# im2vec

April 14, 2012 Bill Freeman, CSAIL, MIT
this file: notes/im2vec.tex

## 1 Introduction

See also "notes/transrep.tex" for background to this image representation. This note is more of a lab notebook recording the experiments and conclusions. I the notation below, I think I need to separate out the image itself from the pixel representation, which in the write-up below, are confounded.

## 2 Toy worlds

### 2.1 toy world 1

Consider this simple sequence of 1-d images.

$$v_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \tag{1}$$

$$v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{2}$$

The observed temporal mid-point between $v_1$ and $v_3$ is

$$v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{3}$$

#### 2.1.1 pixel representation

But, of course, the vector average of $v_1$ and $v_3$ is

$$\frac{v_1 + v_3}{2} = \begin{pmatrix} 0.5 \\ 0 \\ 0.5 \end{pmatrix} \tag{4}$$

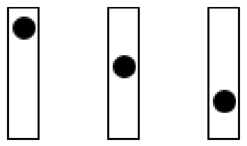
This particular example is a "hard-turning curve", where each vector in the sequence is perpendicular to all the previous vectors. But we can get an approximate measure for how well the pixel representation does by measuring the bending angle between the line segments $v_2 - v_1$ and $v_3 - v_2$. The cosine of this angle, $\theta$, is one if the representation captures the temporal propagation so the temporal curve of the images we view is a straight line, and less than one otherwise. How close that number is to one is a measure of how "good" an image representation we have, a representation that will let us treat images like vectors.

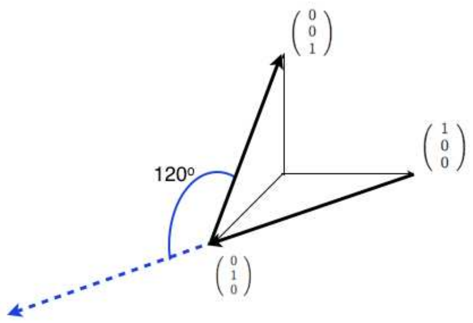
$$\cos(\theta) = \frac{(v_2 - v_1) \cdot (v_3 - v_2)}{\sqrt{[(v_2 - v_1) \cdot (v_2 - v_1)][(v_3 - v_2) \cdot (v_3 - v_2)]}} \tag{5}$$

In this example, we have $\cos(\theta) = -0.5$. (We can visualize this particular example–vectors along the unit axes in 3-space. The curve turns so sharply back on itself that the cosine is negative).

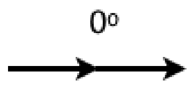
#### 2.1.2 better representation

For this simple world, a good representation is simply a scalar value that tells the coordinate where the "1" feature is. In this representation, $v_1 = 1$, $v_2 = 2$, $v_3 = 3$,. In this 1-d representation, there are only two options for the bending angle (0 or $\pi$), and the angle is 0 in this case.


(a) 3 frames



(b) sequence curvature, pixel representation



(c) sequence curvature, offset representation

Figure 1: (a) the images of toy world 1, 3 frames in temporal succession. (b) Showing the angle between successive frames as represented in a “pixel” representation. (c) The angle between successive frames in a “feature position” representation.

### 2.2 toy world 2

multiple dots: 5 entries, an image of 3 things moving.

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \tag{6}$$

The observed temporal mid-point between $v_1$ and $v_3$ is

$$v_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \tag{7}$$

$$v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{8}$$

#### 2.2.1 pixel representation

$$\cos(\theta) = \frac{(v_2 - v_1) \cdot (v_3 - v_2)}{\sqrt{[(v_2 - v_1) \cdot (v_2 - v_1)][(v_3 - v_2) \cdot (v_3 - v_2)]}} \tag{9}$$

Here, $\cos(\theta) = 0$.

#### 2.2.2 better representation

Now we have a vector of ink particles. We’ll use 3 here. (See Coulomb warp method for how to deal with a different number of reference shape ink particles than in the target image). We need a canonical assignment of ink particles to shape. We’ll assume a Coulomb-warp type of assignment system. Now each coordinate says how much to push this ink particle.

$$v_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{10}$$

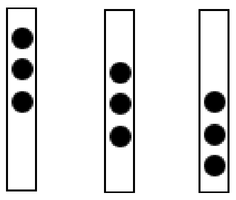
$$v_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{11}$$

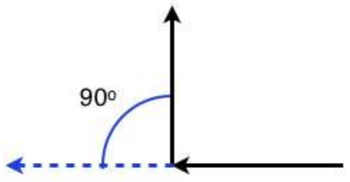
$$v_3 = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} \tag{12}$$

Now $\cos(\theta)$ = 1. Note we can get the same images out with a non-coordinated assignment of translations to ink particles, but then we get a bad angle. For example, if we use

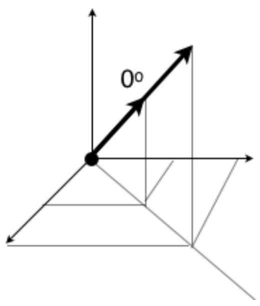
$$v_1 = \begin{pmatrix} 2 \\ -1 \\ -1 \end{pmatrix} \tag{13}$$

but keep the other two vectors the same, we get $\cos(\theta)$ = 0.5774, not as good as before.


(a) 3 frames



(b) sequence curvature, pixel representation



(c) sequence curvature, offset representation

Figure 2: (a) the images of toy world 2, 3 frames in temporal succession. (b) Showing the angle between successive frames as represented in a "pixel" representation. (c) The angle between successive frames in a "feature position" representation.

### 2.3 toy world 3

Now let's say we have multiple types of features

$$v_1 = \begin{pmatrix} A \\ A \\ G \\ 0 \\ 0 \end{pmatrix} \tag{14}$$

The observed temporal mid-point between $v_1$ and $v_3$ is

$$v_2 = \begin{pmatrix} 0 \\ A \\ B \\ G \\ 0 \end{pmatrix} \tag{15}$$

$$v_3 = \begin{pmatrix} 0 \\ 0 \\ A \\ B \\ G \end{pmatrix} \tag{16}$$

Let's say that features A and B are more similar to each other than they are to feature G.

## 2.4 pixel representation

Let's say the Euclidean representation has A=1, B=2, and G = 7, and we have $v_1 = \begin{pmatrix} 1 \\ 1 \\ 7 \\ 0 \\ 0 \end{pmatrix}$, $v_2 = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 7 \\ 0 \end{pmatrix}$, and $v_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 2 \\ 7 \end{pmatrix}$. Then $\cos(\theta) = \frac{(v_2-v_1)\cdot(v_3-v_2)}{\sqrt{[(v_2-v_1)\cdot(v_2-v_1)][(v_3-v_2)\cdot(v_3-v_2)]}} = -0.3974$, ($\theta = 113°$).

## 2.5 better representation

for $v_1$

| Feature | position | contrast |
|---|---|---|
| A | 1 | 1 |
| A | 2 | 1 |
| B | 1 | 0 |
| B | 2 | 0 |
| C | 0 | 0 |
| C | 0 | 0 |
| D | 0 | 0 |
| D | 0 | 0 |
| E | 0 | 0 |
| E | 0 | 0 |
| F | 0 | 0 |
| F | 0 | 0 |
| G | 3 | 1 |
| G | 3 | 0 |

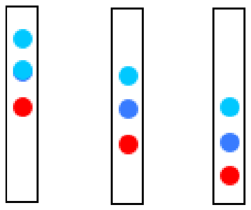
for $v_2$

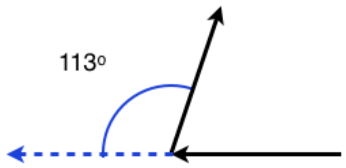
| Feature | position | contrast |
|---|---|---|
| A | 2 | 1 |
| A | 3 | 0 |
| B | 2 | 0 |
| B | 3 | 1 |
| C | 0 | 0 |
| C | 0 | 0 |
| D | 0 | 0 |
| D | 0 | 0 |
| E | 0 | 0 |
| E | 0 | 0 |
| F | 0 | 0 |
| F | 0 | 0 |
| G | 4 | 1 |
| G | 4 | 0 |

for $v_3$

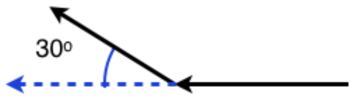
| Feature | position | contrast |
|---|---|---|
| A | 3 | 1 |
| A | 4 | 0 |
| B | 3 | 0 |
| B | 4 | 1 |
| C | 0 | 0 |
| C | 0 | 0 |
| D | 0 | 0 |
| D | 0 | 0 |
| E | 0 | 0 |
| E | 0 | 0 |
| F | 0 | 0 |
| F | 0 | 0 |
| G | 5 | 1 |
| G | 5 | 0 |

The $\cos(\theta)$ for these 3 vectors is 0.866. ($\theta = 30°$) Note that features A and B are assumed to be similar here, so that's why both A and B representation columns respond to each of the A and B features. We treat both the position and contrast entries on equal footing and treat them as equal units in the squared vector computation.


(a) 3 frames



(b) sequence curvature, pixel representation



(c) sequence curvature, offset representation

Figure 3: (a) the images of toy world 3, 3 frames in temporal succession. Now we have 3 features, translating across space. The color codes feature index. For the sake of this toy illustration, the states A, B, G in the text correspond to values 1, 2, 7. (b) Showing the angle between successive frames as represented in a "pixel" representation. (c) The angle between successive frames in a "feature position" representation. Because one feature changes into a nearby one in this sequence, the vector alignment is not perfect and there is an angle of 30 degrees.

### 2.5.1 another possible tweak

Note: we say, "quantizing in position is bad, let's treat that as a variable and take vector averages". But then the above representation needs to quantize in *appearance*. But we need to have these be sensitive to sift features somehow. So we could add a low-dim set of coefficients describing how the *appearance* of the sift feature changes. Then the row for a particular sift feature would include: position, appearance, and contrast values.

When you subtract the mean of these vectors, does that mean you subtract the mean *contrast* value, too? Do you then have features coming in with negative contrast? And how do you render a sift feature at negative contrast?

### 2.6 toy world 4

does additivity hold? if include extra things in the image, do the representations add? (eg, background clutter?) what would then let us use PCA on our new representation?

## 3 desired outputs

a collection of images and the desired mean image.

I guess the figure of merit could be how close the representation of the concocted mid-point image is to the mid-point of the representations. Like what fraction of the distance between the end-point vectors is it?

More training sequences could be temporal sequences from videos. Plus/minus 1 second, relative to zero time. So you could get lots of training images

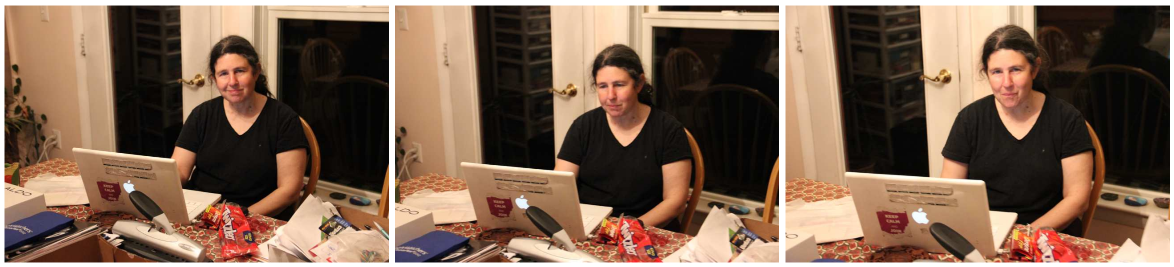


Figure 4: 3 images. What is the angle between these in a pixel representation? Originals are 3456x2304 pixels. In general we want temporal sequences to be straight lines in the image representation.

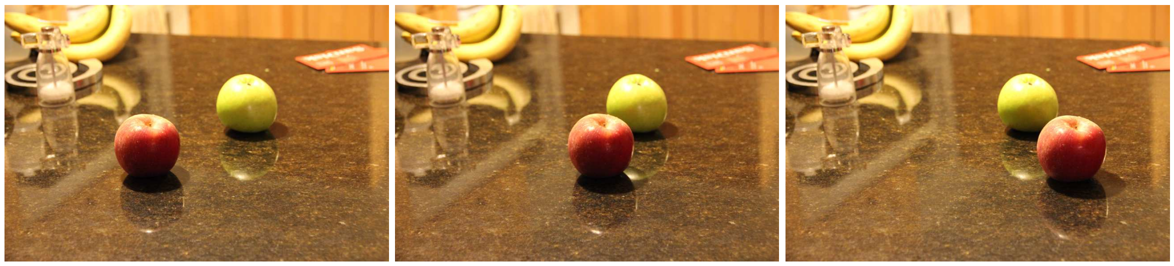


Figure 5: 3 images. What is the angle between these in a pixel representation? Originals are 3456x2304 pixels, and are in matlab/im2vec/photos. In general we want temporal sequences to be straight lines in the image representation.

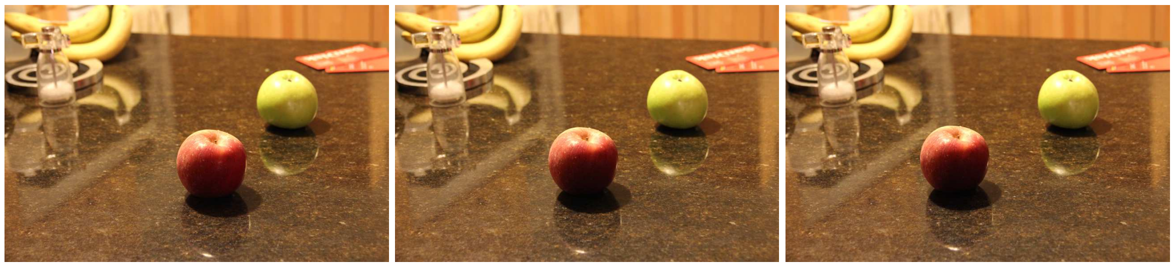


Figure 6: 3 images. What is the angle between these in a pixel representation? Originals are 3456x2304 pixels. In general we want temporal sequences to be straight lines in the image representation.

Show the curvature of these images in the different represetnations.

## 4 The representation

### 4.1 The distance function

### 4.2 Things to plot

You want temporal video sequences of images to lie on straight lines in your representation, not on curved lines. So (a) can use dimensionality reduction methods, and pairwise distances to plot the curvature of a video cliplet line in a pixel-wise representation. That will be cute little curved lines. And making straight lines of video sequences is a nice, implementable goal for a computer or an organism. (b) The easy way: for lots of image trios, calculate the angle of bending in the various representations. Calculating all 3 pairwise distances gives you that angle?. or

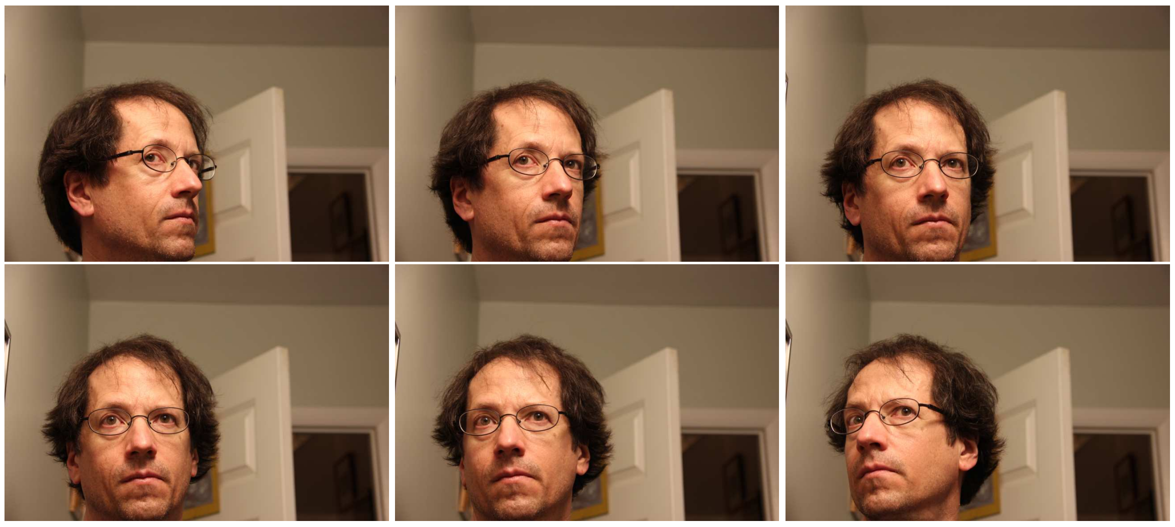


Figure 7: 3 images. What is the angle between these in a pixel representation? Originals are 3456x2304 pixels. In general we want temporal sequences to be straight lines in the image representation.

you could take the cosine of that angle. I suppose that all has to agree. So for some test set, compare the curvature values for the different experimental conditions you try out.

Think thru: For the single dot moving, and a sift feature change for the feature over time, with it being labeled A once, and A' a second time, what is the bending angle in the different candidate representations? want to rig it up so that it's nearly a straight line representation, even though the feature label for the dot changes over the sequence.

## 5 The im2vec algorithm

### 5.1 algorithm 1

send particles out into 3-space, with the attractor charges being in the plane, but the attraction and repulsion depending on the distance in scale or position or descriptor space. In addition, there is some kind of contrast mask needed to say which sift features are contrast-less.

How do the contrast-less feature matches enter into the distance function?

### 5.2 algorithm 2

single atoms of charge, in a very high-dimensional space. adjust the total charge to fit, then send off the field lines and see where they land.

## 6 functions to write and test

### 6.1 sift dist

functionalize the distance function between sift descriptors, since we expect to use AZ's silver bullet normalization on that. Of course, use vl_feat, and see the vl_feat demos for examples to use. cd to toolbox to run vl_setup. then you can run vl_demo and programs therein.

### 6.2 decide on sift vocabulary

3000 possible descriptors. quantize in scale, too?

make a figure showing the representation. and show a toy version of the representation.

create a visualization function. Perhaps come up with a better way of visualizing the 128 dimension sift feature descriptors.

### 6.3 control experiment

evaluate how well the control experiment would work: assign each sift feature in the image to the nearest neighbor descriptor.

There needs to be competition in both descriptors and in positions in order to get a canonical assignment. Is there a nice 2-d toy version of this?

can you come up with an objective function: you want a representation that is most robust to a small change in position or value of the image's positions or description? So can you make differential change arguments for stability? You can perhaps tune some of the affinity parameters that way.

Then you can also tune affinity parameters from a test set.

## 7 SIFT experiments to do (april 19, 2012)

- calculate pixel representation angle between trios of images.
- calculate sift vectors. put into some canonical assignment. calculate trio angles.
  - index sift features based on order program returns the features.
  - index sift features based on feature alone, ignoring spatial position
  - vector quantize sift features, and index sift features in a coulomb warp representation, using spatial position and taking into some account feature value.

## 8 Midpoint principle

midpoint principle: Let A, B, and C be 3 successive frames of a video, each temporally offset by some time shift $\delta$. We want to tune, tweak, or train our representation, R, so that, in the representation R, frame B is at the midpoint of frames A and C. ie, $R(B) = \frac{R(A)+R(C)}{2}$. Another way to state that is to say $\cos(R(A) - R(B), R(B) - R(C))$ should be maximized. (averaged over many videos, of course) This is a nice design principle that will let us optimize our representations.

what's the generalization of that for find good reps for object recognition? the subspace principle?

## 9 programming notes

## 10 distribution of SIFT distances.

```
im = single(imread([root num2str(1) '.jpg']));

load('randsift');
[f,d] = vl_sift(im) ;
vec = zeros(size(fallsel));
for i = 1:2
    [yy, ii] = min(sum((repmat(dallsel(:,i), [1,size(d,2)]) - d).^2, 1));
    vec(:,i) = f(:,ii);
end
vec = vec(:);

sum((repmat(dallsel(:,i), [1,size(d,2)]) - d).^2, 1)
% 5500 is a reasonable threshold for the error.  No:  it totally depends on
% the sift feature.  I should look and see how closely the image area
% matches to the areas of close match in the image.  like take an area of
% the input image and see what images are matching to it.  In descending
% order of the match.
```

make an insight matlab script: visualize_sift. input 2 images. look at sift matches. pull out the close-by images, and show them.

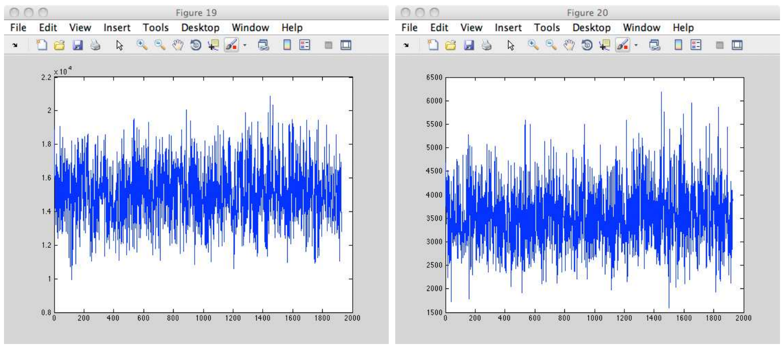


Figure 8: Note the widely different scales and variances of any given sift feature to all the others in the test image.

### 10.1 Bending angle as function of image temporal offsets

see tim.m, for the video 'photos/MVI_0078.MOV', we plot the bending angle for a pixel representation as a function of temporal offset. How do we expect it to behave? For very small temporal offsets, even the pixel representation images should be interpolatable (steerable, if you will). You see that in the bending angle. (a) of Fig. (10) shows the bending angle as a function of frame offset. ie, for x-axes label 10, (Not quite: For robustness, we averaged over that value taken at 4 neighboring time starts (with the appropriate subsequent frames offset accordingly).)

As a control, to see if the video itself was changing over that time, in (b) we plot the 1-frame offset bending angle for a range of starting frames in the video ranging from 1 to 57 (there were 59 frames in the whole video).

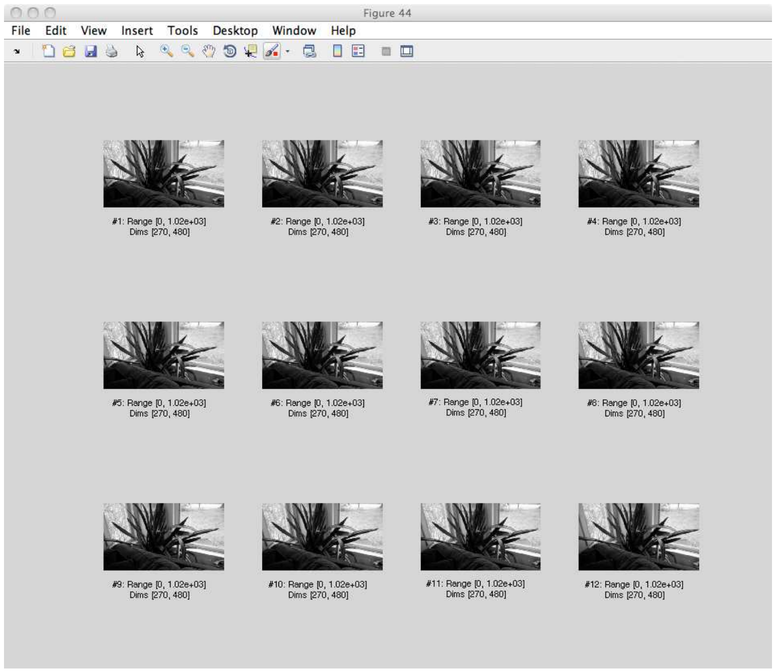


Figure 44
File Edit View Insert Tools Desktop Window Help
#1: Range [0, 1.02e+03] Dims [270, 480]
#2: Range [0, 1.02e+03] Dims [270, 480]
#3: Range [0, 1.02e+03] Dims [270, 480]
#4: Range [0, 1.02e+03] Dims [270, 480]
#5: Range [0, 1.02e+03] Dims [270, 480]
#6: Range [0, 1.02e+03] Dims [270, 480]
#7: Range [0, 1.02e+03] Dims [270, 480]
#8: Range [0, 1.02e+03] Dims [270, 480]
#9: Range [0, 1.02e+03] Dims [270, 480]
#10: Range [0, 1.02e+03] Dims [270, 480]
#11: Range [0, 1.02e+03] Dims [270, 480]
#12: Range [0, 1.02e+03] Dims [270, 480]

Figure 9: Plant video used for the calculations below. Static scene, slight camera pan. A 59 frame sequence. Here, frames offset by 5 frames are shown.

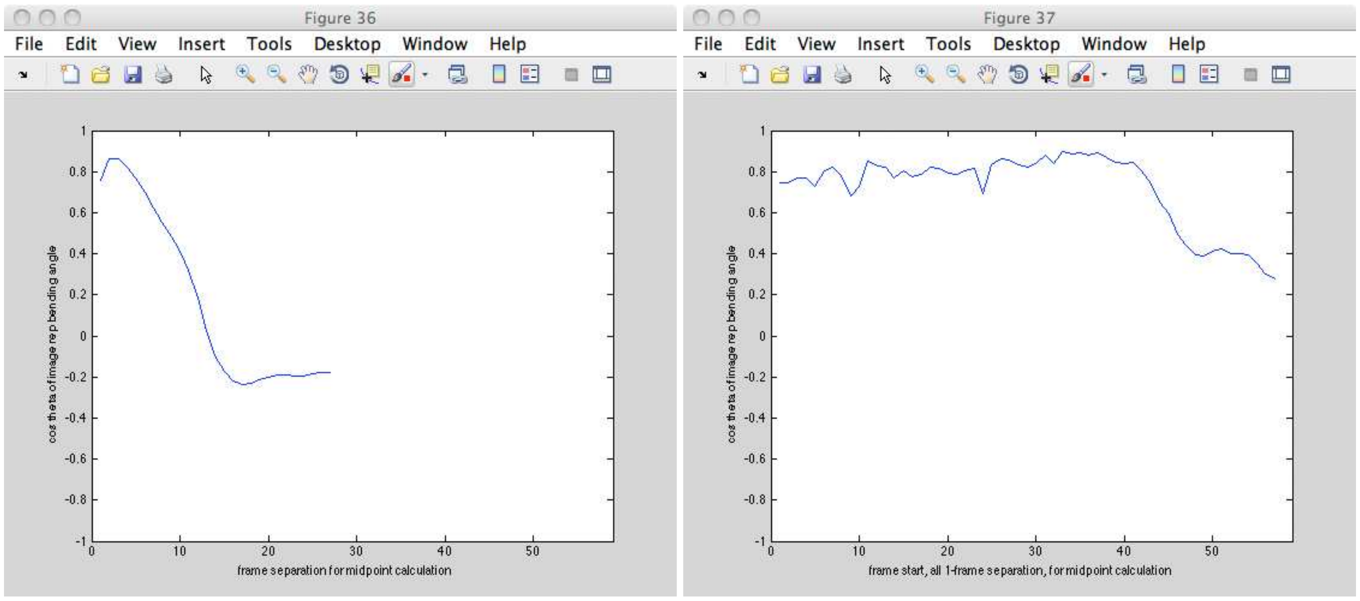


Figure 10: Both of these plots are for a pixel representation. (a) Bending angle, as a function of frame offset between each of the 3 frames in the mid-point calculation. Note that, even for a pixel representation, the mid-point assumption eventually holds, if you make the temporal offsets between frames small enough. (b) As a control (maybe the different in (a) comes because the video itself changes over that time), here is the 1-frame offset bending angle, sweeping over frame 1 through 57 of the sequence. There is some temporal variation, but it occurs at the wrong time to cause the effect from increasing the frame offset in the midpoint calculation.