Computer Vision Project 4

T. A. Rupprecht & Can Uner

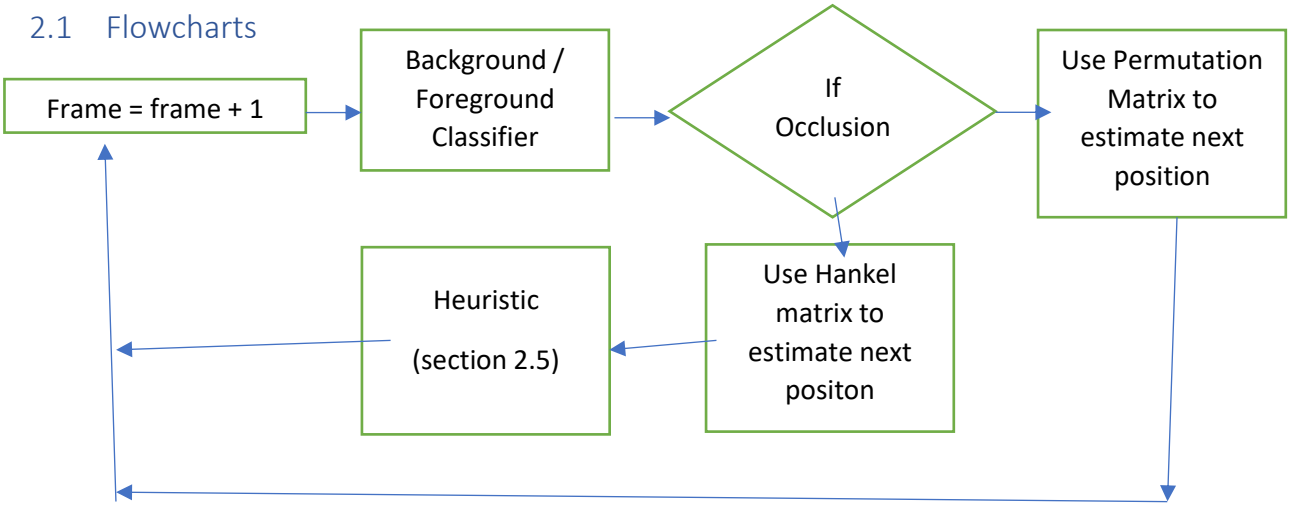
December 09, 2018

1 Abstract

The purpose of this project is to understand the advantages and limitations of the Circulant Tracking algorithm discussed in class. The underlying algorithm is introduced in the paper, “Exploiting the Circulant Structure of Tracking-by-detection with Kernels.” The project asks us to take the algorithm a step further and attempt to detect when the target object being tracked becomes occluded. The project asks when occlusion is detected to use the Hankel matrix methodology discussed in class to estimate the new positions of the occluded target. We decided to take this a step further and added a heuristic. This heuristic adds the rule that the kernel response measured in the estimated position needs to be higher than the last non-occluded response that was measured in order for the Hankel estimated position to be accepted. Taken altogether the algorithm implemented is called Circulant Tracker Plus. The results of our experiments show that adapting to occlusion remains challenging and an active area of research. Using the Hankel matrix works in ideal situations but does little to overcome the problems present when tracked target objects become obscured or occluded from the camera.

2 Algorithm Descriptions

2.1 Flowcharts



2.2 CT Tracker

The original CT tracker code for MATLAB is available, published by the authors of the paper that introduces the circulant tracker. The circulant tracker relies on exploiting the natural circulant properties that exist within the tracking by detection problem. A classifier that labels background and foreground pixels can track the object through time by estimating the next position by leveraging a permutation matrix. This results in a robust and easy to implement real time tracking algorithm. However it is still vulnerable to occlusion and drastic changes in appearance.

2.3 Occlusion

The code provided by the authors of the CM tracker does not check for occlusion. We do this by including a test that measures the response of the filter against the rest of the search window using the "Peak to Sidelobe Ratio" (PSR). To do this, we split the response of the filter into the maximum value and the "sidelobe" consisting of the rest of the pixels in the region, excluding a small window (i.e. 11 × 11) around the peak. Then the PSR is defined as:

$$(gmax - \mu) / \sigma$$

where gmax is the value of the peak, and μ and σ are the mean value and the standard deviation of the sidelobe, respectively. A low PSR indicates a poor match and possible occlusion.

2.4 Hankel

When occlusion is detected, the tracker uses the locations of the target in the past to predict where the target is now (even if it is behind some occluding object) and predicts where it should be in the next frame. In this way, rather than giving up, the tracker attempts to find the target in the next frame at this predicted location. This prediction of the location of the target based on previous measurements can be done using the Hankel matrix of the target locations.

2.5 Heuristic

A heuristic was added that will make use of the position calculated from section 2.4 but will only use that calculated position in situations where the response for the template at that estimated position is higher than the response of the template at the position in the last non-occluded frame.

3 Experiments

For each experiment we compared the tracking precision of the four different algorithms in 4 separate videos. The algorithms and ground truth are visualized in each frame as the experiment runs. The four videos we experimented with are:

1. David.mp4
2. Dollar.mp4
3. Tiger1.mp4
4. Tiger2.mp4

While visualizing the experiments, each frame has potentially three different bounding boxes. CT and CT+ are often overlapping, except in cases where occlusion is detected. CT+ uses the heuristic defined in section 2.5.

1. Ground Truth – Green
2. Original Circulant Tracker – Red
3. Circulant Tracker Plus – Blue

In the results section (section 4), you will see a comparison between the precision of the original Circulant tracker, the circulant tracker we implemented with the heuristic, and a circulant tracker that we were asked to implement without the heuristic from section 2.5 added. We compare these three algorithms to the MIL TR001 tracking algorithm.

3.1 Experiment 1

Experiment 1 uses “David.mp4” and shows a graduate student walking behind the path of a cameraperson. This creates the effect of having David in the center of the camera scene as the camera travels. This centers David’s face in the camera scene and in effect makes it so only the background of the scene changes. Occasionally David turns his head or fiddles with his glasses. David also walks under lights and through shadows that also alter the appearance of his head.

Figure 1 below shows the video as the experiment progresses. The green bounding box is for the ground truth – the actual location of David’s head, the object being tracked. There is a red bounding box for the original circulant tracker, and a blue bounding box often overlaps this red bounding box, and that belongs to the circulant tracker we implemented for our project.

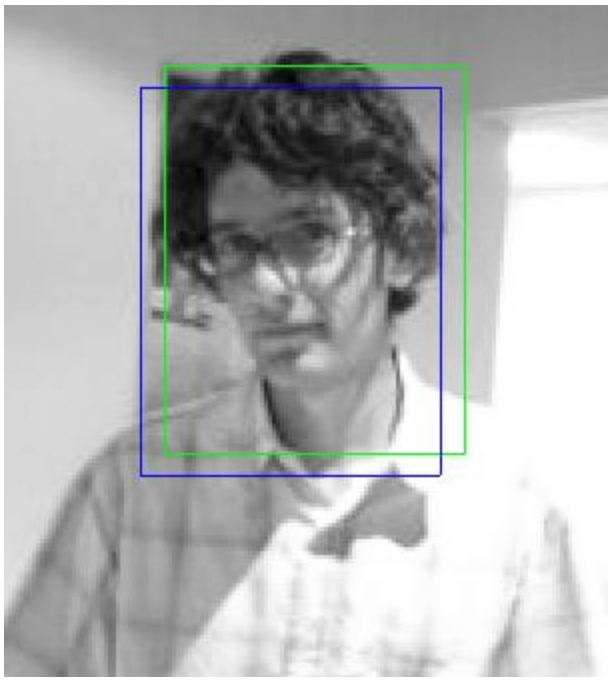


figure 1

3.2 Experiment 2

Dollar.mp4 shows a stack of identical papers. The circulant tracker is to track the top most image in the stack. The stack is divided in two, and the top half is folded in a way to obscure the original image. The purpose of this experiment is to show how the circulant tracker adapts to a change in appearance. It also shows that when a newer object appears that looks like the former template, the tracker does not lose track of the original object with a changed appearance.

Figure 2 shows one frame from this video experiment.

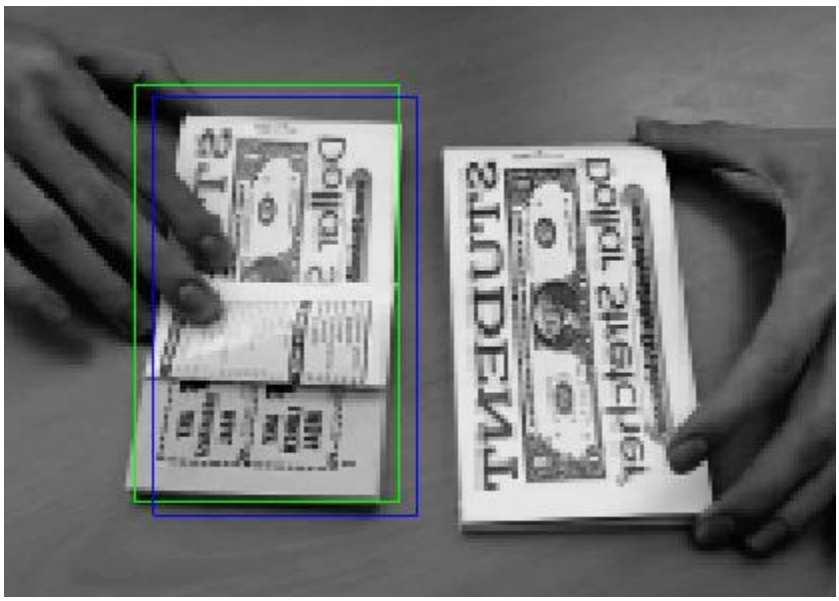


figure 2

3.3 Experiment 3

The 3rd experiment is the first experiment that should challenge the algorithms. There is a toy tiger filmed that travels under a bright light, and behind a plant occluding the tracked object. Occasionally this object moves fast enough that it blurs in the image. It also has the option of having its mouth open to shine a bright light at the camera.

3.4 Experiment 4

The 4th experiment is the second experiment that should challenge the algorithms and is nearly identical to the third experiment video. There is a toy tiger filmed that travels under a bright light, and behind a plant occluded the object. Occasionally this object moves fast enough that it blurs in the image. It also has the option of having its mouth open to shine a bright light at the camera.

4 Results & Observations

In the results section, you will see the comparison between the original Circulant tracker (red), a circulant tracker that makes use of the heuristic added in section 2.4 (blue), and the circulant tracker we were asked to implement called CT+ w/o Heuristic (green). We compare these three algorithms to the MIL TR001 tracking algorithm (black).

4.1 Experiment 1

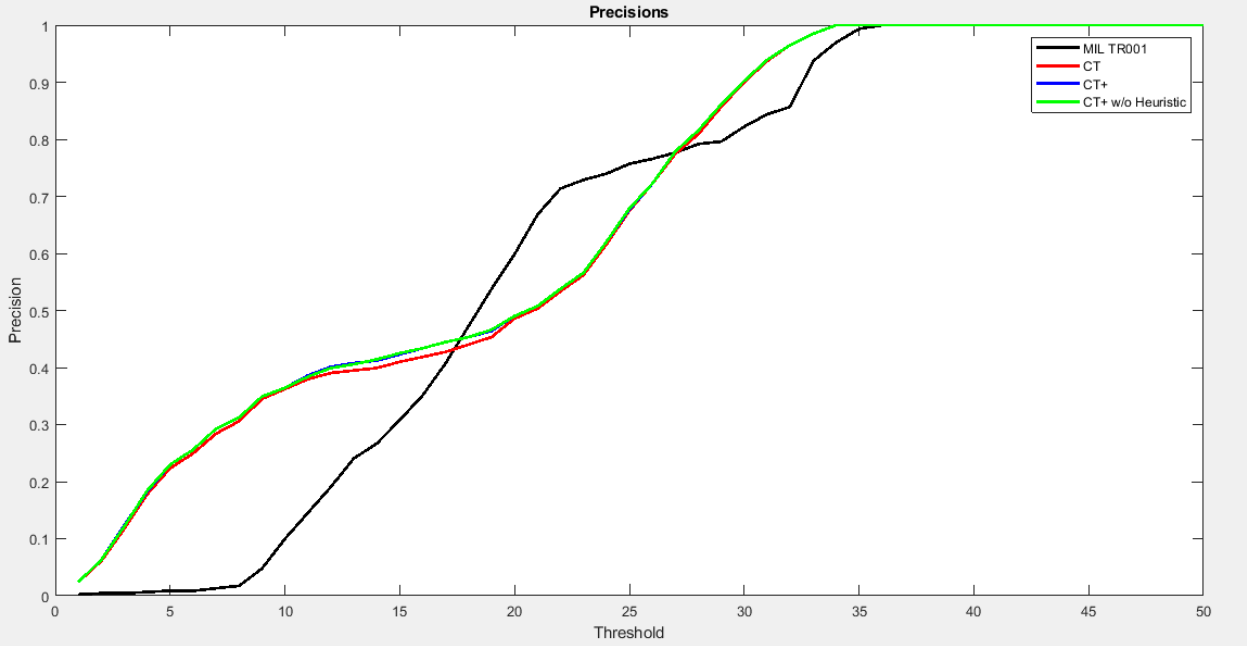


Figure 3

Frames-per-second: 53.9703

From the above precision graph one can observe that the Circulant Tracker Plus performs better than the original Circulant Tracker. Circulant Tracker Plus works relatively the same regardless of using the Heuristic developed during the project.

From *figure 4* shown below shows a moment where David turns his head and causes the CT (red) to try and follow him, while the CT+ algorithm (blue) doesn't feel as confident in the location the hankel matrix predicts so keeps the previous frames coordinates for its reported bounding box. The heuristic has made the algorithm robust to small changes in appearances whether it is the result of occlusion or lighting changes.

figure 4

4.2 Experiment 2

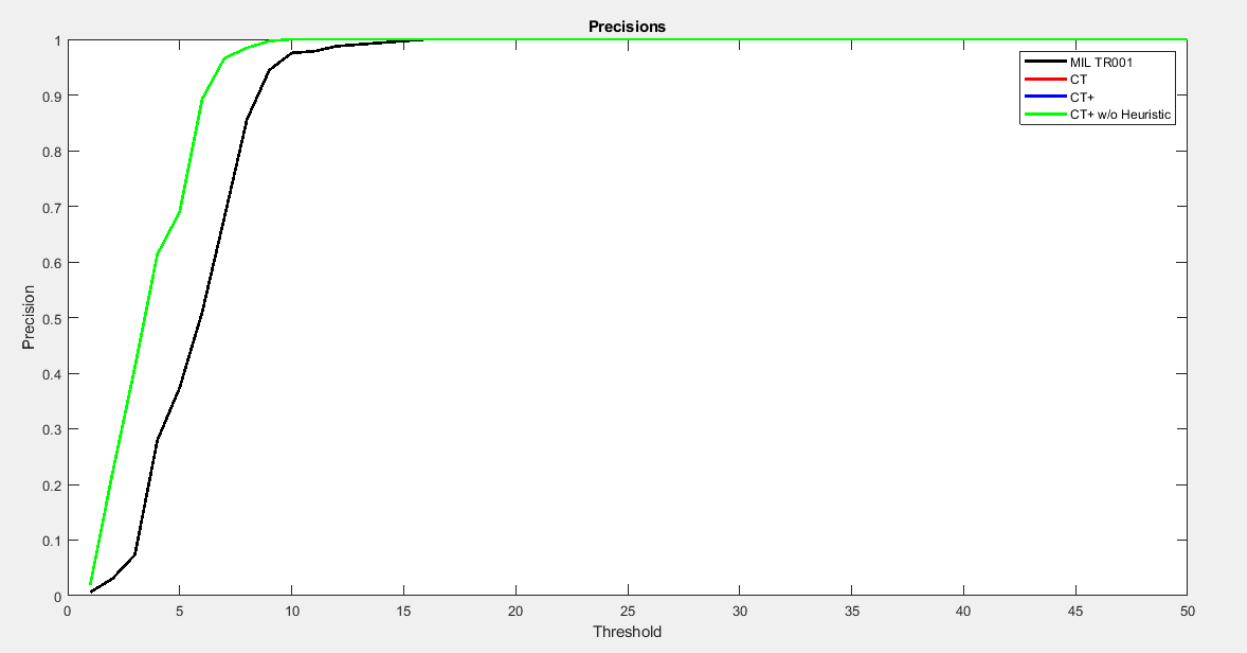


Figure 5

Frames-per-second: 86.8096

Experiment two demonstrates that all versions of the Circulant Tracker work better than MIL TR001. Experiment two has the most consistent lighting, and occlusion once occurring, remains constant. This

change in appearance is what CT and CT+ excel at, as it tracks an object whose foreground and background ratio change overtime. The agorithms have no memory of the original template once a change in appearance has occurred. This experiment also shows that the addition of the Hankel Matrix, and the heuristic made no impact on the precision of the circulant tracker.

4.3 Experiment 3

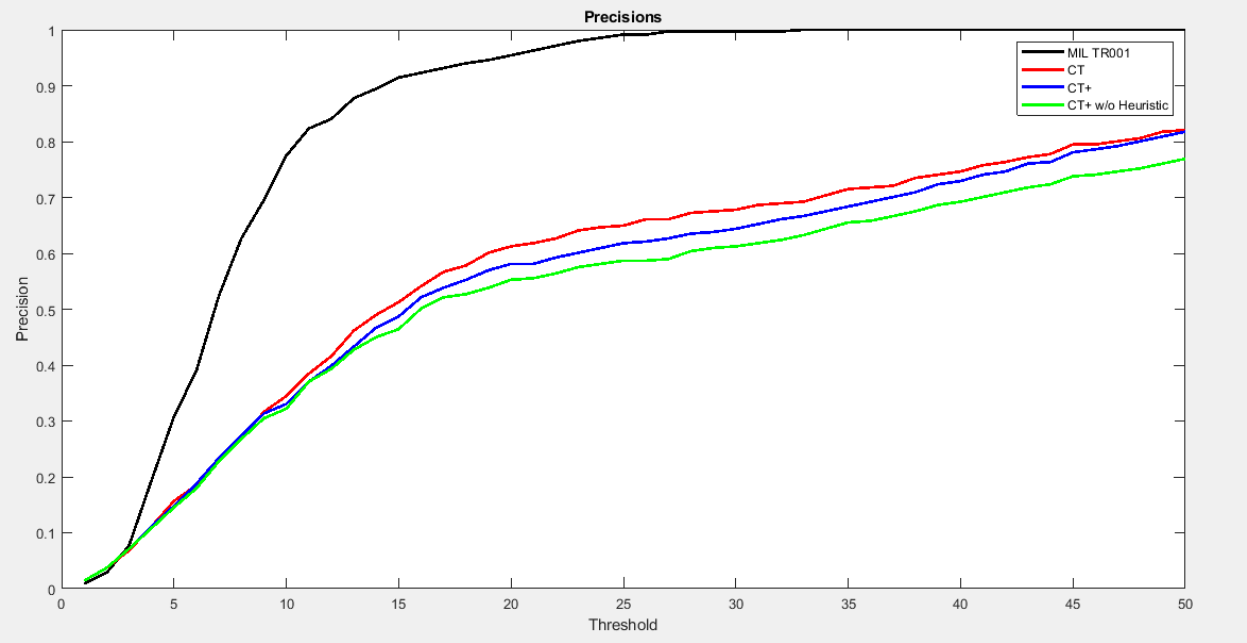


Figure 6

With Frames-per-second: 136.4022

The last two experiments demonstrate that the Circulant Tracker is still not good enough to adapt to more typical forms of occlusion that involve more substantial obstruction of the target object or when the target changes appearance drastically either due to mechanics or lighting. In these experiments a toy tiger is obscured behind a plant and has its mouth open at various points like a sock puppet. The movement is faster and causes blurring and the object occasionally passes under a bright light further obscuring the toy. MIL TR001 adapts better to these changes in appearance than any version of the Circulant Tracker.

However, the tiger experiments are the first two experiments where the addition of the heuristic is demonstrably better than the Circulant Tracker the project asked us to implement. However, it is still not as good as the original Circulant Tracker algorithm.

4.4 Experiment 4

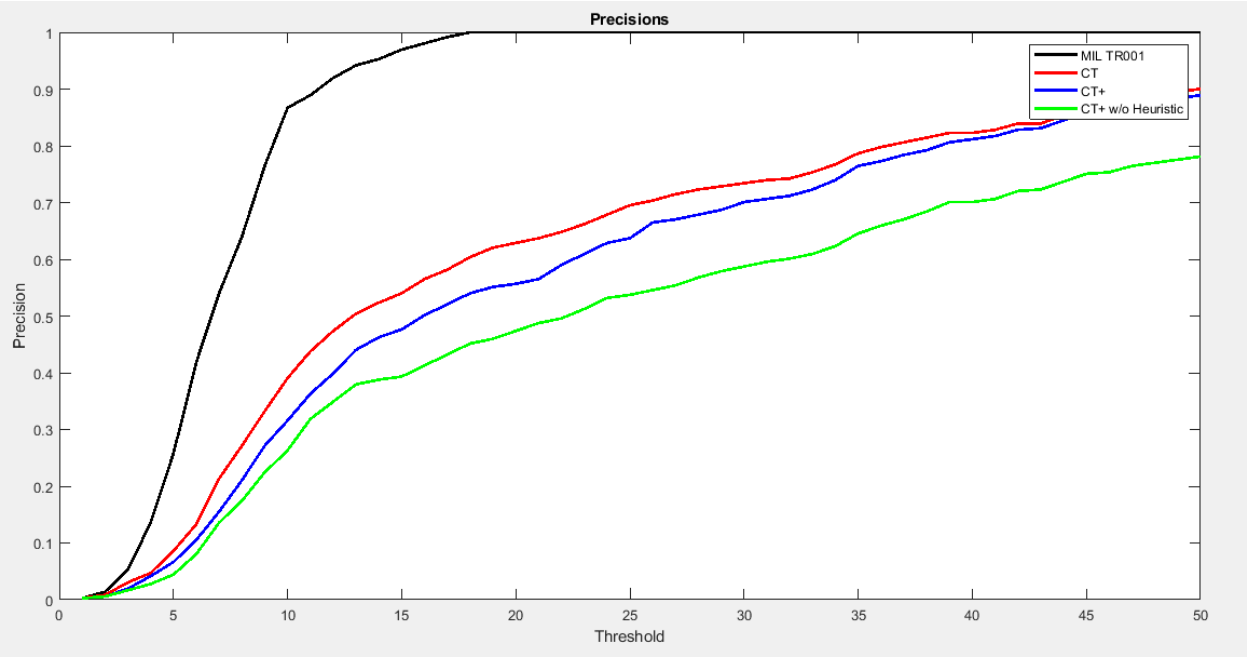


Figure 7

With Frames-per-second: 151.5122

Experiment 4 again demonstrates that the Circulant Tracker Plus that includes the heuristic mentioned in section 2.5 outperforms the circulant tracker that the project asked us to implement. However, all Circulant Tracking algorithms are beaten by the MIL TR001 tracking algorithm.

For the following observations the colors of the bounding box are not consistent with the precision graphs. The new colors in *figure* 8 and on correlate to, green: ground truth, Blue: CT+, and red: the original CT alongside black: the MIL TR001 tracking algorithm.

Figure 8 below demonstrates one of the short comings of the algorithm as the tiger toy moves left in the scene. In this circumstance the image blurs in a way that prevents the algorithm from accepting the estimated position of the Hankel matrix because the response from the kernel is degraded due to blurring compared to the score attained when the sub window was not blurred.

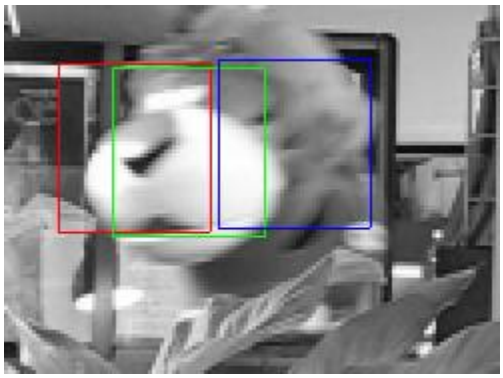


Figure 8

Figure 9 shows how the Hankel matrix improves the circulant tracker during ideal movement. The toy moves right slow enough to not cause blurring. The original circulant tracker loses the toy when it moves behind the bush. But due to the dynamics captured by the hankel matrix, the CT+ algorithm 'finds' the template as it moves to the right out from the bush.

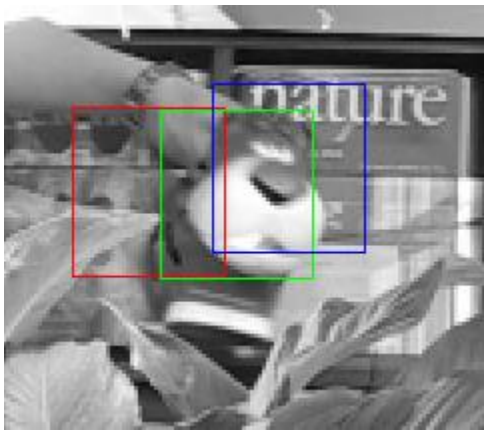


Figure 9

Figure 10 shows that even with the improvements the Circulant tracker brings to the table the improvements also have a cost associated with them. The tracker trains so quickly on a new template that if the CT fails to find the target object, it can potentially lose that object forever as it adapts to the new object it confused as the obscured target. *Figure 10* shows the estimated position of both CT and CT+ at a point in the video where it completely stops following the toy and begins tracking this segment of the bush that previously occluded the toy.

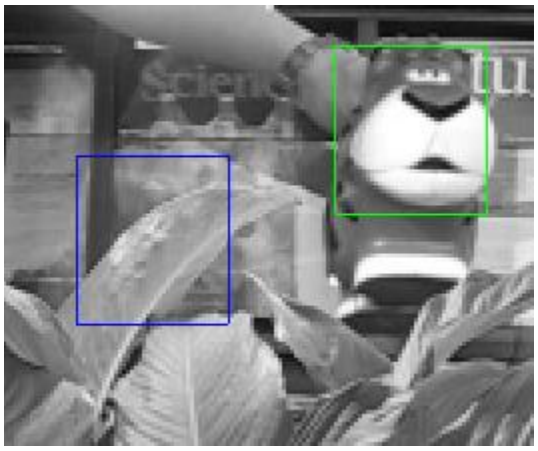


Figure 10

Figure 11 shows the object moving as it opens its mouth exposing a bright light pointed at the camera. This light obscures the template image and the estimated position from the hankel matrix is rejected for having a response that is lower than the expected template that was captured without any light. Drastic changes in appearance especially due to lighting prevent the CT+ with the heuristic from working as well as the original CT. However, from *figure 7* the CT+ algorithm developed still handles the dynamics in the experiment better overall compared to not including the heuristic even when it leads to imperfect results like in the captured frame seen in *figure 11*.

figure 11

5 Conclusion

The purpose of this project was to understand the advantages and limitations of the Circulant Tracking algorithm discussed in class. The underlying algorithm was introduced in the paper“Exploiting the Circulant Structure of Tracking-by-detection with Kernels.”. This project asked us to take the algorithm a step further and attempt to detect when the target object being tracked becomes occluded. The project asked when occlusion is detected to use the Hankel matrix methodology discussed in class to estimate the new positions of the occluded target. We took this a step further and added a heuristic that added the rule that the kernel response measured in the estimated position needed to be higher than the last

non-occluded response that was measured for the Hankel estimated position to be accepted. The results of our experiments showed that adapting to occlusion remains challenging and an active area of research. The heuristic we added improves the results of the CT tracker the project asks us to implement but we suspect that is only because it usually leads to the Hankel matrix estimate to be ignored. Using the Hankel matrix, while maybe works in ideal situations, did little to overcome the problems present when a tracked target object becomes obscured or occluded from the camera.

6 Appendix

6.1 run_tracker.m

bb

```
%  Exploiting the Circulant Structure of Tracking-by-detection with Kernels
%
%  Main script for tracking, with a gaussian kernel.
%
%  João F. Henriques, 2012
%  http://www.isr.uc.pt/~henriques/


%choose the path to the videos (you'll be able to choose one with the GUI)
base_path = './data/';


%parameters according to the paper
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;          %linear interpolation factor for adaptation



%notation: variables ending with f are in the frequency domain.

%ask the user for the video
video_path = choose_video(base_path);
if isempty(video_path), return, end  %user cancelled
[img_files, pos, target_sz, resize_image, ground_truth, mil_tr001, ...
    mil_tr002, mil_tr003, mil_tr004, mil_tr005, video_path] = ...
    load_video_info(video_path);


%window size, taking padding into account
sz = floor(target_sz * (1 + padding));

%desired output (gaussian shaped), bandwidth proportional to target size
output_sigma = sqrt(prod(target_sz)) * output_sigma_factor;
[rs, cs] = ndgrid((1:sz(1)) - floor(sz(1)/2), (1:sz(2)) - floor(sz(2)/2));
y = exp(-0.5 / output_sigma^2 * (rs.^2 + cs.^2));
yf = fft2(y);

%store pre-computed cosine window
cos_window = hann(sz(1)) * hann(sz(2))';

algos = ["CT", "gt", "MIL_TR001", "CT+"];
positions = zeros(numel(algos), numel(img_files), 2);  %to calculate
precision
time = 0;  %to calculate FPS
% positions = zeros(numel(img_files), 2);  %to calculate precision
v = [];
```

```
occlusionPos = zeros(numel(img_files), 2);
lastKnownLoc.loc = [0 0];
lastKnownLoc.score = 0;
lastKnownLoc.frame = 1;

for frame = 1:numel(img_files),
    %load image
    im = imread([video_path img_files{frame}]);
    if size(im,3) > 1,
        im = rgb2gray(im);
    end
    if resize_image,
        im = imresize(im, 0.5);
    end

    tic()

    %extract and pre-process subwindow
    x = get_subwindow(im, pos, sz, cos_window);

    if frame > 1,
        %calculate response of the classifier at all locations
        k = dense_gauss_kernel(sigma, x, z);

        response = real(ifft2(alphaf .* fft2(k)));   %(Eq. 9)
        v(end + 1) = occlusionTest2(response);

        %target location is at the maximum response
        [row, col] = find(response == max(response(:)), 1);
        pos = pos - floor(sz/2) + [row, col];
    end

    %get subwindow at current estimated target position, to train classifer
    x = get_subwindow(im, pos, sz, cos_window);

    %Kernel Regularized Least-Squares, calculate alphas (in Fourier domain)
    k = dense_gauss_kernel(sigma, x);
    new_alphaf = yf ./ (fft2(k) + lambda);   %(Eq. 7)
    new_z = x;

    if frame == 1,  %first frame, train with a single image
        alphaf = new_alphaf;
        z = x;
    else
        %subsequent frames, interpolate model
        alphaf = (1 - interp_factor) * alphaf + interp_factor * new_alphaf;
        z = (1 - interp_factor) * z + interp_factor * new_z;
    end

    if frame > 10 && v(end) < 10
        coloumns = 6;
        eArr = [0];
        order = -1;

        for index=1:2
```

```
            p = positions(1,:,index);
            H = hankel(p(1:coloumns), p(coloumns:frame));
            e = svd(H);
            [U, S, V] = svd(H);
            sArr = zeros(size(S));

            eArr = 0;
            for i = 1:coloumns
                eArr = eArr + e(i);
                if eArr/sum(e) > .9
                    order = i;
                    break;
                end
            end

            for i = 1:order
                sArr(i,i) = e(i);
            end

            H = U*sArr*V';
            A = hankel(H(1:order), H(order:2*order -1));
            b = H(order+1:order*2)';

            ai = inv(A)*b;
            for i = 1:order
                occlusionPos(frame,index) = occlusionPos(frame, index) +
ai(i)*occlusionPos(frame - i, index);
            end
        end

        resp = real(ifft2(alphaf .* fft2(dense_gauss_kernel(sigma,
get_subwindow(im, occlusionPos(frame,:), sz, cos_window), z))));
        % compare to last known...
        if max(resp(:)) > lastKnownLoc.score
            positions(4, frame,:) = occlusionPos(frame,:);
        else
            % positions(4, frame,:) = occlusionPos(frame,:);
            positions(4, frame,:) = lastKnownLoc.loc
            disp('Previous score overrules estimate...')
            % pause
        end
    else
        positions(4, frame,:) = pos;
        occlusionPos(frame,:) = pos;

        %dense_gauss_kernel(sigma, x
        %x = get_subwindow(im, pos, sz, cos_window);
        resp = real(ifft2(alphaf .* fft2(dense_gauss_kernel(sigma,
get_subwindow(im, pos, sz, cos_window), z))));
        lastKnownLoc.score = max(resp(:))
        lastKnownLoc.loc = pos;
        lastKnownLoc.frame = frame;
    end

    %save position and calculate FPS
```

```
    positions(1, frame,:) = pos;
    positions(2, frame,:) = ground_truth(frame,:);
    positions(3, frame,:) = mil_tr001(frame,:);
    time = time + toc();

    %visualization
    rect_position_ct = [pos([2,1]) - target_sz([2,1])/2, target_sz([2,1])];
    a = squeeze(positions(4, frame,:))';
    rect_position_ct_plus = [a([2,1]) - target_sz([2,1])/2,
target_sz([2,1])];

    p = [0 0];
    p(1) = positions(2,frame,2);
    p(2) = positions(2,frame,1);
    rect_position_gt = [p - target_sz([2,1])/2, target_sz([2,1])];

    p(1) = positions(3,frame,2);
    p(2) = positions(3,frame,1);
    rect_position_tr001 = [p - target_sz([2,1])/2, target_sz([2,1])];

    if frame == 1,  %first frame, create GUI
        % figure('Number','off', 'Name',['Tracker - ' video_path])
        im_handle = imshow(im, 'Border','tight', 'InitialMag',200);
        rect_handle1 = rectangle('Position',rect_position_ct,
'EdgeColor','r');
        rect_handle2 = rectangle('Position',rect_position_gt,
'EdgeColor','g');
        rect_handle4 = rectangle('Position',rect_position_ct_plus,
'EdgeColor','b');

    else
        try  %subsequent frames, update GUI
            set(im_handle, 'CData', im)
            set(rect_handle1, 'Position', rect_position_ct)
            set(rect_handle2, 'Position', rect_position_gt)
            % set(rect_handle3, 'Position', rect_position_tr001)
            set(rect_handle4, 'Position', rect_position_ct_plus)
            % legend(['Ground Truth')
        catch  %#ok, user has closed the window
            if size(img_files,1) == frame
                break
            end
            break
        end
    end

    drawnow
end

if resize_image, positions = positions * 2; end


%show the precisions plots
show_precision(positions, ground_truth, video_path)
disp(['Frames-per-second: ' num2str(numel(img_files) / time)])
```

6.2 Occlusion Test

```
function [ret] = occlusionTest2(response)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
[row, col] = find(response == max(response(:)), 1);
Gmax = max(response(:));

for i=-5:5
    for j=-5:5
        response(row + i,col + j) = 0;
    end
end

mu = sum(sum(response)) / (size(response, 1) * size(response, 2) - 121);
sigma = std(response(:));

v = (Gmax - mu) / sigma;
ret = v;
end
```

6.3 Show Precisions

```
function show_precision(positions, ground_truth, title)
%SHOW_PRECISION
%   Calculates precision for a series of distance thresholds (percentage of
%   frames where the distance to the ground truth is within the threshold).
%   The results are shown in a new figure.
%
%   Accepts positions and ground truth as Nx2 matrices (for N frames), and
%   a title string.
%
%   João F. Henriques, 2012
%   http://www.isr.uc.pt/~henriques/

    %load previous run from text file (MILTrack's format)
    % text_files = dir([title(1:end-5) 'previousRun.txt']);
    assert(~isempty([title(1:end-5) 'previousRun.txt']), 'No previous run
(previousRun.txt) to load.')

    previousRun = csvread([title(1:end-5) 'previousRun.txt'])
    %f = fopen([title(1:end-5) 'previousRun.txt']);
    %previousRun = textscan(f, '%f,%f,');  %[x, y]
    %previousRun = cat(2, previousRun{:});
    %fclose(f);

    positions(5,:,:) = previousRun;

    max_threshold = 50;  %used for graphs in the paper
    colors = ['k' 'r' 'b' 'g'];
    %hold on
    c_index = 1;

    %for index = [3 1 4]
    for index = [3 1 4 5] %1:size(positions,1)
        if size(positions(index,:,:),2) ~= size(ground_truth,1),
            disp('Could not plot precisions, because the number of ground')
            disp('truth frames does not match the number of tracked frames.')
            return
        end

        %calculate distances to ground truth over all frames
        distances = sqrt((positions(index,:,1) - ground_truth(:,1)').^2 + ...
                         (positions(index,:,2) - ground_truth(:,2)').^2);
        distances(isnan(distances)) = [];

        %compute precisions
        precisions = zeros(max_threshold, 1);
        for p = 1:max_threshold,
            precisions(p) = nnz(distances < p) / numel(distances);
        end

        %plot the precisions
        % figure('Number','off', 'Name',['Precisions - ' title])
        plot(precisions, colors(c_index), 'LineWidth',2)
        hold on
        c_index = c_index + 1;
```

```
        %legend('MIL TR001','CT','CT+')
        legend('MIL TR001','CT','CT+', 'CT+ w/o Heuristic')
        xlabel('Threshold'), ylabel('Precision')
    end
    csvwrite([title(1:end-5) 'previousRun.txt'],squeeze(positions(4,:,:)))
end
```