Gamesourcing to Acquire Labeled Human Pose Estimation Data

Richard Souvenir, Ayman Hajja, Scott Spurlock
University of North Carolina at Charlotte
Charlotte, NC 28223 USA
{souvenir,ahajja,sspurloc}@uncc.edu

Abstract

In this paper, we present a gamesourcing method for automatically and rapidly acquiring labeled images of human poses to obtain ground truth data as input for human pose estimation from 2D images. Typically, these datasets are constructed manually through a tedious process of clicking on joint locations in images. By using a low-cost RGBD sensor, we capture synchronized, registered images, depth maps, and skeletons of users playing a movement-based game and automatically filter the data to keep a subset of unique poses. Using a recently-developed, learning-based human pose estimation method, we demonstrate how data collected in this manner is as suitable for use as training data as existing, manually-constructed data sets.

1. Introduction

The recent emergence of low-cost, RGBD sensors ignited a flurry of projects that took advantage of real-time, vision-based depth sensing for gaming, gesture-based control, and virtual reality. RGBD sensors have simplified difficult computer vision problems, such as person detection and pose estimation, by sidestepping the depth ambiguity inherent in flat images. However, there is still a need to solve these types of problems from 2D imagery. In this paper, we present an approach for using RGBD sensors to rapidly acquire annotated training data for human pose estimation from 2D images. We take advantage of the fact that these sensors are commonly used for gaming to tap into a diverse pool of willing participants. With this approach, large volumes of labeled data can be obtained more quickly and efficiently than the traditional approach of manual annotation.

In recent years, many recognition problems in computer vision have been formulated as supervised learning problems. Getting labeled examples to develop robust learners usually involves tedious manual effort on thousands (or more) images. Recently, the computer vision community has embraced the idea of "gamesourcing" labeled data. To annotate images for object recognition, the Google Image Labeler and the ESP game [15] allow an anonymous pair of players to suggest descriptive terms for unlabeled images. Peekaboom [16] allows two players to localize objects in a scene by identifying and labeling regions of images using terms previously collected from the ESP game. PhotoSlap [7] is a game for face recognition that lets players match the same person in different images. These methods have been very useful for rapidly acquiring new sources of training data for a variety of supervised learning tasks.

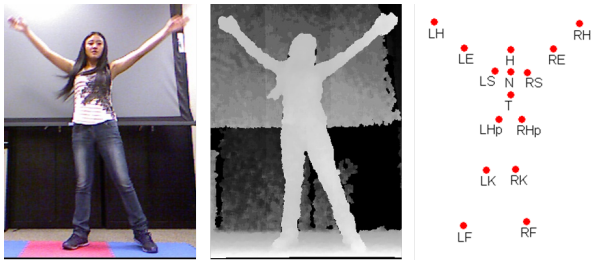


Figure 1. We collect annotated images from motion-based games as training data for pose estimation from 2D images. The Microsoft Kinect outputs (from L-R) an RGB image, depth map, and joint locations.

The main contribution of this work is an automated approach for capturing labeled data for human pose estimation through gamesourcing. This will provide a new avenue for obtaining training and testing data for pose estimation algorithms, avoiding tedious manual annotation.

2. Data Collection

Pose estimation methods that incorporate depth information far outperform their 2D analogs. In fact, it is the accuracy of 3D pose estimation that we will leverage to generate labeled data for the 2D problem. The Kinect employs fast supervised learning algorithms for joint position estimation that are based on randomized decision forests [12], which can be quickly evaluated to provide real-time 3D pose estimates. Figure 1 shows an example of output from the Kinect during gameplay: an RGB image, depth map, and joint locations. A typical gaming session generates a large amount

of data, much of it not needed. First, although the depth-based pose estimation algorithms are usually reliable, there can be instances where unusable data is extracted. Second, consecutive frames or images of a player repeating a pose can be quite similar and, therefore, redundant. One approach would be to filter the data after the gaming session. However, as our data collection runs alongside the game on consumer-grade hardware, the I/O overhead of saving this amount of data to disk could negatively impact real-time gameplay. In this section, we describe our methods for real-time filtering of noisy and redundant data.

Obtaining Registered Joint Positions The IR sensor used for depth estimates is not synchronized with the RGB camera. This does not affect gameplay since the RGB data is rarely used; in the few games where video is even displayed, it is not overlaid onto a skeletal model or avatar where the synchronization issues would be noticed. To use this data for training pose estimation algorithms, however, it is important that the joints and images are registered.

In a call to the Kinect data acquisition function, the current RGB image, depth image, and joint positions are returned with individual timestamps (in ms), which we refer to as t_{RGB}, t_{DEP}, and t_{JNT}, respectively. We determined the registration offsets empirically by manually identifying the joint positions in a small set of depth and RGB images and measuring the sum of the squared differences to those returned by the Kinect. Because the joint estimates are derived from the depth images, the timestamps for corresponding depth and RGB images precede the joint data. Alignment error was minimized for depth images when $t_{JNT} - t_{DEP} \approx 33$, and for RGB images when $t_{JNT} - t_{RGB} \in [38, 48]$. On an unloaded system, data can be acquired at 30fps. So, for a discrete data acquisition event, i, we collect an RGB and depth image with timestamps t^i_{RGB} and t^i_{DEP}, respectively. At $t + 1$, we collect joint locations with timestamp t^{i+1}_{JNT}. If the timestamp differences correctly align, the data is stored for further processing. Due to both the asynchronicity between sensors and occasional dropped frames, roughly $\frac{1}{3}$ of the data is registered during a typical gaming session.

Data Filtering Two additional checks are performed on the joint data that has been matched to an image. First, most image-based pose estimation methods extract patches surrounding the joint location. Cases where a joint location is too close to the image border for a patch to be extracted are discarded. Second, in cases of very fast motion or joint self-occlusion, the joint estimates can be erroneous. Using the foreground mask supplied with the depth data, any instances where joints fall outside of the foreground, such as in Figure 2, are also discarded.

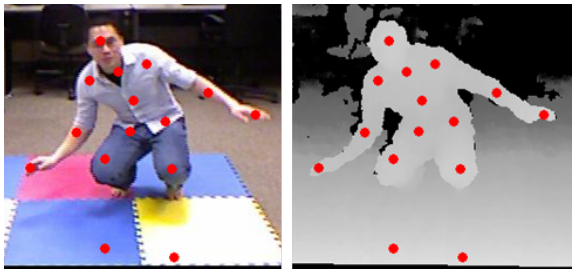


Figure 2. Instances where joints (already registered with images) are near the image border or outside the foreground are discarded.

Skipping Redundant Data For learning methods, similar examples in the training data provide little additional benefit. In our case, this corresponds to images of a player in similar poses, which often occurs in sequential frames or when the player is performing a repeated action in a game. To quickly determine if a similar pose has already been collected, we use locality-sensitive hashing (LSH) [2] for finding nearest neighbors in high-dimensional spaces. Given a query point and a database, LSH discovers the approximate nearest neighbors, using the Euclidean distance, in sublinear (in the number of database elements) time.

LSH projects high-dimensional points onto sets of random vectors under the principle that nearby points (in the ambient space) will project to similar locations more often than more distant points. LSH uses L groups of k hash functions that map a high-dimensional point, $\mathbf{v}$, onto the set of integers. The hash functions, h, take the form: $h(\mathbf{v}) = \lfloor \frac{\mathbf{a}\cdot\mathbf{v}+b}{w} \rfloor$ where (vector) $\mathbf{a}$ and (scalar) b are randomly selected and w is a small integer (4 is the default value). Two points $\mathbf{v}_i$ and $\mathbf{v}_j$ which match all k hash values for any of the L groups are putative nearest neighbors. The Euclidean distance is used to determine whether or not the points meet the threshold, R, to be considered similar. Typically the number of matches is small, so this scheme avoids the linear-time approach of calculating the distance between the query and all database points.

Figure 3 depicts the process of discarding non-unique poses. Let $\mathbf{y}_t = \langle H_t, N_t, \ldots, RF_t \rangle$ be the vector of concatenated pixel locations of the joints (as defined in Figure 1(c)) at time t. To provide invariance to global pose shifts, we apply an affine transformation to each point from canonical image coordinates to a space defined (in the positive y direction) by the unit vector from the torso (T) to the neck (N). Let $\mathbf{y}'_t$ represent the array of transformed points. At each timestep, t, $\mathbf{y}'_t$, is hashed using LSH. If there is a collision with an existing database pose (i.e., the two pose vectors are within R units), the current data is discarded. If not, the pose is added to the database as a unique pose, and the corresponding image is also saved. This process continues until the gaming session ends or resource limitations are reached. In Section 4, we describe LSH parameter

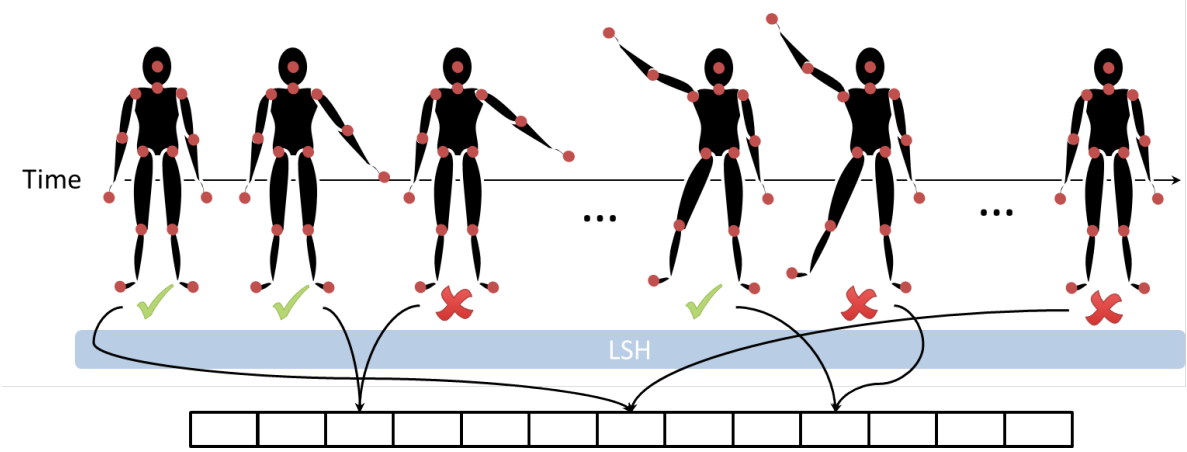


Figure 3. Real-time pose uniqueness check. If a new pose collides with a existing database pose, it is discarded, otherwise it is added to the database. Using LSH, the collision checks occur in sublinear time (in the number of database items).

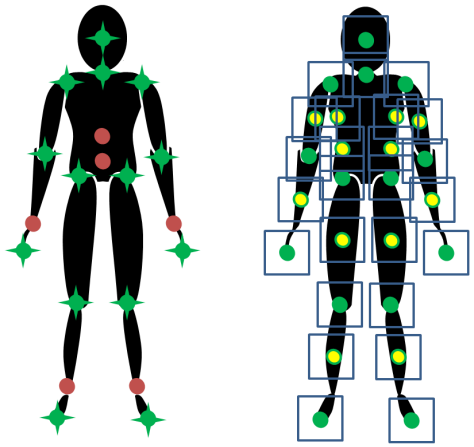
selection.

3. Generating Training Examples

Once the data has been collected, the last step is to transform it to be usable with supervised methods for 2D pose estimation. Recent methods for 2D pose estimation use a discriminative approach that collects statistics on labeled datasets for model building [4, 11, 13]. Methods for pose estimation "by parts" use human body models that differ in the number or type of joints specified. In our experiments, we employ the 14-joint pose estimation model of Yang and Ramanan [17]. Up to 24 joint locations can be obtained from the Kinect.[1] Figure 4(a) shows a mapping of the Kinect joint positions to the input of this model.

The method in [17] uses a mixture of parts to estimate pose in 2D images. Through interpolation, the 14 input joint locations are extended to 26 total locations and, equal-sized images patches are extracted from each location (Figure 4(b)). Each patch is represented using the Histogram of Oriented Gradients (HOG) descriptor, and SVM is used for classification. For pose estimation, instead of searching for body parts individually to find a global configuration, the authors noted that certain configurations between neighboring body parts tend to be over-represented. The limbs are modeled based on the relative positions of parent and child joints, and a compatibility measure is calculated based on the co-occurrences of pairs. In Section 4, we compare the performance of this learning-based method on human pose estimation from 2D images when trained with gamesourced examples or manually-constructed data sets. It should be noted that any pose estimation "by parts" method could have been selected if the Kinect joints could be easily mapped to the model.

[1]The Windows Kinect driver returns 20 locations (head, neck, shoulders, elbows, wrists, hands, torso, waist, hips, knees, ankles, and feet). The OpenNI driver adds four (collars and fingertips).



(a) Joints (b) Image Patches

Figure 4. (a) 20 joints can be obtained from the Kinect (with the Windows SDK); those represented as green stars are the subset used as input in [17]. (b) For joint locations and additional interpolated points (shown in yellow), image patches are extracted.

4. Experiments with Gamesourced Data

To evaluate the use of gamesourced data for 2D pose estimation, we used *Astrojumper* [6], a full-body Kinect-based game developed to motivate exercise. The player controls a flying avatar as different objects speed by. Points are earned by touching certain objects, avoiding others, and "shooting" enemies using punching motions. This version of Astrojumper was modified to use the Kinect on a PC via the Flexible Action and Articulated Skeleton Toolkit (FAAST) [14]. All experiments were carried out on a Windows 7 laptop with a 3.33 GHz CPU and 8GB RAM.

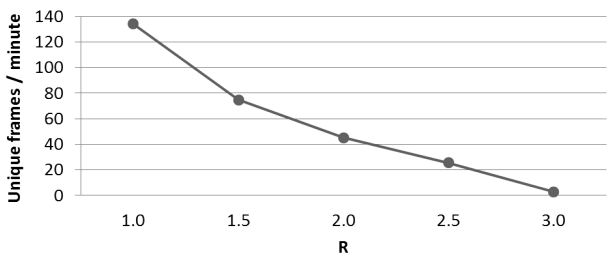


Figure 5. The number of unique images saved per minute as a function of the similarity threshold, R.

LSH Parameters A key step in the real-time processing of the matched joint/image pairs is the uniqueness check using locality-sensitive hashing (LSH). LSH has four free parameters, R (Euclidean distance threshold to be considered a match), k (number of projection vectors), L (number of hash functions), and γ (the probability that a near neighbor is not reported). The E^{2}LSH package provides a method for optimizing k, and L given sample data, R and γ, to minimize the expected query time. For γ, we used the LSH default value of .1 (90% success probability). R implies a trade-off between the amount of data collected and the similiarity of the "unique" poses saved. We tested a range of values for a player engaged in a 4-minute session of Astrojumper. Figure 5 shows the plot of the number of unique images saved per minute as a function of R. Figure 6 shows examples of the closest (in Euclidean distance) poses of the resulting sets of saved images. With $R = 2$, the set of unique poses returned generally appeared to be sufficiently distinct across different users. For the remaining experiments, we use $R = 2$, $\gamma = .1$, $k = 10$, and $L = 55$.

Related Data Sets We compared gamesourced data to recently curated, manually-labeled sets. PARSE [10] consists of 305 outdoor images of people (∼150 pixels tall) mostly playing sports and contains high background clutter and self-occlusions. BUFFY [5] was collected from several TV episodes of "Buffy the Vampire Slayer" and consists of 748 images from (mostly) indoor scenes. People appear at different scales and the background is highly cluttered. PASCAL STICKMEN [3] is a subset of PASCAL VOC 2008 and contains 549 low to medium quality images of people mainly standing. Similar to BUFFY, only upper body annotations are provided. HUMANS IN 3D (H3D) [1] contains 1240 high quality images of people with 3D joint positions. The LEEDS Sports Pose [8] dataset consists of 2,000 annotated images (recently extended to 10,000 [9]) of people (∼150 pixels tall) performing sports activities.

Table 1 provides a summary of manually constructed data sets compared to gamesourced data. Compared to these data sets, the main advantage of gamesourcing is scalability and flexibility. RGBD sensors for gaming are used in a wide variety of environments by a wide variety of people of various sizes and body styles. These sensors work both indoors and outdoors and result in images of reasonable quality with more joint estimates than current manual approaches.

Figure 6. Examples of the closest unique poses from a 4-minute gaming session. At $R = 1$, the poses are similar and would be redundant for training. At $R = 3$, only 12 unique poses were saved, so interesting poses were likely missed.

Data Set	Images	Scene	Body	Joints
PARSE	305	Outdoor	Full	14
BUFFY	748	Indoor	Upper	12
STICKMEN	549	Indoor	Upper	14
H3D	1240	Both	Full	20
LEEDS Sports	10,000	Outdoor	Full	14
Gamesourced	**-**	**Both**	**Full**	**24**

Table 1. Comparison of pose estimation datasets.

(a) ASTRO (b) PARSE

(c) LEEDS (d) BUFFY

Figure 7. Images from the data sets used in the experiments.

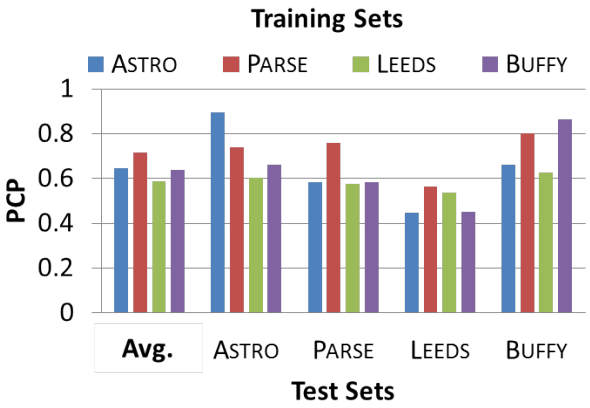


Figure 8. Pose estimation results using a learning algorithm [17] trained and tested with images from different data sources. PCP is the percent of joints close to the ground truth. Higher is better.

2D Pose Estimation ASTRO is a gamesourced dataset using 15 different people playing Astrojumper for roughly four minutes each. For ASTRO and three of the manually-constructed data sets (PARSE, BUFFY, LEEDS), we randomly selected 300 images each, split evenly into training and testing sets. Using Yang and Ramanan's algorithm as the learning method, we performed 16 pairwise pose estimation experiments with each of the four training and testing sets. Figure 8 shows the results for these tests, reported as Percentage of Correctly estimated Parts (PCP) [5]. A body part is considered correctly labeled if it lies within 50% of the length of the ground truth body-part segment.[2]

Overall, ASTRO performed as well as the manually curated sets when used as training data. Except for LEEDS, each training set performed best when the test set was drawn from the same source. ASTRO for training and testing resulted in the highest performance over all experiments. This may be explained by the similarity of the scenes (e.g., a few locations on a university campus) and few people, all playing the same game. In contrast, each image in PARSE and LEEDS contains a different person and background. However, the lack of data diversity in ASTRO did not negatively affect the performance when tested on the other sets. Figure 9 shows example results with the estimated pose overlaid. Each row shows a test image used with different training data. The first image, from ASTRO, shows an example where, when trained with the gamesourced data, the algorithm was able to correctly discover a case of self-occlusion. The second image, from LEEDS, shows a simple pose in cluttered a background, that was missed in some cases. The third image, from PARSE, was difficult in all cases.

One difference between the images in each set was the apparent variance of the poses. When used for training, PARSE performed the best overall, and LEEDS performed well on each of the test sets. These data sets contain examples of athletes engaged in complex mid-air maneuvers. To estimate the pose variance, we performed Principal Component Analysis (PCA) on the vectors of joint positions for the 300 images from each set and computed the number of components necessary to represent 95% of the variance. Figure 10 shows the pose variance estimates for the upper, lower, and full body joints for the four datasets.

[2] BUFFY only contains annotations for the upper body, so the corresponding results only include upper body joints.

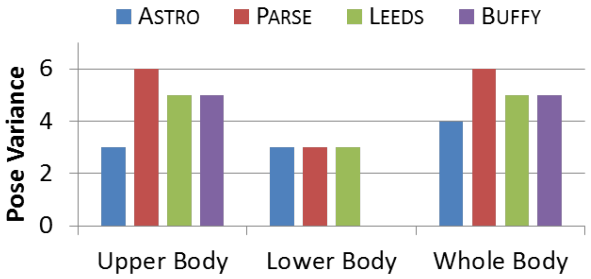


Figure 10. Pose variance, measured as the number of PCA components needed to account for 95% of the data variance.

5. Conclusions and Future Work

We presented a method for collecting labeled data through gamesourcing. Our approach does not involve a game specifically developed to facilitate crowdsourcing; it can be used with existing games. With data collected from only a small number of players in a limited number of settings, we achieved results comparable to training with manually annotated data. While we expect the data diversity to increase when used with more games and users, an inherent limitation is the variety of poses that can be induced by gesture-based games. Even the most enthusiastic gamer is unlikely to execute the type of acrobatic actions performed by athletes, as seen in other image sets.

Future work includes improving exemplar pose selection. For example, when a pose collides with the database, one could use estimates of motion blur to determine which instance to save. Also, comparisons could be added across users that incorporate features to discard matches already represented by previous players. Additionally, beyond passive data collection, the module could be incorporated into games to dynamically adjust the goal or position of objects to compel users to perform underrepresented poses.

References

[1] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision*, 2009. 4

[2] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In

Figure 9. Example results from the experiments summarized in Figure 8. Each row shows example results on the same image with training data from the gamesourced data (ASTRO) or the manually constructed data sets (PARSE, LEEDS, and BUFFY).

Proc. of the Symposium on Computational Geometry, pages 253–262. ACM, 2004. 2

[3] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *British Machine Vision Conf.*, 2009. 4

[4] 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, September 2009. 3

[5] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 4, 5

[6] S. Finkelstein, A. Nickel, Z. Lipps, T. Barnes, Z. Wartell, and E. A. Suma. Astrojumper: Motivating exercise with an immersive virtual reality exergame. *Presence: Teleoperators and Virtual Environments*, 20(1):78–92, 2011. 3

[7] C.-J. Ho, T.-H. Chang, and J. Y. jen Hsu. Photoslap: A multiplayer online game for semantic annotation. In *Proc. of AAAI Conf. on Artificial Intelligence*, pages 1359–1364, 2007. 1

[8] S. Johnson and M. Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*, 2010. 4

[9] S. Johnson and M. Everingham. Learning effective human pose estimation from inaccurate annotation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2011. 4

[10] D. Ramanan. Learning to parse images of articulated bodies. *Advances in Neural Information Processing Systems*, 19:1129, 2007. 4

[11] R. Rosales and S. Sclaroff. Learning body pose via specialized maps. In *NIPS*, pages 1263–1270, 2001. 3

[12] 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 Conf. on Computer Vision and Pattern Recognition*, 2011. 1

[13] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Discriminative density propagation for 3D human motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 390–397, 2005. 3

[14] E. A. Suma, B. Lange, A. Rizzo, D. Krum, and M. Bolas. FAAST: the flexible action and articulated skeleton toolkit. In *IEEE Virtual Reality*, pages 247–248, 2011. 3

[15] L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proc. of Int'l Conf on Human Factors in Computing Systems*, pages 319–326. ACM, 2004. 1

[16] L. Von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proc. of Int'l Conf on Human Factors in Computing Systems*, pages 55–64. ACM, 2006. 1

[17] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011. 3, 5