Markerless 3D Pose Estimation from RGB Data

Daniel Murphy

May 1, 2019

Abstract

Obtaining accurate markerless 3D pose estimations of subjects is an important problem in neuroscience and other fields. Recent work has employed CNNs very successfully to make pose estimations in 2D using limited data. We explore several models that attempt to leverage the accuracy of 2D models while extrapolating them to 3D, operating on RGB images taken from several viewpoints. These include: methods to refine 2D predictions, a model that learns the appropriate confidence that should be placed each on each of a set of 2D predictions when merging them to 3D, a model that explicitly aligns relevant features between camera views, and a model that incorporates error from 3D predictions directly into its objective function. Our results show that of these techniques, the most accurate 3D predictions are produced by learning appropriate confidence levels to attribute to the 2D predictions on which they are based.

1 Background

The field of neuroscience is full of open questions regarding the relationship between neural activity and motor behavior, such as how complex activities are encoded in the brain before they are carried out. To answer such questions, researchers must often analyze datasets describing subjects' body positions and correlate them with datasets describing subjects' neural activity. The collection of quality data from which to construct these datasets is therefore a task in high demand.

The current "gold-standard" for collecting this data involves placing markers on animals and humans, which allows the locations of joints to be tracked as they are filmed[1]. However, using markers is restrictive because the tracked joints must be determined beforehand, and they can be intrusive, potentially altering the behavior of subjects.

Though the marker system is imperfect, markerless pose estimation is a complicated task, which must address questions like how to identify joint locations in an image and what to do when a joint is occluded from view. In the case of a neural network based system, it is also non-trivial to obtain enough labeled data with which to train the network in the first place. Nonetheless, developments in deep neural network (DNN) architectures have yielded vision systems – such as OpenPose[2][3][4][5] and DeepLabCut[6] – that are better than ever at handling these challenges.

OpenPose is a system that allows simultaneous real-time tracking of multiple human poses. It first passes images through a pre-trained convolutional network that is initialized with the first ten layers of VGG-19[7] to extract features of interest. It then passes these features in parallel through two other CNNs, one of which produces confidence maps for the location of each joint, and the other of which produces "part affinity maps" that indicate which body parts are attached to which joints. These two maps are then combined to match up joints belonging to each person, in order to produce skeletons for as many people as are present in the image.

DeepLabCut is a similar system that is specialized for tracking a single laboratory animal. It passes images through several layers of the ResNet-50 CNN[8], which was pre-trained for ImageNet image classification, to obtain features of interest. It then passes these features through two parallel transpose convolutional upsampling layers, one of which produces confidence maps for the location of each joint, and the other of which produces "offset maps" that represent the amount that each joint's predicted location should be shifted to account for the inaccuracies resulting from the confidence maps being at a lower resolution than that of the original image.

Due to DeepLabCut's being designed for the purpose of tracking one animal at a time and its simpler architecture, we hoped that it would be more easily adaptable to our use cases, so it served as the starting point for the models we implemented. Both systems are still limited, however, in that they have been developed to predict 2D locations, while behavioral tracking would most benefit from 3D data.

3D pose estimation systems, on the other hand, can vary widely and have seen differing degrees of success. Some, for example, rely purely on 2D pose predictions, onto which they fit a 3D model[9]. Others fuse 2D predictions with monocular 3D image data[10]. We focus on fusing information from multiple calibrated 2D views. We hope that combining 2D visual cues from multiple perspectives will allow us to 1) take advantage of the accuracy of current 2D methods, 2) use redundant cues to refine predictions, and 3) better handle joints that are occluded from some views.

1.1 Related Work

Combining cues from different streams of data has proven helpful in completing similar tasks.

One straightforward way to combine 2D visual cues is to film a subject from multiple viewpoints, perform 2D pose estimation on each video, and stitch together the 2D information to construct 3D pose estimations. This approach is proposed in [11] (see section L of the Step-by-Step Procedure). It is also incorporated into a real-time pose-tracking system[12].

This model works well when all viewpoints give confident 2D predictions. Unfortunately, any error in these predictions can be greatly compounded when transforming them into 3D coordinates. To minimize such errors, we tested the effects of cropping images, as well as the performance of a network that incorporated attention maps to segment out background. We also tried to refine our predictions by combining information from multiple views within the network itself, and the remainder of our models are aimed at achieving this.

Instead of combining 2D information at the last step, we can also combine it earlier in our process. Sharing information across views within a neural network has been successful in tasks like object classification[13]. Cue combination inside a CNN for multiple information streams has also been successfully applied to 3D pose prediction for monocular data [14].

One challenge for cue combination in the multiview case while using a CNN for 3D pose prediction is that cues for the same joint location, taken from different viewpoints, do not necessarily appear in the same spatial regions of their respective images. CNNs overcome the large dimensionality of images by only operating on features grouped within a local spatial area, and so they cannot immediately use cues that are not spatially aligned. Furthermore, we cannot just pool feature activations across an entire image without losing spatial information that is required for pose tracking. Our models attempt to address this issue.

2 Data

We analyze RGB images collected through the Simi Motion markerless system, which provides calibration parameters and images taken from up to eight viewpoints. The presence of multiview data allows us to handle occlusions much more easily; however, the lack of direct depth information greatly affects how we can obtain 3D coordinates.

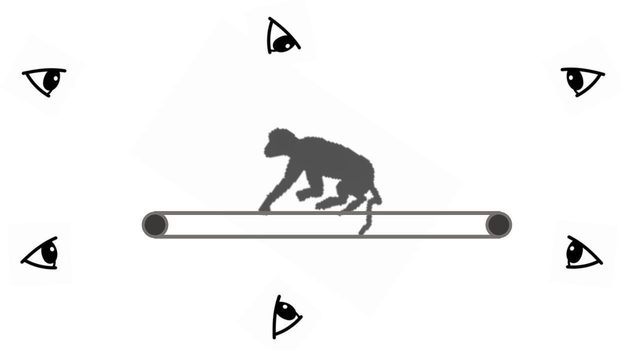


Figure 1: Illustration of Recording Setup for 6-view Dataset

The images depict a rhesus monkey performing various tasks on a treadmill, including walking, jumping, and reaching. At the same time, their brain activity is being monitored via a multielectrode array implanted in the motor cortex, attached to a head-mounted wireless neurosensor[15]. The purpose of this setup is to track the monkey's motion while keeping constraints due to wires at a minimum.

Our multiple-view labeled data comes from a recording session that involved six calibrated cameras completely surrounding the monkey, with about 8000 annotated frames per camera. The purpose of this session was to observe the monkey walking naturally on a treadmill. The tracked joints were the groin, hip, knee, ankle, metatarsal, and toe, on both the left and right sides of the body. We refer to this data as our 6-view dataset.

Additional data was later added from a recording session that involved three calibrated cameras capturing only one side of the monkey, with about 46000 frames per camera. Of these, only frames from one camera perspective were annotated. The purpose of this session was to observe the monkey leaping over an obstacle or reaching for a treat, which is why it was recorded in a separate setting. This session tracked the groin, hip, knee, ankle, metatarsal, toe, elbow, shoulder, fingertip, clavicle, wrist, and metacarpal, only on the left side of the body. We refer to this data as our 3-view dataset.

3 Base Model Architecture

We employ DeepLabCut - the state-of-the-art in 2D pose-prediction for laboratory settings - as our base model.

3.1 Notation

In describing our models, we use the following conventions:

- S indicates a 'score map' - a matrix of values between $-\infty$ and ∞
- P indicates a 'confidence map' - the result of the sigmoid function applied to a score map, where each element gives a confidence that a certain joint is located at a certain pixel
- O indicates an 'offset map' - a matrix of values providing x and y offsets for a joint prediction to account for low-resolution confidence maps
- p indicates a 2D point (x, y) in pixel coordinates
- ρ indicates a 3D point in (x, y, z) world coordinates

3.2 Original DeepLabCut Model

In this model, an input RGB image, represented by a tensor of shape $H \times W \times 3$, is first passed through a ResNet that has been pretrained for classifying ImageNet images[8]. The result is a tensor F of shape $H' \times W' \times C$, where each $H' \times W'$ slice is a feature map with lower resolution than the original image, and C is the number of such maps. These feature maps are then upsampled via two separate transpose convolutional operations to create a tensor of score maps, S, of shape $H'' \times W'' \times J$, and a tensor of offset maps for each joint, O, of shape $H'' \times W'' \times 2J$, which are both at one eighth of the original image resolution, where J is the number of tracked joints. This flow is illustrated in Figure 2. The scoremaps are later transformed into confidence maps, P, via the sigmoid function.

$P_{i,j,k}$ gives scores for the likelihood of joint k being present in the region of the original image corresponding to the pixel at (i, j), and $(O_{i,j,2k}, O_{i,j,2k+1})$ represents the (x, y) offsets of joint k from the center of the region of the original image corresponding to pixel (i, j).

The model thus predicts the location of joint k via the formula:

$$(\hat{x}_k, \hat{y}_k) = \operatorname{argmax} P_{\cdot,\cdot,k}$$

$$p_k = r \cdot (\hat{x}_k, \hat{y}_k) + O_{\hat{x}_k,\hat{y}_k,2k:2k+2} \tag{1}$$

where p_k is the predicted x and y pixel coordinates of joint k, r is the ratio of the image and score map resolutions (8 in our case), $(\hat{x}_k, \hat{y}_k)$ is the x and y pixel coordinates corresponding to the maximum value of the score map for joint k, and $O_{\hat{x}_k,\hat{y}_k,2k:2k+2}$ is the x and y offsets stored in O for joint k at pixel coordinate $(\hat{x}_k, \hat{y}_k)$. The confidence that the model attributes to this prediction is $P_{\hat{x}_k,\hat{y}_k,2k:2k+2}$.

The combined use of score maps and offset maps allows predictions to be made after only one upsampling, without the score maps needing to be at the same resolution as the original image. This allows accurate pose predictions without increasing the number of parameters through multiple transpose convolutional upsamplings, which is beneficial for training more quickly on limited data.

To train the model, ground-truth confidence maps and offset maps, P^* and O^*, are also computed from small regions surrounding the ground-truth pixel coordinates of each joint, and gradient descent is used to minimize the difference between the predicted and ground-truth confidence maps and offset maps. The objective

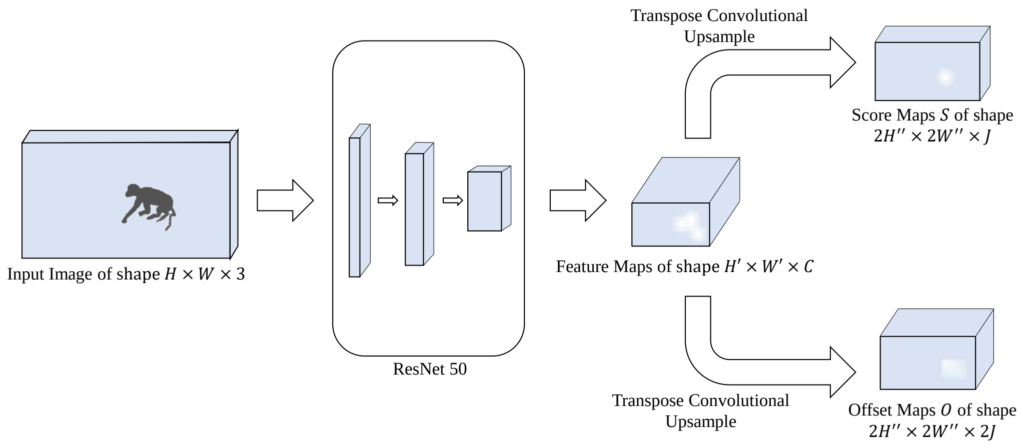

Input Image of shape $H \times W \times 3$
ResNet 50
Feature Maps of shape $H' \times W' \times C$
Transpose Convolutional Upsample
Score Maps S of shape $2H'' \times 2W'' \times J$
Transpose Convolutional Upsample
Offset Maps O of shape $2H'' \times 2W'' \times 2J$

Figure 2: DeepLabCut Architecture

function is to minimize $L_1 + L_2$, where L_1 is the cross-entropy loss between P and P^*, and L_2 is a Huber loss for O and O^* applied only where P^* is 1.

3.3 Baseline Multi-View DeepLabCut Model

In order to coordinate pose predictions between views, we augment the base DeepLabCut architecture to process multiple images at once.

An image from each view is passed separately through the same shared, pre-trained ResNet, giving us N tensors of feature maps, where N is the number of different views. Each view has its own sets of weights for the convolutional transpose operations that extract probability and offset maps.

We use these maps to obtain separate 2D pose predictions and confidence scores for each view in the way described above. We will use $p_k^{(n)}$ to denote the predicted (x, y) coordinates of joint k from the perspective of view n, and we will use $c_k^{(n)}$ to denote the confidence that our model attributes to this prediction. For every pair of views, we then use the associated pair of camera projection matrices to project their sets of pose predictions in 2D pixel coordinates into a set of predictions in 3D world coordinates, which we will denote $\rho_k^{(n_1,n_2)}$ for the pair of views (n_1, n_2) and joint k. We compute the final 3D prediction across all view pairs as a weighted sum of each set of 3D predictions:

$$\rho_k = \frac{\sum\limits_{n_1,n_2} \rho_k^{(n_1,n_2)} c_k^{(n_1)} c_k^{(n_2)}}{\sum\limits_{n_1,n_2} c_k^{(n_1)} c_k^{(n_2)}} \tag{2}$$

We also experimented with setting a threshold to dictate when to completely disregard a joint prediction, setting a prediction's weight to 0 if it is less than a certain amount. However, we found that setting a threshold did not significantly improve performance.

4 Experiments

4.1 2D Refinement Techniques

Though the baseline model performs well when all joints are clearly visible, it sometimes mistakes other objects, such as a robotic arm, for joints that are occluded. We evaluated three techniques to try to reduce this error.

4.1.1 Manual Cropping

The first was to manually crop the images to a region of interest. In our case, we found that most failed detections still occurred within the chosen bounding box, leading to no significant improvement. The need to manually choose a bounding box to cover new, unanalyzed video sessions would also increase manual labor.

4.1.2 Saliency Maps (Linsley et al.)

The next approach was to augment the network to learn saliency masks and automatically segment the relevant parts of the image from the background. To do this, heatmaps were produced by taking the max of 2D Gaussian bumps, one per joint, with a peak at each joint location. These heatmaps could thus be automatically generated from the ground truth annotations we already had. Each block of the ResNet was modified to produce a predicted heatmap through the application of global and local attention, as described in [16] by Linsley et al. Ultimately, the predicted heatmaps would be multiplied by the block's original output, and the result would be fed into the next block. Predicted heatmaps were encouraged to be similar to the ground truth by adding the weighted L2 difference to our network's total loss function.

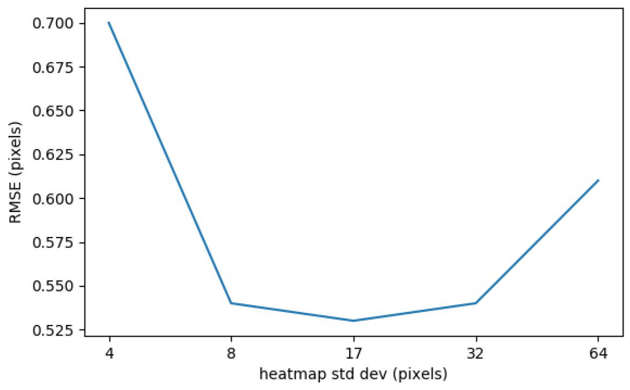
We applied this approach to heatmaps with Gaussian bumps of various spreads as well as different attention weights. We first tested various spreads using attention weight 1. We then selected the best spread parameter (17, in our case) and tested various attention weighting values. The results of these two searches are plotted in Figures 3a and 3b. Stricter attention masks, consisting of pure 1s and 0s as opposed to Gaussians, were also evaluated; however, the loss was less stable for this case, and performance was significantly worse. Best performance was achieved using heatmaps with standard deviation 17 and attention weight 1. Significantly, the ground-truth score maps used for DeepLabCut also had radius 17.

4.1.3 Saliency Maps with Scene Priors (Torralba et al.)

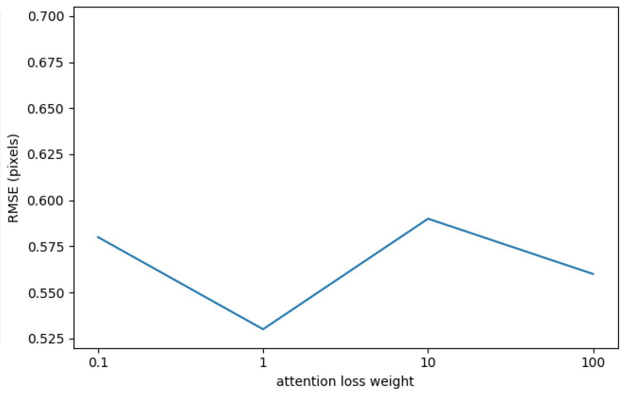
One final model that we evaluated used yet another variation of global and local attention. In contrast with the previous model, which uses purely local features to predict saliency, this model additionally uses a fully connected layer to weight the saliency of each pixel while taking into account all other pixels. The hope is that by learning scene priors relevant for locating different joints, much like Torralba et al. [17] proposes, we can reduce misdetections in far parts of the image.

In implementing this model, saliency maps were again produced using convolutional filters. Global scene priors were learned by passing our feature maps through a fully connected layer, which we hoped would activate areas in which joints appear most often, while inhibiting regions that usually contain only noise.

The best results from all three models, as well as the baseline, are expressed in Figure 4. These



(a)



(b)

Figure 3: For our attention augmentation, we plot the (a) the std. dev. of the Gaussian bumps used in producing ground-truth heat maps and (b) the weight attributed to the attention loss function versus the RMSE when evaluated on held-out frames from our 3-view dataset

Model	**RMSE (pixels)**
Baseline	0.58
Cropped	0.58
Linsley	0.53
Torralba	0.64

Figure 4: RMSE of various 2D refinement techniques evaluated on held-out frames from the 3-view dataset

were derived by evaluating each model on a held-out set of frames taken from the 3-view video session in our dataset and calculating the RMSE from the ground truth in pixels.

One can see that cropping the images showed little improvement over baseline. This was consistent with the observation that most of our 2D misdetections were still fairly close to the monkey and would have been located inside the image frame after cropping. The model that incorporated only saliency maps showed slight improvement, perhaps due to its ability to segment the image into more refined, non-rectangular regions, which could better exclude regions where misdetections occurred. Finally, the Torralba model appeared to do slightly worse than baseline. This could have been due to its more rigid scene priors; by learning that the monkey tended to be in one area, it would perform worse if the monkey were to change locations. The Torralba model also used significantly more parameters, which may have been difficult to accurately train given the limited size of our dataset.

4.2 Multiple View Techniques

In light of the limited success of 2D refinements in improving over the baseline, we also developed models to incorporate information from multiple views within the neural network itself, rather than waiting until the very last step. We wish for combining multiple views in this way to accomplish two things: First, it should allow clear detections of joints in some views to inform other views in which those joints may be occluded, explaining away the appearance of the joints elsewhere in the views with occlusions. Second, it should account for errors in the transformations between 2D and 3D coordinate systems, to produce more consistent 3D predictions.

One major issue in attempting to combine spatial information from multiple viewpoints within a convolutional neural network is the fact that spatial regions in one view will not necessarily correspond to the same region in another view. Nonetheless, Su et al.[13] shows that such information can be meaningfully combined for the related task of object classification, so we expect our models to be able to similarly take advantage of multiple views.

4.2.1 Confidence Score Re-Weighting

Our first attempt to combine information between views does so through a re-weighting of the confidence scores for the predictions from each view. We begin by concatenating all confidence scores for all views and joints into a vector, c. We then compute a new set of confidence scores, c', through the function:

$$c' = W_2 * \texttt{ReLU}(W_1 * c + b_1) + b_2$$

where W_1 and W_2 are square matrices of learned weights and b_1 and b_2 are vectors of learned biases.

This type of function has several desirable properties.

First, it allows the confidence levels of different predictions to interact with each other - for example, if one view makes a very confident prediction for joint j, this should reduce our confidence for a prediction made by a view on the opposite side of the setup, since joint j is unlikely to be visible to both sides at once. It also allows confidence levels for different views to remain independent, if the weights that would link them are set to zero.

Second, the `ReLU` function allows thresholds to be learned, beyond which predictions can be thrown out. This achieves what we had previously tried to accomplish manually through thresholds, and reduces sensitivity to unconfident predictions that could disproportionately sway our final results.

We did not expect this weighting function to meaningfully influence our convolutional filters earlier in the network, so we separated our training into two parts.

The first part was to train our baseline model using the same objective function as before. This allowed us to save time by simply reusing our previously trained baseline model.

We then froze the parameters of the baseline model and added our new function. A new objective function trains the parameters that we added, which is the mean squared error between the 3D pose prediction computed with these new confidence scores and the set of ground truth 3D coordinates. Our 3D ground truth labels were obtained simply by computing 3D points from every pairing of ground truth 2D points, and averaging them.

4.2.2 Cue Relocation

While confidence score re-weighting provides a more informed way of determining how much each prediction should be trusted in comparison to other predictions, it still does not allow us to share spatial cues between views to better refine 2D predictions. To achieve this, we propose a method that extracts relevant local patches from each view and spatially aligns them.

We start by computing the 3D predictions that correspond to every pair of views and every joint, $\rho_k^{(n_1,n_2)}$. We then use our camera projection matrices to project these 3D predictions onto every other camera's perspective. We let $p_k^{(n_1,n_2)\to n_3}$ denote the 2D prediction from the perspective of view n_3 that corresponds to the 3D point predicted by views n_1 and n_2.

For joint k, for each combination of 3 views (n_1, n_2, n_3), we then extract a 5×5 patch from the score map for view n_1 and joint k, centered at $p_k^{(n_1)}$. We insert this patch into a tensor of zeros such that in the new tensor, it is centered at $p_k^{(n_1,n_2)\to n_3}$. We do the same for n_2 as well. We denote these tensors $P_k^{n_1\to n_3}$ and $P_k^{n_2\to n_3}$, to represent predictions from views n_1 and n_2 that have been aligned with n_3. This process is illustrated in the left half of Figure 5.

For each view, we then concatenate all tensors that have been aligned to it, as well as its own score map tensor, along the last dimension. Finally, we perform convolutional operations to obtain new score maps and offset maps for that view. This is illustrated in the right half of Figure 5.

Our final 2D predictions for each view are computed in the same way as before, using the new score and offset maps. The objective function for any parameters present in the baseline model is also the same as before. To this, we add another separate objective, which is the same as the original objective function applied to the new score maps and offset maps, which tunes the parameters of the added convolutional operations.

We can draw some parallels between this model and the previously discussed model. Since a single convolutional filter is applied to each channel, and since each channel is either zeros or a patch containing the maximum confidence score for a particular joint and pair of views, the magnitudes of the values in each of our filters will re-weight the confidence scores provided by the other views. However, score maps and offset maps for each view are still computed through independent convolutional operations, so this model will not necessarily resolve the problem of inconsistent final confidence scores when constructing a single 3D prediction in the same way as the previous model.

Several variations of this model were also considered. As one alternative, we could extract patches from feature map tensors instead of score map tensors. One problem with this is the fact that there are many more channels in the feature map tensors, and we do not know which channels are most pertinent to which joints, if any. We would likely have to include all channels, which would greatly increase the number of tunable parameters in our convolutional operations, and which may not even all be relevant.

Another alternative could be to replace the local patches with maps where each pixel encodes an x and y offset from a point prediction. In other words, instead of computing $P_k^{n_1\to n_3}$ and $P_k^{n_2\to n_3}$, if $G_{x,y}$ is a grid of x and y pixel coordinates, we could compute offset maps $D = G_{x,y} - p_k^{(n_1,n_2)\to n_3}$, where our subtraction is applied to every element of $G_{x,y}$. A downside of this is that elements of D that are far from any predictions will be irrelevant, and our convolutional filters will have to learn to throw them out. Furthermore, these maps encode only a single piece of information, $p_k^{(n_1,n_2)\to n_3}$, disregarding any confidence levels. By extracting patches instead, we align more information. For example, the magnitude of a patch would indicate a more confident prediction, and a patch that is sharply spiked at one point is also a more confident prediction than a patch that is high in

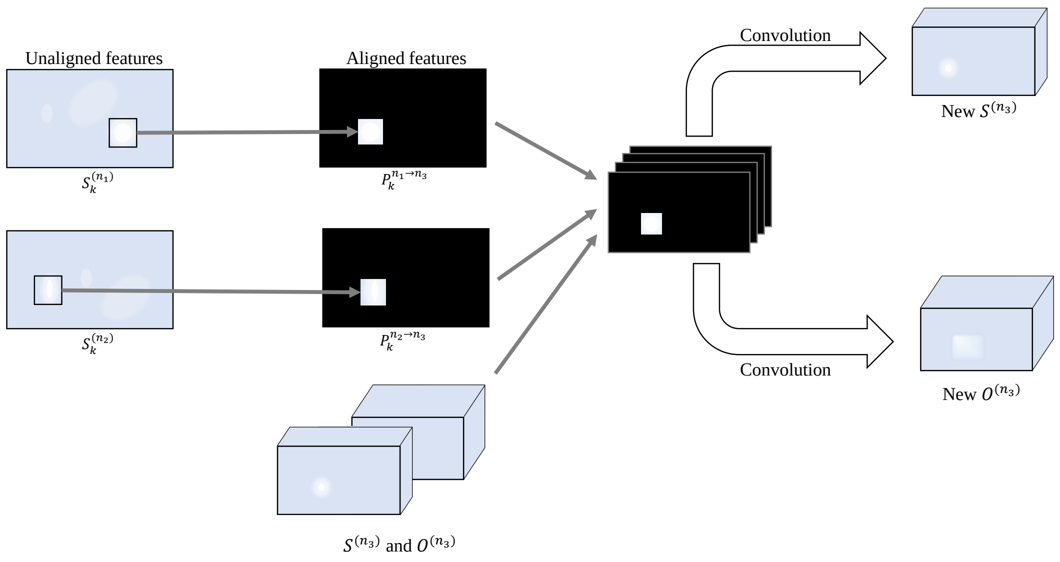


Figure 5: Cue Relocation: We first extract 5×5 patches from score map $S_k^{(n)}$ centered at p_k^n for each view n and joint k. Next, for every combination of a pair of views (n_1, n_2) and a third view n_3, we re-center these patches along $p_k^{(n_1,n_2)\to n_3}$ by inserting them into maps filled with zeros. Finally, we concatenate these aligned maps with the score maps and offset maps for n_3 and convolve them with a set of filters to obtain new score maps and offset maps for n_3.

magnitude but flat.

4.2.3 Direct 3D Loss

One final model we explored is the incorporation of a direct loss between 3D ground truth and predicted points. We hope that such a loss function will allow us to adjust for errors in projecting from 2D to 3D by changing the way the convolutional filters are tuned earlier in the network.

To create a practical loss function, we had to change how we made our 2D predictions. Previously, these were obtained by applying the non-differentiable argmax function to each confidence map. To minimize our objective using gradient descent, we need it to be differentiable. Therefore we replaced the argmax function with the differentiable softargmax function, defined in one dimension as:

$$\text{softargmax}(x) = \sum_{i=1}^{n} i \cdot \text{softmax}(x)_i$$

We also cannot index into our offset maps when extracting 2D predictions if we want our function to be completely differentiable. However, by using the softargmax function, we are now also no longer restricted to extracting integer-valued coordinates from our confidence maps, so we do not necessarily need our offset maps anymore, and we exclude them from this model.

Our 2D predictions for joint k and view n are therefore now computed as:

$$p_k^{(n)} = \text{softargmax}(P_{\cdot,\cdot,k}^{(n)})$$

where the softargmax function has been generalized to two dimensions.

Finally, we can no longer index into our confidence maps to obtain confidence levels for each prediction either. Instead, we use a differentiable approximation to the max function to approximate a maximum score map value, then apply the sigmoid function to approximate the maximum confidence value:

$$\hat{c}_k^{(n)} = \sum_{i,j} S_{i,j,k}^{(n)} \cdot \text{softmax}(S_{\cdot,\cdot,k}^{(n)})_{i,j}$$
$$c_k^{(n)} = \text{sigmoid}(\hat{c}_k^{(n)})$$

Using these new values for $p_k^{(n)}$ and $c_k^{(n)}$, 3D predictions are again computed according to Equation 2

Our objective function is now the sum of the original loss function for confidence maps (L_1) plus the mean squared error between the predicted and ground truth 3D points.

4.3 Results

Our results are visualized in Figure 6. Models were trained on the frames taken from the first half of the session recorded in our 6 view dataset and tested on frames taken from the second half. One can see that each model performs well in most cases; however the differing maximum error levels show that the Reweighting and Differentiable 3D models are better able to deal with outlier predictions.

Of note is the fact that the cue relocation model performed very similarly to the baseline, suggesting that it was unable to take advantage of the information in the aligned patches to improve 2D estimations. This could be due to a number of reasons. First, there were a large number of channels used when extracting the new score maps and offset maps, many of which may have contained all zeros (due to different joints' relevant patches being aligned to different locations). This may have caused the convolutional filters to only incorporate the original score maps and offset maps into the new maps, ignoring information from all other maps. Furthermore, inaccuracies in predictions from other viewpoints would cause the extracted patches to be misaligned. This variability may also have resulted in the network ignoring that information. Finally, it could simply be the case that not enough information was retained in a view's confidence maps to be relevant to other views, and extracting patches from the feature maps themselves might be necessary.

On the other hand, both the confidence score re-weighting and direct 3D loss models performed better than the baseline. Confidence score re-weighting allowed us to directly adjust for better consistency between our weighting of different predictions and outlier handling. Meanwhile, our 3D loss model drove the network toward better 2D predictions by affecting the tuning of the convolutional filters.

4.4 Conclusions

This work explores the approaches of combining visual cues across views at several different phases of processing.

The base model keeps view information separate until the very end, at which point single points and confidence scores are combined into the final prediction. The re-weighting model allows confidence scores from different views to in-

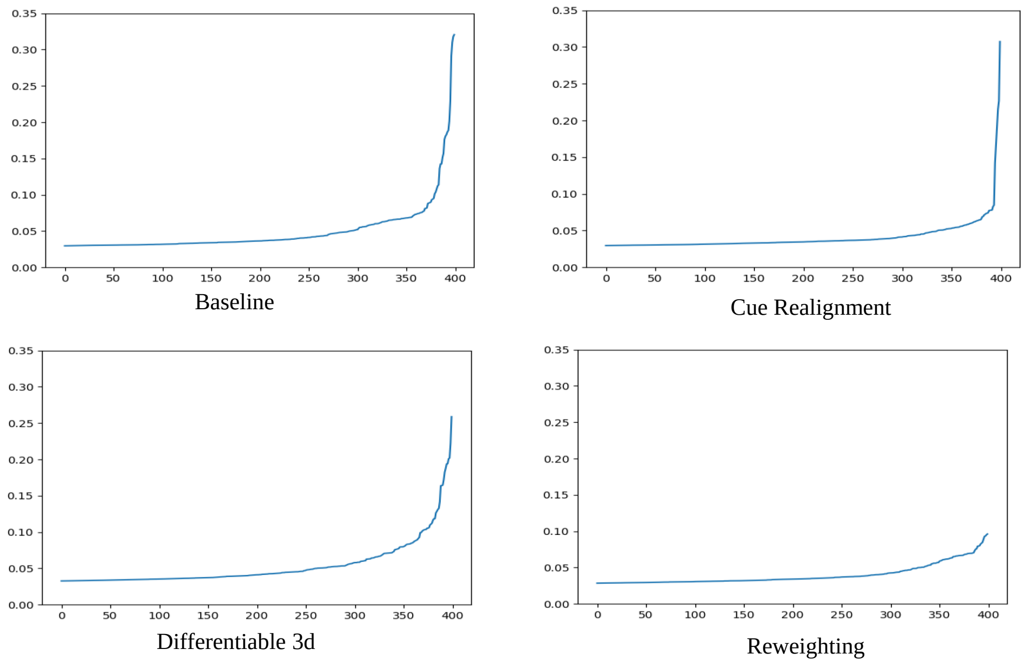


Figure 6: Plots of the error levels of the 500 highest-error predictions in meters that each of our models gave, evaluated on the last half of the 6 view session

teract right before the final prediction is made. The model with a direct 3D loss allows information between views to make an impact even earlier in the network, as the gradient from the final combined prediction can back-propagate up to the convolutional filters. Finally, the patch-relocation model explicitly combines information early in the network by learning convolutional filters that operate on information from all views at once.

4.5 Future Directions

One limitation of our models was that our ground truth annotations had to be projected from 2D to 3D. Small human errors would therefore cause individual 3D labels to be inconsistent with each other, which may have limited our models' accuracies. An interface to allow researchers to conveniently produce 2D annotations that are consistent with each other when projected to 3D may help alleviate this issue.

Another method to explore is whether information can be combined even earlier, perhaps before passing it through the network. Stereo matching between adjacent cameras could be used to obtain approximate depth maps, which we could use in conjunction with 2D predictions from individual cameras to go directly to 3D predictions. Lots of work has been done toward using RGB-D images in pose prediction: [14] tries to infer pose from a single depth map, [18] fuses multiple depth maps into a full 3D reconstruction that better handles errors in individual maps, [19] combines images from two RGB-D cameras to reduce occlusion during pose estimation. We did not go this route because accurate stereo matching itself is not easy, and in order to best take advantage of depth techniques, it may be more accurate to use camera hardware that is specialized in producing depth maps.

Finally, with respect to our models, nothing prevents us from combining the techniques we discussed here within one network. Future work might explore whether we can combine aspects of each of these models in the same network to achieve even better results.

4.6 Acknowledgements

I would like to thank Daniel Ritchie and Matthew Harrison for being my official advisors for this project - for helping to get it off the ground and for their feedback along the way. I would also like to thank David Borton and his lab for investing their time and labor to produce all the data and annotations for these experiments. I would like to thank the members of the Serre Lab, especially Drew, Lakshmi, and Kalpit, for their expertise and constant openness to discussing new ideas, and Zack, for introducing me to the Serre Lab and for his advice over the years. Finally, I want to thank Thomas Serre, for welcoming me into his lab four years ago, for providing me with guidance and projects, and for being the primary enabler with regard to my getting involved in research.

References

[1] E. Ceseracciu, Z. Sawacha, and C. Cobelli, “Comparison of markerless and marker-based motion capture technologies through simultaneous data collection during gait: proof of concept,” *PloS one*, vol. 9, no. 3, p. e87640, 2014.

[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields,” in *arXiv preprint arXiv:1812.08008*, 2018.

[3] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in *CVPR*, 2017.

[4] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping,” in *CVPR*, 2017.

[5] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” in *CVPR*, 2016.

[6] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. Murthy, M. Mathis, and M. Bethge, “Deeplabcut: markerless pose estimation of user-defined body parts with deep learning,” vol. 21, 08 2018.

[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[9] F. Bogo, A. Kanazawa, C. Lassner, P. V. Gehler, J. Romero, and M. J. Black, “Keep it SMPL: automatic estimation of 3d human pose and shape from a single image,”

CoRR, vol. abs/1607.08128, 2016. [Online]. Available: http://arxiv.org/abs/1607.08128

[10] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard, and T. Brox, "3d human pose estimation in RGBD images for robotic task learning," *CoRR*, vol. abs/1803.02622, 2018. [Online]. Available: http://arxiv.org/abs/1803.02622

[11] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, and M. W. Mathis, "Using deeplabcut for 3d markerless pose estimation across species and behaviors," *bioRxiv*, 2018. [Online]. Available: https://www.biorxiv.org/content/early/2018/11/24/476531

[12] M. Carraro, M. Munaro, J. Burke, and E. Menegatti, "Real-time markerless multi-person 3d pose estimation in rgb-depth camera networks," *CoRR*, vol. abs/1710.06235, 2017. [Online]. Available: http://arxiv.org/abs/1710.06235

[13] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[14] B. Tekin, P. Márquez-Neila, M. Salzmann, and P. Fua, "Fusing 2d uncertainty and 3d cues for monocular body pose estimation," *CoRR*, vol. abs/1611.05708, 2016. [Online]. Available: http://arxiv.org/abs/1611.05708

[15] M. Yin, D. A. Borton, J. Komar, N. Agha, Y. Lu, H. Li, J. Laurens, Y. Lang, Q. Li, C. Bull *et al.*, "Wireless neurosensor for full-spectrum electrophysiology recordings during free behavior," *Neuron*, vol. 84, no. 6, pp. 1170–1182, 2014.

[16] D. Linsley, D. Scheibler, S. Eberhardt, and T. Serre, "Global-and-local attention networks for visual recognition," *CoRR*, vol. abs/1805.08819, 2018. [Online]. Available: http://arxiv.org/abs/1805.08819

[17] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search." *Psychological review*, vol. 113 4, pp. 766–86, 2006.

[18] M. Ylimäki, J. Kannala, and J. Heikkilä, "Accurate 3-d reconstruction with RGB-D cameras using depth map fusion and pose refinement," *CoRR*, vol. abs/1804.08912, 2018. [Online]. Available: http://arxiv.org/abs/1804.08912

[19] S. Hong and Y. Kim, "Dynamic pose estimation using multiple rgb-d cameras," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/11/3865