MATH 801
Amir Assadi

Applications of Differential Geometry in Vision

Perceptual Geometry of Motion

Introduction. Study of vision, especially human visual perception of the environment surrounding us, has a long history in science and philosophy. The emergence of computational methods due to the advent of digital computers influenced progress in this field, bringing a host of mathematical methods to the vision research arena. Geometry, especially differential geometry and topology, have played a central role in modern vision research. A key player and one of the most original minds in modern vision research is the late David Marr whose book "Vision" is now a classic. David Marr was originally educated as a differential topologist at Cambridge, and switch to study brain research at late stages of his graduate work. Equipped with a profound understanding of mathematics, Marr proposed the first comprehensive computational theory of vision, and proposed a detailed program for solution of problems of visual perception. Despite its great popularity and many successful breakthroughs, Marr's computational theory could solve a limited number of problems and left the door open for many more developments to follow. The problem of computational modeling of visual perception is still an open challenging area of active research in brain research, artificial intelligence and psychology. Besides Marr's book, there are excellent expositions of progress to date in several books. The comprehensive textbook by Steve Palmer (Vision Science, MIT Press) is a superb resource with numerous well-exposed examples. A very short and nice reference is the first two chapters of (about 40 pages of easy reading) "Sensation and Perception", Edited by Richard Gregory and Andrew Colman (Longman Essential Psychology).

I will try to provide a basic review of the main issues in human vision that are pertinent to the course this semester. My approach is to propose some open research problems in vision that are particularly interesting from the viewpoint of differential topology and geometry for class projects. You could or could not solve the main questions, as they are truly difficult in full generality. But you could be sure that attempting to solve the open problems would inspire creativity and a much deeper understanding of the materials in the course. Also, partial solutions to these problems, as long as the methods and/or the results are new (i.e. have not appeared in print in journals) could be presented in conferences and also used as a good first draft for a publishable research article in a high quality journal.

Please note that some of the mathematics discussed here may not have been covered yet in the lectures. You should have in mind a set of mile-stones for required readings and doing the mathematics and programming that is compatible with the progress in the mathematics in class. I will meet with you individually or in a small group to help you formulate a project, write a brief outline for the work that is involved, and plan the overall activity during the semester.

Theory of Curves and Problems in Motion Perception

The first topic in Math 561 this semester is differential geometry of curves. Since we think of curves as the trajectory of motion of a particle, it is useful to explore how far the classical theory of curves could shed light on modeling visual perception of motion of objects, people and other elements in the environment. The human visual system is very sensitive to the detection of patterns that are implicit in motion of human beings and other animals. Psychologists have long noticed that humans can efficiently recognize other human action patterns, and even they can attribute many features of psychological, biological and social relevance to other persons. In visual perception, the term *biological motion* refers to the motion of image parts created by the edges and extremities of a 3-dimensional object in space or its 2-dimensional photograph, drawing etc. Informally, the adjective *biological* is meant to exclude the visual effects that create a perceived sensation of movement in the image as a result of varying textures and other image information that do not physically move, such as in visual illusions. The beautiful and highly influential 1973 paper by Johansson provided a compelling perception of biological movements of human forms from movies in which only a very small amount information was made available to the observer's eyes (Visual perception of biological motion and a model for its analysis. Perception & Psychophysics, 14 (1973), 201-211; also look up the very readable Scientific American article June 1975, volume 232, pages 76-88, by Johansson with very nice illustrations.) Johansson attached small light sources at the joints of human subjects who were dressed in non-reflective black outfits. He asked them to move around in a completely dark room, to walk, to dance and to make a variety of other movements, and then filmed them. Johansson movies appeared as a collection of white dots moving on a black background. Observers viewing the film reported vivid impressions of human figures, even though the images contained only a very small number of isolated bright points. Modern demonstrations of the Johansson's effect have been created by a number of researchers using computers with similar results. Please see the web pages by G. Mather and M. Bach , Nikolaus Troje (follow the links in http://www.michaelbach.de, especially, http://www.michaelbach.de/ot/mot_biomot/index.html).

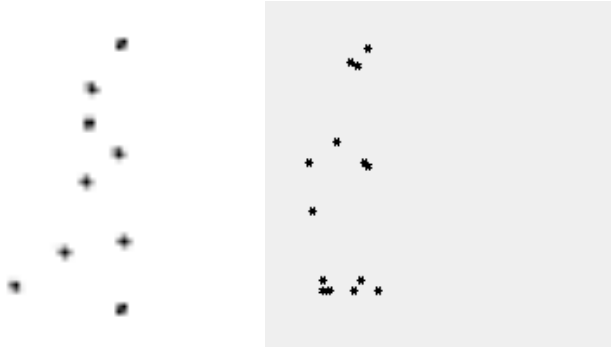
In the following, we wish to formulate several differential geometric problems that arise in the study of biological motion. Some of the following questions provide *open research problems*; that is, there is still plenty of room for coming up with innovative solutions, and such research has the potential to provide valuable and novel contributions to our understanding of how the brain perceives motion.

Part I. Systems of Space Curves and Perception of Biological Motion

Read through Johansson's Scientific American article June 1975, volume 232, pages 76-88, or look up related references in WWW as mentioned above, where you can also examine a number of computer animations of moving sets of dots. Using a image-capture software, save a sequence of frames such as the ones below. The sequence of such images can be imported as data in MATLAB as a sequence of matrices whose entries are mostly zero (for background) and a small number of ones corresponding to the moving dot.

Problem One

When you observe the moving dots, you see the motion of an object (with volume!) in a three-dimensional environment that is implicit in the background. Can you model a realistic motion of the dots in a fixed coordinate system of $\Re^3$ (i.e. a preferred frame of reference for the environment's spatial coordinates) using a system of parameterized curves? One sensible approach is to construct the piecewise linear trajectories, then approximate the resulting curves by piecewise smooth curves (such as *splines*, e.g. using the SPLINE Toolbox of MATLAB that was written by Carl de Boor and is made freely available to UW-Madison community).



Discussion. The problem of reconstruction of space curves from their 2-dimensional projections is an example of *ill-posed problems* typical in vision science, and more generally, in modeling biological

and behavioral phenomena. In circumstances such as visual perception of natural scenes (or their synthetic reconstruction), the brain solves the ill-posed problem by extracting a host of additional visual cues (i.e. mathematical constraints) such as motion, texture, shadows of objects or shading due to variation of illumination in the scene. The mathematical modeling of the moving dots requires that you also extract the constraints by making concrete explicit hypotheses that relies on the implicit assumptions that your visual system relies upon when you perceive the coherent spatial motion of the object. Some of the web sites that illustrate research in biological vision allow you to change the representation from moving dots to moving *match-stick figures* (see G. Mather's or M. Bach's web pages mentioned above); that is, each frame depicts a 1-dimensional mathematical graph that describes the rigid rod-like connections among appropriate pairs of points. If we wish to encode more information, we could include directionality of the line segments joining pairs of points by prescribing a fixed orientation for the surface of the object (i.e. a vector field defined on the surface to signify the outward normal direction, and a choice of ordering for the three axes positioned on the surface whose third axis is the normal direction. You must also impose a globally coherent choice of all these local orientations by requiring that they all follow the right-hand rule when you translate one frame to the one neighboring them.) These additional constraints are certainly mathematical consequences of assuming that the coherent motion of dots is due to movement of a 3-dimensional object. How do consecutive frames relate to each other, say in a motion forward? The line segment joining the two points corresponding to motion of the same dot is the first approximation to the velocity vector of the curve that encodes the trajectory of that dot in space. The consecutive frames, when superimposed, provide projections of the spatial dots to displacement of dots in the image, or what is the same, the displacement of the dot in the image plane. In turn, this provides the numerical information regarding the projection of velocity vectors onto the image plane, i.e. a piecewise linear approximation to the projection of the space curve.

Problem TWO (Research Project)

Suppose we have many pieces of image sequences that record light spots (i.e. dots, as in the Johansson experiments) in various instances of movements by various actors, such as walking, running, dancing etc. Each image sequences corresponds to a 3-dimensional array of zeros and ones, say $\mathbf{A}=\{A(t_k)$: t_1, t_2 … are times of capturing a frame; each frame $A(t_j)$ is an n by m matrix $A(t_k)=(A_{ij}(k))$ $\}$. This a data set in which the laws of 3-dimensional motion of actors and objects are implicitly encoded, and from which an intelligent system could extract the mathematical relationships among them, for instance as the brain of observers reconstructs the biological motion from the Johansson experiments described above. Use the ideas of statistical learning theory as in the article

'A Learning Theoretic Approach to Differential Geometric …' to construct a model for differential geometry of curve-collections that arise from spatial motion of objects (also people, animals …). Such a theory could be used to distinguish the identities of objects that are sources of motion.

Problem THREE (Research Project)

This project provides a concrete industrial application for the theory that we seek in the preceding problem. Suppose we have many pieces of video sequences that record low resolution gray-scale images of people and objects moving in the range of image-capture of a fixed video camera (as in various instances of surveillance or security camera). Intuition shows (also rigorous mathematical reasoning using statistical learning theory supports) that there exist an optimal collection of points that capture most of the information in a piece of movie captured by the above-mentioned camera. If we could extract such a useful set of points, then we could use that to compress massive data sets by a very remarkable factor of data compression. As a result, security video cameras could be designed to use such algorithms in order to analyze on-line (i.e. at the time of operation) the motions and signal identity, threats, or harmless movements. Similarly, satellite images could be highly compressed for online analysis and identification of objects and their motion-types.

Part II. Movement of Crowds

We consider three scales for visual perception of motion. Each scale has something to do with how close the observer is to what she/he sees. For small distances, perception of motion of parts of body of another person is appropriate. For example, motion of facial muscles creates perception of smiling, laughter, sadness, fear, surprise, dislike … From a somewhat farther distance, movement within the environment is appropriate. These include, walking, running, ballet dance movements, jumping, diving, … While more muscles are employed by the subject (i.e. the person being observed) to accomplish such movements, the observer needs only partial information regarding the changes in time of various parts of body. This makes the problem both flexible (choose only a few landmark locations in subject's body) and challenging (which choices are the best, in the sense that they contribute most to our perception of motion,…). The last scale in this discussion is perception of movements of a group of people, doing specific tasks, or moving freely in a normal mode of daily life. The perception of motion of a group requires that the observer stay far enough so as to include most of the members in her/his natural viewing position. An application of modeling group motion is in animation art. Automatic understanding and classification of group motion from a video camera

can be useful in security issues,…. Below, we list the problems and add a brief discussion for the third problem.

- Motion of facial muscles/perception of emotions.
- Human walking or other movements.
- Motion of groups of organisms (crowds of people.)

The main task today (see below) is to model a simple version of #3 and to learn the theoretical and computational mathematics that would be used to produce the model. We can first generate an experiment model and then think about its genuine implementation. I will add some quick review of mathematical concepts to help the mathematically absent-minded!

Project Steps:

1) compute basic data statistics (mean, variance, etc.) using MATLAB. We have covered good ways to write scripts using `eval` and `feval`.
2) Generate a few examples from visual motion to study intuitively simple modeling of group motion. Use MATLAB, JAVA, C or anything you prefer.
3) Pattern recognition will play an important role, and requires some background to understand the statistical issues. Imagine that you have encoded the motion and represented the data somehow. Also, for the time being, think of the data as a collection of vectors in some vector space.
4) Statistical description of group motion could be done by computation/answering the following:
    a) Mean of the data.
    b) Is mean adequate? What is the variance of the data?
    c) Are mean and variance adequate when we have multidimensional data? In what direction is the most variance and how much?
    d) Possible observation in most multi-dimensional data analysis cases: the data representation is not appropriate for the statistics of data coordinates.
5) A possible approach is to find directions of maximum variance and use them as part of the new coordinate system adapted to the statistics of the data.
6) Another view of variance is the inner product of the feature space.
7) Correlation Matrix and it's linear algebra: it's a symmetric real matrix which is diagonalizable (has real eigenvalues and a basis for vector space consisting of eigenvectors that are orthonormal.

A very useful and classical tool in linear algebra that we have studied is "singular value decomposition" (SVD) for matrices. This method goes back to Eugenio Beltrami, the Italian mathematician who discovered it and published the result in 1876. If this method is still popular, it is

a good indication of how useful it must be! Let us apply this result in the context of data analysis that we have learned, and where the data is from motion. We have also learned the Principal Components Analysis (PCA), and how it relates to SVD. The first published paper on PCA in this form is by Hotellings (appeared approximately about 1930). Just like SVD, analysis of data using PCA is very popular and useful. PCA is sometimes useful for re-parameterization of data in terms of much a smaller number of *uncorrelated* parameters, that is, a good selection of new *orthogonal* coordinate axis for the space containing the data. This procedure is sometimes called *"reduction of dimensionality."*

Ultimately, we aim at designing a neural network that "perceives" motion from a sequence of images. Such a goal is quite difficult to achieve in full generality, and in fact, an active area of computer vision and perception. We will make a compromise. First, we solve a basic case, described below. Next, we try to find a more sophisticated architecture for a neural network that would generalize the first case as far as possible. Let us begin with the simpler problem and a simpler architecture.

We wish to apply PCA in this project/exercise in two ways: First, using procedural programming, to reduce dimensionality of motion data to the bare minimum, where the motion parameters become visible and the motion can be understood in simple terms. Another look at PCA is to devise a two-layer feed-forward artificial neural network that extracts the first few principal components. This network has a very simple architecture. The input vector is X, and the output vector is Y. The synaptic weight matrix $W = (W_{ij})$ acts by matrix multiplication. Therefore, the entire calculation performed by a neuron β is to receive its input from all neurons α, via the formula

$$y_\beta = \sum_\alpha W_{\beta\alpha} x_\alpha ,$$

and to broadcast it via a nonlinear function $f(y_\beta)$. We have seen that an iterative algorithm starts with a general random vector of length 1, and converges to the first principal component of the symmetric matrix A. Therefore, in this set up the weight matrix does not change, and the network is simply a parallel distributed implementation of matrix multiplication: $Y = WX$ followed by a nonlinear operation $Z = (f_1(Y), f_2(Y), ...)$. We could use the output of step one as input of step two, or do a preprocessing on the output of step one and use the result as the input for step two. For example, starting with a unit vector X_1=X, we obtain the output vector Y=Y_1, and use it as part of data to feed in the network. To do this, the preprocessing stage could consist of testing if the input vector is an eigenvector, and if not, normalizing it and feeding it in the network. Thus here, we calculate X_1 = (1/||Y||), and use X_1 as the input for stage 2, and continue until we text for the convergence of the sequence of unit vectors to an eigenvector of A.

Exercise. (1) Assume a symmetric matrix A is given. Write the MATLAB code that sets up the ANN above, and computes the first principal component from any random unit vector as input. **(2)** Generalize this result to an ANN that extracts the first two or three principal components.

Therefore, you should generate the data, perform the basic statistical organization of the data, prepare the data for the experiments as the preliminary steps before doing PCA, or using other methods such as Artificial Neural Networks.

Review of The Basic Idea of PCA. Imagine the data is a set of vectors in R^N, and the mean of these set of vectors (i.e. the mean of the data) is exactly the origin (i.e. the 0 element) at the vector space. Then it is possible to choose a (possibly new) set of orthogonal coordinates by rotating the current standard coordinates such that the new coordinate directions (i.e. the new basis vectors) point to the directions of maximum variance in a decreasing order (i.e. the first direction points to the direction of maximum average dispersion of the vectors around 0, the second direction is the next largest variance, and so on.) These are the *principal directions.*

Algorithm for PCA:

1) Find the matrix C by finding the cosine of the angle between all pairs of vectors in the data set. This is called the correlation matrix for the data.
2) Find ***the eigenvalues*** and sort them in decreasing order. An ***eigenvalue*** is a solution for the unknown scalar value λ in the system of equation given by the matrix form $CX = \lambda X$. The unknown vector can be also determined, and that is called the eigenvector corresponding to the eigenvalue λ.
3) Find the corresponding basis of eigenvectors $\{e_{1,\ldots,} e_k\}$ (often only the first ones are needed.)
4) Find the new coordinates of the data points with respect to principal components
5) Algorithm for reduction of dimensionality:
6) Assume the error we can tolerate is some $\epsilon > 0$, given in a percentage.
7) Find the eigenvalues and sort them in decreasing order.
8) Sum the eigenvalues to find the trace(**C**)
9) Use the sum $\lambda_1 + \lambda_2 + \ldots + \lambda_k$ for the smallest k such that the ratio satisfies the error tolerance bound $(\lambda_1 + \lambda_2 + \ldots + \lambda_k)/tr(\mathbf{C}) < \varepsilon$.

1) $\{e_{1,\ldots,}e_k\}$ = W spans the subspace. Now project the data on W, the subspace of smaller dimension k versus the original space of dimension n.

An Example:

1) Let's consider a set up where a camera in a public place, say an airport or a busy shopping center, looks down on the crowd and focuses on a specific area. The camera is assumed to have a sensor made of one CCD chip (Charged Couple Device), and creates an input consisting of black and white images that are 100x100 pixels, "1" meaning black, "0" signaling white. A 2x2 block represents each individual (or use simply one pixel, given by a 1x1 block.) At most there are 2500 people in each frame. In reality, the matrix is sparse and only a few blocks are non-zero and the rest of the matrix is all zero. If the camera sees 30 frames/second, then in a 5 second recording, it will see 150 frames. We will assume the only movement the camera will see within those few seconds is an individual moving in a straight line or at most changing direction (5 seconds isn't that long). We can sort the video frames as an array of such matrices 150=5x30, so then we have 150 matrices in the array recording a 5-second time interval. This is the *data acquisition step*, passing its output to the computer (the brain).

2) Now we can make a simulated model of the 150 frames, by introducing an array of 100x100 matrices for each person P_m. for each instant, which we call $P_m(t)$. Each $P_m(t_s)$ at frame t_s has only one non-zero block. The array of all the individuals then describes the motion of one person, while all other individuals are filtered out from the frames.

3) Now we need to make a choice for how many people within the 5 seconds are within the range of the camera (some enter and leave the scene or are only partially in the scene).

4) Use a linear (or piecewise linear with at most 2 line pieces) function for each short motion of individuals to make a somewhat random selection of individuals to determine their motion as walking in a hallway, exiting a room, etc.

5) Overlay all the matrices to make the camera movie: M(t) = sum from m=0 to 24 of $P_m(t)$ [for 25 people]. Use MATLAB to run the movie. Observe if there is any *dominant motion.*

6) You can experiment by reducing the matrix to 50x50 or 20x20 rather than 100x100 to see if you can still determine the general direction of motion.

The following is a more mathematical description of the above example:

Question: What structures (i.e. geometric) can be perceived in this 5 seconds of video?

Every person's motion will produce a vector $v_m \in R^{150\times10,000}$. Suppose we have a data set D. The data corresponding to this model is from r individuals where $r.10^4$ points in $R^{150,000}$. Therefore, our data set Ψ is sparse. Also, the geometry of this data set has a meaning because it has a metric (from inner product).

Examples of geometric structures:

* points in D line on a line, plane, sphere, etc.

* examine the spread of the points to interpret specific types of movement (i.e. if patterns are in a line, movement is in the same general direction)

We need to single out individuals:

t = time; r = # individual; p = person

$M(t) = P_1(t) + ... + P_r(t)$

$t = t_0, ... , t_{149}$ frames each $P_i(t_j)$ has only one non-zero 5x5 block.

PROBLEM: Find an algorithm that decomposes the video into a sum of individuals moving. (structure from motion).

For synthetic data, define $P(t_j)$ as a matrix as above with only one block non-zero. Use a 20x20 block to reduce the size of data. We will lose information in terms of velocity and the representation of the person. (Can we reduce the size even more though?) Represent $P_m(t_k)$ in R^{400}. Motion of one person is $v_m \in R^{60,000}$.

BASIC PROBLEMS:

Suppose $D \subset R^n$; n = 6x 10^4; N = 1.5 x 10^5

a) Find any geometric structure in D

b) Describe or represent this structure in its simplest form

c) Suppose an error tolerance $\gamma > 0$ (given by a %) is given. Then do (b) taking advantage of this error tolerance. D = $\{v_r : r = 0,...,9999\}$.

The following are good examples for 1-dimensional data:

mean: $v_{mean} = (1 / \#D) \Sigma\{ v_s ; s = 0,...,9999\}$

variance: sum of squares of deviations

Translate coordinate system so that $v_{mean} = 0$. Variance will not be adequate, so we use other statistical moments:

s-th moment = Ix^s d: (over D)

variance (2^{nd} moment) = Ix^2 d: (over D)

mean (1^{st} moment) = Ix d: (over D)

Algorithm:

1) Form the correlation matrix C for D (normalized dot products between 2 vectors) i.e. $\cos\tau = \langle v_l, v_k \rangle / (\langle v_l, v_l \rangle^{1/2} \langle v_k, v_k \rangle^{1/2})$

C = $(\langle v_i, v_j \rangle)$ is symmetric with real entries

Ψ diagonalizable matrix with real entries that are the eigenvalues of C; $\lambda_1 > \lambda_2 > \lambda_3 > ....$ (we can select a new ON (orthonormal) coordinate system for R^n such that C is similar to a diagonal matrix with the eigenvalues on the diagonal.

Now we find $\alpha_k = (\lambda_1 + \lambda_2 + ... + \lambda_k) / (\lambda_1 + \lambda_2 + ... + \lambda_n)$ where the norm of each eigenvector equals one. This is the percentage of "information" that can be carried by projecting the data onto the span $\{e_1 ... e_k\}$. If $1-\alpha_k < \gamma^{error}$, then we are within γ-error. If γ=5%, we should choose the smallest k such that $\alpha_k < 0.95$. The following is the MATLAB code that generates a sequence of random dots moving within a window. The explanation and help for understanding the code is provided below. You should first try to understand the ideas in this code, and then modify it (and/or write new code with similar or more elaborate properties.) Such codes generate data that helps you to subsequently experiment with ideas of PCA, motion perception, ….

walksequ1.m

Description for the coherent movement code[1]

This is the code for the sequence of random dot movements that is extensively commented. Before you read the code the following short explanation may help you understand the basic idea behind the code.

This code renders the movement of a single dot in a 20x20 pixel window. By repeatedly rendering a single pixel and adding the effects the overall motion of the dots is constructed. The routine is supplied with three inputs that specify the number of dots to render, the number of frames to render and the percentage of dots that move coherently. Each invocation of the routine creates a random 3x3 matrix that is used to determine the predominant direction of motion. For example, the four matrices below (from left to right) indicate movement to the left, and movement to a location approximately to the middle of top and right. This last matrix is then decomposed into the two final matrices of movement toward the top and movement to the right.

1	0	0
0	0	0
0	0	0

0	0.5	0.5
0	0	0
0	0	0

0	0.5	0
0	0	0
0	0	0

0	0	0.5
0	0	0
0	0	0

The direction matrix is computed once (see line 15) and randomly perturbed each time the dot is moved by element-wise multiplication with another random 3x3 matrix (see line 28). The lines highlighted in yellow in the code below are the crucial steps for the approach outlined here.

[1] The code and explanation is kindly provided by Dr. Hamid Eghbalnia (eghbalni@nmrfam.wisc.edu).

```
1.  function wss = walksequ(ndots, nframes,dc)
2.
3.
4.  walksequ renders ndots pixesl for nframes frames
5.  with degree of coherence dc (0<dc<1) in a 20x20
6.  pixel image.  dc determines what percentage of the pixels move coherently.  Simultaneous occupation of the
    same location is allowed!
7.  Example: ws=walksequ(15,20,0.8); 15 dots, 20 frames with 80% coherence
8.  For illustration purposes only
9.
10. dc = round(ndots*dc);              % dc is the number of dots that move coherently
11. dc = ndots - dc;                   % now it is the number of incoherent dots
12. wss=zeros(20,20,nframes);
13. for j = 1:ndots
14.     if dc > 0                      % we first render the incoherent dots
15.             dm = round(6*full(sprand(3,3,0.4)))   % for each incoherent dot choose a direction
16.             dc = dc -1;
17.     end
18.     h = zeros(20,20);
19.     stp = floor(20*rand);        %random starting point for the pixel
20.     if stp ==0
21.             stp = 10;
22.     end
23.
24.     h(stp,stp)=10*j;               % assign a color value to the dot
25.     p1=stp;p2=stp;
26.
27.     for i=1:nframes
28.             cm=dm.*rand(3,3); %randomly perturb the direction
29.             [k,l]=find(cm==max(max(cm)));   %move in the direction of the maximum weight
30.             k=k-2;
31.             l=l-2;
32.             %imagesc(h); drawnow;
33.             %uncomment the line below if you want to see
34.             %dot movements with no traces
35.             h(p1,p2)=0;       % the logic below is to deal with boundaries
36.             if (p1+k)==0      % this logic makes the dot move on a torus
37.                     %k = -k;  %bounce
38.                     k = 20-p1;
39.             elseif (p1+k)==20
40.                     %k = -k;
41.                     k=1-p1;
42.             end
43.             if (p2+l)==0
44.                     %l = -l;
45.                     l = 20-p2;
46.             elseif (p2+l)==20
47.                     %l = -l;
48.                     l = 1-p2;
49.             end
50.
51.             h( min(p1+k,20),min(p2+l,20) )=4*j;
52.             %size(h), size(wss)
53.             wss(:,:,i) = h+wss(:,:,i);       % add the effect of the previous dots
54.             p1=min(p1+k,20);
```

```
55.                 p2=min(p2+l,20);
56.                 %pause;
57.         end
58.  end
59.  for i=1:nframes        %now render the whole movement for 20 frames
60.         for j=1:5        % repeat each frame 5 times.
61.                 imagesc(wss(:,:,i),'EraseMode','Xor');drawnow;
62.         end
63.  end
64.
```

Web Sites of Interest

http://www.michaelbach.de/ot/mot_biomot/index.html.
www.michaelbach.de
http://photo.ucr.edu/photographers/muybridge/contents.html

http://www.psy.vanderbilt.edu/faculty/blake/BM/BioMot.html

REFERENCES and SUGGESTED READING

Johansson G (1973) Visual perception of biological motion and a model for its analysis. Perception & Psychophysics, 14, 201-211.

Kozlowski L T & Cutting J E (1977) Recognizing the gender of walkers from dynamic point-light displays. Perception and Psychophysics 21 575-580.

Mather G & West S (1993) Recognition of animal locomotion from dynamic point-light displays. Perception, 22, 759-766.

Mather G & Murdoch L (1994) Gender discrimination in biological motion displays based on dynamic cues. Proceedings of the Royal Society of London, B, 258, 273-279.

Fox R & McDaniel C (1982) The perception of biological motion by human infants. Science, 218, 486 487.

Grossman E & Blake R (1999) Perception of coherent motion, biological motion and form-from-motion under dim-light conditions. Vision Research, 39, 3721-3727.

Grossman E, Donnelly M, Price R, Morgan V, Pickens D, Neighbor G & Blake R (2000) Brain areas involved in perception of biological motion. Journal of Cognitive Neuroscience, 12, 711-720.

Troje, N. F. (2002). Decomposing biological motion: A framework for analysis and synthesis of human gait patterns. Journal of Vision, 2:371-387