VLG: Vision Lab Geometry Library

UCLA Vision Lab

Taehee Lee
11/24/2009

Overview

- Introduction
- Installation
- Functions
- Example

Introduction

- VLG
  - Vision Lab Geometry Library
  - A collection of functions for 3D reconstruction
    - Feature matching, Epipolar geometry, triangulation, pose estimation, auto-calibration, bundle adjustment, visualization, etc.
  - Matlab implementation (with a few Mex files)
  - http://vision.ucla.edu/vlg

Installation

- Install VLFeat
  - http://vlfeat.org
- Install VLG
  - Copy the VLG directory at the same place as VLFeat:

    /lib/
    /lib/vlfeat/
    /lib/vlg/

  - Set Matlab path:

    >> cd /lib/vlg
    >> vlg_setup

Installation: Compile Mex files

- The tar file contains binary files of Mex files for Windows, Linux and Mac.

- If you need to compile them on a different platform/version, try this:
  - For Linux and Mac, “**make**”
  - For Windows, “**nmake /f Makefile.mak**”

First Run: demo_vlmvg

- After installing VLG, run demo_vlmvg to see a demo of 3D reconstruction.

  >> demo_vlmvg

Function: Feature Matching

- **[matches] = match_sift_unique(d1, d2)**

- Script:

```
I1 = imread('UCLA_01.jpg');
I2 = imread('UCLA_02.jpg');
[f1 d1] = vl_sift(single(rgb2gray(I1)));
[f2 d2] = vl_sift(single(rgb2gray(I2)));

matches = match_sift_unique( d1, d2 );

x1 = [f1(1:2,matches(1,:));ones(size(matches(1,:)))];
x2 = [f2(1:2,matches(2,:));ones(size(matches(2,:)))];
display_image_pair(I1,I2,x1,x2);
```

Function: Robust Feature Matching

- **[inlier] = ransac_epipolar_constraint(x1, x2, iter, thresh)**

- Script:

```
[inlier] = ransac_epipolar_constraint( x1, x2, 100, 0.1 );
x1_ = [f1(1:2,matches(1,inlier));ones(size(matches(1,inlier)))];
x2_ = [f2(1:2,matches(2,inlier));ones(size(matches(2,inlier)))];
display_image_pair(I1,I2,x1_,x2_);
```

Function: Fundamental Matrix

- **[F] = fundamental(x1, x2)**
  [F] = fundamental(x1, x2, 'calibrated')

- Script:

```
[F] = fundamental( x1_, x2_ );
display_image_pair_with_F( I1, I2, F, x1_, x2_ );
```

Function: 2-view Reconstruction

- **[X lambda R T F error] = two_view(x1, x2, varargin)**
  - **‘calibrated’**
  - **‘K’, K**

- Script:

```
cx = size(I1,2) / 2;
cy = size(I1,1) / 2;
fx = 2 * cx;
fy = fx;
Kguess  = [ fx 0 cx ; 0 fy cy ; 0 0 1 ];

[X lambda R T F error] = two_view( x1_, x2_, 'K', Kguess );
display_points( X, 'pixelsize', 10 );
```

Function: Multi-view Reconstruction

- **[Xp Pp x_ vis] = multi_view(x, depth, varargin)**
  - **'K', K**
  - **'calibrated'**
  - **'visibility', vis**
  - **'visualize'**

- Script:

```
cd([vlg_root,'/data']);
filelist = dir('UCLA*.jpg');
[featx,featy,vis,color] = generate_feature_track( filelist );
vis = (featx~=0 | featy~=0);
index = vis(:,1) & vis(:,2);
featx = featx(index,:);
featy = featy(index,:);
```

- Script (Continued):

```
cx = ( max(max(featx)) - min(min(featx)) ) / 2;
cy = ( max(max(featy)) - min(min(featy)) ) / 2;
fx = 2 * cx - min(min(featx));
fy = fx;
Kguess  = [ fx 0 cx ; 0 fy cy ; 0 0 1 ];

[n m] = size(featx);
x = ones(3, n, m);
x(1,:,:) = featx;
x(2,:,:) = featy;

[Xp lambda] = two_view( x(:,:,1), x(:,:,2), 'K', Kguess );

vis = (featx~=0 | featy~=0);
vis(lambda<0,:) = 0;

[Xp Pp x_ vis] = multi_view( x, lambda, 'K', Kguess, 'visibility', vis, 'visualize' );
```

Function: Euclidean Upgrade

- **[Xe Re Te K] = upgrade_euclid(Xp, Pp)**

- Script:

```
Kparam = repmat(calibration_parameter(Kguess), 1, m);
[Pp_ Xp_] = bundle_projective( Pp, Xp, x_, 'visibility', vis, 'verbose' );
[Xe Re Te K] = upgrade_euclid( Xp_, Pp_ );
K = Kguess * K(:,:,1);

w = zeros(3,m);
for i=1:m, w(:,i) = vl_irodr( Re(:,:,i) ); end;
[K T Omega X] = bundle_euclid( Kparam, Te, w, Xe, x, 'fix_calibration', 'visibility', vis, 'verbose' );

[T_ Omega_ X_] = align_scene( T, Omega, X );

figure, display_points(X_, 'pixelsize', 10, 'color', color(index,:));
display_cameras(T_, Omega_, K, 'frustum_scale', 0.1), box on, view(3);
```

Function: Bundle Adjustment

- **[Pp_ Xp_ error_] = bundle_projective(Pp, Xp, x, varargin)**
  - **‘fix_structure’**
  - **‘fix_motion’**
  - **‘visibility’, vis**
  - **‘verbose’**

- **[K_ Te_ w_ Xe_ error_] = bundle_euclid(K, Te, w, Xe, x, varargin)**
  - **‘fix_structure’**
  - **‘fix_motion’**
  - **‘fix_calibration’**
  - **‘fix_principal’**
  - **‘visibility’, vis**
  - **‘verbose’**

Function: Triangulation

- **[X] = triangulation(K, T, Omega, x)**

- Script:

```
for i = 1:n
    X0(:,i) = triangulation( K, T, Omega, x(:,i,:) );
end
figure, display_points(X0, 'pixelsize', 10 );
```

Function: Pose Estimation

- **[K T Omega inlier] = estimate_camera(x, X, varargin)**
  - **‘K’, K**

- Script:

```
[K3 T3 Omega3] = estimate_camera( x(:,vis(:,3),3), X(:,vis(:,3)), 'K', K(:,3) );

% check with T and Omega from the previous reconstruction
T
T3
Omega
Omega3
```

Function: Incremental Bundle Adjustment

- **[K T Omega X goodfeat status] = incr_reconstruction(featx, featy, varargin)**
  - **‘visibility’, vis**
  - **‘calibrated’**
  - **‘K’, K**
  - **‘initialized’, T, Omega, X**
  - **‘status’, status**
  - **‘X’, X**

- Script:

```
[K1 T1 Om1 X1] = incr_reconstruction( featx, featy, 'K', Kguess, 'visibility', vis );
```

Function: Visualization

- **display_points(X, varargin)**
  - **‘pixelsize’, pixelsize**
  - **‘color’, color**

- **display_cameras(T, Omega, K, varargin)**
  - **‘frustum_scale’, scale**
  - **‘trajectory_color’, color**
  - **‘no_frustum’**
  - **‘no_trajectory’**

Function: All-in-one

- **[K_ T_ Om_ X_ vis_ inlier_] = VLmvg(featx, featy, varargin)**
  - **‘visibility’, vis**
  - **‘calibrated’**
  - **‘K’, K**
  - **‘T’, T**
  - **‘Omega’, Om**
  - **‘X’, X**
  - **‘ransac’**
  - **‘sampson_threshold’, threshold**

- Script:

```
cd /library/data/oldhouse
filelist = dir('*.bmp');
[featx,featy,vis,color] = generate_feature_track( filelist );
[K_ T_ Om_ X_ vis_ inlier_] = VLmvg( featx, featy, 'ransac' );

figure, display_points(X_, 'pixelsize', 5, 'color', color);
display_cameras(T_, Om_, K_, 'frustum_scale', 0.1), box on, view(3);
```

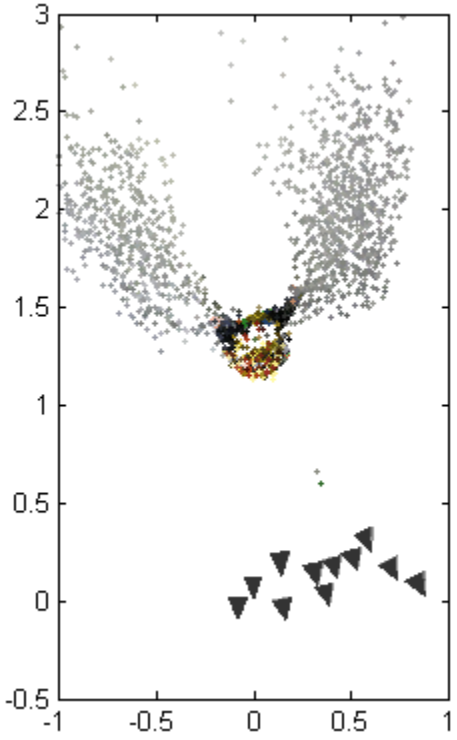
Function: Others

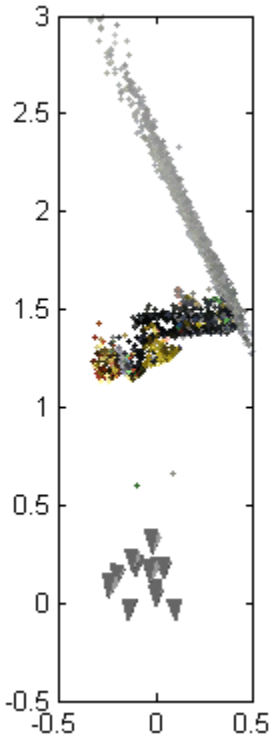
- Other functions
  - Outlier rejection
  - Scene alignment
  - Quaternion
  - Demo scripts
  - Synthetic data generation
  - Test scripts

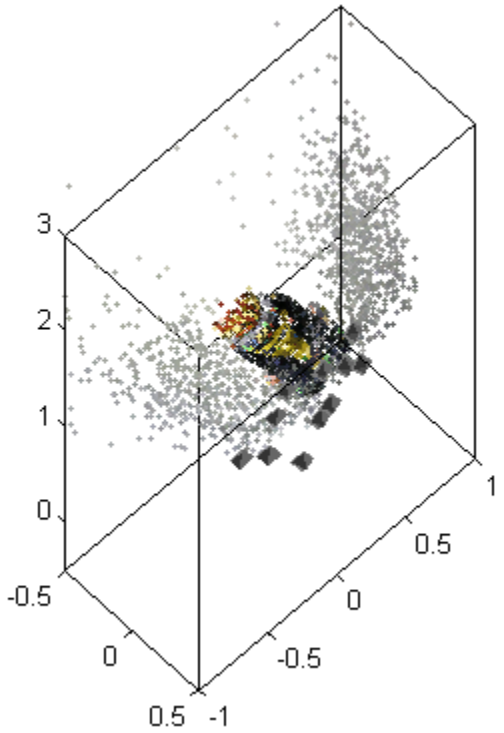
- Please read the documentation:
  - http://vision.ucla.edu/vlg/doc/index.html

Example

- ‘Pharaoh’
  - 1024x768 resolution
  - 3037 points, 16 cameras
  - SIFT features matched

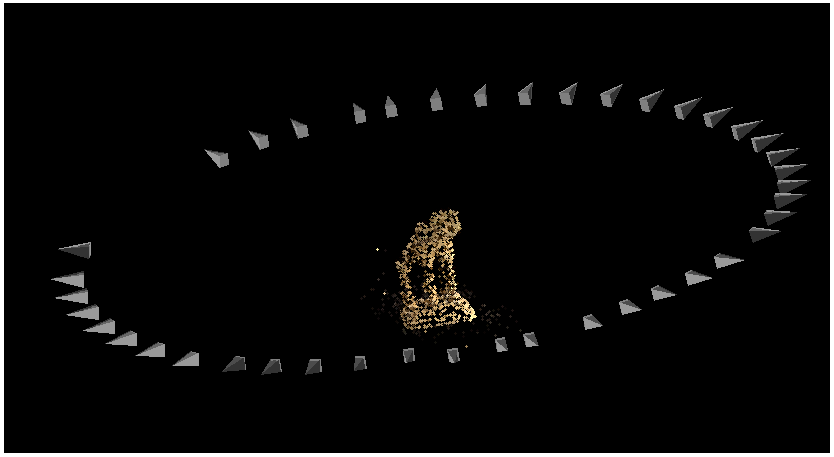


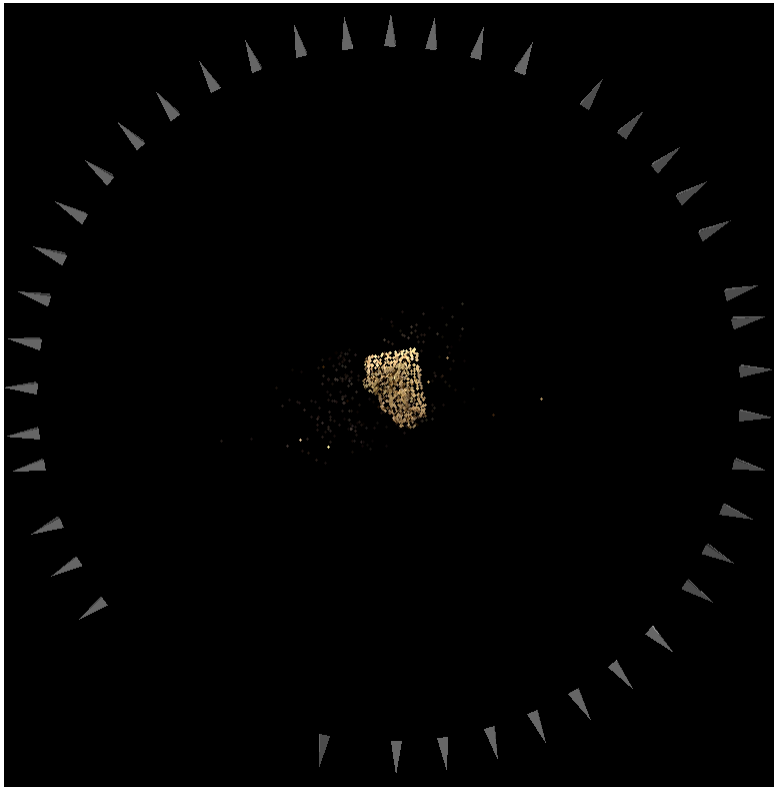




Example

- templeRing
  - 640x480 resolution
  - 47 cameras, 2236 SIFT features matched





Thank you

http://vision.ucla.edu/vlg

taehee @ cs.ucla.edu