A Machine Learning System for Identifying Animate Motion

Marek P. Michalowski
Yale University Department of Computer Science
New Haven, CT
marek.michalowski@yale.edu

Abstract

The ability to distinguish between the motion of inanimate objects being acted on by physical forces and the motion of animate beings is present in humans from an early age. A stimulus as simple as a moving dot is generally sufficient to perform this classification. One method of designing a machine to perform this task is to endow it with encoded knowledge of physical laws and the ability to recognize when a motion trajectory deviates from what would be expected from an inanimate object. However, this model is developmentally implausible and rigid in its design. An alternative method is to allow a machine to learn the animate/inanimate classification task from a set of labeled exemplars. In this paper, I present an architecture for a machine learning system that divides a trajectory into segments represented by motion vectors and spline curves. It is then trained to associate the trajectory with an animacy classification via a neural network. The system is compared to human judgments as well as to a system operating on knowledge of physical laws.

1 Introduction

The ability of an autonomous agent to determine when it is in the presence of another autonomous agent is a critical skill. For many animals, including humans, this ability is primarily dependent upon the visual sensory system. From the most basic survival instincts to the most sophisticated technical and leisurely pursuits, our daily activities rely heavily upon our capacity to quickly and accurately recognize the presence of a living being, whether tasty prey, hostile predator, or helpful social partner.

The facility to perform this identification must be fast and simple, and it must occur relatively early in the information processing pipeline. The identification of animacy cannot require the use of complex or high-level cognitive processing, which may not be available for an animal fleeing from danger or an infant seeking the attention of nurturing caregivers. It is rather the perceptual nature of this ability that is interesting, as it encroaches on domains traditionally thought to be purely cognitive [11].

Such a skill is equally important in the context of building robots and other artificial autonomous agents. As it becomes increasingly necessary and useful for artificial agents to interact with their human creators, the ability to detect the motion of an animate being is fundamental. The ability should, as in biological organisms, be fast and robust. Therefore, the classification of a motion as being animate or inanimate cannot be derived from a comprehensive decomposition and analysis of the vi-
ual field; current hardware is not fast enough and visual processing algorithms are not accurate (or general) enough. Even if the underlying object-recognition technologies were more advanced, it would still be necessary (and possibly more important) to be able to identify as animate those objects which are unknown or unfamiliar.

We may conceptualize such a classifier as being reducible to a “black box” which returns an animate/inanimate judgement based on a description of a particular motion in the visual field. Several design questions are raised in the creation of such a visual module: What is the appropriate format for a motion description? What are the characteristics of animate motion? Furthermore, is the ability to make this classification innate or acquired?

Prior work on such questions looks to Michotte’s [8] finding that we tend to describe animate motion in terms of intent and desire while we describe inanimate motion in terms of physical forces and laws. Leslie [7] proposed a series of modules which we may use to categorize the different types of motion perception. The Theory of Body (ToBY) module describes inanimate objects in terms of mechanical agency and an intuitive understanding of physical laws. If, for some reason, it cannot describe an object’s motion in such a mechanical way, the Theory of Mind (ToMM) is engaged and explains the events in terms of an agent’s actions and attitudes. In this framework, the ability of the Theory of Body module to distinguish animate from inanimate entities is prerequisite for the Theory of Mind module to explain events in terms of intentions (also known as “mentalizing” [5]).

1.1 Motion Classification

In order to build a machine capable of performing ToBY’s classification between animate and inanimate motion, it is useful to be able to extract information about the perceived spatio-temporal trajectory of the object (ignoring other information such as texture, size, or context). Reid [9] proposed a tracking algorithm, implemented by Cox and Hingorani [3], which produces a trajectory consisting of a set of points that identify an object as it moves through the field of view:

\[ T = \{ P_1, P_2, ... P_3 \} \]

Animations of such points (known as “moving-dot videos”) have been demonstrated to be capable of conveying a sense of animacy in human viewers ([6], [12]), which suggests that such a motion trajectory contains enough information to assign animacy to the object.

Scassellati [10] designed an architecture that classifies animate and inanimate stimuli using such motion trajectories, which were obtained from color saliency, motion detection, and skin color feature detection. A group of agents, comprising a sort of Theory of Body module, were each designed to represent knowledge of a basic property of the physical behavior of simple inanimate objects (i.e. straight lines, acceleration sign changes, and energy were evaluated by different agents). Based on how a given trajectory deviated from the behavior expected by an agent, it voted on the animacy of the object and expressed a certainty of its vote. This system as a whole represented an “innate” knowledge of how physical laws govern the motion of inanimate objects. When these basic physical laws are “violated” in some way, the system judges that an animate agent may be responsible. The system is ultimately judged on how closely its judgments match the ground truth (and the judgments of humans, which are not always correct).

There are, however, limitations to this model. First, it is dependent upon an explicit formulation of the laws of physics (or an ap-
proximation thereof). These laws are selected based on the degree to which they contribute to correct classifications, but they are not able to classify some types of stimuli properly, such as pendula, because humans are likely using a different set of rules to formulate their judgments (e.g. repetition of motion). It is unlikely that there exist neural structures which are innately organized to represent knowledge of physical laws. Secondly, the model is inflexible: it is unable to correct for errors that it has made, when these errors may contain very useful information about meaningful characteristics of animate motion. The idea of learning this discrimination has a biological basis: evidence shows that the ability to discriminate animate from inanimate motion is learned and then hard-wired at an early age ([2], [1], [4]).

This paper investigates a simple method for learning to perform the classification of animate and inanimate motion, without knowledge of physical laws, using a neural network. The system, which we will call Learned Theory of Body or L-ToBY, was trained on labeled exemplars of trajectory information. The results were then compared to human classifications and to the classifications of Scassellati’s ToBY. The learning system operates on (and thus evaluates the validity of) the general assumption that animate motion is more erratic and unpredictable than inanimate motion.

2 System Description

2.1 Information preprocessor

The first element of the system is the information preprocessor, which extracts information from the trajectory data and packages it as useful input to the neural network (Figure 1).
2.1.1 Obtaining trajectories from videos

The training data consisted of videos obtained by artificially generating trajectories (series of points, as described above). In the case of inanimate motion (static objects, pendula, and objects being rolled, thrown, and dropped), the trajectories were calculated from the appropriate physical equations. In cases of animate motion, the trajectories were obtained by capturing the mouse paths of an animate computer user. In order that we would be able to compare our system to Scassellati’s ToBY and to human subjects, we used Scassellati’s test data set [10] for evaluation. This data consisted of similar moving-dot videos obtained using the tracking algorithm due to Cox & Hingorani [3]. In both cases, the videos are sequences of frames in which there is a single identifiable bright dot representing the position of the object. A trajectory is assembled for a video from the coordinates of the brightest pixel in each frame of the video. The points of the trajectory are then spatially normalized (centered at zero mean and scaled to unit standard deviation).

As the videos can be of arbitrary length, each trajectory is broken up into thirty subdivisions (we will refer to the points which divide these sections as “breaks”). As each movie is approximately 120 frames, this number of breaks divides most of the trajectories into subdivisions of approximately four data points.

2.1.2 Fitting splines to trajectories

Because the points obtained from the video frames may be noisy and contain some jitter, a piecewise-polynomial cubic spline curve is fitted to the coordinate points using least-squares approximation. A polynomial curve, represented by four coefficients, is fitted to each of the subdivisions. The spline curve serves both to smooth the data as well as to provide potentially useful information (in the coefficients) about the curvature of the trajectory (Figure 2).

2.1.3 Obtaining motion vectors

Once the data has been smoothed by the spline curve, motion vectors (consisting of an angle $\theta$ and a length $\rho$) are calculated between the breaks. These motion vectors provide potentially useful information about the changes in direction and velocity of the object (Figure 3).
2.1.4 Statistical data analysis

A given trajectory has by now been divided into thirty sections, each of which is represented by four polynomial coefficients, an angle \( \theta \), and a distance \( \rho \). For each of these features, the mean, variance, and range are calculated for the entire trajectory, thus reducing it to a vector of eighteen components.

These statistics are motivated by real temporal characteristics of the trajectory: for example, the mean of the distance \( \rho \) is related to the average velocity of the object, and the variance gives an indication of how much the object may have accelerated or decelerated. The angle \( \theta \) gives information about directional changes and the spline coefficients relate to the sharpness of these changes. In general, these simple analyses indicate not only general characteristics of the motion in question but also how much the motion changes over the course of the trajectory. Such information may be useful if some of the primary characteristics of animate motion are unpredictability and variability.

2.2 Artificial neural network

The neural network is a multilayer feedforward backpropagation network (Figure 4). The input layer is fully connected to a hidden layer of the same size (eighteen neurons), which projects to a single neuron that performs the classification. The output neuron fires if and only if it classifies the motion as animate.

![Figure 4: Structure of the neural network.](image)

Transfer functions of all neurons are log sigmoid. The network training function updates weight and bias values according to Levenberg-Marquardt optimization. The network’s performance is measured according to the mean of squared errors.

The initialization of a neural network involves a randomization of the synaptic weights and neuronal biases. Therefore, multiple networks, though trained on the same training data set, may produce slightly different results in response to the test data. In order to compensate for this dependence on initial randomization conditions, one hundred networks were trained and tested. The system’s overall response to a test stimulus was recorded as a percentage certainty of the animacy of the stimulus (or the number of networks that classified it as animate).

3 Results

The training data consisted of thirty labeled exemplars, sixteen of which were inanimate and fourteen of which were animate (Table 1). As described above, the inanimate exemplars were generated from the physical equations of a number of different types of motion, including objects that are swung, thrown, rolled, and dropped. After the networks were trained on this data, they were simulated with Scassellati’s test data presented as input (Table 2). The results are displayed in Table 3. Twenty-two out of thirty test stimuli were correctly classified, only one less than Scassellati’s ToBY.

3.1 Additional tests

Four additional tests were conducted to evaluate the effect of different configurations on performance. The first two tests modified the manner in which the trajectories (of both training and testing data) were broken into subdivisions. The second two tests modified the structure of the neural network and the data pre-
Table 1: Training data. Exemplars 1-16 are inanimate and exemplars 17-30 are animate.

Table 2: Scassellati’s testing data. Animacy given in Table 3.

1. Twenty subdivisions rather than thirty: The trajectories were divided into fewer and longer subdivisions. Twenty out of the thirty test stimuli were classified correctly. The differences from the normal network were that it was correct on stimulus 12, but was incorrect on stimuli 3, 24 and 27.

2. Forty subdivisions rather than thirty: The trajectories were divided into more and shorter subdivisions. Twenty-one out of the thirty test stimuli were classified correctly. The differences from the normal network were that it was correct on stimulus 20, but was incorrect on stimuli 4 and 15.

3. Motion vectors only: The input vector consisted of only the statistical information (mean, variance, and range) of the motion vectors (composed of $\theta$ and $\rho$). Therefore the size of the input vector (and the hidden layer) was six. Twenty-one out of the thirty test stimuli were classified correctly. The differences from the normal network were that it was correct on stimuli 10, 12, 13, and 21, but was incorrect on stimuli 8, 15, 19, 24, and 25.

4. Splines only: The input vector consisted of only the statistical information (mean, variance, and range) of the four spline coefficients. Therefore the size of the input vector (and the hidden layer) was twelve. Twenty out of the thirty test stimuli were classified correctly. The differences from the normal network were that it was correct on stimuli 10, 12, 13, and 30, but was incorrect on stimuli 11, 15, 18, 24, 25, and 27.

All four tests performed slightly more poorly than the normal network, indicating a relative optimality of the normal system parameters. It also indicates that there is useful information contained in both the motion vectors and the spline coefficients.
4 Conclusion

Unfortunately, the sets of trajectories created by animate objects and those created by inanimate objects do not compose two linearly separable classes. The often subjective nature of this identification task, as well as the errors made by humans in such tasks, is a testament to the often blurred line between the two types of motion. After all, the wind blowing through the leaves of a tree creates motion that would hardly be described by any high-school-level physics we could hope to encode; conversely, a human being is quite capable of executing extremely precise “inanimate” motions. However, given a large enough input-vector dimensionality and enough exemplars, the proposed architecture should be capable of emulating an agent-based voting system operating on physical principles such as Scassellati’s. In this particular implementation, L-ToBY was able to correctly classify twenty-two out of thirty new stimuli after having been trained on only thirty training stimuli. This is only one more error than Scassellati’s ToBY, and six more errors than humans.

Since pendula were specifically a portion of the training data, L-ToBY was able to correctly classify stimulus 3 as inanimate where ToBY had failed. However, it also incorrectly classified the other pendula, 10, 12, 21, and 30. This is likely due to the difference in precision between the artificially created training pendula and the noisy results of the tracking algorithm in the test data. L-ToBY incorrectly classified stimulus 22, a straight-line movement, as animate, probably because the velocity of the point is not quite constant as it moves down the frame. L-ToBY also incorrectly classified thrown objects 7 and 13 as animate (agreeing with humans’ mistake on the latter) and 20 as inanimate (making the same mistake as ToBY).

Further work to attempt to correct these errors would involve the use of more exemplars, especially those obtained from the results of Reid’s tracking algorithm on real videos. The perfection of the inanimate stimuli in the training data likely accustomed the network to expect far less noisy data for inanimate motion. Additionally, the exploration of different types of trajectory decomposition and statistical analysis, as well as a deeper investigation of which elements contributed most strongly to the classification task, would be in order.

In general, the L-ToBY system performed very similarly to ToBY, being fooled by mostly the same stimuli. It did this without having had any innate knowledge of physics; rather, it was trained on “inanimate” stimuli that were created according to physical laws and on “animate” stimuli that were not. This points to the real potential strength of this architecture: it is able to incorporate new knowledge about the animate/inanimate classification, including knowledge that may not have been immediately apparent to the designer of such a system, based on errors it has made. This type of learning system is more flexible, more in line with what we may expect to find in biological brains, and requires less explicit knowledge than a strictly physics-based approach.

References


hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. 


<table>
<thead>
<tr>
<th>Stimulus category</th>
<th>Stimulus number</th>
<th>Ground truth</th>
<th>Human judgment</th>
<th>Scassellati’s ToBY judgment</th>
<th>L-ToBY judgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static objects</td>
<td>2</td>
<td>Inanimate</td>
<td>3%</td>
<td>Inanimate</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Inanimate</td>
<td>6%</td>
<td>Inanimate</td>
<td>0%</td>
</tr>
<tr>
<td>Thrown objects</td>
<td>7</td>
<td>Inanimate</td>
<td>44%</td>
<td>Inanimate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>Inanimate</td>
<td>53%</td>
<td>Inanimate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>Animate</td>
<td>78%</td>
<td>Inanimate</td>
<td>11%</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>Animate</td>
<td>81%</td>
<td>Animate</td>
<td>97%</td>
</tr>
<tr>
<td>Circular movements</td>
<td>5</td>
<td>Animate</td>
<td>59%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Animate</td>
<td>81%</td>
<td>Animate</td>
<td>56%</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Animate</td>
<td>81%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>Animate</td>
<td>78%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>Animate</td>
<td>56%</td>
<td>Animate</td>
<td>81%</td>
</tr>
<tr>
<td>Straight line movements</td>
<td>4</td>
<td>Inanimate</td>
<td>47%</td>
<td>Inanimate</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Inanimate</td>
<td>36%</td>
<td>Inanimate</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>Inanimate</td>
<td>30%</td>
<td>Inanimate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>Animate</td>
<td>53%</td>
<td>Animate</td>
<td>89%</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Inanimate</td>
<td>37%</td>
<td>Inanimate</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>Animate</td>
<td>75%</td>
<td>Animate</td>
<td>90%</td>
</tr>
<tr>
<td>Pendula</td>
<td>3</td>
<td>Inanimate</td>
<td>16%</td>
<td>Animate</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Inanimate</td>
<td>12%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>Inanimate</td>
<td>31%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>Inanimate</td>
<td>19%</td>
<td>Animate</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Inanimate</td>
<td>6%</td>
<td>Animate</td>
<td>79%</td>
</tr>
<tr>
<td>Erratic movements</td>
<td>1</td>
<td>Animate</td>
<td>97%</td>
<td>Animate</td>
<td>97%</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Animate</td>
<td>75%</td>
<td>Animate</td>
<td>87%</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Animate</td>
<td>31%</td>
<td>Animate</td>
<td>97%</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Animate</td>
<td>75%</td>
<td>Animate</td>
<td>94%</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Animate</td>
<td>87%</td>
<td>Animate</td>
<td>97%</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Animate</td>
<td>93%</td>
<td>Inanimate</td>
<td>95%</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>Animate</td>
<td>81%</td>
<td>Animate</td>
<td>88%</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>Animate</td>
<td>90%</td>
<td>Animate</td>
<td>81%</td>
</tr>
</tbody>
</table>

Table 3: Comparison of animacy judgements between humans, Scassellati’s ToBY system, and Learned-ToBY (with trajectories divided into 30 subdivisions). For L-ToBY, percentage indicates number of networks (out of 100) that classified the stimulus as animate. Bold items indicate a disagreement with the ground truth.
A MATLAB code

A.1 runall.m

% Create training data
traindata = getinput('train', 1:30, 30);
traintarget = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
% Create test data
testdata = getinput('test', 1:30, 30);
testtarget = [1 0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 1 1 1 1 0];

for i = 1:100
    traininput = traindata;

    % Create neural network
    [inputsize numsamples] = size(traininput);
    net = newff(minmax(traininput), [inputsize 1],
            {'logsig' 'logsig'}, 'trainlm', 'learngd', 'mse');
    net.trainParam.epochs = 200;
    net = train(net, traininput, traintarget);
    trainoutput = round(sim(net,traininput));

    % Run test data on neural net
    testinput = testdata;
    testoutput(i,:) = round(sim(net, testinput));
end
result = sum(testoutput);

A.2 getinput.m

function [ data ] = getinput( directory, number, deslength )

%GETINPUT Generates input vectors for neural network from avi movies

for i = number
    filename = [[directory '\movie'], int2str(i), '_1.avi'];
    [pnts(:,:,i) pntsimages(:,:,i)] = avi2pnts(filename);
    tempsp = pnts2sp(pnts(:,:,i), deslength);
    %figure;
    %plot(pnts(1,:,i), pnts(2,:,i), 'xr'), hold on;
    %fnplt(tempsp, '-b'), hold off;
    %figure;
    %plot(pnts(1,:,i), pnts(2,:,i), 'xr'), hold on;
    tempvecs = sp2vecs(tempsp);
% Motion vectors
data(1,i) = mean(tempvecs(1,:)') % Theta mean
data(2,i) = var(tempvecs(1,:)') % Theta variance
data(3,i) = max(tempvecs(1,:)') - min(tempvecs(1,:)') % Theta range
data(4,i) = mean(tempvecs(2,:)') % Rho mean
data(5,i) = var(tempvecs(2,:)') % Rho variance
data(6,i) = max(tempvecs(2,:)') - min(tempvecs(2,:)') % Rho range

%Spline coefficients
data(7,i) = mean(tempsp.coefs(1,:)') % 1st order mean
data(8,i) = mean(tempsp.coefs(2,:)') % 2nd order mean
data(9,i) = mean(tempsp.coefs(3,:)') % 3rd order mean
data(10,i) = mean(tempsp.coefs(4,:)') % 4th order mean
data(11,i) = var(tempsp.coefs(1,:)') % 1st order var
data(12,i) = var(tempsp.coefs(2,:)') % 2nd order var
data(13,i) = var(tempsp.coefs(3,:)') % 3rd order var
data(14,i) = var(tempsp.coefs(4,:)') % 4th order var
data(15,i) = max(tempsp.coefs(1,:)') - min(tempsp.coefs(1,:)') % 1st order range
data(16,i) = max(tempsp.coefs(2,:)') - min(tempsp.coefs(2,:)') % 2nd order range
data(17,i) = max(tempsp.coefs(3,:)') - min(tempsp.coefs(3,:)') % 3rd order range
data(18,i) = max(tempsp.coefs(4,:)') - min(tempsp.coefs(4,:)') % 4th order range
end

figure;
for i = number
    subplot(3,10,i);
    imshow(pntsimages(:,:,i));
end

A.3 avi2pnts.m

function [ pnts, pathimage ] = avi2pnts( inputavi )

%AVI2PNTS Converts a moving-dot avi movie to a set of coordinate points

% Read in the movie file
fileinfo = aviinfo(inputavi);
numframes = fileinfo.NumFrames;
aviheight = fileinfo.Height;
aviwidth = fileinfo.Width;
rgbmovie = aviread(inputavi);
pathimage = zeros(aviheight,aviwidth);

% Extract point coordinates
for f = 1:numframes

frame = rgbmovie(f).cdata;
maxrg = max(frame(:,:,1), frame(:,:,2));
maxframe = max(maxrg, frame(:,:,3));
[i,row] = max(max(maxframe'));
[i,col] = max(max(maxframe));
pathimage(row, col) = .25 + .75*f/numframes;
coords(:,f) = [col; aviheight - row];
end

% Normalize the data
largestd = max([std(coords(1,:)) std(coords(2,:))]);
if largestd == 0
    pnts(1,:) = (coords(1,:) - mean(coords(1,:)));
    pnts(2,:) = (coords(2,:) - mean(coords(2,:)));
else
    pnts(1,:) = (coords(1,:) - mean(coords(1,:)))./largestd;
    pnts(2,:) = (coords(2,:) - mean(coords(2,:)))./largestd;
end

A.4 pnts2sp.m

function [ sp ] = pnts2sp( points, deslength )

%PNTS2SP Converts a set of coordinate points to a ppform-spline

[xy, time] = size(points);
bsp = spap2(deslength, 4, 1:time, points);
sp = fn2fm(bsp, 'pp');

A.5 sp2vecs.m

function [ vecs ] = sp2vecs( sp )

%SP2VECS Converts a ppform-spline to a set of motion vectors

hold on;
for i = [2:sp.pieces+1]
    brk = fnval(sp,sp.breaks(i));
    prevbrk = fnval(sp,sp.breaks(i-1));
    vector = brk - prevbrk;
    [THETA, RHO] = cart2pol(vector(1), vector(2));
    vecs(:,i-1) = [THETA; RHO];
    %plot([prevbrk(1); brk(1)], [prevbrk(2), brk(2)], '-o')
end