An Evaluation of Gamesourced Data for Human Pose Estimation

SCOTT SPURLOCK and RICHARD SOUVENIR, University of North Carolina at Charlotte

Gamesourcing has emerged as an approach for rapidly acquiring labeled data for learning-based, computer vision recognition algorithms. In this article, we present an approach for using RGB-D sensors to acquire annotated training data for human pose estimation from 2D images. Unlike other gamesourcing approaches, our method does not require a specific game, but runs alongside any gesture-based game using RGB-D sensors. The automatically generated datasets resulting from this approach contain joint estimates within a few pixel units of manually labeled data, and a gamesourced dataset created using a relatively small number of players, games, and locations performs as well as large-scale, manually annotated datasets when used as training data with recent learning-based human pose estimation methods for 2D images.

Categories and Subject Descriptors: I.4.9 [**Image Processing and Computer Vision**]: Applications; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation*

General Terms: Experimentation, Measurement

Additional Key Words and Phrases: Crowdsourcing, data collection, datasets, Kinect, depth images, automatic annotation, dataset generation

ACM Reference Format:
Scott Spurlock and Richard Souvenir. 2015. An evaluation of gamesourced data for human pose estimation. ACM Trans. Intell. Syst. Technol. 6, 2, Article 19 (March 2015), 16 pages.
DOI: http://dx.doi.org/10.1145/2629465

1. INTRODUCTION

Recently, many recognition problems in computer vision have been formulated as supervised machine-learning problems in which, typically, the most tedious step is acquiring accurately labeled examples for training and testing. For images and video, depending on the problem, this typically involves a human clicking pixel locations or delineating image regions. Amassing and validating the hundreds or thousands of annotated examples needed for modern large-scale learning methods can be a time-consuming process. In recent years, the computer-vision community has embraced the idea of crowdsourcing this aspect of ground-truth annotation to distribute the manual effort to large numbers of nonexpert users. An increasingly popular method of crowdsourcing, known as "gamesourcing," uses video-game achievements as incentive to engage users. In the computer-vision domain, a number of games have been developed to aid in the process of collecting data for problems such as object recognition [Von Ahn and Dabbish 2004], object localization [Von Ahn et al. 2006], and face detection [Ho et al. 2007]. In these cases, games have been designed specifically to be used for gamesourcing data for the respective problems. In this article, we present an approach for using gamesourced data to train and test learning-based human pose estimation algorithms.

Authors' addresses: S. Spurlock and R. Souvenir, Department of Computer Science, 9201 University City Blvd., Charlotte, NC 28223 USA; emails: {sspurloc, souvenir}@uncc.edu.

Our approach takes advantage of the benefits of gamesourcing without the effort of explicitly designing a custom game.

The emergence of low-cost, RGB-D sensors, such as Microsoft's Kinect, ignited a flurry of projects that took advantage of real-time, vision-based depth sensing for gaming, gesture-based control, and virtual reality. RGB-D 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. Our gamesourcing method uses RGB-D sensors to rapidly acquire annotated 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. As with all crowdsourced data, it is important to validate that the data is reliable. In this article, which extends a small pilot on single game with a few users [Souvenir et al. 2012], we describe how we collect and filter data from an RGB-D sensor during gameplay and compare our automatically generated data to manually curated datasets regularly used to train and test learning-based human pose estimation algorithms on 2D images.

2. RELATED WORK

We present an automated approach for capturing labeled data for human pose estimation through gamesourcing, specifically gameplay using an RGB-D sensor. Here, we review related work in gamesourcing for computer vision, Kinect-based datasets, and pose estimation datasets.

2.1. Gamesourcing for Computer Vision

To annotate images for object recognition, the Google Image Labeler and the ESP game [Von Ahn and Dabbish 2004] allow an anonymous pair of players to suggest descriptive terms for unlabeled images. Peekaboom [Von Ahn et al. 2006] 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 [Ho et al. 2007] 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, but, unlike our approach, which can be applied to any gesture-based game with an RGB-D sensor, require the development of a new game targeted to the specific data collection task.

2.2. Kinect-Based Datasets

The increase in popularity of inexpensive RGB-D sensors has correlated with an increase in publicly available datasets generated using these sensors. Generally, these datasets are the 3D analogs of popular 2D datasets, with much of the focus on object recognition. The Berkeley 3-D Object Dataset (B3DO) [Janoch et al. 2011] is an indoor-environment object detection dataset that includes RGB and depth images, as well as manual class annotations, crowdsourced via Amazon Mechanical Turk (AMT). Unlike many datasets, B3DO is not fixed; there is a mechanism to add data and annotations through online submission. The RGB-D Object Dataset [Lai et al. 2011] from the University of Washington similarly combines RGB and depth images, and includes multiple views of each object recorded. The Microsoft Research Cambridge-12 Kinect gesture dataset [Fothergill et al. 2012] includes several hours of people performing specific gestures, and contains joint estimates from a Kinect sensor. The RGB-D People Dataset [Spinello and Arras 2011] was developed for human detection and is manually annotated with the locations of the people in the scenes. Because we are proposing not a dataset but a method for generating datasets, compared to these approaches, our method allows for an ever-growing dataset as more data is collected from gameplay.

Table I. Comparison of Pose Estimation Datasets

Dataset	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	-	*Indoor*[a]	*Full*	*24*

[a]MS Kinect is typically used indoors but has also been used outdoors [Milani and Calvagno 2012].

Moreover, each of these datasets requires some form of manual annotation, which is not necessary for our method.

2.3. Pose Estimation Datasets

Our gamesourcing model results in an annotated dataset that can be used as training and testing data for 2D pose estimation. There are several existing manually annotated datasets. PARSE [Ramanan 2007] consists of 305 outdoor images of people (∼150 pixels tall) mostly playing sports, and contains high background clutter and self-occlusions. BUFFY [Ferrari et al. 2008] 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 [Eichner and Ferrari 2009] 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) [Bourdev and Malik 2009] contains 1240 high-quality images of people with 3D joint positions. The LEEDS Sports Pose [Johnson and Everingham 2010] dataset consists of 2,000 annotated images (recently extended to 10,000 [Johnson and Everingham 2011]) of people (∼150 pixels tall) performing sports activities.

Table I summarizes the related manual datasets and provides a comparison to the type of data possible using a gamesourced approach. The advantages of gamesourcing in annotating this type of data are scalability and flexibility. The number of possible images that can be collected is not limited by budgets for manual annotation or time of human annotators. Data can be collected in a wide variety of environments by a wide variety of people of various sizes and body styles. The resulting data contains images of reasonable quality with more joint annotations than current manual approaches. Some potential drawbacks to gamesourced data include the backgrounds typically being limited to indoor environments and the constraints in poses introduced by most games (e.g., players face the camera, complex or acrobatic poses are not used). In Section 5.2, we build a gamesourced dataset to compare to manually annotated datasets and demonstrate that these perceived limitations do not have a significant effect on the accuracy of pose estimation algorithms.

3. DATA COLLECTION AND FILTERING

RGB-D sensors, such as Kinect, provide depth estimates at each pixel in addition to color information. 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. Middleware commonly used with Kinect sensors (e.g., Microsoft SDK, OpenNI/NITE) employs fast algorithms for real-time 3D joint position estimation. Up to 24 joint locations can be obtained from Kinect.[1] Figure 1 shows an example of output from Kinect during gameplay:

[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).

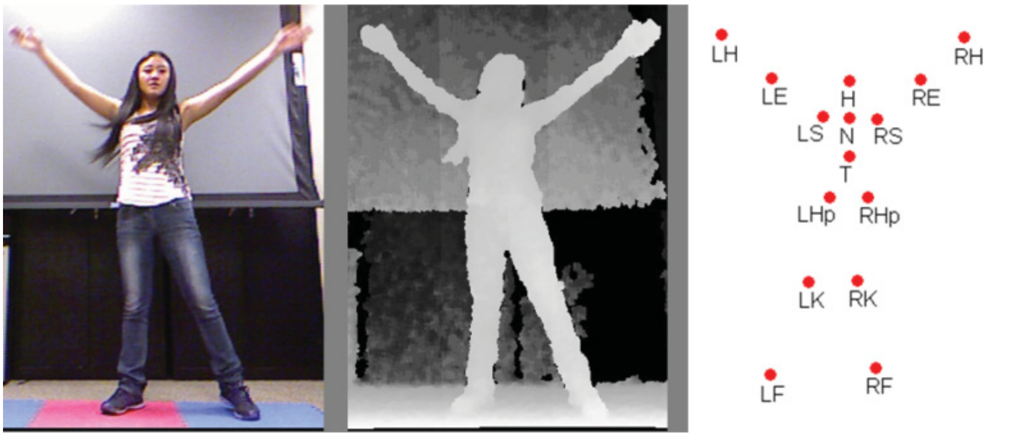


Fig. 1. We collect annotated images from motion-based games as training data for pose estimation from 2D images. Microsoft Kinect outputs (from *left* to *right*): an RGB image, depth map, and joint locations.

an RGB image, depth map, and joint locations. A typical gaming session generates a large amount of data, much of it not needed. The most direct approach would be to simply store all of this data over the course of the gaming session. However, much of the data is either unusable or unnecessary. First, although the depth-based pose estimation algorithms are usually reliable, there can be instances in which erroneous data is extracted. Second, consecutive frames or images of a player repeating a pose can be quite similar and, therefore redundant. These two issues could be addressed as a postprocessing operation 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.

3.1. Misregistered Data

At each time step, a candidate image and joint position estimates are evaluated to check for registration. We perform three simple tests to quickly eliminate potentially erroneous data. First, we rely on the confidence values returned by the 3D pose estimation algorithms used by the RGB-D sensor. Images containing at least one joint of low confidence are discarded. Figure 2(a) shows a sample image in which the positions of the right elbow and right-hand joints have low confidence scores; this frame would not be saved. Second, using the foreground mask supplied with the depth data, any instances in which joints fall outside of the foreground, such as in Figure 2(b), are also discarded. This situation most often arises in cases of very fast motion or joint self-occlusion. Finally, most image-based pose estimation methods extract patches surrounding the joint location. Cases in which a joint location is too close to the image border for a patch to be extracted are discarded. Each of these tests can be applied in real time, and depending on the amount of player motion, typically preserve 10% to 30% of the candidate frames. In Section 5.1, we show how these simple tests significantly reduce the registration error between the joint estimates and ground truth.

3.2. 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 a variant of locality-sensitive hashing (LSH) [Datar et al. 2004] for finding nearest neighbors

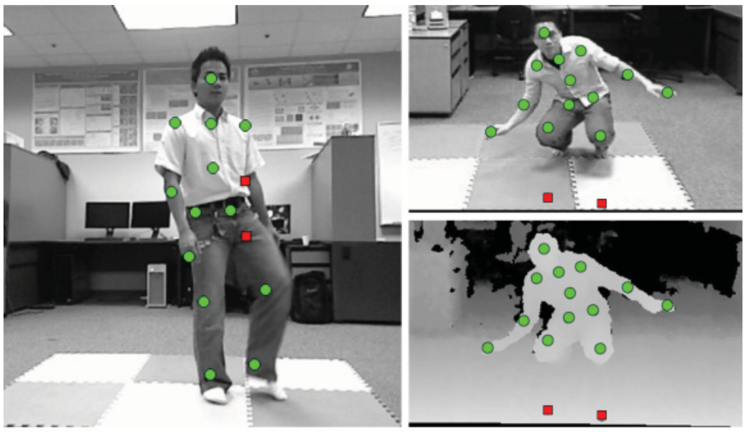


Fig. 2. Examples of filtering misregistered data. Green circles represent joint estimates of high confidence within the foreground mask and red squares represent estimates of low confidence or outside the foreground mask. In both cases, these images and joint estimates would be discarded.

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, thus this scheme avoids the linear-time approach of calculating the distance between the query and all database points.

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 3(b)) 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 neck (N) to the head (H). 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. Figure 4 depicts the process of discarding nonunique poses. In Section 4.3, we describe LSH parameter selection.

4. BUILDING GAMESOURCED DATASETS

In this section, we describe how we built a gamesourced dataset using the framework outlined in Section 3. Our method, PoseGrabber, was developed in C++ and runs alongside Kinect-based PC games.

4.1. Hardware

All experiments were carried out on a Windows 7 laptop with a 3.33GHz CPU and 8GB RAM. PoseGrabber, as described, could be deployed using either of the two most

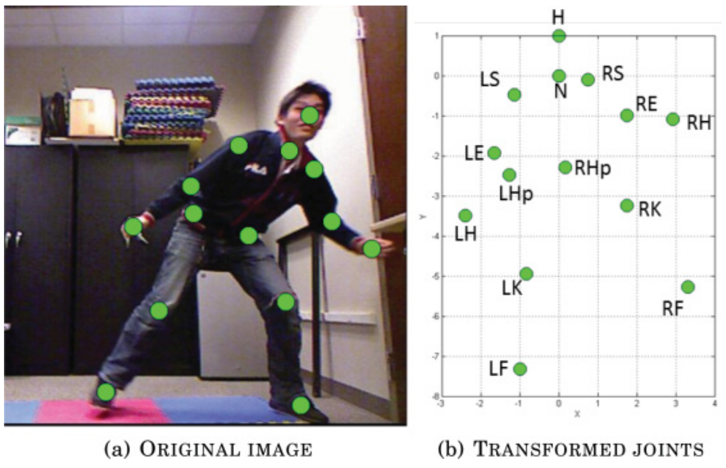


Fig. 3. To provide invariance to global pose shifts, we apply an affine transformation to each point from canonical image coordinates to a space defined (in the positive y direction) by the unit vector from the neck (N) to the head (H).

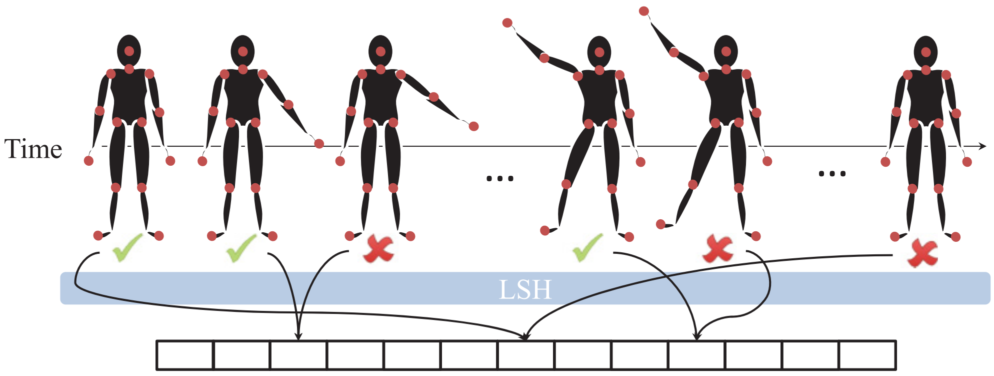
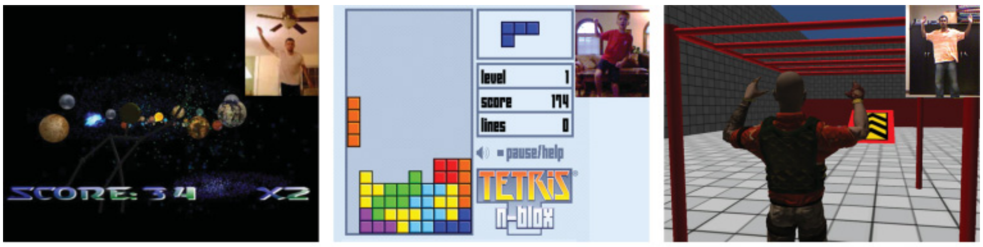


Fig. 4. Real-time pose uniqueness check. If a new pose collides with an existing database pose, it is discarded; otherwise it is added to the database. Using LSH, the collision checks occur in sublinear time (in the number of database items).

popular RGB-D sensors, PrimeSense 3D Sensor or Microsoft Kinect. The PrimeSense device provides built-in capabilities for temporally synchronized RGB and depth images, which is not the case with Kinect. Slight asynchronicity between the RGB and depth images does not affect gameplay; however, to use this data for training pose estimation algorithms, it is important that the joints and images are captured as nearly simultaneously as possible. In order to ensure that the data is temporally synchronized, we include an additional check of the data. In a call to the Kinect data acquisition function, the current RGB image and depth image are returned with individual timestamps (in ms), which we refer to as t_{RGB} and t_{DEP}, respectively. Because the joint estimates are derived from the depth images, their timestamps are the same for a given instance. 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 Kinect. Alignment error was minimized when $t_{DEP} - t_{RGB} \in [5, 15]$. On an unloaded system, data can be acquired at 30 frames per second. If the timestamp differences align within the time window, the data is assumed



(a) Astrojumper (b) Kung Fu Tetris (c) ICM Virtual Playground

Fig. 5. Three different games were used to collect data for evaluation.

to be synchronized and retained for further processing, as described in Section 3. Due to both the asynchronicity between sensors and occasional dropped frames, during a typical gaming session, between 2% and 30% of the data is preserved.

4.2. Games

To vary the set of poses the players would perform, we selected three different gesture-based Kinect games for the PC: Astrojumper [Finkelstein et al. 2011], Kung Fu Tetris [Kinect Hacks 2012], and ICM Virtual Playground. Each of the games was evaluated by a variety of users in a variety of locations. Figure 5 shows a screenshot of each game and the corresponding image of the user in the inset.

—*Astrojumper* is a full-body, Kinect-based game developed to motivate exercise. The player controls a flying avatar as different objects (e.g., asteroids, planets, ships) 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 Kinect on a PC via the Flexible Action and Articulated Skeleton Toolkit (FAAST) [Suma et al. 2011].
—*Kung Fu Tetris* is a Kinect-based modification of the traditional Tetris game, allowing players to direct blocks by kicking. Side kicks move blocks left and right, while forward kicks rotate the blocks. Points are earned for positioning blocks to complete rows. This version of Tetris uses FAAST to map player foot motion to keyboard input.
—*ICM Virtual Playground* is a virtual environment designed to showcase the potential for controlling a virtual character using Kinect. The game allows a player to control a full-body avatar and interact with a series of objects, such as monkey bars, ramps, and a zipline. There are no points or time limit; the player is free to explore the environment as desired.

4.3. Parameter Selection

The data-filtering steps involve a number of free parameters. For all experiments, the following settings were used.

Data Registration. The OpenNI drivers for Kinect provide confidence values for each joint estimate in the range $[0, 1]$. Low confidence values tend to correspond to inaccurate matches due to self-occlusion or fast motion, thus we discard any poses containing at least one joint with a confidence score less than 1. This conservative approach discards potentially useful poses, but we observed that we still collected many registered matches. Additionally, to allow for image patch extraction around each joint, poses with joint estimates within 10 pixels of the border were also discarded.

Redundant Data. A key step in the real-time processing of the matched joint/image pairs is the uniqueness check using LSH. LSH has four free parameters, R (Euclidean

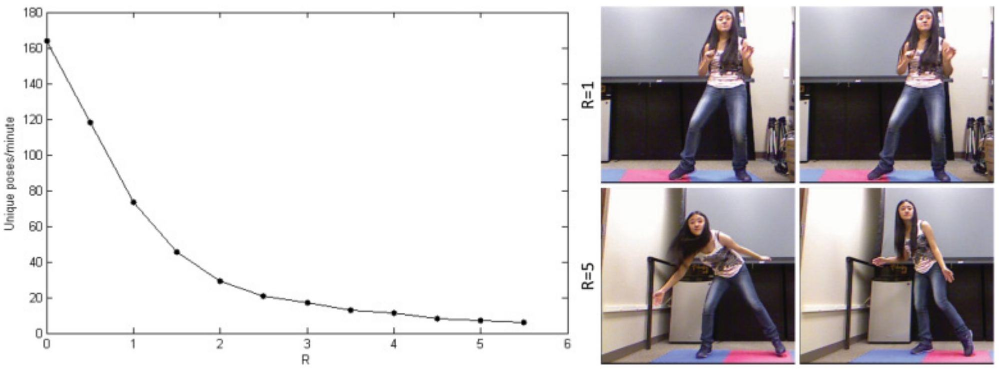


Fig. 6. The graph shows the amount of data saved per minute as a function of the similarity threshold, R. The images show examples of the closest unique poses from a 3-minute gaming session. At $R = 1$, the poses are similar and would be redundant for training. At $R = 5$, only nine unique poses were saved, thus interesting poses were likely missed.

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 players engaged in a 3-minute session of Astrojumper. Figure 6 shows the plot of the amount of data saved per minute as a function of R and examples of the closest (in Euclidean distance) poses of the resulting sets of saved images. This parameter provides a way to tune the amount of data collected over the course of a gaming session, perhaps due to storage limitations or I/O overhead. For our tests on a standard laptop, $R = 2$, provided a reasonable balance between the size of the set of unique poses returned and the file size of the data saved. For the remaining experiments, we use $R = 2$, $\gamma = .1$, $k = 10$, and $L = 55$.

4.4. Gamesourced Datasets

We constructed datasets for each of the three games (see Section 4.2) and a fourth dataset that combines the three. ASTRO consists of data from 15 different people playing Astrojumper for roughly 4 minutes each. KFT contains data from 17 different people playing Kung Fu Tetris for roughly 3 minutes each. ICM contains data from 14 different people playing ICM Virtual Playground for roughly 3 minutes each. Over all game sessions, we recorded in excess of 50,000 images. Each dataset is constructed by randomly selecting 300 images[2] each from the total collected for each game, split evenly into training and testing sets. The combined dataset, GS, was constructed by randomly selecting 100 images from ASTRO, KFT, and ICM.

5. EVALUATION

To evaluate the use of PoseGrabber to collect annotated gamesourced data automatically requires a multifaceted approach. First, we examine the accuracy of the joint estimates returned by the sensor and quantify the effects of the simple filtering methods described in Section 3. Second, we evaluate how a dataset constructed using this

[2]Only 300 images were used in the evaluation of the gamesourced data to allow for a fair comparison to the manually annotated datasets. One of these, PARSE only contains 305 total images.

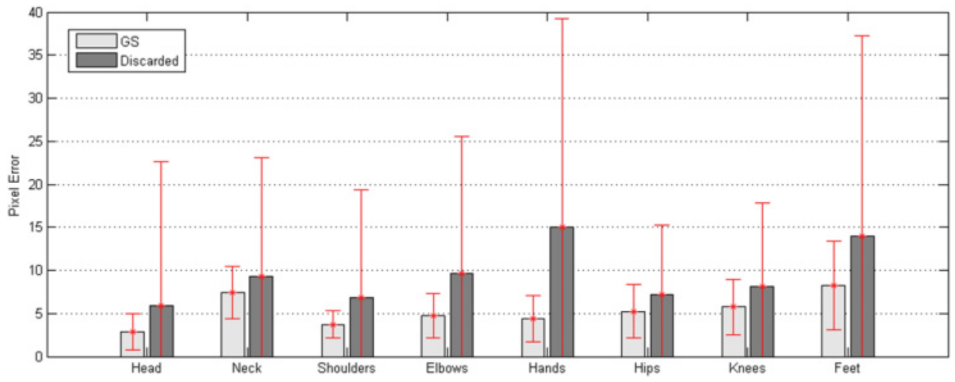


Fig. 7. Error (in pixel units) of Kinect joint estimates for data used in gamesourced dataset and discarded by our real-time filtering method. Error bars represent one standard deviation.

gamesourced approach compares to existing, manually annotated 2D pose estimation datasets using two recently developed, learning-based pose estimation algorithms [Yang and Ramanan 2011; Sapp and Taskar 2013]. Third, we evaluate the effect of game choice on the quality of the resulting dataset.

5.1. Joint Estimate Accuracy

To estimate the accuracy of the joint estimates in the gamesourced data, the images in GS were manually annotated. Additionally, we selected a subset of images and joint estimates that were discarded using the methods described in Section 3 and compared those estimates to manually annotated labels. Figure 7 shows the mean error, in pixel units, between the Kinect joint position estimates and ground truth for both the GS data and the discarded data. For all joints, the filtering process returned more accurate data; in some cases (e.g., head, hands), the error was worse by more than 100%. For the GS dataset, generally error is low (within a few pixels). The highest error joints are the feet, which the Kinect typically estimates higher than a human observer would, and neck, which is typically estimated lower.

5.2. 2D Pose Estimation Experiments

Recent methods for 2D pose estimation use a discriminative approach that collects statistics on labeled datasets for model building [Felzenszwalb et al. 2009; Rosales and Sclaroff 2001; Sminchisescu et al. 2005]. Methods for pose estimation "by parts" use human body models that differ in the number or type of joints specified. Our experiments include two recent pose estimation methods.

5.2.1. Pose Estimators. We employ the 14-joint pose estimation model of Yang and Ramanan [2011], which we refer to as YR. These 14 joints are a subset of those returned by RGB-D sensors, thus the joint positions can be mapped directly. YR uses a mixture of parts to estimate poses in 2D images. Through interpolation, the 14 input joint locations are extended to 26 total locations and equal-sized image patches are extracted from each location. 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 overrepresented. 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.

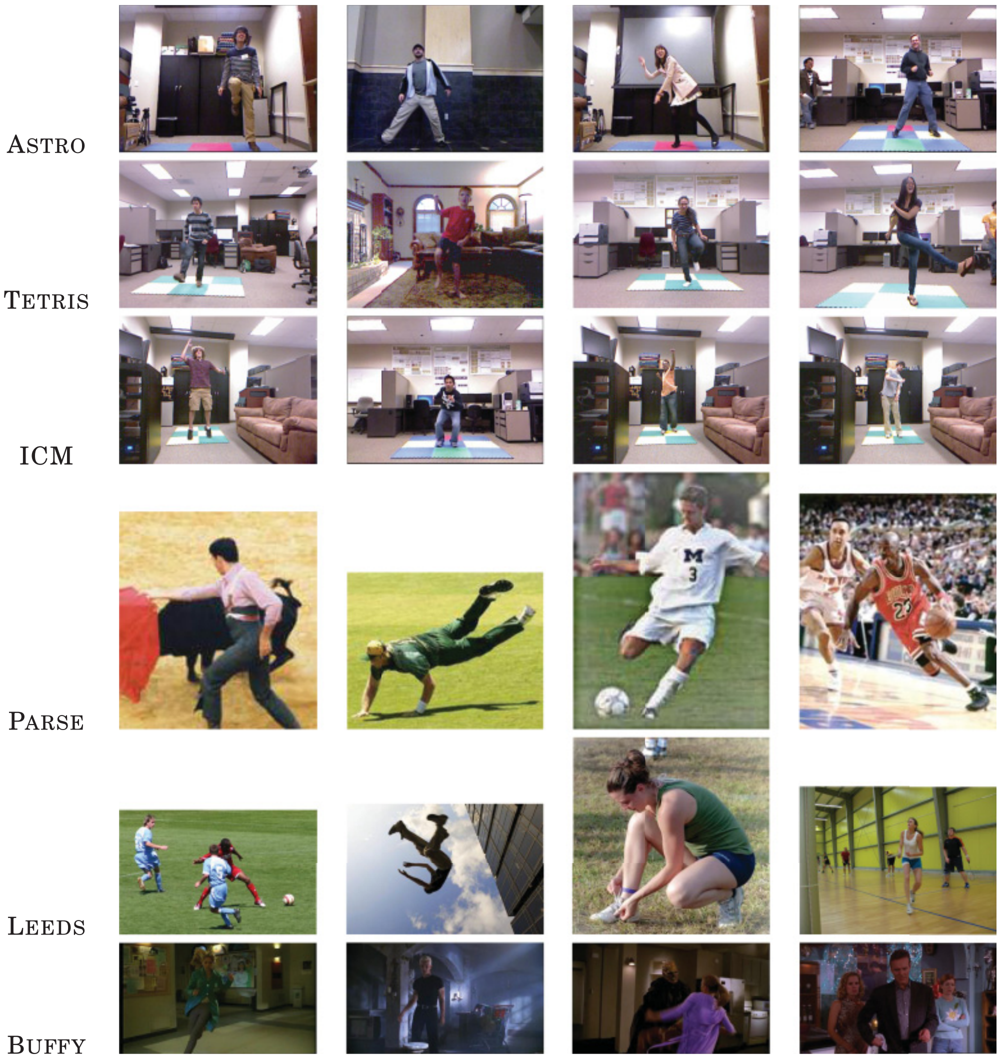


Fig. 8. Example images from the datasets used in the experiments.

The second model is the Multimodal Decomposable Models for Human Pose Estimation (MODEC) [Sapp and Taskar 2013]. MODEC trains a set of locally linear models for representative joint configurations based on HOG descriptors. Whereas YR focuses on local-part reasoning (i.e., parent and child joints), MODEC considers global pose configurations in addition to locating individual joints. These configurations, or modes, are learned in an unsupervised fashion from training data. Global mode inference enables MODEC to estimate poses more quickly, avoiding the necessity of considering all possible pairs of joint combinations. For this experiment, we use the MODEC 6-joint model (shoulders, elbows, and wrists), mapping the relevant joints from the larger set returned by the RGB-D sensor.

5.2.2. Data and Experiments. We compare our combined gamesourced dataset GS to three of the manually constructed datasets (Parse, Buffy, Leeds). Figure 8 shows

Table II. YR Pose Estimation Results in PCP (Higher is Better)

	Training			
Testing	GS	Parse	Leeds	Buffy
GS[a]	0.92	0.84	0.62	0.78
Parse	0.70	0.78	0.59	0.70
Leeds	0.61	0.70	0.55	0.57
Buffy	0.68	0.71	0.67	0.85
Average	0.73	0.76	0.61	0.73

[a]The GS test set uses manually annotated joint locations.

Table III. MODEC Pose Estimation Results in PCP (Higher is Better)

	Training			
Testing	GS	Parse	Leeds	Buffy
GS[a]	0.54	0.38	0.37	0.46
Parse	0.48	0.48	0.40	0.41
Leeds	0.33	0.33	0.38	0.32
Buffy	0.42	0.23	0.34	0.49
Average	0.44	0.35	0.37	0.42

[a]The GS test set uses manually annotated joint locations.

sample images from each dataset. For each dataset, we randomly selected 300 images each, split evenly into training and testing sets. Using each of the two pose estimation methods, we performed pairwise pose estimation experiments with each of the four training and testing sets. For GS, we used the automatically estimated joints for training. However, for test images, we used the manually annotated joint locations. Results for these tests are reported as Percentage of Correctly estimated Parts (PCP).[3] Table II shows the results using the YR model, and Table III shows the results with MODEC.

5.2.3. Results. Overall, using the YR model, the gamesourced dataset, GS, performed about as well as the manually curated sets when used as training data. On average, Parse performed best, with GS and Buffy proving nearly as effective. Except for Leeds, each training set performed best when the test set was drawn from the same source.[4] GS for training and testing resulted in the highest performance over all experiments. Compared to Parse and Leeds, for which each image contains a different person and background, the limited number of different activities, players, and locations could explain the similarity of the scenes. However, the apparent relative lack of data diversity in GS did not negatively affect the performance when used as training data with test data from other datasets. Overall, GS data outperformed data from Leeds and was equally as effective as Buffy when used as training data across all of the images.

With the MODEC model—similar to YR—for each dataset used in testing, the highest result is achieved when the matching dataset is used for training. In each case, the second-best results are achieved by the GS model. On average, the model trained on GS performs best across all the test sets. Overall, PCP scores are lower for MODEC than YR. ISapp and Taskar [2013] observe that the method is sensitive to the amount

[3]Percentage of Correctly estimated Parts (PCP) [Ferrari et al. 2008] is a widely used metric for 2D pose estimation accuracy. A body part is considered correctly labeled if it lies within 50% of the length of the ground truth body-part segment. Buffy contains annotations for only the upper body, thus the corresponding results include only upper-body joints.

[4]This phenomenon has also been observed in the domain of object recognition from images; in this domain it was discovered that, when performing training and testing on combinations of datasets, the highest recognition scores resulted from matched sets [Torralba and Efros 2011].

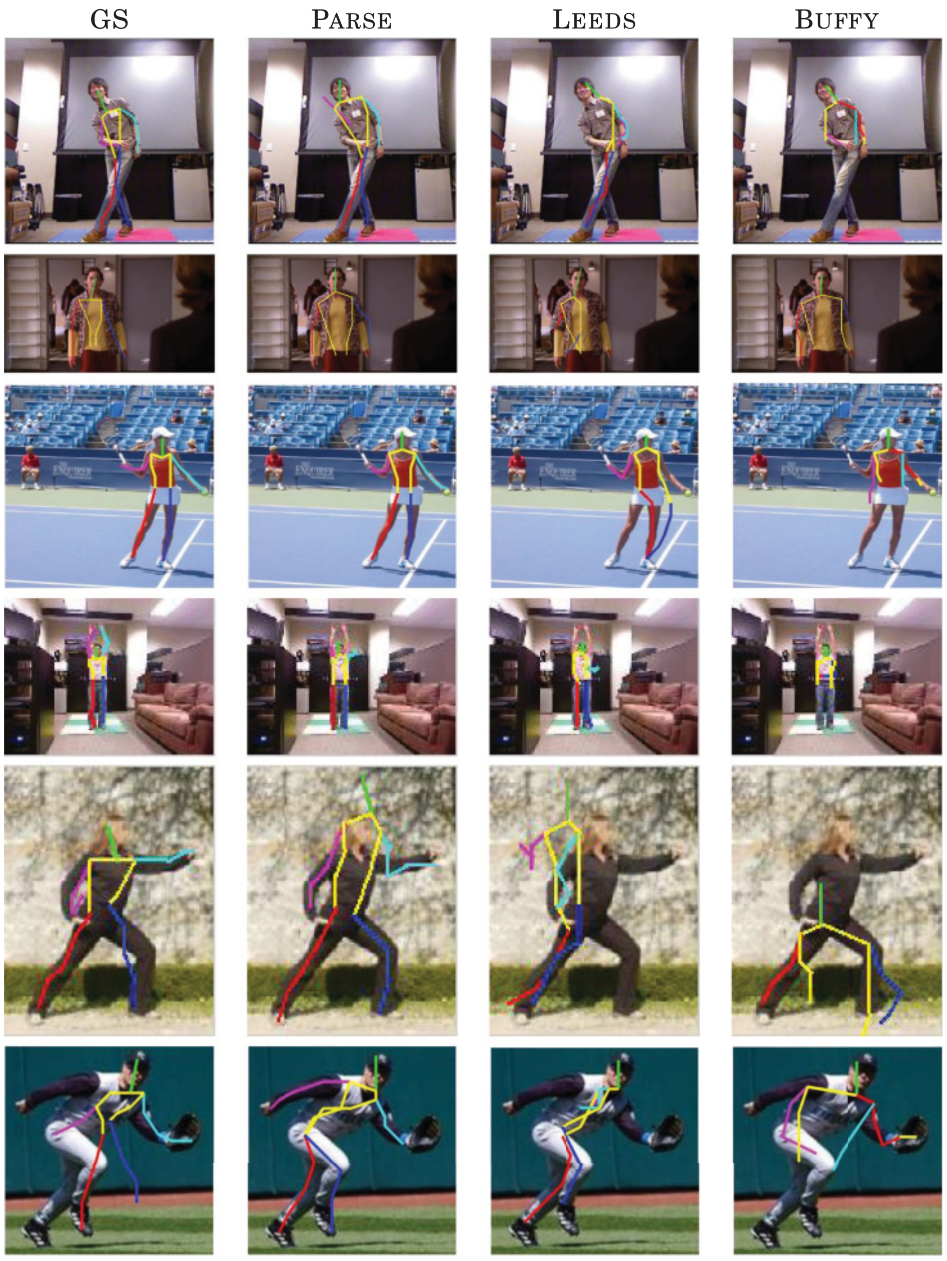


Fig. 9. Example results from the YR experiments summarized in Table II. Each row shows example results on the same image with training data from the gamesourced data (GS) or the manually constructed datasets (PARSE, LEEDS, and BUFFY).

of training data; increasing the training data by an order of magnitude significantly increased the training accuracy and computational resource consumption.

Figure 9 shows example results using the YR model with the estimated pose overlaid. Each row shows a test image used with different training data. The first image, from GS, shows an example for which, when trained with the gamesourced data, the algorithm was able to correctly discover a case of self-occlusion. The second image, from BUFFY, and third image, from LEEDS, both show a simple pose in a cluttered background, that was missed in some cases. The fourth image, from GS, and fifth image, from PARSE, both show a moderately difficult pose that was successfully identified when trained on the GS dataset. The sixth image, from PARSE, was difficult in all cases.

Figure 10 shows example results using MODEC with the estimated pose overlaid. Each row shows a test image used with different training data. The first three images show relatively simple poses from BUFFY, GS, and LEEDS, respectively, in which MODEC generally performs well irrespective of which dataset is used for training. The fourth and fifth images, from GS and LEEDS, show examples of self-occlusion, which MODEC successfully identifies when trained on the GS dataset. The sixth and seventh images, both from PARSE, show more difficult poses for which MODEC performs poorly when trained with any of datasets.

One notable difference between the images in each set was the apparent variance of the poses. PARSE and LEEDS 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, transformed as described in Section 3, for 150 images from each set and computed the number of components necessary to represent 95% of the variance. Table IV shows the pose variance estimates for the upper, lower, and full body joints for the four datasets, averaged over 100 trials with subsets of the datasets of 150 images. The measure appears to match the visual diversity in the poses present in the datasets. However, in general, the pose variance did not correlate with performance on pose estimation. For example, PARSE has the greatest whole-body variation among the four datasets. However, while we observed the highest average PCP with PARSE when using the YR model, with the MODEC model, PARSE has the lowest average PCP.

5.3. Game Selection

The three games selected each induced a different series of actions in the players. Astrojumper typically involves full-body motion (e.g., hopping, ducking), while Kung Fu Tetris responded only to leg actions, and ICM Virtual Playground relied mostly on upper-body motion. To determine whether or not game selection affected the quality of the resulting data source, we performed 2D pose estimation experiments using data from each game individually. Using the YR model from Section 5.2, we performed 9 pairwise pose estimation experiments with ASTRO, KFT, and ICM. Table V shows the results for these tests, reported as Percentage of Correctly estimated Parts (PCP). In spite of the different requirements and actions induced by each game, the PCP scores were similar regardless of which dataset was used for testing and training. We also compared the pose variance of each game. Table VI shows the pose variance estimates for the upper, lower, and full-body joints for the three datasets, averaged over 100 trials with subsets of the datasets of 150 images. The similarities in PCP and pose variance scores matches the visual analysis of the images. Even though certain games emphasized lower- or upper-body motions, most players made full-body motions in the course of playing the games. This suggests that any game that incorporates some real-world actions can be suitable for use in this gamesourcing framework.

6. DISCUSSION AND FUTURE WORK

We presented a framework for collecting automatically labeled data through gamesourcing, dramatically reducing the manual effort of data annotation. Our approach does not involve a game specifically developed to facilitate crowdsourcing; it can be used with existing gesture-based games. In an evaluation of a dataset created with our method using two different pose detection algorithms, we achieved results comparable to training with manually annotated data. These results may be unexpected since the

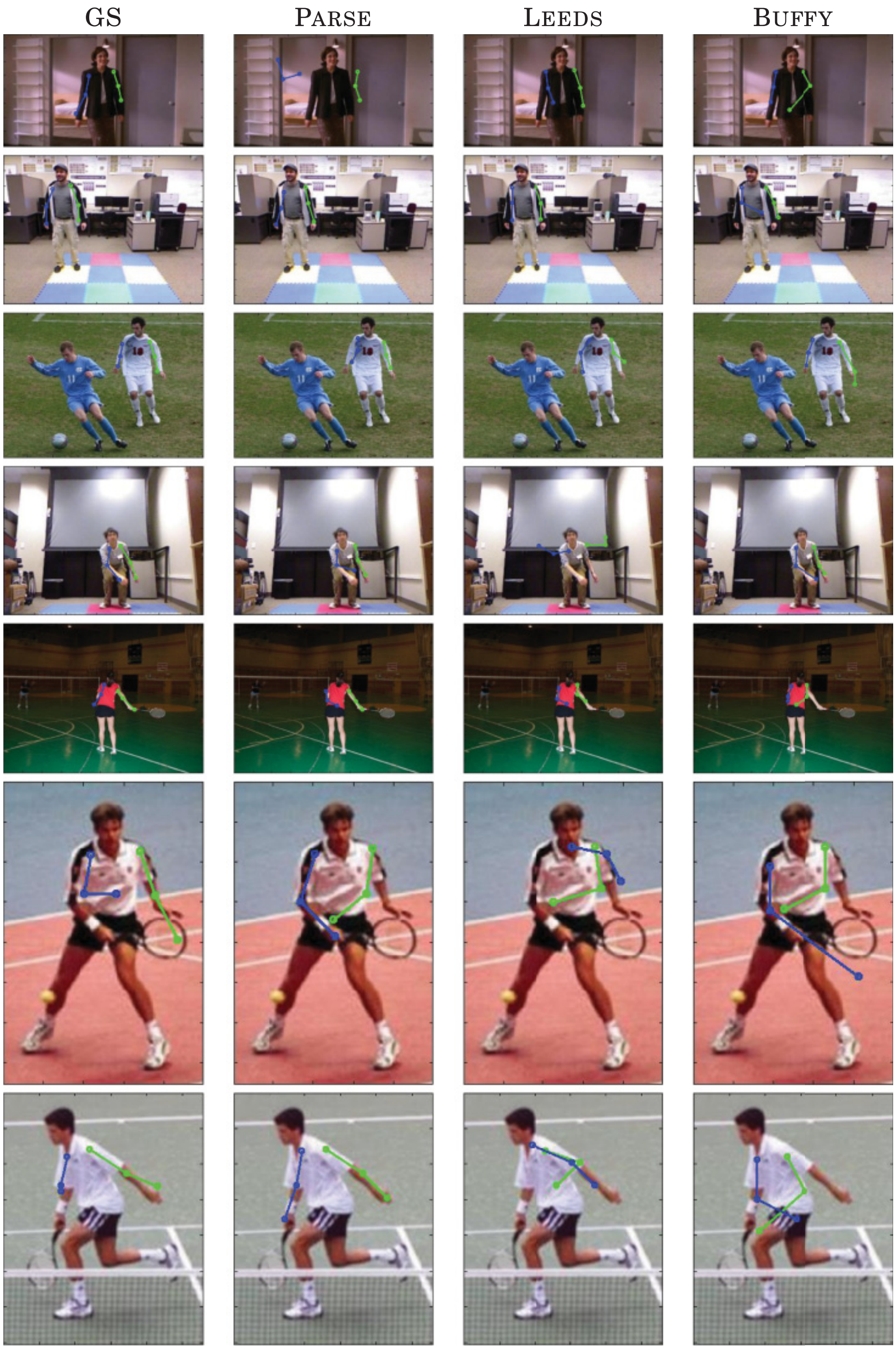


Fig. 10. Example results from the MODEC experiments summarized in Table III. Each row shows example results on the same image with training data from the gamesourced data (GS) or the manually constructed datasets (PARSE, LEEDS, and BUFFY).

Table IV. Pose Variance Estimates

Dataset	Upper	Lower	Whole body
PARSE	7.08	4.00	9.36
LEEDS	6.83	3.74	8.55
BUFFY	7.29	N/A	7.21
GS	4.02	3.94	7.17

Note: Upper refers to the top 10 joints and lower to the bottom 4.

Table V. PCP Results (Higher is Better) for Training and Testing with Images from Three Gamesourced Datasets

	Training		
Testing	ASTRO	KFT	ICM
ASTRO	0.92	0.85	0.82
KFT	0.85	0.88	0.80
ICM	0.82	0.81	0.89

Table VI. Pose Variance from Gamesourced Datasets

Dataset	Upper	Lower	Whole body
ASTRO	4.00	3.90	7.15
KFT	4.22	4.00	7.31
ICM	4.19	3.91	6.16

Note: Upper refers to the top 10 joints and lower to the bottom 4.

gamesourced data was collected from a relatively small number of players in a limited number of indoor settings, playing only a few different games and performing a restricted set of motions dictated by the game. In spite of these perceived limitations, our evaluation shows that a pose estimation algorithm trained on gamesourced data was able to detect poses from a wide variety images, including those containing actors from television shows and outdoor athletes in the middle of complex midair maneuvers.

While we expect the diversity of data collected via gamesourcing to increase with the number and type of games and users, this will not overcome the inherent limitations of gesture-based gaming. Even the most enthusiastic gamer is unlikely to execute the type of acrobatic actions performed by outdoor athletes, and most games require the player to face the camera and monitor. However, when compared to data from existing pose databases, our results suggest that these limitations do not negatively impact the quality of the data for training pose estimation algorithms.

Beyond passive data collection, one possible direction for future work would be to incorporate this module into a specific game or game engine to dynamically adjust the goal or position of objects to compel users to perform underrepresented or desired poses. This proposed extension moves away from the general-purpose, game-agnostic approach of our framework and more closely follows the model of game-specific crowdsourcing, but could be used to collect specific poses or incentivize players to perform different or complex motions.

REFERENCES

Lubomir Bourdev and Jitendra Malik. 2009. Poselets: Body part detectors trained using 3D human pose annotations. In *Proceedings of the International Conference on Computer Vision*. IEEE.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Symposium on Computational Geometry*. ACM, New York, NY, 253–262.

Marcin Eichner and Vittorio Ferrari. 2009. Better appearance models for pictorial structures. In *Proceedings of the British Machine Vision Conference*.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. 2009. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence* 32, 1627–1645.

Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. 2008. Progressive search space reduction for human pose estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

Samantha Finkelstein, Andrea Nickel, Zachary Lipps, Tiffany Barnes, Zachary Wartell, and Evan A. Suma. 2011. Astrojumper: Motivating exercise with an immersive virtual reality exergame. *Presence: Teleoperators and Virtual Environments* 20, 78–92.

Simon Fothergill, Helena M. Mentis, Pushmeet Kohli, and Sebastian Nowozin. 2012. Instructing people for training gestural interactive systems. In *Proceedings of the CHI*, Joseph A. Konstan, Ed H. Chi, and Kristina Höök (Eds.). ACM, New York, NY, 1737–1746.

Chien-Ju Ho, Tsung-Hsiang Chang, and Jane Yung jen Hsu. 2007. PhotoSlap: A multi-player online game for semantic annotation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1359–1364.

Allison Janoch, SSergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. 2011. A category-level 3-D object dataset: Putting the Kinect to work. In *Proceedings of the International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 1168–1174.

Sam Johnson and Mark Everingham. 2010. Clustered pose and nonlinear appearance models for human pose estimation. In *Proceedings of the British Machine Vision Conference*.

Sam Johnson and Mark Everingham. 2011. Learning effective humanpose estimation from inaccurate annotation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE.

Kinect Hacks. 2012. Kung Fu Tetris. Retrieved February 17, 2015 from http://www.kinecthacks.com/kinect-game-kungfu-tetris/.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. 2011. A large-scale hierarchical multi-view RGB-D object dataset. In *Proceedings of the International Conference on Robotics and Automation (ICRA'11)*. IEEE, 1817–1824.

Simone Milani and Giancarlo Calvagno. 2012. Joint denoising and interpolation of depth maps for MS Kinect sensors. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 797–800.

Deva Ramanan. 2007. Learning to parse images of articulated bodies. In *Advances in Neural Information Processing Systems*. 1129.

Rómer Rosales and Stan Sclaroff. 2001. Learning body pose via specialized maps. In *Advances in Neural Information Processing Systems*. 1263–1270.

Benjamin Sapp and Ben Taskar. 2013. Multimodal decomposable models for human pose estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE.

Cristian Sminchisescu, Atul Kanaujia, Zhiguo Li, and Dimitris Metaxas. 2005. Discriminative density propagation for 3D human motion estimation. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, 390–397.

Richard Souvenir, Ayman Hajja, and Scott Spurlock. 2012. Gamesourcing to acquire labeled human pose estimation data. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 1–6.

Luciano Spinello and Kai O. Arras. 2011. People detection in RGB-D data. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS'11)*. IEEE, 3838–3843.

Evan A. Suma, Belinda Lange, Albert Rizzo, David Krum, and Mark Bolas. 2011. FAAST: The Flexible Action and Articulated Skeleton Toolkit. In *Virtual Reality*. IEEE, 247–248.

Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE, 1521–1528.

Luis Von Ahn and Laura Dabbish. 2004. Labeling images with a computer game. In *Proceedings of the International Conference on Human Factors in Computing Systems*. ACM, New York, NY, 319–326.

Luis Von Ahn, Ruoran Liu, and Manuel Blum. 2006. Peekaboom: a game for locating objects in images. In *Proceedings of the International Conference on Human Factors in Computing Systems*. ACM, New York, NY, 55–64.

Yi Yang and Deva Ramanan. 2011. Articulated pose estimation with flexible mixtures-of-parts. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*. IEEE.

Received June 2013; revised January 2014; accepted March 2014