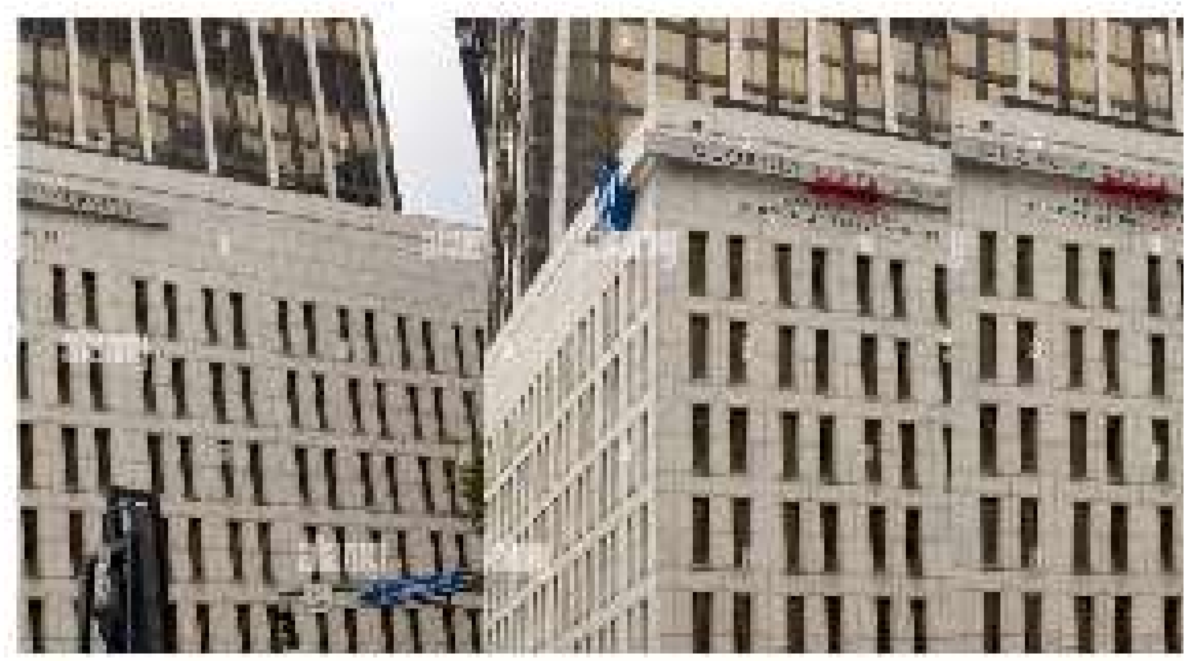
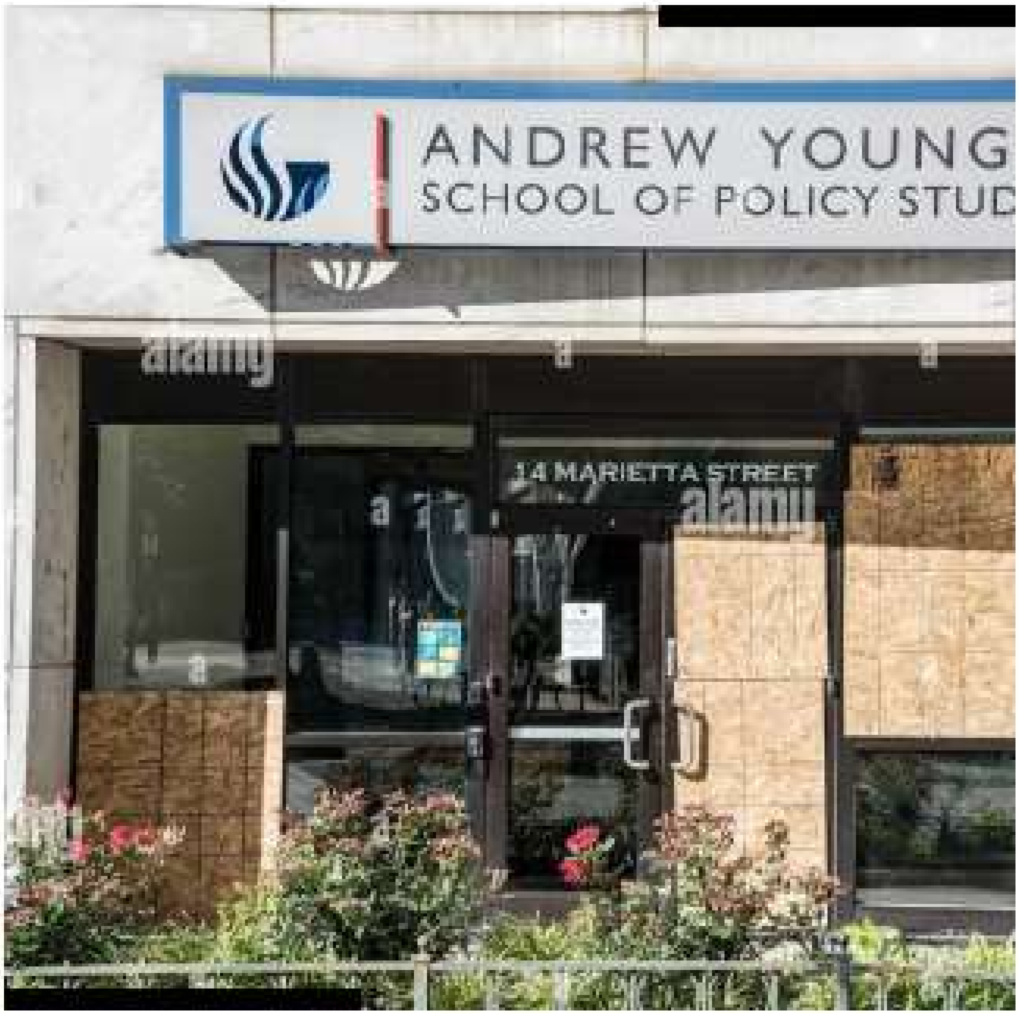
```
doimagestitching('C:\Users\aswin\25pp_images')
```

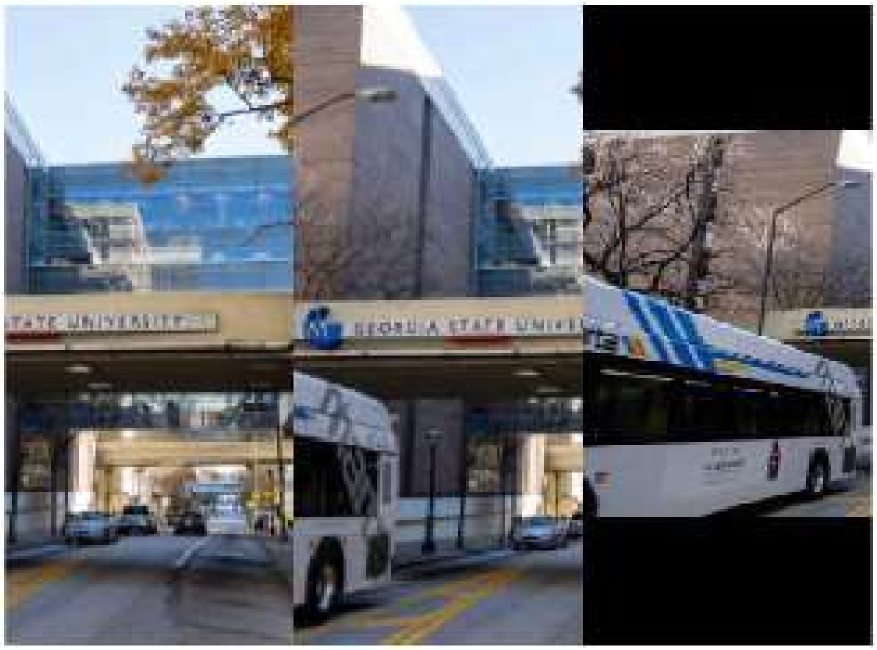


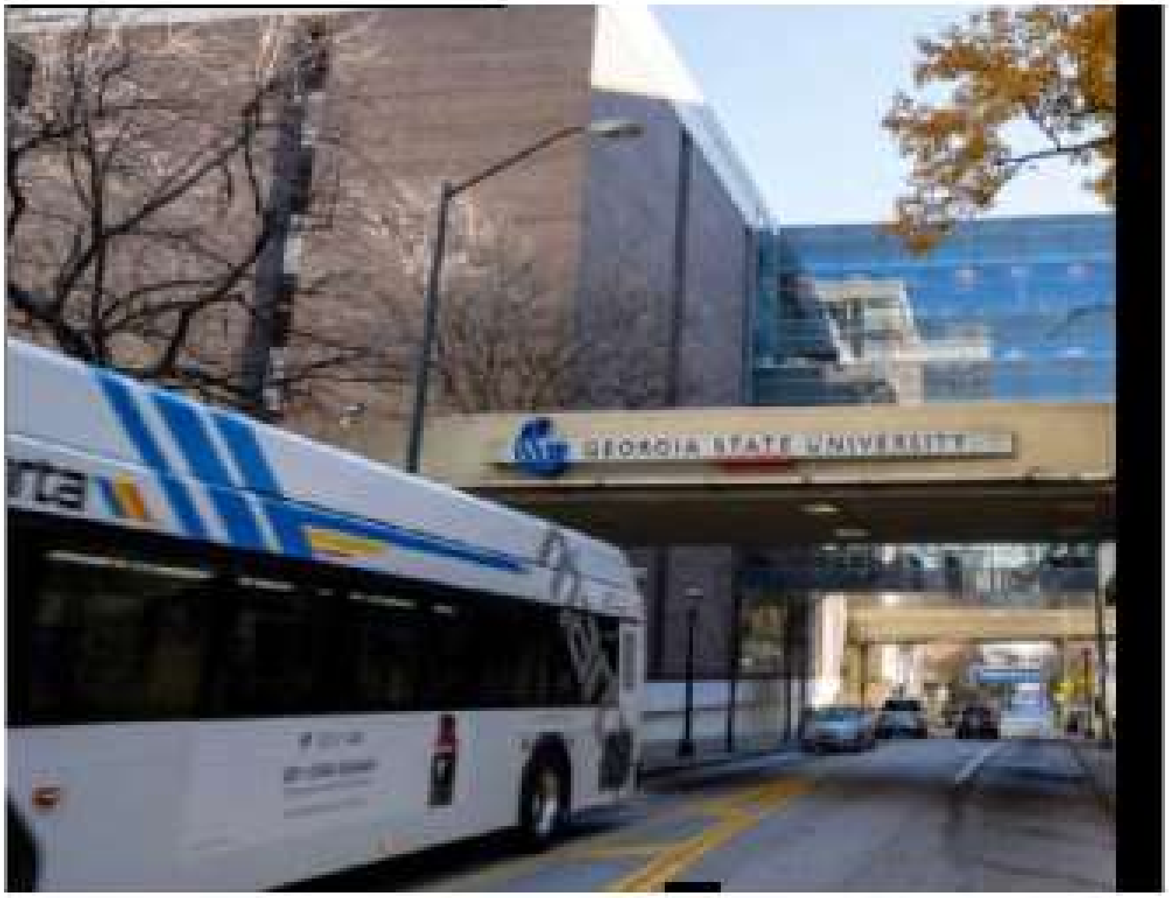
```
doimagestitching('C:\Users\aswin\55pp_images')
```

OUNG
ICY STUDIES
ANDREW YOUNG
SCHOOL OF POLICY STUD



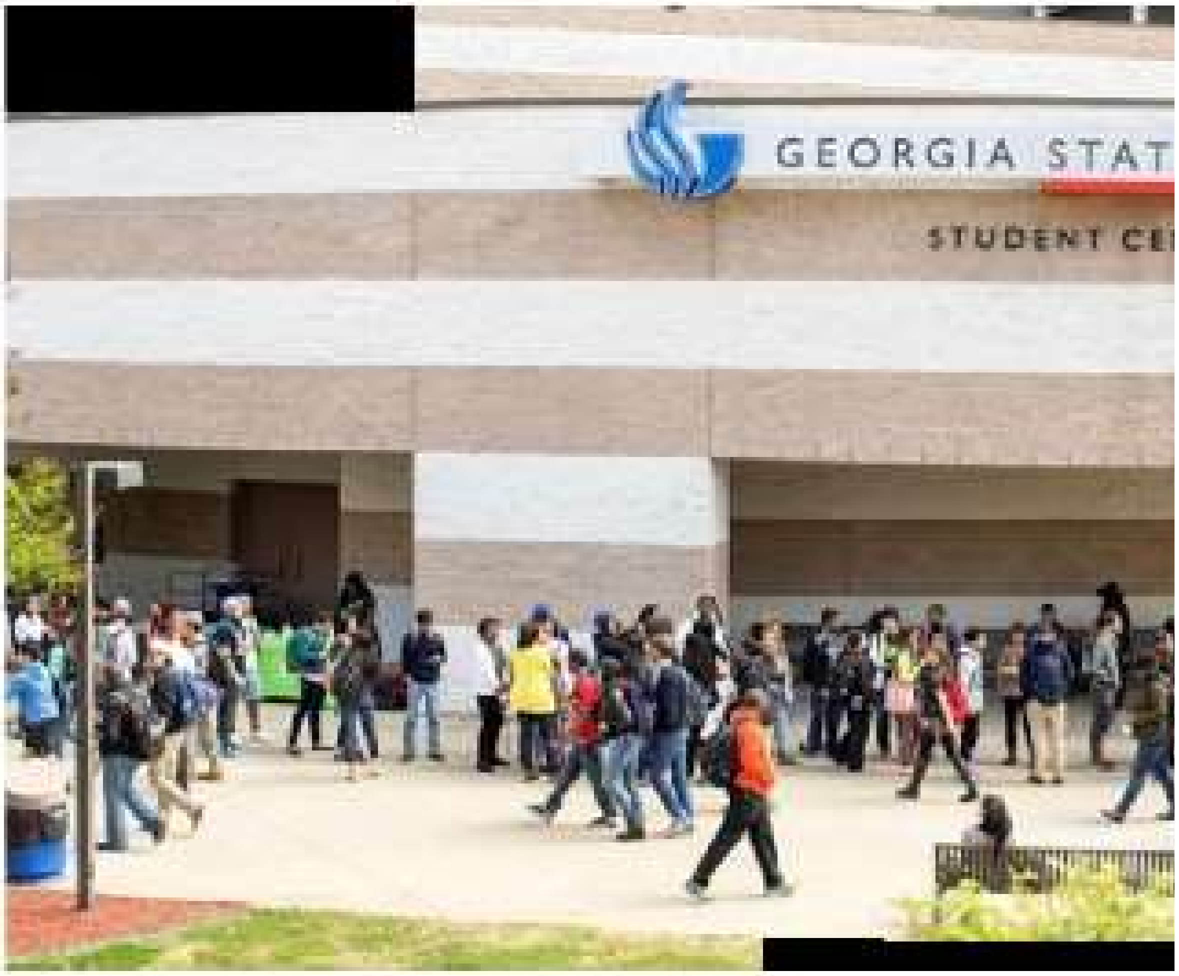
```
doimagestitching('C:\Users\aswin\aderhold_images')
```



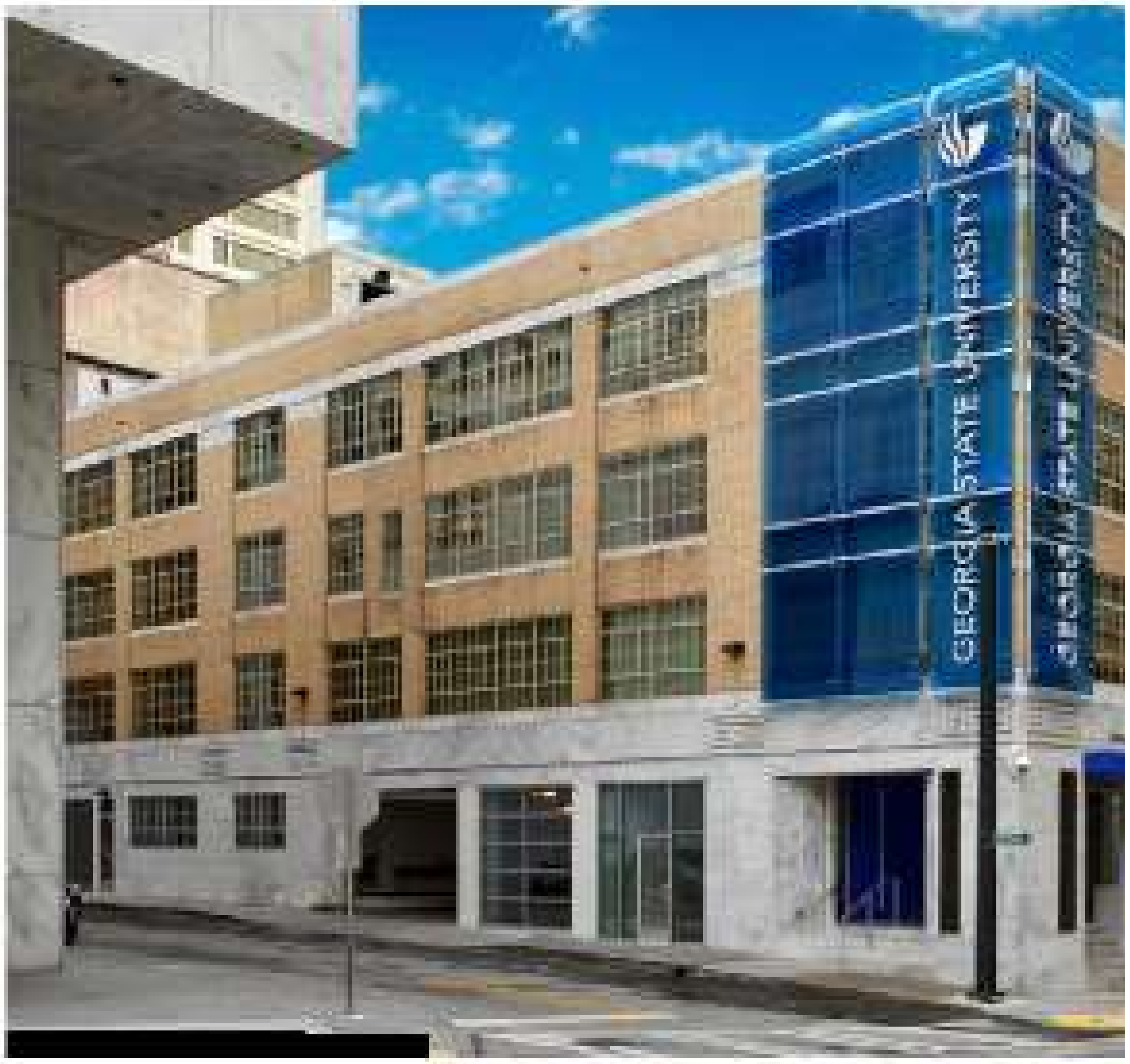


```
doimagestitching('C:\Users\aswin\classsouth_images')
```

RGIA STATE UNIVERSITY
STUDENT CENTER
GEORGIA STATE
STUDENT CENTE



```
doimagestitching('C:\Users\aswin\mdeck_images')
```



```
function doimagestitching(foldername)
buildingDir = fullfile(foldername);
buildingScene = imageDatastore(buildingDir);


montage(buildingScene.Files)

% Read the first image from the image set.
I = readimage(buildingScene,1);


grayImage = im2gray(I);
points = detectSURFFeatures(grayImage);
[features, points] = extractFeatures(grayImage,points);

numImages = numel(buildingScene.Files);
tforms(numImages) = projective2d(eye(3));
```

```
imageSize = zeros(numImages,2);

% Iterate over remaining image pairs
for n = 2:numImages


    pointsPrevious = points;
    featuresPrevious = features;


    I = readimage(buildingScene, n);

    % Convert image to grayscale.
    grayImage = im2gray(I);


    imageSize(n,:) = size(grayImage);

    % Detect and extract SURF features for I(n).
    points = detectSURFFeatures(grayImage);
    [features, points] = extractFeatures(grayImage, points);

    % Find correspondences between I(n) and I(n-1).
    indexPairs = matchFeatures(features, featuresPrevious, 'Unique', true);

    matchedPoints = points(indexPairs(:,1), :);
    matchedPointsPrev = pointsPrevious(indexPairs(:,2), :);


    tforms(n) = estimateGeometricTransform2D(matchedPoints, matchedPointsPrev,...
        'projective', 'Confidence', 99.9, 'MaxNumTrials', 5000);


    tforms(n).T = tforms(n).T * tforms(n-1).T;
end

for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1 imageSize(i,1)]);
end

avgXLim = mean(xlim, 2);
[~,idx] = sort(avgXLim);
centerIdx = floor((numel(tforms)+1)/2);
centerImageIdx = idx(centerIdx);

Tinv = invert(tforms(centerImageIdx));
for i = 1:numel(tforms)
    tforms(i).T = tforms(i).T * Tinv.T;
end

for i = 1:numel(tforms)
    [xlim(i,:), ylim(i,:)] = outputLimits(tforms(i), [1 imageSize(i,2)], [1 imageSize(i,1)]);
end

maxImageSize = max(imageSize);


xMin = min([1; xlim(:)]);
```

```
xMax = max([maxImageSize(2); xlim(:)]);

yMin = min([1; ylim(:)]);
yMax = max([maxImageSize(1); ylim(:)]);

% Width and height of panorama.
width  = round(xMax - xMin);
height = round(yMax - yMin);

% Initialize the "empty" panorama.
panorama = zeros([height width 3], 'like', I);

blender = vision.AlphaBlender('Operation', 'Binary mask', ...
    'MaskSource', 'Input port');


xLimits = [xMin xMax];
yLimits = [yMin yMax];
panoramaView = imref2d([height width], xLimits, yLimits);

% Create the panorama.
for i = 1:numImages

    I = readimage(buildingScene, i);

    % Transform I into the panorama.
    warpedImage = imwarp(I, tforms(i), 'OutputView', panoramaView);


    mask = imwarp(true(size(I,1),size(I,2)), tforms(i), 'OutputView', panoramaView);

    % Overlay the warpedImage onto the panorama.
    panorama = step(blender, panorama, warpedImage, mask);
end

figure
imshow(panorama)

end
```