Colorado School of Mines

Computer Vision

Professor William Hoff
Dept of Electrical Engineering & Computer Science
http://inside.mines.edu/~whoff/
Image Stitching
Image Stitching

• Take a collection of images, align them, and stitch them into a seamless photomosaic

• Steps:
  – Find the transforms relating pairs of images
    • E.g., homography transforms
  – Choose a compositing surface to display the mosaic on
    • E.g., a plane or cylinder
  – Align the images and blend them
    • Remove unwanted objects (such as moving objects)
    • Compensate for exposure differences
    • Smooth the seams between images
Figure 9.11 Recognizing panoramas (Brown, Szeliski, and Winder 2005), figures courtesy of Matthew Brown:
(a) input images with pairwise matches; (b) images grouped into connected components (panoramas); (c) individual panoramas registered and blended into stitched composites.
Simple Stitching Algorithm

• Assume:
  – Input is a sequence of images
  – Each image overlaps with the ones adjacent to it in the sequence
• Use a plane for the compositing surface
• Each subsequent image is aligned to the previous image, using a homography
• Warp the images to a reference frame, and blend it with the reference image by simple averaging
• The first image defines the reference frame
Combining Homographies

$\frac{1}{2}H$ is the homography from image 2 to image 1 (in the code we write it as H_{2_1})

This homography maps points from image 2 to the coordinate frame of image 1: $^1p = \frac{1}{2}H \cdot ^2p$

We can combine homographies by multiplying:

$\frac{1}{3}H = \frac{1}{2}H \cdot \frac{2}{3}H$
Pseudocode

Get the first image -> Iref  The reference image is initialized to the first image
Iprev = Iref  Initialize the “previous” image for the loop below
H_prev_ref = identity  The homography from previous to reference frame is the identity matrix

Repeat until no more images
  Get the next image -> Inext  Get the next image in the sequence
  Compute H_next_prev  Compute the homography from the next image to the previous
  H_next_ref = H_prev_ref * H_next_prev  Get homography from next image to reference
  Warp Inext using H_next_ref, to get Inext_ref  Align next image to reference frame
  Iref = blend(Iref, Inext_ref)  Blend next image with reference image
  Iprev = Inext  The next image becomes the previous image, for the next iteration
  H_prev_ref = H_next_ref  Update homography from previous image to reference
end
Keeping Track of Coordinate Frames

• When we warp an image, its coordinate frame changes.

• We need to keep track of the actual bounds of the new image, in the original’s coordinate frame.

\[
\begin{align*}
[\text{I}_\text{new}, \text{imref2d}] &= \text{imwarp}(\text{I}, \text{tform});
\end{align*}
\]

“imwarp” returns a “imref2d” structure that contains the bounds of the output image.
clear variables
close all
rng(0); % Reset random number generator

MAXWIDTH = 640;

% Get list of images to process.
fileNames = { ...
    'carpet/image1.jpg', 'carpet/image2.jpg', 'carpet/image3.jpg', ...
    'carpet/image4.jpg', ...
};

% Get the first image. It will be the "previous" image in each iteration of
% the loop below.
Iprev = imread(fileNames{1});

% Reduce size if image is too big.
if size(Iprev,2) > MAXWIDTH
    Iprev = imresize(Iprev, MAXWIDTH/size(Iprev,2));
end

Iref = Iprev; % Initialize our "reference" image
Rref = imref2d(size(Iref));
H_prev_ref = eye(3,3); % initialize homography from previous image to ref

for iFile=2:length(fileNames)
    fprintf('Processing %s ... 
', fileNames{iFile});

    % Get next image.
    Inext = imread(fileNames{iFile});

    % Reduce size if image is too big.
    if size(Inext,2) > MAXWIDTH
        Inext = imresize(Inext, MAXWIDTH/size(Inext,2));
    end
% SIFT matching requires grayscale images.
if (size(Iprev,3)>1)
    Ip = rgb2gray(Iprev);  % Convert to grayscale
else
    Ip = Iprev;
end
if (size(Inext,3)>1)
    In = rgb2gray(Inext);  % Convert to grayscale
else
    In = Inext;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract SIFT features. % These parameters limit the number of features detected
peak_thresh = 0;  % increase to limit; default is 0
edge_thresh = 10;  % decrease to limit; default is 10

% First make sure the vl_sift code is in the path
if exist('vl_sift', 'file')==0
    run('C:\Users\William\Documents\Research\vlfeat-0.9.20\toolbox\vl_setup');
end

% First image
Ip = single(Ip);  % Convert to single precision floating point
figure(1), imshow(Ip,

[f1,d1] = vl_sift(Ip, ...
    'PeakThresh', peak_thresh, ...
    'edgethresh', edge_thresh);
fprintf('Number of frames (features) detected: %d\n', size(f1,2));

% Show all SIFT features detected
h = vl_plotframe(f1) ;
set(h,'color','y','linewidth',1.0) ;
% Second image
In = single(In);
figure(2), imshow(In,[]);

[f2,d2] = vl_sift(In, ...
    'PeakThresh', peak_thresh, ...
    'edgethresh', edge_thresh);
fprintf('Number of frames (features) detected: %d\n', size(f2,2));

% Show all SIFT features detected
h   = vl_plotframe(f2);
set(h,'color','y','linewidth',1.0);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Match features
% The index of the original match and the closest descriptor is stored in
% each column of matches and the distance between the pair is stored in
% scores.

% Define threshold for matching. Descriptor D1 is matched to a descriptor
% D2 only if the distance d(D1,D2) multiplied by THRESH is not greater than
% the distance of D1 to all other descriptors
thresh = 1.5;   % default = 1.5; increase to limit matches
[matches, scores] = vl_ubcmatch(d1, d2, thresh);
fprintf('Number of matching frames (features): %d\n', size(matches,2));

indices1 = matches(1,:);        % Get matching features
f1match = f1(:,indices1);
d1match = d1(:,indices1);

indices2 = matches(2,:);
f2match = f2(:,indices2);
d2match = d2(:,indices2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = size(f1match,2);   % Number of corresponding point pairs
sigma = 1.0;   % Sigma of errors of points found in lowest scale

MATLAB code for stitching multiple images (3 of 5)
% Let's say that we want to get a good sample with this probability.
confidence = 0.99;

% Try to fit a homography from the points in image 1, to the points in
% image 2. Returns:
%   indices: the indices of the original points, that are inliers
%   rmsErr: root mean squared error of the inliers (in pixels)
% tform_1_2: a "projective2d" object, computed by "fitgeotrans".
% Note: if you want to get the 3x3 homography matrix such that p2 = H*p1,
% do Hom_1_2 = tform_1_2.T'.
[tform_1_2, indices, rmsErr] = fitHomographyRansac( ...  
  f1match, ... % image 1 keypoints, size 4xN  
  f2match, ... % image 2 keypoints, size 4xN  
  size(In,1),size(In,2), ... % height and width of image 2  
  sigma, ... % uncertainty of image points at lowest scale  
  1000, ... % don't go above this many iterations  
  confidence, ... % desired confidence level, 0..1  
  Ip, In ... % show images (for visualization only)  
);

% Show final matches.
f1Inlier = f1match(:,indices);f2Inlier = f2match(:,indices);
figure(3), subplot(1,2,1), imshow(Ip,[]);
h = vl_plotframe(f1Inlier) ;set(h,'color','y','linewidth',1.5) ;
subplot(1,2,2), imshow(In,[]);h = vl_plotframe(f2Inlier) ;set(h,'color','y','linewidth',1.5) ;

% This is the 3x3 homography matrix, from the previous image to next.
H_prev_next = tform_1_2.T';
H_next_prev = inv(H_prev_next); % invert to get next to previous

% Combine homographies to get the transform from next to reference.
H_next_ref = H_prev_ref * H_next_prev;
tform_next_ref = projective2d(H_next_ref);
% Warp next image to align it with reference image.
    [Inext_ref, Rnext] = imwarp(Inext, tform_next_ref);

% Fuse the two images.
    [Iref, Rref] = imfuseWarped(Iref, Rref, Inext_ref, Rnext);

    figure(4), imshow(Iref, []);
    pause;

    Iprev = Inext;
    H_prev_ref = H_next_ref;
end

MATLAB code for stitching multiple images (5 of 5)
Example

• “carpet” images

image1.jpg  image2.jpg  image3.jpg  image4.jpg