

Key-Point based Image retrieval under ISO still image compression standards (JPEG, JPEG2000, JPEG XR))

Alexander Kollert
Babak Maser
Sascha Wernegger

July 1, 2016

Table of Contents

- 1 Introduction
- 2 Performance Evaluation
- 3 Experiment Setup
- 4 Feature Detectors
- 5 Results

Content Based Image Retrieval

CBIR

- **retrieval:** Search for similar images in large database
- **content based:** Use image data of reference to find similar images

How to measure the influence of compression?

- Use retrieval system and calculate some performance measure for the results
- Use lossless compressed dataset and run retrieval system to calculate the performance
- Compress dataset to different and formats ratios and do the same
- Compare results

Software

Matlab

ViBenchmark, VLFeat

- feature extractors
- retrieval system
- performance measure
- database

OpenCV/mexopencv

- feature extractors

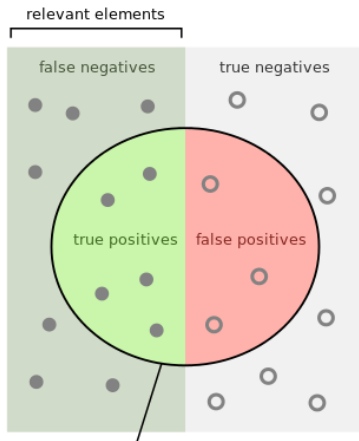
ImageMagick, JXRLIB

- compression

Table of Contents

- 1 Introduction
- 2 Performance Evaluation
- 3 Experiment Setup
- 4 Feature Detectors
- 5 Results

Recall Precision



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Recall Precision



= the relevant documents

Ranking #1

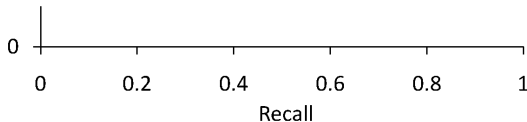


Recall	0.17	0.17	0.33	0.5	0.67	0.83	0.83	0.83	0.83	1.0
Precision	1.0	0.5	0.67	0.75	0.8	0.83	0.71	0.63	0.56	0.6

Ranking #2

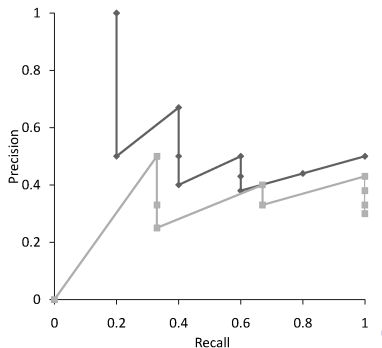


Recall	0.0	0.17	0.17	0.17	0.33	0.5	0.67	0.67	0.83	1.0
Precision	0.0	0.5	0.33	0.25	0.4	0.5	0.57	0.5	0.56	0.6

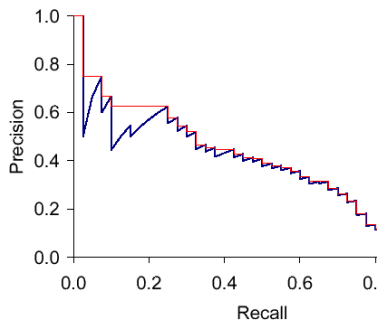


Precision Recall Curves

To visualize precision and recall, we plot precision over recall.
If we have many queries, it is easier to interpret an interpolated precision recall curve



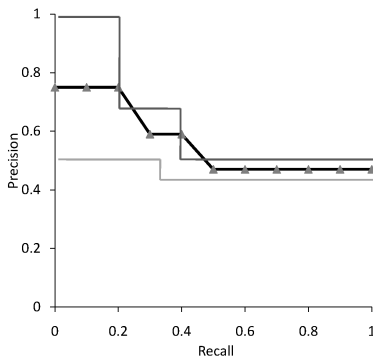
11-Point Precision-Recall Curve



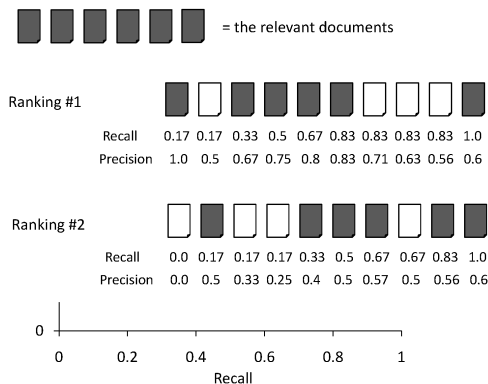
Calculate precision at each recall level R as the maximum precision observed in any recall-precision point at a higher recall level

11-Point Precision-Recall Curve

Calculate precision
at each recall level R as
the maximum precision
observed in any
recall-precision point
at a higher recall level



Mean Average Precision



average

precision of a query is
the average precision from
all relevant documents.

mean average precision
is the average of the
query average Precisions.

Table of Contents

- 1 Introduction
- 2 Performance Evaluation
- 3 Experiment Setup**
- 4 Feature Detectors
- 5 Results

Dataset

Oxford Buildings Dataset

- 5062 images
- compressed loss-less or with minimal loss in JPEG
- collected from Flickr by searching for particular Oxford landmarks
- manually annotated to generate ground truth for 11 different landmarks
- 5 queries per landmark
- total of 55 queries

Random sampled subsets to improve compression and benchmark speed!

Queries

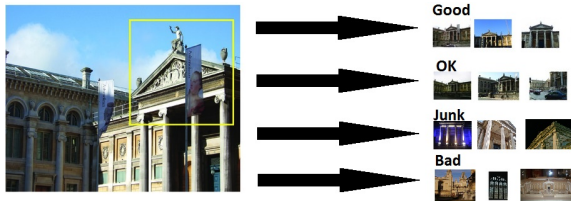
The query consists of a reference image and 4 query sets:

good A nice, clear picture of the object

ok More than 25% of the object is clearly visible.

junk Junk Less than 25% of the object is visible, or there are very high levels of occlusion or distortion.

bad Object not present



Mean Average Precision add

VIBenchmark uses a slightly different scheme, but the calculation stays the same.

How use the four query classes

- good and ok images are relevant
- junk will be ignored
- bad will count as irrelevant

Junk images do not influence precision or recall!

Estimate Compression Ratio

Because the dataset we used was JPEG compressed we first estimated the compression ratio.

t

s : avg size = 391.659 bytes.

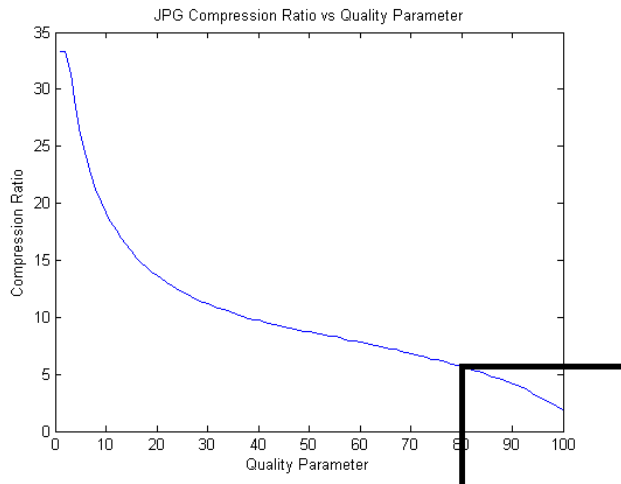
p : avg pixels = 765.969 pixel.

bpp : byte per pixel = 3.

r : avg raw size = $p * bpp = 2297908$.

e : estimated ratio = $r/s = 5.867$.

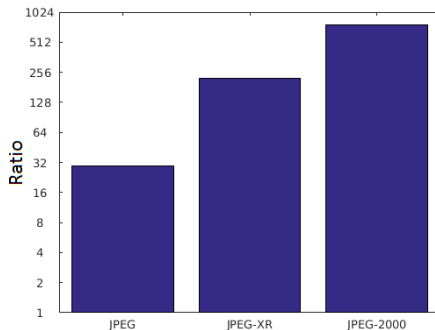
So we have a ratio smaller 6.



Compression

Achieved compression ratio compared to the original(391.659 bytes).

	jpg	jxr	jp2
min avg size in bytes	13229	1756	511



Sample

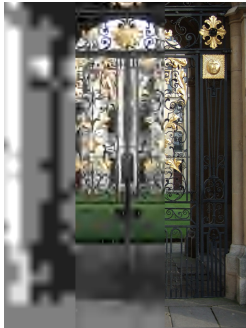


Comparison

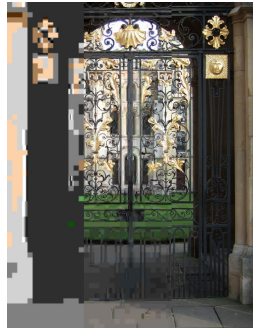
jpg



jxr



jp2



Generic Local Feature Extractor

The Ranking in VIBenchmark is done using Local Feature Extractors. The purpose of the Feature Extractor is to calculate frames and corresponding descriptors from the image data.

Local Feature Frames

- search image for interest points
- define a frame for that point(points,circles,ellipses)

Descriptor

- compute descriptor using the image data defined by the frame
- every descriptor is a vector of same dimension

So we got n frames and n descriptors

Retrieval System

Ranking

- calculate reference descriptors
- calculate KNN's of the query descriptors
- for every Descriptor of the KNN's Vote for the corresponding image of the descriptor
- repeat for all reference descriptors
- Rank images by their recieved votes

Img 1 (good)



Img 2 (good)



Img 3 (ok)



Img 4 (bad)



Img 5 (good)



Img 6 (good)



Img 7 (good)



Img 8 (good)



Img 9 (good)



Img 10 (good)



Img 11 (good)



Img 12 (ok)



Img 13 (bad)



Img 14 (bad)



Img 15 (bad)



Img 16 (bad)



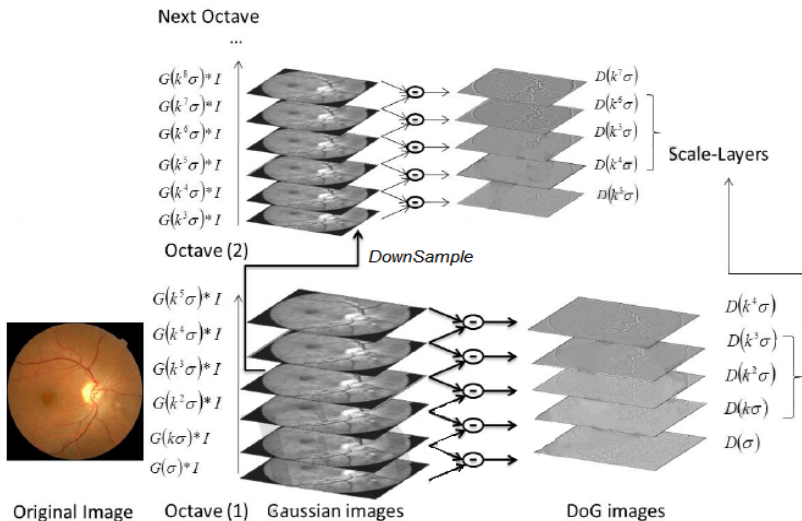
Table of Contents

- 1 Introduction
- 2 Performance Evaluation
- 3 Experiment Setup
- 4 Feature Detectors**
- 5 Results

SIFT (Scale-Invariant Feature Transform)

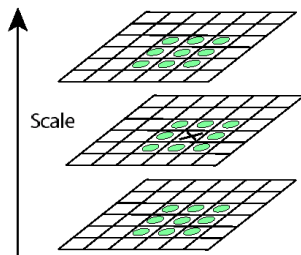
- SIFT Algorithm is comprised by 3 steps:
 - 1 Extracting key points (features)
 - 1 Local extrema detection
 - 2 Accurate keypoint localization
 - 3 Orientation assignment
 - 2 Feature descriptor (based on Gradient Histogram)
 - 3 Feature matching

Difference of Gaussian



Accurate keypoint localization

- Compare a pixel (x) with 26 pixels in current and adjacent scales (Green Circles)
- Scale a pixel (X) if larger/smaller than all 26 pixels
- Eliminating the Edge Response
- Eliminating poor contrast points

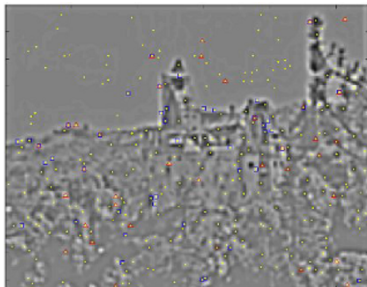


$$feature_i = x_i, y_i, s_i$$

Orientation Assignment

For each remaining key point

Choose surrounding $N \times N$ window at DOG level it was detected
(Size of the window: The window size is equal to the size of the kernel for Gaussian Blur of amount $1.5 * \sigma$).

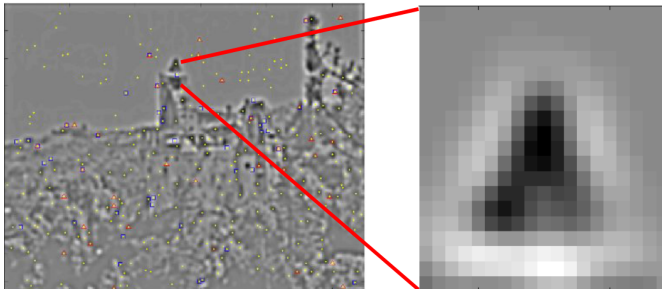


Orientation Assignment

For each remaining key point

Choose surrounding $N \times N$ window at DOG level it was detected
(Size of the window: The window size is equal to the size of the kernel for Gaussian Blur of amount $1.5 * \sigma$).

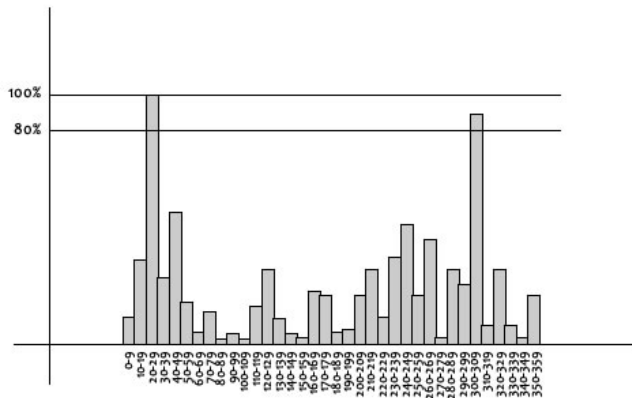
[t]



Orientation Assignment

- A histogram is created for mentioned images (Gradient Orientation and Weighted Magnitude)
- In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees). Once you've done this for all pixels around the keypoint, the histogram will have a peak at some point
- Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But its orientation is equal to the other peak
- So, orientation can split up one keypoint into multiple keypoints

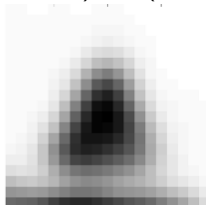
Orientation Assignment



Orientation Assignment

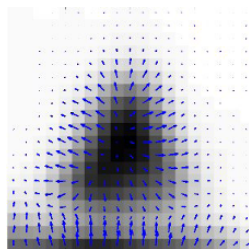
Gradient Magnitude

$$m(x, y) = \frac{\sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}}{2}$$



Gradient Orientation

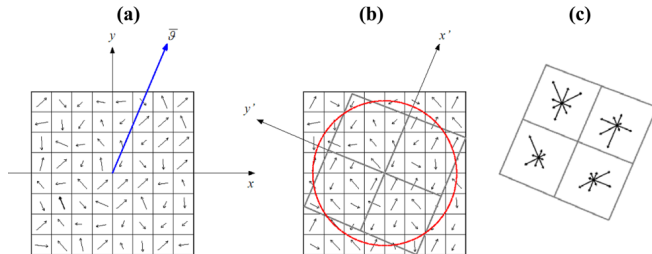
$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$



Orientation Assignment

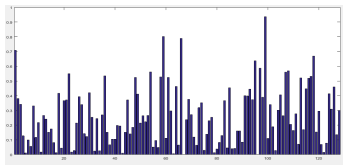
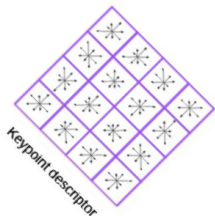
- A histogram is created for mentioned images (Gradient Orientation and Weighted Magnitude).
- In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees).
- Once you've done this for all pixels around the keypoint, the histogram will have a peak at some point.
- Also, any peaks above 80
- **So, orientation can split up one keypoint into multiple keypoints.**

Feature Descriptor



Feature Descriptor

- 4x4 arrays of 8 bin histogram is used
- a total of 128 features for one keypoint



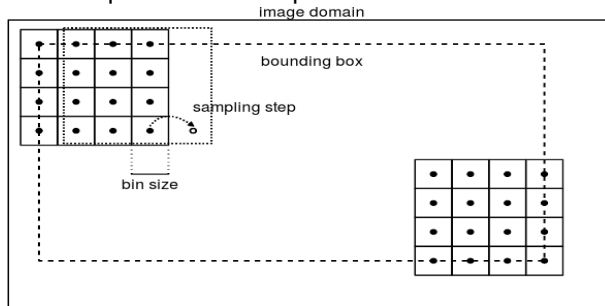
SURF(Speed Up Robust Feature)

- **Scale Space:** In SURF, square-shaped filters are used as an approximation of Gaussian smoothing.
- **Key point detection:** Hessian matrix and Non-maximum suppression.
- **Orientation:** Sliding orientation window using Gaussian weighted Haarwavelet responses from circular neighborhood
- **Descriptor:** An oriented 4x4 grid that defines subregion. Wavelet responses computed for 5x5 samples of the subregion.

PHOW

The PHOW features are a variation of dense SIFT descriptors, extracted at multiple scales. A color version, extracts descriptors on the three color channels and stacks them up.

- When computing descriptors for many keypoints differing only by their position (and with null rotation), further simplifications are possible.

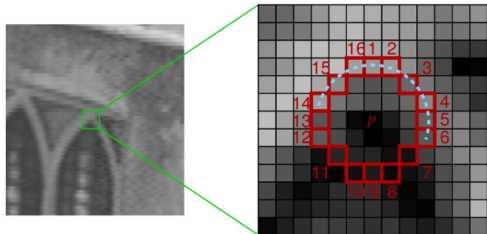


ORB(Oriented FAST and Rotated BRIEF)

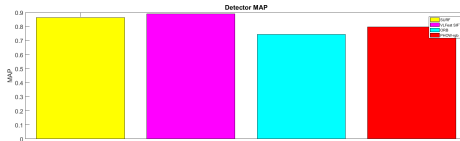
- FAST used to find key-points.
- Unlike the orientation operator in SIFT, which can have multiple value on a single keypoint, the centroid operator gives a single dominant result.
- Dimensionality reduction can be achieved by using hash functions that reduce SIFT descriptors to binary strings

Corner Detection

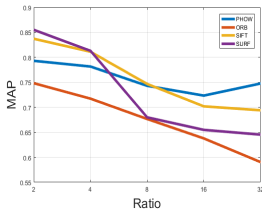
A corner is detected if n contiguous pixels in the circle are brighter or darker than the pixel in the center



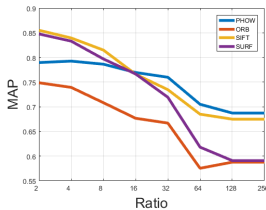
MAP over Ratio



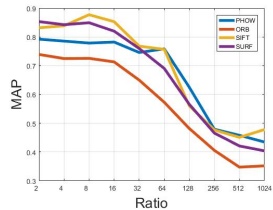
JPEG



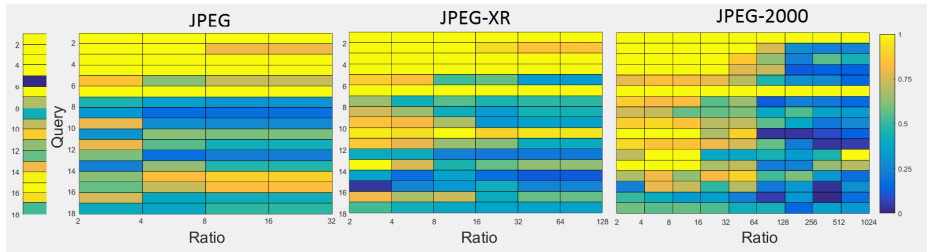
JPEG XR



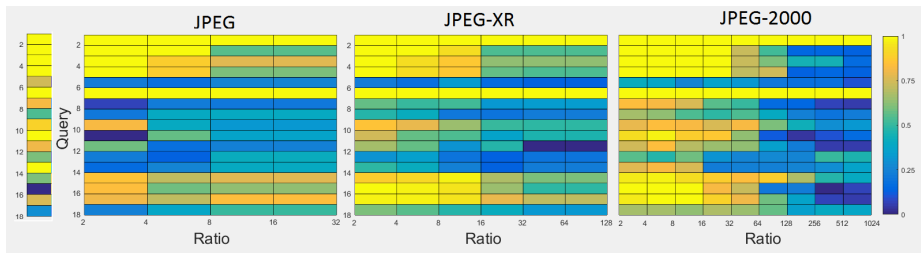
JPEG 2000



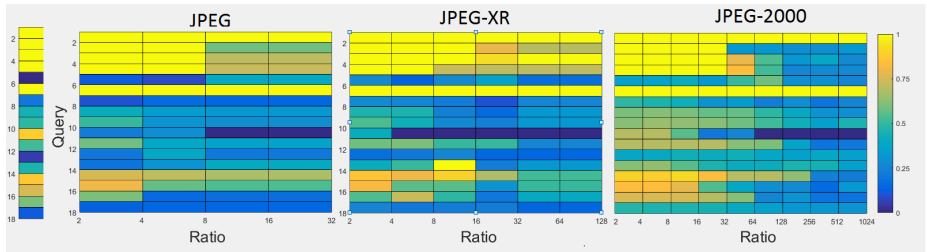
SIFT Query AP over Ratio



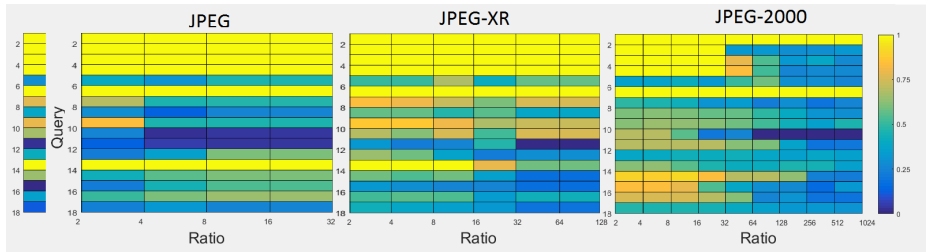
SURF Query AP over Ratio



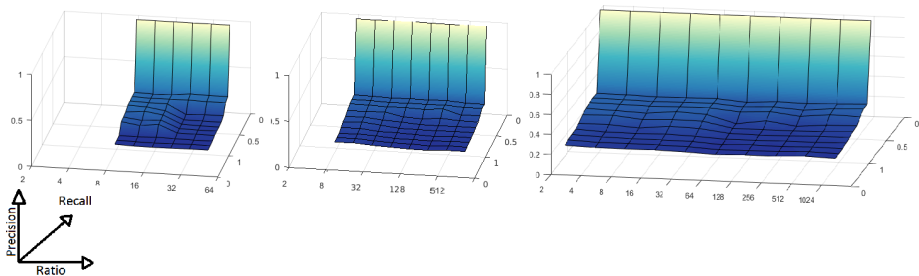
ORB Query AP over Ratio



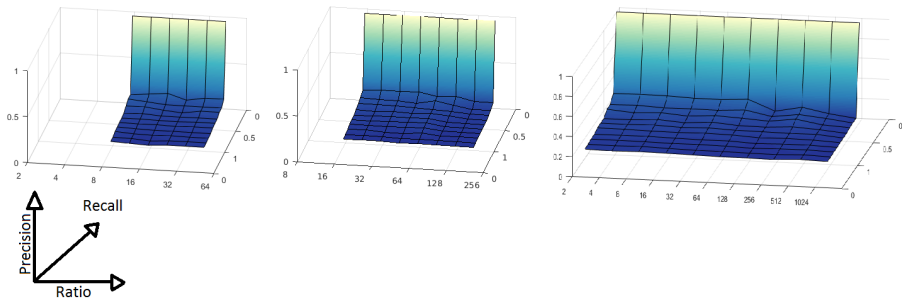
PHOW Query AP over Ratio



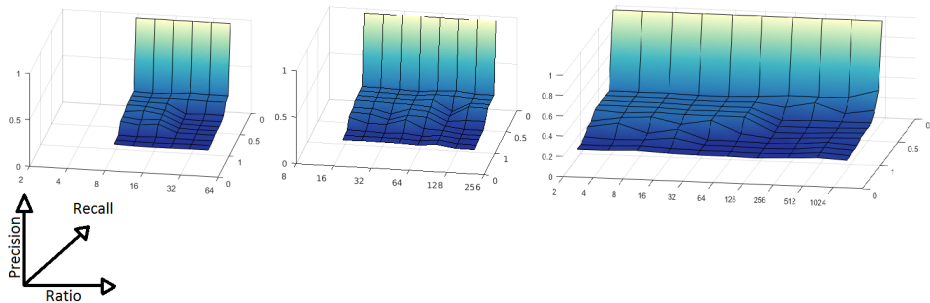
SIFT PRC over Ratio



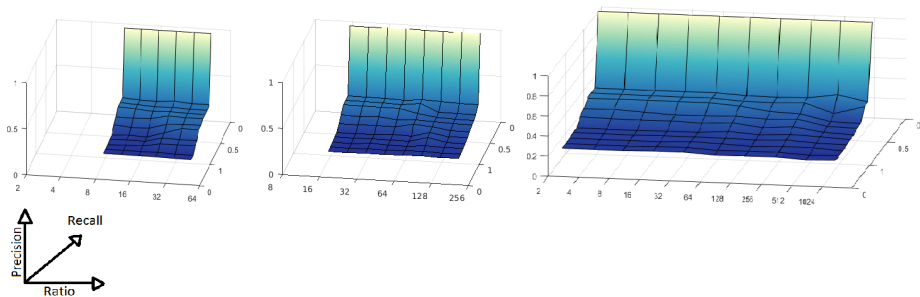
SURF PRC over Ratio



ORB PRC over Ratio



PHOW PRC over Ratio



Thank You!