CS 280 Computer Vision
Spring 2018 Kanazawa Lecture 25

1 Lecture

Humans are everywhere in visual data, and we want to make sense of their appearances. Applications include *motion analysis*, *behavior analysis*, and *VR/AR virtual presence*. However, perceiving humans is hard because they can vary a lot in appearance and pose, and there's often a lot of occlusion, clutter, and weird illumination in the images.

History: 2D

- In the beginning, it was a geometric era. We wanted to model humans geometrically in 3D (e.g. Marr with his generalized cylinders in '78). In 1980, O'Rourke and Badler published a paper which used skeletons to represent humans (close to what we do now).

  Nevatia and Binford. "Structured Descriptions..." 1973.
  Marr and Nishihara. "Representation and Recognition..." 1978.
  O'Rourke and Badler. "Model-Based Image Analysis..." 1980.

- Unfortunately, geometric modeling is not practical and doesn't work in real images. More recently, we've instead looked at appearance in 2D images.

- The first step in perceiving humans is to see where they are (detection). The '05 HoG (histogram of oriented gradients) paper by Dalal/Triggs addressed this task. As a refresher, it involved HoG features (i.e. frequencies of oriented edges) and high SVM response on locations with positive [human] class.

  Dalal and Triggs. "Histograms of Oriented Gradients..." 2005.

- Another work: deformable part models. Humans move around a lot, so a big fixed template can be an issue. Thus the idea is to go from bounding box to parts. Instead of having one big template for a human, we would have multiple different templates (capturing "parts" and their appearances).

  Felzenszwalb et al. "A Discriminatively Trained..." 2008.

- Although parts were initially blobby segmentation regions or bounding boxes, they would later develop into finer joints and keypoints (i.e. parts defined by single points). The representation was something like semantic points on a skeleton form – a more stable representation due to ease of labeling.

  Ferrari et al. "Progressive Search Space Reduction..." 2008.
  Yang and Ramanan. "Articulated Human Pose Estimation..." 2011.

- Recently, more datasets have arrived in this domain. Leeds Sports Pose (LSP), Frames Labeled in Cinema (FLIC), COCO... It's not as large as ImageNet, but it's starting to get big. And as a result of all the data, we can train deep networks on this stuff.

  Johnson and Everingham. "Clustered Pose..." 2010.
  Sapp and Taskar. "MODEC: Multimodal Decomposable Models..." 2013.

- DeepPose was one of the first such papers attempting to predict human keypoints on images. It takes an image, runs it through AlexNet, and regresses the (x, y) locations of fourteen keypoints. Because it's always outputting the locations of all of the joints, it provides a holistic view: even if a person's left arm is not visible, the network has to say something about it.

  Toshev and Szegedy. "DeepPose: Human Pose Estimation..." 2014.

- Nevertheless, it tends to be hard to regress such keypoints in a continuous space. Another approach, which has become more mainstream, is to predict the heatmap of each part. This is similar to semantic segmentation: now the classes are the keypoints and our convolutional network says, for each pixel, "is it likely that there's a wrist at this location?"
  - The end volume has a number of channels equal to the number of keypoints.
  - The target is a Gaussian around (x, y), the correct location of each keypoint.
- However, local evidence alone isn't that strong. The receptive field of the final layer is very limited. And locally a lot of things look similar (a blurry left leg versus a blurry right leg, for example). In such cases we need more context, or a prior for structure and spatial dependencies.
- One way to do this is to use a graphical model. But a more recent approach is to have context in the network, to use a *convolutional pose machine*.
  - In the first stage, we detect a (potentially noisy) heatmap.
  - The heatmap is then sent to another network, where it is concatenated with an intermediate image feature representation, and used to update another belief map.
  - We then send the latest belief map to *another* network, which does the same thing.
  - This continues until we get to our final belief map at stage T.
  - As we progress through these stages, the effective receptive field size gets larger and we also see the detection from the past. Thus, the network learns what it should look at in order to learn the right spatial prior for part dependencies used in detection.
  - (The localization is refined over stages.)

  Wei et al. "Convolutional Pose Machines." 2016.

  * *Note: all of these methods are single-image. CPMs are close to real-time as well.*
- OpenPose: open-source tracking software that's so good, it's sometimes used as a pseudo-ground truth.

History: 3D

- One of the first 3D papers was written by Bregler and Malik. They followed a tracking-based approach as follows:
  1. Initialize 3D model (e.g. rectangles connected by rotation angles) in first frame.
  2. Do tracking over subsequent frames using Lucas-Kanade (local gradient response). It's like optical flow, where we need to solve for the displacement, except now the displacement is parameterized by joint angles (3D rotations of the parts).
  3. (This is more stable with multiple views.)

  Bregler and Malik. "Tracking People..." 1998.
- This is more of a geometry-based approach. Another, more recent approach by Taylor is based on *reconstruction*: if we know the 2D joint locations, can we lift them to 3D? [The 2D points are (x, y), and we just need to add the z component.] Here, we run into the same issue as single-view 3D reconstruction; there are multiple explanations of the human body for one 2D observation.

  Taylor. "Reconstruction of Articulated Objects..." 2000.
- So we need priors. In the paper, the prior was "known ratio of limb length."
- A more recent version of the reconstruction approach imposes more priors from the training data. We have a large collection of 3D skeletons from motion capture, and we're going to solve for

$$\mathbf{X} = \mu + \sum_{i=1}^{K} \mathbf{b}_i \omega_i$$

where $\mathbf{X}$ is the 3D joint locations of the ~14 points, which we represent as the mean skeleton μ plus a linear combination of skeletons $\mathbf{b}_i$ seen in the past.

  - The $\mathbf{b}_i$'s are like a basis of human poses.
  - This formulation restricts the solution space. Instead of solving for all (x, y, z) of the ~14 points, we solve for these coefficients ω_i to represent $\mathbf{X}$.
  - In other words, we solve for (a) $\mathbf{X}$ parameterized by these coefficients and (b) the relative camera pose, s.t. $\mathbf{X}$ projects onto the 2D joints and limb lengths are plausible.
  - Takeaway: we look at many 3D skeletons and learn some kind of prior over this, and use the prior when we are inferring what the 3D pose could be.

Ramakrishna. "Reconstructing 3D Human Pose..." 2012.

- More datasets have been introduced – motion capture datasets, so that we have the 3D coordinates of the humans. One such dataset is Human3.6M, which is quite large.

Ionescu et al. "Human3.6M..." 2014.

- I'm sure you know what's coming: with the advent of large datasets, we have the advent of deep learning approaches. Most of the progress in this area has been made in the last two years.
- We can again detect 3D keypoint locations with a fully convolutional network (similarly to the 2D case), in the form of voxels this time. This is the standard supervised approach, and is totally powered by these large datasets.
- Another approach that works surprisingly well is to take the 2D keypoint input (which we can obtain from ground truth or OpenPose) and use a very simple fully connected network to just regress the z component.
  - Since the network is simple, it trains very quickly.
  - Note that this approach doesn't look at the image.

Martinez et al. "A Simple Yet Effective Baseline for 3D Human Pose Estimation." 2017.

- Now, how about the 3D representations? We're not just stick figures; we have surface, shape, etc. Stick figures don't explicitly represent a lot of information, like which direction the limbs are going. There's more to humans than 3D skeletons.
- To do more than joints, we need to talk about how human bodies can be modeled in detail. Practically, we often want to model what we can see, i.e. the surface of the human as a mesh.
  - A mesh can be thought of as a graph $\{V, F\}$ of vertices V and faces F (which are triangles). It's like a 3D point cloud with some fixed connectivity.
  - A mesh could be parameterized (modeled) in *full space* [directly outputting vertex locations, with size (# vertices) $\cdot$ 3]. But this doesn't make a lot of sense; it's not how humans deform.
  - We are actually piecewise linear objects to some degree (this is how we articulate). And we want a low-dimensional parameterization. The key idea in modeling 3D human surfaces is to factorize each human into *shape* and *pose*.
  - There are changes in intrinsic shape, and changes due to articulation (i.e. pose). We can think of these things as orthogonal components.
  - We can learn *shape* from a certain dataset which contains 4000 3D scans of bodies in the same T-pose. (The scan results are 3D point clouds.) We can then represent shape as mean plus linear combination of basis shapes (PCA should be run to make it low-dimensional; we only need about 10 PCA components). This is analogous to eigenfaces.
  - And then there is pose (articulation). Say we have a template skeleton in some shape. Previous approaches output the (x, y, z) coordinates of skeleton points, but now we'll output the relative 3D rotation of each joint w.r.t. its parent. We can do forward kinematics to obtain the full pose.

  - (This pose representation is nice because it lends itself to animation in a straightforward way.)

Loper et al. "SMPL..." 2015.

- How to obtain 3D shape and pose from a single image? In SMPLify, the idea is that we start from an image and compute the 2D keypoints using a CNN. Then we fit the SMPL pose and shape parameters to explain the detected joints.
  - The SMPLify objective function, to be minimized, is

$$\underbrace{E_J(\beta, \theta, K; J_{est})}_{\text{data term}} + \underbrace{E_a(\theta) + E_\theta(\theta) + E_{sp}(\theta, \beta) + E_\beta(\beta)}_{\text{priors}}$$

    where K is the camera parameters, J_{est} is the estimated joints, β is the 10-dimensional coefficients representing the shape, and θ is the rotation matrices representing articulation of the pose.
  - (β, θ) specifies a mesh with a skeleton, i.e. we can get the 3D keypoints from it.
  - We should solve for the β and the θ that minimize the L2 distance between the estimated 2D image points and the projected (β, θ)-based keypoints. This is the data term, which explains the evidence.
  - We also impose the priors so that the skeleton isn't allowed to transform into weird shapes.

Bogo et al. "Keep it SMPL..." 2016.

- Note: these approaches are fitting-based. At test time, we have some evidence in the image that we want to explain, and we have to solve an optimization problem to do so (meaning not real-time). Also, they're not looking at the image, and we'd like to try to get more juice from that. *Can we do this in an end-to-end manner?*
- Of course. Again we have the (β, θ) parameterization, and we also add six [scaled orthographic] camera parameters (scale, tx, ty, axis-angle rotation). In total this is 82 parameters to represent the human.
  - Issue: not many datasets with **real** paired 2D-to-3D labels, nor with image-to-SMPL parameters.
  - How do we get 3D annotations from real images, even in the skeleton form? This has to be done in a weakly supervised setting, because we don't have many options for supervision.
  - We also have the same *depth ambiguity* problem as before.
  - Idea: even though we don't have 2D-to-3D labels for real images, we have a lot of unpaired labels (lots of 2D images with 2D labels, and lots of 3D scans whose 2D images are restricted or nonexistent). So we know what 3D humans are like; we know the distribution of the 3D humans. Let's explain the 2D using the model, keeping it within the distribution of real human scans.
  - The network thus regresses the 82 parameters (pose, shape, and camera) which create a human. From the human we can get the joints and again minimize the reprojection error. But this alone doesn't guarantee we get a valid human, so we send it to an adversary which says whether it's a real human or not (i.e. can it discriminate it from the distribution of 3D scans?). This captures the prior of human bodies.
  - We learn separate discriminators for each joint angle, a discriminator for the whole thing, and a discriminator for the shape. This factorization is nice because we don't need to see every combination of shape and pose to know whether a pose is right. (It's data-efficient.)

Kanazawa et al. "End-to-End Recovery of Human Shape and Pose." 2018.

Action Recognition

- Action recognition is challenging for many reasons: (a) it's unclear how to separate actions into classes, (b) depending on the range of time we're looking at, motion can acquire a different meaning, (c) interactions with objects or people can influence the meaning of an action too, and (d) there are perhaps complex goals and intentions we want to discover.

- Often the problem formulation involves classification and detection. "Is there a person diving in the video? and where?"
- Typical pipelines use appearance and motion cues (e.g. optical flow) as input.
- Much future work remains to be done: we need to make better use of temporal information in videos, and even ask ourselves if action recognition is the right task to be solved. A very open problem!