If there are no stupid questions, then what
kind of questions do stupid people ask?
Do they get smart just in time to ask questions?
(Scott Adams)

Appendix F

Shape-Based Analysis of Object Behaviour

Once the shapes of the video objects have been determined by an automatic segmentation algorithm, it is interesting to apply further processing to extract semantically high information. For example, the obtained object masks can be used to identify the object and assign it to classes like *human*, *car*, *bird*, and so on. Furthermore, objects usually do not appear static, but they perform some action in the video sequence, which can also be analysed and assigned to sub-classes like "walking human", "standing human", or "sitting human". The analysis of the sequence of object sub-classes over time can be considered as extraction of object behaviour.

In this appendix, experiments are described that we conducted to extract a description of the object behaviour based on the object shape. For the analysis, we combined a classification of the object shape into several pre-defined classes with a model of the transition probability between these classes over time. Having a model that describes the transitions between classes makes the classification more robust than an independent classification for each input frame, because occasional false classifications are avoided by small transition probabilities.

Figure F.1: *Best database match for some automatically segmented object masks. The query masks as shown in the top row and the best matching shape is depicted in the bottom row, respectively.*

F.1 Classification of object shapes

A popular technique to classify objects based on their shape is the *Curvature Scale Space* (CSS) technique [128, 4]. Essentially, this technique represents the shape of an object with a low-dimensional feature-vector. The representation makes it easy to obtain rotation and scaling-invariant feature vectors. To classify a specific object, we use a database of manually-classified objects, in which the CSS feature-vector and the class identifier is saved for each object. Using a specifically designed distance function for CSS feature-vectors [62, 105], we compare the shape of the segmented object with all objects in the database. For an independent classification, we would select the object class with the smallest CSS distance value. Unfortunately, segmentation errors can distort the shape of the object, and also the CSS technique itself has a certain error rate. Both can lead to false classifications. An example of independent classification is depicted in Figure F.1.

To increase the robustness, we do not perform an independent classification for the shape extracted from each frame, but we use contextual information from other frames to make the classification more robust. In order to do this, we compute for each input frame f the CSS distances of the query shape to each object class c and store it as $d_c(f)$.

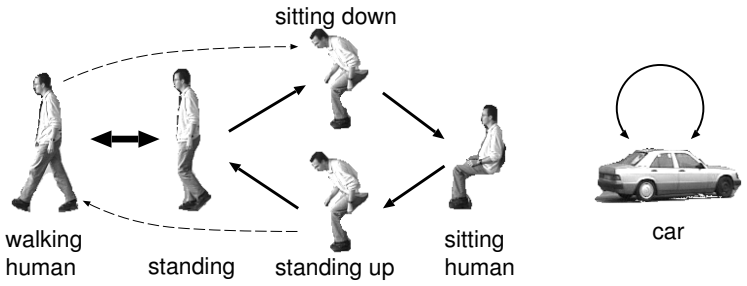

Figure F.2: *Transitions between various object classes. Bolder arrows indicate more probable transitions.*

F.2 Simple model for object behaviour

Real-world objects cannot suddenly change their class in an arbitrary way. For example, a human can never become a car for just a couple of frames, even if its shape suggests this. But, if we further subdivide each class into behaviour sub-classes, transitions may occur. Usually, even though an object can appear in all sub-classes, there are restrictions for state changes. In Figure F.2, transitions between some sub-classes for human motion and an independent car class are shown. According to this model, a human can walk, stand, and sit, but prior to sitting, he has to pass the *sitting-down* state first. The possible transitions can also be weighted, such that *sitting-down* has a higher cost (because it is not so probable) as just continuing to walk. More formally, the weights for all transitions from a general state i to state k can be collected in a state transition matrix $w_{i,k}$. For our experiments, we have manually edited this transition matrix based on typical error values as observed in the CSS shape matching step.

F.3 Behaviour analysis

Having the independent shape-matching costs and the state-transition costs available, we can compute the most probable class labels c_f for each frame f by minimizing the total cost

$$\min_{(c_f)_f} \left\{ d_{c_1}(1) + \sum_{k=2}^{N} (d_{c_k}(k) + w_{c_k-1,c_k}) \right\}, \tag{F.1}$$

where N is the total number of frames. This minimization problem can be solved by considering it a minimum-cost path problem in a graph with nodes $V = \bigcup_f V_f$ composed of columns $V_f = \{(c, f)\}_c$ and edges $E = \bigcup_f V_f \times V_{f+1}$

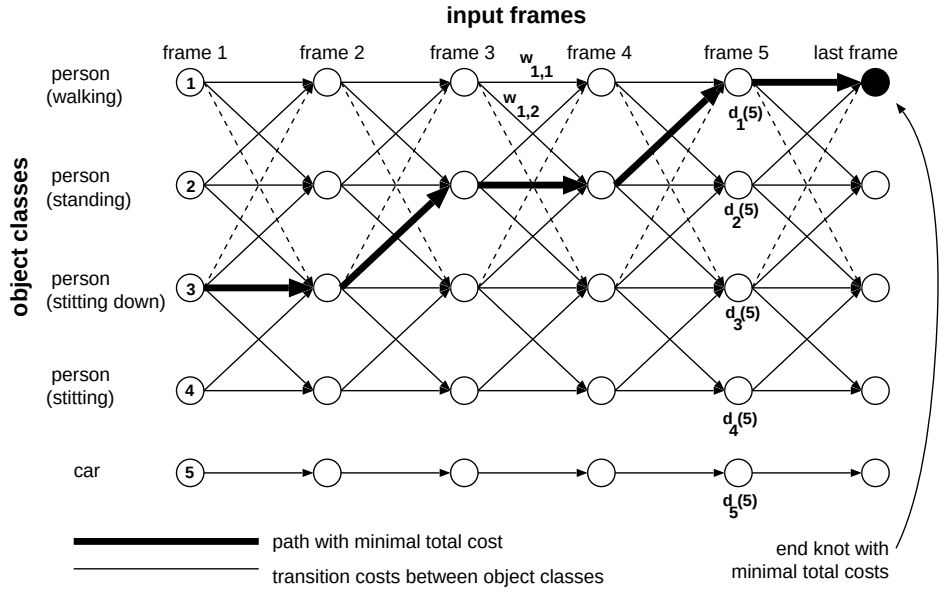

Figure F.3: *Computation graph for the classification of object shapes. The minimum-cost path in the graph defines the class for each frame.*

between successive columns. Nodes (c, f) are attributed with costs $d_c(f)$ and edges $((c_i, f), (c_k, f + 1))$ with $w_{i,k}$. The resulting graph is depicted in Figure F.3 for the example outlined above.

Results for the described example model are depicted in Figure F.4. The results for both humans were obtained with the same model without any parameter adaptation. We consider the presented algorithm and example result as a proof-of-concept implementation. Future work should replace the heuristic cost functions with experimentally-determined probabilities $p_f(c)$ instead of $d_c(f)$, and transition probabilities $p(c_f|c_{f-1})$. Similarly to Eq. (F.1), we obtain the total probability

$$\min_{(c_f)_f} \left\{ p_1(c_1) \cdot \prod_{k=2}^{N} (p(c_k|c_{k-1}) p_k(c_k)) \right\}. \tag{F.2}$$

The products in this equation can be transformed to sums by considering the log-likelihoods. This results in an optimization problem similar to the above shortest-path problem.

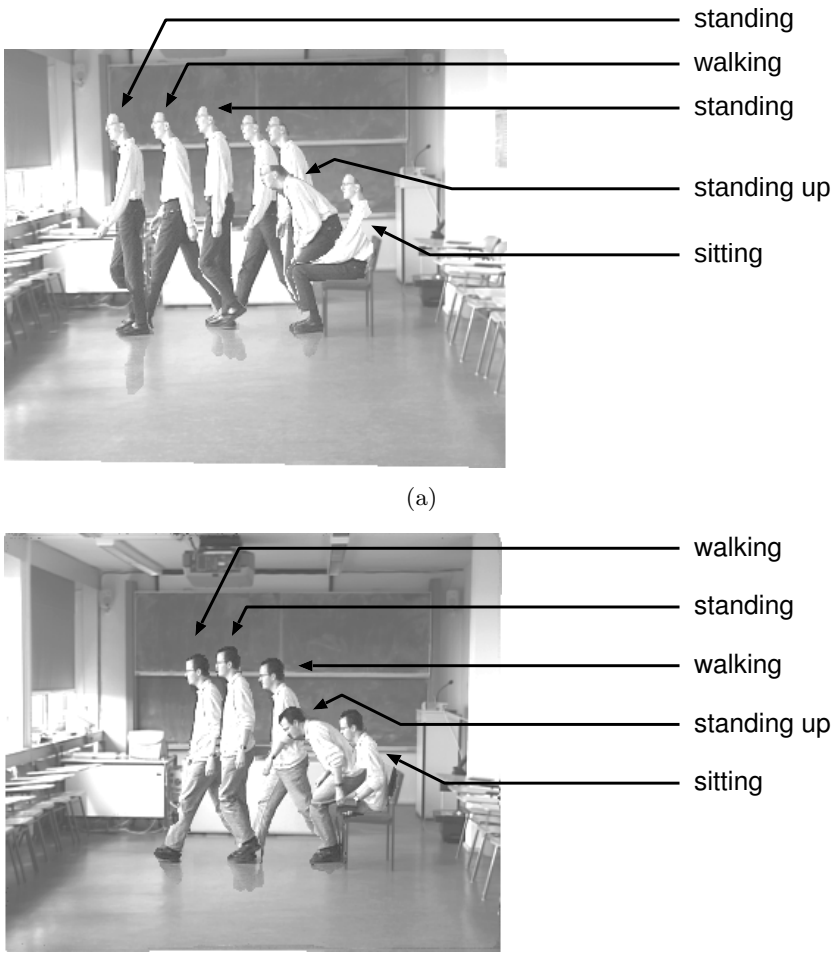

Figure F.4: *Example results for automatic classification of object behaviour for the human-motion model of Fig. F.2.*