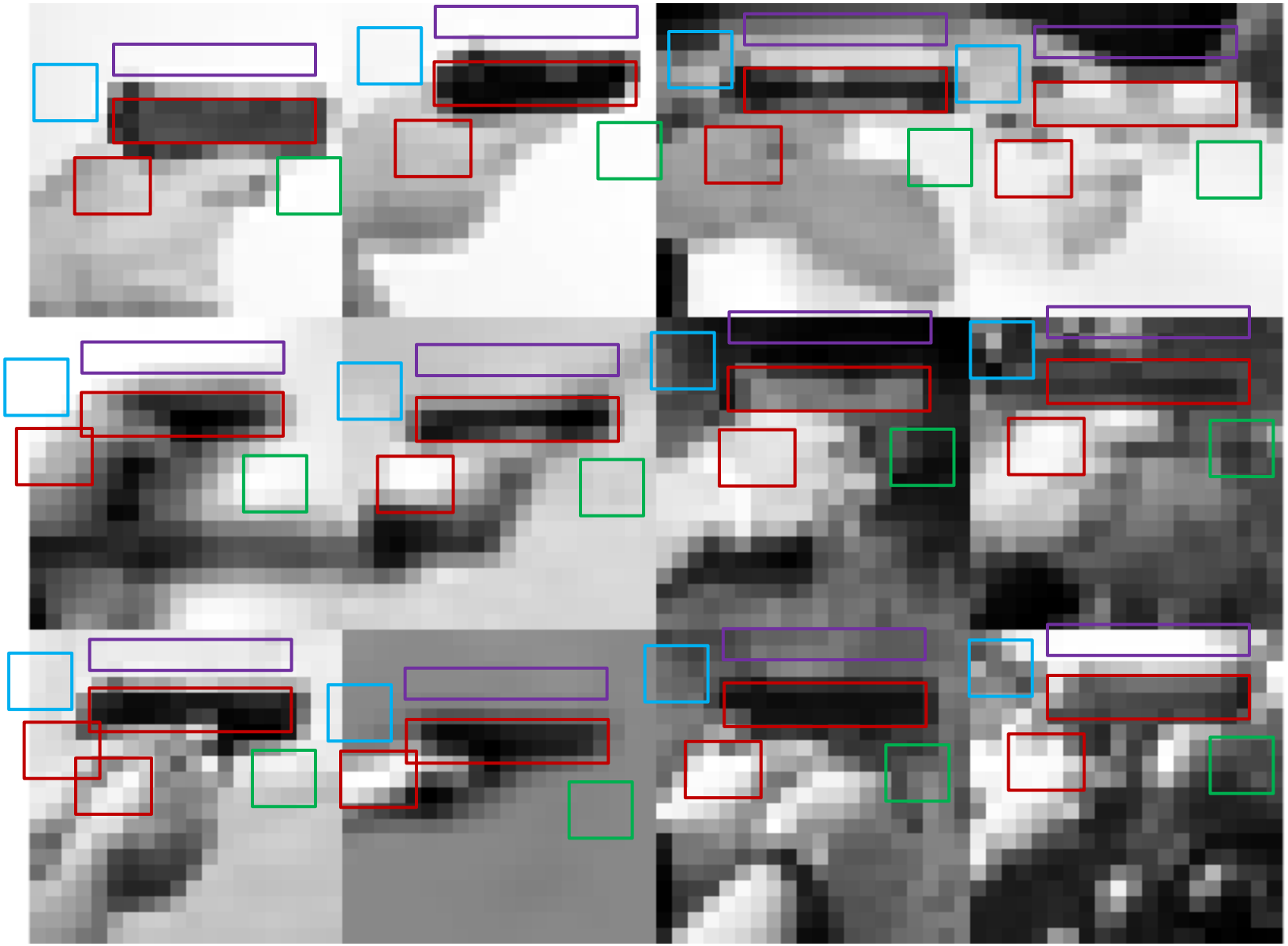
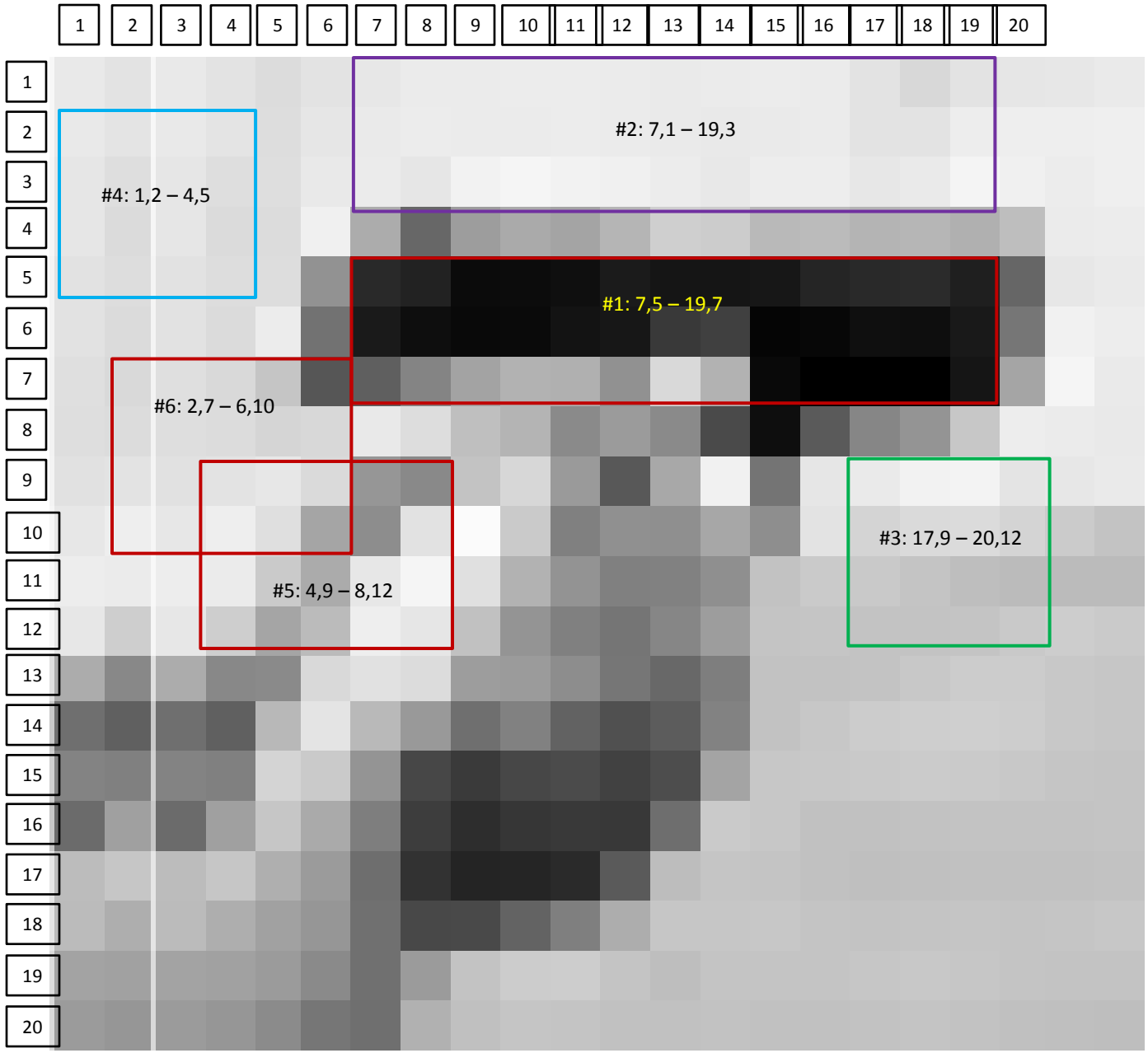
Gun Detection Algorithm

Collage of handgun images

Templates – Take 1



Gun Template – Take 1



Matching Gun Template

MATLAB ▸ Seecure

Editor - C:\Users\ilya\Documents\MATLAB\Seecure\testGuns05.m

testGigE_CL_R2014a_Basler.m | testGigE_CL_R2014a_Basler_8.m | template_means.m | rect_

```
1   %testGuns05
2   %% Prepare
3   close all
4   clear all
5   framenum = 35;
6   prefix = 'C:\Users\ilya\Saphicon\Seecure\Images\Guns\win';
7   suffix = '.jpg';
8   scale = 0.167;
9
10  %% Read
11  name = [prefix num2str(framenum,'%03.0f') suffix];
12  a0 = imread(name);
13  a0 = imresize(a0,scale);
14  figure(1); imshow(a0);
15
16  %% Integral image
17  iimg = integralImage(a0);
18  %figure(2); imshow(uint8(255*iimg./max(max(iimg))));
19
20  %% Template
21  tpl = [ ... % top left bottom right
22      [5 7 7 19]; ...
23      [1 7 3 19]; ...
24      [9 17 12 20]; ...
25      [2 1 5 4]; ...
26      [9 4 12 8]; ...
27      [7 2 10 6] ...
28  ];
29  tpl_ye = max(tpl(:,3));
30  tpl_xe = max(tpl(:,4));
31  %% Calculate
32  [coords, rect_means] = template_means(iimg, tpl);
33
34  %% Test
35  thresh = 80; %80-130
36  flt = test_template(coords, rect_means, thresh);
37
38  %% Display
```

Command Window

```
>> size(flt,1)

ans =

    12

fx >>
```

Figure 1

File Edit View Insert Tools Desktop Window Help

Scale = 0.167. Number of matches = 12.

Matching Gun Template

MATLAB ▸ Seecure

Editor - C:\Users\ilya\Documents\MATLAB\Seecure\testGuns06.m

testGigE_CL_R2014a_Basler.m | testGigE_CL_R2014a_Basler_8.m | template_me

```
6 -   prefix = 'C:\Users\ilya\Saphicon\Seecure\Images\Guns\w
7 -   suffix = '.jpg';
8 -   scale = 0.3333;
9
10    %% Read
11 -  name = [prefix num2str(framenum,'%03.0f') suffix];
12 -  a0 = imread(name);
13 -  a0 = imresize(a0,scale);
14 -  figure(1); imshow(a0);
15
16    %% Integral image
17 -  iimg = integralImage(a0);
18    %figure(2); imshow(uint8(255*iimg./max(max(iimg))));
19
20    %% Template
21 -  tpl = [ ... % top left bottom right
22        [5 7 7 19]; ...
23        [1 7 3 19]; ...
24        [9 17 12 20]; ...
25        [2 1 5 4]; ...
26        [9 4 12 8]; ...
27        [7 2 10 6] ...
28    ];
29 -  tpl = tpl*2;
30 -  tpl_ye = max(tpl(:,3));
31 -  tpl_xe = max(tpl(:,4));
32    %% Calculate
33 -  [coords, rect_means] = template_means(iimg, tpl);
34
35    %% Test
36 -  thresh = 90; %80-130
37 -  flt = test_template(coords, rect_means, thresh);
38
39    %% Display
40 -  figure(1); imshow(a0);
41 -  for i = 1 : size(flt,1)
42 -      rt = flt(i,1); rl = flt(i,2);
43 -      rectangle('Position',[rl rt tpl_xe tpl_ye], 'EdgeC
```

Command Window

```
>> size(flt,1)

ans =

    52

fx >>
```

Scale = 0.333. Number of matches = 52.

Matching Gun Template

MATLAB ▸ Seecure

Editor - C:\Users\ilya\Documents\MATLAB\Seecure\testGuns06.m

testGigE_CL_R2014a_Basler.m | testGigE_CL_R2014a_Basler_8.m

```
1   %testGuns06
2   %% Prepare
3   close all
4   clear all
5   framenum = 35;
6   prefix = 'C:\Users\ilya\Saphicon\Seecure\
7   suffix = '.jpg';
8   scale = 0.5; %0.3333;
9
10  %% Read
11  name = [prefix num2str(framenum,'%03.0f')
12  a0 = imread(name);
13  a0 = imresize(a0,scale);
14  figure(1); imshow(a0);
15
16  %% Integral image
17  iimg = integralImage(a0);
18  %figure(2); imshow(uint8(255*iimg./max(ma
19
20  %% Template
21  tpl = [ ... % top left bottom right
22      [5 7 7 19]; ...
23      [1 7 3 19]; ...
24      [9 17 12 20]; ...
25      [2 1 5 4]; ...
26      [9 4 12 8]; ...
27      [7 2 10 6] ...
28  ];
29  tpl = tpl*3; %2;
30  tpl_ye = max(tpl(:,3));
31  tpl_xe = max(tpl(:,4));
32  %% Calculate
33  [coords, rect_means] = template_means(iim
34
35  %% Test
36  thresh = 90; %80-130
37  flt = test_template(coords, rect_means, t
38
```

Command Window

```
>> size(flt,1)

ans =

   123

fx >>
```

Scale = 0.5. Number of matches = 123.

Matching Gun Template

MATLAB ▸ Seecure

Editor - C:\Users\ilya\Documents\MATLAB\Seecure\testGuns06.m

testGigE_CL_R2014a_Basler.m | testGigE_CL_R2014a_Basle

```
1   %testGuns06
2   %% Prepare
3   close all
4   clear all
5   framenum = 35;
6   prefix = 'C:\Users\ilya\Saphicon\Seec
7   suffix = '.jpg';
8   scale = 1; %0.5; 0.3333;
9
10  %% Read
11  name = [prefix num2str(framenum,'%03.
12  a0 = imread(name);
13  a0 = imresize(a0,scale);
14  figure(1); imshow(a0);
15
16  %% Integral image
17  iimg = integralImage(a0);
18  %figure(2); imshow(uint8(255*iimg./ma
19
20  %% Template
21  tpl = [ ... % top left bottom right
22      [5 7 7 19]; ...
23      [1 7 3 19]; ...
24      [9 17 12 20]; ...
25      [2 1 5 4]; ...
26      [9 4 12 8]; ...
27      [7 2 10 6] ...
28  ];
29  tpl = tpl*6; %3; 2;
30  tpl_ye = max(tpl(:,3));
31  tpl_xe = max(tpl(:,4));
32  %% Calculate
33  [coords, rect_means] = template_means
34
35  %% Test
36  thresh = 100; %80-130
37  flt = test_template(coords, rect_mean
38
```

Command Window

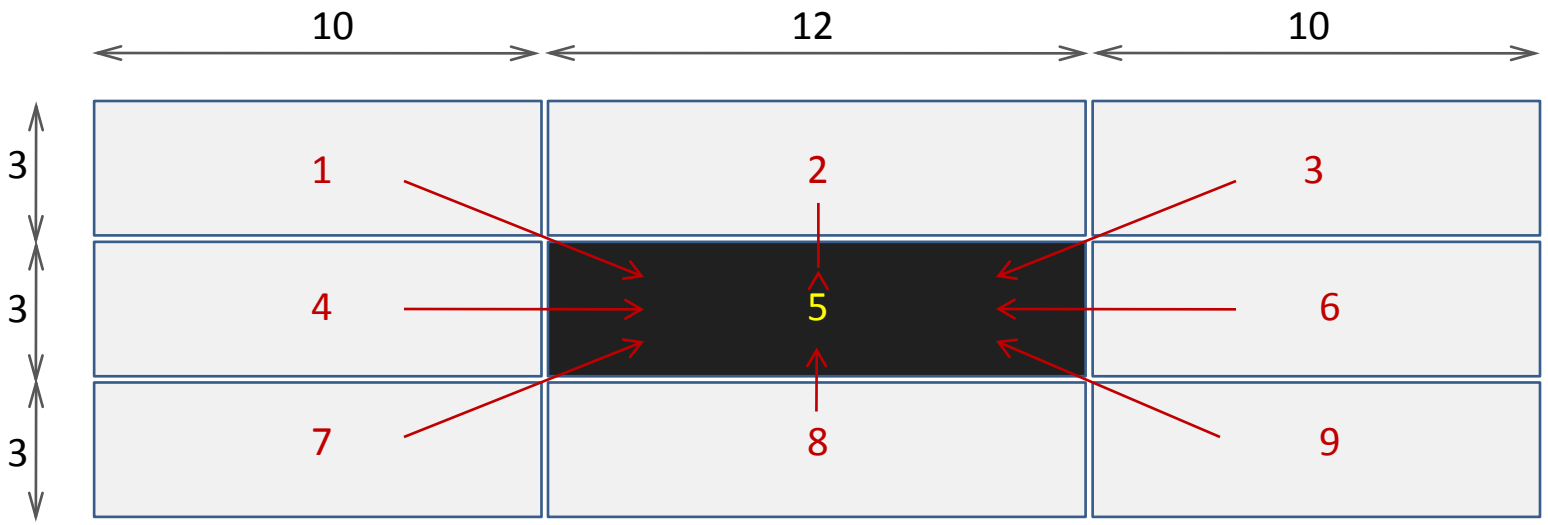
```
>> size(flt,1)

ans =

   402

fx >>
```

Scale = 1. Number of matches = 402.

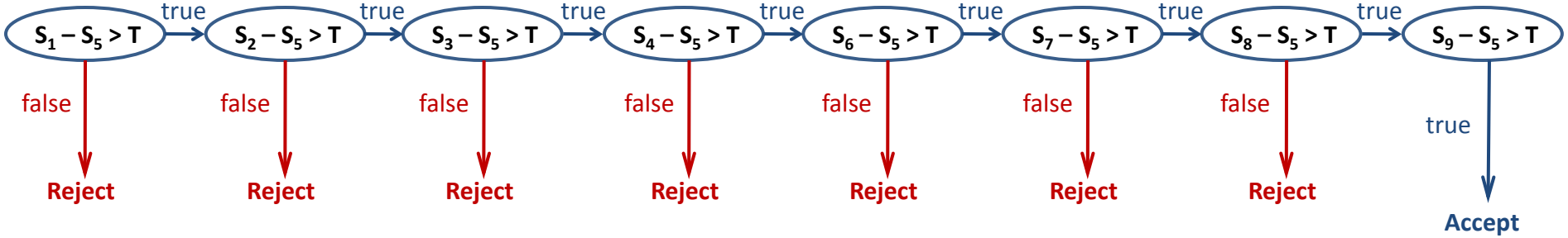
Gun Template – Take 2.

Narrow formulation: looking for black handgun barrels

Overall size: 9 * 32

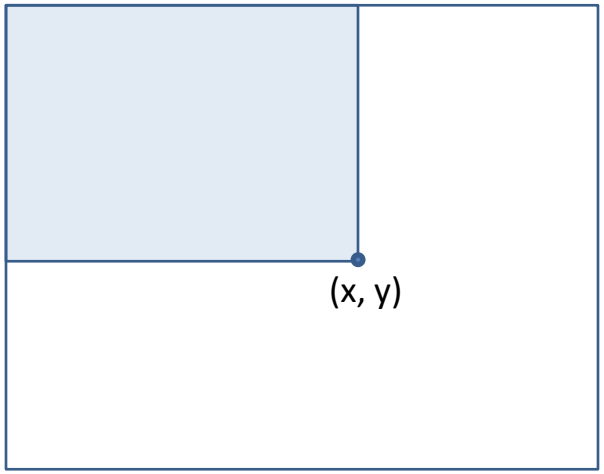


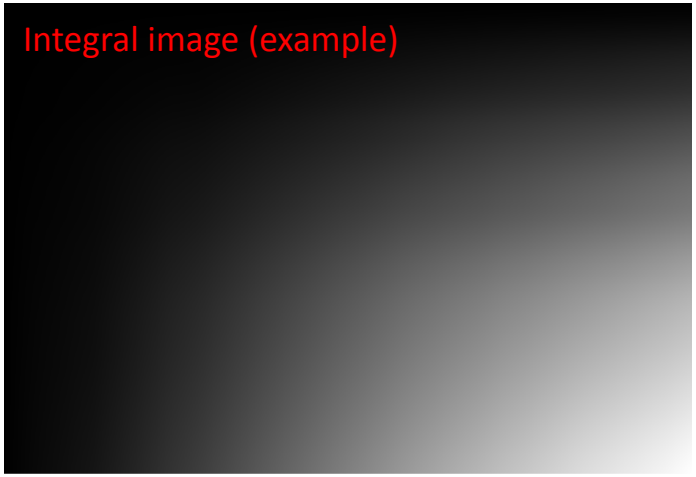
Calculate average image intensity (S) in each of the 9 rectangles and compare it between 8 peripheral and the middle rectangle. The difference has to be greater than threshold (T).



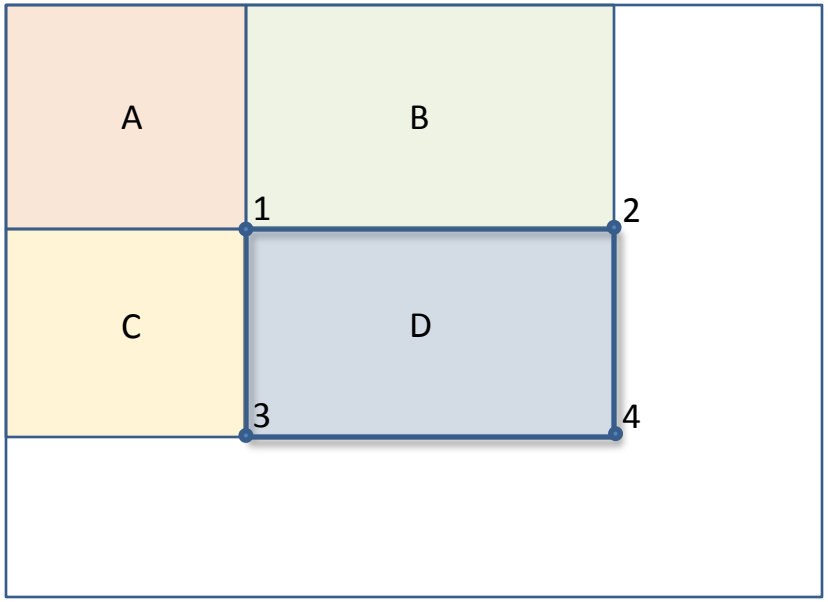
Rejection happens as early as possible.

Rectangle Averages through Integral Image (Viola-Jones)





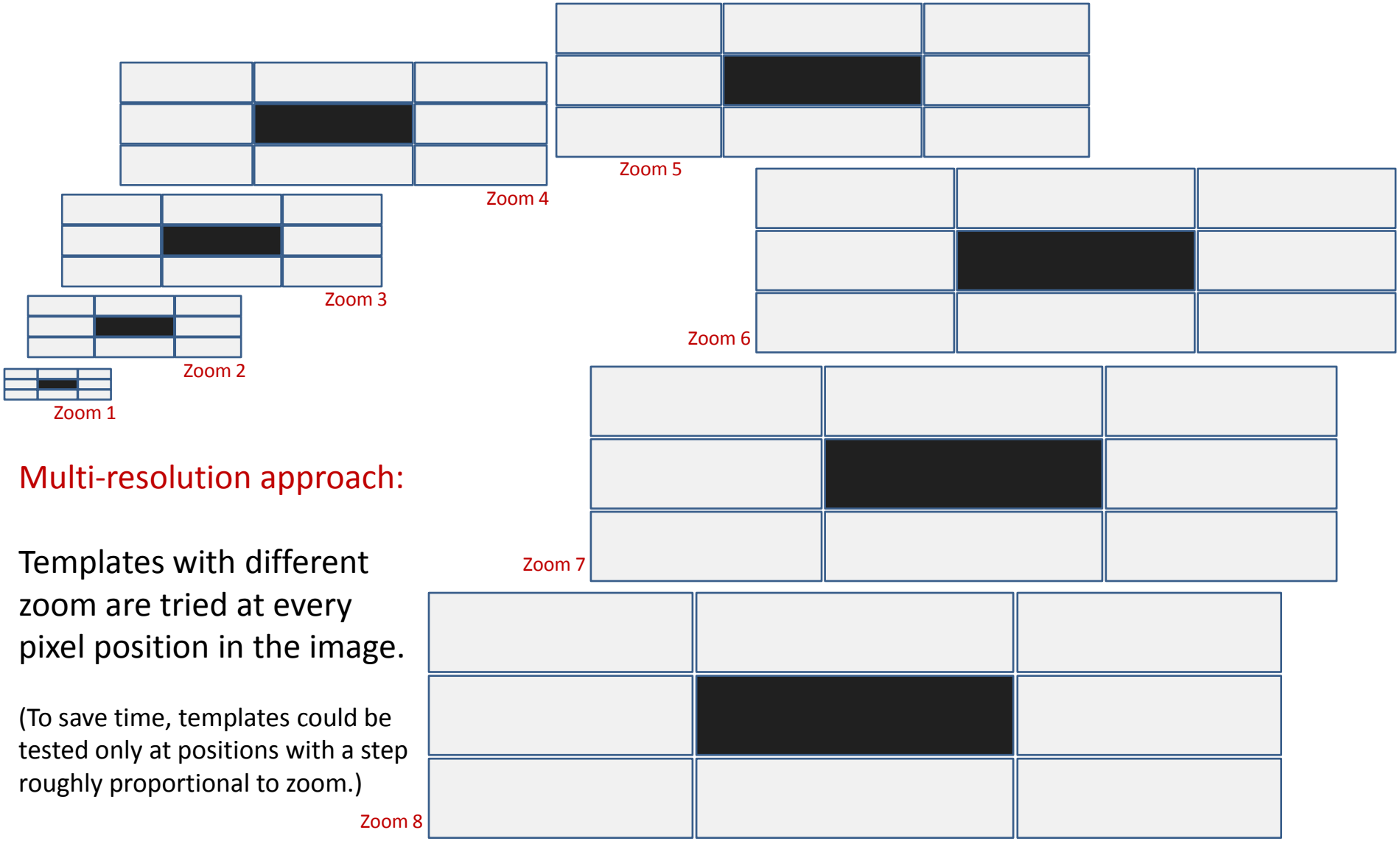
Definition: The value of the integral image at point (x, y) is the sum of all pixels of the original image above and to the left. The integral image is pre-computed once for all further analysis.



The sum (S) of pixels within rectangle D can be computed with four references into the integral image (I).

$$S_D = I_4 + I_1 - I_2 - I_3$$

Zooming Gun Template



Multi-resolution approach:

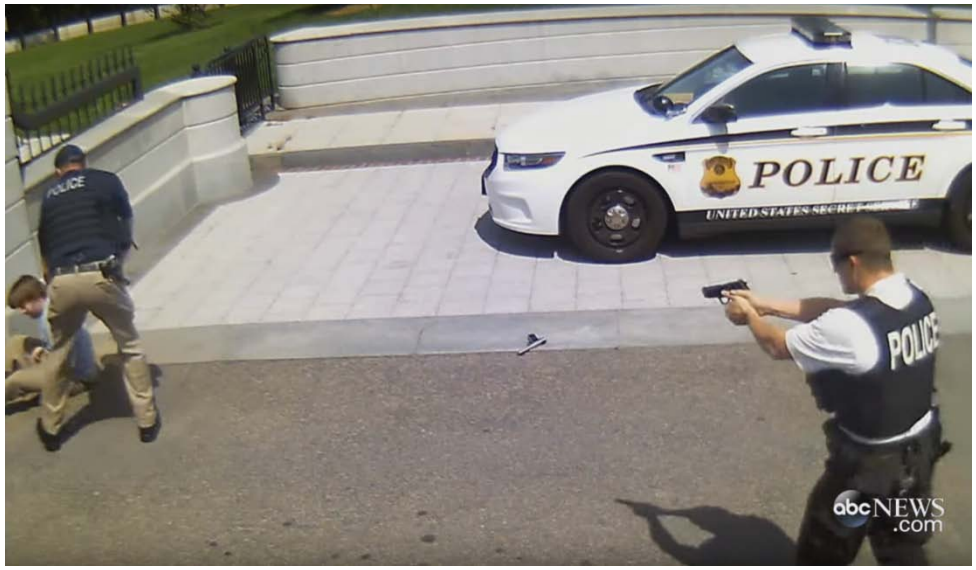
Templates with different zoom are tried at every pixel position in the image.

(To save time, templates could be tested only at positions with a step roughly proportional to zoom.)

Images for Testing

https://www.youtube.com/watch?v=MqRZ-_zRFsU

https://www.youtube.com/watch?v=pkRofcvfnHg



https://www.youtube.com/watch?v=dBllnaSKVfQ

https://www.youtube.com/watch?v=VAedmsJ-Fb0

Our approach vs. Viola-Jones

Our method (for guns)	Viola-Jones (for faces)
Template consists of as many rectangles as needed.	Each feature is computed on two rectangles.
Variations of objects are detected by different templates. Accepts only tight matches.	Removes wrong candidates in successive cascades of classifier.
Low false-positive. Moderate false negative (improved by adding templates).	High false-positive (improved by cascade). Low false negative.
Slower to execute (as of now).	Faster to execute.
No image library is needed and there is no training.	Needs large library of images for training.

Example 1

zoom = 4;
thresh = 70;
step = 2;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);

Several matches are detected for the same object, which are then combined into a single cluster.

Clusters from different template zoom are shown in colors

Loop by:	zoom	thresh	step
	3	70	2
	4	70	2

Example 2

zoom = 7;
thresh = 70;
step = 2;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);

Several matches are detected for the same object, which are then combined into a single cluster.

Clusters from different template zoom are shown in colors

Loop by:	zoom	thresh	step
	6	70	3
	7	70	3
	8	70	3
	9	70	3

Example 3


```
zoom = 6;
thresh = 80;
step = 3;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);
```

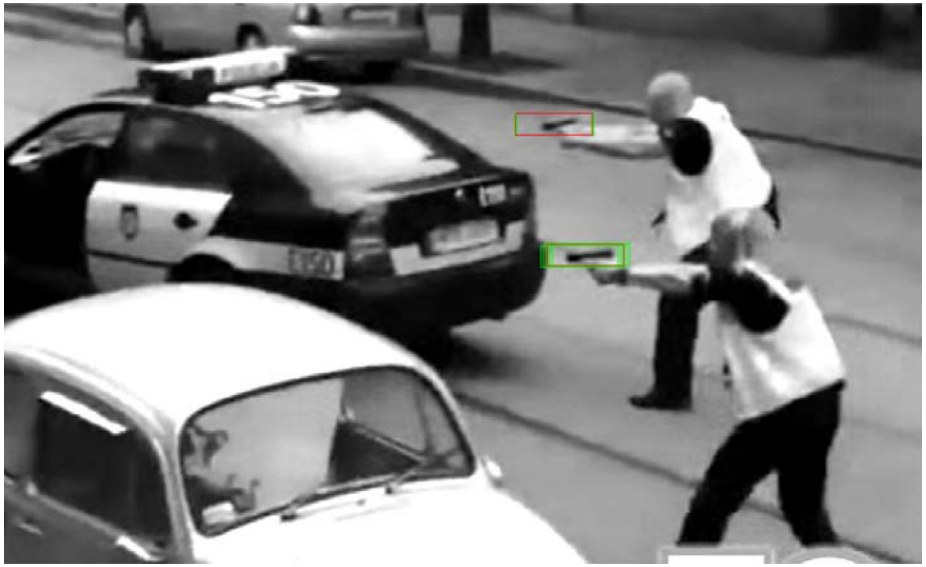
Several matches are detected for the same object, which are then combined into a single cluster.

Loop by:	zoom	thresh	step
	6	80	3
	7	80	3
	8	80	3

Clusters from different template zoom are shown in colors

Loop by:	zoom	thresh	step
	5	80	3 (this zoom generates 2 false-positives)
	6	80	3
	7	80	3
	8	80	3

Example 4



```
zoom = 2;
thresh = 57;
step = 2;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);
```

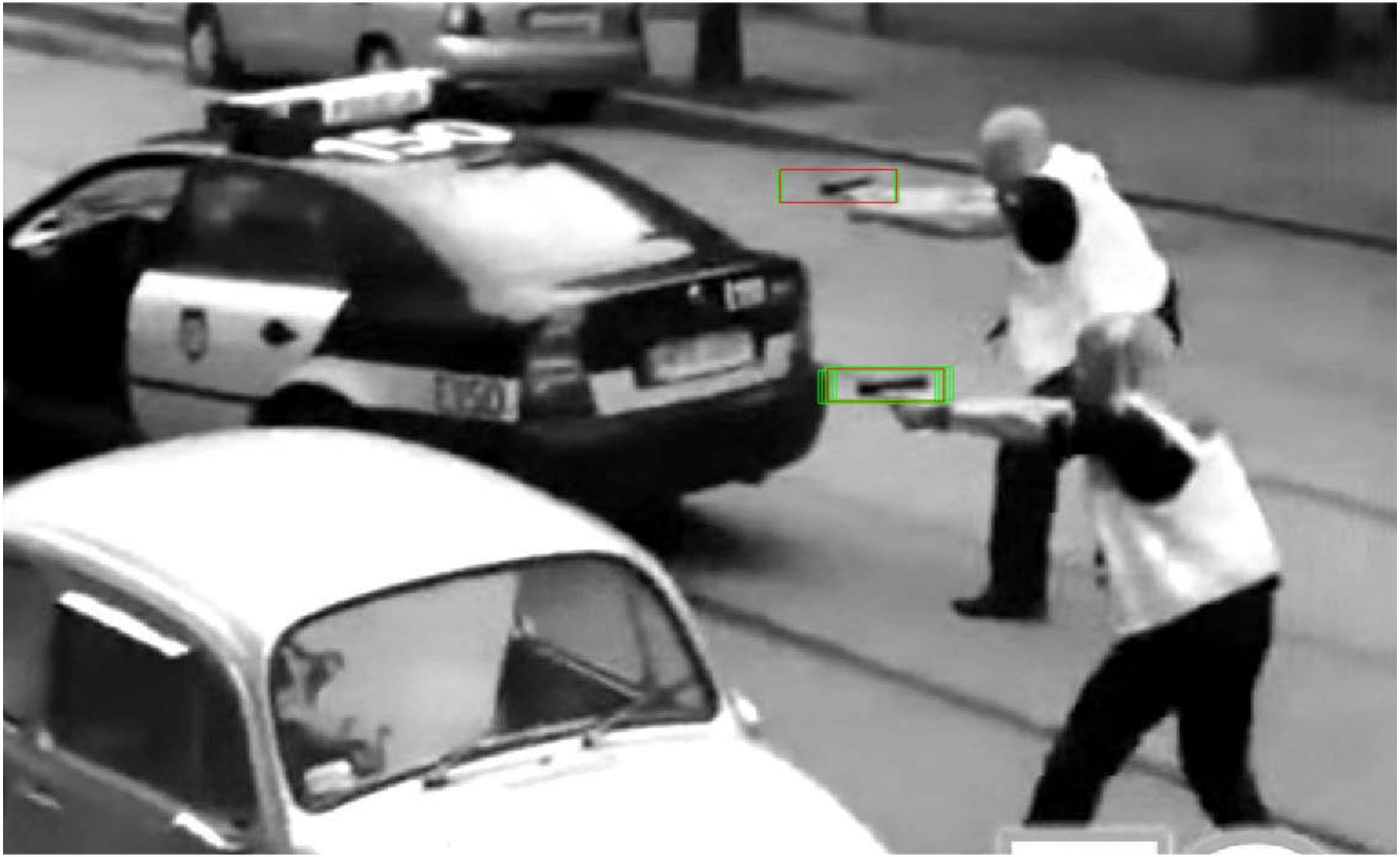
Several matches are detected for the same object, which are then combined into a single cluster.

Loop by:	zoom	thresh	step
	1.75	55	1
	2	57	2
	2.5	65	2

Clusters from different template zoom are shown in colors

Loop by:	zoom	thresh	step
	1.5	50	1 (this zoom generates 2 false-positives)
	1.75	55	1
	2	57	2
	2.5	65	2

Example 4 (1)



```
zoom = 2;
thresh = 57;
step = 2;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);
```

Several matches are detected for the same object, which are then combined into a single cluster.

Example 4 (2)

Loop by:	zoom	thresh	step
	1.75	55	1
	2	57	2
	2.5	65	2

Clusters from different template zoom are shown in colors

Example 4 (3)

Loop by:	zoom	thresh	step	
	1.5	50	1	This zoom generates 2 false-positives (not counting false-positives generated by text superimposed on image)
	1.75	55	1	
	2	57	2	
	2.5	65	2	

Clusters from different template zoom are shown in colors

Example 5


```
zoom = 8;
thresh = 65;
step = 3;
template = zoom * form_template_9 (3, 12, 10);
matches = match_template
                (img, template, thresh, step);
cluster = cluster_template (matches);
```

Several matches are detected for the same object, which are then combined into a single cluster.

Clusters from different template zoom are shown in colors

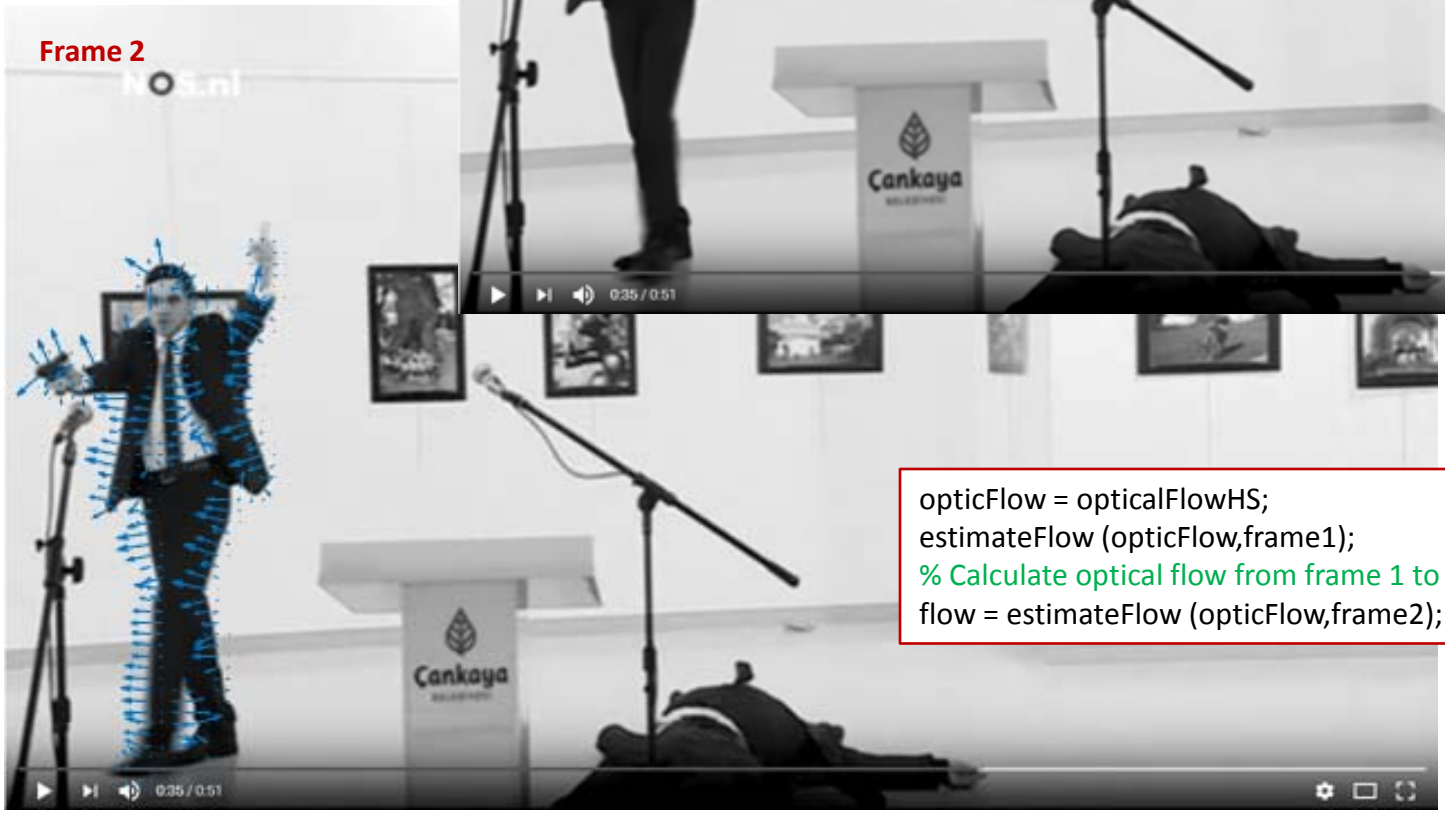
Loop by:	zoom	thresh	step
	7	63	3
	8	65	3
	9	65	3

Example 6 – Using Sequences of Images

scale = 0.5;
zoom = 1;
thresh = 70;
step = 1;
template = zoom * form_template_9 (6,12,6);
matches = match_template (img, template, thresh, step);
cluster = cluster_template (matches);

Sometimes it is hard to eliminate false positives. Analyzing motion can provide a solution.

Example 6 – Optical Flow




```
opticFlow = opticalFlowHS;                % Create optical flow object
estimateFlow (opticFlow,frame1);          % Initialize
% Calculate optical flow from frame 1 to frame 2
flow = estimateFlow (opticFlow,frame2);
```

Example 6 – Finding Guns in Areas of Motion

Range of Parameters

Zoom	1-1.5	2-2.5	>=3
Threshold	~50	50-60	70-80
Step	1	2	>2

Conclusion

- The algorithm can handle rotation within +/-10 degrees, but results are more reliable when barrel is horizontal.
- Need to know the approximate scale, either from the distance between the camera and the object, or derived from the image by independent means.
- Matches with different zoom can be used for confirmation of finding.
- Speed?
- Stability with parameters?
- This algorithm was fast to develop and used a minimal image library.

Next Steps

- Include rotation +/- 20 degrees.
- Analyze shape of candidate objects to reduce false positives and allow more freedom in setting parameters.
- Leverage existing methods based on learning, e.g., use proximity to head/ upper body/ face.
- Create templates for other types and colors of guns.
- Perform preprocessing (contrasting, color correction) to improve gun detection