```
clear

fnames = ["cmii_team6_1"
"librarynorth"
'rialto_team'
'TDeck_team06'
'sportsarena'
'sciencecenter'
]
```

```
fnames = 6×1 string
"cmii_team…
"libraryno…
"rialto_team"
"TDeck_tea…
"sportsarena"
"sciencece…
```

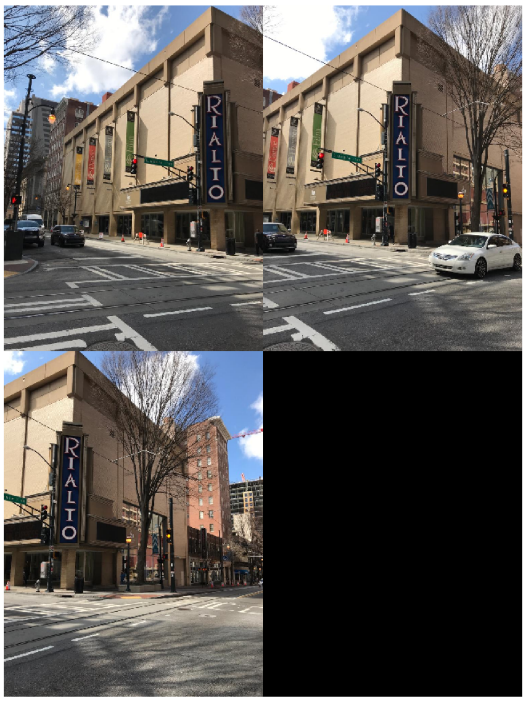
```
%failed
%"artandhumanities"
%'urbanlife'
%'studentcenterwest'
%'studentcentereast'
%'langdale'
%'lawschoolbuilding'

cd("C:\Users\wsash\Google Drive\computervision\homework2\images")
D = dir; % A is a struct ... first elements are '.' and '..' used for navigation.
for m = 1:length(fnames())
    for k = 3:length(D) % avoid using the first ones
        %currD = D(k).name; % Get the current subdirectory name
        %Run your function. Note, I am not sure on how your function is written,
        %but you may need some of the following
        %cd(currD) % change the directory (then cd('..') to get back)
        %fList = dir(currD); % Get the file list in the subdirectory
        %disp(currD)
        %disp(D(k))
        %disp(strcat(D(k).folder,'\',D(k).name))
        %contains = any(fnames(:) == D(k).name)
        %disp(strcat(strcmp(fnames(m) , D(k).name),fnames(m),D(k).name))
        if strcmp(fnames(m) , D(k).name)
            disp(fnames(m))
            disp(D(k).name)
            disp(strcmp(fnames(m) , D(k).name))
            stitchGsu(strcat(D(k).folder,'\',D(k).name))
        end
    end
end
```

```
cmii_team6_1
cmii_team6_1
   1
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\cmii_team6_1
buildingDir =
'C:\Users\wsash\Google Drive\computervision\homework2\images\cmii_team6_1'
```

```
librarynorth
librarynorth
   1
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\librarynorth
buildingDir =
'C:\Users\wsash\Google Drive\computervision\homework2\images\librarynorth'
```

```
rialto_team
rialto_team
   1
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\rialto_team
buildingDir =
```

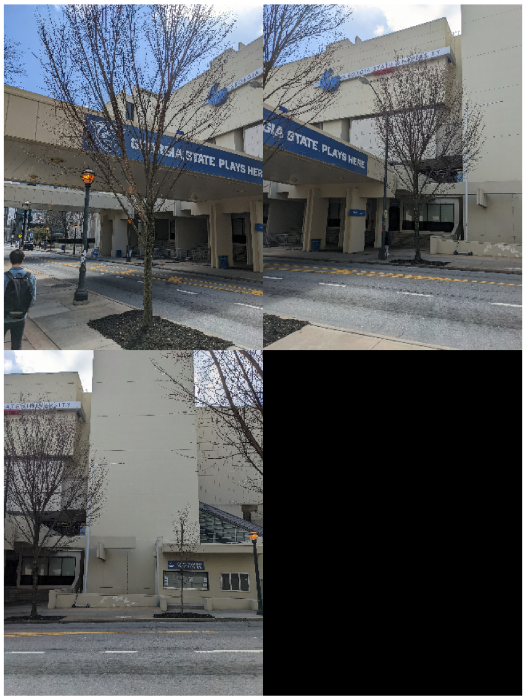
```
'C:\Users\wsash\Google Drive\computervision\homework2\images\rialto_team'
```




```
TDeck_team06
TDeck_team06
   1
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\TDeck_team06
buildingDir =
'C:\Users\wsash\Google Drive\computervision\homework2\images\TDeck_team06'
```

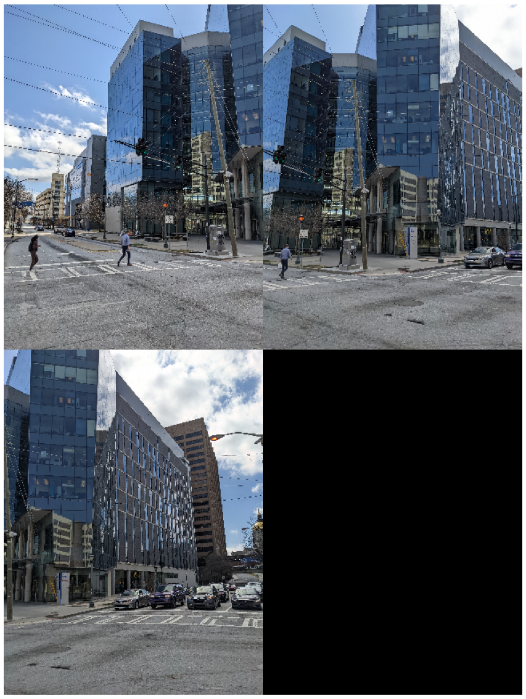
sportsarena
sportsarena
1

```
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\sportsarena
buildingDir =
'C:\Users\wsash\Google Drive\computervision\homework2\images\sportsarena'
```



```
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the
desired confidence.
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the
desired confidence.
```


```
sciencecenter
sciencecenter
   1
begin_C:\Users\wsash\Google Drive\computervision\homework2\images\sciencecenter
buildingDir =
'C:\Users\wsash\Google Drive\computervision\homework2\images\sciencecenter'
```



```
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the
desired confidence.
Warning: Maximum number of trials reached. Consider increasing the maximum distance or decreasing the
desired confidence.
```

```
function stitchGsu(fold)
    disp(strcat('begin_',fold))
    buildingDir = fullfile(fold)

    buildingScene = imageDatastore(buildingDir);

    figure
    montage(buildingScene.Files)

    I = readimage(buildingScene,1);

    %imshow(I)

    % Initialize features for I(1)
    grayImage = im2gray(I);
```

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

% Initialize all the transforms to the identity matrix. Note that the
% projective transform is used here because the building images are fairly
% close to the camera. Had the scene been captured from a further distance,
% an affine transform would suffice.
numImages = numel(buildingScene.Files);
%disp('numImages'+numImages)
tforms(numImages) = projective2d(eye(3));

% Initialize variable to hold image sizes.
imageSize = zeros(numImages,2);

for n = 2:numImages

    % Store points and features for I(n-1).
    pointsPrevious = points;
    featuresPrevious = features;

    % Read I(n).
    I = readimage(buildingScene, n);

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

    % Save image size.
    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), :);

    % Estimate the transformation between I(n) and I(n-1).
    tforms(n) = estimateGeometricTransform2D(matchedPoints, matchedPointsPrev,...
        'projective', 'Confidence', 99.9, 'MaxNumTrials', 2000);

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

% Compute the output limits for each transform.
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);

    % Find the minimum and maximum output limits.
    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');

    % Create a 2-D spatial reference object defining the size of the panorama.
    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);

        % Generate a binary mask.
        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
```

```
function hello()
    disp('hello')
end
```