Table of Contents

load and crop

```
imgs = Prj2TB.read_all_imgs('raw_img/FV5','','TIF');
imgs_cropped = Prj2TB.crop_images(imgs);
```

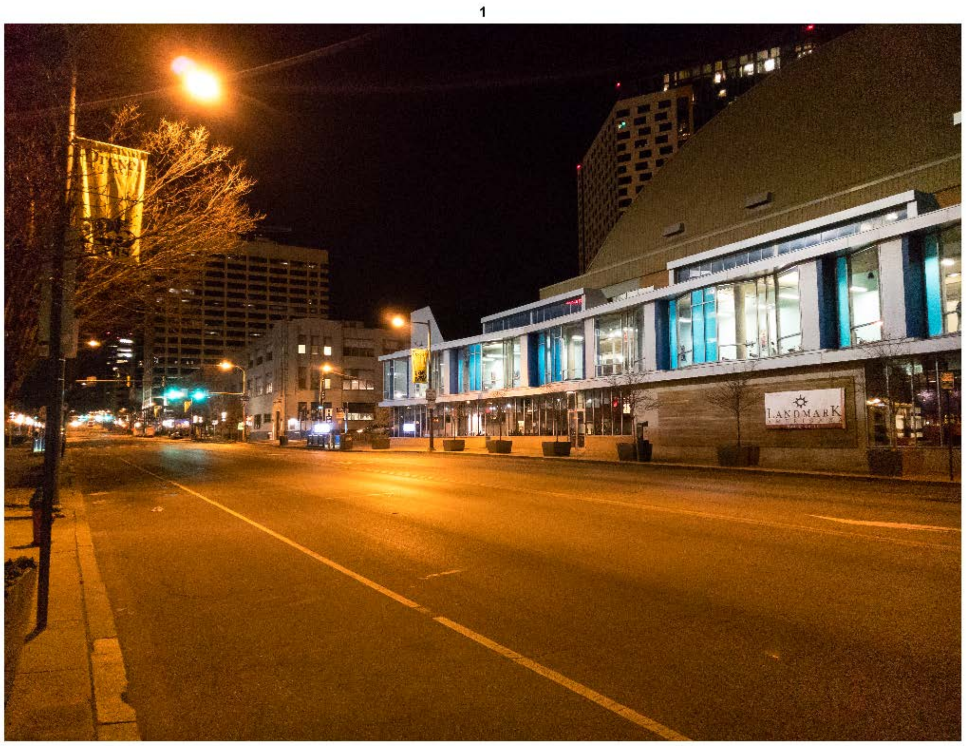
```
Warning: Escaped character '\.' is not valid. See 'doc
sprintf' for supported special characters.
loaded: raw_img\FV5\DSC_0005.TIF
loaded: raw_img\FV5\DSC_0006.TIF
loaded: raw_img\FV5\DSC_0007.TIF
loaded: raw_img\FV5\DSC_0008.TIF
loaded: raw_img\FV5\DSC_0009.TIF
loaded: raw_img\FV5\DSC_0010.TIF
loaded: raw_img\FV5\DSC_0011.TIF
loaded: raw_img\FV5\DSC_0012.TIF
loaded: raw_img\FV5\DSC_0013.TIF
```

show original

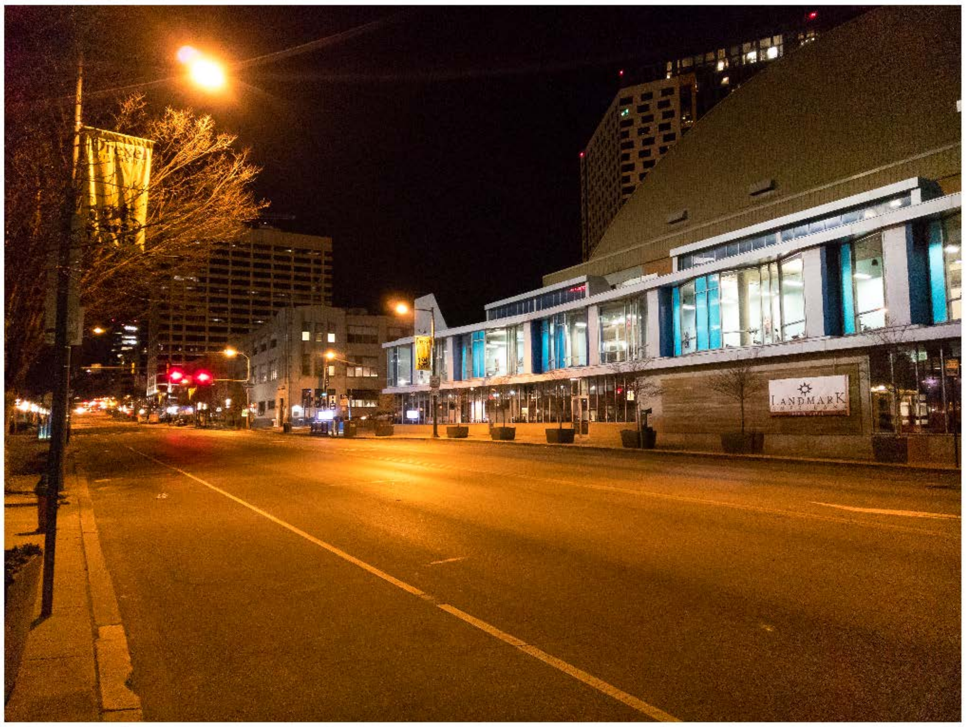
```
close all

for i=1:length(imgs_cropped)
    figure;
    imshow(imgs{i});
    title(num2str(i));
end
```

```
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
```

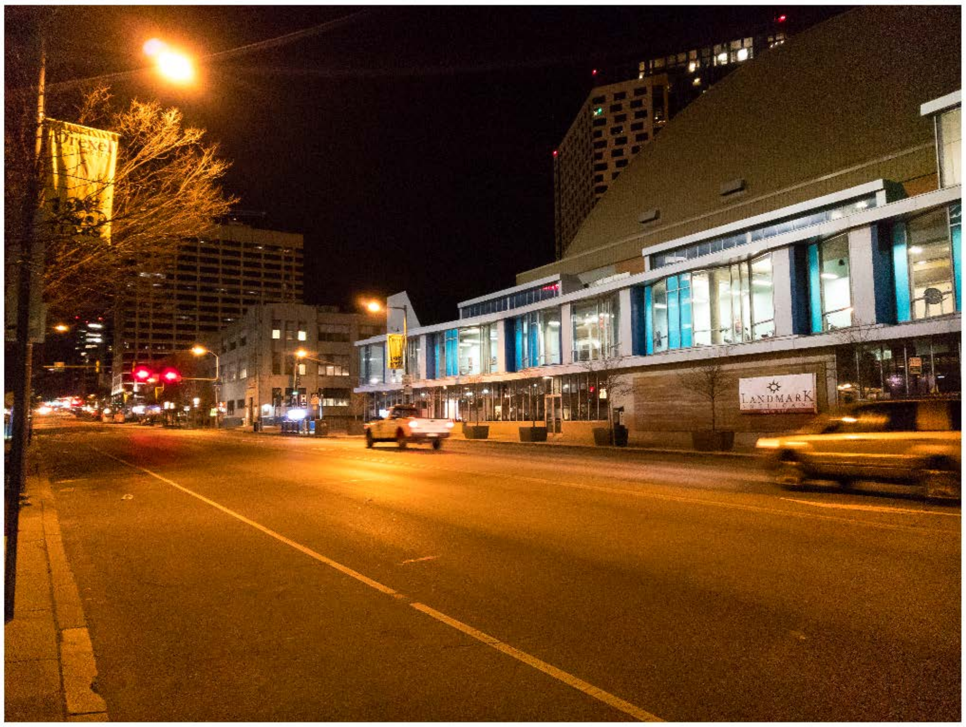
```
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
Warning: Image is too big to fit on screen; displaying at
25%
```



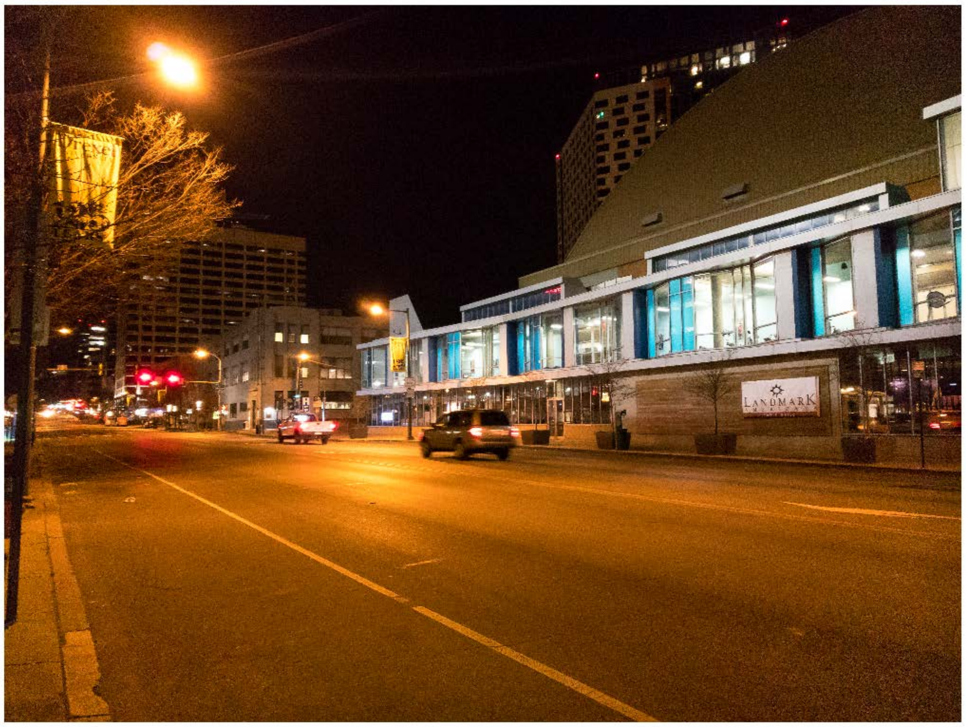
2



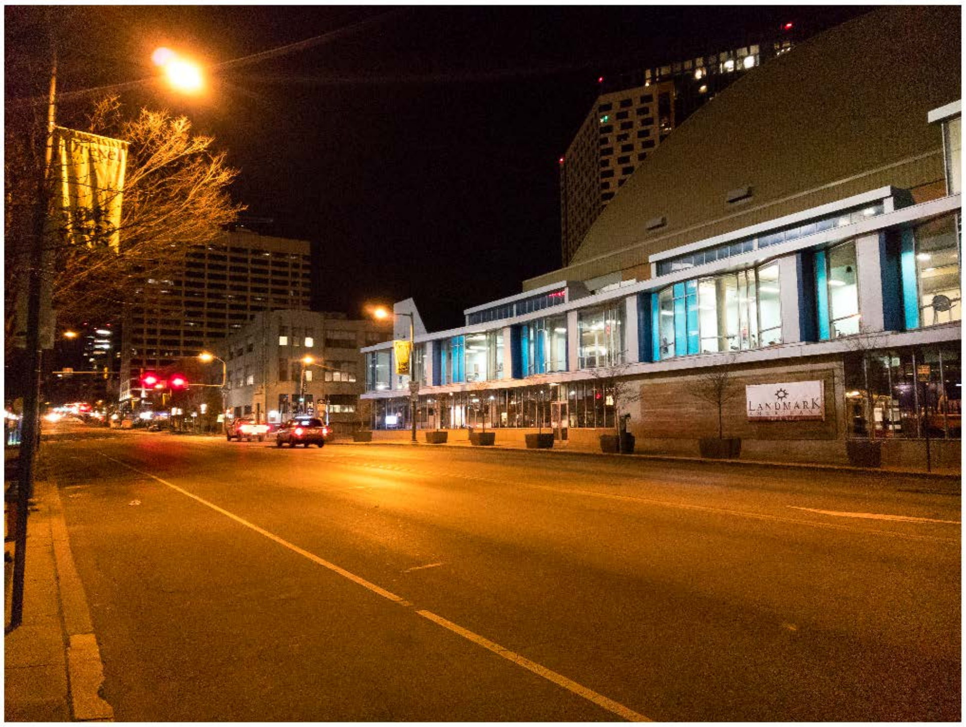
3



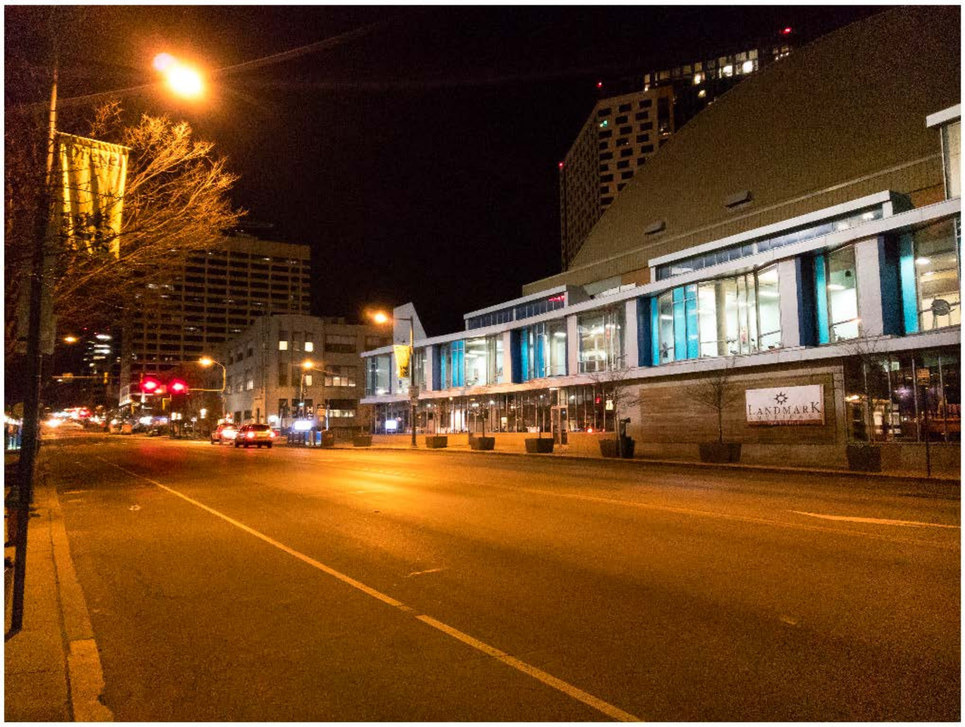
4



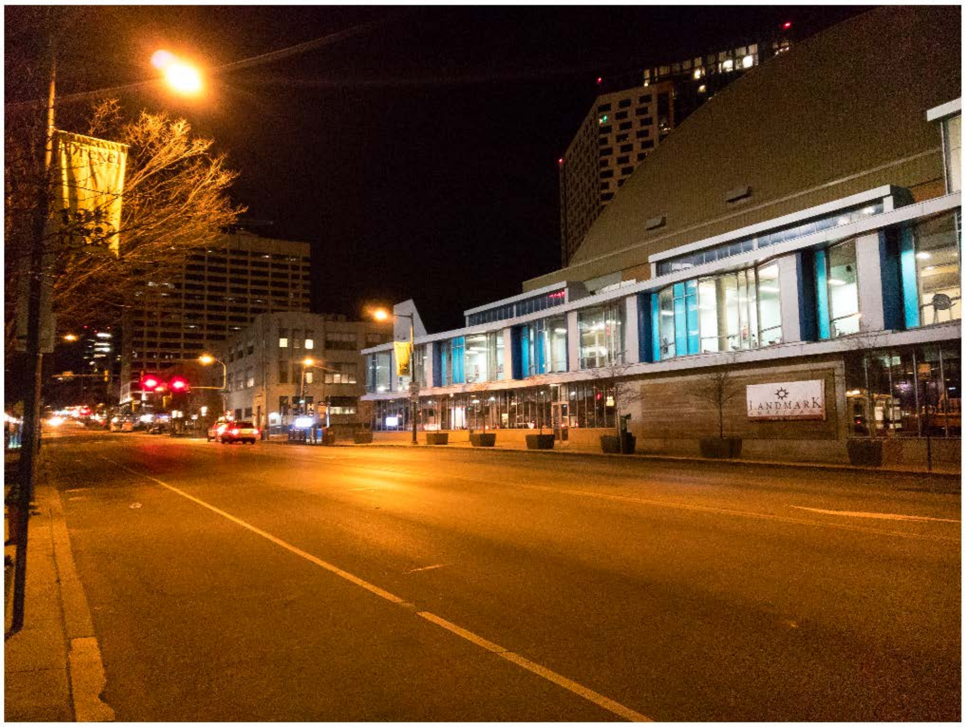
5



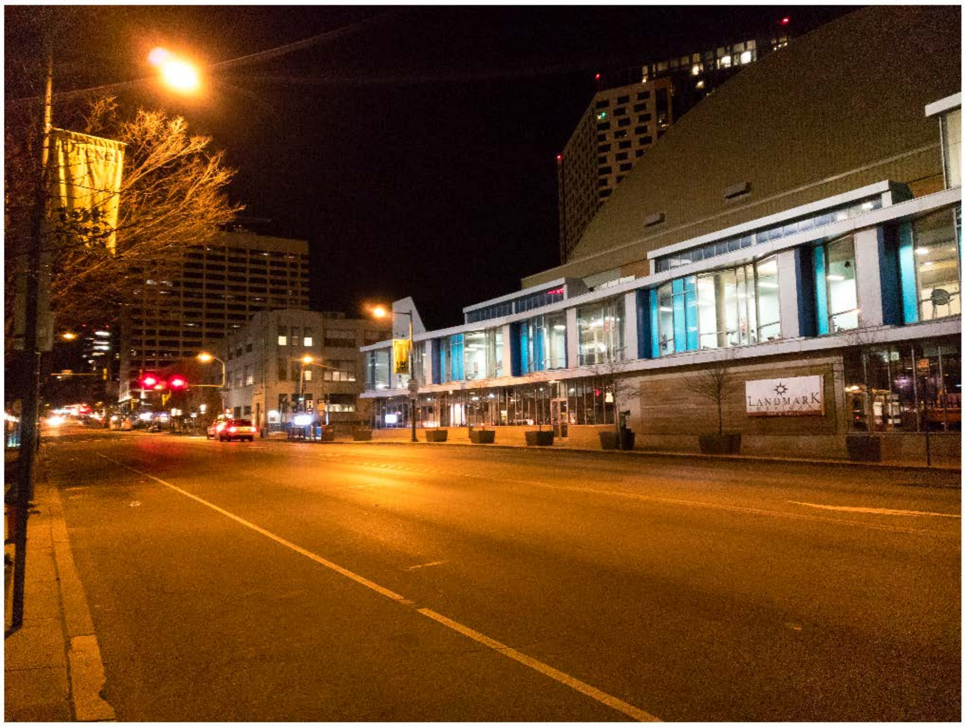
6



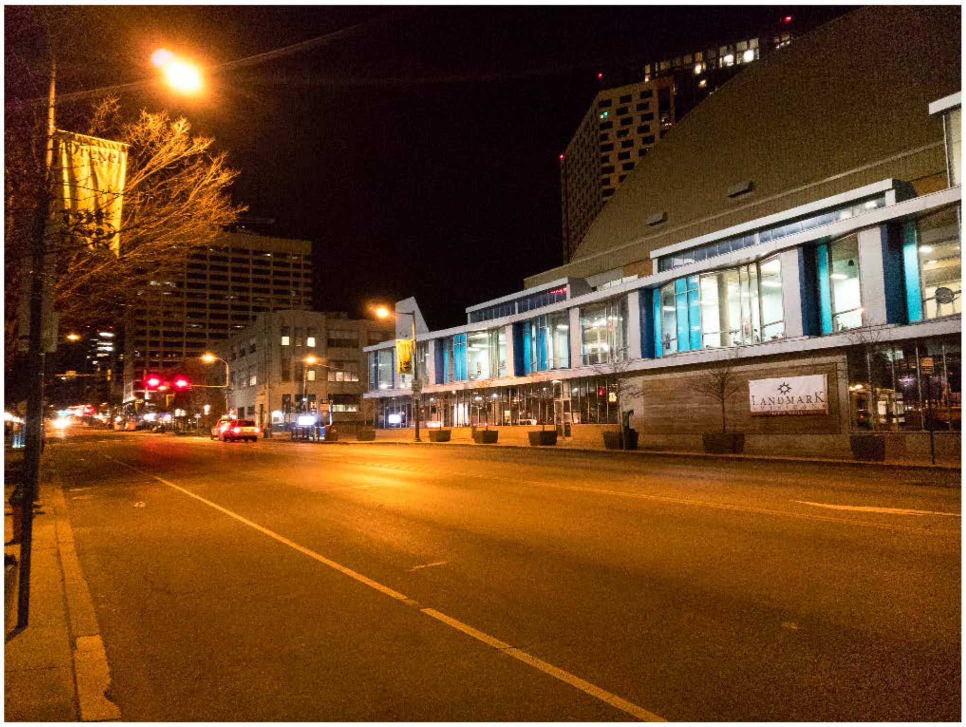
7



8



9



show cropped

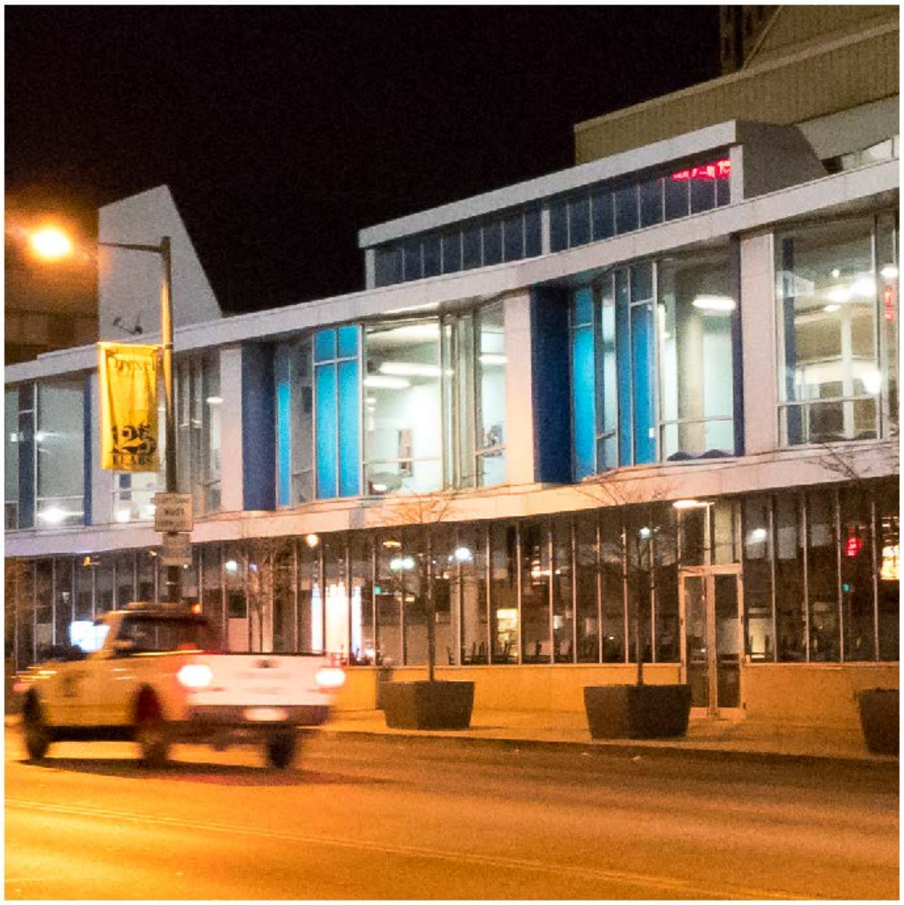
```
close all

for i=1:length(imgs_cropped)
    figure;
    imshow(imgs_cropped{i});
    title(num2str(i));
end

% use cropped images for afterwards
imgs = imgs_cropped;
```

```
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
```

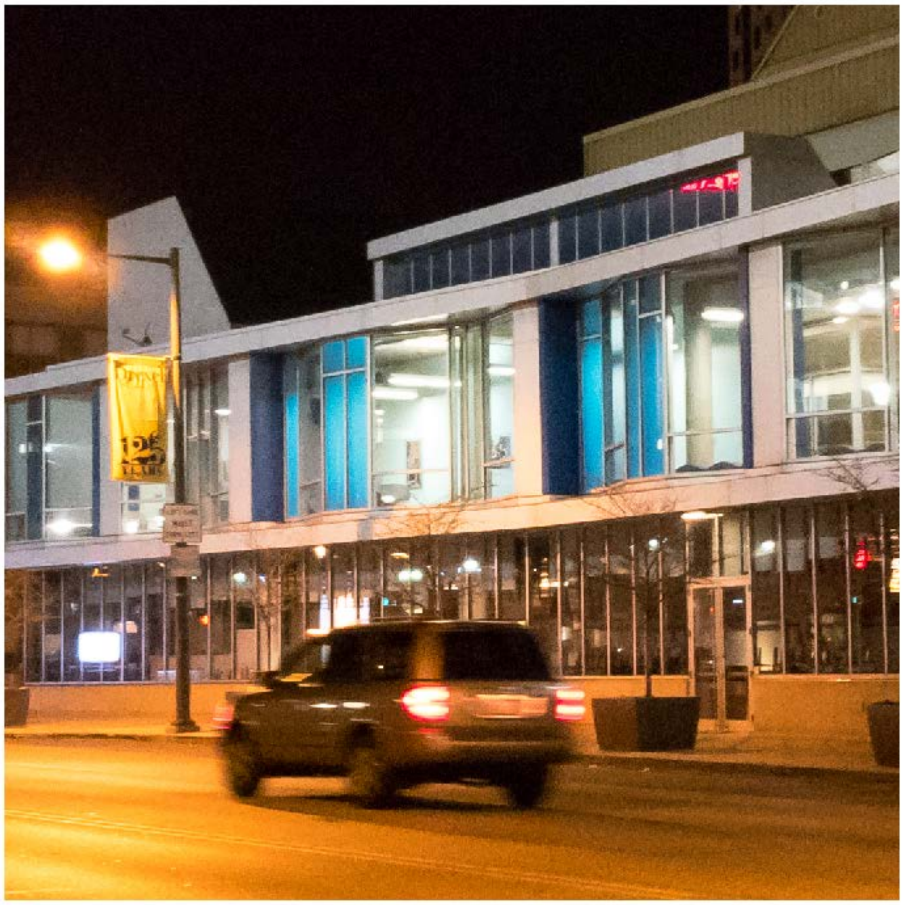
1

2

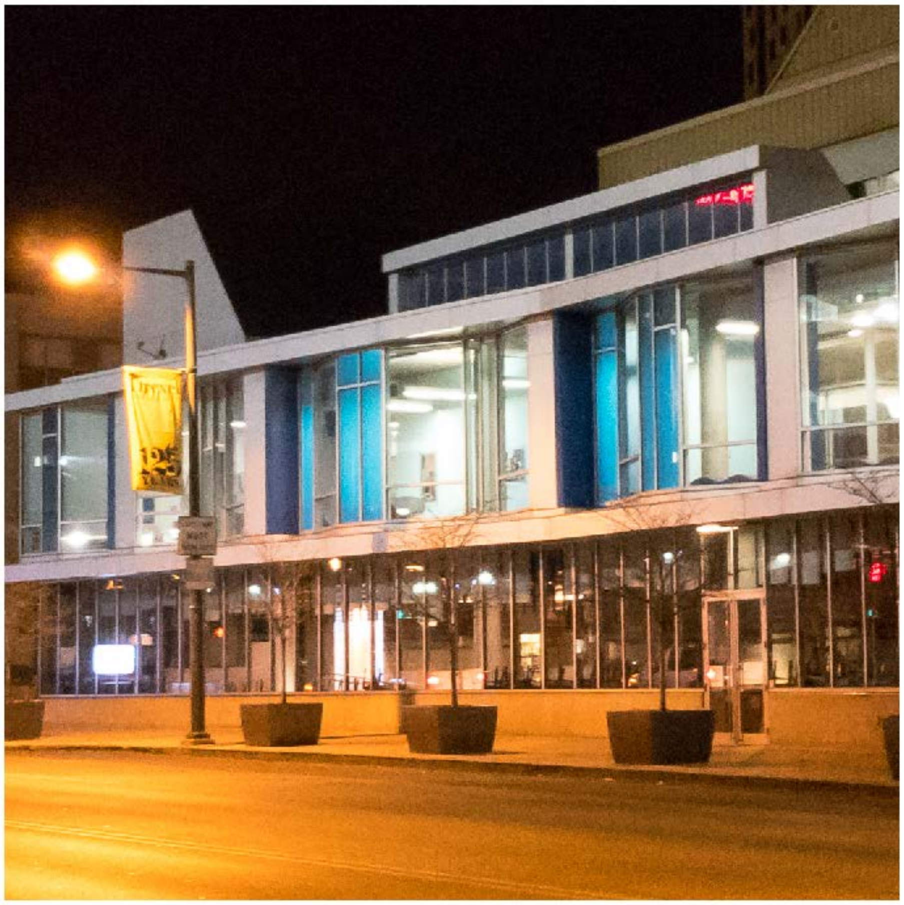
3



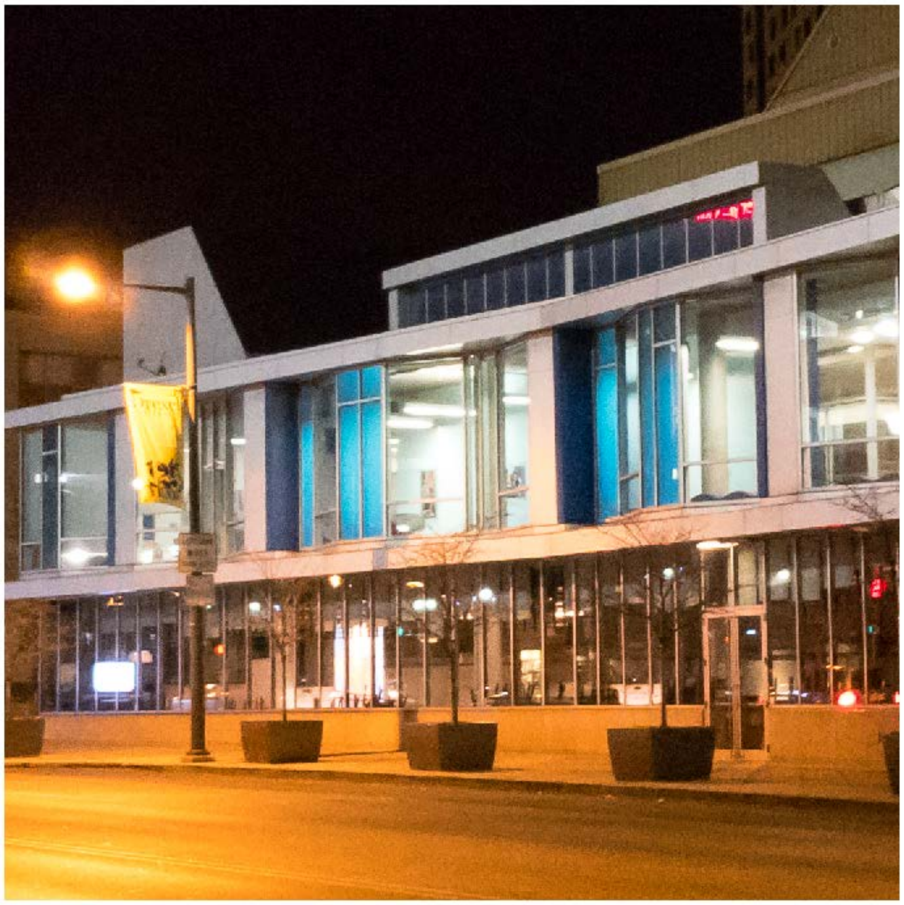
4



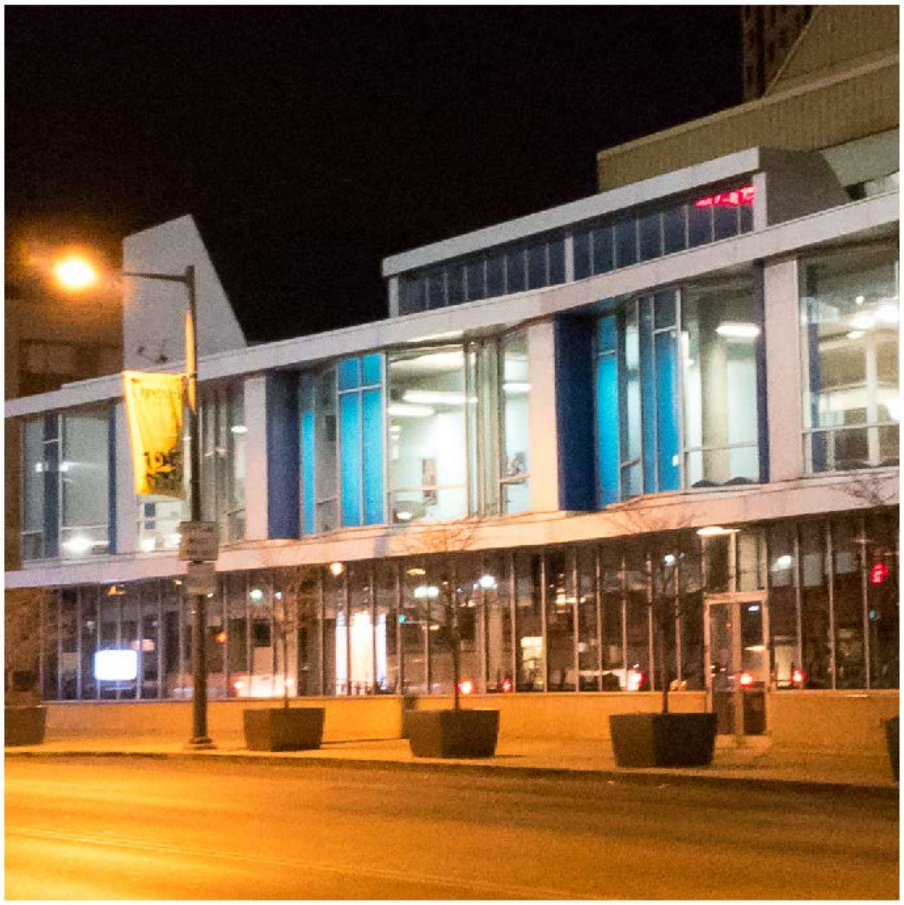
5



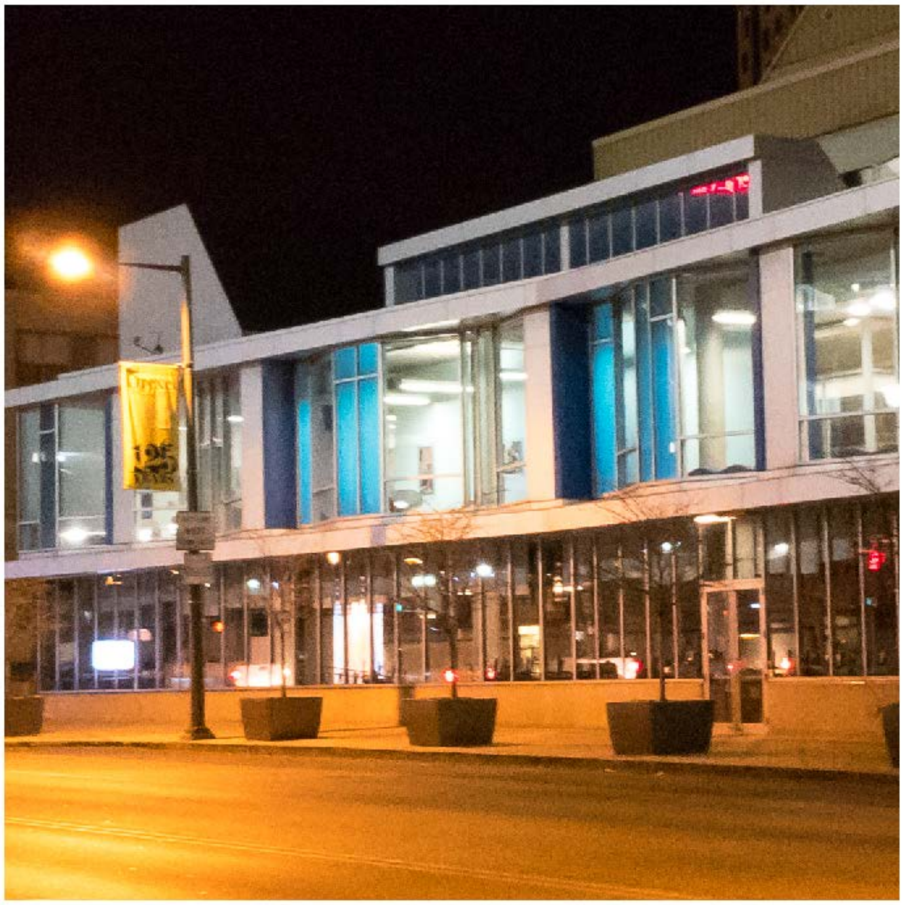
6



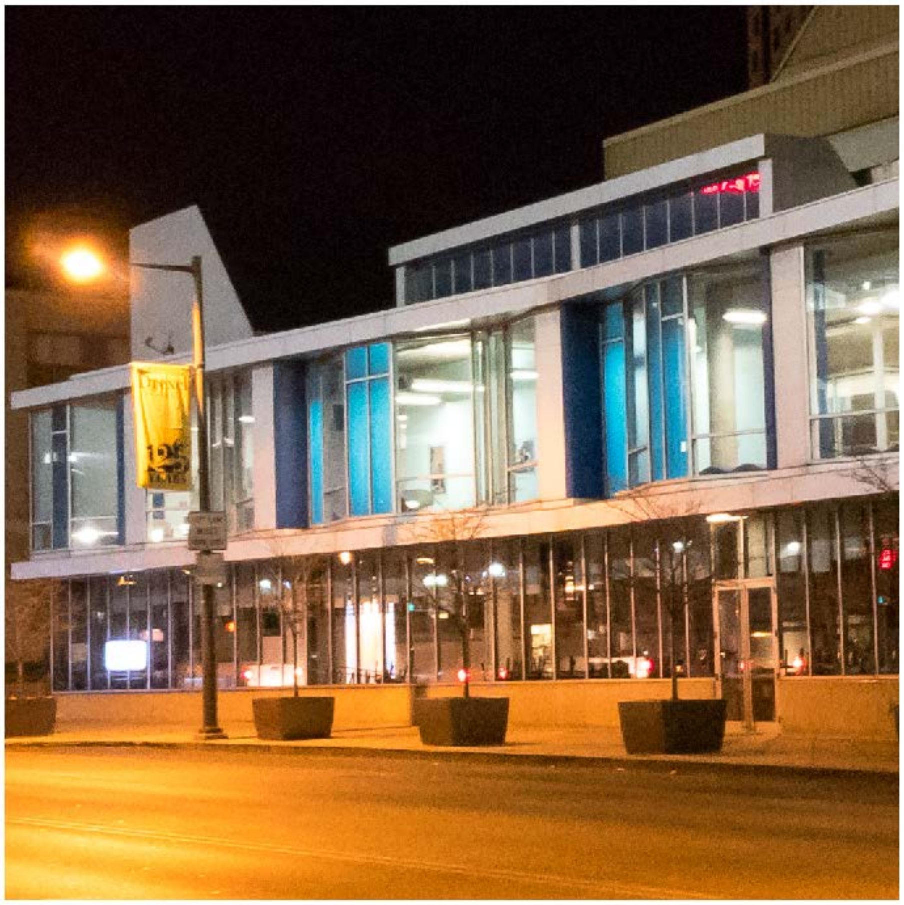
7



8



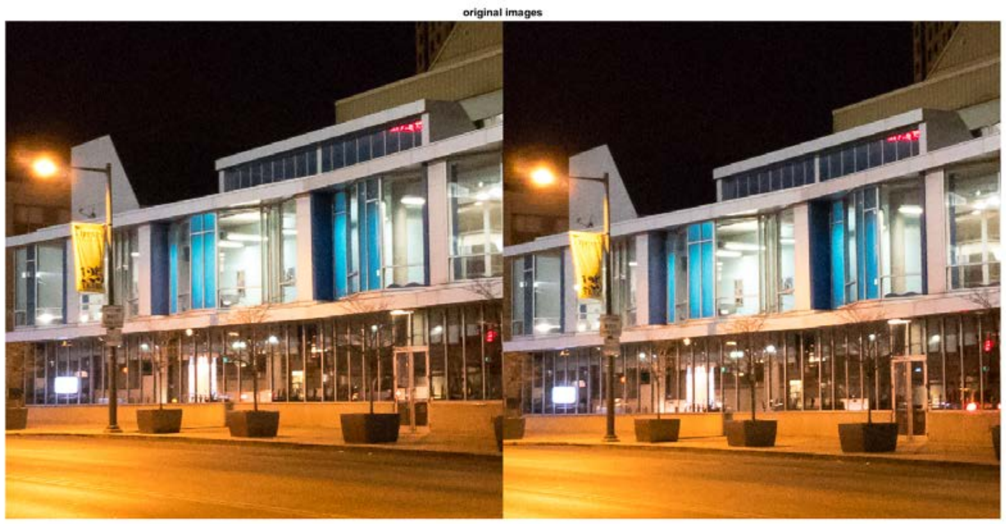
9

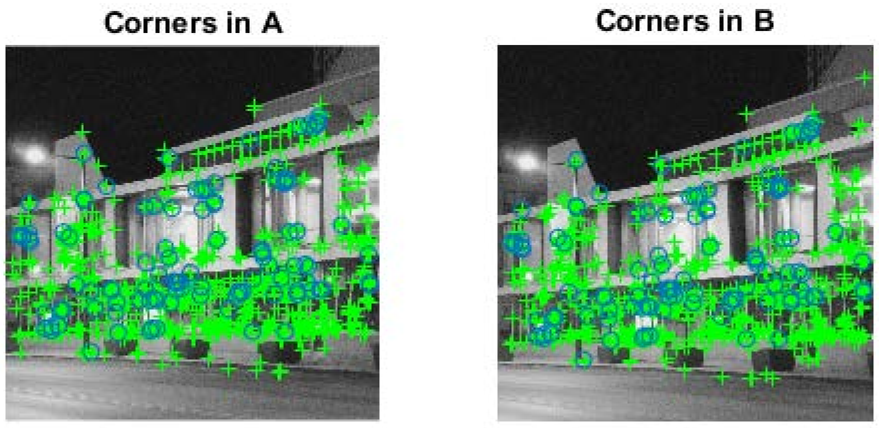


align all

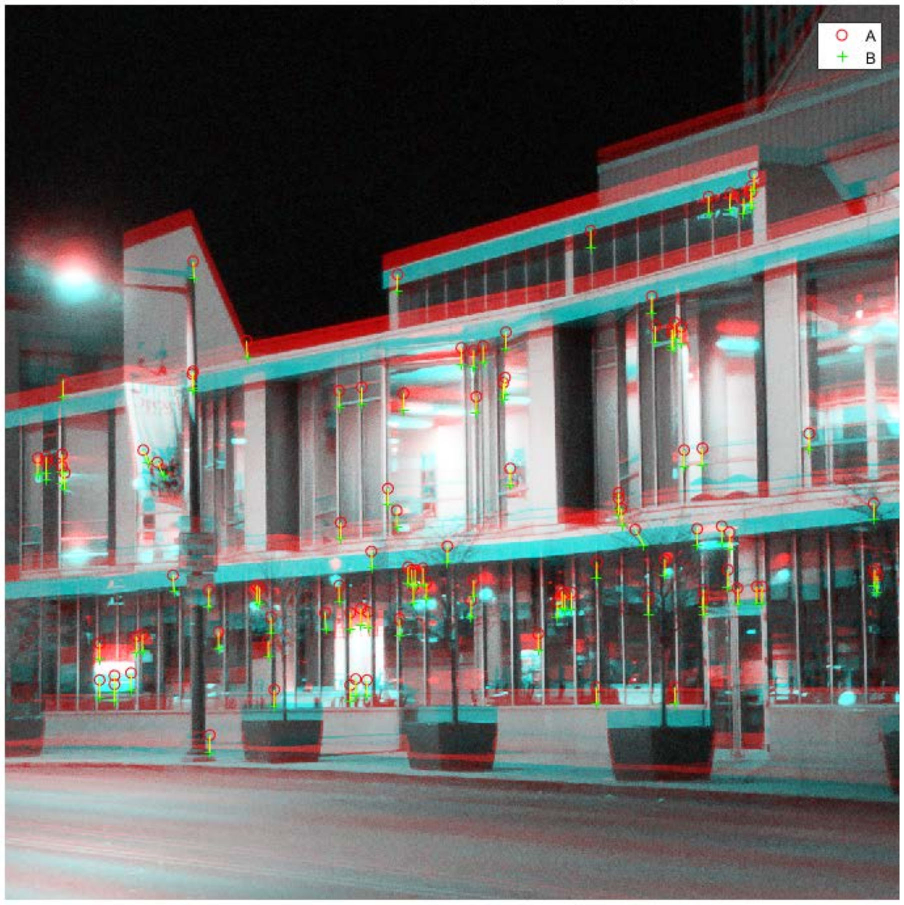
```
close all
imgs_tfed = {};
imgs_cumu = {};
ind_range = [5,6,7,8,9];
imtermediate_plots = true;
count = 1;
img_base = imgs{ind_range(1)};
img_pool = uint32(imresize(img_base,1));
for i = ind_range(2:end)
    count = count + 1;
    if count>4 && imtermediate_plots
        % just print the first few
        imtermediate_plots = false;
        % close all;
    end
    this_img = Prj2TB.align(img_base, imgs{i}, imtermediate_plots);
    imgs_tfed{end+1} = this_img;
    img_pool = img_pool + uint32(this_img);
    imgs_cumu{end+1} = uint16(img_pool / count);
    figure;
    imshow(imgs_cumu{end});
    title(sprintf('img number %i', i))

    %{
    center = flip([509, 780]);
    box = floor([256 256]/2)*2;
    lc = center - box/2 + 1;
    uc = center + box/2;
    this_img = imgs_cumu{end}(lc(1):uc(1), lc(2):uc(2), :);
    filename = sprintf('progressive_%i.PNG',count);
    imwrite(imresize(uint8(this_img ./ 2^8), 8, 'nearest'), filename);
    %}
end
img_out = uint16(img_pool ./ length(ind_range));

%{
this_img = img_base(lc(1):uc(1), lc(2):uc(2), :);
filename = 'progressive_0.PNG';
imwrite(imresize(uint8(this_img ./ 2^8), 8, 'nearest'), filename);
%}
```

```
Warning: Image is too big to fit on screen; displaying at
67%
num of corners matched: 105
num of corners selected: 102
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
num of corners matched: 79
num of corners selected: 74
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
num of corners matched: 76
num of corners selected: 71
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
Warning: Image is too big to fit on screen; displaying at
67%
num of corners matched: 107
num of corners selected: 98
Warning: Image is too big to fit on screen; displaying at
67%
```

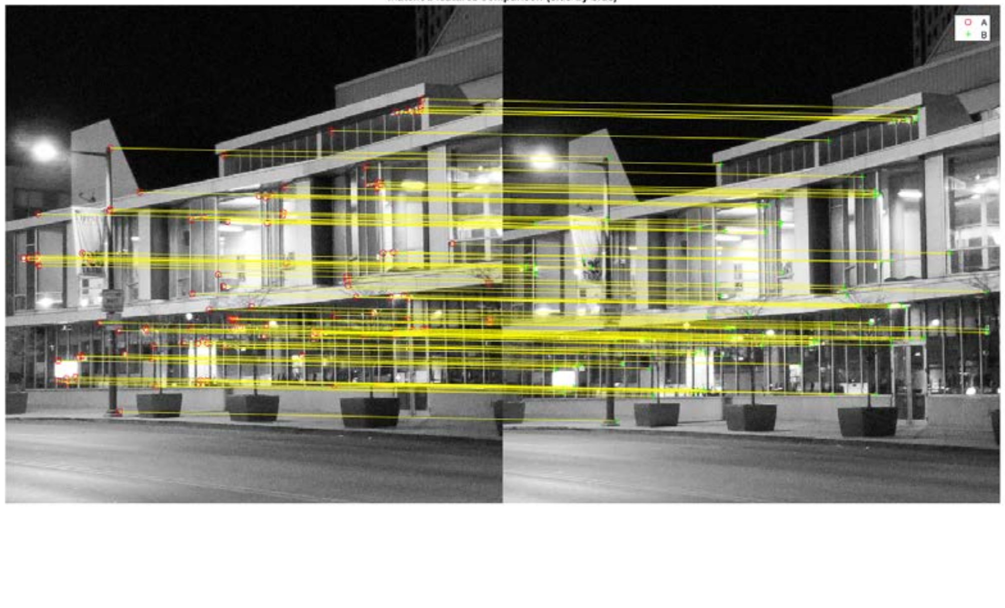


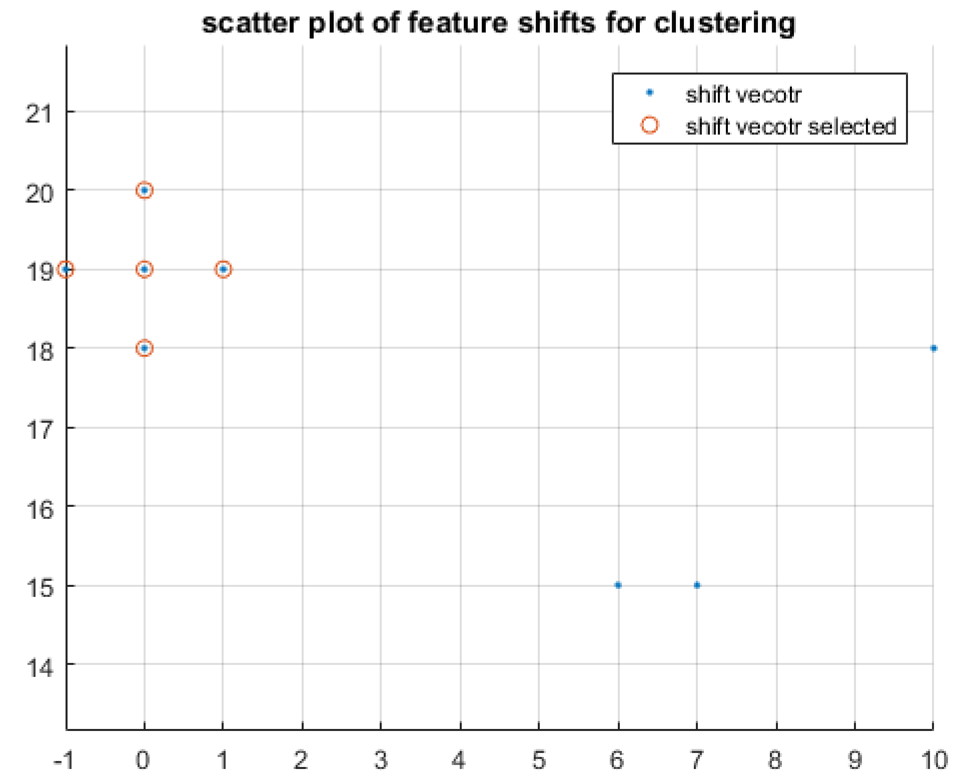


matched features comparison (overlay)



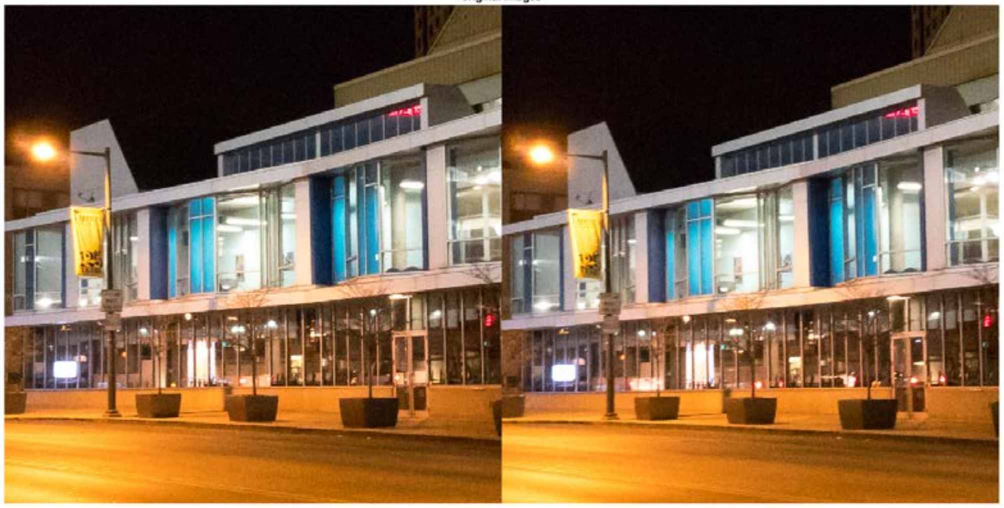
matched features comparison (side-by-side)

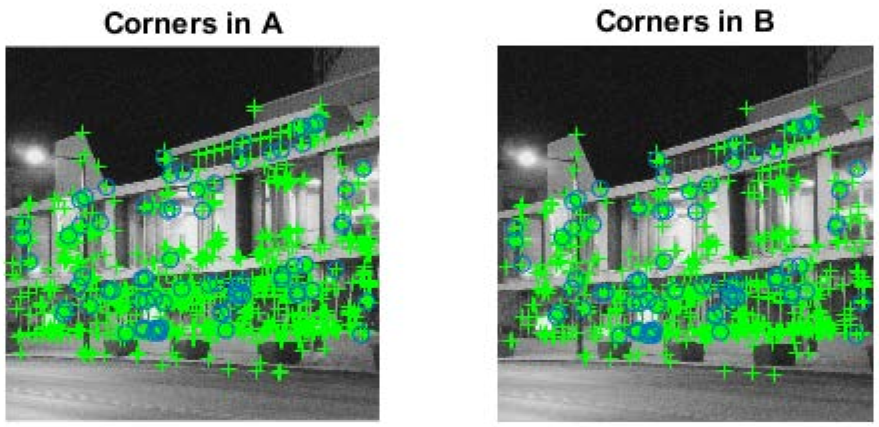



scatter plot of feature shifts for clustering
shift vecotr
shift vecotr selected
21
20
19
18
17
16
15
14
-1
0
1
2
3
4
5
6
7
8
9
10

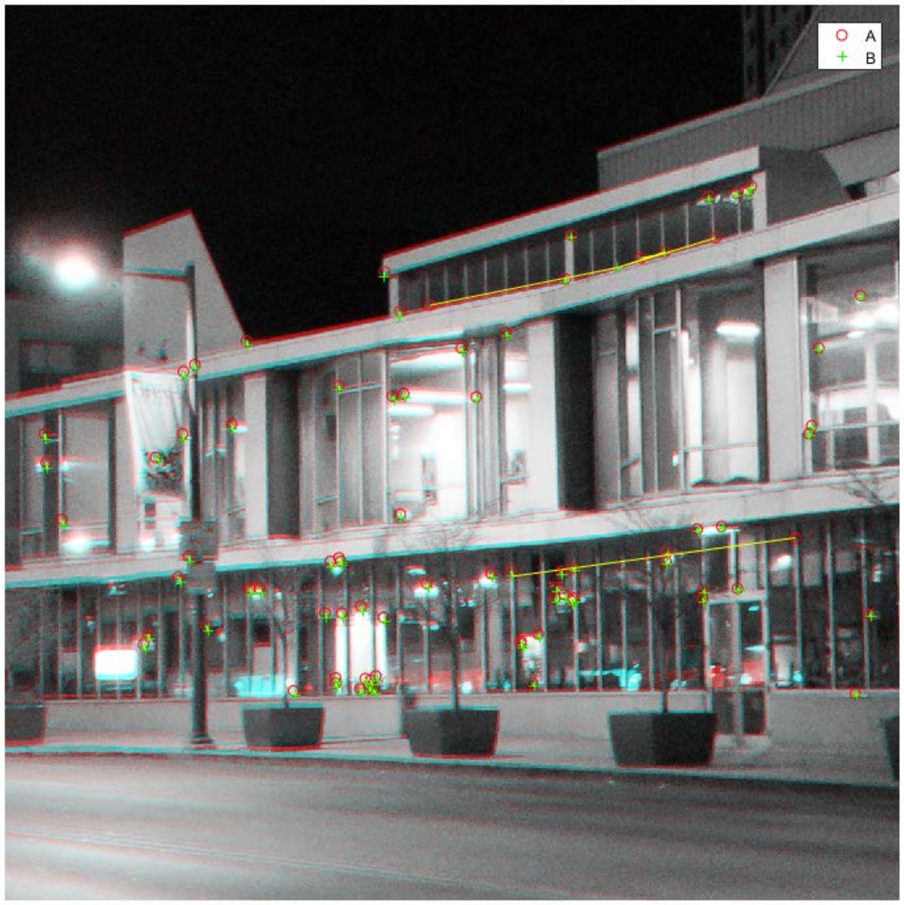
img number 6

original images

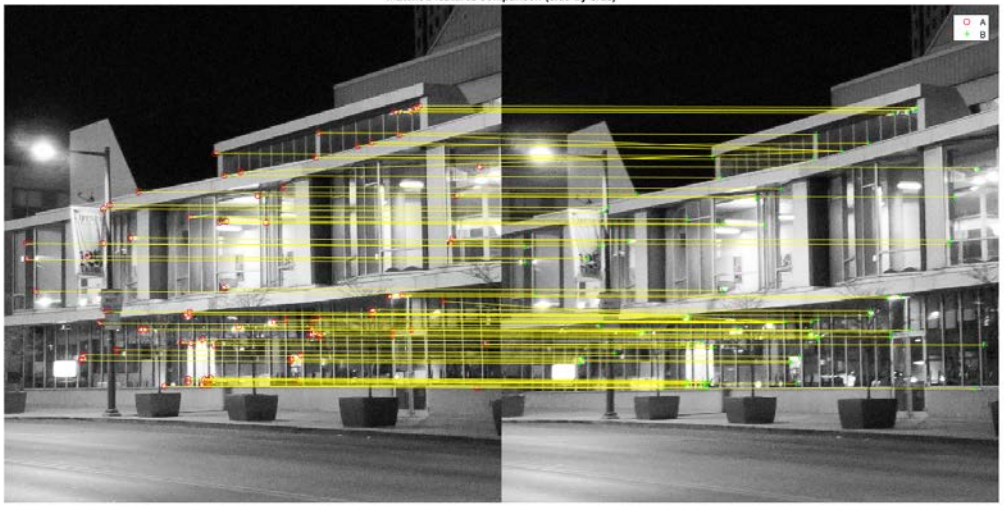


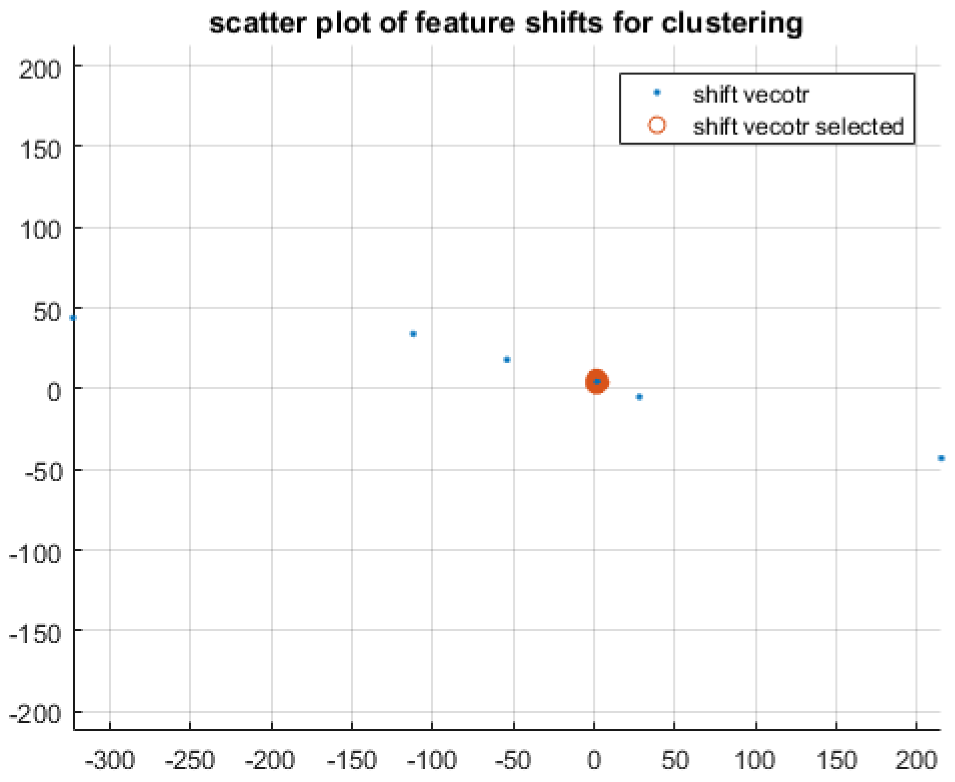

Corners in A
Corners in B

matched features comparison (overlay)

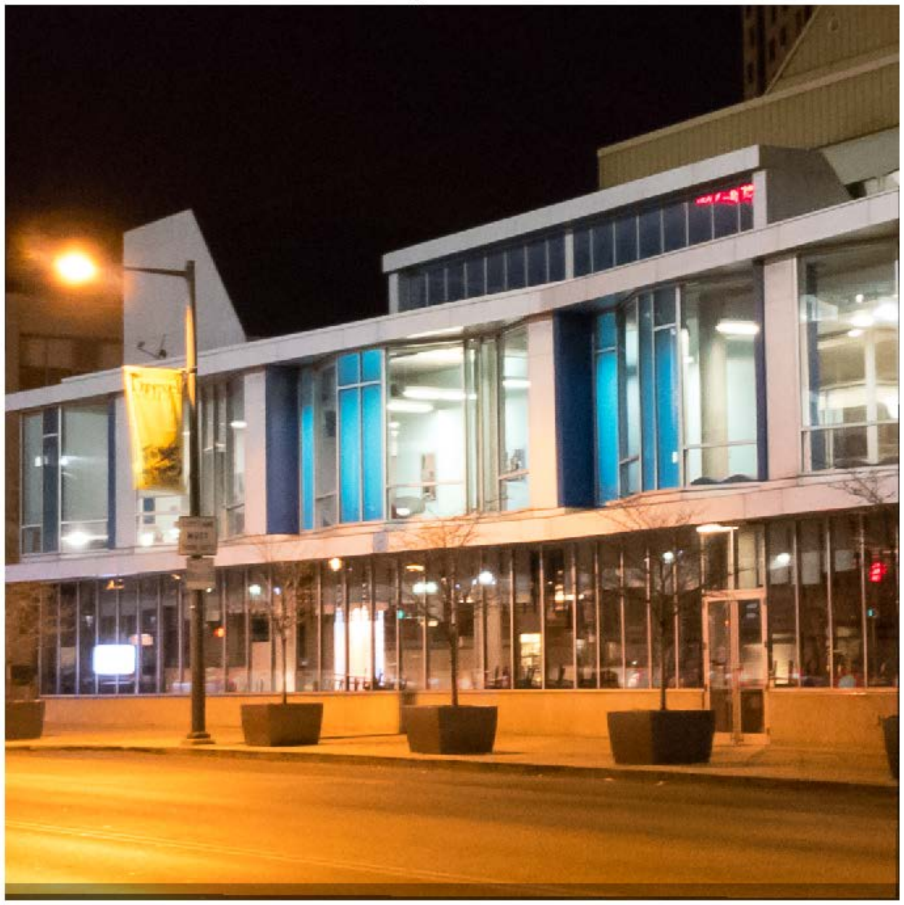


matched features comparison (side-by-side)

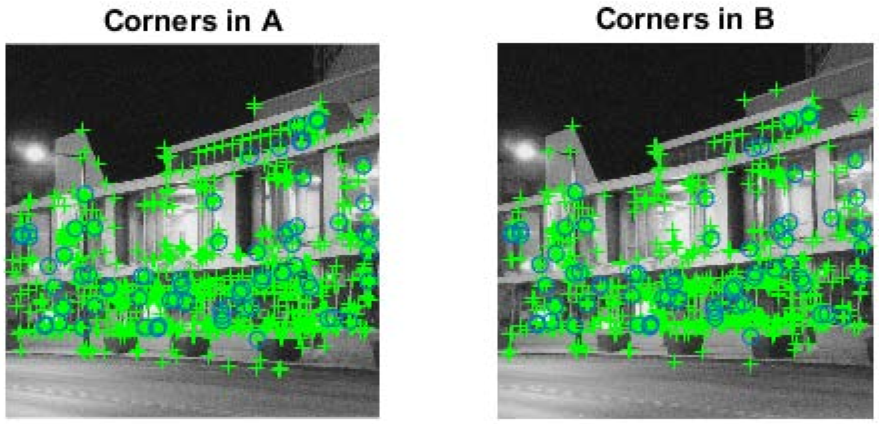


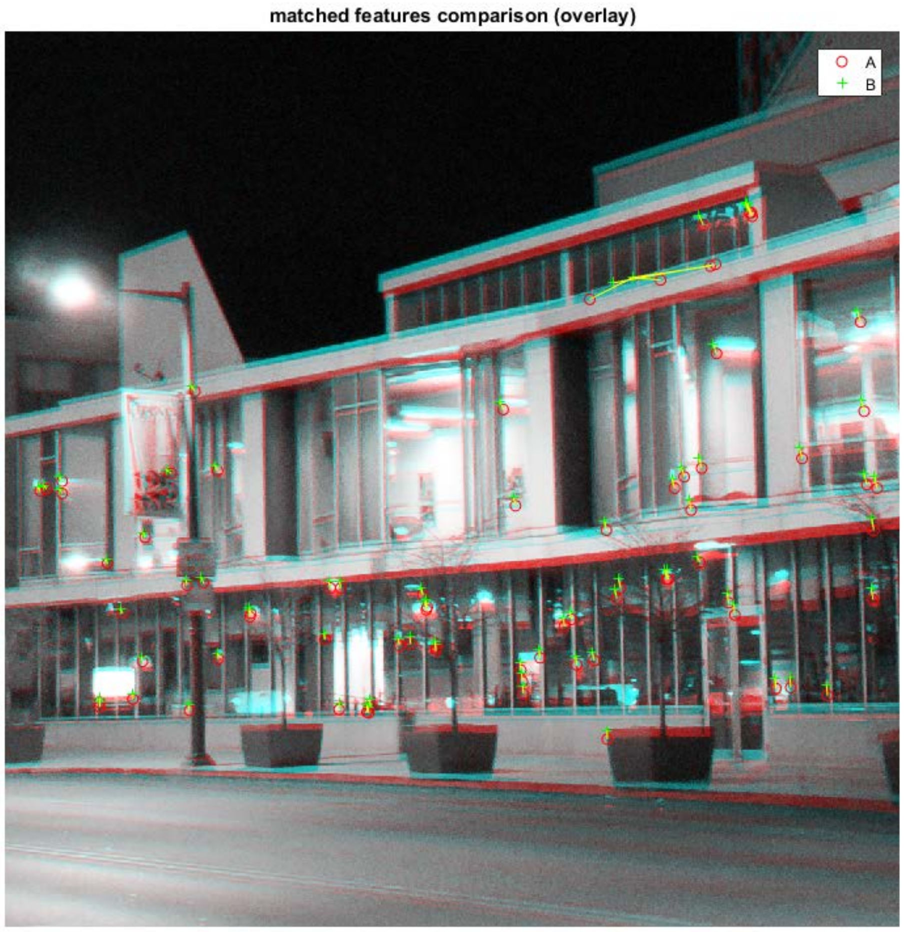

scatter plot of feature shifts for clustering
shift vecotr
shift vecotr selected
200
150
100
50
0
-50
-100
-150
-200
-300
-250
-200
-150
-100
-50
0
50
100
150
200

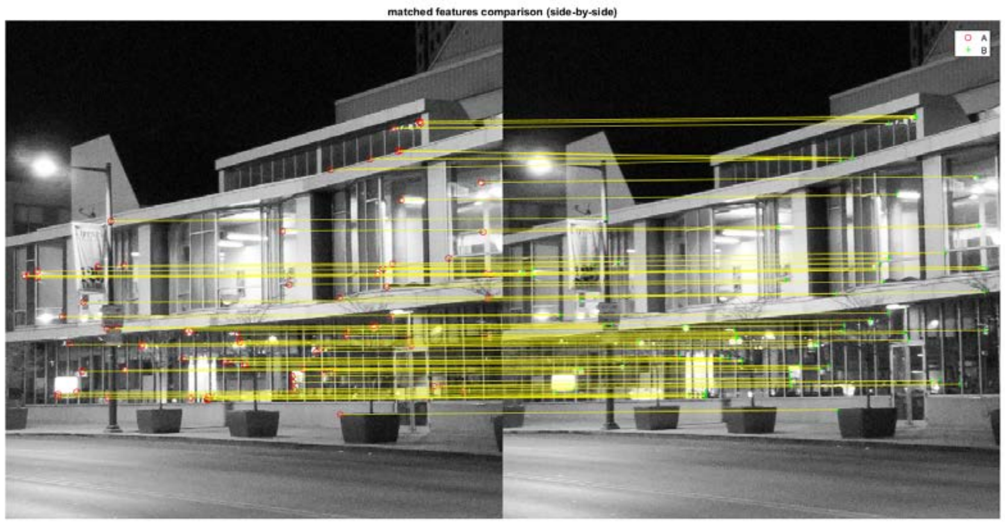
img number 7

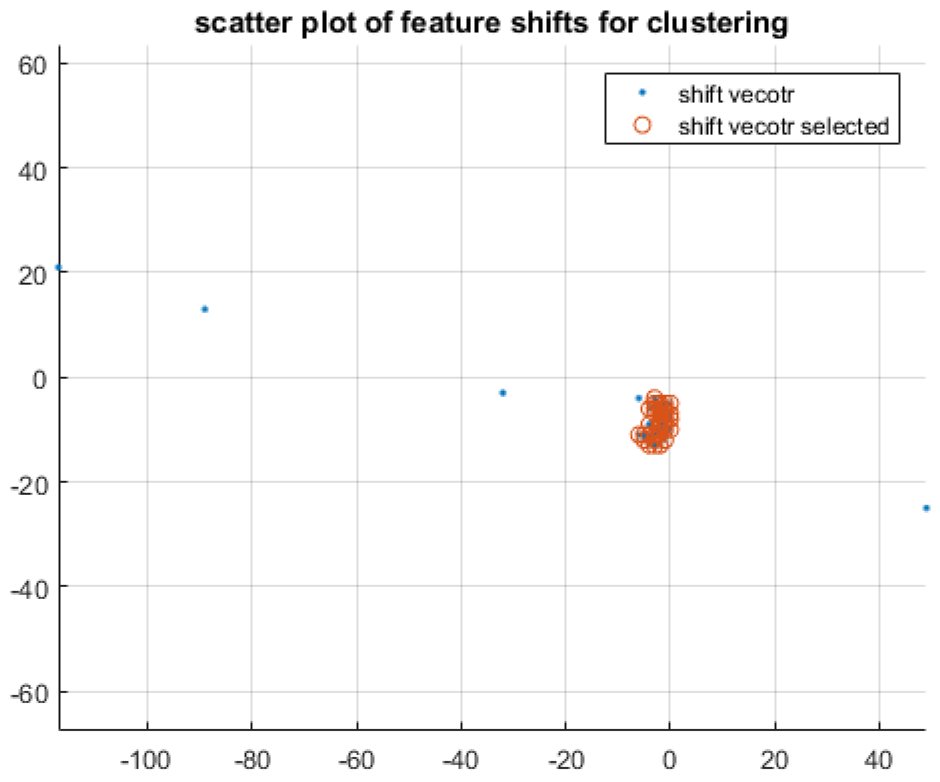


original images


Corners in A
Corners in B


matched features comparison (overlay)
A
B


matched features comparison (side-by-side)
A
B


scatter plot of feature shifts for clustering
shift vecotr
shift vecotr selected
60
40
20
0
-20
-40
-60
-100
-80
-60
-40
-20
0
20
40

img number 8

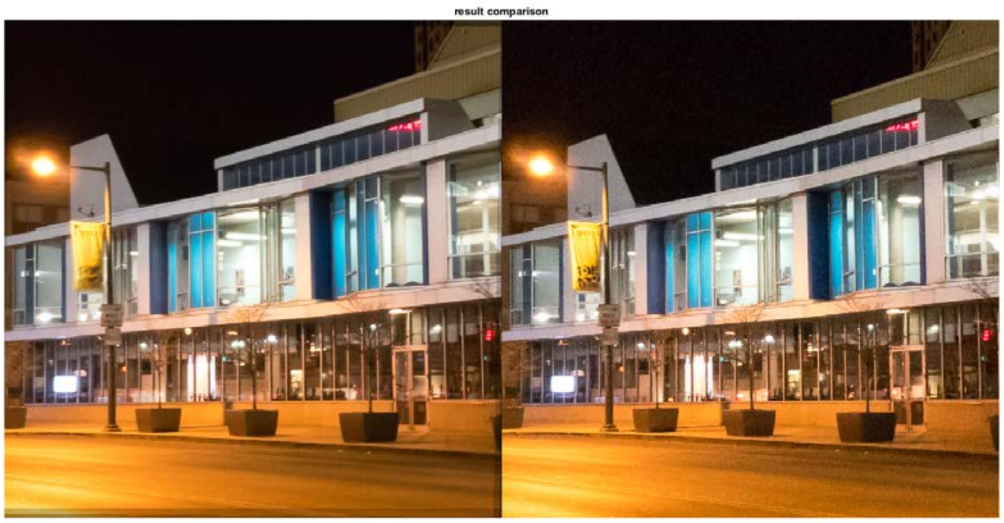
img number 9

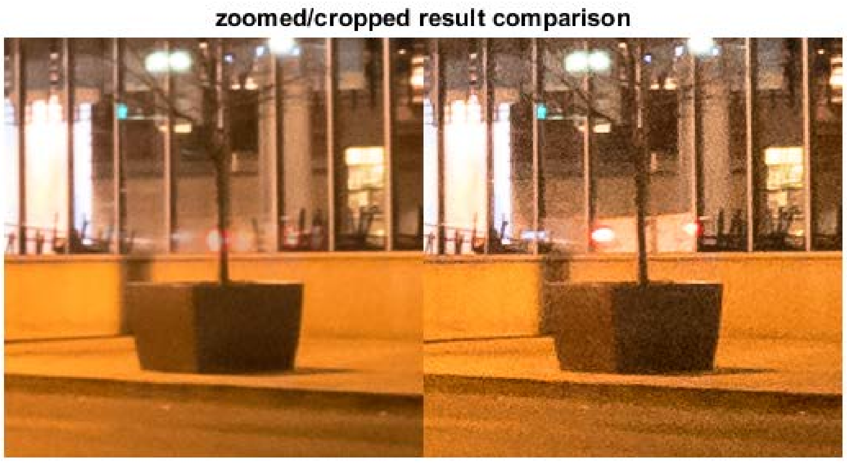
show and compare

```
close all

center = flip([509, 780]);
box = floor([256 256]/2)*2;
lc = center - box/2 + 1;
uc = center + box/2;

figure;
imshowpair(img_out, img_base, 'montage');
title('result comparison');

figure;
imshowpair( ...
    img_out(lc(1):uc(1), lc(2):uc(2), :), ...
    img_base(lc(1):uc(1), lc(2):uc(2), :), 'montage');
title('zoomed/cropped result comparison');
```

```
Warning: Image is too big to fit on screen; displaying at
```

```
67%
```





show Prj2TB file

```
type('Prj2TB.m');

close all


classdef Prj2TB
    % Functions used for ECES 435 Project 2
    %   Detailed explanation goes here

    properties
    end
```

```
methods
    function obj = Prj2TB(self)
        % imgs = Prj2TB.read_all_jpgs('raw_img/FV5', '', 'TIF');
        % imgs_cropped = Prj2TB.crop_images(imgs);

    end
end

methods (Static)

    function img_out = read_all_imgs(dir_path, prefix, ext)
        dir_s = dir(dir_path);
        img_out = {};
        regexp_pat = sprintf('%s\.%s$',prefix,ext);
        for i = dir_s'
            hit = regexpi(i.name, regexp_pat);
            if ~isempty(hit)
                try
                    path = fullfile(dir_path, i.name);
                    img_out{end+1} = imread(path);
                    fprintf('loaded: %s\n', path);
                catch
                    fprintf('*** ERROR loading: %s\n', path);
                end
            end
        end
    end

    function img_out = crop_images(img_in)
        img_out = {};
        for i = img_in
            img = i{1};
            img_size = size(img);
            center = floor(img_size(1:2)/2);
            box = floor([1024 1024]/2)*2;
            lc = center - box/2 + 1;
            uc = center + box/2;
            img_out{end+1} = img(lc(1):uc(1), lc(2):uc(2), :);

        end
    end

    function [] = save_images(img_in, dir, prefix)
        len = length(img_in)
        for i = 1:len
            filename = sprintf('%s_%03i.TIF', prefix, i);
            filepath = fullfile(dir, filename)
            imwrite(img_in{i}, filepath);
        end
    end

    function img_aligned = align(imgA_raw, imgB_raw, sw_plot)
        ptThresh = 0.15;
```

```
            if sw_plot
                figure;
                imshowpair(imgA_raw, imgB_raw, 'montage');
                title('original images');
            end

            % super sampling mutiplier
            SS = 1;
            % select Y channel
            imgA = imresize(rgb2ycbcr(imgA_raw), SS);
            imgB = imresize(rgb2ycbcr(imgB_raw), SS);
            imgA = imgA(:,:,1);
            imgB = imgB(:,:,1);

            pointsA = ...
                detectFASTFeatures(imgA, 'MinContrast', ptThresh/
SS^.5);
            pointsB = ...
                detectFASTFeatures(imgB, 'MinContrast', ptThresh/
SS^.5);

            %{
            if sw_plot
                figure;
                subplot(1,2,1); imshow(imgA); hold on;
                plot(pointsA); hold off;
                title('Corners in A');
                subplot(1,2,2); imshow(imgB); hold on;
                plot(pointsB); hold off;
                title('Corners in B');
            end
            %}

            [featuresA, pointsA] = ...
                extractFeatures(imgA, pointsA, 'BlockSize', 1+10*SS);
            [featuresB, pointsB] = ...
                extractFeatures(imgB, pointsB, 'BlockSize', 1+10*SS);
            indexPairs = ...
                matchFeatures(featuresA, featuresB, ...
                'MatchThreshold', SS*10);
            pointsA_matched = pointsA(indexPairs(:, 1), :);
            pointsB_matched = pointsB(indexPairs(:, 2), :);

            shift = pointsB_matched.Location -
 pointsA_matched.Location;

            % prepare for DBSCAN
            % find 2 norm (distance to origin)
            % dto = sqrt( sum( (shift.^2)' )' );
            % epsilon = median(dto);
            epsilon = 2 * SS^2;
            min_pts = ceil(size(shift, 1) * 0.05);
            idx = DBSCAN(shift, epsilon, min_pts);
            shift_sel = shift(idx==1, :);
            pointsA_sel = pointsA_matched(idx == 1);
            pointsB_sel = pointsB_matched(idx == 1);

            fprintf('num of corners matched: %d\n',length(shift));
            fprintf('num of corners selected: %d\n',sum(idx==1));


            if sw_plot
                figure;
                subplot(1,2,1); imshow(imgA); hold on;
                plot(pointsA);
                plot(pointsA_matched.Location(:,1), ...
                    pointsA_matched.Location(:,2), 'o');
                hold off; title('Corners in A');
                subplot(1,2,2); imshow(imgB); hold on;
                plot(pointsB);
                plot(pointsB_matched.Location(:,1), ...
                    pointsB_matched.Location(:,2), 'o');
                hold off; title('Corners in B');


                figure;
                showMatchedFeatures(imgA, imgB, ...
                    pointsA_matched, pointsB_matched);
                legend('A', 'B');
                title('matched features comparison (overlay)')

                figure;
                showMatchedFeatures(imgA, imgB, ...
                    pointsA_matched, pointsB_matched, 'montage');
                legend('A', 'B');
                title('matched features comparison (side-by-side)')

                figure; hold on;
                plot(shift(:,1),shift(:,2), '.')
                plot(shift_sel(:,1),shift_sel(:,2), 'o')
                grid on; hold off; axis equal
                title('scatter plot of feature shifts for clustering')
                legend('shift vecotr','shift vecotr selected');
            end

            [tform, pointsBm, pointsAm] =
  estimateGeometricTransform(...
                pointsB_sel, pointsA_sel, 'affine');
            img_aligned = imwarp(imresize(imgB_raw,SS), ...
                tform, 'OutputView', imref2d(size(imgB)));

        end


    end

end
```

Published with MATLAB® R2016a