Active Annotation Translation

Steve Branson
Caltech
sbranson@caltech.edu

Kristján Eldjárn Hjörleifsson
University of Iceland
keh4@hi.is

Pietro Perona
Caltech
perona@caltech.edu

Abstract

We introduce a general framework for quickly annotating an image dataset when previous annotations exist. The new annotations (e.g. part locations) may be quite different from the old annotations (e.g. segmentations). Human annotators may be thought of as helping translate the old annotations into the new ones. As annotators label images, our algorithm incrementally learns a translator from source to target labels as well as a computer-vision-based structured predictor. These two components are combined to form an improved prediction system which accelerates the annotators' work through a smart GUI.

We show how the method can be applied to translate between a wide variety of annotation types, including bounding boxes, segmentations, 2D and 3D part-based systems, and class and attribute labels. The proposed system will be a useful tool toward exploring new types of representations beyond simple bounding boxes, object segmentations, and class labels, and toward finding new ways to exploit existing large datasets with traditional types of annotations like SUN [36], Image Net [11], and Pascal VOC [12]. Experiments on the CUB-200-2011 and H3D datasets demonstrate 1) our method accelerates collection of part annotations by a factor of 3-20 compared to manual labeling, 2) our system can be used effectively in a scheme where definitions of part, attribute, or action vocabularies are evolved interactively without relabeling the entire dataset, and 3) toward collecting pose annotations, segmentations are more useful than bounding boxes, and part-level annotations are more effective than segmentations.

1. Introduction

Datasets for object recognition research have traditionally focused mostly on image class labels, bounding boxes, and object-level segmentations. Should we be exploring other possible choices of ways to annotate images? The decision of how to annotate data affects how image features can be extracted (*e.g.*, from a full image vs. from a bounding box vs rotatable parts, *etc.*), which types of statistical models admit computationally tractible learning problems (*e.g.*, the convexity and NP-hardness properties of incorporating different types of alignment models, sharing models for transfer learning, *etc.*), what types of applications we can solve (*e.g.*, do we want to solve problems beyond predicting object class labels and bounding boxes?), the amount of time it takes humans to annotate an image, how consistently different humans can provide annotations (*e.g.*, is an annotation task ambiguous?). In short, the decision of how to best annotate data is a complicated decision that is intimately related to properties of learning and vision algorithms, the statistical properties of data, the logistics of human-computer interaction, and the tasks we want to solve. We believe that figuring out which type of models/annotations work best is an underexplored problem in computer vision research relative to its importance. The reason is simple: annotating data is time-consuming, expensive, and non-intellectually stimulating. It is difficult to find the time and money to collect large datasets with one type of annotation, and it is nearly impossible to justify annotating a large dataset with several different possibly redundant annotation schemes. Consequently, in depth understanding and analysis of the relative utility of different possible annotation representations is still embarassingly inadequate.

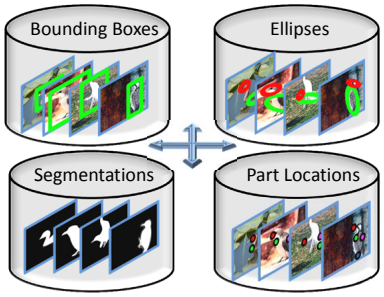


Figure 1: We introduce a framework for quickly augmenting a dataset with one type of pre-existing source annotation (*e.g.*, segmentations) with a new type of target annotation (*e.g.*, part locations) using a combination of computer vision, learning, and human feedback.

Toward moving beyond class labels, bounding boxes, and segmentations, representations based on nameable parts and attributes have recently become popular and resulted in emerging datasets with part [34, 16, 21] or attribute annotations [20, 34, 19, 23]. Part location annotations have been shown to (often significantly) improve detection or recognition performance in domains where people have collected them (*e.g.*, faces [2], human pose [5, 37], birds [3, 9, 14], and dogs [21]). Attribute annotations have been shown to be useful for applications such as transfer learning [20, 19].

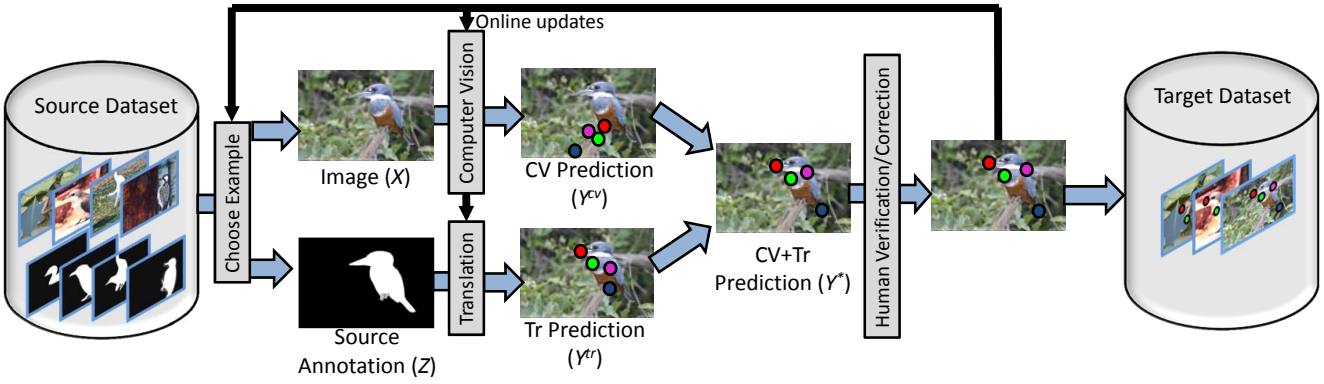


Figure 2: System Overview: A human-in-the-loop learning scheme for translating a dataset with pre-existing object segmentations to semantic part annotations. An annotator incrementally annotates images with part annotations, which is used to train online learning algorithms to predict parts using computer vision and using source segmentations. These are combined probabilistically to form an improved predictor that is used to accelerate data collection via interactive labeling and verification annotation tools.

While these types of annotations are useful, there exist many possible ways of decomposing objects into parts or attributes, with some ways being better than others. For example, if we decompose a human arm into a forearm and hand, should we also decompose a hand into fingers? Should we annotate parts in terms of point locations [5, 34], line segments [16], bounding boxes [12], or some other shape? Should we allow the annotator to rotate them in 2D or 3D? How should we handle occlusion or multiple viewpoints/aspects? In our experience, we have found that such answers tend to arise only after we begin to annotate data, as we observe the areas where the existing model doesn't fit the data well, where annotators struggle to provide consistent labels, or where computer vision algorithms perform poorly. In retrospect, we wish we had chosen a different annotation model; however because we don't want to start over and re-label the entire dataset, we are stuck with the decision we made in the beginning. One motivation for this paper is to provide a framework in which researchers can efficiently explore these types of incremental changes to the annotation model.

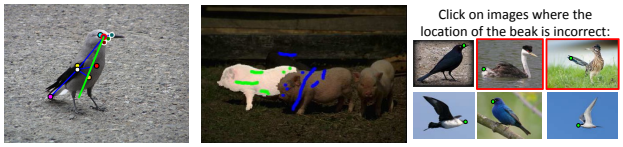
Related Work: Our system builds off of the method of [9], which combines online structured learning with *interactive labeling* [7, 1, 33, 17, 9], where a progressively improving structured predictor is used to accelerate annotation of new training examples. This is a different type of *active learning* than the more widely-studied problem of actively selecting which example to label from an unlabeled pool [31, 25, 32, 29, 22]. Our method is complementary to such methods–actively selecting examples and interactively labeling them are two separate ways to speedup annotation. The main difference between this paper and [9] is that 1) we incorporate source annotations as an additional information source to accelerate interactive labeling, and 2) whereas [9] focuses on annotation of deformable part models, we introduce a more general approach that is applicable to other alignment schemes, segmentations, and class labels. Fig. 3 shows a few examples of how a computer prediction can be used to accelerate annotation of different types of data.

Our method can be viewed as an extension of transfer learning and *domain adaptation* [4, 10, 24, 18] by adding humans-in-the-loop. In domain adaptation, labeled examples in a source domain are transferred to a different but related target domain. Besides incorporating human interactivity, our problem is different in that the observed images are identical for both the source and target annotation representations. In this way, our problem also has some similarities to multi-task learning [13].

Contributions: Our main contributions are: 1) We develop a general framework for learning a translation system from one type of annotation that can be combined with computer vision. We provide details to apply it to a wide variety of translation tasks including bounding boxes $\rightarrow$ parts, segmentation $\leftrightarrow$ parts, parts $\leftrightarrow$ parts, classes $\leftrightarrow$ attributes, with a wide-variety of different possible ways of representing parts including points, bounding boxes, rotatable ellipses, mixture models, and 3D objects. 2) We show empirically that our system is extremely effective at converting between different semantic part representations, yielding a 3-20 times speedup in collecting part annotations on the CUB-200-2011 and H3D datasets. We show that among non-domain specific annotation representations, segmentations are more effective than bounding boxes

2. Problem Definition and Notation

Suppose we have a pre-existing dataset of images $\{X_i\}_{i=1}^n$ and source annotations $\{Z_i\}_{i=1}^n$. Our goal is to augment the dataset with a new type of target annotation $\{Y_i\}_{i=1}^n$. For example, given a dataset with segmentations $\{Z_i\}_{i=1}^n$, one may wish to add additional annotations of se-



(a) Interactive Parts **(b)** Interactive Seg. **(c)** Verification

Figure 3: Smart GUIs for interactive labeling/verification where a computer vision based structured predictor is used to accelerate annotation: **(a)** From [9], an annotator interactively corrects a set of predicted part locations, **(b)** A computer vision predicted segmentation is interactively corrected using brush strokes, **(c)** Any structured annotation can be preceded by a simple binary verification of a prediction, which invokes a full manual annotation only if verification fails.

mantic part locations $\{Y_i\}_{i=1}^n$. Here each annotation Z and Y encapsulates some type of structured represention of an image, according to families $\mathcal{Z}$ and $\mathcal{Y}$, respectively.

An overview of our approach is summarized in Algorithm 1. As we incrementally label new target annotations Y, we progressively obtain more training data to learn an annotation translator $g : Z \to Y$ and an automated computer vision-based structured predictor $f : X \to Y$. Collectively, these are combined in a probabilistic model $p(Y|X, Z)$, where the goal is to compute the probability of a target annotation given an image and source annotation. This prediction probability is used to accelerate annotation using smart GUIs (Fig 3).

In Section 2.1, we begin by introducing notation to describe annotations such as bounding boxes, segmentations, semantic segmentations, class and attribute labels, deformable parts, 3D part models, and aspect mixture models. In Section 3, we describe our model of $p(Y|X, Z)$ and its efficient computation by augmenting existing computer vision algorithms.

2.1. Representing Different Annotation Schemes

In this section we describe notation for representing a variety of different possible annotation models.

1) Categorical Labels: Let Y be a binary vector of C possible class memberships that are used to tag an object or image. An element Y_c is 1 if the c-th category is present and 0 otherwise. Note that this representation can be used to encode both multiclass labels and non-mutually exclusive categories such as attribute memberships.

2) Semantic Segmentations: Let X be a $W \times H$ image and Y be an associated segmentation that assigns each pixel to one of C possibe discrete classes. Let Y_c be a binary segmentation mask for the c-th category.

3) Alignment/Pose Annotations: A general family of alignment annotations can be built using the following ingredients:

- **Warping function:** A spatial transformation function $v = P^{\mathcal{Y}_p \to I}(V, Y_p)$ that maps a point V in the reference system of the model to a point v in image coordinates. Let $V = P^{I \to \mathcal{Y}_p}(v, Y_p)$ be the inverse function, transforming from image to model coordinates. Let $W^{I \to \mathcal{Y}_p}(X, Y_p)$ denote the function that warps an image X by applying $P^{I \to \mathcal{Y}_p}(v, Y_p)$ to each pixel coordinate in X.
- **Annotation GUI:** A shape or a collection of vector graphics (*e.g.*, rectangles, ellipses, lines) G', expressed in model coordinates, that are use to display a visualization of an annotation $G = W^{\mathcal{Y}_p \to I}(G', Y_p)$. A set of controls that allows the annotator to explore all possible values of $Y_p \in \mathcal{Y}_p$ (*e.g.*, by dragging/rotating a rectangle or ellipse).
- **Image Features:** A pose-aligned feature extraction function $\psi_{\mathcal{Y}_p}(X'_p, Y_p)$ (*e.g.*, a HOG descriptor) operates on a normalized image $X'_p = W^{I \to \mathcal{Y}_p}(X, Y_p)$.
- **Part sets:** An annotation Y that is a collection of $P \geq 1$ parts $Y = \{Y_p\}_{p=1}^P$, resulting in concatenated feature spaces $\psi_{\mathcal{Y}}(Y) = [\psi_{\mathcal{Y}_1}(X'_1, Y_1), ..., \psi_{\mathcal{Y}_P}(X'_P, Y_P)]$ and $\Phi_{\mathcal{Y}}(Y) = [\Phi_{\mathcal{Y}_1}(Y_1), ..., \Phi_{\mathcal{Y}_P}(Y_P)]$

In the supplementary material, we give examples illustrating how annotation schemes based on bounding boxes, rotatable ellipses, keypoint antotations, 3D shapes, and mixture models can be expressed using this notation.

3. Annotation Translator

In this section, we define general and efficient procedures to estimate $\max_Y p(Y|X, Z)$, a probabilistic prediction of a target annotation Y that combines computer vision with additional information contained in a source annotation Z. We model $p(Y|X, Z)$ as

$$\log p(Y|X, Z; \mathbf{w}) \propto \Gamma(Y; X, \alpha) + \Omega(Y; Z, \beta) \quad (1)$$

where $\mathbf{w} = [\alpha, \beta]$, and $\Gamma(Y; X, \alpha)$ and $\Omega(Y; Z, \beta)$ are potential functions that are parameterized by learnable model parameters α and β. Here, we have assumed that Z is conditionally independent of X given Y. This is a reasonable assumption in the setting that the target annotation Y is a richer representation than the simpler source annotation Z (which is primarily the setting that we are interested in) and is an approximation otherwise.

The main benefit is that it allows for a decomposition into terms $\Gamma(Y; X, \alpha)$, which can be understood as a computer vision score, and $\Omega(Y; Z, \alpha)$, a source-to-target translation score. We give their exact forms in Sections 3.1 and 3.2.

Prediction: Given parameters, α and β, we define the following prediction problems:

$$\textbf{Computer Vision: } Y^{cv} = \arg\max_Y \Gamma(Y; X, \alpha) \quad (2)$$

$$\textbf{Translation: } Y^{tr} = \arg\max_Y \Omega(Y; Z, \beta) \quad (3)$$

$$\textbf{Combined: } Y^* = \arg\max_Y \Gamma(Y; X, \alpha) + \Omega(Y; Z, \beta) \quad (4)$$

Algorithm 1 Active Dataset Translation

Input: Dataset of images $\{X_i\}_{i=1}^n$ and pre-existing source annotations $\{Z_i\}_{i=1}^n$
Output: Human-Verified target annotations $\{Y_i\}_{i=1}^n$ according to a new annotation scheme
Initialize: Labeled/unlabeled pools $L \leftarrow \emptyset, U \leftarrow \{1, ..., n\}$, and weights $\mathbf{w_0} \leftarrow 0$

1: **for** $t = 1, 2, ...,$ until $U = \emptyset$ **do**
2: Define $Y_i^* := \arg\max_Y p(Y|X_i, Z_i; \mathbf{w_{t-1}})$, the most likely target label given an image and source label
3: Query human annotator(s) to verify/correct a batch $B \subseteq U$, outputing $\{Y_i\}_{i \in B}$ given $\{Y_i^*\}_{i \in B}$
4: Update labeled and unlabeled sets: $U \leftarrow U \setminus B, \quad L \leftarrow U \cup B$
5: Update learning model to minimize new training loss: $\mathbf{w}_t \leftarrow \arg\min_{\mathbf{w}} \sum_{i \in L} \ell(Y^*(X_i, Z_i, \mathbf{w}), Y_i; \mathbf{w})$
6: **end for**

where Y^{cv} is the most likely prediction given just computer vision, Y^{tr} is the most likely prediction given just source translation information, and Y^* is the most likely label using both sources of information.

Learning: Let $\Delta(Y, Y_i)$ be an application specific loss for predicting Y instead of groundtruth label Y_i (*e.g.*, the number of incorrect part locations for part detection) and $\ell(Y^*; \mathbf{w})$ be a convex upper bound on $\Delta(Y^*, Y_i)$ We consider two possible learning methods in our experimental results. The first learns parameters α and β jointly and discriminatively to minimize loss on a training set L of triplets $\{(X_i, Y_i, Z_i)\}_{i \in L}$:

$$\textbf{Joint: } \mathbf{w}^* \leftarrow \arg\min_{\alpha,\beta} \sum_{i \in L} \ell(Y^*(X_i, Z_i, \alpha, \beta), Y_i; \alpha, \beta) \quad (5)$$

where with some abuse of notation $Y^{cv}(X, \alpha)$, $Y^{tr}(Z, \beta)$, and $Y^*(X, Z, \alpha, \beta)$ denote the solutions to Eq 2-4 with the applicable parameters. The second method learns parameters α^* for computer vision and translation β^* independently and then combines them using piecewise scaling [27, 26] on a validation set $\{(X_i, Y_i, Z_i)\}_{i \in V}$:

$$\textbf{CV: } \alpha^* \leftarrow \arg\min_{\alpha} \sum_{i \in L} \ell(Y^{cv}(X_i, \alpha), Y_i; \alpha) \quad (6)$$

$$\textbf{Tr: } \beta^* \leftarrow \arg\min_{\beta} \sum_{i \in L} \ell(Y^{tr}(Z_i, \beta), Y_i; \beta) \quad (7)$$

$$\textbf{PW: } \gamma^* \leftarrow \arg\min_{\gamma} \sum_{i \in V} \Delta(Y^*(X_i, Z_i, \alpha^*, \gamma\beta^*), Y_i) \quad (8)$$

Here piecewise scaling learns a weighted combination of Γ and Ω via a scalar γ. In practice we found that both joint learning and piecewise scaling gave similar results, and thus joint learning is preferred (no validation set is required).

In all cases, we solve Eq 5-7 using a structured SVM [28, 30], where $\ell()$ is a variant of hinge loss. We use the solver from [8], which provides a network-based interface for adding new examples in online fashion that can be invoked by web-based GUIs. In the next section, we give examples of existing computer vision algorithms for solving Eq 2 and how they can be altered to solve Eq 3-4.

3.1. Augmenting Computer Vision Algorithms With a Translation Prior

In this section, we describe how many popular computer vision algorithms can be interpreted as predicting a label Y^{cv} that optimizes a score function $Y^{cv} = \arg\max_Y \Gamma(Y; X, \alpha)$ (exactly or approximately). For each such algorithm, we define how additional energy terms $\Omega_c(Y_c; Z)$ can be incorporated as prior terms at prediction time with a neglible increase in computational cost. The main requirement is that when $Y \in \mathcal{Y}$ is some multivariate representation and $\Gamma(Y; X, \alpha) = \sum_c \Gamma_c(Y_c; X, \alpha)$ is decomposed into energy terms over cliques of variables $Y_c \in \mathcal{Y}_c$, then prior terms $\Omega(Y; X, \alpha) = \sum_c \Omega_c(Y_c; Z)$ should respect the same clique structure. In Section 3.2, we will define terms $\Omega_c(Y_c; Z)$ for different possible applications, such that we can implement efficient solvers for Y^* (Eq 4).

3.1.1 Detection Algorithms

Pictorial structure methods (*e.g.*, constellation models [35] and deformable part models [15]) define prediction scores as a sum over per-part unary detection scores $\Gamma_p(Y_p; X, \alpha_p)$ and spatial scores $\Gamma_{pq}(Y_p, Y_q; X, \alpha_{pq})$ between pairs of parts:

$$\Gamma^{\text{dpm}}(Y; X, \alpha) = \sum_{p=1}^{P} \Gamma_p^{\text{dpm}}(Y_p; X, \alpha_p) + \sum_{(p,q) \in E} \Gamma_{pq}^{\text{dpm}}(Y_p, Y_q; \alpha_{pq}) \quad (9)$$

Our implementation of part-based methods is based on [9]; we represent each part by a mixture of bounding boxes of fixed aspect ratio $Y_p = \{c_x, c_y, w, a\}$, using sliding window scores for each aspect j

$$\Gamma_{p,j}^{\text{dpm}}(Y_p; X, \alpha_{pj}) = \alpha_{p,j} \cdot \psi_{\mathcal{Y}_p}(X'_p, Y_p) \quad (10)$$

that are the dot product of part-localized HOG features $\psi_{\mathcal{Y}_p}(X'_p, Y_p)$ and templates $\alpha_{p,j}$, with $\Gamma_p^{\text{dpm}}(Y_p; X, \alpha_p) = \Gamma_{p,Y_p(a)}^{\text{dpm}}(Y_p; X, \alpha_{p,Y_p(a)})$. Spatial scores are quadratic spring costs that are defined over a pre-defined tree of parts

$$\Gamma_{pq,j}^{\text{dpm}}(Y_p, Y_q) = \alpha_{pq,j} \cdot [1, \delta(Y_q, Y_p), \delta(Y_q, Y_p)^2] \quad (11)$$

with $\Gamma^{\text{dpm}}_{pq}(Y_p, Y_q) = \Gamma^{\text{dpm}}_{pq,Y_p(a)}(Y_p, Y_q)$. Here, $\delta(Y_q, Y_p)$ is the location of child q in the reference system of its part p:

$$\delta(Y_q, Y_p) = P^{\mathcal{Y}_q \to \mathcal{Y}_p}(Y_q, Y_p) \tag{12}$$

Fast algorithms for prediction (maximizing Eq 9 with respect to Y using dynamic programming) and learning α are available in publicly available software packages [8, 37]. In Section 3.2, we will define energy terms $\Omega^{\text{dpm}}_{p,j}(Y_p; Z)$ that induce a prior on the location of part p given source annotation Z. These can simply be added into the detection scores $\Gamma^{\text{dpm}}_{p,j}(Y_p; X, \alpha_{pj})$ at prediction time.

3.1.2 Segmentation Algorithms

Let $Y(i) \in 1...C$ indicate the class label of the i-th pixel of a semantic segmentation. A popular CRF-based segmentation model [6, 26] optimizes a score function that is a sum over unary and pairwise potentials:

$$\Gamma^{\text{seg}}(Y; X, \alpha) = \sum_{i \in \mathcal{V}} \Gamma^{\text{u}}_i(Y(i); X) + \sum_{j \in \mathcal{N}_i} \Gamma^{\text{p}}_{ij}(Y(i), Y(j); X) \tag{13}$$

where $\mathcal{N}_i$ defines a neighborhood of pixels (*e.g.*, the 4 neighboring pixels of i), $\Gamma^{\text{u}}_i(Y(i); X)$ is a unary potential at the i-th pixel (often a per-pixel classifier that applies per-class weights $\alpha_{Y(i)}$ to image features around a patch centered at i), and $\Gamma^{\text{p}}_{ij}(Y(i), Y(j); X)$ is a pairwise potential that favors labeling neighboring pixels with similar color as the same class (*e.g.*, $-1_{Y(i) \neq Y(j)} \exp\{-\|X(i) - X(j)\|^2\}$). The most likely segmentation Y^{cv} that maximizes Eq 13 is solvable using a graph cuts algorithm [6] that is optimal for binary segmentation and has approximation guarantees for $C > 2$. We can incorporate prior information from a source annotation Z via a unary potential $\Omega^u_i(Y(i); Z)$ (see Section 3.2) that can simply be added into $\Gamma^{\text{u}}_i(Y(i); X)$.

3.2. Translating Source to Target Annotations

In the previous section, we described how different computer vision algorithms can be augmented to incorporate prior terms $\Omega_i(Y_i; Z)$. In this section, we define the form of these prior terms and describe a general model for learning a source-to-target predictor $Y^{tr} = \arg\max_Y \Omega(Y; Z)$. We define the translation score as

$$\Omega(Y; Z) = \beta^T \Phi_{\mathcal{YZ}}(Y, Z) \tag{14}$$

for some arbitrary application-specific feature function $\Phi_{\mathcal{YZ}}(Y, Z)$ that models the joint likelihood of source-target pairs Y and Z. We give the form of $\Phi_{\mathcal{YZ}}(Y, Z)$ and provide details for implementing solvers for Y^{tr} and Y^* for different types of annotation schemes for Y and Z below:

Parts $\to$ Parts: Let $\mathcal{Y}$ and $\mathcal{Z}$ denote different families of part-based representations that are composed of $P_{\mathcal{Y}}$ and $P_{\mathcal{Z}}$ parts, respectively. For example, this could be useful for translating from a 2D annotation to a 3D annotation or exploring the space of different 2D part-based representations (*e.g.*, adding a new part or mixture model). Our choice of $\Phi_{\mathcal{YZ}}(Y, Z)$ is motivated by the belief that different part locations should be at predictable offsets from one another when normalized in the coordinate system of the closest part, with Gaussian-like noise when parts $\mathcal{Y}_p$ and $\mathcal{Z}_q$ the alignment schemes are inprecise:

$$\Omega(Y; Z) = \sum_{p,q,j \in A_p} \beta_{pqj} \cdot \Phi_{pqj}(Y_p; Z_q) \tag{15}$$

$$\Phi_{pqj}(Y_p; Z_q) = [1, \delta(Z_q, Y_p), \delta(Z_q, Y_p)^2] 1_{Y_p(a)=j} \tag{16}$$

where $\delta(Z_q, Y_p)$ is the location of part Z_q in the reference system of part Y_p (defined in Eq 12), and $Y_p(a)$ is the aspect label of part p. Here, each part q in $\mathcal{Z}$ votes for the location of each part p in $\mathcal{Y}$, with a learnable offset and variance (due to the quadratic cost) and a different offset/variance based on the aspect label of Y. To solve for Y^*, this results in a simple additive term that can be incorporated into existing part-based solvers (see Section 3.1.1).

$$\Omega^{\text{dpm}}_{pj}(Y_p; Z) = \sum_q \beta_{pqj} \cdot \Phi_{pqj}(Y_p; Z_q) \tag{17}$$

Bounding Box $\to$ Parts: This is a special case of a Parts $\to$ Parts translation problem, with $P_{\mathcal{Z}} = 1$. However, we additionally prevent part predictions outside the bounding box by incorporating an additional energy $-\infty$.

Semantic Segmentation $\to$ Parts: Let Y be a part-based annotation and Z be a semantic segmentation, where Z_c is a $W \times H$ binary segmentation map that is 1 if a pixel is labeled as class c and 0 otherwise. We choose translation feature vectors $\Phi_{\mathcal{YZ}}(Y, Z) = [\Phi_{cpj}(Y_p, Z_c)]_{c,p,j}$ that concatenate vectors $\Phi_{cpj}(Y_p, Z)$ for all possible combinations of class c, part p, and aspect j

$$\Phi_{cpj}(Y, Z) = 1_{Y_p(a)=j} W^{I \to \mathcal{Y}_p}(Z_c, Y_p) \tag{18}$$

This transformation warps segmentation Z_c from image coordinates into the model coordinates of part p. We can intuitively think of the corresponding model parameters β_{cpj} as a likelihood map expressing how likely each pixel location, represented in model coordinates of part p, belongs to class c. To compute $Y^* = \arg\max p(Y|X, Z)$, the above transformation induces a score map for part p and aspect j

$$\Omega^{\text{dpm}}_{pj}(Y_p; Z) = \sum_c \beta^T_{cpj} \Phi_{cpj}(Z_c, Y_p) \tag{19}$$

Parts $\to$ Semantic Segmentation: Let Y be a semantic segmentation and Z be a part-based annotation. We use the same basic model, with $\Phi_{\mathcal{YZ}}(Y, Z) = [\Phi_{cpj}(Y_c, Z_p)]_{c,p,j}$, and the same intuition of β_{cpj}, swapping Y and Z:

$$\Phi_{cpj}(Y_c, Z_p) = 1_{Z_p(a)=j} W^{I \to \mathcal{Z}_p}(Y_c, Z_p) \tag{20}$$

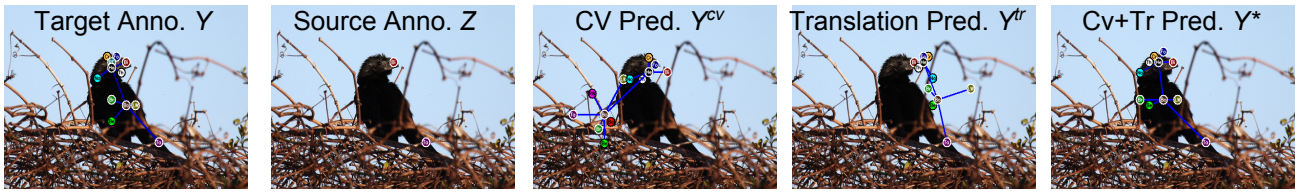


Figure 4: 2 Parts → 15 Parts: Source annotations of 2 parts (beak and tail) are used to speedup collection of 15 semantic parts. The last 3 columns show predictions when labeling the 31st training example (see Fig 5d) using just computer vision, just translation, and both. Computer vision alone (3rd column) confuses the bird with background clutter, while the translation predictor (4th column) cannot infer the bird's pose from just 2 points due to an abnormally turned head. The combined prediction (5th column) correctly predicts the location of most parts.

To compute $Y^* = \arg\max p(Y|X,Z)$, we warp likelihood maps to image coordinates and use that to induce a unary potential for a graph-cuts-based segmentation algorithm (see Section 3.1.2):

$$U_c = \sum_{p,j} W^{Z_p \to I}(\beta_{cpj}^T \Phi_{cpj}(Y_c, Z_p), Z_p) \quad (21)$$

where $\Omega_i^{\mathrm{u}}(Y(i); Z) = U_{Y(i)}(i)$ is the likelihood that the i-th pixel has label c given observed part locations Z.

Classes → Attributes: See supplementary material.

4. Experiments

Datasets: To demonstrate the effectiveness of our method, we perform experiments on two datasets: CUB-200-2011 [34]–a dataset of 11,788 images of birds that are annotated by bounding boxes, segmentations [14], and locations of 15 semantic parts (*e.g.*, beak, belly, crown, *etc.*)–, and H3D [5]–a dataset of 521 images of humans with locations of 20 semantic parts (*e.g.*, left ankle, right hip, nose, *etc.*) as well as segmentations of 19 different semantic regions (*e.g.*, sunglasses, hair, dress, *etc.*). We concentrate experiments on the application of part location prediction (where the target annotation is the 15 part locations for CUB-200-2011 or 20 part locations for H3D). This concentration allows us to establish a common benchmark to analyze the relative utility of different types of source annotations toward reducing annotation time, including bounding boxes, segmentations, and different types of part representations.

Measuring Performance: To avoid the expense of running human experiments each time we change source annotations or parameter settings, we simulate human interaction from ground truth labels using the same method as [9]: 1) The computer displays its prediction of the most likely part locations Y^* 2) The simulated user selects and drags the part with maximum distance to his/her click response (normalized by a per part standard deviation) 3) If all part predictions are within 1.5 standard deviations from groundtruth (measured using multiple MTurk responses), the session ends. Otherwise, steps 1-2 are repeated.

We record the total elapsed time to annotate an image as the sum of MTurk response times (available with the CUB-200-2011 dataset) for each dragged part. Although this simulated interface may not exactly match human users, it provides a means to directly compare performance to [9]–which is equivalent to our method using just computer vision and no source annotations (Y^{cv} in place of Y^*).

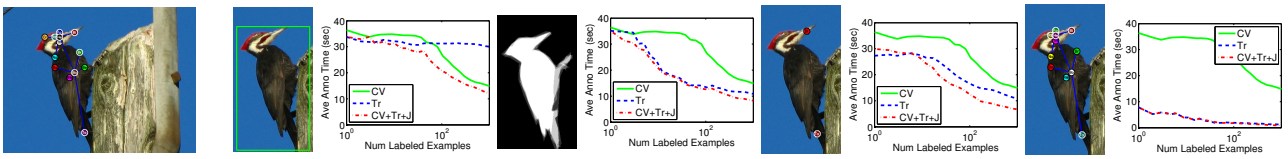
Baselines and Variants of Our Algorithms: For each type of source annotation, we measure performance of 4 different variations of our algorithm: 1) Using just computer vision and no source annotation [9] (CV curves in each table/plot), where prediction and learning occur using Eq 2 and Eq 6, 2) Using just source annotations and no computer vision (Tr curves in each table/plot), Eq 3 and Eq 7, 3) Combining computer vision and source annotations while jointly learning their parameters (CV+Tr+J curves in each table/plot), Eq 4 and Eq 5, 4) Combining computer vision and source annotations using piecewise scaling (CV+Tr curves in each table/plot), Eq 4 and Eq 8.

Implementation Details: For both datasets, we use part detection based on deformable part models from the implementation of [8], with 7×7 HOG templates for each part, 15 mixture components (including visibility) per part, and 100 mixture components for the root part. For CUB-200-2011, we restricted attention to a random 1000 image subset, because we observed performance was already roughly saturated. For the H3D dataset, since we don't have statistics for MTurk users, we assumed each part correction action took 2.6 seconds (the median response time for CUB-200-2011) and a standard deviation of 16 pixels normalized by the object scale. To avoid ambiguity of which person to label for images with multiple people, we restricted attention to cropped bounding boxes for the H3D dataset, and focused on a random 500-person subset.

4.1. Parts → Parts

We consider a couple of different scenarios where one may wish to convert one part representation to another.

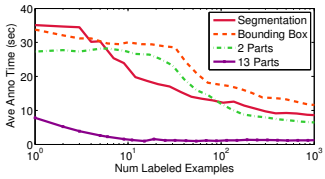
Simple-to-Detailed Representation: For both datasets, we choose source annotations consisting of two part locations (beak and tail for birds, right shoulder and left hip for H3D), and use those to help infer target annotations consisting of all parts. Fig 4 and Fig 5d,5f show qualitative and quantitative results on the CUB-200-2011 dataset. We see that



(a) Target (Y) **(b)** Bounding Box (Z) **(c)** Segmentation (Z) **(d)** 2 Parts (beak+tail) (Z) **(e)** 13 Part (no belly/crown)

Source Anno.	**Manual**	**CV** [9]	**Tr**	**CV+Tr**	**CV+Tr+J**
Bounding Box	10.8hr	4.13hr	8.34hr	**3.18hr**	3.32hr
Segmentation	10.8hr	4.13hr	3.0hr	2.40hr	**2.29hr**
2 parts	9.39hr	3.51hr	3.0hr	1.80hr	**1.62hr**
13 parts	1.44hr	0.62hr	0.38hr	0.31hr	**0.23hr**

(f) Time to annotate 1000 image bird dataset with 15 parts



(g) Annotation time for CV+Tr+J methods

Figure 5: CUB-200-2011 part annotation results: **(a)** The goal was to annotate a 1000 image bird datset with 15 semantic parts. Different types of source annotation were tested, including: **(b)** bounding boxes, **(c)** segmentations, **(d)** 2 part locations (beak and tail), **(e)** 13 part locations (everything except the belly and crown). For each method, we incrementally trained a computer vision predictor (CV curves), source→target translation predictor (Tr curves), and combined predictor (CV+Tr+J curves), which were used to speedup annotation via interactive labeling. **(f)** Summary of the amount of human time needed to label the dataset. Each row shows a different type of source annotation and each column shows different crippled versions of our algorithms. We see that 1) all sources speedup annotation by at least a factor of 3, 2) segmentations and part source annotations further speedup annotation by an additional factor of at least 1.7, whereas bounding box annotations were less useful, 3) combining computer vision and source annotations yielded significant gains for all methods. **(g)** Combined plot of CV+Tr+J methods for each type of source annotation.

computer vision reduces the time to annotate the 1000 image bird dataset from 9.39 hours (traditional manual labeling) to 4.13 hours (this corresponds to the method of [9]), while incorporating these 2 part source annotations yields a significant further reduction to 1.62 hours. Fig 6c shows results on the H3D dataset, where total annotation time is reduced from 6.86 hours (traditional manual labeling) to 2.44 hours (CV+Tr).

Part Augmentation: This simulates the scenario in which one interactively defines a part model; after collecting initial annotations according to one part-based representation, one identifies some deficiency and wishes to add a new part. We perform this experiment on the CUB-200-2011 dataset, where we begin with source annotations of 13 parts and then add 2 new parts (belly and crown). We see that collecting new annotations for the 1000 image dataset takes only 0.23 hours, suggesting that interactively augmenting an existing part model is practical.

4.2. Bounding Box → Parts

We begin with source annotations of bird object-level bounding boxes and use that to accelerate collection of 15 part locations. Results are shown in Fig 5b,5f. Bounding boxes yield an expectedly modest reduction in annotation time, from 4.13 hours using just computer vision to 3.18 hours using computer vision and bounding boxes. The amount of gain is small because bounding boxes provide little information about the pose of the bird; they are mostly only useful for removing predictions in background clutter.

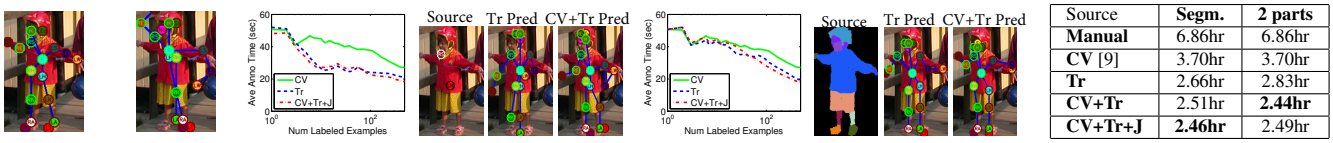
4.3. Segmentation → Parts

We begin with source annotations of object-level bird segmentations and use those to predict 15 part locations. We see that segmentations reduce annotation time from 4.13 hours using just computer vision to 2.29 hours using computer vision and segmentations. Additionally, using just segmentations and no computer vision requires 3.0 hours (outperforming computer vision by itself). At the same time, segmentations and computer vision are complementary–see the supplementary material for qualitative examples. The segmentation-based predictor tends to be better at localizing the rough pose of the bird, while the computer vision predictor is better at more finely localizing parts (especially parts internal to the bird contour).

We also run a Segmentation → 20 Parts experiment on the H3D dataset, where we begin with 19-class semantic segmentations (see Fig 6d). Note that although these semantic segmentations provide part-level region information, they correspond to an entirely different set of parts (often articles of clothing or accessories). Segmentations help reduce annotation time from 3.7 hours (just computer vision) to 2.46 hours.

4.4. Relative Utility of Different Source Annotations

Fig 5f analyzes the relative utility of different types of source annotations (bounding boxes, segmentation, parts). We see that part annotations are most effective; however, they require domain-specific knowledge (one must define a part model for each application). Among non-domain specific representations (bounding boxes and segmentations), segmentations are most useful. This suggests a possible scenario where one first obtains annotations according to some universal representation like segmentations (due to not knowing what representation to use), and later translates these to some domain specific representation (which maybe more applicable to a specific application, and because part-level annotations tend to be more effective as direct inputs



Source	Segm.	2 parts
Manual	6.86hr	6.86hr
CV [9]	3.70hr	3.70hr
Tr	2.66hr	2.83hr
CV+Tr	2.51hr	**2.44hr**
CV+Tr+J	**2.46hr**	2.49hr

(a) Target **(b)** CV pred. **(c)** Source: Right Shoulder + Left Hip **(d)** Source: Segmentation **(e)** Time to annotate H3D

Figure 6: H3D Part annotation results: (a) The goal was to label a 500 image dataset with 20 part locations. **(b)** Computer vision prediction Y^{cv} for a qualitative example. **(c)** Results when using labeled right shoulder and left hip locations as a source annotation. **(d)** Results when using semantic segmentations as a source annotation. **(e)** Total annotation time on the H3D dataset. The 5 rows show the average annotation time when using i) traditional non-accelerated annotation, ii) just computer vision, iii) just source annotations, and iv-v) two different methods for combining computer vision and source annotations.

to detection/categorization learning algorithms).

5. Conclusion

We introduced a framework for quickly augmenting a dataset with one type of source annotation to a new type of target annotation. We introduced a human-in-the-loop method to translate between a wide variety of annotation types, including bounding boxes, segmentations, 2D and 3D part-based systems, and class and attribute labels. Our results show that 1) our method can be used to accelerate collection of part annotations by a factor of 3-20 on the CUB-200-2011 and H3D datasets, and 2) segmentations are a more useful for translating to different representations than bounding boxes, 3) simple part-based representations can be effectively translated into more complicated representations. In future work, we hope to run experiments applying our method to translate from parts to segmentations and class labels to attribute labels. Additionally, we hope to use our method in conjunction with an interactive algorithm for evolving definitions of part and attribute vocabularies.

Acknowledgments

Funding for this work was provided by the Google Focused Research Award.

References

[1] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *CVPR*, 2010.

[2] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers for face verification. In *BMVC*, 2012.

[3] T. Berg and P. N. Belhumeur. Poof: Part-based 1-vs-1 features for fine-grained categorization. *CVPR*, 2013.

[4] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *EMNLP*, 2006.

[5] L. Bourdev and J. Malik. Poselets: Body part detectors using 3d human pose annotations. In *ICCV*, 2010.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[7] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *CVPR*, 2001.

[8] S. Branson, O. Beijbom, and S. Belongie. Efficient large-scale structured learning. In *CVPR*, 2013.

[9] S. Branson, P. Perona, and S. Belongie. Strong supervision from weak annotation: Interactive labeling of deformable part models. In *ICCV*, 2011.

[10] H. Daumé III and D. Marcu. Domain adaptation for statistical classifiers. *JAIR*, 2006.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale database. In *CVPR*, 2009.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal voc challenge. *IJCV*, 2010.

[13] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *SIGKDD*, 2004.

[14] R. Farrell, O. Oza, N. Zhang, V. I. Morariu, T. Darrell, and L. S. Davis. Birdlets: Subordinate categorization using volumetric primitives. In *ICCV*, 2011.

[15] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale deformable part model. In *CVPR*, 2008.

[16] V. Ferrari, M. Marin-Jimenez, and A. Zisserman. Progressive search space reduction for human pose estimation. In *CVPR*, 2008.

[17] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature and Methods*, 2013.

[18] B. Kulis, K. Saenko, and T. Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR*, 2011.

[19] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and Simile Classifiers for Face Verification. In *ICCV*, 2009.

[20] C. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes. In *CVPR*, 2009.

[21] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur. Dog breed classification using part localization. *ECCV*, 2012.

[22] C. N. D. Mac Aodha, Oisin, J. Kautz, and G. J. Brostow. Hierarchical Subquery Evaluation for Active Learning on a Graph. In *CVPR*, 2014.

[23] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *CVPR*, 2012.

[24] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*. 2010.

[25] B. Settles. *Curious machines: active learning with structured instances*. PhD Thesis, 2008.

[26] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. In *ECCV*. 2006.

[27] C. Sutton and A. McCallum. Piecewise pseudolikelihood for efficient training of conditional random fields. In *ICML*, 2007.

[28] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, volume 16, page 25. MIT Press, 2004.

[29] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*.

[30] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and output variables. *JMLR*, 2006.

[31] S. Vijayanarasimhan and K. Grauman. What's It Going to Cost You? Predicting Effort vs. Informativeness for Multi-Label Image Annotations. In *CVPR*, 2009.

[32] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning. In *CVPR*, 2011.

[33] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2013.

[34] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, Caltech, 2011.

[35] M. Weber, M. Welling, and P. Perona. *Unsupervised learning of models for recognition*. PhD Thesis, 2000.

[36] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[37] Y. Yang and D. Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. In *CVPR*, 2011.