Université Concordia University
FACULTY OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science and Software Engineering

COMP498G/691G COMPUTER VISION

TUTORIAL 3
Image features/matching/stitching

Features Points/Descriptors

some of them are feature detectors, some of them are descriptor extractors.

OpenCV features2D interface:

available features:

Harris-corner, **Shi-Tomasi** corner, **FAST** detector, **BRISK** features, **ORB** features, **GFTT** features, **KAZE** features...

OpenCV xfeatures2D interface:

available features:

SIFT, **SURF**(faster), **BRIEF** descriptors, **DAISY** descriptors, **FREAK** descriptors, **LATCH** features, **LUCID** features, **Star**Features

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

Feature Detection with OpenCV

OpenCV 3 requires creating ptr class(Different from OpenCV 2 !)

…

```
cv::Ptr<cv::Features2d> detector = cv::xfeatures2d::SIFT::create();
detector->detect(data.img, data.keypoints);
detector->compute(data.img, keypoints, data.descriptors);
```

...

src:http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/warp_affine/warp_affine.html

Corner Detection

OpenCV supports raw corner detection :

//prototype

```
void cv::cornerHarris
(
InputArray  src,
OutputArray dst,  //output is a image as well.
int blockSize,   //neighborhood size to exam the corner
int ksize,  //size for sobel kernel
double k,  //harris formula paramater
int borderType = BORDER_DEFAULT )
```

Corner Detection

Post processing after Corner detection:

```
/// Normalizing
 normalize( dst, dst_norm, 0, 255, NORM_MINMAX, CV_32FC1, Mat() );
 convertScaleAbs( dst_norm, dst_norm_scaled );

/// Drawing a circle around corners
 for( int j = 0; j < dst_norm.rows ; j++ )
    { for( int i = 0; i < dst_norm.cols; i++ )
         {
           if( (int) dst_norm.at<float>(j,i) > thresh )
             {
              circle( dst_norm_scaled, Point( i, j ), 5,  Scalar(0), 2, 8, 0 );
             }
         }
    }
```

http://docs.opencv.org/2.4/doc/tutorials/features2d/trackingmotion/harris_detector/harris_detector.html#harris-detector

SIFT and SURF

SURF is an improvement over SIFT, SURF is faster and claimed to more robust

1) Both are patent,
2) Both are multi scaled.

SIFT feature detection:

//prototype

```
static Ptr<SIFT>
cv::xfeatures2d::SIFT::create
(
int nfeatures = 0,
int nOctaveLayers = 3,
double contrastThreshold = 0.04,
double edgeThreshold = 10,
double sigma = 1.6 )
```

SURF feature detection:

//prototype

```
static Ptr<SURF>
cv::xfeatures2d::SURF::create
(
double hessianThreshold = 100,
int nOctaves = 4,
int nOctaveLayers = 3,
bool extended = false,
bool upright = false )
```

Other features2D implementations

1) FAST detector

```
static Ptr<FastFeatureDetector>
cv::FastFeatureDetector::create
(
int threshold = 10,
bool nonmaxSuppression = true,
int type = FastFeatureDetector::TYPE_9_16
)
```

2) BRICK features

```
static Ptr<BRISK> cv::BRISK::create
(
int thresh = 30,
int octaves = 3,
float patternScale = 1.0f
)
```

3) ORB features

```
static Ptr<ORB> cv::ORB::create(
//with default parameters
)
```

4) KAZE features

```
static Ptr<KAZE> cv::KAZE::create
(
bool extended = false,
bool upright = false,
float threshold = 0.001f,
int nOctaves = 4,
int nOctaveLayers = 4,
int diffusivity = KAZE::DIFF_PM_G2
)
… and more ...
```

http://docs.opencv.org/3.1.0/d0/d13/classcv_1_1Feature2D.html

OpenCV xfeatures2D features

1) DAISY descriptors

```
static Ptr< DAISY > create (
float radius=15,
int q_radius=3,
int q_theta=8,
int q_hist=8,
int norm=DAISY::NRM_NONE,
InputArray H=noArray(),
bool interpolation=true, bool use_orientation=false
)
```

2) FREAK descriptors

```
static Ptr< FREAK > create (bool
orientationNormalized=true,
bool scaleNormalized=true,
float patternScale=22.0f,
int nOctaves=4,
const std::vector< int > &selectedPairs=std::vector< int >()
)
```

3) LUCID features

```
static Ptr<LUCID> cv::xfeatures2d::LUCID::create
(
const int lucid_kernel, //could 1,3,5, 7
const int blur_kernel, //could 1,3,5,7
)
```

...

http://docs.opencv.org/3.1.0/d0/d13/classcv_1_1Feature2D.html

Recap on OpenCV features2D data-structure

Important Data structures

1) **SIFT-keypoint**
std::vector<cv::KeyPoint>

2) **Match results**
std::vector<cv::DMatch>

3) **Descriptors**
cv::Mat

and function calls

1) **SIFT-keypoints**
cv::xfeatures2d::SIFT::create()->
detect(...)

2) **Feature descriptor**
cv::xfeatures2d::SIFT::create()->
compute(...)

3) **Match descriptors**
cv::BFMatcher.match(...)
cv::FlannMatcher.match(...)

And next step?

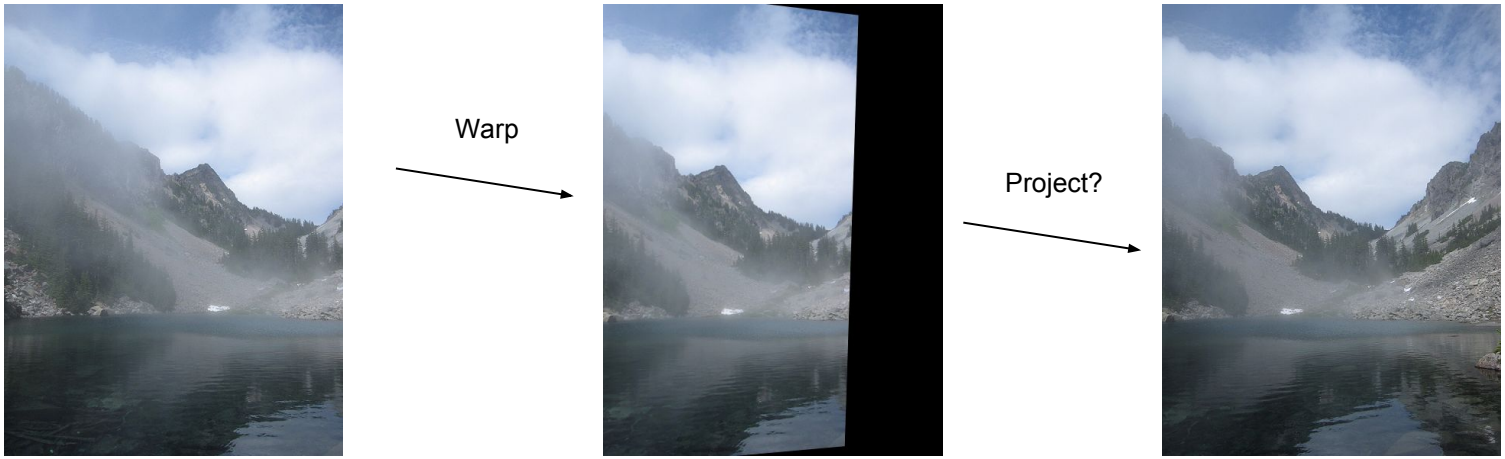
Recap on Transformation

Projective transformation:

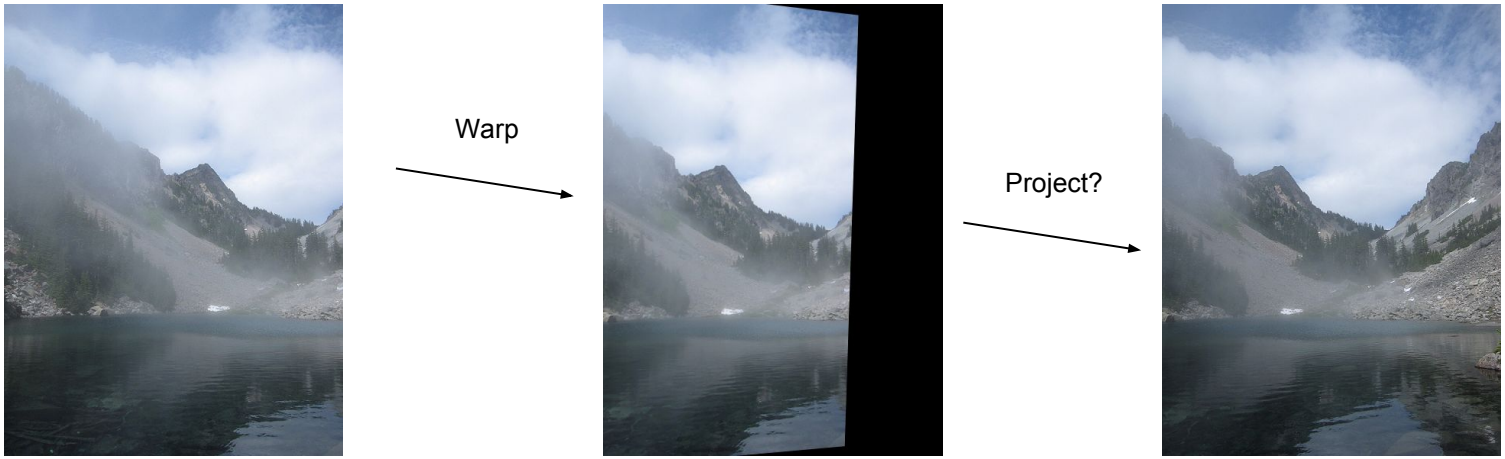
Mat cv::findHomography(InputArray *srcPoints*,InputArray *dstPoints*,int *method* = 0,double *ransacReprojThreshold* = 3,OutputArray *mask* = noArray(),const int *maxIters* = 2000,const double *confidence* = 0.995)

```
void warpPerspective(InputArray src, OutputArray dst, InputArray M, Size
dsize, int flags=INTER_LINEAR, int borderMode=BORDER_CONSTANT, const Scalar&
borderValue=Scalar());
```

WarpPerspective() from left to right

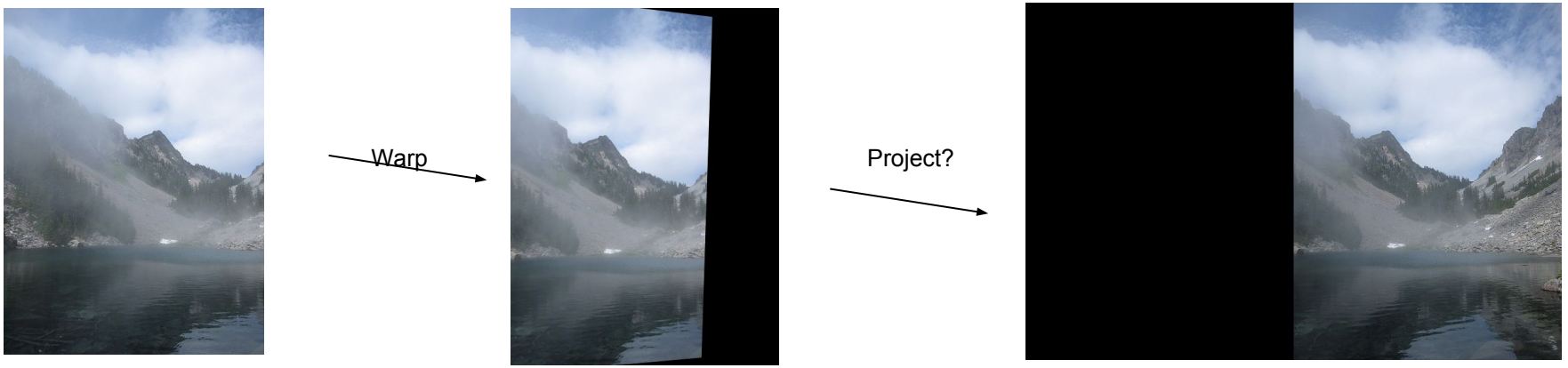


WarpPerspective() from left to right



Part of the left Image get cropped.

Possible solution



enlarge the right image.

Possible solution

reprojection

CONCORDIA.CA