $\varUpsilon_f$ on all samples in $\varLambda$. If $\varUpsilon_f$ has been generated using a seed from class i, and q_{jk} represents the quality obtained by localizing $\varUpsilon_f$ on some sample $\varLambda_{jk}$, then the quality Q_f of $\varUpsilon_f$ is computed as follows:

$$\eta_s = \frac{1}{|\varLambda_i|} \sum_{j=1}^{Z} \sum_{k=1}^{|\varLambda_j|} \delta(i,j) e^{q_{jk}} \tag{8}$$

$$\eta_d = \frac{1}{|\varLambda| - |\varLambda_i|} \sum_{j=1}^{Z} \sum_{k=1}^{|\varLambda_j|} (1 - \delta(i,j)) e^{q_{jk}} \tag{9}$$

$$Q_f = \frac{\eta_s}{\eta_s + \eta_d} \tag{10}$$

where $\delta(.,.)$ is the Dirac delta function.

If Q_f is high enough, then the component action cells are assigned weights equal to Q_f and added to the consensus set $\varPsi_i$. If Q_f is not high enough, the fragment is discarded and the process repeats. This step ensures that the learned fragment is *discriminative* across classes.

Note that the graphs representing all $\Upsilon_j \in \kappa$ have the same adjacency matrix, since they were all obtained by localizing a single seed. While this structure remains consistent, the spatial location of each of the corresponding action cells is likely to vary. This makes the fragment model *translation-invariant* and *deformable*. Changes in the camera mount and/or changes in the user's pose will thus, have little effect on classification.

5.2 Fragments to Actions

Section 5.1 described the procedure to obtain a single fragment model, and assign weights to the component action cells. This procedure is repeated a number of times for each class, resulting in consensus sets $\{\Psi_i \, : \, i = 1, ..., Z\}$. The next step is to convert each of these consensus sets into models that can be used for action classification. Figure 5(a) shows an example consensus set, that consists of a number of action cells, obtained from "high quality" action fragments. It is likely that highly discriminative action cells are present as part of more than one action fragment, while the less discriminative ones may occur only once. In order to retain the more important action cells, we perform Euclidian distance based K-Means clustering, as shown in Figure 5(b). Clusters with low populations are discarded. In each cluster that remains, the action cell with the highest weight is retained as a representative, and is assigned a weight equal to the sum of the weights of all elements in the cluster. The remaining action cells in the cluster are discarded.

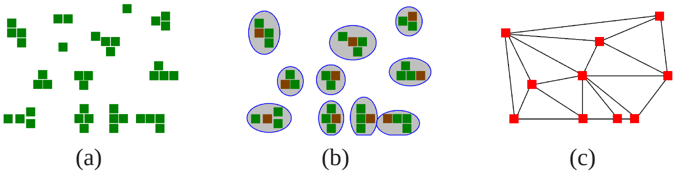

(a) (b) (c)

Fig. 5. Consensus set Ψ_i (*a*), clustered using KMeans (*b*) to form the action model ψ_i (*c*)

We now have a set of weighted highly discriminative action cells. We use these action cells as vertices of a graph, whose edges are given by performing a Delaunay Triangulation on the centroids of the action cells. An example is shown in Figure 5(c). The vertex weights in this graph are normalized to form a probability distribution, while the edges are assigned equal weights. This gives rise to an *action model* ψ_i for each class.

The model-building process described above takes place offline on a subset of the available dataset. Classification of the actions, on the other hand, is designed to be online, and happens immediately after the action is complete.

5.3 Action Classifier

Given a test action, it is matched with the learned models using the MRF-based method described in Section 4.2. This time, instead of finding action fragments, the matcher attempts to find the graphical action model on the test action. The recognized class ω for any input action Ω is given by $\omega = argmax_i(Q(\psi_i, \Omega))$, where $Q(\psi_i, \Omega)$ is the quality of localizing action model ψ_i on action Ω.

6 User Location for Action Classification

Daily routine actions and activities are usually performed in the same environment(s) on a regular basis. We aim to take advantage of consistent information available in the user's surroundings to improve event classification. Sparse maps of 3D features representing the user's environment are built, from the camera that is used for action classification, using a Simultaneous Localization and Mapping system. We are more interested in the localization part, and thus the maps are built offline and stored. Probability distributions of actions performed in each map are learned, and used as priors to improve the accuracy of the action classifier.

Given some action sequence, the SLAM system is pushed into relocalization mode. In this manner, there is a resemblance with the work of [20] where disjoint locations are used for placing information. Here however, we use the method for relocalization described in [17] for its performance and reduced memory requirements. The localization method is robust to a degree of alterations in the mapped environment as produced by objects moving or being occluded. If the system manages to relocalize in one of the stored maps, the corresponding action prior is loaded and used to improve classification accuracy.

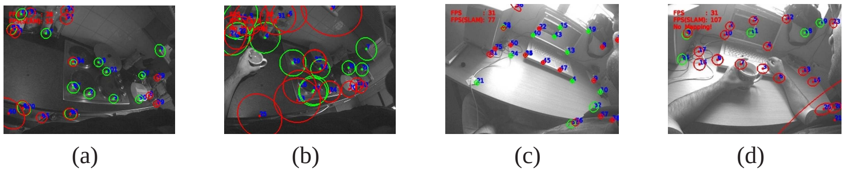
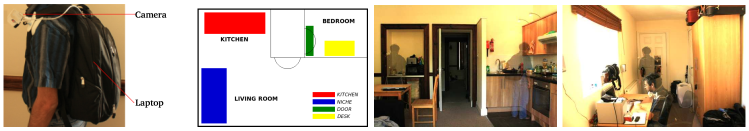

(a) (b) (c) (d)

Fig. 6. User Location is provided by Relocalization in a 3D SLAM map. Figures (*a*,*c*) show examples of building a SLAM map offline, while (*b*,*d*) show relocalization in the classifier

7 Experiments and Results

7.1 Dataset

Manipulation samples were collected by a single candidate over different days spanning 4 weeks - in order to capture natural variations in action. Samples were collected for 7 manipulation classes in 4 locations. Figure 7 shows the user setup and a map of the environment, followed by snapshots of the 4 locations where samples were collected. The manipulation classes (and the location(s) they were performed in) include *Answer Phone* (Niche), *Chop* (Kitchen), *Drink* (Kitchen, Desk), *Open Door* (Door), *Pour* (Kitchen), *Spray* (Kitchen, Niche), *Unscrew* (Kitchen, Desk). In all, the dataset consists of 277 manipulations, with a minimum of 34 per class.

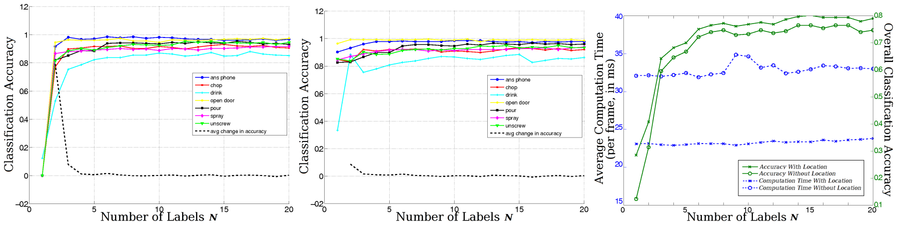


(a) (b) (c) (d)

Fig. 7. User Setup and Environment. (*a*) shows the user with the wearable setup. (*b*) shows the environment used, consisting of (*c*) kitchen, niche, (*d*) desk and door locations

7.2 Results

A subset of the dataset - 12 samples from each class - was randomly selected to train the classifier. The learned models provided a classification accuracy of 95.24% on the training set alone. In order to analyze the performance of the classifier, statistics were generated over the entire dataset for varying values of N (number of labels in MRF), both with and without the use of location-based priors. Classification accuracies are measured for each individual action model as $\eta_i = \frac{TP(i)+TN(i)}{|\Lambda|}$, where $TP(i)$ and $TN(i)$ are respectively the number of "true positive" and "true negative" classifications for class i. The overall classification accuracy is computed as $\eta_{all} = \frac{\sum_{i=1}^{Z} TP(i)}{|\Lambda|}$.



(a) (b) (c)

Fig. 8. Classification accuracies for each class (a) without location and (b) with location, with the dashed line indicating the change in average individual classification accuracy. (c) Overall classification accuracy, compared with computation times per frame

Figure 8 contains classification accuracy plots for individual classes, and for the overall dataset analyzed against varying values of N. Classification accuracies for individual classes (see Figures 8(a) and 8(b)) remain consistently above 80% both with and without the use of location, for $N \geq 3$. With location, the overall classification improves by 4.3% on average, but more importantly reduces the computation time by 29.23%, due to the reduced number of models to be matched (Figure 8(c)).

8 Conclusion

We have presented a method that learns probabilistic graphical models to describe actions observed from a wearable camera. Further, we used the same sensor to estimate the user's location, and combined this information with the action classifier to improve its accuracy and performance. The results also validate the use of a monocular camera as a stand-alone sensor, capable of recognizing user manipulation activity without the need to recognize individual objects. Future work involves tests of our method for a number of candidates over longer periods of time. An extension to this work will involve automatic detection and classification of events from continuous video.

References

1. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
2. Klaser, A., Marszalek, M., Schmid, C.: A spatio-temporal descriptor based on 3d-gradients. In: BMVC. (2008)
3. Yuan, J., Liu, Z., Wu, Y.: Discriminative subvolume search for efficient action detection. In: CVPR. (2009)
4. Hu, Y., Cao, L., Lv, F., Yan, S., Gong, Y., Huang, T.: Action detection in complex scenes with spatial and temporal ambiguities. In: ICCV. (2009)
5. Tapia, E.M., Intille, S.S., Larson, K.: Activity recognition in the home using simple and ubiquitous sensors. Pervasive Computing (2004)
6. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. Artificial Intelligence **171** (2007) 311–331
7. Van Laerhoven, K., Berlin, E.: When else did this happen? efficient subsequence representation and matching for wearable activity data. In: ISWC. (2009)
8. Kuniyoshi, Y., Inaba, M., Inoue, H.: Learning by watching: Extracting reusable task knowledge from visual observation of human performance. IEEE Transactions on Robotics and Automation **10** (1994) 799–822
9. Hovland, G.E., Sikka, P., McCarragher, B.J.: Skill acquisition from human demonstration using a hidden markov model. In: IROS. (1996)
10. Aoki, H., Schiele, B., Pentland, A.: Realtime personal positioning system for wearable computers. In: ISWC. (1999)
11. Clarkson, B., Mase, K., Pentland, A.: Recognizing user context via wearable sensors. In: ISWC. (2000)
12. Laptev, I., Lindeberg, T.: Space-time interest points. In: ICCV. (2003)
13. Dollar, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: VS-PETS. (2005)
14. Patron Perez, A., Reid, I., Patron, A., Reid, I.: A probabilistic framework for recognizing similar actions using spatio-temporal features. In: BMVC. (2007)
15. Wang, H., Ullah, M.M., Klaser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC. (2009)
16. Williams, B., Klein, G., Reid, I.: Real-time SLAM relocalisation. In: ICCV. (2007)
17. Chekhlov, D., Mayol Cuevas, W., Calway, A.: Appearance based indexing for relocalisation in real-time visual slam. In: BMVC. (2008)
18. Werner, T.: A linear programming approach to max-sum problem: A review. PAMI **29** (2007) 1165–1179
19. Donner, R., Micusik, B., Langs, G., Bischof, H.: Sparse mrf appearance models for fast anatomical structure localisation. In: BMVC. (2007)
20. Castle, R.O., Klein, G., Murray, D.W.: Video-rate localization in multiple maps for wearable augmented reality. In: ISWC. (2008)