Through Image Processing

Dr. Daniel B. Rowe
Professor of Computational Statistics
Department of Mathematical and Statistical Sciences
Marquette University

Outline

Introduction

Videos in Matlab

Pixel Temporal Convolution

Pixel Recursive Filtering

Discussion

Homework

Introduction

We often observe repeated images of a dynamic scene, a video.

We previously examined processing of a single image frame.

We can also process a single pixel over time, a time series.

When we performed convolution on an image, we were able to use wrap-around because the entire image was measured at the same time.

We generally process a pixel's time series of images as they come in, and occasionally after they have all come in forensically.

Videos in Matlab

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Videos in Matlab

Applied histogram equalization to enhance.

Note cat.

Videos in Matlab

Applied histogram equalization to enhance.

Cat moved.

Videos in Matlab

Applied histogram equalization to enhance.

Cat turned.

Videos in Matlab

Applied histogram equalization to enhance.

Cat leaving.

Videos in Matlab

Applied histogram equalization to enhance.

Cat gone.

Videos in Matlab

If you had repeated measurements, probably the first thing you want to do is average them.

But we lost the cat.

Videos in Matlab

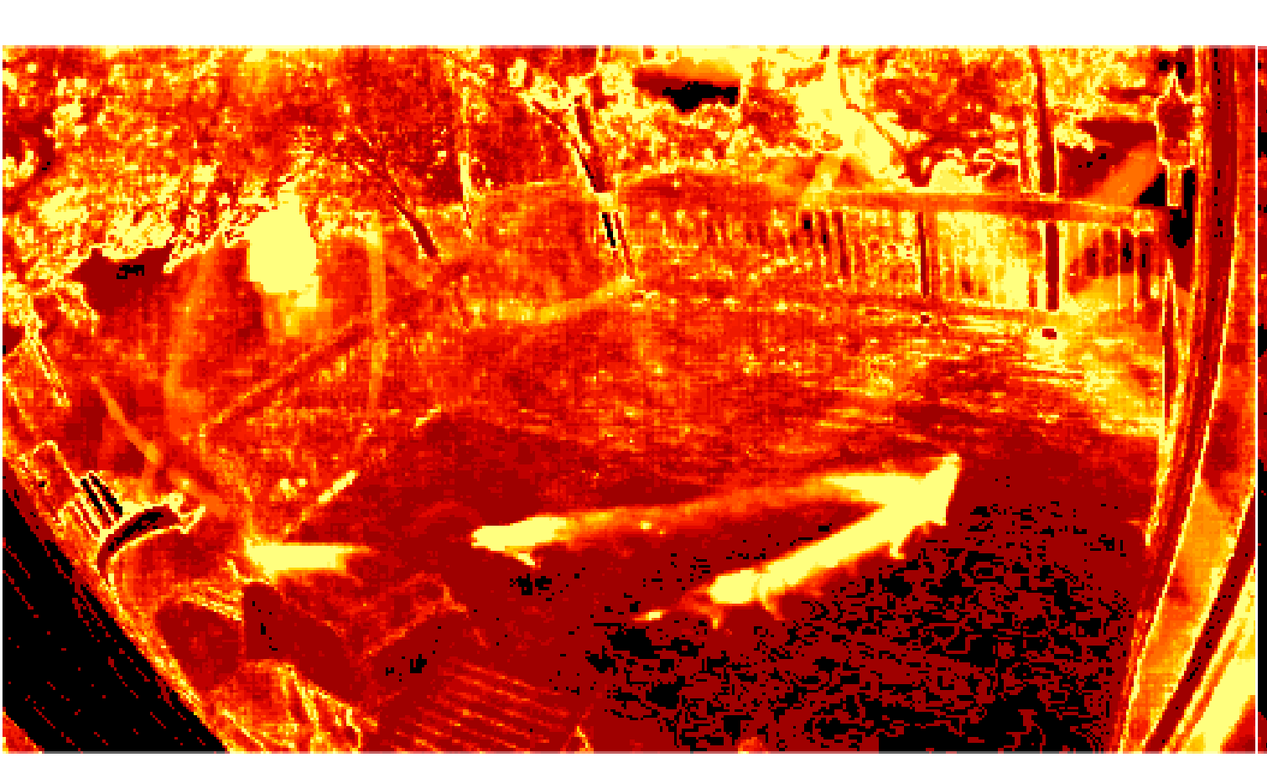


Image Mean

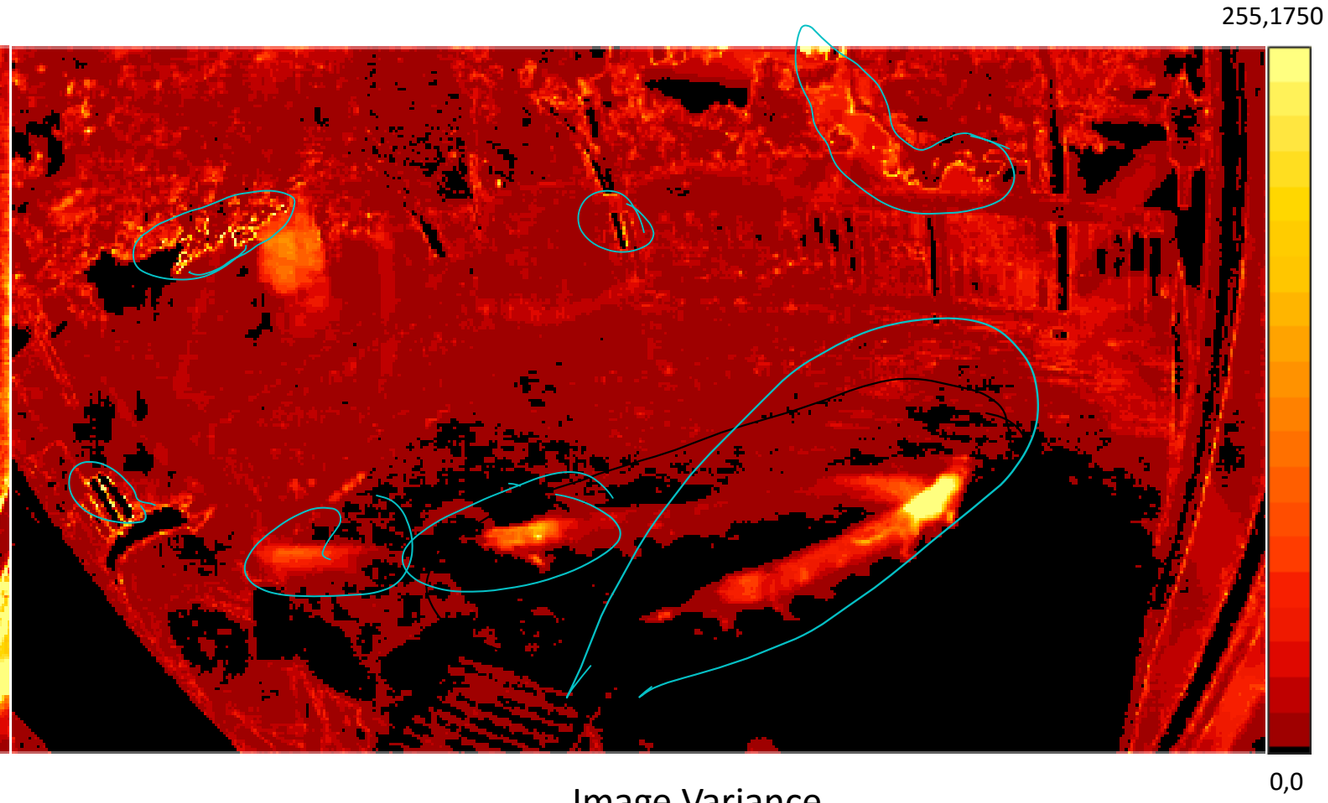
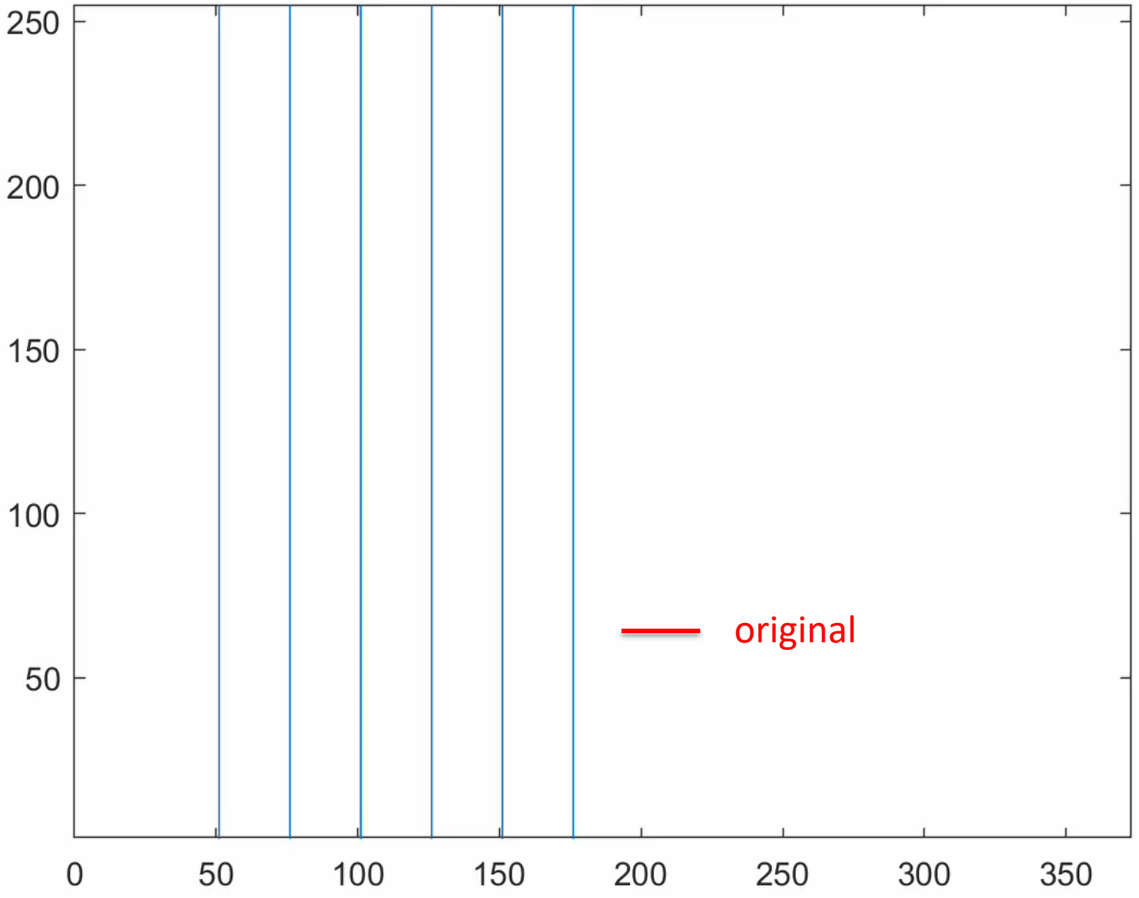
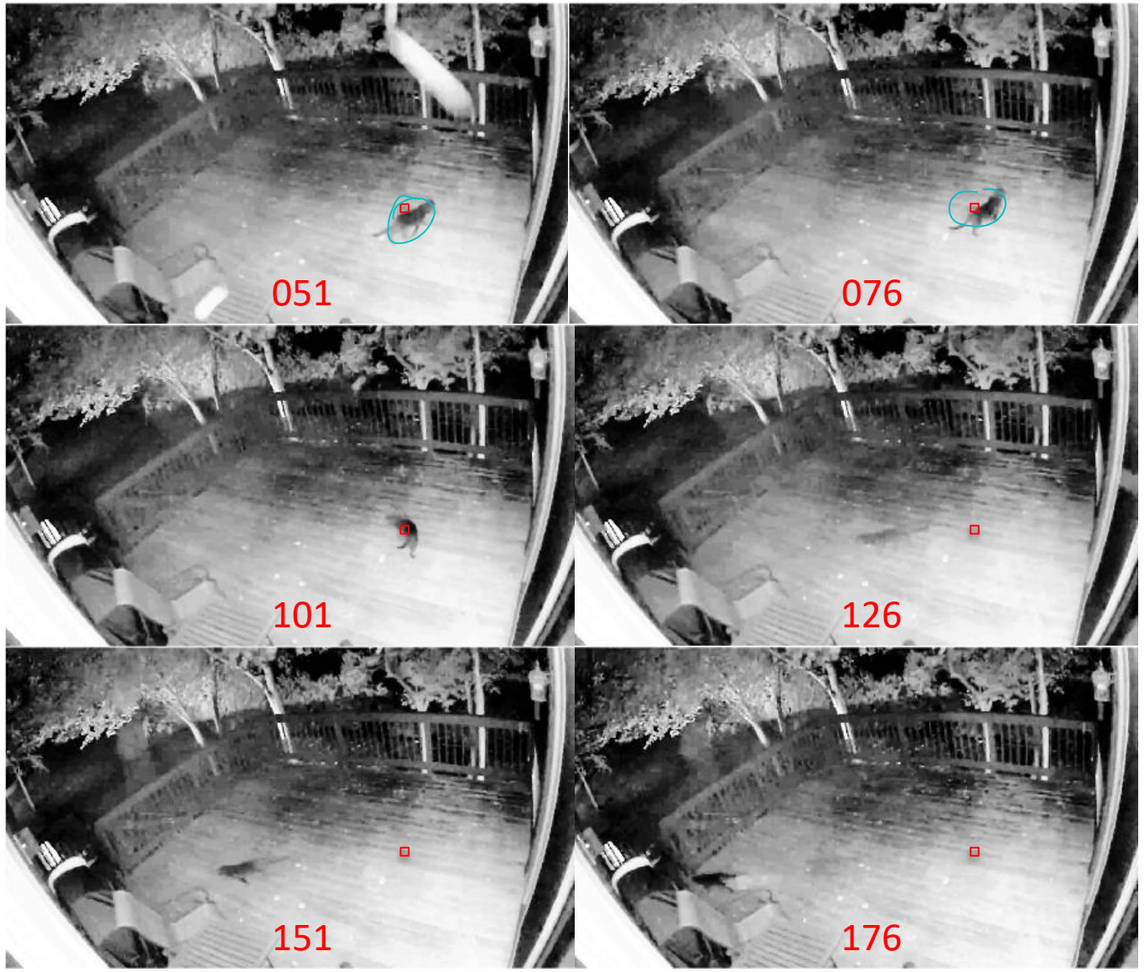


Image Variance

Videos in Matlab

View the time series of a particular pixel.





Videos in Matlab

```
% read mp4 files
load myposnegmapblk.txt
load myposmapblk.txt

% video file name
filename='Cat.mp4';
% get video information
vidObj = VideoReader(filename);
nt = vidObj.NumFrames;
nx = vidObj.Width;
ny = vidObj.Height;
fr = vidObj.FrameRate;
% convert video to grayscale and equalize
videoRGB = read(vidObj,[1 nt]);
I=zeros([ny,nx,nt]);
for t=1:nt
    I(:,:,t)=histeq(rgb2gray(videoRGB(:,:,:,t)));
end
```

```
% average all frames together
vidmean=mean(I,3);
figure;
imagesc(vidmean,[0,255])
colormap(myposmapblk),axis image, axis off
vidvar=var(I,0,3);
figure;
imagesc(vidvar,[0,1750])
colormap(myposmapblk),axis image, axis off
figure;
histogram(vidvar(:))

px=915; py=460;
figure;
plot(squeeze(I(py,px,:)),'r')
xlim([0,nt]),ylim([1,255])
```

Videos in Matlab

```
figure;
vidfile = VideoWriter(['catseriesIpy',num2str(py),'px',num2str(px),'.mp4'],'MPEG-4');
open(vidfile);
for t=1:nt
    plot(squeeze(I(py,px,1:t)),'r','LineWidth',1.1)
    set(gcf,'color','w');
    line([ 51, 51],[0,255])
    line([ 76, 76],[0,255])
    line([101,101],[0,255])
    line([126,126],[0,255])
    line([151,151],[0,255])
    line([176,176],[0,255])
    xlim([0,nt]),ylim([1,255])
    F(t) = getframe(gcf);
    writeVideo(vidfile,F(t));
    pause(.05)
end
close(vidfile)
```

Pixel Temporal Convolution

When we filter measurements as they come in, we use filters that only weight the current time t and past time point measurements. No future.

t-6	t-5	t-4	t-3	t-2	t-1	t
1/7	1/7	1/7	1/7	1/7	1/7	1/7

Seven day average.

1/8	2/8	5/8

Three point weighted averaging.

1/16	3/16	5/16	7/16

Four point weighted averaging.

-1	1

Temporal derivative. Running difference.

Pixel Temporal Convolution

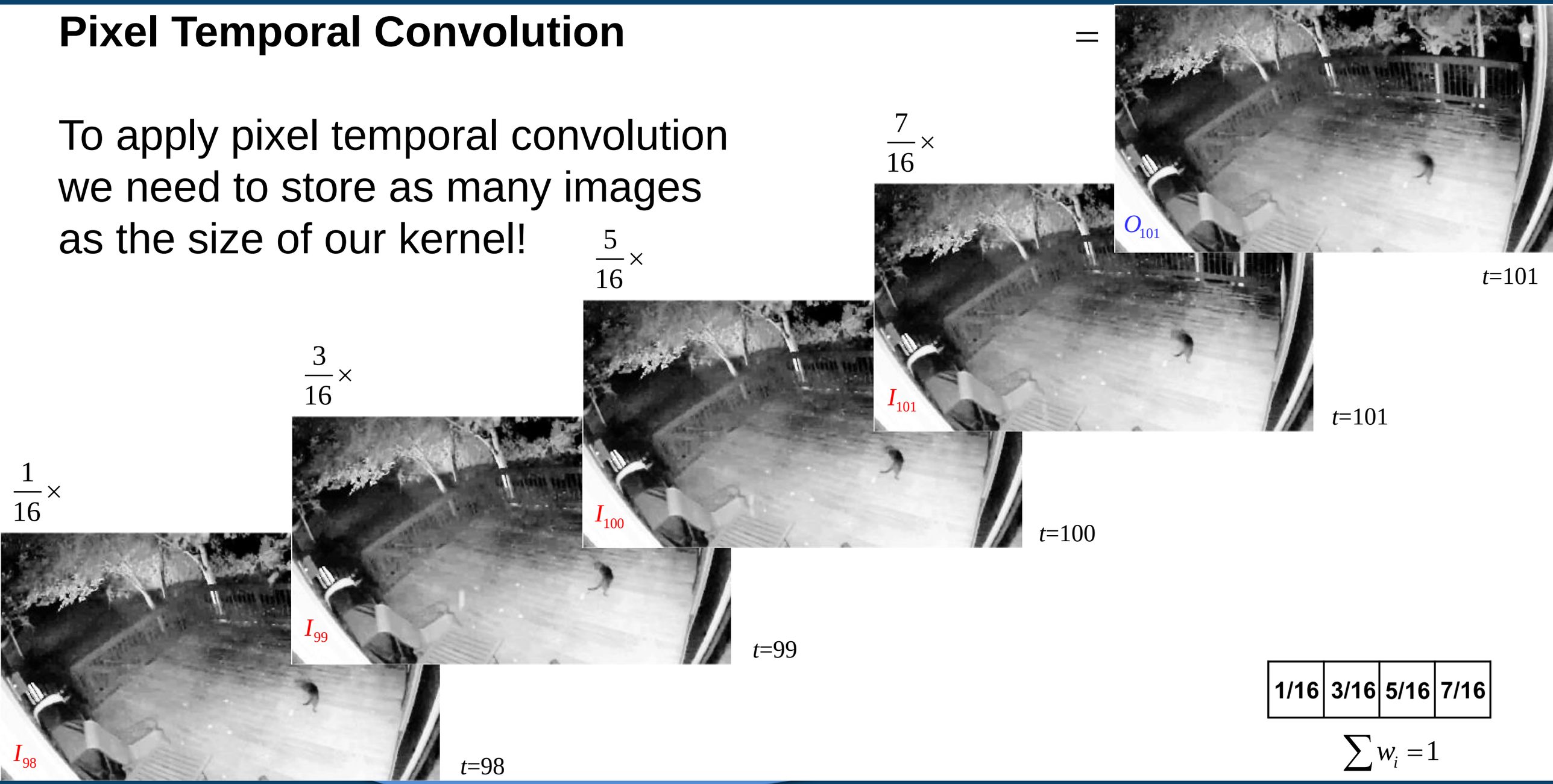
We can apply the kernel individually to the time series of each pixel.

We can smooth the image before or after the temporal smoothing.

We need to be careful not to temporally smooth too much in pixels that change due to moving objects.

Pixel Temporal Convolution

To apply pixel temporal convolution we need to store as many images as the size of our kernel!



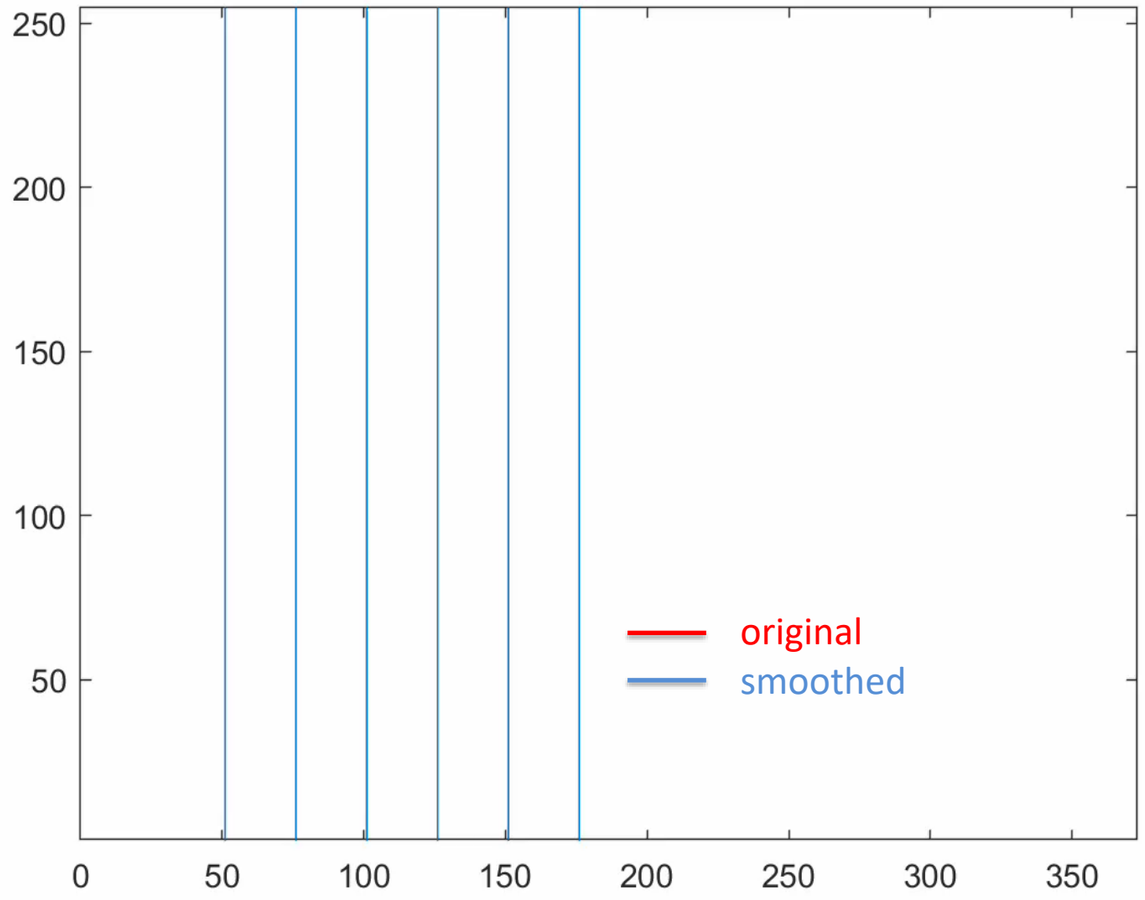
1/16	3/16	5/16	7/16

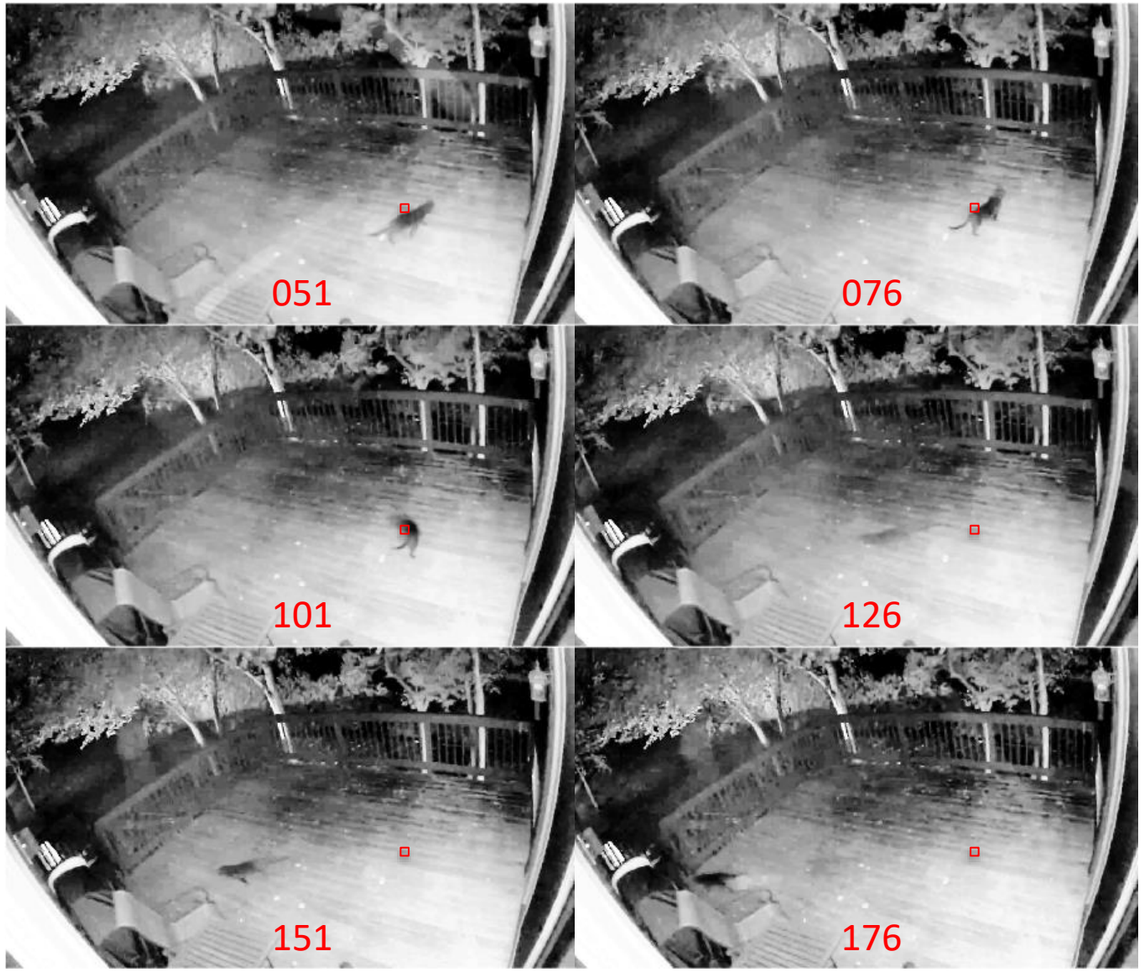
$$\sum w_i = 1$$

Pixel Temporal Convolution

1/16	3/16	5/16	7/16

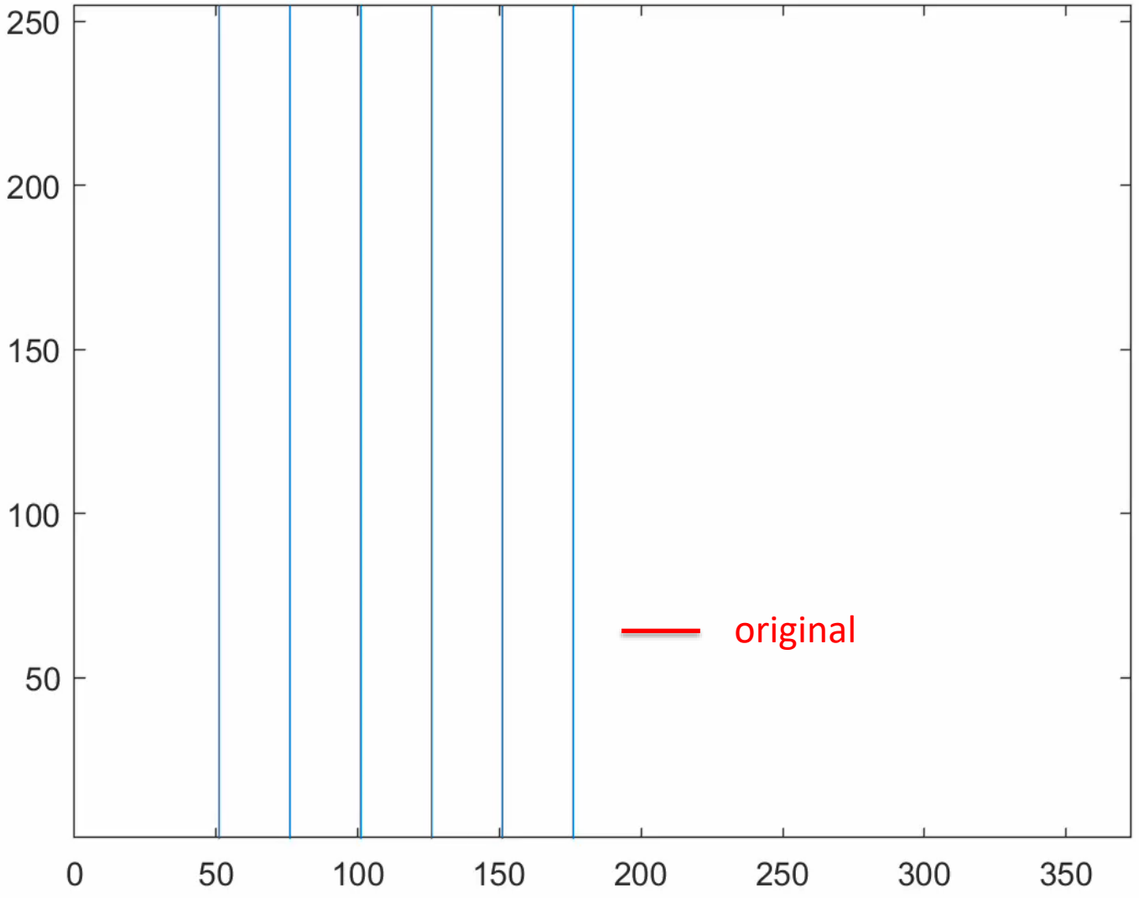
Applied the 4 point weighted temporal averaging.

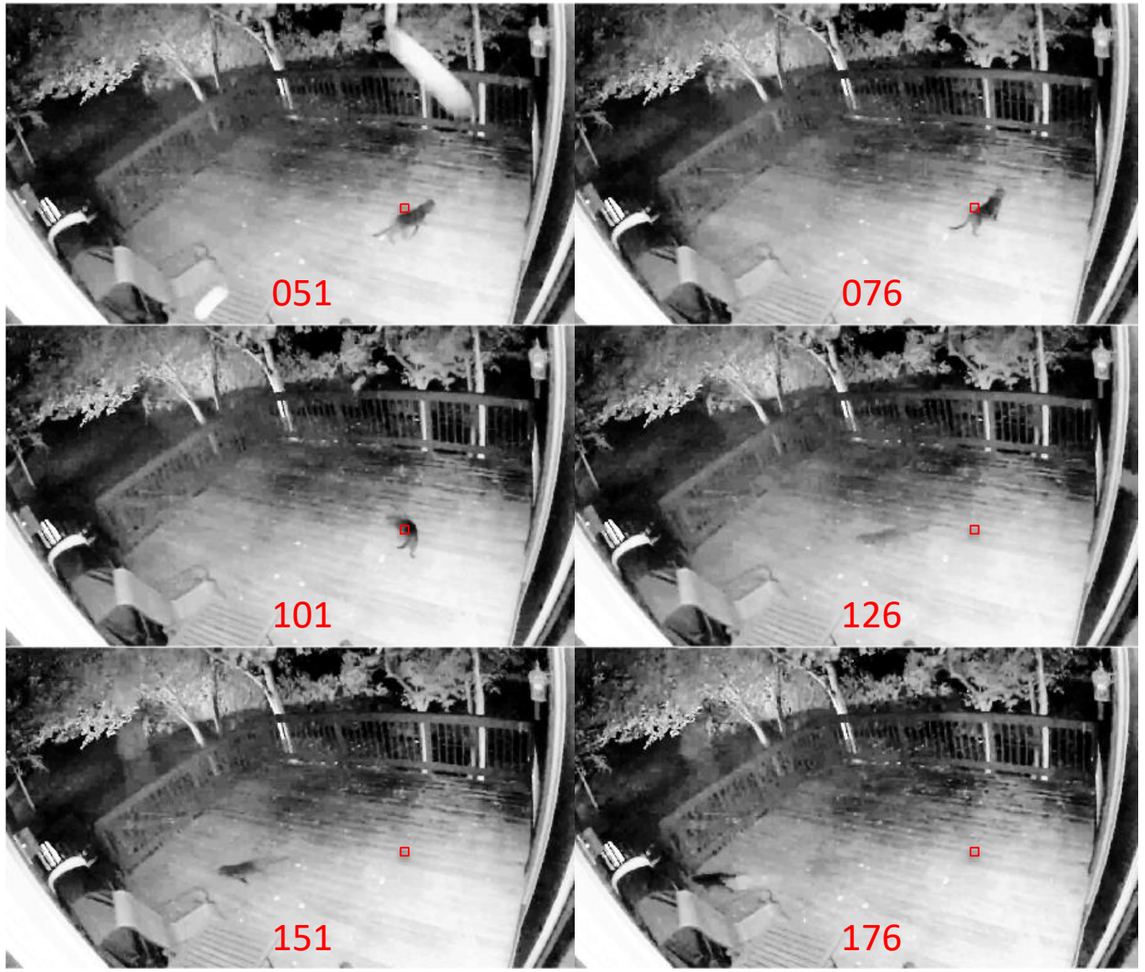




Pixel Temporal Convolution

View the time series of a particular pixel.





Pixel Temporal Convolution

Smoothed Images

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Pixel Temporal Convolution

```
% perform 4 pt temporal averaging
kernel=[1/16,3/16,5/16,7/16];
k=length(kernel);
Ovid4=zeros(ny,nx,nt);
for t=1:nt
    if (t==1)
        Ovid4(:,:,t)=I(:,:,t);
    elseif(t==2)
        ksum=sum([kernel(1,k-1),kernel(1,k)]);
        Ovid4(:,:,t)=(kernel(1,k-1)/ksum)*I(:,:,t-1)+...
            (kernel(1,k)/ksum)*I(:,:,t);
    elseif(t==3)
        ksum=sum([kernel(1,k-2),kernel(1,k-1),kernel(1,k)]);
        Ovid4(:,:,t)=(kernel(1,k-2)/ksum)*I(:,:,t-2)+...
            (kernel(1,k-1)/ksum)*I(:,:,k-1)+(kernel(1,k)/ksum)*I(:,:,t);
    elseif(t>=4)
        Ovid4(:,:,t)=kernel(1,k-3)*I(:,:,t-3)+kernel(1,k-2)*I(:,:,t-2)+...
            kernel(1,k-1)*I(:,:,t-1)+kernel(1,k)*I(:,:,t);
    end
end
```

Pixel Temporal Convolution

```
% view and write grayscale video
figure;
vidfile = VideoWriter('catVid4.avi','Uncompressed AVI');
V.FrameRate = fr;, V.Quality = 100;
set(gca, 'Position', get(gca, 'OuterPosition'));
set(gca,'visible','off')
open(vidfile);
for t=1:nt
    imagesc(Ovid4(:,:,t))
     colormap(gray), axis image, axis off
    set(gca,'position',[0 0 1 1],'units','normalized')
    F(t) = getframe(gcf);
    writeVideo(vidfile,F(t));
end
close(vidfile)
```

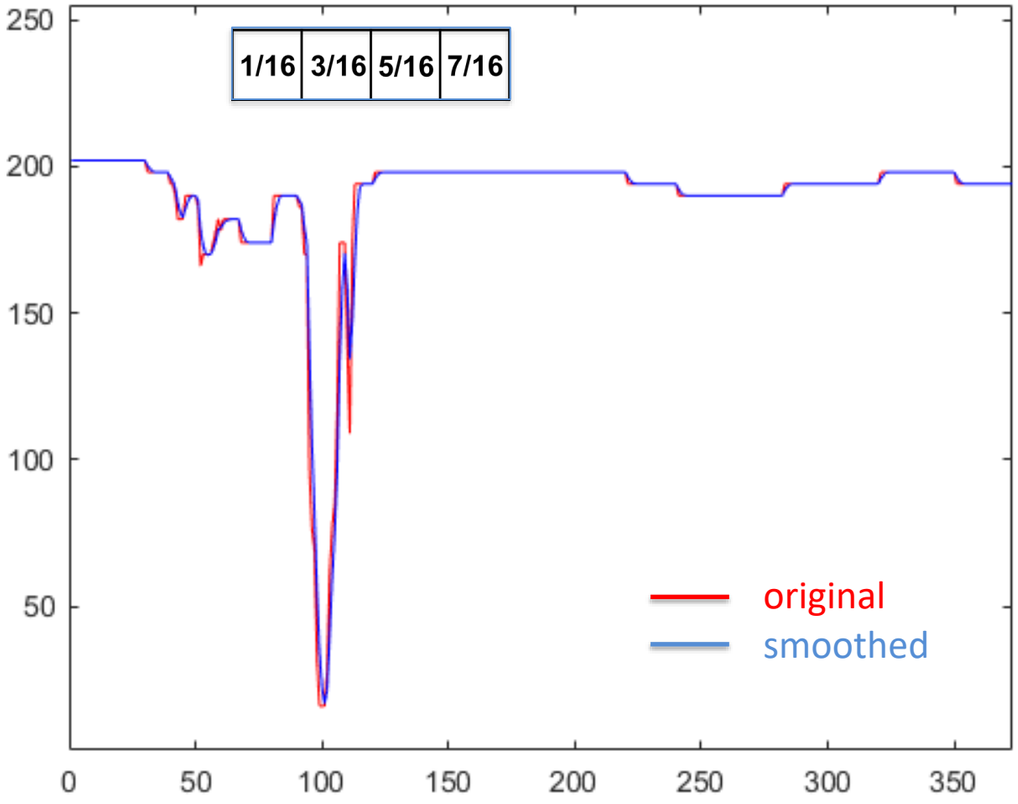
Pixel Temporal Convolution

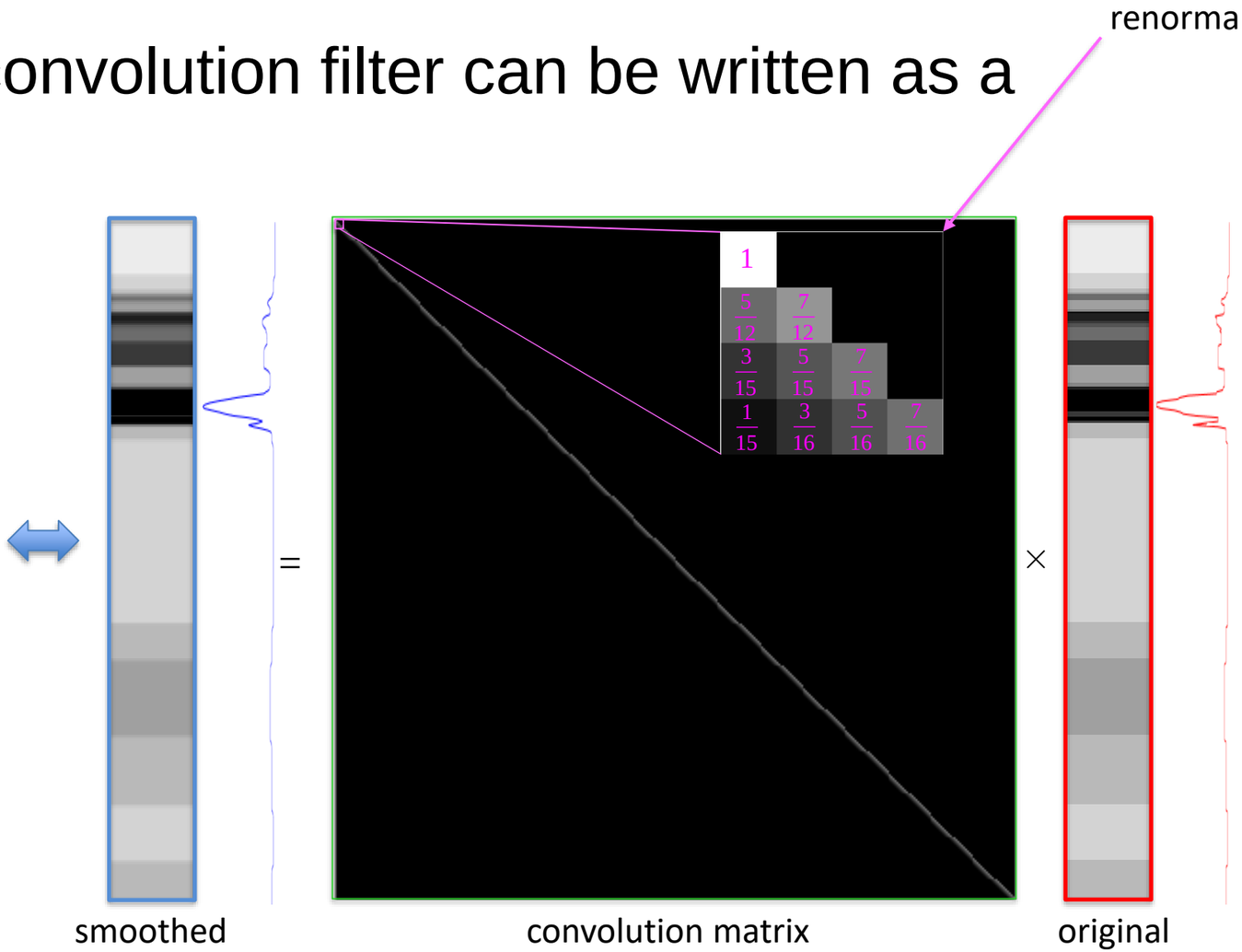
```
kernel=[1/16,3/16,5/16,7/16];
k=length(kernel);
A=zeros(nt,nt);
for t=1:nt
    if (t==1)
        A(1,1)=1;
    elseif (t==2)
        ksum=sum(kernel(1,k-1:k));
        A(t,1:2)=kernel(1,k-1:k)/ksum;
    elseif (t==3)
        ksum=sum(kernel(1,k-2:k));
        A(t,1:3)=kernel(1,k-2:k)/ksum;
    elseif (t>=4)
        A(t,t-k+1:t)=kernel;
    end
end
figure;
imagesc(A,[-1,1])
colormap(myposnegmapblk), axis image, axis off
colormap(myposnegmapblk), axis image, axis off
```

```
AAt=A*A';
D=diag(1./sqrt(diag(AAt)));
corAAt=D*AAt*D;
figure;
imagesc(corAAt,[-1,1])
colormap(myposnegmapblk), axis image, axis off
figure;
imagesc(corAAt(1:7,1:7),[-1,1])
colormap(myposnegmapblk), axis image, axis off
```

Pixel Temporal Convolution

Smoothing a time series with a convolution filter can be written as a matrix multiplication.





Pixel Temporal Convolution

1/16	3/16	5/16	7/16

We can calculate theoretically what the induced covariance is.

Σ/σ^2 = A × A'

covariance

convolution matrix

convolution matrix'

Pixel Temporal Convolution

1/16	3/16	5/16	7/16

correlation induced backward and forward in time

Then we can calculate theoretically what the induced correlation is.

R = A × A'

covariance

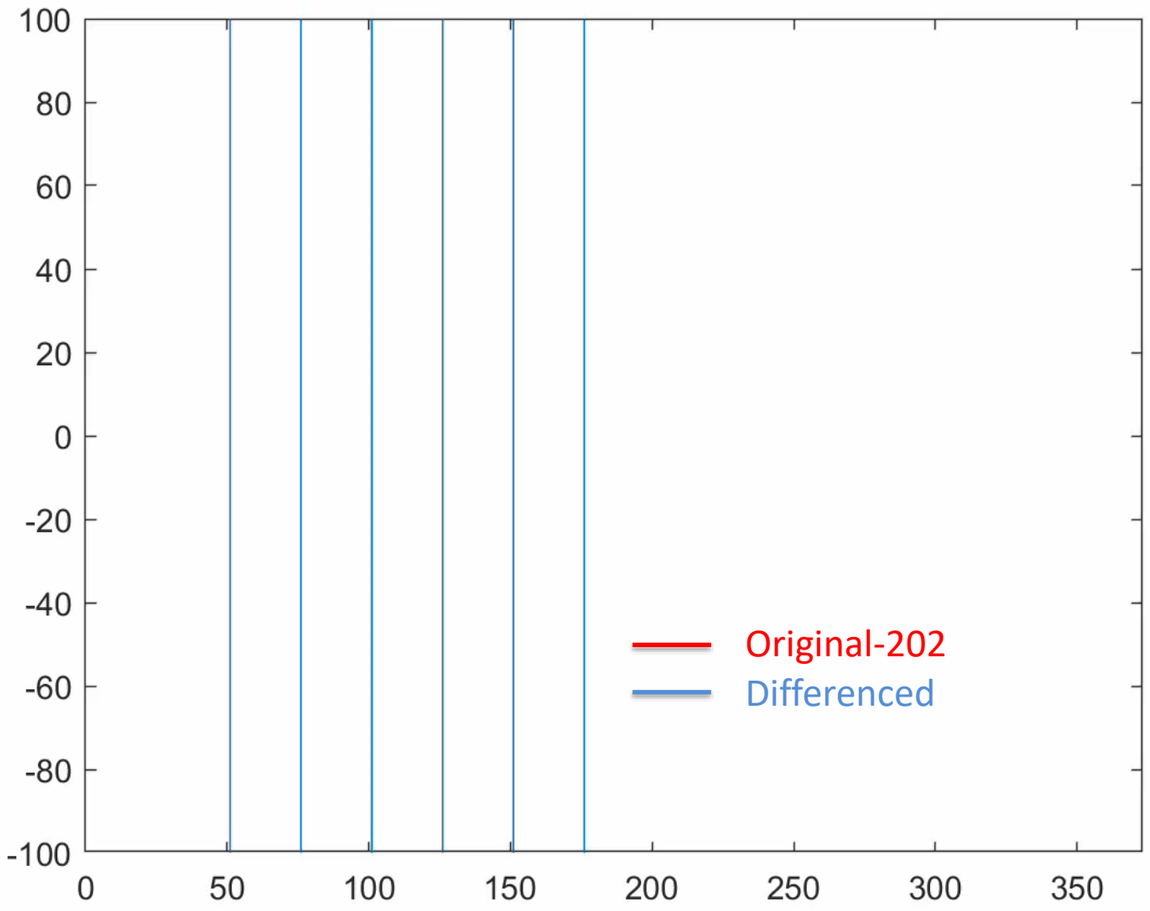
convolution matrix

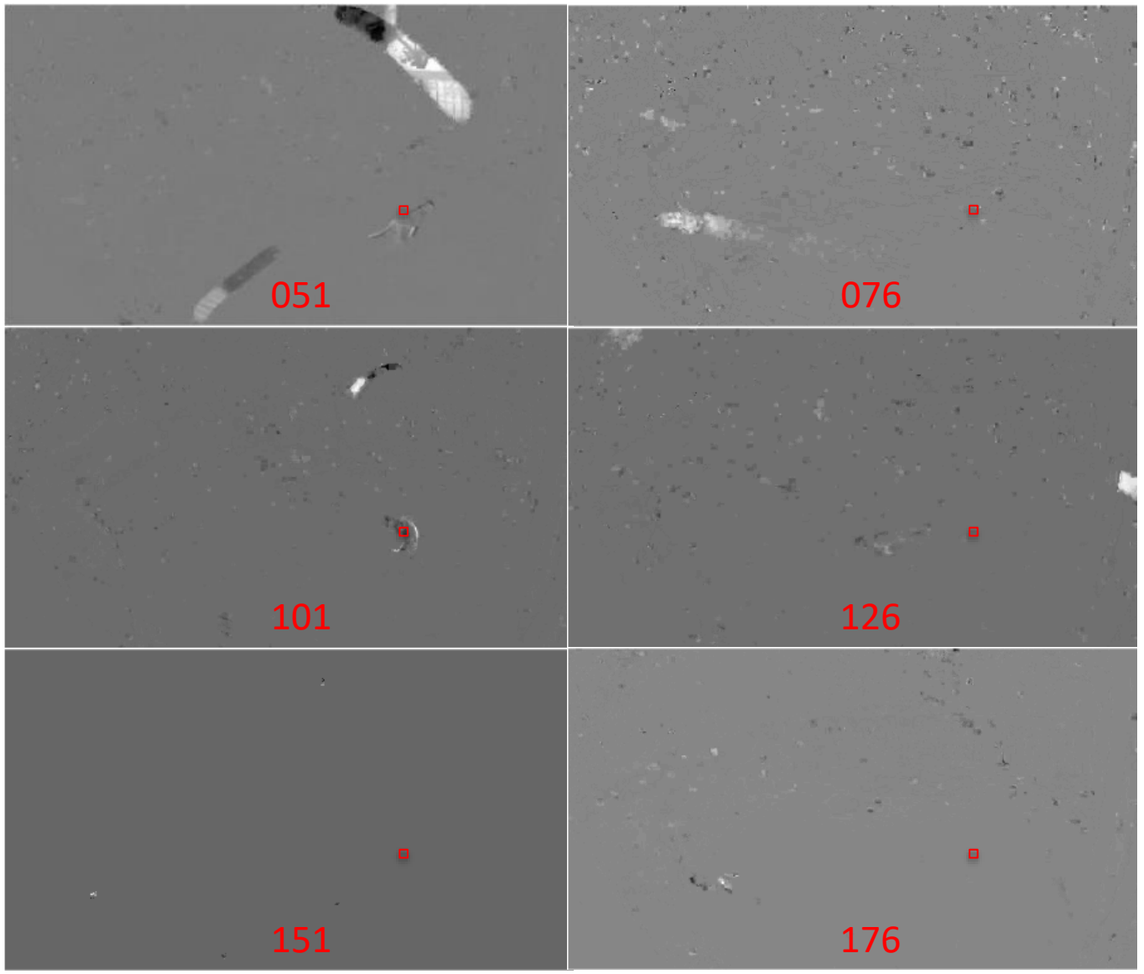
convolution matrix'

Pixel Temporal Convolution

-1	1

Applied the 2 temporal difference.





Pixel Temporal Convolution

Differenced Images

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Pixel Temporal Convolution

Static time window subtraction

Subtracted image of mean of last 100 from all.

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Pixel Temporal Convolution

```
for t=2:nt
    Ovid2(:,:,t)=I(:,:,t)-I(:,:,t-1);
end
lim2=100;
figure;
vidfile=VideoWriter(['catseriesO2py',num2str(py),'px',num2str(px),'.mp4'],'MPEG-4');
open(vidfile);
for t=1:nt
    plot(squeeze(I(py,px,1:t))-202,'r','LineWidth',1.1)
    hold on
    plot(squeeze(Ovid2(py,px,1:t)),'b','LineWidth',1.1)
    set(gcf,'color','w');
    line([ 51, 51],[-lim2,lim2]),line([ 76, 76],[-100,100])
    line([101,101],[-lim2,lim2]),line([126,126],[-100,100])
    line([151,151],[-lim2,lim2]),line([176,176],[-100,100])
    xlim([0,nt]),ylim([-lim2,lim2])
    F(t) = getframe(gcf);
    writeVideo(vidfile,F(t));
    pause(.05)
end
close(vidfile)
```

Pixel Temporal Convolution

```
figure;
vidfile = VideoWriter('catVid2.avi','Uncompressed AVI');
V.FrameRate = fr;
V.Quality = 100;
set(gca, 'Position', get(gca, 'OuterPosition'));
set(gca,'visible','off')
open(vidfile);
for t=1:nt
    imagesc(Ovid2(:,:,t))
    colormap(gray), axis image, axis off
    set(gca,'position',[0 0 1 1],'units','normalized')
    F(t) = getframe(gcf);
    writeVideo(vidfile,F(t));
end
close(vidfile)
```

Pixel Recursive Filtering

We can average all n which requires us to wait for and store all n,
or we can use the kernel weighted moving average of k and only store k.

Alternatively, we can apply a *first order recursive filter* (FORF) and only store 1 additional image besides the current 1.

For the FORF, multiply the first input observation I_1 with weight w to get the first output observation $O_1=wI_1$. The first image will be “short.”

Multiply the input observation I_2 by weight w, add to the output observation O_1 with weight $(1\text{-}w)$ to get the second output observation $O_2=wI_2+(1\text{-}w)O_1$.

Pixel Recursive Filtering

In general, the t^{th} image is $O_t = wI_t + (1-w)O_{t-1}$. We can work out the recursion

Input: I_t, w

Output: O_t

$0 < w \leq 1$

In each pixel at time t:

t	I_t	O_t	effective convolution
0	0	$O_0 = 0$	$= 0$
1	I_1	$O_1 = wI_1 + (1-w)O_0$	$= wI_1$
2	I_2	$O_2 = wI_2 + (1-w)O_1$	$= wI_2 + w(1-w)I_1$
3	I_3	$O_3 = wI_3 + (1-w)O_2$	$= wI_3 + w(1-w)I_2 + w(1-w)(1-w)I_1$
⋮	⋮	⋮	⋮
n	I_n	$O_n = wI_n + (1-w)O_{n-1}$	$= w\sum_{t=1}^{n}(1-w)^{n-t}I_t$

First few short.

General Structure

If we know w, can reverse as $I_t = \frac{1}{w}[O_t + (w-1)O_{t-1}]$

Pixel Recursive Filtering

We can represent this as a matrix process:

$$\begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ \vdots \\ O_n \end{bmatrix} = \begin{bmatrix} w & 0 & 0 & 0 & 0 & \cdots & 0 \\ w(1-w) & w & 0 & 0 & 0 & & 0 \\ w(1-w)^2 & w(1-w) & w & 0 & 0 & & 0 \\ w(1-w)^3 & w(1-w)^2 & w(1-w) & w & 0 & & 0 \\ w(1-w)^4 & w(1-w)^3 & w(1-w)^2 & w(1-w) & w & & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & & & \cdots & w \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ \vdots \\ I_n \end{bmatrix}$$

$$O = AI$$

$$O_0 = 0 \qquad 0 \le w \le 1$$

$$O_t = (1-w)O_{t-1} + wI_t$$

$$O_n = w\sum_{s=0}^{n-1}(1-w)^s I_s$$

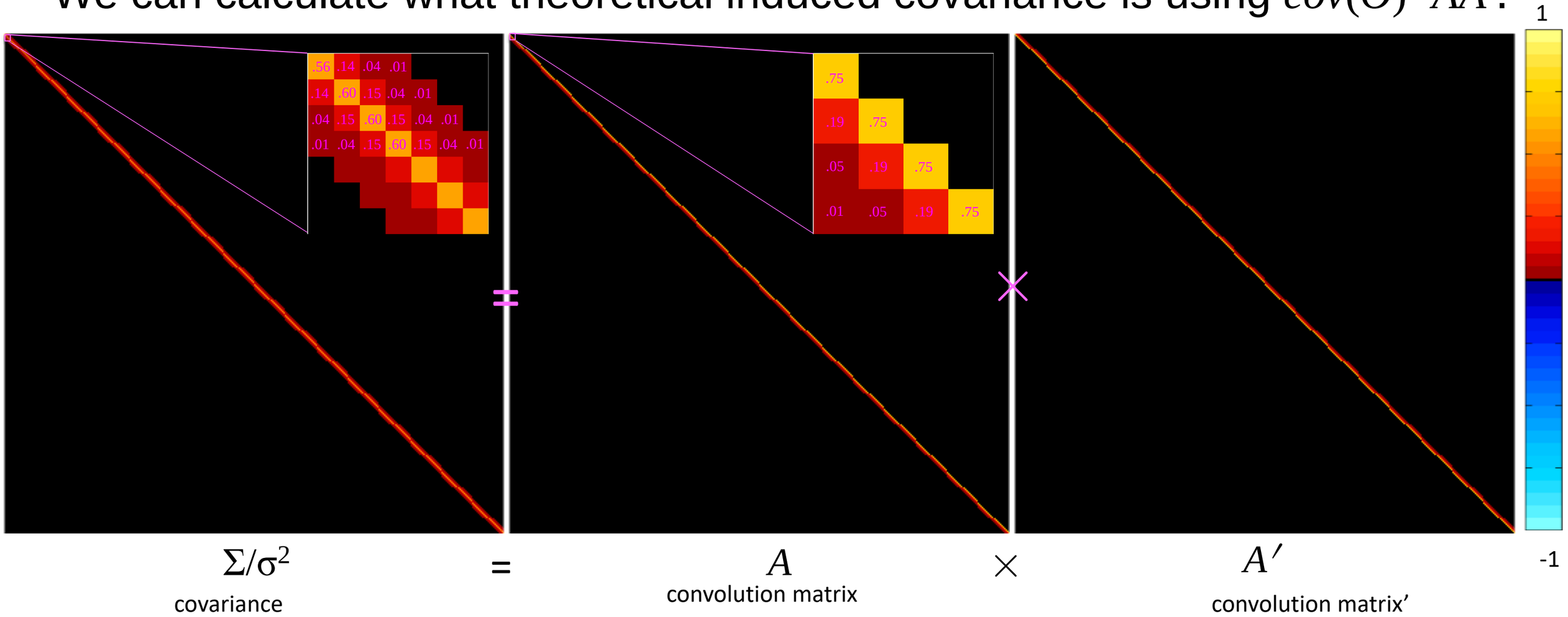
$$O_n = w\sum_{t=1}^{n}(1-w)^{n-t} I_t$$

There is actually a lot of math we can do here.

Pixel Recursive Filtering

$$w = 0.75$$

We can calculate what theoretical induced covariance is using $cov(O)=AA'$.



Σ/σ^2 = A × A'

covariance

convolution matrix

convolution matrix'

Pixel Recursive Filtering

correlation induced backward and forward in time

We can calculate theoretically what the induced correlation is.

R = A $\times$ A'

correlation

convolution matrix

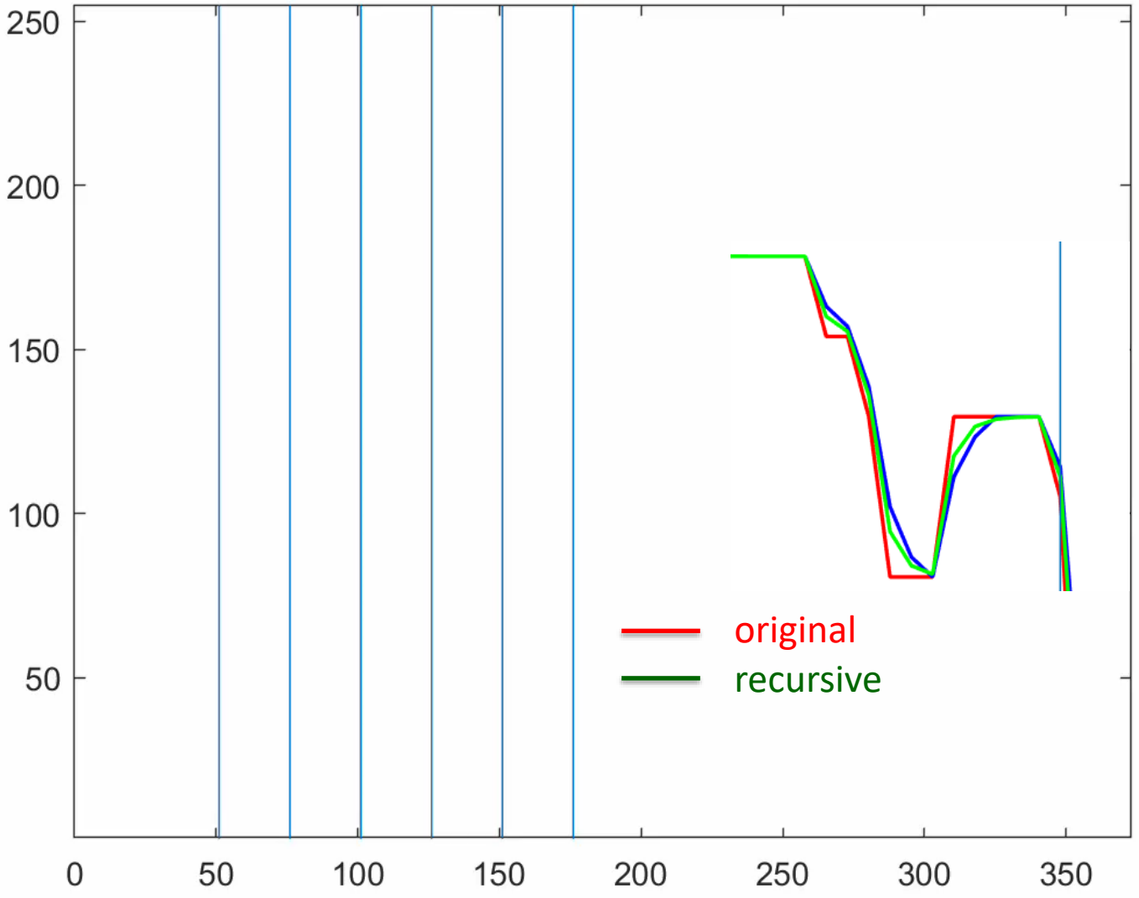
convolution matrix'

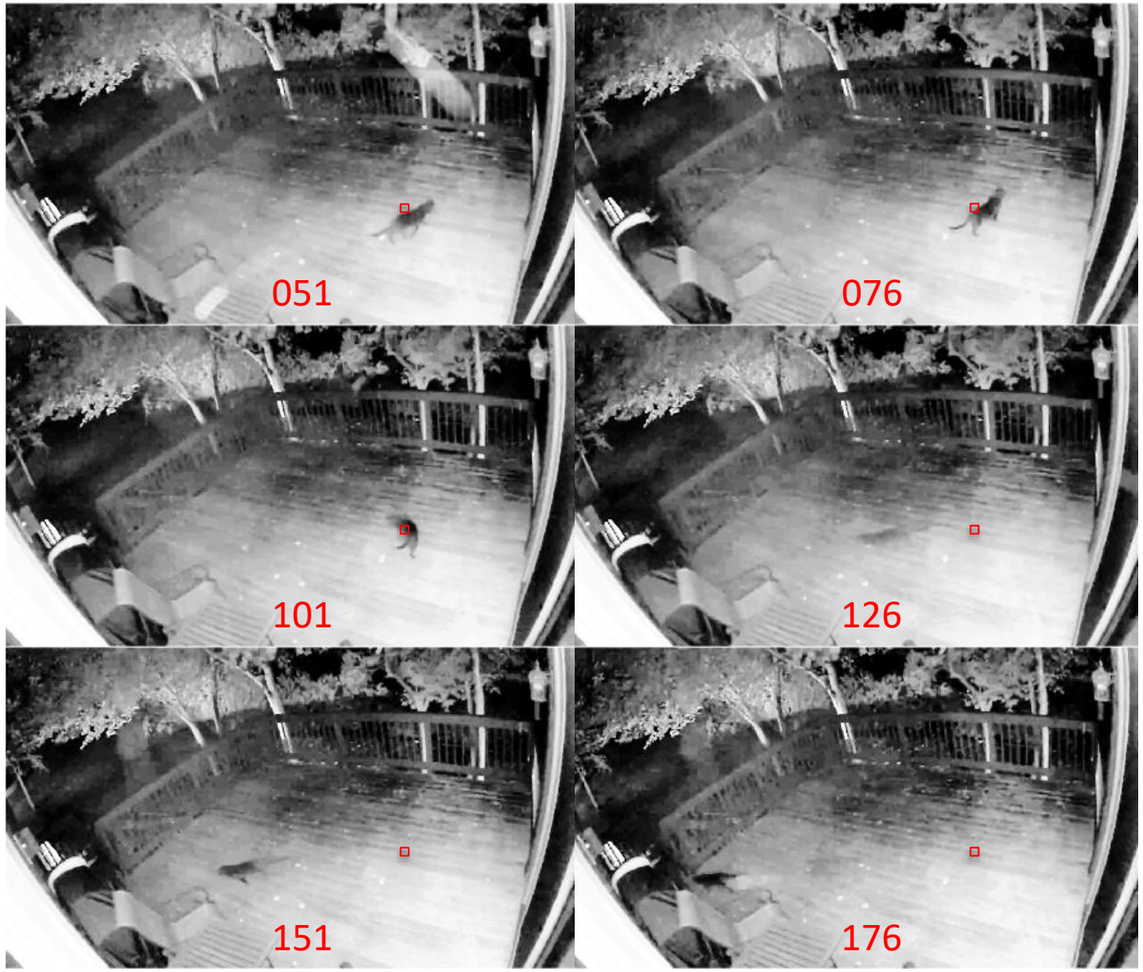
Pixel Recursive Filtering

.0002	.0007	.0029	.0117	.0469	.1875	.7500

$$O_n = w\sum_{t=1}^{n}(1-w)^{n-t} I_t$$

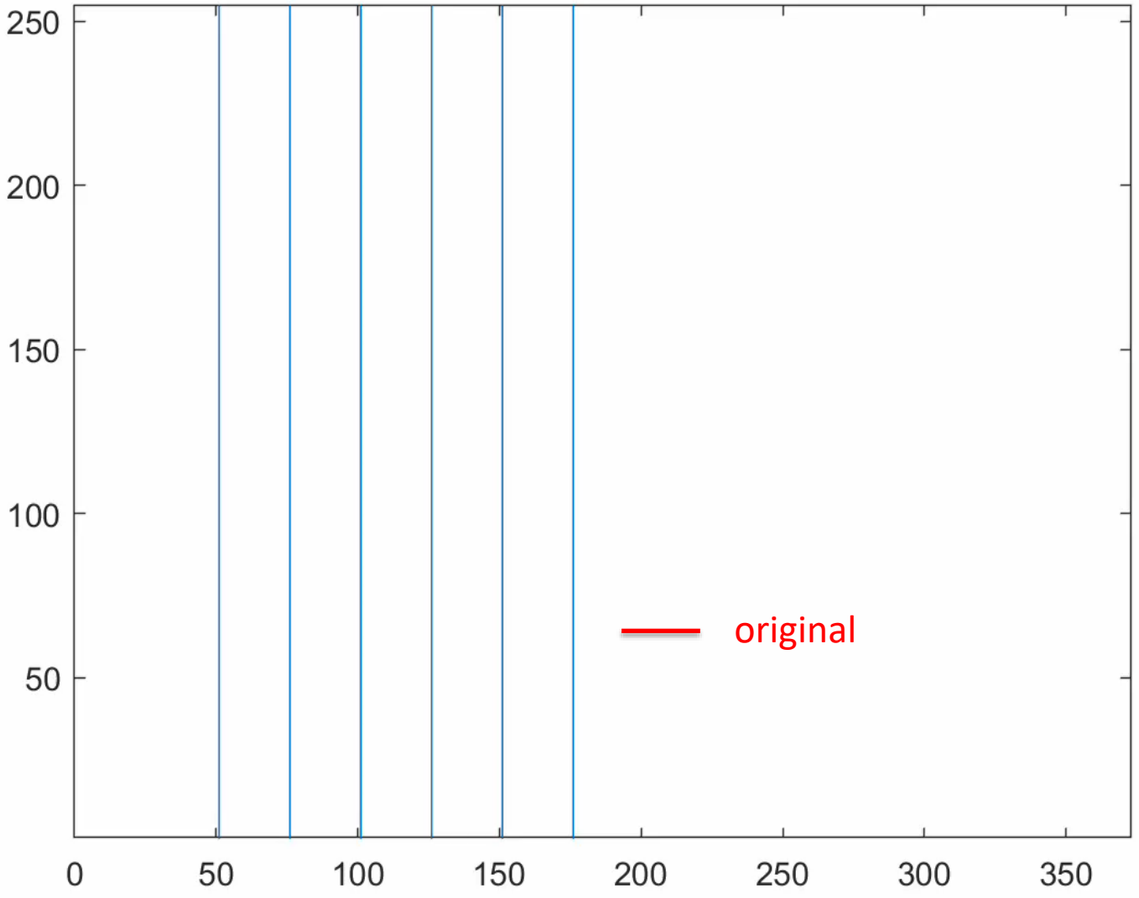
Applied the recursive filter weighted averaging.

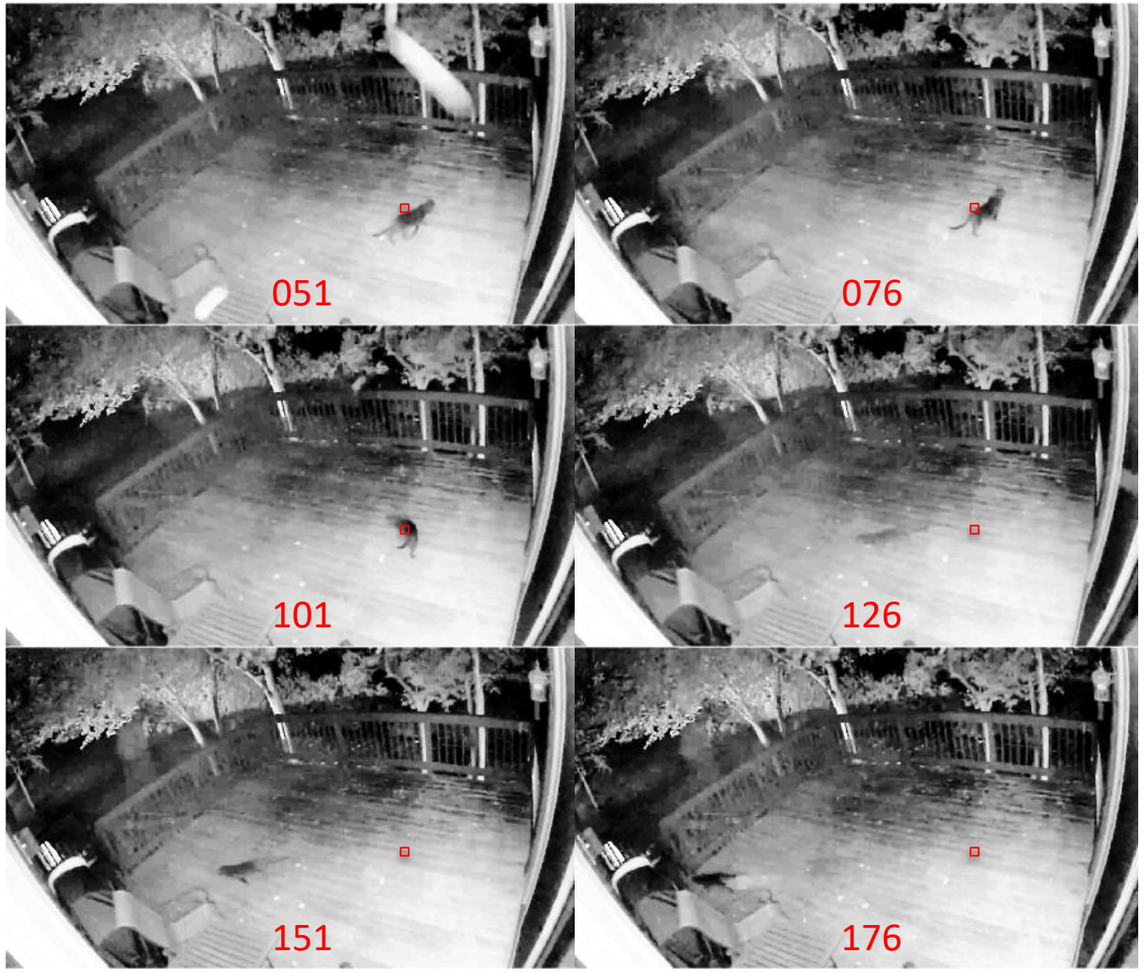




Pixel Recursive Filtering

View the time series of a particular pixel.





Pixel Recursive Filtering

Filtered Images

General Properties:
Name: 'Cat.mp4'
Path: 'C:MATH4931
Duration: 24.8686
CurrentTime: 24.8686
NumFrames: 373

Video Properties:
Width: 1280
Height: 720
FrameRate: 15.0282
BitsPerPixel: 24
VideoFormat: 'RGB24'

Pixel Recursive Filtering

```
% temporal recursive filter
w=3/4;
B=zeros(nt,nt);
B(1,1)=1;
for t=2:nt
    B(t,:)=[(1-w)^(t-1),B(t-1,1:nt-1)];
end
B=w*B;

figure;
imagesc(B,[-1,1])
colormap(myposnegmapblk), axis image, axis off
figure;
imagesc(BBt(1:7,1:7),[-1,1])
E=diag(1./sqrt(diag(BBt)))
corBBt=E*BBt*E;
figure;
imagesc(corBBt,[-1,1])
colormap(myposnegmapblk), axis image, axis off
```

```
OR=zeros(ny,nx,nt);
for j=1:ny
    for i=1:nx
        OR(j,i,:)=B*squeeze(I(j,i,:));
    end
end
```

```
figure;
plot(squeeze(I(py,px,:)),'r','LineWidth',1.1)
hold on
plot(squeeze(Ovid3(py,px,:)),'b','LineWidth',1.1)
plot(squeeze(OR(py,px,:)),'g','LineWidth',1.1)
set(gcf,'color','w');
line([ 51, 51],[0,255]),line([ 76, 76],[0,255])
line([101,101],[0,255]),line([126,126],[0,255])
line([151,151],[0,255]),line([176,176],[0,255])
xlim([0,nt]),ylim([1,255])
```

Discussion

We have seen that we can record a series of image frames as a video.

We can load the video into Matlab and perform processing.

We can perform spatial convolution operations on each image as before.

We can apply temporal convolution on the time series or a recursive filter.

We can apply spatial convolution before, after, or both before and after.

Discussion

Questions?

Homework 7

1. Load your own video into Matlab.
   Cycle through the frames.

2. Apply a spatial convolution kernel to each frame of your video in #1.
   Compare to original. Do not use a built in Matlab convolution function.

3. Apply a temporal convolution kernel to each pixel time series of your video in #1. Compare to original. Do not use a built in Matlab convolution function.

4*.Implement a recursive filter of the time series in your video from #1.
   Compare to original. Do not use a built in Matlab convolution function.

*For students in MSSC 5770.