**EECE5639**

**COMPUTER VISION**

**PROJECT 3**

**- ROHIT RAO NAINENI**

**- THURSTON BREVETT**

**Abstract**:

The goal of this project is to improve the Circulant-Matrix tracker by:

1. Implementing an occlusion detection test;

2. Trying to recover from occlusion.

The occlusion detection is done using the Peak to Sidelobe Ratio" (PSR). The recovery from occlusion is done by predicting the dynamic state using a Hankel matrix.

**Description of Algorithms**:

1. **Occlusion Detection**:
2. The code provided by the authors of the CM tracker does not check for occlusion. We implemented this by including a test that measures the response of the filter against the rest of the search window through the use of the Peak to Sidelobe Ratio" (PSR).
3. The response of the filter is split into the maximum value and the sidelobe consisting of the rest of the pixels in the region.

i.) First we pass the image through the gaussian filter. The filter is composed of a dense guassian kernel which computes the cross correlation of the expected image and the actual image with a fixed bandwidth. Then, the result is cross-correlated with the expected result, to produce a response.

ii.)Secondly, we take the maximum value of the filter response.

% maximum value of filter response

[xx, yy] = find(filterResponse == maxresponse, 1);

iii.) Next we check if the neighborhood is out of bounds.

iv.) Then exclude a small window (i.e. 11 x 11) around the peak, this is done by assigning the small window around the peak as 0,

filterResponse(idx,idy)=0;

1. Then we calculate the PSR, which is defined as:

%mean value and the standard deviation of the sidelobe

m = sum(filterResponse(:))/(numel(filterResponse)- (windowSize^2));

d=sqrt(size(filterResponse(:),1)\*var(filterResponse(:))/(numel(filterResponse)- (windowSize^2)));

psr =(maxresponse - m)/d ;

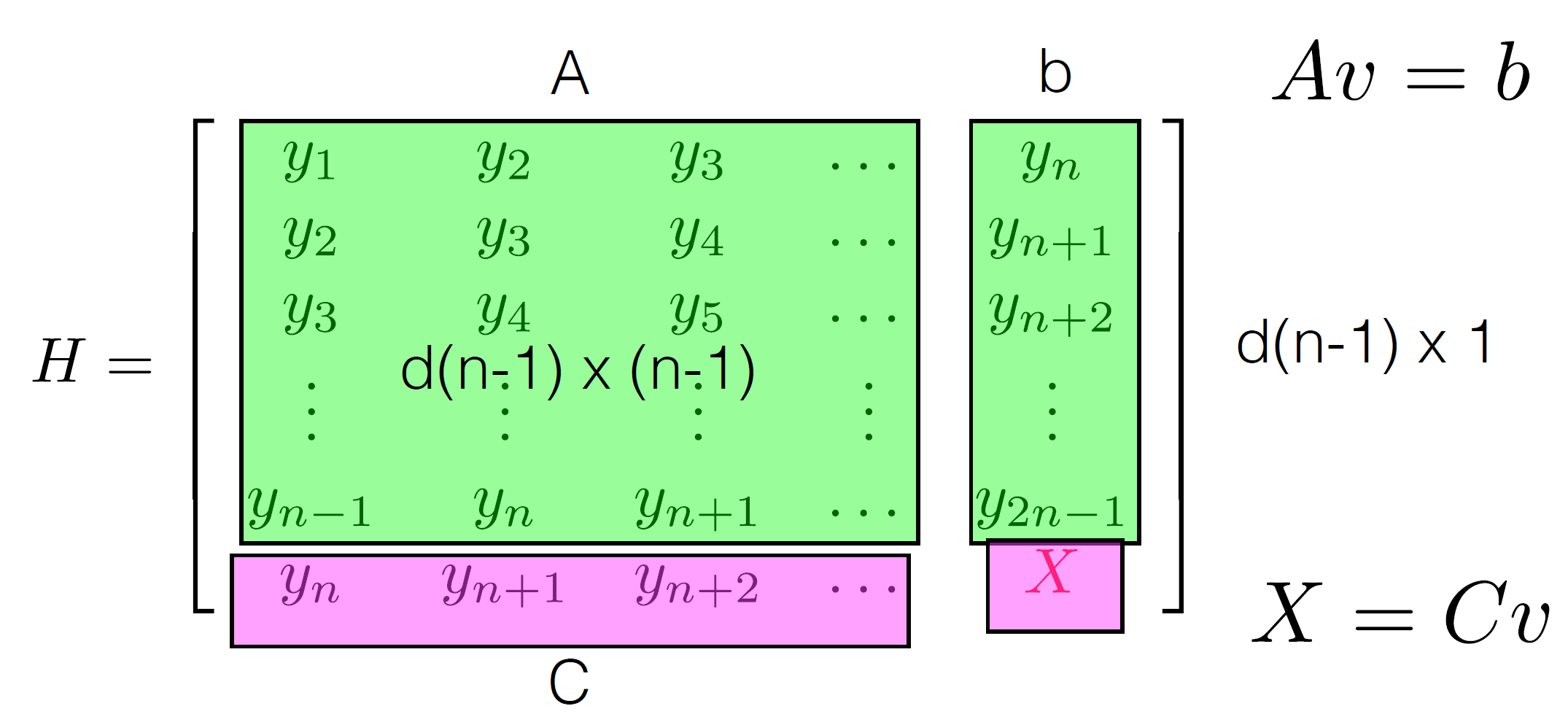
1. If the PSR value of the response is below a threshold, occlusion is detected.
2. **Recovery from Occlusion:**

The prediction of the location of the target based on previous measurements is done using a Hankel matrix.

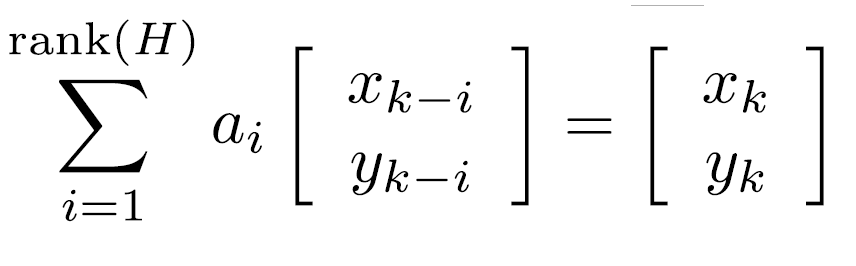
Capturing Dynamics from Experimental Data: The Hankel Matrix

Given a sequence of measurements of d-dimensional vectors:

Its Hankel matrix is defined as:



We use a regressor to model the dynamics:

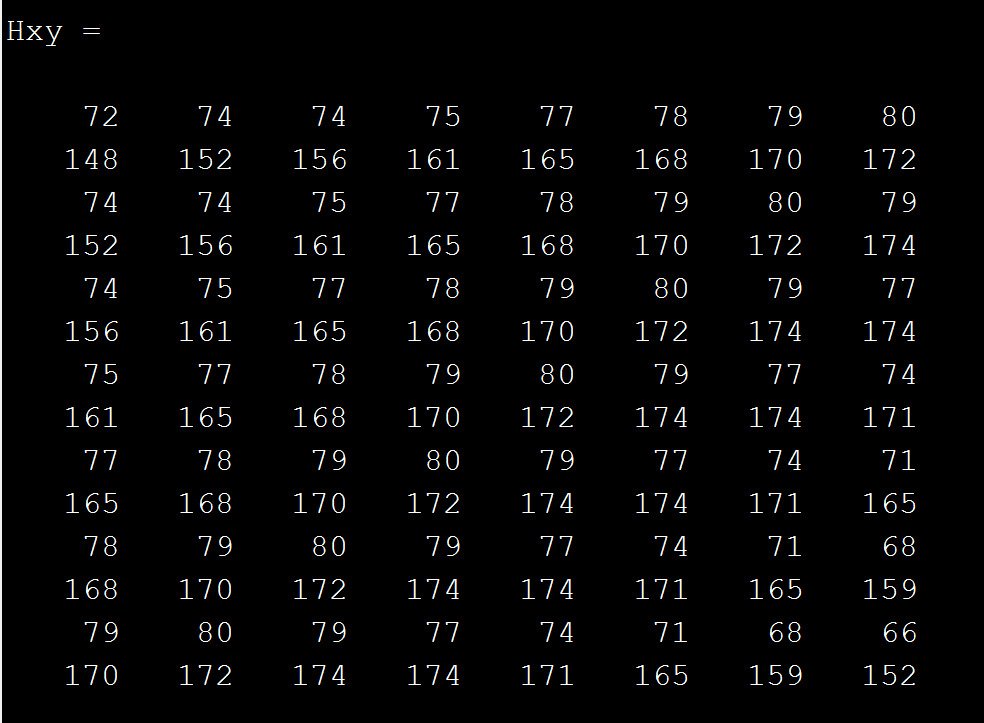


Where the complexity of the dynamics is the rank of the H matrix.

**Algorithm along with the circulant-matrix tracker:**

1. Calculate the PSR of the response. If it’s below a threshold then occlusion is detected.  
   If occlusion is detected, steps we perform steps b) – g).
2. Initially assume a complexity “” and create a vector of states (2D) and populate the vector.
3. After enough frames build the H matrix using the vector of states.

For example:

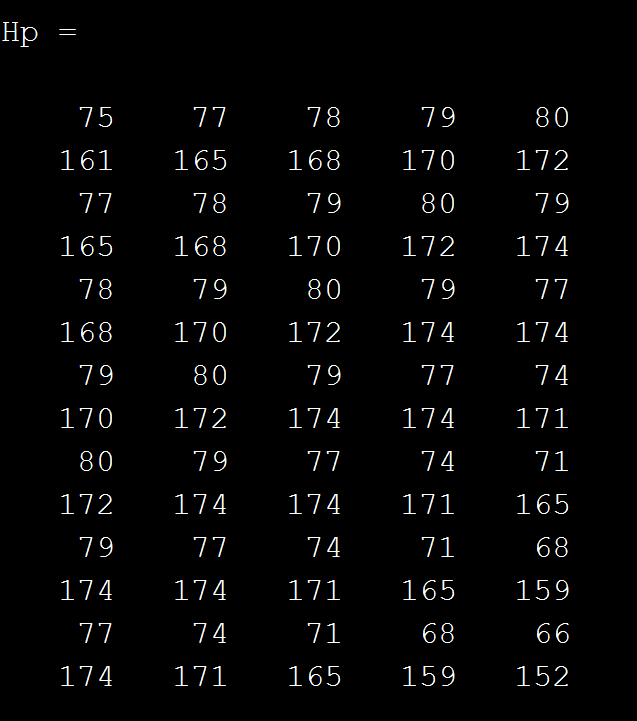


1. We estimate the complexity of the dynamics using the rank of the above Hankel matrix. However the Hankel matrix is full rank due to noise. We try to estimate the rank by using svd, looking at the singular values and deciding on how many are significantly large.

For the above matrix we get the rank as 5.

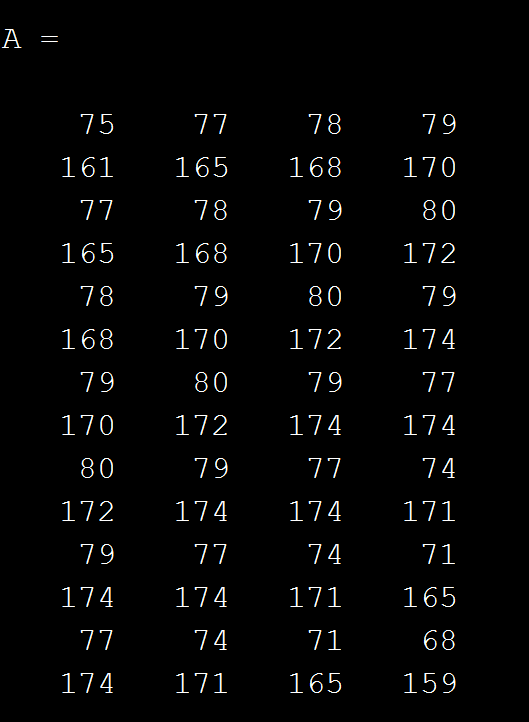
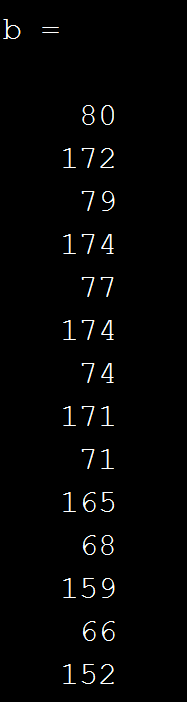
1. Now we rebuild the Hankel matrix to enforce minimum rank (Note: We don’t rebuild the Hankel matrix by recomputing the product since it doesn’t produce a Hankel structure).
2. Instead the regressor is found in a "least square sense". We keep the number of columns of our recomputed Hankel matrix to be the rank found above and rebuild the block Hankel matrix.

The above matrix becomes:

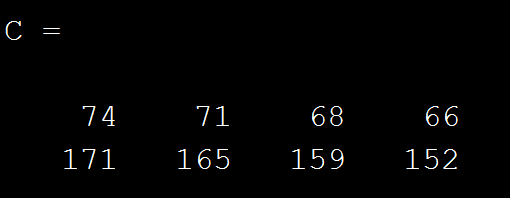


Where is the Hankel matrix we enforced rank to be 5.

1. Now we calculate the A matrix which is the Hankel matrix excluding the last column and the b matrix is the last column as shown:

The C matrix is the last row of the hankel matrix except where the next position is to be calculated:



Now we calculate the matrix using :

% compute dynamic linear regressor coefficients, v

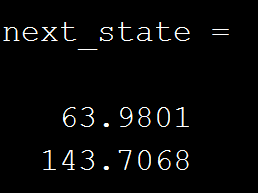
v = A \ b;

As shown in Hankel matrix figure the X which is the prediction of the location of the target based on previous measurements is a linear combination of the previous ones.

% predict next state (location)

next\_state = C\*v;

Which is found to be:



1. If occlusion was not detected, the central position of the object in the next frame is expected to be near enough to the prior position, so that the object falls within the same window, and the expected next center position is where the target was in the prior frame.  
   Otherwise, if the occlusion was detected, the central position of the object in the next frame is expected to be at the center position predicted by the Hankel matrix in part g).
2. The new sample is pulled from the portion of the image with a center at the predicted object center and the same size window. The sample is passed through the filter to get a response for the new frame

## Experiments

To test our occlusion detection algorithm, we used multiple thresholds to evaluate when occlusion is detect. To test our occlusion recovery algorithm, we used the show\_precisions.m function to compare how well our tracking method fared compared to without.

## Values of Parameters used

n = 8; memory of the system  
d = 2; dimensions of the state vector  
psr\_threshold = [8]; threshold for determining oclusion  
svd\_threshold = 1.5; threshold for determining rank  
padding = 1; extra area surrounding the target  
output\_sigma\_factor = 1/16; spatial bandwidth (proportional to target)  
sigma = 0.2; gaussian kernel bandwidth  
lambda = 1e-2; regularization  
interp\_factor = [0.075, 0.200]; linear interpolation factor for adaptation

## Observations & Conclusions

The circulant matrix was able to track

We found that our dynamics prediction was able to predict future positions from a set of past positions, but not always correctly in the setting of the tiger videos. This is because in this video, there is a large frame-to-frame difference between what the tiger looked like in the previous frame, and what it looked like in the next frame, causing our filter to fail. At times this occurred due to partial occlusion, but at other times this occurred without any occlusion. This shows that subsequent estimates of the appearance of the target are not always accurate, causing our filter to produce a low response and detect an occlusion when the target changes appearance.

In addition, although the interpolation of the expected target image modifies the expectation of the next image, interpolating too much will cause the kernel to train on the background of the image, while not enough interpolation produces a low response when partially occluded. We were not able to find a good interpolation value that mediated both effects for the tiger video. This caused our filter to have a strong response centered on the background of the image as the object was occluded.

In some cases, the Hankel matrix predicted a correct next position based on the past states, but the predicted position was not the true location of the target in the next frame. This often happened because the past positions of the target were predicted incorrectly due to the failure of the Gaussian kernel to identify the object and reliable locate its center. In some cases, the object changed appearance rapidly, causing the center of the response peak to shift rapidly, and creating a chaotic motion. In other cases, the peak-to-sidelobe ratio of the response was just as large while the object was occluded as it was when the object was not occluded. In these cases, the Hankel matrix predicted the correct neighborhood, while the filter indicated a peak response in the wrong location.

## Appendices