Task-Based Control of Articulated Human Pose Detection for OpenVL

Georgii Oleinikov[1], Gregor Miller[2], James J. Little[1] and Sidney Fels[2]
[1]Department of Computer Science, [2]Department of Electrical and Computer Engineering
University of British Columbia, Vancouver, Canada
{olgeorge,little}@cs.ubc.ca, {gregor,ssfels}@ece.ubc.ca

Abstract

Human pose detection is the foundation for many applications, particularly those using gestures as part of a natural user interface. We introduce a novel task-based control method for human pose detection, encoding specialist knowledge in a descriptive abstraction for application by non-experts, such as developers, artists and students. The abstraction hides the details of a set of algorithms which specialise either in different estimations of pose (e.g. articulated, body part) or under different conditions (e.g. occlusion, clutter). Users describe the conditions of their problem, which is used to select the most suitable algorithm (and automatically set up the parameters). The task-based control is evaluated with images described using the abstraction. Expected outcomes are compared to results and demonstrate that describing the conditions is sufficient to allow the abstraction to produce the required result.

1. Introduction

Human pose estimation is motivated by many applications, such as natural user interfaces, augmented reality and person recognition. It is a challenging problem, with active research on various issues: full body, upper-body and body part detection; articulated pose estimation; and extensions to include temporal data or estimate in 3D.

Commercially available solutions constrain the problem using depth images in conjunction with colour, such as Microsoft's Kinect$^{\text{TM}}$[1], the Leap Motion[2] and GestureTek's products[3]. Using depth more easily enables real-time processing with a sufficiently accurate pose estimation but requires specialised devices to compute depth (through active sensing, stereo vision or both). The methods used are typically hidden behind an application programming interface (API) which is specific to the expected constraints of the system, and not portable to other devices or methods.

Extracting a pose from a 2D colour or intensity image remains the most challenging problem; various methods have been recently developed which perform well on certain classes of images [27, 23, 26] (although not necessarily in real-time). While these methods are capable of solving the pose detection problem in limited scenarios, they are generally inaccessible to those outside of computer vision since expert knowledge is required to ascertain in which scenarios each method works, and to tune the many parameters to receive the best result. To provide access to non-experts such as developers, artists and students we need to provide a higher-level abstraction which hides the algorithmic detail.

The contribution of this paper is a task-based control method as an abstraction of human pose detection, to hide the details of specific methods and their configuration. The abstraction may be used to describe the type of pose detection problem they are trying to solve, and our novel interpreter uses the description to select an appropriate algorithm (with parameters derived from the description).

There are many important reasons for designing a higher-level abstraction: 1) Users can focus on the application of pose detection, rather than the algorithms; 2) Advances in the state-of-the-art can be incorporated without re-implementation; 3) Hardware acceleration may be used transparently; 4) The limitations of a particular platform can be taken into account automatically e.g. mobile devices may require a set of low-power consumption algorithms. If an abstraction is used to access detection methods, hardware and software developers of the underlying mechanisms are free to continually optimise and add new algorithms.

This idea has been applied successfully in many other fields, notably graphics with OpenGL, and is the main goal of OpenVL[18] for computer vision. OpenVL is an abstraction framework which hides algorithmic detail and provides mainstream developers with access to sophisticated vision methods, such as face detection[11] and segmentation[19]. The work presented here applies a similar methodology to construct a task-based description of pose estimation at a low enough level to maintain flexibility but high enough for mainstream developers to apply successfully.

[1]http://www.microsoft.com/en-us/kinectforwindows
[2]http://www.leapmotion.com
[3]http://www.gesturetek.com

2. Related Work

There is a long history of developing abstractions and frameworks to help create vision systems. Matsuyama and Hwang introduced *SIGMA*, a three-tier expert system using geometric reasoning based on scene context to select an appearance model, which is subsequently used to extract the relevant image object [17]. The system had a narrow focus on object detection from a learned appearance model. Kohl and Mundy developed the *Image Understanding Environment* to provide high-level access to vision methods through an object-oriented interface, although it required knowledge of how each algorithm operated [13]. Firschein and Strat created *RADIUS*, which employed geometric models defined and manipulated by the user to help guide the choice of image processing algorithms [9]. This operated at a higher level than the previous frameworks by directly wrapping specific algorithms, although this limited its application outside the narrowly-defined scope. Clouard *et al*. used a knowledge-based system to create tools for a domain expert (e.g. biology, geography etc.) to define their problem and for a vision expert to solve it [4]. A rich library of tools would gradually develop, allowing re-use by vision experts, although this system always required two operators.

Konstantinides and Rasure used a visual programming metaphor in *Khoros* for users to create vision applications by connecting components in a data flow [14]. Example components included colour conversion, feature extraction and spatial filtering: the low-level of these units required significant expertise to apply in a vision system. More recently Chiu and Raskar developed *Vision on Tap*, a web-based vision system toolkit aimed at web developers [3]. While this presents a high-level abstraction for developers (the system is resource and system agnostic) it still requires knowledge of vision algorithms to apply effectively and is limited by access to resources available to the web browser. Declarative programming languages such as *ShapeLogic*[4] and *FVision* [21] provide vision functionality as small, connected units, although again these present at a low level, requiring specialist knowledge to implement robust systems.

Many openly available libraries, such as OpenCV [2], FastCV[5] and the Vision Toolbox[6], provide common vision functionality. Recently OpenTL [20] has been developed specifically for tracking in real-world scenarios. These all provide vision components and algorithms without any context of how and when they should be applied, and therefore require expert knowledge for effective use.

Miller and Fels introduced a task-based methodology targeting a general vision abstraction with *OpenVL* [18]; they describe generally how various problems such as segmentation, correspondence and registration can be described at a task level and mapped to underlying algorithms to provide the result. The work we present is part of OpenVL and has been integrated into the OpenVL framework, to help progress towards a more general solution with wide coverage of computer vision problems.

While we do not claim a novel contribution in regard to the technical aspect of obtaining a human pose, we present a brief survey of the state-of-the-art as a basis for the abstraction and to help justify our choice of included methods.

The human pose detection problem has seen the most success when utilizing depth images in conjunction with colour: real-time estimation of 3D body joints and pixel-wise body part labelling is possible, based on randomized decision forests [24]; real-time head pose estimation is also possible using random regression forests [6]. We recognize that a full-featured task-based control would need to support depth imagery, although we have focussed on single colour image solutions for our abstraction. Yu *et al*. introduced a method for monocular 3D pose estimation from video using action detection on top of 2D deformable part models [28]. Amin *et al*. derived a 3D pose estimation from multiple calibrated cameras, incorporating evidence from every camera obtained with 2D pictorial structures [1]. Although estimating a 3D pose solely from 2D image is a severely under-constrained problem, it has been attempted by jointly solving 2D detection and 3D inference problems [25].

The problem of 2D body pose estimation has been traditionally approached with variations of a pictorial structures framework [8]. Recently, a flexible mixture of parts model (FMP) was introduced in [27], which extended deformable parts model (DPM) [7] for articulated 2D body pose estimation which was further improved using a compositional and-or graph grammar model [23]. Pixel-wise body-part labellings were obtained by combining part-based and pixel-based approaches in a single optimization framework [15]. Hara and Chellappa introduced a real-time 2D pose estimator with the help of multi-dimensional output regressors along the body part dependency paths [10]. Wang *et al*. recently improved the FMP pose estimation performance for video by incorporating additional segmentation cues and temporal constraints [26].

The variety of algorithms tailored to various input conditions and output requirements presents a challenge for non-expert users to choose the optimal method for their particular application. Our novel contribution is an interface to describe the conditions of pose detection and a method to map the description onto an algorithm and its parameters.

3. A Task-Based Description of Human Pose

The current state-of-the-art in *access* to pose detection methods is either to use the methods themselves directly or

[4]http://www.shapelogic.org

[5]http://developer.qualcomm.com/mobile-development/mobile-technologies/computer-vision-fastcv

[6]http://www.mathworks.com/products/computer-vision

to employ an API designed for specific devices (as mentioned in the introduction). Non-experts therefore have three problems: first, no single method exists which detects human pose in all circumstances; second, most methods' circumstances are not readily understandable; and finally, the interface to most methods will often require an understanding of the parameters. We argue that the solution to this problem is via a description of the *task*. We define a task as a combination of input type, variables which can affect the result (the conditions) and the output data description. This provides the necessary context of the problem and the result required – sufficient information to abstract the algorithmic detail away from the user.

Therefore we provide an abstraction of the human pose detection problem using a task-based description: users provide a description through our interface, then the method most likely to succeed in those conditions is chosen and its parameters derived from the description. Furthermore, if our abstraction covers enough of the problem space then new methods can be seamlessly added without changing the interface, providing users with continuous updates to the state-of-the-art without changing their description.

The description is based on four categories: input, image, target and output requirements. A summary of these is presented in Table 1, along with the set of algorithms we use to satisfy a large volume of the problem condition space. Users must provide as accurate as possible a description of the *expected* circumstances in the image and target, along with precisely the output format they require.

3.1. Input Type and Image Description

Our abstraction focuses on the general class of pose detection involving single-view colour images, and so the only control we need is to know if the input consists of one or more frames. This allows us to choose a method (such as FMP-Video [26]) when multiple frames are to be processed. Temporal arrangements represent video sequences and contain data such as frame rate and whether the video is being streamed or is fully available at once. If another method better suits the general problem conditions we can do a per-frame analysis instead, although it may not provide as reliable results.

The image description has a more direct impact on the choice of method, since each behaves differently under conditions such as clutter and occlusion. We loosely define the *clutter* control as how many features would likely be found in regions of the image not belonging to a *target* (the persons for whom the user requires a pose estimate, described in detail below), however we believe this would not be well understood by our intended demographic. Instead we employ a segmentation approach: users describe what kind of result they would expect from a good over-segmentation of the *scene*. The scene is defined as all parts of the image which do not contain the targets (i.e. the scene and targets are mutually exclusive based on image area). This involves specifying what varies across the scene (e.g. colour, texture, intensity), how much it varies (the distinctiveness of each segment) and perhaps the size or quantity of segments expected. (This builds on previous work by Miller *et al*. that created a task-based abstraction for segmentation [19].) The general *appearance* of the scene can also be derived from this segmentation description – this is used to compare against the target appearance (discussed in the following section). For problems which perhaps do not require the in-depth description, users may use a three-level scale: Low, $Medium$ and $High$ (hereafter referred to as the LMH scale). The concept of clutter would still be defined using segmentation (in documentation, for example) to ensure users' intuition is developed based on the correct principle.

The *occlusion* control is provided to let users describe the level of expected occlusion of human bodies by elements in the scene, for example someone partially occluded by a desk, and is specified using the LMH scale. The final control for the image description is *population*, through which the user specifies how many people are expected to be in the scene. This is included to differentiate methods that: fail in the presence of more than one human; can only detect one human from many and it is not possible to provide prior information to detect the 'correct' one; can detect more than one person but a specific quantity (e.g. 1) has been requested - these can be selected based on target appearance (described below). The last point highlights the difference between *population* (image description) and *quantity* (target description): the population is the expected number of people in the scene, the quantity is for how many of those people the user requires pose estimates.

3.2. Target Description

The target description allows users to encode their prior knowledge of the conditions of bodies in the images; these form the *expectation* of the object to be detected. The first control in this set is *appearance*, and is described in the same way as scene appearance (previous section) through a description of segmentation. Target appearance is used to establish any distinguishing properties with the rest of the image (the *scene*), which may help with image preprocessing, evaluation of clutter, selecting the correct target from a set of targets, or providing prior information (such as known clothing colour) to aid in the detection process.

The next few controls are as follows: *self-occlusion* describes how much each person is expected to occlude themselves; *quantity* is the required number of detected bodies to return; *size* is the target size, provided by the user as a ratio of image width and converted by the abstraction into a pixel value (for algorithm compatibility). *3D orientation* is

Table 1. The task controls are presented in the first two columns, followed by the algorithms used in our proof-of-concept abstraction. The level of satisfaction for each control per algorithm forms the basis for algorithm selection based on the user-supplied description. Note: FB = Full Body; UB = Upper Body; LB= Lower Body; px = pixels

	Controls	Grammar [23]	FMP FB [27]	FMP UB [27]	FMP-Video [26]	Head/Torso Yaw [16]	Face Yaw [29]
Input Type:	Video	✗	✗	✗	✓	✗	✗
Image Description:	Clutter	Low	$[Med, High]$	$[Med, High]$	$[Med, High]$		$Invariant$
	Occlusion	Low	Med	Med	Med	$High$	Low
	Appearance	-	-	-	-	-	-
	Population	1	> 1	> 1	1	> 1	> 1
Target Description:	Appearance	-	-	-	-	-	-
	Self-Occlusion	$[Low, Med]$	$[Med, High]$	$[Med, High]$	$[Med, High]$	$[Med, High]$	$[Med, High]$
	Quantity	1	$>= 1$	$>= 1$	1	$>= 1$	$>= 1$
	Size	$[80px, 300px]$	$[50px, 500px]$	$> 300px$	$[50px, 500px]$	$Invariant$	$> 80px$
	3D Orientation	✗	✗	✗	✗	✗	$[-90°, 90°]$
	Visibility	✗	LB Invisible	LB Invisible	LB Invisible	✗	✗
	Pose	All	Regular	Front-Facing	Regular	All	$[-90°, 90°]$
Output Requirements:	Skeleton	FB	FB	UB	FB	✗	Head
	Accuracy/Speed	Image Scale	✗	✗	Smoothness	Direct	Direct
	Joint Orientation	✗	✗	✗	✗	✓	✓

a prior users can supply if they are looking for a particular orientation (although few methods currently support this).

The *visibility* control is slightly more subtle than the others: it lets the user describe which body parts may not be visible in the target image, but which a guess at their location would be preferred. For example, a user may just want an upper-body detection, which can be requested using the output requirements (outlined in the next section). On the other hand, using the visibility control a user can request a full-body detection in all cases, and if the lower body happens to be hidden the framework can return a best guess (joints are labelled 'inferred' in this case).

The final control for this set is *pose*, to describe the type which is important for their problem. A set of pre-defined options are provided for this control: *Front-Facing*, indicating the person is facing towards the view which captured them; *Regular*, implying the pose is a general and 'natural' posture (which would work best with methods which have physical constraints or over-fit the model); and *All*, which works with methods that use image evidence to establish pose (possibly at the expense of accuracy in the presence of occlusion). This control may also be used as a prior, for instance in the situation where the orientation of a face is expected within a certain range. In general, the more precise a description provided, the better the result.

Furthermore, it is possible to set pose priors for individual sets of body parts – the user could require a set of body parts (details below) and then indicate visibility, size and pose for these sets. This may be useful in cases such as gesture systems which only need to know the location of arm and shoulder joints.

3.3. Output Requirements

The final description set is the output requirements, which let the user specify exactly what they want as a result. The central control of this set is the *skeleton*, with a relatively small 15-joint model which can be extracted from various methods which all use a higher or equal number of joints. Each joint in the skeleton is individually selectable to allow users to request only the joints which matter to their problem. Presets such as *Full-Body*, *Upper-Body*, *Head+Torso* and *Head* are provided to simplify the most common requests. When the skeleton is returned, each joint is labelled with a general confidence (inspired by the Microsoft Kinect^{TM}API): *Detected*, *Inferred* or *Not Detected* (we support a scalar quantity in the interval $[0, 1]$, although most methods do not currently support this and it may be more than is needed by general users). The set of joints requested plays a large role in the selection of method to provide a result. The *joint orientation* control may also be used to request an orientation (generally, or within a range).

The final control is the practical matter of the *accuracy vs. speed* trade-off: there are various methods which can be used to generally decrease computation time, such as downsampling images, however we are more interested in parameters or controls offered by the algorithms for this purpose. Most tend to provide search space limitations (e.g. scale), which can be trivially derived from the description; others offer constraints such as detection smoothness (over time), or direct controls into the methods for accuracy or speed. When methods don't provide any simple method of optimizing for one or the other, they are chosen based on their known behaviour (e.g. FMP is not chosen for speed if this is a prime factor).

4. Mapping from Task to Algorithm

Based on the previous section's outline of the abstraction used for articulated human pose detection, we now present the proof-of-concept framework designed to demonstrate the utility of the abstraction.

We consider four algorithms for 2D body pose estimation and two for head/face orientation estimation: a combined segmentation and grammar-based model [23]; FMP for upper body [27]; FMP for full body [27]; temporally-consistent full-body FMP [26]; face orientation estimation [29]; and head+torso orientation prediction algorithm from [16]. We selected these algorithms based on their coverage of the problem space and their performance (speed or accuracy). All methods operate on colour images (single, or multiple from the same view) and provide the 2D joint locations and body part size; none provide body part masks or body shape.

4.1. Task Conditions

We provide a map from the task conditions (combination of input type, image description, target description and output requirements) to an algorithm. The algorithms incorporated into the current system are mostly HOG-based 2D body part estimators working on single colour images. In order to select the appropriate algorithm for a specific task, the system searches the task conditions matrix (Table 1) for all methods that would satisfy the input type, image description, etc.; that is, for methods that are able to take as an input the specified type of data and infer the required information. Our framework determines to which algorithm specifications the current task description belongs, and selects the corresponding algorithms. If no algorithm covers the provided description, the closest algorithm is chosen, defined using the shortest Euclidean distance from the point in multi-dimensional description space given by the user to each volume in the same space defined by the acceptable ranges of conditions for each algorithm. Furthermore, the system chooses the fastest method among the selected algorithms (for a given accuracy, unless the user has requested a specific speed or precision). Finally, it adjusts the algorithm's parameters according to the description and executes it to retrieve the result.

4.2. Parameter Derivation

Task conditions are also used to derive appropriate parameters for the chosen algorithm. Parameters of some algorithms are learned and fixed in the model, while others allow their customization based on the user's needs. The grammar-based model [23] requires scale parameters to be specified, which is set based on the required target size and the requested speed requirements. FMP is fully automated, but its temporally consistent version allows an adjustment of smoothing and tracking levels. Larger smoothing increases the computation time, which is taken into account when attempting to satisfy the speed requirement. Head+torso and face orientation estimation methods [16, 29] both provide a parameter to control speed over accuracy, which is also set by the current system automatically. Furthermore, the face estimation algorithm comes with three pre-trained models of different level of detail and computation speed, which is likewise set by a current system according to speed requirements.

5. Evaluation of the Task Description

Some values in the conditions matrix follow directly from reported algorithm capabilities, while others require conducting a set of experiments. In order to determine the performance of our chosen algorithms in various conditions, we selected 120 images from four pose estimation datasets: Buffy Stickmen [8], Image Parse [22], Leeds Pose Dataset [12] and Synchronic Activities Stickmen [5]. We selected the images based on the maximum coverage of the task description problem space. We manually annotated each image with: (1) the quantity of clutter as a measure of target colour and texture distinctiveness from the background; (2) amount of occlusion; (3) lower body visibility; (4) target size; (5) pose label – *Regular*, *Front-Facing* and *Unusual* (in the abstraction, *All* is equal to *Regular+Unusual*). Non-vertical or highly articulated body configurations were labelled as *Unusual*; roughly facing the camera as *Front-Facing*; all others as *Regular*.

5.1. Pose Estimation Evaluation

Since the grammar-based method works only on single person images, for a fair comparison we cropped images such that they each contained a single person only. FMP frequently omits some people when they partially occlude each other, mostly due to non-maximum suppression, and also results in a higher rate of false positives in high resolution images. Therefore, in real-world tasks we would also use a person detector prior to running pose estimators, whenever the speed constraint allows.

We executed grammar-based and both upper-body and full-body versions of FMP on 70 images out of 120 selected and filled the task conditions matrix based on obtained evidence on performance of the algorithms. We found that for full body pose estimation, FMP [27] is the algorithm of choice, as in general it is more robust to clutter. (The grammar-based method [23] is more likely to pick clutter as a body part than FMP.) However, FMP often double counts single legs and is more likely to miss a body part in an uncluttered environment than the grammar-based method.

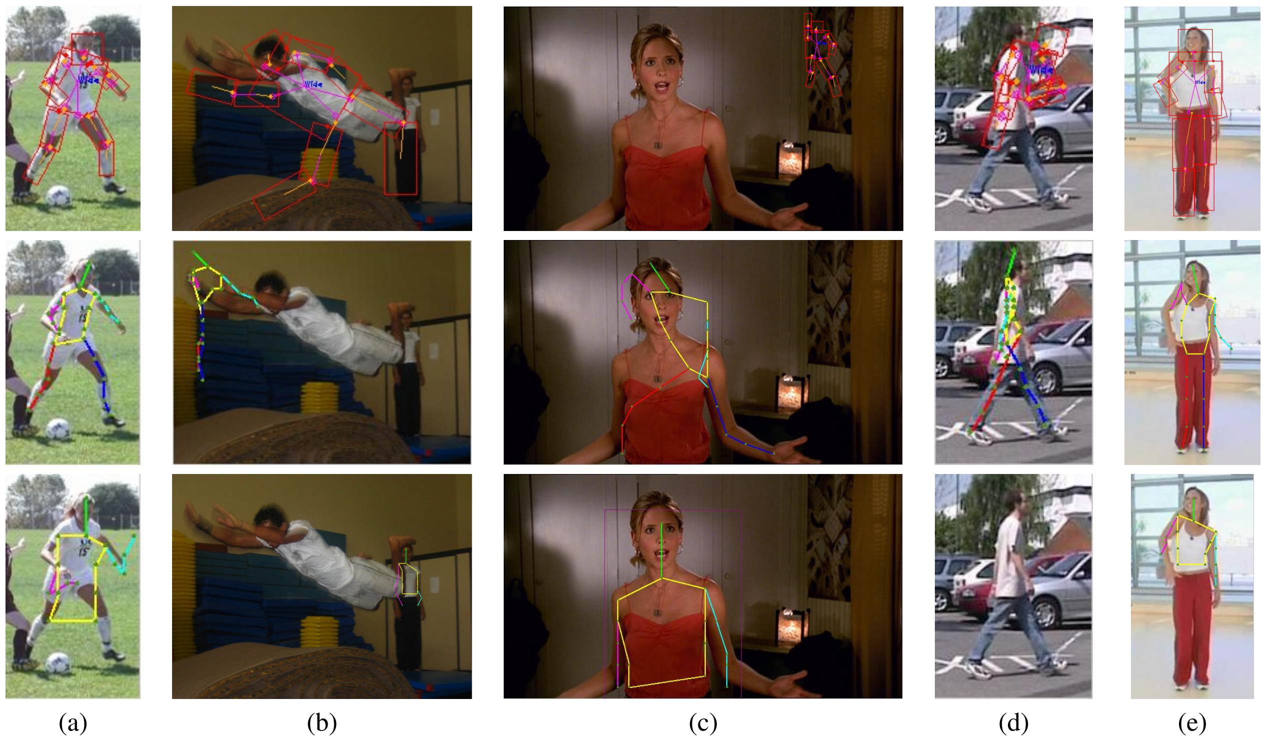

Figure 1. Examples of scene conditions, with results from three algorithms, from top to bottom: Grammar-based [23], FMP [27], upper-body FMP [27]. (a) Regular Pose (b) Unusual Pose (c) Lower Body is invisible (d) High Clutter (e) Low Clutter and Large Size

Furthermore, FMP seems to have stronger pose priors that do not favour large hand motions or non-upright torso positions. In this regard the grammar-based method seems to be more flexible. Therefore, when pose priors indicate that the pose is likely to be unusual, the latter method is preferable.

In order to evaluate the task-based algorithm mapping, we ran three pose estimators on the remaining 50 images and determined the system success rate, which was defined by how often the best algorithm for a specific task is selected. The success rate was 76%, and in cases where it picked a non-optimal method the difference in performance between the best and selected algorithms was approximately 15% (therefore the result would be relatively close to optimal). This supports our insights in the performance of the algorithms and shows that the system selects pose estimation algorithms reasonably well. The images in Figure 2 demonstrate the power of the task-based abstraction: given a description, an algorithm is selected and provides a result (highlighted in green); the qualitatively-determined best result is highlighted in blue. As can be seen, in most cases the best result is chosen by our abstraction, although in all cases the result is generally acceptable.

6. Conclusion

The core of this work is intended to demonstrate that a wide-coverage abstraction over articulated human pose detection for use by non-experts is possible, and using a task-based methodology provides a reasonable level of control while still removing the complexity of individual methods and their parameters. Our novel task-based control method utilizes descriptions of input, image, target and output to cover a large volume of the pose detection problem space. The conditions table maps the description to a suitable method, parameters are derived from the description and a result returned. Our results demonstrate the advantages of this methodology, with a clear link from the task description to the quality of result.

Various challenges remain to be solved: we would like to provide more access to the many parameters of the detection methods, but these tend to be set while training and not in use of the model. Therefore using the abstraction as a mechanism to allow users to train their own detectors would greatly increase the flexibility and power of the method. Once an automatically learned conditions table is created (to map descriptors to method capabilities) we can start to combine methods to achieve higher accuracy or new features, such as head orientation plus articulated pose. We are also using a very simple joint model, which is not sufficient for other more accurate systems such as those using depth imaging. The abstraction itself can use a much higher accuracy and downsample based on the result it is capable of acquiring in the current conditions.

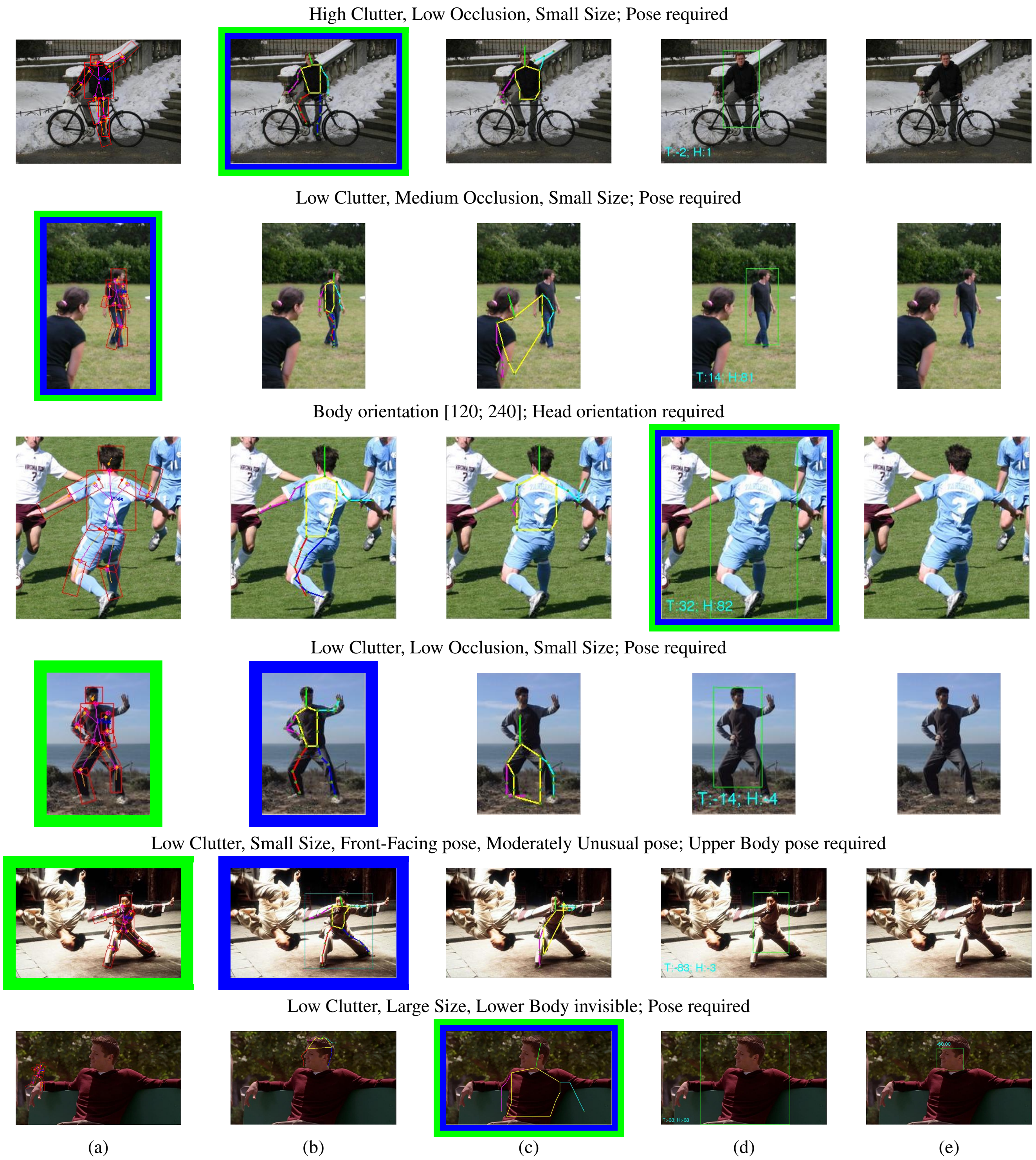
High Clutter, Low Occlusion, Small Size; Pose required
T:-2; H:1
Low Clutter, Medium Occlusion, Small Size; Pose required
T:14; H:81
Body orientation [120; 240]; Head orientation required
T:32; H:82
Low Clutter, Low Occlusion, Small Size; Pose required
T:-14; H:-4
Low Clutter, Small Size, Front-Facing pose, Moderately Unusual pose; Upper Body pose required
Low Clutter, Large Size, Lower Body invisible; Pose required
(a)
(b)
(c)
(d)
(e)

Figure 2. This is a sample of the description/image combinations used to evaluate the abstraction. Images with various descriptions are fed into the system, which selects an algorithm and produces an output (highlighted in green). We also show results from the other methods: the algorithm which returns the best result is highlighted in blue. In most cases the task-based control returns the best result, and in cases where it does not the result is generally acceptable. Results from 5 algorithms are provided for comparison, in the same order as presented in Table 1 (FMP-Video excluded). Yaw estimates in (d) and (e) are noted on the left-bottom of images. Note that images with no annotations on them are results of mis-detections, which indicate algorithm failure.

References

[1] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele. Multi-view pictorial structures for 3D human pose estimation. In *British Machine Vision Conference*, 2013. 2

[2] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, October 2008. 2

[3] K. Chiu and R. Raskar. Computer vision on tap. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 31–38, 2009. 2

[4] R. Clouard, A. Elmoataz, C. Porquet, and M. Revenu. Borg: A knowledge-based system for automatic generation of image processing programs. *Transactions on Pattern Analysis and Machine Intelligence*, 21:128–144, February 1999. 2

[5] M. Eichner and V. Ferrari. Human pose co-estimation and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2282–2288, Nov. 2012. 5

[6] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 617–624, June 2011. 2

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 2

[8] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 2, 5

[9] O. Firschein and T. M. Strat. *RADIUS: Image Understanding For Imagery Intelligence*. Morgan Kaufmann, 1st edition, May 1997. 2

[10] K. Hara and R. Chellappa. Computationally efficient regression on a dependency graph for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3390–3397, June 2013. 2

[11] D. Jang, G. Miller, S. Fels, and S. Oldridge. User oriented language model for face detection. In *Proceedings of the 1st Workshop on Person-Oriented Vision (POV)*, WVM'11, pages 21–26, New York City, New York, U.S.A., January 2011. IEEE. 1

[12] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *British Machine Vision Conference*, 2010. 5

[13] C. Kohl and J. Mundy. The development of the image understanding environment. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, CVPR'94, pages 443–447, Los Alamitos, California, U.S.A., June 1994. IEEE Computer Society Press. 2

[14] K. Konstantinides and J. R. Rasure. The Khoros software development environment for image and signal processing. *IEEE Trans. on Image Processing*, 3:243–252, 1994. 2

[15] L. Ladicky, P. Torr, and A. Zisserman. Human pose estimation using a joint pixel-wise and part-wise formulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3585, June 2013. 2

[16] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3177–3184, 2011. 4, 5

[17] T. Matsuyama and V. Hwang. SIGMA: a framework for image understanding integration of bottom-up and top-down analyses. In *Proceedings of the 9th international joint conference on Artificial intelligence*, volume 2, pages 908–915. Morgan Kaufmann Publishers Inc., 1985. 2

[18] G. Miller and S. Fels. OpenVL: A task-based abstraction for developer-friendly computer vision. In *Proceedings of the 13th IEEE Workshop on the Applications of Computer Vision (WACV)*, WVM'13, pages 288–295, New York City, New York, U.S.A., January 2013. IEEE. 1, 2

[19] G. Miller, D. Jang, and S. Fels. Developer-friendly segmentation using OpenVL, a high-level task-based abstraction. In *Proceedings of the 1st IEEE Workshop on User-Centred Computer Vision (UCCV)*, WVM'13, pages 31–36, New York City, New York, U.S.A., January 2013. IEEE. 1, 3

[20] G. Panin. *Model-based Visual Tracking: the OpenTL Framework*. John Wiley and Sons, 1st edition, 2011. 2

[21] J. Peterson, P. Hudak, A. Reid, and G. D. Hager. Fvision: A declarative language for visual tracking. In *Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages*, PADL '01, pages 304–321, London, UK, 2001. Springer-Verlag. 2

[22] D. Ramanan. Learning to parse images of articulated bodies. *Advances in Neural Information Processing Systems*, 2006. 5

[23] B. Rothrock, S. Park, and S.-C. Zhu. Integrating grammar and segmentation for human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3221, June 2013. 1, 2, 4, 5, 6

[24] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2011. 2

[25] E. Simo-Serra, A. Quattoni, C. Torras, and F. Moreno-Noguer. A joint model for 2D and 3D pose estimation from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 2

[26] C. Wang, Y. Wang, and A. Yuille. An approach to pose-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 915–922, June 2013. 1, 2, 3, 4, 5

[27] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 1, 2, 4, 5, 6

[28] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained monocular 3D human pose estimation by action detection and cross-modality regression forest. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013. 2

[29] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, 2012. 4, 5