This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Click Here: Human-Localized Keypoints as Guidance for Viewpoint Estimation

Ryan Szeto and Jason J. Corso
Electrical Engineering and Computer Science
University of Michigan
{szetor,jjcorso}@umich.edu

Abstract

We motivate and address a human-in-the-loop variant of the monocular viewpoint estimation task in which the location and class of one semantic object keypoint is available at test time. In order to leverage the keypoint information, we devise a Convolutional Neural Network called Click-Here CNN (CH-CNN) *that integrates the keypoint information with activations from the layers that process the image. It transforms the keypoint information into a 2D map that can be used to weigh features from certain parts of the image more heavily. The weighted sum of these spatial features is combined with global image features to provide relevant information to the prediction layers. To train our network, we collect a novel dataset of 3D keypoint annotations on thousands of CAD models, and synthetically render millions of images with 2D keypoint information. On test instances from PASCAL 3D+, our model achieves a mean class accuracy of 90.7%, whereas the state-of-the-art baseline only obtains 85.7% mean class accuracy, justifying our argument for human-in-the-loop inference.*

1. Introduction

It is well understood that humans and computers have complementary abilities. Humans, for example, are good at visual perception—even in rather challenging scenarios such as finding a toy in a cluttered room—and, consequently, subsequent abstract reasoning from visually acquired information. On the other hand, computers are good at processing large amounts of data quickly and with great precision, such as predicting viewpoints for millions of images within an exact, but possibly inaccurate, degree. Although we, as a community, design automatic systems that seek to extract information from images automatically—and have done this quite well, e.g., [9, 17]—there are indeed situations that are beyond the capabilities of current systems, such as inferring the extent of damage to two vehicles involved in a car accident from data acquired by a dash-cam.

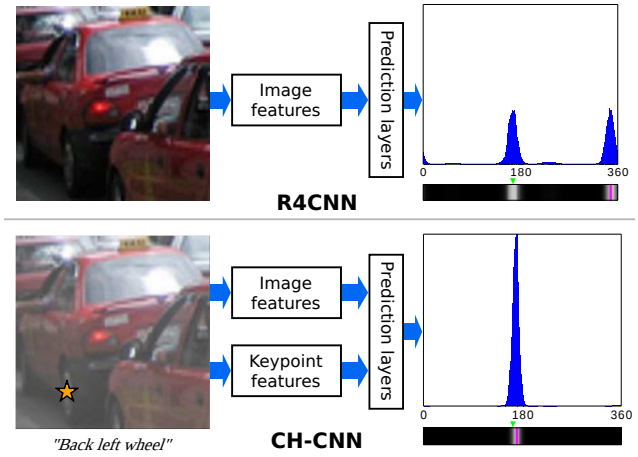


Figure 1: Semantic keypoint information can help address ambiguities that are difficult to resolve from the image alone. Each diagram shows the available information on the left, the high-level structure of the model in the middle, and the confidences of the azimuth angle on the right. In the black bars, gray indicates confidence, magenta marks the final prediction, and the green triangle marks the ground truth. The orange star indicates the human-provided keypoint. Both the light mask and orange star on the bottom left image are for visualization purposes only, and are not part of the input to any network.

In such exceptionally challenging cases, integrating the abilities of both humans and computers during inference is necessary; we call this methodology *hybrid intelligence*, borrowing a term from social computing [18]. This strategy can lead to pipelines that achieve better performance than fully automatic systems without incurring a significant burden on the human (Figure 1 illustrates such an example). Indeed, numerous computer vision researchers have begun to investigate tasks inspired by this methodology, such as learning on a budget [24] and Markov Decision Process-based fusion [20].

Continuing in this vein of work, we focus on integrating

the information provided by a human as additional input during inference to a novel convolutional neural network (CNN) architecture. We refer to this architecture as the *Click-Here Convolutional Neural Network*, or CH-CNN. In training, we learn how to best make use of the additional keypoint information. We develop a means to encode the location and identity of a single semantic keypoint on an image as the extra human guidance, and automatically learn how to integrate it within the part of the network that processes the image. The human guidance keypoint essentially determines a weighting, or attention mechanism [31], to identify particularly discriminative locations of information as data flows through the network. To the best of our knowledge, this is the first work to integrate such human guidance into a CNN at inference time.

To ground this work, we focus on the specific problem of monocular viewpoint estimation—the problem of identifying the camera's position with respect to the target object from a single RGB image. This challenging problem has applications in numerous areas such as automated driving, robotics, and scene understanding, many of which we envision a possible human-in-the-loop during inference. Although discriminative CNN-based methods have achieved remarkable performance on this task [23, 22, 14, 28], they often make mistakes when faced with three types of challenges: *occlusion*, *truncation*, and *highly symmetrical objects* [22]. In the first two cases, there is not enough visual information for the model to make the correct prediction, whereas in the third case, the model cannot identify the visual cues necessary to select among multiple plausible viewpoints.

Monocular viewpoint estimation is well-suited to our hybrid intelligence setup as humans can locate semantic keypoints on objects, such as the center of the left-front wheel on a car, fairly easily and with high confidence. CH-CNN is able to integrate such a keypoint directly into the inference pipeline. It computes a distance transform based on the keypoint location, combines it with a one-hot vector that indicates the keypoint class label, and then uses these data to generate a weight map that is combined with hidden activations from the convolutional layers that operate on the image. At a high level, our model learns to extract two types of information—global image information and keypoint-conditional information—and uses them to obtain the final viewpoint prediction.

We train CH-CNN with over 8,000 computer-aided design (CAD) models from ShapeNet [3] annotated with a custom, web-based interface. To our knowledge, our keypoint annotation dataset is an order of magnitude larger than the next largest keypoint dataset for ShapeNet CAD models [14] in terms of number of annotated models. As our thorough experiments show, we are able to use this human guidance to vastly improve viewpoint estimation performance: on human-guidance instances from the PASCAL 3D+ validation set [29], a fine-tuned version of the state-of-the-art model from Su et al. [22] achieves 85.7% mean class accuracy, while our CH-CNN achieves 90.7% mean class accuracy. Additionally, our model is well-suited for handling challenges that the state-of-the-art model often fails to overcome, as shown by our qualitative results.

We summarize our contributions as follows. First, we propose a novel CNN that integrates two types of information—an image and information about a single keypoint—to output viewpoint predictions; this model is designed to be incorporated into a hybrid-intelligence viewpoint estimation pipeline. Second, to train our model, we collect keypoint locations on thousands of CAD models, and use these data to render millions of synthetic images with 2D keypoint information. Finally, we evaluate our model on the PASCAL 3D+ viewpoint estimation dataset [29] and achieve substantially better performance than the leading state-of-the-art, image-only method, validating our hybrid intelligence-based approach. Our code and 3D CAD keypoint annotations are available on our project website at `ryanszeto.com/projects/ch-cnn`.

2. Related Work

Monocular Viewpoint Estimation. Viewpoint estimation and pose estimation of rigid objects have been tackled using a wide variety of approaches. One line of work has extended Deformable Part Models (DPMs) [7] to simultaneously localize objects and predict their viewpoint [29, 19, 8]. However, DPM-based methods can only predict a limited set of viewpoints, since each viewpoint requires a separate set of models. Patch alignment-based approaches identify discriminative patches from the test image and match them to a database of rendered 3D CAD models [1, 16]. More recent approaches have leveraged CNNs [5, 4, 28, 14, 23, 22], which achieve high performance without requiring the hand-crafted features used by earlier work. Additionally, unlike DPM-based approaches, CNNs extend easily to fine-grained viewpoints by regressing from the image to either a continuous viewpoint space [5, 4] or a discrete, but fine-grained space [23, 22]. Even better performance can be achieved by supervising the CNN training stage with intermediate representations [28, 14]. Nonetheless, most fully-automatic approaches struggle from three specific challenges: occlusion [29, 22, 1], truncation [29, 22], and highly symmetric objects [22, 16]. As we show in Section 5, CH-CNN helps reduce the error caused by these challenges.

Human Interaction for Vision Tasks. Most prior work in the vision community on integrating information from humans at inference time are examples of either active learning or dynamic inference. Active learning approaches reduce the amount of labeled data required for sufficient per-

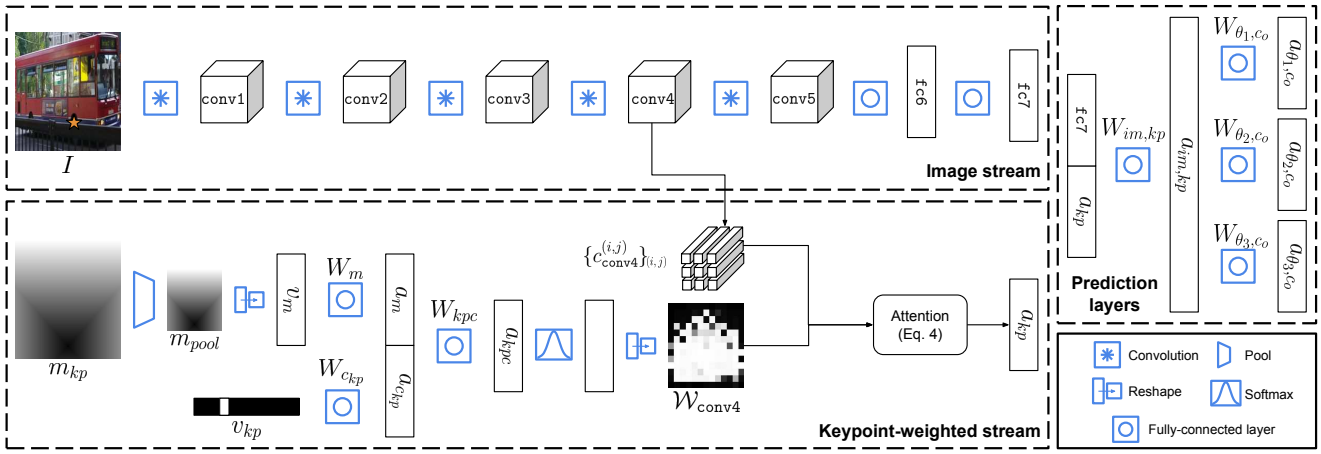


Figure 2: The architecture for CH-CNN. A weighting over the conv4 activation depth columns is learned by taking linear transformations of the keypoint data and applying a softmax operation to the result. The keypoint features are obtained by taking the sum of each activation depth column weighted by the corresponding value in the weight map. These features are concatenated to the fc7 image features to aid with inference. The orange star only visualizes the keypoint in this figure; it is not used as input to the network.

formance by intelligently selecting unlabeled instances for the human to annotate [24, 25, 24, 15]. Our task differs from active learning in that the information from the human (the keypoint) is available at *inference time* rather than *training time*, and we leverage auxiliary human information to improve the accuracy of our model rather than to achieve sufficient performance with fewer examples. In dynamic inference, a system proposes questions with the goal of improving the confidence or quality of its final answer [20, 2, 26, 27, 10]. This line of work has demonstrated the potential of incorporating human input at inference time. Contrasting with work in dynamic inference, which emphasizes the process of selecting questions for the human to answer, we focus on the problem of learning how to integrate answers in an end-to-end approach for viewpoint estimation CNNs.

3. Click-Here CNN for Viewpoint Estimation

Our goal is to estimate three discrete angles that describe the rotation of the camera about a target object, where we are given a tight crop of the object, the location of a visible keypoint in the image, and the keypoint class (e.g. the center of the front right wheel, for a car). We do so with a novel CH-CNN that outputs confidences for each possible angle.

Formally, let $I \in \mathbb{R}^{h \times w \times 3}$ be a single RGB image, (x, y) be the 2D coordinate of the provided keypoint location in the image, and c_{kp} be the keypoint class. The label c_{kp} can take on one of $\sum_{c_o \in \mathcal{C}_o} |\mathcal{C}_{kp}(c_o)|$ values, where $\mathcal{C}_o$ is the set of object classes and $\mathcal{C}_{kp}(c_o)$ is the set of keypoint classes for a given object class c_o. Furthermore, for a given instance $s = (I, x, y, c_{kp}, c_o)$, let $\theta_{gt} = (\theta_1, \theta_2, \theta_3)$ be a tuple associated with s representing the ground-truth azimuth/longitudinal rotation, elevation/latitudinal rotation, and in-plane rotation of the camera with respect to the object's canonical coordinate system; each angle is discretized into N bins (following Su et al. [22], we consider $N = 360$). For each object class c_o, we seek a probability distribution function $P(\theta|s)$ that is maximized at θ_{gt} for any instance s. We approximate this set of functions with our CH-CNN.

Prior work [23, 22] has explored the case where $s = (I, c_o)$, i.e. the image and object class are available at test time, by fine-tuning popular CNN architectures such as AlexNet [13] and VGGNet [21]. Note that after fine-tuning, the intermediate activations of these models can be interpreted as image features that are useful for viewpoint estimation [22]. In our case, we have access to additional information at test time, i.e. the keypoint location (x, y) and class c_{kp}. We believe that for viewpoint estimation, this information can be used to produce features that complement the global image features extracted from popular CNN architectures. We incorporate this idea in CH-CNN by learning to weigh features from certain regions in the image more heavily based on the keypoint information.

Figure 2 illustrates the architecture of CH-CNN. The early layers of our architecture are divided into two streams: the first generates features from the image, and the second produces "keypoint features" to complement the high-level image features. The keypoint feature stream produces features in three steps. First, a weight map is produced by passing the keypoint map and class through a series of linear transformations and taking the softmax of the result. Second, the activation depth columns from a convolutional layer (conv4 in our case) are multiplied by the correspond-

ing weights from the weight map. Finally, the keypoint features are created by taking the sum of the weighted columns.

CH-CNN concatenates the features from the image and keypoint streams and performs inference with one fully-connected hidden layer and one prediction layer for each angle. The fact that we seek a probability distribution function for each object class suggests that a separate network must be trained for each object class. To avoid this, we adopt the approach used in Su et al. [22] where lower-level feature layers are shared by all object classes, and object class-dependent prediction layers are used for each angle.

3.1. Implementation of CH-CNN

We implement the image stream of CH-CNN with the hidden layers of AlexNet [13] (i.e. the layers up to the second fully-connected layer `fc7`); we take the activations of the `fc7` layer as our image features. We stress that while AlexNet is a less powerful model than more recent ones such as ResNet [9], our choice allows for a sensible comparison with Su et al. [22], who fine-tune the same architecture for viewpoint estimation. Additionally, the choice of architecture used for the image stream is independent of our primary contribution, which is to leverage the additional guidance from the provided keypoint at inference time.

The keypoint feature stream takes representations of (x, y) and c_{kp} and generates a weighting over activation depth columns from a convolutional layer in the image stream (the fourth layer `conv4` in our case), where spatial, but high-level information is retained. We use $c_{conv4}^{(i,j)}$ to denote the column at position (i, j) in the `conv4` activation depth column grid. We represent (x, y) with a matrix $m_{kp} \in \mathbb{R}^{s \times s}$, where each entry $m_{kp}^{(i,j)}$ is the Chebyshev distance of (i, j) from (x, y) divided by the largest possible distance from the keypoint; the label c_{kp} is represented with a one-hot vector encoding v_{kp}.

To learn weights over the activation depth columns, we first learn keypoint map features by downsampling m_{kp} with max pooling, and applying a linear transformation to the vectorized result:

$$\begin{aligned} m_{pool} &= \text{pool}(m_{kp}) \\ v_m &= \text{flatten}(m_{pool}) \\ a_m &= W_m v_m \ . \end{aligned} \tag{1}$$

Similarly, features from the keypoint class vector are obtained with a linear transformation:

$$a_{c_{kp}} = W_{c_{kp}} v_{kp} \ . \tag{2}$$

Finally, the weight map for the `conv4` activation depth columns $\mathcal{W}_{\texttt{conv4}}$ is obtained by linearly transforming the concatenated keypoint features, applying the softmax function, and reshaping the result to match the shape of the `conv4` activation depth column grid $(h_{\texttt{conv4}}, w_{\texttt{conv4}})$:

$$\begin{aligned} a_{kpc} &= W_{kpc}[a_m^\top \, a_{c_{kp}}^\top]^\top \\ \mathcal{W}_{\texttt{conv4}} &= \text{reshape}(\text{softmax}(a_{kpc}), (h_{\texttt{conv4}}, w_{\texttt{conv4}})) \ . \end{aligned} \tag{3}$$

The keypoint feature vector a_{kp} is the sum of the `conv4` activation depth columns weighted by $\mathcal{W}_{\texttt{conv4}}$:

$$a_{kp} = \sum_{i=1}^{h_{\texttt{conv4}}} \sum_{j=1}^{w_{\texttt{conv4}}} {\mathcal{W}_{\texttt{conv4}}}^{(i,j)} c_{conv4}^{(i,j)} \ , \tag{4}$$

where i and j index into $\mathcal{W}_{\texttt{conv4}}$ and the `conv4` activation depth column grid.

To perform inference, $a_{\texttt{fc7}}$ and a_{kp} are concatenated. The result is passed through one non-linear hidden layer with an activation function σ (e.g. the rectified linear activation function) and a set of class-wise prediction layers for each angle θ_j:

$$\begin{aligned} a_{im,kp} &= \sigma(W_{im,kp}[a_{kp}^\top \, a_{\texttt{fc7}}^\top]^\top) \\ a_{\theta_j,c_o} &= W_{\theta_j,c_o} a_{im,kp}, \qquad\qquad j \in \{1, 2, 3\} \ . \end{aligned} \tag{5}$$

3.2. Training

To train our network, we use the geometric structure aware loss function from Su et al. [22],

$$L_\theta(S) = -\sum_{s \in S} \sum_{\theta \in \Theta} e^{-d(\theta, \theta_{gt})/t} \log P(\theta|s) \ , \tag{6}$$

where $s = (I, x, y, c_{kp}, c_o)$ is a sample from object class c_o, S is the set of training instances, Θ is the set of possible viewpoints, $P(\theta|s)$ is the estimated probability of θ given instance s, $d(\theta, \theta_{gt})$ is a distance metric between viewpoints θ and θ_{gt} (e.g. the geodesic distance defined in Sec. 5.1), and t is a hyperparameter that tunes the cost of an inaccurate prediction. This loss is a modification of the cross-entropy loss that encourages correlation between the predictions of nearby views.

To train the network, we begin by generating sets of training instances from synthetic data from ShapeNet [3] and real-world data from the PASCAL 3D+ dataset [29] (see Section 4 for details). Then, we initialize the layers from AlexNet with the weights learned from Su et al. [22]; the layers in the keypoint feature stream $W_m, W_{c_{kp}}, W_{kpc}$, as well as the prediction layers $W_{im,kp}$ and W_{θ_j,c_o}, are initialized with random weights. Next, we train on the synthetic data until the validation performance on a held-out subset of the synthetic data plateaus. Finally, we fine-tune on the real-world training data until the loss on that data plateaus. We develop and train our models in Caffe [11].

4. Generating Data for CH-CNN

The annotations available in the PASCAL 3D+ dataset [29] allow us to generate about 14,000 training instances from real-world images (see Section 4.1 for details

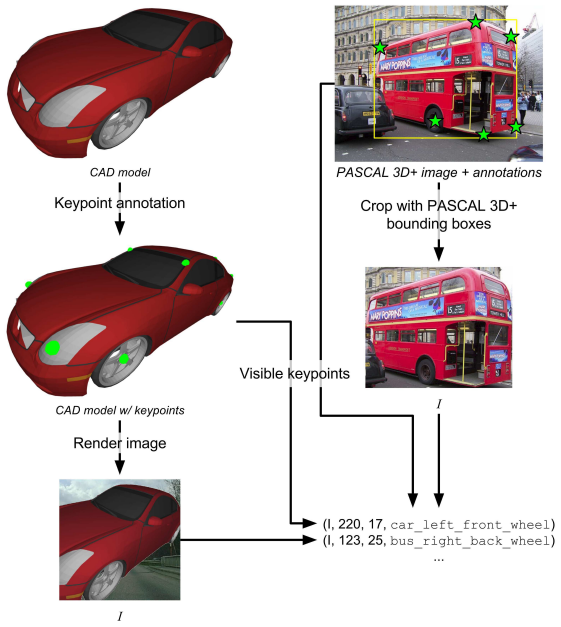


Figure 3: The pipeline for generating synthetic training data (left) and real-world training data (right).

on this process), but this number is insufficient for training CH-CNN. To overcome this limitation, we have extended the synthetic rendering pipeline proposed by Su et al. [22] to generate not only synthetic images with labels, but also 2D keypoint locations, resulting in about two million synthetic training instances. Because this procedure requires knowledge of the 3D keypoint locations on CAD models, we have collected keypoint annotations on 918 bus, 7,377 car, and 320 motorcycle models from the CAD model repository ShapeNet [3] with the use of an in-house annotation interface (refer to the supplemental material for details on the CAD model filtering and annotation collection processes). We focus on vehicles to help advance applications in automotive settings, but note that our method is applicable to any rigid object class with semantic keypoints. To the best of our knowledge, the number of annotated CAD models in our dataset is greater than ten times that of the next largest ShapeNet-based keypoint dataset from Li et al. [14], who collected keypoints on 472 cars, 80 chairs, and 80 sofas. Our annotated CAD models are publicly available on our project website.

4.1. Dataset Details

We render images of the annotated CAD models using the same pipeline used in Su et al. [22], which we now describe here. First, we randomly sample light sources and camera extrinsics. Then, we render the CAD model over a random background from the SUN397 dataset [30] to reduce overfitting to synthetic instances. Finally, we crop the object with a randomly perturbed bounding box. From a single rendered image I, we generate one instance of the form (I, x, y, c_{kp}, c_o) with label θ_{gt} for each visible keypoint, which can be identified by ray-tracing in the rendering environment. We focus on visible keypoints because in the hybrid intelligence environment, we assume that the human locates unambiguous keypoints, which disqualifies occluded and truncated keypoints. We follow this approach to generate about two million synthetic training instances.

PASCAL 3D+ provides detailed annotations that make generating labeled instances a straightforward process. To obtain instance-label pairs from PASCAL 3D+, we extract ground-truth bounding box crops of every vehicle in the dataset. For each cropped vehicle image I and ground-truth keypoint contained inside I that is labeled as visible, we produce one labeled instance. We augment the set of training data by horizontally flipping and adjusting (x, y), c_{kp}, and θ_{gt} appropriately. In total, we extract about 14,000 training instances and 7,000 test instances from the PASCAL 3D+ training and validation sets, respectively.

5. Experiments

We conduct experiments to compare image-only viewpoint estimation with our human-in-the-loop approach, as well as analyze the impact of keypoint information on our model. First, we quantitatively compare our model against the state-of-the-art model R4CNN [22] on the three vehicle object classes in PASCAL 3D+ (Section 5.1). Second, we analyze the influence of the keypoint information on our model via ablation tests and perturbations in the keypoint location at inference time (Section 5.2). Finally, we provide qualitative results to compare our model's predictions to those made by R4CNN (Section 5.3).

5.1. Comparison to Image-Only Models

We compare multiple viewpoint estimation models by evaluating their performance on instances extracted from the PASCAL 3D+ validation set [29]. To be consistent with prior work [23, 22], we report two metrics, $Acc_{\pi/6}$ and $MedErr$, which are defined as follows. Let $\Delta(R_{pr}, R_{gt}) = \frac{||\log(R_{pr}^\top R_{gt})||_F}{\sqrt{2}}$ be the geodesic distance between the predicted rotation matrix R_{pr} and the ground-truth rotation matrix R_{gt} on the manifold of rotation matrices. We define $Acc_{\pi/6}$ as the fraction of test instances where $\Delta(R_{pr}, R_{gt}) < \pi/6$ in radians, and $MedErr$ as the median value of $\Delta(R_{pr}, R_{gt})$ in degrees over all test instances.

Table 1 summarizes the performance of various models on the instances extracted from the PASCAL 3D+ validation set. We include R4CNN with and without fine-tuning (Section 3.2) to account for the difference in object classes used in Su et al. [22]. We also compare against two baselines that use a fixed weight map for $\mathcal{W}_{\text{conv4}}$ (Equation 4)

	$Acc_{\pi/6}$				$MedErr$			
	bus	car	motor	*mean*	bus	car	motor	*mean*
R4CNN [22]	92.4	78.5	81.4	84.1	5.04	7.86	14.5	9.14
R4CNN [22], fine-tuned	90.6	82.4	84.1	85.7	2.93	5.63	11.7	6.74
Keypoint features (Gaussian fixed attention)	88.9	81.3	82.8	84.4	3.00	5.88	11.4	6.76
Keypoint features (uniform fixed attention)	90.6	82.0	83.7	85.4	3.01	5.72	12.1	6.93
CH-CNN (keypoint map only)	90.6	82.0	84.2	85.6	3.04	5.73	11.3	6.68
CH-CNN (keypoint class only)	90.9	86.3	83.1	86.8	2.92	5.29	**11.0**	6.41
CH-CNN (keypoint map + class)	**96.8**	**90.2**	**85.2**	**90.7**	**2.64**	**4.98**	11.4	**6.35**

Table 1: PASCAL 3D+ performance for R4CNN [22] with and without fine-tuning on our data, models using a fixed activation depth column weight map, and variants of our CH-CNN model. The CH-CNN models weigh the `conv4` columns based on the keypoint map, the keypoint class, or both. See Section 5.1 for details on the reported metrics.

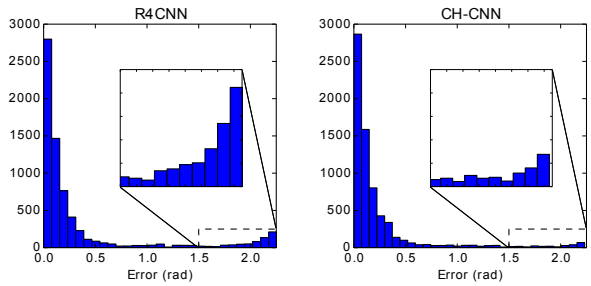


Figure 4: Distribution of angle error across all classes from fine-tuned R4CNN and our model. In each graph, the area in the dashed box is blown up for clarity.

instead of learning attention from the keypoint data. The first baseline (Gaussian fixed attention) sets $\mathcal{W}_{\texttt{conv4}}$ to a normalized 13 $\times$ 13 Gaussian kernel with a standard deviation of 6, and the second baseline (uniform fixed attention) sets $\mathcal{W}_{\texttt{conv4}}$ to a 13 $\times$ 13 box filter. Aside from the baselines, we evaluate three versions of our CH-CNN model described in Section 3.1. The first two learn a weight map using either the keypoint map or the keypoint class vector exclusively, and the third is our full model that integrates both sources of information into the weight map computation.

As shown in Table 1, our full CH-CNN model obtains the highest accuracies out of all tested models by a wide margin; noticeable drops in median error also occur. A conclusion that we draw from these results is that a weighted sum of feature columns can help improve viewpoint estimates. Most importantly, *learning to weigh these features based on the keypoint information is critical to substantially improving performance over image-only methods.* This indicates that providing a single keypoint during inference can indeed help viewpoint estimation by providing features that compliment those extracted solely from the image.

Figure 4 shows the histograms of angle errors across all object classes obtained by our full CH-CNN model and fine-tuned R4CNN (we refer to this model simply as R4CNN for the remainder of the paper). The most notable difference between the two error distributions occurs along the tails: CH-CNN obtains high errors noticeably less frequently than

Keypoint	R4CNN f.t.	CH-CNN	% ↑
Left front wheel	86.9	89.5	2.99
Left back wheel	80.6	89.0	10.4
Right front wheel	89.4	91.2	2.01
Right back wheel	85.9	90.8	5.70
Left front light	90.5	94.5	4.42
Right front light	**93.2**	**95.5**	*2.47*
Left front windshield	87.3	91.0	4.24
Right front windshield	88.9	91.7	3.15
Left back trunk	76.8	89.5	16.5
Right back trunk	72.8	88.0	20.9
Left back windshield	72.1	*84.7*	17.5
Right back windshield	*70.8*	87.6	**23.7**
Overall	82.4	90.2	9.47

Table 2: Values of $Acc_{\pi/6}$ for the fine-tuned R4CNN model [22] and CH-CNN, stratified by car keypoint class. The % ↑ column lists relative percent increase in $Acc_{\pi/6}$ of CH-CNN over R4CNN. The smallest value in each column is italicized, and the largest value is bolded.

R4CNN, which we attribute to our model's ability to take advantage of keypoint features when the image features are not informative enough to make a good estimate.

Table 2 stratifies performance by car keypoint classes. In all cases, our model estimates the viewpoint more accurately than R4CNN. However, relative improvement varies greatly, meaning that if certain keypoints can be provided, the improvement from using our model over R4CNN will become more apparent. For instance, CH-CNN yields the greatest relative increase in accuracy when the right back windshield keypoint is provided, but the lowest relative improvement when the right front light keypoint is provided. We attribute this difference to the varying amount of visual information that an image-only system can leverage, which depends on which keypoints are visible: front lights are often more visually distinguishable from their rear counterparts than windshield corners are to their front counterparts. Stratified performance for bus and motorcycle keypoints can be found in the supplementary materials.

5.2. Sensitivity to Keypoint Information

In this section, we explore how changing the keypoint information at inference time affects our trained CH-CNN

	KPM	KPC	bus	car	mbike	mean
$Acc_{\pi/6}$	✘	✘	75.1	67.2	80.0	74.1
$Acc_{\pi/6}$	✘	✓	78.0	79.4	81.8	79.7
$Acc_{\pi/6}$	✓	✘	89.2	77.2	82.9	83.1
$Acc_{\pi/6}$	✓	✓	**96.8**	**90.2**	**85.2**	**90.7**
$MedErr$	✘	✘	3.81	8.00	12.1	7.98
$MedErr$	✘	✓	3.68	6.03	12.1	7.27
$MedErr$	✓	✘	2.92	6.08	11.9	6.97
$MedErr$	✓	✓	**2.64**	**4.98**	**11.4**	**6.35**

Table 3: Impact of blank keypoint data on predictions. The KPM and KPC columns respectively indicate whether the ground-truth keypoint map or class was used. ✘ indicates that a blank keypoint map or keypoint class vector was used.

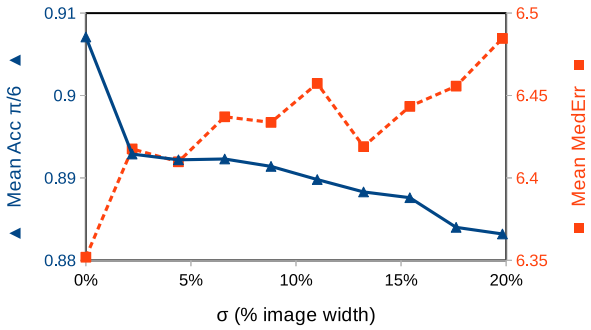
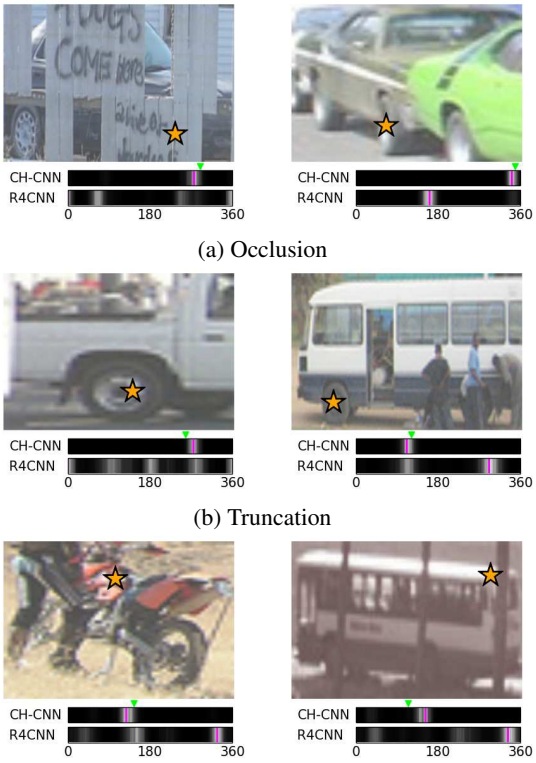


Figure 5: Sensitivity of CH-CNN to perturbations in the keypoint map. The mean class accuracy is plotted with a solid curve, and the mean class median error is plotted with a dashed curve.

model. To argue that CH-CNN adapts to the keypoint features rather than ignoring them in favor of the image features, we experiment with providing a keypoint map of all zeros, a keypoint class vector of all zeros, or both to our trained model at test time. As shown in Table 3, CH-CNN attains the worst performance when both the keypoint map and class vector are blank. In the cases where either the keypoint map or class is available, but not both, the model achieves better performance. Finally, the best performance is obtained by providing both sources of information. These results indicate that our model adapts to the keypoint information, rather than relying solely on the image features.

Next, we demonstrate that CH-CNN is robust to noise in the keypoint location at inference time, which is required in order to be useful for the hybrid intelligence environment. The noise is modeled by sampling the keypoint location from a 2D Gaussian whose mean is at the true keypoint location. We accomplish this by creating a new test set for each standard deviation σ as follows. We replace each instance (I, x, y, c_{kp}, c_o) from the PASCAL 3D+ validation set with one instance of the form (I, x', y', c_{kp}, c_o), where $[x', y']^\top \sim \mathcal{N}([x, y]^\top, \sigma^2 \mathbf{I}_2)$. Here, $\mathbf{I}_2$ is the 2×2 identity matrix and σ parameterizes the covariance matrix.

In Figure 5, we plot the mean class performance of CH-CNN as σ increases. We see that our model is robust to misplaced keypoints, retaining over 98% of its maximum performance even when the standard deviation is about 20% of the image dimensions. This is likely due to our method of downsampling the keypoint map, which would map the perturbed keypoint to a similar depth column weight map.



(a) Occlusion

(b) Truncation

(c) High symmetry

Figure 6: Visualization of challenging instances. Each grayscale bar is the azimuth confidence across all 360 degrees for a model. The green triangle marks the ground truth, and each magenta line marks a final prediction. The light masks and orange stars are for visualizing the keypoint location in this figure only, and are not part of the input to any network.

5.3. Qualitative Results

To conclude our analysis, we present qualitative comparisons between CH-CNN and R4CNN [22] by illustrating the confidences across azimuth, the most challenging angle to predict for PASCAL 3D+ [29]. In Figure 6, we compare the two models for images that exhibit either occlusion, truncation, or highly symmetric objects, observing that CH-CNN tends to estimate viewpoint more robustly than R4CNN under these circumstances. In the shown examples, our model estimates a narrow band around the true azimuth with high confidence. On the other hand, R4CNN exhibits a variety of behaviors, such as multiple peaks (all rows, left), wide bands (middle row, left), or high confidence for the angle opposite the true azimuth (top row, right). We attribute the

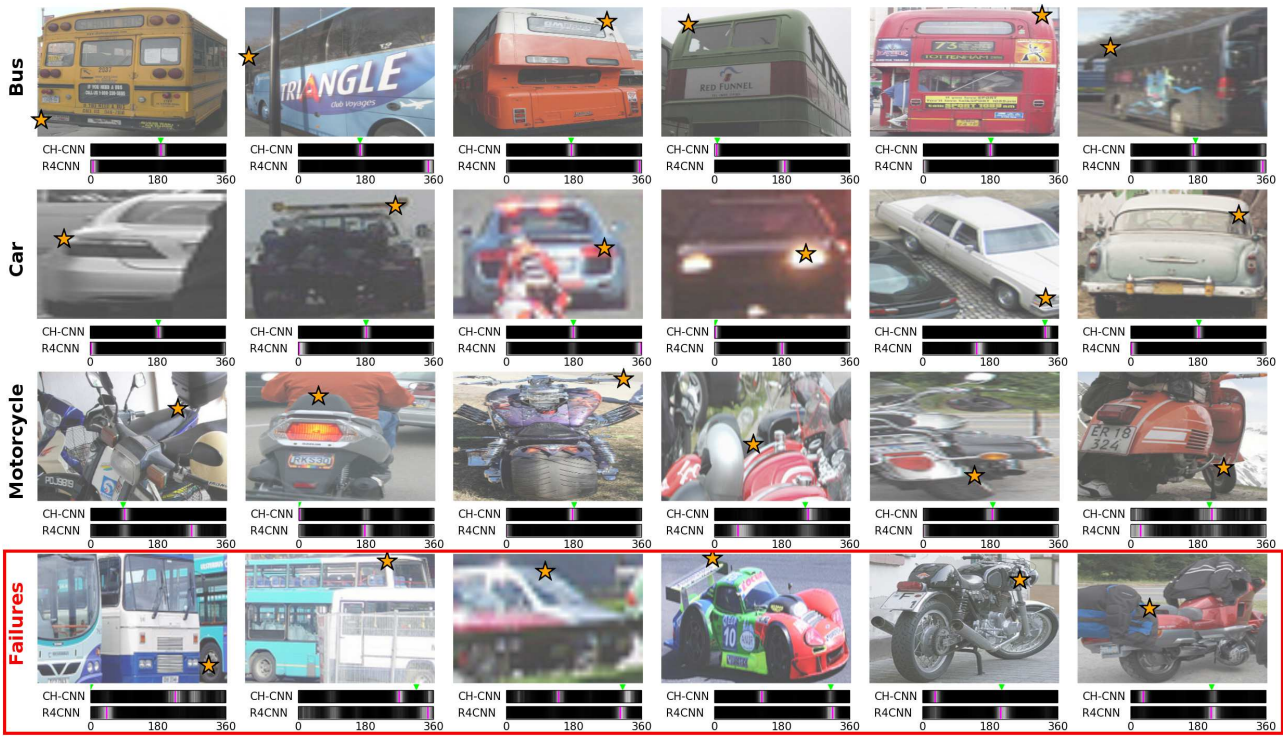


Figure 7: Azimuth confidences across all object classes, as well as failure cases where our model made an incorrect prediction. See Figure 6 for a description of each plot.

relative improvement of CH-CNN to the keypoint features, which can help suppress contradictory viewpoint estimates.

Figure 7 includes multiple examples of each object class, as well as failure cases for our model. In the positive cases, we continue to see narrower, but more accurate, bands of high confidence from CH-CNN than from R4CNN. Although the negative cases show that CH-CNN does not entirely overcome the main challenges of viewpoint estimation, the improved performance as shown in Table 1 indicates that these factors impact our model less severely than they impact R4CNN.

6. Conclusion

Limitations and Suggestions. Our work makes a few critical assumptions that are worth addressing in future work. First, we assume that information about only one keypoint is provided; in reality, we should be able to leverage multiple keypoints to further improve the estimate. Second, we assume that viewpoint estimates of the same object with different keypoint data are unrelated, whereas a better approach would be to enforce the consistency of viewpoint estimates of the same object. Third, we assume that the provided keypoint is both unoccluded and within the object bounding box. However, this is sensible in the context of hybrid intelligence because we can trust the human to suggest unambiguous keypoints or indicate that none exist, in which case we can fall back on image-only systems.

Summary. We have presented a hybrid intelligence approach to monocular viewpoint estimation called CH-CNN, which leverages keypoint information provided by humans at inference time to more accurately estimate the viewpoint. Our method combines global image features with keypoint-conditional features by learning to weigh feature activation depth columns based on the keypoint information. We train this model by generating synthetic examples from a new, large-scale 3D keypoint dataset. As shown by our experiments, our method vastly improves viewpoint estimation performance over state-of-the-art, image-only systems, validating our argument that applying hybrid intelligence to the domain of viewpoint estimation can yield great benefits with minimal human effort. To spur further work in hybrid intelligence for 3D scene understanding, we have made our code and keypoint annotations available at ryanszeto.com/projects/ch-cnn.

Acknowledgements. We thank Vikas Dhiman, Luowei Zhou, and Madan Ravi Ganesh for their helpful discussions and management of computing resources. We also thank Alex Miller, Matthew Dorow, Bhavika Reddy Jalli, Hojun Son, Guangyu Wang, Ronald Scott, and the other student annotators for collecting the keypoint dataset. This work was partially supported by the Denso Corporation, NSF CNS 1463102, and DARPA W31P4Q-16-C-0091.

References

[1] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[2] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, 2010. 3

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, and others. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 4, 5

[4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[5] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, 2015. 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 2

[8] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in Neural Information Processing Systems*, 2012. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 4

[10] S. D. Jain and K. Grauman. Click carving: Segmenting objects in video with point clicks. *arXiv preprint arXiv:1607.01115*, 2016. 3

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. *arXiv preprint arXiv:1408.5093*, 2014. 4

[12] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 3, 4

[14] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker. Deep Supervision with Shape Concepts for Occlusion-Aware 3d Object Parsing. *arxiv*, abs/1612.02699, 2016. 2, 5

[15] L. Liang and K. Grauman. Beyond comparing image pairs: Setwise active learning for relative attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 3

[16] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *IEEE International Conference on Computer Vision*, 2013. 2

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016. 1

[18] D. Merritt, J. Jones, M. S. Ackerman, and W. S. Lasecki. Kurator: Using the crowd to help families with personal curation tasks. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 1835–1849, New York, NY, USA, 2017. ACM. 1

[19] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 2

[20] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 3

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[22] H. Su, C. R. Qi, Y. Li, and L. J. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3d Model Views. In *IEEE International Conference on Computer Vision*, 2015. 2, 3, 4, 5, 6, 7

[23] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 3, 5

[24] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 1, 3

[25] C. Vondrick and D. Ramanan. Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems*, 2011. 3

[26] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *IEEE International Conference on Computer Vision*, 2011. 3

[27] C. Wah, G. Van Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 3

[28] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman. Single image 3d interpreter network. In *European Conference on Computer Vision*, 2016. 2

[29] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*. IEEE, 2014. 2, 4, 5, 7

[30] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2010. 5

[31] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, Attend and

Tell: Neural Image Caption Generation with Visual Attention. 2015. 2