EECS 542 Programming Assignment 1

Pose Estimation using Efficient Matching of Pictorial Structures

Abhishek Thakur
thakabhi@umich.edu

Ankit Bansal
arbansal@umich.edu

Vivek Reddy Alla
vareddy@umich.edu

Xiao Huang
krishx@umich.edu

University of Michigan

Table of Contents

List of Figures

I. OBJECTIVE

The objective of this report is to implement the methodology proposed in [1]. This report addresses the base problem of detecting the torso, head, upper arm and the lower arm in the image by using the original match energy cost function defined beforehand. For the first bonus problem, we have modified the model to have a tree structure with a depth of 2 to account for the lower arms. We could achieve accurate results for most of the test images. For the second bonus problem, we have implemented our own match energy cost function which uses feature extraction and template matching to determine the location of the desired part in the image and returns the cost.

II. METHODOLOGY

The methodology used in our implementation takes motivation from [1]. A pictorial structure is represented as a collection of parts with connection between certain pairs. Such a model can be expressed as a tree with the nodes representing parts of the model and edges representing the pairwise connection between different parts of the model. For the pose detection problem, we use a person object model consisting of head, torso and the upper arms for the base problem. Torso is taken as the root node and the other parts form the leaf nodes of the tree. The model shown in Figure 1. has a depth of 2.

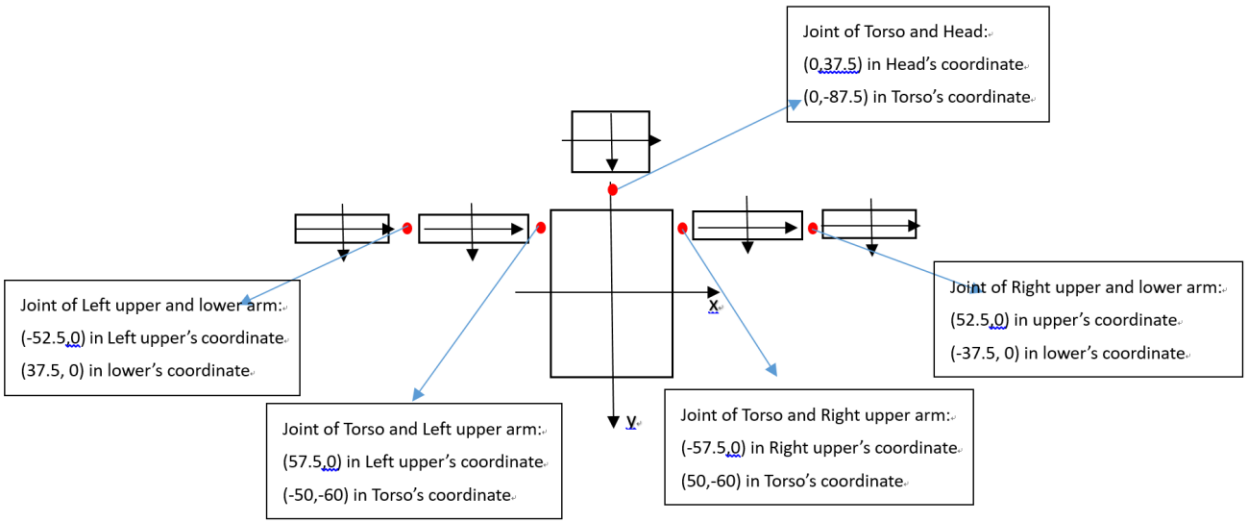


Fig 1. Person Object Model

An instance of a part in an image is specified by a location l which specifies the position, rotation and scale parameters. The match cost function $m_i(I, l)$ measures how well the part matches the image I when placed at location l. The connections between parts indicate relationships between their locations. For each connection (v_i, v_j) there is a deformation cost function $d_{ij} = (l_i, l_j)$ measuring how well the locations l_i of v_i and l_j of v_j agree with the object model.

A configuration $L = (l_1, \ldots, l_n)$ specifies the location for each of the parts, $v_i \, \epsilon \, V$ with respect to the image. The problem of estimating the pose can be formulated as finding the optimal location L^*, that minimizes the deformation and match cost as,

$$L^* = \arg\min_L \left(\sum_{(v_i, v_j) \epsilon E} d_{ij}(l_i, l_j) + \sum_{v_i \epsilon V} m_i(I, l_i) \right)$$

We have chosen Torso as the 'root node' of the tree-structured model. To find the optimal location of the Torso, we first discretize the space of all possible part locations into 50 buckets for each x and y positions, 10 buckets for size, and 20 buckets for orientation. Using the `match_energy_cost` function we choose five best possible locations having the least cost. The quality of the best location of a vertex v_j given the location l_i of v_i is,

$$B_j(l_i) = \min_{l_j}(d_{ij}(\, l_i, l_j) + m_j(I, l_j))$$

Given the optimal locations of the root node, we then find the optimal locations of the leaf nodes that minimize $B_j(l_i)$.

In bonus problem 1, we first find the optimal location of the torso followed by the optimal locations of the head, upper left and right arms. Since the lower arms are connected to the upper arms, we find their optimal locations based on the optimal locations of the upper arms.

Given the observed locations $l_i = (x_i, y_i, \theta_i, s_i)$ and $l_j = (x_j, y_j, \theta_j, s_j)$ of two parts, the deformation cost measures the deviation between the ideal values and the observed values. The `deformation_cost` function takes l_i and l_j as the input parameters and returns the deformation cost between the two locations. We define the pairwise deformation cost for the person model to be,

$$d_{ij}(l_i, l_j) = w_\theta * |\theta_i - \theta_j| + w_s * |s_i - s_j| + w_x * |x_i - x_j| + w_y * |y_i - y_j|$$

Where the first term is the difference between the ideal relative angle and the observed relative angle, the second term is the difference between the ideal relative size and the observed relative size. The ideal relative angle, θ_i and the ideal relative size s_{ij} are set to 0 and 1 respectively. The third and fourth terms are the horizontal and vertical distances between the observed joint positions.

The weights denote the contribution of the corresponding terms in the deformation cost. We set the value of w_θ to be 0.1, to emphasize the fact that parts can rotate freely about the joint. The values of the remaining weights are set to 5. Next, we convert the coordinates of joint points from the parts' coordinate system to the image coordinate system as,

$$(x_i', y_i')^T = W_{ij}((x_i, y_i)^T + s_i R_{\theta i}(x_{ij}, y_{ij})^T)$$

Where W_{ij} is a diagonal weight matrix with entries w_{ij}^{x} and w_{ij}^{y} , and $R_{\theta i}$ is a matrix that performs rotation of θ radians about the origin.

For the bonus problem 2, we implement our own `match_cost_energy` function using feature detection and template matching. We use the templates of head, torso, upper and lower arms to find the corresponding match in the image. To do this, we used a SURF feature detector to extract features from the template and the image. These features are then matched using the Matlab `matchfeatures` function. Once the template has been found in the image we use this location as our ground truth. The match energy cost function takes the location l, part number, and the sequence number as the input and calculates the $L-2$ norm of the location l from the ground truth location found using feature detection.

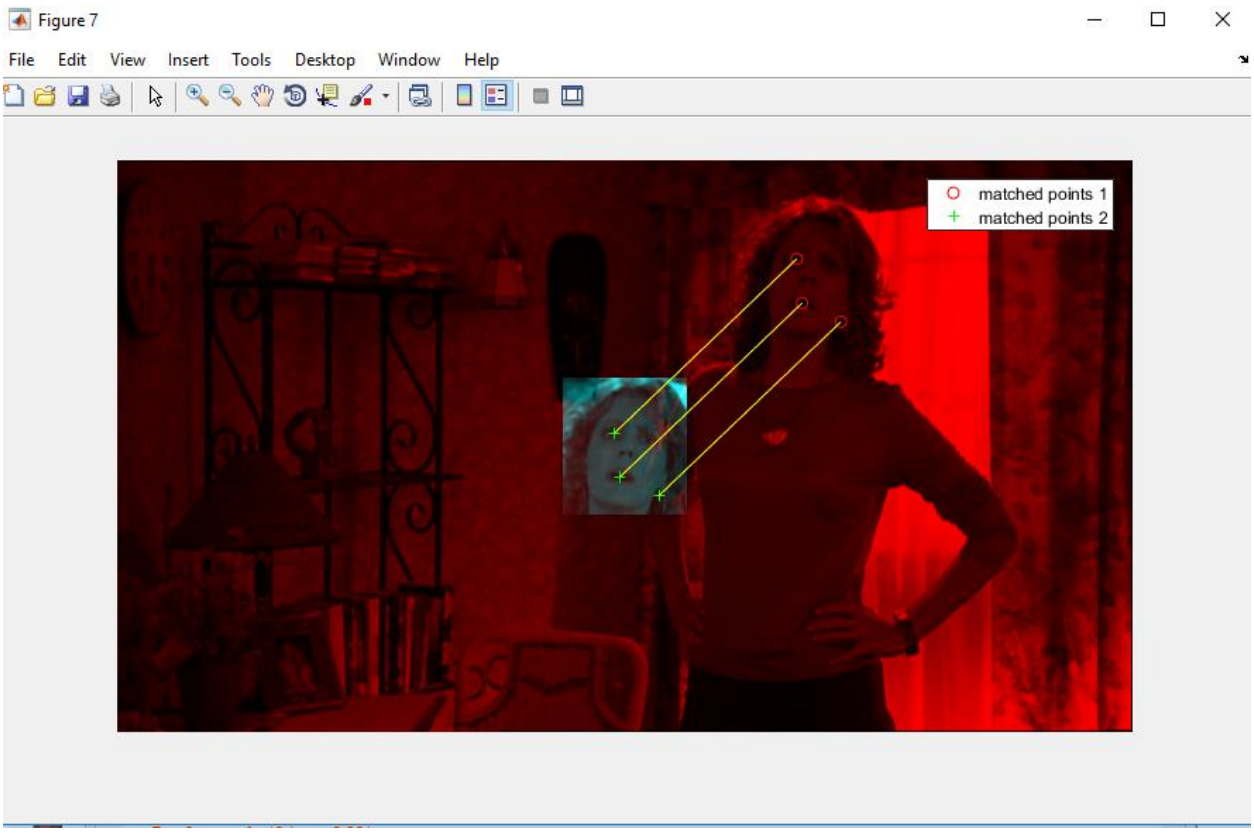


Fig 2. Template Matching

III. EXPERIMENTAL RESULTS

We were successfully able to estimate the pose using the 4-part (head, torso, and upper arms) Person Model for the base problem and 6-part Person Model for the bonus problem 1. For the second bonus problem, our implementation of the feature detection based `match_energy_cost` function returned the match cost given the part number and a query location l. On an average, the code takes about 150 seconds to compute.

(a)

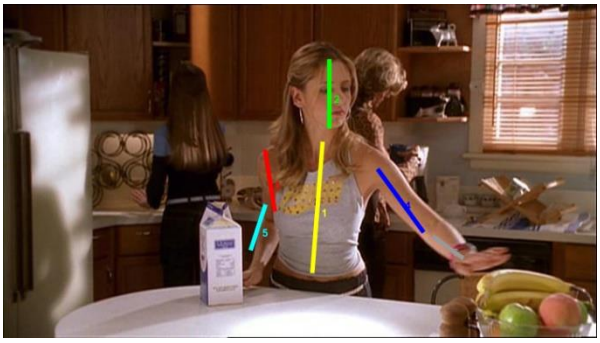
(b)

(c)

(d)

(e)

(f)



(g)

(h)

(i) (j)

Fig. 3 Experimental Results

IV. REFERENCES

[1] Felzenszwalb, P.F. & Huttenlocher, D.P. International Journal of Computer Vision (2005) 61: 55. doi:10.1023/B:VISI.0000042934.15159.49