Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Plzeň, 2021 **Tomáš Majer**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš MAJER**
Osobní číslo:	**A18B0520P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Metody popisu obrázků pro účely vizuálního vyhledávání**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Nastudujte a popište stávající metody popisu obrázků v úloze vizuálního vyhledávání.
2. Zvolte nebo vytvořte databázi obrázků pro testování kvality těchto metod.
3. Vyberte vhodné metody a otestujte je na zvolené databázi.
4. Získané výsledky vyhodnoťte.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

https://www.cs.ubc.ca/ lowe/papers/ijcv04.pdf
Lowe, David G. „Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

https://ieeexplore.ieee.org/abstract/document/6126544?casa_token=VXIicM4iCyEAAAAA:SFEiCXgTi_xrjRwQysps_h6jfKNjrtVgBfN5owJrjwud43S3A4DXIZYOWUhH-Drt6waLzdIOIA
Rublee, Ethan, et al. „ORB: An efficient alternative to SIFT or SURF." 2011 International conference on computer vision. Ieee, 2011.

https://www.researchgate.net/publication/236985005_KAZE_Features
Alcantarilla, Pablo Fernández, Adrien Bartoli, and Andrew J. Davison. „KAZE features." European Conference on Computer Vision. Springer, Berlin, Heidelberg, 2012.

https://www.researchgate.net/publication/224164326_Aggregating_local_descriptors_into_a_compact_image_representati
Jégou, Hervé, et al. „Aggregating local descriptors into a compact image representation." 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE, 2010.

https://arxiv.org/abs/1511.07247
Arandjelovic, Relja, et al. „NetVLAD: CNN architecture for weakly supervised place recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Vedoucí bakalářské práce: **Ing. Marek Hrúz, Ph.D.**
Výzkumný program 1

Datum zadání bakalářské práce: **15. října 2020**
Termín odevzdání bakalářské práce: **24. května 2021**

Západočeská univerzita v Plzni Fakulta aplikovaných věd

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

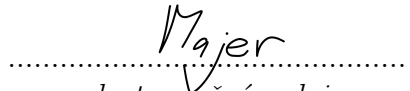
V Plzni dne 15. října 2020

P R O H L Á Š E N Í

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 17.května 2021



.....

vlastnoruční podpis

Poděkování

Tímto děkuji vedoucímu své práce panu Ing. Marku Hrúzovi, Ph.D. za odborné vedení, za cenné a četné rady a konzultace a za vstřícný přístup.

Anotace

Cílem této práce je nastudování a popis metod pro tvorbu detektorů a deskriptorů klíčových bodů v obraze, tvorba vhodného testovacího datasetu a otestování popsaných metod na tomto datasetu s následným vyhodnocením.

V první části práce se věnujeme principu detekce a deskripce klíčových bodů. Následně jsou představeny zvolené metody (SIFT, AKAZE, ORB, LATCH). V druhé části jsou popsány metody vyhodnocení deskriptorů, které budou v práci použity. Ve třetí části je popsána tvorba datasetu, kategorie transformací použitých na obrázky a výsledné datasety. V závěru práce je popis nastavení experimentů a jejich vyhodnocení co se efektivity a náročnosti metod týče.

Klíčová slova: detekce klíčových bodů, SIFT, AKAZE, ORB, LATCH, vyhledávání obrázků

Annotation

The goal of this thesis is studying and description of the methods for image keypoints detection and description, creation of suitable dataset and testing of the described methods on this dataset with subsequent evaluation.

The first part of the thesis is dedicated to the principle of keypoints detection and description. Afterwards, the chosen methods are introduced (SIFT, AKAZE, ORB, LATCH). In the second part the evaluation methods used in the thesis are described. In the third part we describe the dataset creation, transformations categories used on the source images and the created datasets. In the last part of our work the experimental setup is described. The results are then evaluated regarding the performance and computational expense of the methods.

Keywords: keypoints detection, SIFT, AKAZE, ORB, LATCH, image retrieval

Contents

1 Introduction

A new copyright issue appeared along with the possibility of uploading any image to the Internet or downloading any image from the Internet. Anyone can download an image and use it in their work, but it is fairly difficult to verify if they should be paying for that particular usage of the image. For such verification in a large database of images, one could use their simple comparison. But a simple comparison fails even for the slightest of image modifications. This issue can be and is solved by a variety of methods that produce an image description in a form of image descriptors. These descriptors are designed to be easier and more reliable to compare.

As we stated already there is a wide range of image description methods that had gone through lengthy development. In our work, we operate only with the more current and widely used methods. Every method is in general more robust to some kinds of image transformations while being prone to some other kinds of image transformations. Some methods emphasize robustness while other focus on low computational cost. The robustness and computational complexity are dependent on the methods modus operandi.

In many cases, the methods are designed to describe images in such a way that the descriptors of similar images are matched together when compared, e.g. the Eiffel Tower viewed from two different viewpoints should have some matched descriptors. For recognition of the copyrighted image, it is important to determine whether the images are the same or whether they are different, even though they capture the same scene or object from different viewpoints. In an ideal world, we could require the method to differentiate between two images captured at the same place, at the same time, but by two different people. This is naturally not possible, but it sheds light on the problematics we are dealing with.

The objective of our work is to find out, which method is the most capable of producing such descriptors of images that match an image with its derivatives and do not match an image with different but similar images. For this purpose, we need to make a good testing dataset. Images that we want to compare to a copyrighted set of images can be altered by various transformations. We can take inspiration from image editors on our mobile devices and computers. They all provide basic transformations like contrast, brightness or sharpness adjustment, and many more specific transformations. We touch on these transformations later. We also need to take into consideration such transformations of an image that occur by the means of physical acquisition of the image, e.g. a reflection on a piece of art in a gallery while using flash and such. Because of these specific transformations that can occur and because of the problematics we are dealing with, we compose our set of applicable image transformations. With the help of the transformations we then create our testing dataset. The set of source images must be chosen in such a way that there is a sufficient number of similar images from a variety of environments.

Every image in this dataset is described by all of the chosen image description methods. Afterwards, we test the descriptors' ability to differentiate between the same (but transformed) image and a different image.

The Second Chapter of our work deals with the analysis and description of the chosen image descriptions methods. We cover the modus operandi of the chosen methods and also the manner of the resultant image description. We then briefly introduce evaluation methods in the Third Chapter. In the Fourth Chapter, we describe the process of the dataset construction. We describe the ground truth set of images, the transformations used, and the resulting dataset. The Fifth Chapter regards the testing of the methods and evaluation of their effectiveness.

2 Methods of image description

In this chapter, we briefly introduce the keypoint detection and description tasks. Afterwards, we describe the chosen methods of image descriptions. The methods selected for our task are Scale-Invariant Feature Transform (SIFT) [1], Accelerated-KAZE (AKAZE) [2], Oriented FAST and Rotated BRIEF (ORB) [3] and Learned Arrangements of Three Patch Codes (LATCH) [4]. In each section dedicated to the method, we describe the way it detects the keypoints. We then describe the process of the descriptor creation of each of the methods.

2.1 Keypoint detection and description

Before we delve into the theoretical field of image description methods, let us look at what the idea means in practice.

The idea behind efficient and correct image matching is to find points in the images that are highly distinctive or unique. Such points are often located at object edges or corners in the image. The methods then focus on extracting points that are also stable and present in the image even when the image is adjusted. We call such points keypoints. An example of such a keypoint detection can be viewed in Figure 1. We can see that the keypoint detection algorithm detected a lot of the keypoints on the statue and island (SIFT was used for the example). We would call this part of the image distinctive. Little to no keypoints were detected in the background because it is very dull.

These keypoints are now detected in the image and have various properties that describe their location, the size of a meaningful neighborhood, and more. These properties are also depicted in Figure 1. But such a description is still very prone to image modifications. If we flipped the image over, the corresponding keypoints would still have the same meaningful neighborhood but its orientation and coordinates of the point would differ greatly. This is the reason to describe the keypoints by a feature vector that captures its features - the significant keypoint properties and ideally the properties of the keypoints neighborhood. The description should capture the neighborhood in a way invariant to rotation, scale, and other transformations. This feature vector is called a descriptor.

Such descriptors allow us to match keypoints with similar attributes and find points of the image that correspond to each other. We can see some of the matched keypoints in Figure 2.

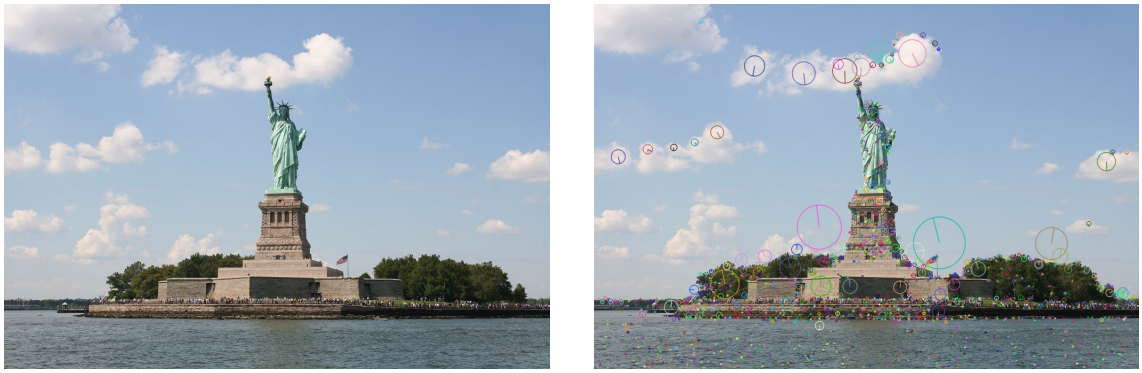


Figure 1: Example of keypoint detection in an image.[1]

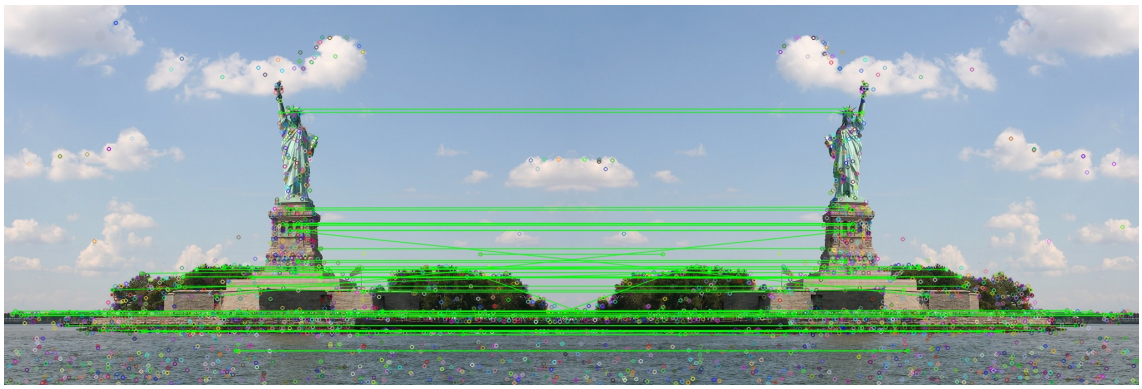


Figure 2: Example of keypoint matching.

Now that we introduced the basic idea of keypoint detection and description, let us present the methods that are used further.

2.2 SIFT

Scale-Invariant Feature Transform [1] is an image description method that computes image features invariant to image scale, rotation of an image. The features are also invariant to noise addition, viewpoint change, and brightness adjustment to a certain degree. This method was first introduced in 1999 by Lowe and later developed up until 2004.

SIFT method divides the computation of the image description into four main steps: extrema detection in so-called scale-space, keypoints localization in the scale-space, keypoint orientation assignment, and the creation of keypoints descriptor.

2.2.1 Extrema detection in scale-space

First, we need to find points in the image that we suspect could be appropriate keypoints. Such suspect points are the image brightness extrema called the candidate points. We require these points to also be invariant to resizing the image. This

[1]Original image in Figure 1 by TheGirlsNY published in 2009 on Flickr.

is achieved by looking for the candidate points across all possible image scales in so-called scale-space.

In SIFT, scale-space is defined as linear (Gaussian) scale-space. One image in the scale-space is defined by the $L(x, y, \sigma)$ function which we compute as a discrete convolution of the Gaussian kernel $G(x, y, \sigma)$ with the described image $U(x, y)$

$$L(x, y, \sigma) = G(x, y, \sigma) * U(x, y). \tag{1}$$

The Gaussian kernel is defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-\left(x^2+y^2\right)}{2\sigma^2}}. \tag{2}$$

We are looking for the extrema in a function called a difference-of-Gaussian. This function is denoted as $D(x, y, \sigma)$. This function is defined as a difference of two neighboring image scales in the scale-space. We also define a constant multiplicative factor k. This factor determines the distances of the neighboring image scales. The difference-of-Gaussian function is then calculated as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma). \tag{3}$$

While searching for local extrema in difference-of-Gaussian $D(x, y, \sigma)$, we compare the currently examined point with its eight neighbors on the current image scale and with nine corresponding points on the neighboring scales. That means a total of 26 points with which the candidate point is compared.

Having completed all of this, we have approximate locations of the candidate points. Now we need to determine if they are suitable for image description and if so, we also need to determine the exact location of the keypoints.

2.2.2 Exact keypoint location

As stated above, this step takes a closer look at the candidate points for a purpose of exact extrema localization. We also want to eliminate points with low contrast values or points that lie near edges.

The location of the candidate point is approximated by second-order Taylor expansion of the difference-of-Gaussian function. This expansion has an origin at the candidate point x and is given by the following equation:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x, \tag{4}$$

where D function and its derivatives are evaluated at the candidate point location and $x = (x, y, \sigma)^T$ means a shift from this candidate point. Extremum $\hat{x}$ of this Taylor expansion corresponds to the exact location of the candidate point. It is

found by setting the function $D(x)$ derivative by x equal to zero. This gives us the following expression:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}. \tag{5}$$

We need to change the origin of Taylor expansion and repeat the computations for this new origin if the shift of $\hat{x}$ is greater than 0.5. If the shift of $\hat{x}$ is greater than 0.5 it means that the extremum we are looking for lies closer to the neighboring point. In the opposite case (the shift of $\hat{x}$ is smaller than 0.5) the shift of $\hat{x}$ is added to the candidate point location. This way we get a more exact location of the extremum.

If we substitute the expression for $\hat{x}$ computation into the Taylor expansion, we get the value of the extremum in the $\hat{x}$ location

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}. \tag{6}$$

We then take an absolute value $|D(\hat{x})|$ in this location and discard the candidate point if the value is smaller than 0.03 because the point has low contrast (considering brightness value of the image in the interval $[0, 1]$).

The difference-of-Gaussian function is also returning extrema that are located too close to the edges in the image. It is not easy to locate such points repeatedly and reliably and we need to discard them. We call such points poorly localized or poorly defined. Points that lie along the edges have large principal curvature and small perpendicular curvature. The eigenvalues of Hessian matrix $\mathbf{H}$ correlate with the principal curvature. Hessian matrix is a square matrix of second-order partial derivatives defined for the D function as follows:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}. \tag{7}$$

We can evade the costly computation of the eigenvalues by using their ratio. Suppose we have the eigenvalues α and β and suppose that α is greater than β. We can calculate the product of these eigenvalues from the trace of the Hessian matrix $\mathrm{Tr}(\mathbf{H})$ and we can calculate their sum from the determinant of the Hessian matrix $\mathrm{Det}(\mathbf{H})$ as

$$Tr(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta, \tag{8}$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \tag{9}$$

Let us now label the eigenvalues ratio as r, so that $\alpha = r\beta$. We can then substitute this expression into the expressions for the calculations of the trace and the determinant getting the following equation:

$$\frac{Tr\left(\mathbf{H}\right)^2}{Det\left(\mathbf{H}\right)} = \frac{\left(\alpha+\beta\right)^2}{\alpha\beta} = \frac{\left(r\beta+\beta\right)^2}{r\beta^2} = \frac{\left(r+1\right)^2}{r}. \tag{10}$$

This equations result is dependant only on the ratio of the eigenvalues, regardless of their real value. SIFT, therefore, computes only following comparison to determine the suitability of the points:

$$\frac{Tr\left(\mathbf{H}\right)^2}{Det\left(\mathbf{H}\right)} < \frac{\left(r+1\right)^2}{r}, \tag{11}$$

where r is a threshold declared by the user. If this inequality is not met, meaning the ratio of principal curvatures is greater than r, the keypoint is excluded.

2.2.3 Orientation assignment

The candidate points that went through the previous step of the exact localization are now considered as the keypoints. They are now invariant to image scale, but we also want them to be invariant to rotation. We assign every keypoint its orientation for the purpose of the rotation invariance.

First, we choose the scale-space image L with the closest scale to the examined keypoint. We also precompute gradient magnitude $m\left(x,y\right)$ and its orientation $\theta\left(x,y\right)$ for every point of this image $L\left(x,y\right)$. To do so, we use the difference in the brightness of the singular points

$$m\left(x,y\right) = \sqrt{\left(L\left(x+1,y\right)-L\left(x-1,y\right)\right)^2+\left(L\left(x,y+1\right)-L\left(x,y-1\right)\right)^2}, \tag{12}$$

$$\theta\left(x,y\right) = \tan^{-1}\left(\frac{L\left(x,y+1\right)-L\left(x,y-1\right)}{L\left(x+1,y\right)-L\left(x-1,y\right)}\right). \tag{13}$$

Then we construct an orientation histogram. We split the 360° circle into 36 sections giving us a histogram with 36 bins. The points near the examined keypoint are then added to this histogram depending on their orientation. Every point added is also weighted with its gradient magnitude $m\left(x,y\right)$ and also by the distance from the keypoint before the addition. This distance weight is Gaussian with σ equals 1.5 scale of the keypoint.

The highest peak of the histogram corresponds to the dominant keypoint orientation. We assign this orientation to the keypoint and also check if any of the other peaks reach at least 80% of this maximum. We construct one keypoint for each peak reaching this 80% threshold and assign it the dominant orientation according to the peak. This means that more keypoints with different orientations can be located at the same point in the image.

2.2.4 Descriptor creation

Our keypoints are now exactly localized and invariant to scale and rotation changes. We can now compute a descriptor describing the surroundings of the keypoint. We want it to be as invariant to brightness transformations and viewpoint changes as possible.

First, we once again choose the scale-space image with the closest scale to the keypoint. Then we divide the keypoint surroundings into 16 square regions, every region consists of 16 points. In the previous step, we calculated the magnitudes and orientations of the gradients. We use these for the construction of the descriptor. We also want to keep the descriptor invariant to the image rotation so we need to rotate the surroundings and the gradients according to the dominant orientation of the keypoint found in the previous step. All of the gradient magnitudes are weighted by Gaussian function with σ equal to one-half of the width of the descriptor window.

For each of the 16 square regions an orientation histogram is created. These orientations are then separated into eight bins of the histogram, this way we compute all 16 histograms. A descriptor is then composed as a vector with elements corresponding to the values in the histogram's bins. We need to include every bin of every histogram into this vector, getting 128 elements feature vector. To assure the invariance to brightness transformations, we need to normalize the vector to unit length.

2.3 KAZE

KAZE (meaning *wind* in Japanese) [5] is a method extracting image features from nonlinear scale-spaces. It was introduced by Alcantarilla et al. in the year 2012.

It uses nonlinear diffusion filtering for feature detection and description. The method argues that while it is relatively simple to obtain Gaussian scale-space, the Gaussian scale-space also does not respect the details and boundaries of the objects. The nonlinear scale-space approach on the other hand allows retaining object boundaries while smoothing the image. We briefly touch on the nonlinear diffusion filtering and then focus on the KAZE features.

2.3.1 Nonlinear diffusion filtering

Nonlinear diffusion is based on the perception of the evolution of the luminance of an image through scale levels as the divergence of a flow function. The divergence is a vector field operator that gives us information about the extent to which an infinitesimal point is a source. The flow function controls the process of diffusion. Given a conductivity function $c\left(x, y, t\right)$ that depends on the local image differential structure, we can use the nonlinear diffusion differential equation to make diffusion adaptive to the local image structure

$$\frac{\partial U}{\partial t} = div\left(c\left(x, y, t\right) \cdot \nabla U\right), \tag{14}$$

where U is the original image and ∇ denotes the gradient operator. Image conductivity function $c(x, y, t)$ used by KAZE is generally defined as

$$c(x, y, t) = g\left(\left|\nabla U_\sigma(x, y, t)\right|\right), \tag{15}$$

where the g function could be any of the following functions

$$g_1 = e^{-\frac{|\nabla U_\sigma|^2}{k^2}}, \tag{16}$$

$$g_2 = \frac{1}{1 + \frac{|\nabla U_\sigma|^2}{k^2}}, \tag{17}$$

$$g_3 = \begin{cases} 1 & , |\nabla U_\sigma|^2 = 0 \\ 1 - e^{-\frac{3.315}{(|\nabla U_\sigma|/k)^8}} & , |\nabla U_\sigma|^2 > 0 \end{cases}, \tag{18}$$

∇U_σ and k are described at the end of this Section.

KAZE uses the Additive Operator Splitting (AOS) schemes introduced in [6] to approximate the solution of the nonlinear diffusion filtering partial differential equation as there are no analytical solutions. The nonlinear diffusion equation is discretized as follows:

$$\frac{L^{i+1} - L^i}{\tau} = \sum_{l=1}^{m} A_l\left(L^i\right) L^{i+1}, \tag{19}$$

where L^i is the scale-space image at the scale σ_i and L^0 is equal to the original image U. A_l is an encoding matrix of the image conductivities. The encoding matrix A_l at position i, j is given as

$$a_{ij} = \begin{cases} \frac{g_i^k + g_j^k}{2h^2} & (j \in \mathcal{N}(i)), \\ -\sum_{n \in \mathcal{N}(i)} \frac{g_i^k + g_n^k}{2h^2} & (j = i), \\ 0 & \text{else}, \end{cases} \tag{20}$$

where $\mathcal{N}(i)$ is the neighborhood of i, h is the grid size and g_i^k is the gradient approximated by central differences. The central differences uses the image points u_i^k in time k and is computed as

$$g_i^k = g\left(\frac{1}{2} \sum_{p,q \in \mathcal{N}(i)} \left(\frac{u_p^k - u_q^k}{2h}\right)^2\right), \tag{21}$$

Combining all of the above equations, the solution of the L^{i+1} can be calculated as

$$L^{i+1} = \left(I - \tau \sum_{l=1}^{m} A_l \left(L^i \right) \right)^{-1} L^i. \tag{22}$$

Regarding the choice of the function g for the gradient approximation, KAZE experimental results have shown that the g_2 is the best in the sense of repeatability. The function ∇U_σ is the gradient of a Gaussian smoothed version of the original image U and k is the contrast parameter that determines which edges to cancel and which to enhance. The contrast parameter k is set as the 70% percentile of the gradient histogram of a smoothed version of the original image.

2.3.2 Nonlinear scale-space

We arrange the scale-space in a series of O octaves and S sub-levels. It is also important to know that KAZE works with the original image resolution without any downsampling. We label the octaves and sub-levels as o, respectively s and map the indices to their scale σ as follows:

$$\sigma_i \left(o, s \right) = \sigma_0 2^{\frac{o+s}{S}}, \tag{23}$$

where σ_0 is the base scale level. Index i is from the interval $[0...N]$, where N is the total number of filtered images.

We also need to map the scale σ_i to time units t_i in an order to acquire a set of evolution times and to be able to use the nonlinear diffusion filtering. This mapping is done as

$$t_i = \frac{1}{2}\sigma_i^2, i = 0...N. \tag{24}$$

To construct the scale-space, we convolve the image with a Gaussian kernel of a standard deviation σ_0. This is done to reduce noise and image artifacts. We also compute the contrast parameter k as the 70% percentile of the gradient histogram that is computed from this smoothed image.

We have the set of evolution times t_i and the contrast parameter k, so we can build the nonlinear scale-space by iteratively solving the following equation:

$$L^{i+1} = \left(I - (t_{i+1} - t_i) \cdot \sum_{l=1}^{m} A_l \left(L^i \right) \right)^{-1} L^i. \tag{25}$$

Having built the scale-space, we can now start detecting the keypoints.

2.3.3 Keypoints detection and descriptor creation

The points of interest are sought for in the nonlinear scale-space L^i at scale levels σ_i by the means of the response of a scale-normalized determinant of the Hessian

$$L_{Hessian} = \sigma^2 \left(L_{xx} L_{yy} - L_{xy}^2 \right), \tag{26}$$

where L_{xx}, L_{yy} and L_{xy} are the second order horizontal respectively vertical respectively cross derivatives. We search for the maxima in both scale and space except the images $i = 0$ and $i = N$. First we look for the extrema over 3×3 window of responses, then over a rectangular window of $\sigma_i \times \sigma_i$ on the neighbouring scales. The specific keypoint position is estimated with sub-pixel accuracy using the SIFT method from [1, 7] described in our Section 2.2.2.

To construct the descriptor, we first need to find the dominant orientation of the keypoint. The whole process of the descriptor construction is very similar to the one described in SURF [8]. The dominant orientation is found in a circular area of radius $6\sigma_i$ and the sampling step σ_i. Then the first order derivatives L_x and L_y are calculated and weighted with a Gaussian centered at the keypoint. Responses are then represented as vectors and all the responses in a window covering an angle of $\frac{\pi}{3}$ are summed. The window with the longest vector corresponds to the dominant orientation of the keypoint.

The Modified-SURF (M-SURF) descriptor is taken and adapted to the nonlinear scale-space. The first order derivatives L_x and L_y at the scale σ_i of the keypoint in a rectangular grid of size $24\sigma_i \times 24\sigma_i$ are calculated. This calculation and the samples in the rectangular grid are both respecting the dominant orientation of the keypoint. The grid is then split into 4×4 sub-regions of size $9\sigma_i \times 9\sigma_i$ with an overlap of $2\sigma_i$. The responses L_x and L_y in each sub-region are weighted with a Gaussian centered on the center of the sub-region with $\sigma = 2.5\sigma_i$. These responses are then summed into a descriptor vector $d_v = (\sum L_x, \sum L_y, \sum |L_x|, \sum |L_y|)$. Additionally, we weight each descriptor vector that we constructed with a Gaussian defined over a mask of 4×4 centered on the keypoint with $\sigma = 1.5\sigma_i$. The descriptor vector is normalized into a unit vector to be invariant to contrast.

2.4 AKAZE

Accelerated KAZE [2] was proposed in the year 2011 by Alcantarilla et al. to reduce the time consumption of KAZE. This time consumption is caused by operating in nonlinear scale space and Alcantarilla et al. introduce Fast Explicit Diffusion (FED) schemes into a pyramidal framework to speed up feature extraction.

2.4.1 Nonlinear scale space

As stated above FED schemes are utilized to build a nonlinear scale space. This drastically accelerates the process of building the scale space. FED schemes are also easy to implement. AKAZE performs M cycles of n explicit diffusion steps with

varying step sizes τ_j. These step sizes originate from the factorization of a box filter and are calculated as follows:

$$\tau_j = \frac{\tau_{max}}{2\cos^2\left(\pi \frac{2j+1}{4n+2}\right)}. \tag{27}$$

2.4.2 Keypoints detection and descriptor creation

To detect keypoints (features) the determinant of the Hessian for each image L^i in the nonlinear scale space is computed. Differential multiscale operators are then normalized with respect to scale. This normalization takes the octave of the image into account. Then AKAZE searches for maxima of the detector response over scale and space.

The descriptor used for the description of the detected features is a Modified-Local Difference Binary descriptor (M-LDB). This descriptor modifies LDB to use gradient and intensity information contained in the nonlinear scale space. The original LDB descriptor is based on the same principle as the BRIEF descriptor (described in Section 2.5.3), but it tests over the average of areas instead of single pixels. M-LDB also uses the mean of the horizontal and vertical derivatives in the areas. The main orientation of the keypoint is determined the same way as in KAZE and the LDB grid is rotated accordingly resulting in invariance to rotation. To also make the descriptor robust to scale changes the grids in steps are sub-sampled and a function of the keypoints scale σ instead of using the average of all pixels inside the grid subdivisions.

2.5 ORB

Oriented FAST and Rotated BRIEF [3] is an image description method proposed in the year 2011 by Rublee et al. This method is described by the authors as "an efficient alternative to SIFT or SURF". As suggested by its full name, it is based on two existing methods.

ORB uses the FAST [9] method for detection of the keypoints but alternates it to describe the orientation of the keypoint as well. For the descriptor construction ORB uses BRIEF [10] as a basis and further expands it.

2.5.1 Keypoints detection

As previously mentioned, ORB detects keypoints using FAST (Features from Accelerated Segment Test). FAST looks at a circle of 16 pixels around the candidate point with intensity I_p. It classifies this candidate point as a corner if there is a set of n contiguous pixels in the circle which are all brighter than $I_p + t$ or are all darker than $I_p - t$, t is given as a parameter. ORB uses $n = 9$ called FAST-9.

We are looking for N keypoints and this is done by setting a low enough threshold to obtain more than N keypoints. FAST has large responses along edges and also gives no information about a measure of "cornerness", so we need to decide which keypoints to keep. To obtain the best possible N keypoints from this set we order

the keypoints by a Harris corner measure described in [11]. After this ordering we simply do not consider the keypoints beyond N.

2.5.2 Orientation assignment

To assign an orientation to the found keypoints we use the intensity centroid. The intensity of a keypoint is supposed to be an offset from its center. This centroid is defined as

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right), \tag{28}$$

where m_{pq} are the moments of a patch calculated as

$$m_{pq} = \sum_{x,y} x^p y^q I\left(x, y\right), \tag{29}$$

where $I(x, y)$ is the intensity at coordinates (x, y)

We then assign an orientation to the keypoint that is the same as the orientation of the patch given by the following equation:

$$\theta = \text{atan2}\left(m_{01}, m_{10}\right), \tag{30}$$

where atan2 is the quadrant-aware arctan.

2.5.3 Descriptor creation

ORB descriptor is based on BRIEF (Binary robust independent elementary features). BRIEF descriptor describes an image patch with a set of binary intensity tests. The image is smoothed before performing the test. We smooth the image using an integral image. Each test point in this integral image is 5×5 sub-window of a 31×31 pixel patch. Test τ on an image patch p of size $S \times S$ is defined as

$$\tau\left(p; x, y\right) := \begin{cases} 1 & \text{if } p\left(x\right) < p\left(y\right) \\ 0 & \text{otherwise} \end{cases}, \tag{31}$$

where $p\left(x\right)$, respectively $p\left(y\right)$ are the pixel intensities of p at a point x, respectively y. BRIEF descriptor is a n_d-dimensional bitstring f_{n_d}

$$f_{n_d}\left(p\right) := \sum_{1 \leq i \leq n} 2^{i-1} \tau\left(p; x_i, y_i\right). \tag{32}$$

ORB uses a Gaussian distribution of tests around the patch center and a vector length $n = 256$.

2.5.4 Steered BRIEF and rBRIEF

To make BRIEF invariant to an in-plane rotation, BRIEF is steered according to the keypoint orientation. We define a matrix $\mathbf{S}$ for feature set at (x_i, y_i)

$$\mathbf{S} = \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{pmatrix}. \tag{33}$$

We then steer this matrix using rotation matrix $\mathbf{R}_\theta$ that corresponds to the patch orientation θ

$$\mathbf{S}_\theta = \mathbf{R}_\theta S. \tag{34}$$

The steered BRIEF operator is then calculated as

$$g_n\left(p, \theta\right) := f_n\left(p\right) | \left(x_i, y_i\right) \in \mathbf{S}_\theta. \tag{35}$$

We then discretize the angle to $2\pi/30$ and construct a lookup table of precomputed BRIEF patterns. The binary tests of steered BRIEF are correlated and its variance is also lower. Lower variance leads to lower discriminativeness of keypoints.

To recover from this loss of variance and binary tests correlation, rBRIEF is introduced. This approach takes all the possible binary tests eliminating these tests that overlap. This leads to 205590 tests from 31×31 pixel patch. The algorithm then runs each test against all training patches and orders them by their distance from a mean of 0.5. This way, we get the vector of tests T. The vector is then greedy searched.

The algorithm of the greedy search proceeds as follows: The first test is removed from T and put into result vector R. Then the next test is taken from T, compared against all tests in R and if its absolute correlation is lesser than a threshold, it is added to R. This step is repeated until there are 256 tests in R. If the algorithm runs out of tests and there are fewer than 256 tests in R, the threshold is raised and the remaining tests from T are compared again.

2.6 LATCH

Learned Arrangements of Three Patch Codes [4] is a feature description method that was first introduced in the year 2015 and presented in the year 2016 by Levi et al. A novel binary descriptor design was proposed that further improves on the shortcomings of other binary description methods such as stability and robustness.

The LATCH method builds on the idea of Local binary patterns (LBP) that were introduced in [12], [13], particularly on and idea of Three-Patch LBP (TPLBP) [14], [15]. LBP are focused on a description of the image as a whole while LATCH takes the TPLBP ideas and applies them on descriptions of sole keypoints.

Since LATCH is only a description method, we suppose that we have the keypoints in the image already detected by some other method.

2.6.1 Descriptor creation

We start with the keypoints detected and we define window W of fixed size that is centered on a keypoint. Lets assume we have $t = 1, ..., T$ pixel patch triplets. We then define an ordered set S of the patch triplets with an anchor patch $p_{t,a}$ as

$$S = \{s_t\}_{t=1,...,T} = \{[p_{t,a}, p_{t,1}, p_{t,2}]\}_{t=1,...,T}, \tag{36}$$

where $p_{t,a} = (x_{t,a}, y_{t,a})$, $p_{t,1} = (x_{t,1}, y_{t,1})$ and $p_{t,2} = (x_{t,2}, y_{t,2})$ are the coordinates of the central pixels for the anchor patch $\mathbf{P}_{t,a}$, respectively "companion" patches $\mathbf{P}_{t,1}$ and $\mathbf{P}_{t,2}$. These patches are of size $k \times k$ pixels. For each anchor patch $\mathbf{P}_{t,a}$ we calculate the Frobenius norm with its companions. This gives us a following binary function:

$$g\left(W, s_t\right) = \begin{cases} 1 \text{ if } ||\mathbf{P}_{t,a} - \mathbf{P}_{t,1}||_F^2 > ||\mathbf{P}_{t,a} - \mathbf{P}_{t,2}||_F^2 \\ 0 \text{ otherwise} \end{cases} . \tag{37}$$

We then compute a binary string b_W for the window W as

$$b_W = \sum_{1 \le t \le T} 2^t g\left(W, s_t\right). \tag{38}$$

LATCH also introduces a novel selection criteria for selecting the best triplet arrangements. This needs to be done because the number of possible arrangements is huge even for small detection windows W.

The arrangements are learned on the dataset that was introduced in [16]. The dataset contains three collections and each collection contains 400k local image windows. Pairs of the windows are labeled as "same" or as "not-same". This depends on whether or not they represent the same physical point in the image. Afterwards, the windows are divided into 500k pairs. Half of those pairs are labeled as "same" and half as "not-same".

56k patch arrangements are then formed by random selection of the anchor patch $p_{t,a}$ and its two companion patches $p_{t,1}$, $p_{t,2}$ with $(t = 1...T = 56000)$. These arrangements are then evaluated over all the window pairs in the collection, yielding 500k bits per arrangement. We evaluate the number of times the arrangement correctly returned the same binary value for the "same" pairs. The corresponding number is also evaluated for the "not-same" pairs in the regard of correct different binary values returned. The quality of the arrangement is given by the sum of the two numbers.

This selection may yield highly correlated arrangements. The arrangements are added incrementally. This allows us to skip over the responses that are highly correlated to previously selected arrangements. A candidate arrangement is select if its absolute correlation with all previous arrangements is smaller than a threshold τ. Threshold $\tau = 2$ is used in LATCH.

3 Keypoint matching and evaluation methods

Having described the methods used to create the image descriptors, let us now cover the fundamentals of keypoints' descriptors matching. There are multiple approaches one can take to match the descriptors.

The simplest approach is the Brute-Force matcher. It takes each descriptor from the first image and matches it with all of the descriptors from the second image using a distance calculation. An example of a distance one could use is the L_2 Norm calculated as

$$||d_1 - d_2||_{L_2} = \sqrt{\sum_{i=1}^{N} \left(d_1\left(i\right) - d_2\left(i\right)\right)^2}, \tag{39}$$

where d_1, d_2 are the descriptors of the two images, N is the dimension of the descriptor and $d_1\left(i\right), d_2\left(i\right)$ are the i-th elements of the descriptors. Other distance metrics can also be used (the Hamming distance, L_1 Norm).

We can use the k-nearest-neighbors approach to find k keypoints that have the lowest distance to the examined keypoint. If we set $k = 2$ we can use the ratio test described by Lowe in [1]. The ratio test states that a keypoint can have only one match. This leads to an assumption that the match with the smaller distance is the "good" match and the second nearest match can be perceived as noise. According to Lowe, if the distance of the "good" match is greater than the distance of the noise multiplied by a set threshold, we need to reject the "good" match because it is not easily distinguished from the noise.

Other more sophisticated approaches like these contained in the Fast Library for Approximate Nearest Neighbors (FLANN) implemented in OpenCV [17] operate with decision trees and can also use the Lowe ratio test.

We use two evaluation metrics to evaluate the matched images - the mean average precision and the receiver operating characteristic curve (ROC curve). Let's now dedicate the following Sections to the description of these two methods.

3.1 Mean average precision

Our experiment is generally an information retrieval task, more specifically it can be viewed as an image retrieval task. In the image retrieval task, we have some query image U and we are asking if it or its derivatives are contained in our set of stored images $Y_1, ..., Y_N$. If that is the case, we want to know what image Y_i matches the query.

For image retrieval we can measure precision as a number of images retrieved relevant to the query divided by a number of images retrieved as stated in [18]. If we suppose that we retrieve all images sorted by their agreement rate, we can then calculate precision P_i for matched image at position i as follows:

$$P_i = \frac{tp}{i}, \tag{40}$$

where tp is the number of relevant images found so far (if i-th image is relevant, then it is included into the tp).

Having the sorted (ranked) matched images, we can also calculate the average precision for the query as described in [19]. We need to assign relevance rel_i to each match at position i defined as

$$rel_i = \begin{cases} 1 \text{ if match } i \text{ is relevant to our query,} \\ 0 \text{ otherwise.} \end{cases} \tag{41}$$

We now use this relevance for the calculation of the average precision AP for the query. It is calculated as follows:

$$AP = \frac{1}{N} \sum_{i=1}^{N} P_i \cdot rel_i, \tag{42}$$

where N is the number of images retrieved by the query. If the average precision is equal to one, we know that the results for the query were sorted perfectly and all of the desired matches were at the foremost positions.

If we now compute the average precision AP_j for M queries we can calculate the mean average precision mAP over all queries as

$$mAP = \frac{1}{M} \sum_{j=1}^{M} AP_j, \tag{43}$$

giving us one number that determines the ability of the methods to distinguish different images.

3.2 Receiver operating characteristic curve

Another task that we can define within our experiment is a search for the best binary classifier with the optimal threshold setting. In our task, this threshold is represented by the relative amount of matched descriptors needed to find a good match. Such a classifier can be evaluated by the receiver operating characteristic curve. This name originates from the radar receivers for which the method was first developed.

The ROC curve is obtained by plotting the sensitivity (true positive rate) against the fall-out (false positive rate) at some threshold settings as described in [20]. The abbreviation can also be interpreted as relative operating characteristic as per [21]. This better represents the comparison of the true and false positive rates at some threshold value.

The true positive rate tpr is generally calculated as

$$tpr = \frac{P_F}{P_R}, \tag{44}$$

where P_F is the number of true positive cases found by the classifier and P_R is the number of real positive cases in the data.

The false positive rate fpr is calculated in the same way using the number of real negative cases in the data in the denominator.

The number of found positive cases changes with some threshold t, making both rates functions of t: $tpr\,(t)$, $fpr\,(t)$. We can then iteratively change the threshold parameter t and plot the resulting curve.

Because of the true and false positive rate domains, the space in which the curve is defined ranges from values zero to one on both axes. The ideal classifier with optimal setting would achieve true positive rate of one while maintaining the false positive rate of zero meaning it would intersect the upper left corner of the graph.

4 Dataset

As stated in the Introduction, we want to construct our dataset of images made to fit our experiment. The reason why this is desired is to have control over the transformations used on the images. We want to have knowledge of the source image for each generated image.

This approach allows us to test the robustness of the methods over various separated groups of transformations and to conduct the experiment as efficiently as possible. We can also utilize uncommon transformations such as these implemented in image editing software. This way we imitate the effect of somebody taking some image that they did not capture and editing it on their phone or computer.

Let us now focus on the fundamental parts of the dataset creation process.

4.1 Ground truth

To test the description methods properly, we need to proceed with caution during the ground truth images selection. The ground truth images should consist out of multiple subsets of images. These subsets should capture the same object from almost the same angle and should ideally be captured by different authors.

For the identical objects in the images, e.g.: natural monuments, buildings, etc. our goal is to have each image differ at the angle and the distance from which it was captured. We demonstrate this at two chosen ground truth images in Figure 3.

For images that capture different objects of the same kind, e.g.: animals, cars, etc. our goal is to have the object captured from almost the same angle. These images should also have a background that is not very distinctive. We demonstrate this at two chosen ground truth images in Figure 4.

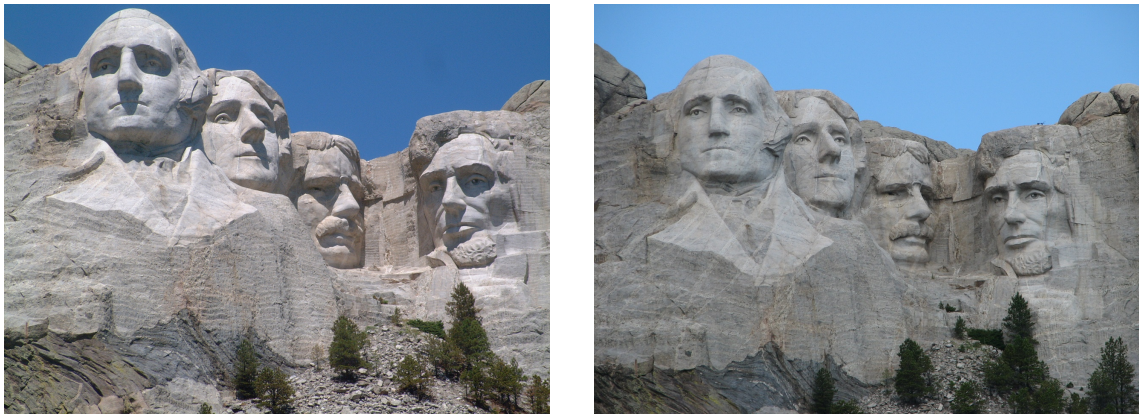


Figure 3: Ground truth images for the identical objects in the image scene.[2]

Figure 4: Ground truth images for the objects of the same kind in the image scene.[3]

4.2 Generation

Now that we defined our ground truth, we also want to define the approach to the generation of the output images.

Our output image Y is generated by the application of some transformation $T(U)$, where U is the source image. This can be also written as

$$Y = T(U). \tag{45}$$

We do not want to limit ourselves to only one transformation at a time. Generally speaking our output images Y are given as follows:

$$Y = (T_n \circ T_{n-1} \circ ... \circ T_2 \circ T_1)(U), \tag{46}$$

[2]Left image in Figure 3 by Seth Werkheiser published in 2007 on Flickr, right image in Figure 3 by Kurt Magoon published in 2009 on Flickr.

[3]Left image in Figure 4 by Magnus Johansson published in 2017 on Flickr, right image in Figure 4 by Beau B published in 2009 on Flickr.

where U is the source image and $T_i, i = 1...n$ are the image tranformations. The $\circ$ operator denotes the function composition that is defined as $(T_i \circ T_j)(U) = T_i(T_j(U))$.

4.2.1 Transformation categories

We have defined our transformation task in general. Let us now be more specific about transformations T_i.

Our transformations T_i used to generate the dataset are separated into six different categories. These categories are noise, blur, histogram, color, local brightness, and geometric. Each of the image transformations belongs in one category only. We now specify what kinds of transformations belong to each category. For each transformation kind inside the category, we list the specific transformations. Their application parameters and the parameters that can be set for each of the transformations are described in Appendix B.

We start with the **noise category**. Transformations in this category make the image noisy in a variety of different ways. Three of them add pure noise (additive white noise, additive salt and pepper noise, and iso noise). Then we have two image transformations that make the image noisy by lowering its quality (image compression to jpeg format and posterization). The last transformation either sharpens or blurs the image around the edges and corners (sharpness).

The **blur category** consists of five blur transformations (averaging blur, Gaussian blur, median blur, motion blur, and glass blur). They all blur the image in a slightly different way and their blur kernel sizes are randomly chosen from the set interval.

The **histogram category** consists of two methods of histogram equalization that are set to be mutually exclusive (histogram equalization and contrast limited histogram equalization).

The next category is the **color category**. As the name suggests this category consists of methods that change the brightness values of the different color channels. Four of the transformations are purely for changing the color of the output image (channel drop, channel shuffle, color invert, and sepia effect). Two transformations alter the image contrast (gamma correction and black point). The last transformation can alter the brightness, contrast, saturation, and hue values of the image (color jitter).

The penultimate category is the **local brightness category**. Two of the transformations lower the brightness on a set area of the image (vignetting and shadow over part of the image). The other two transformations lighten a circular area of the image and are set to be mutually exclusive (random light point and random sunflare).

The last category is the **geometric category**. All of the transformations in this category change the geometric properties of the image (resize, rotate, crop, flip, change of perspective, and grid distortion).

4.3 Generated datasets

We use the defined transformations and generate multiple smaller dataset groups with varying properties to test the description methods thoroughly. As we said in Section 4, this is one of the advantages of generating the dataset ourselves.

We construct one output group for each transformation category from Section 4.2.1. In these six groups (noise, blur, histogram, color, local brightness, and geometric group) only the transformations from the corresponding category are used.

We also construct one group that can use any of the transformations that we described in Section 4.2.1, this group is called the mixed group. It uses the default configuration set by our configuration file. This default configuration can be seen in Appendix B. It uses all of the available transformations hence the name mixed group.

One more group is constructed, and we call that group the enhance group. As stated in the Introduction, the motivation behind our experiment is driven by copyright issues. This group takes our motivation into account and focuses on transformations that one could make in some image editing software to make the image look better.

The examples of images generated by all of these eight categories can be seen in Appendix A.

The description methods claim that they are robust across various image scales. We also want to put this claim to the test. We construct additional datasets that consist of images with modified resolution. The resolution ranges from 0.1 to 1.2 of the original image resolution with step size 0.1. This leads to twelve more datasets.

4.4 Technical details

For the generation of the dataset, we selected 50 ground truth images from various environments. The ground truth images consist out of subgroups of images capturing the same or identical objects, there are three or four images per subgroup.

The implementation of the transformations was done in the Python programming language. We implemented all of the 32 transformations mentioned in Section 4.2.1 with the help of the libraries Albumentations and Pillow. All of the transformations used from the libraries needed to be unified across our script to produce consistent output. We also implemented some of the transformations from scratch with the help of the NumPy library.

The application of the transformations is fully configurable in YAML configuration file. Almost every method has some configurable parameters and one example of the configuration file is in the aforementioned Appendix B.

We used the transformations and the configuration to generate eight dataset groups of the output images - noise, blur, histogram, color, local brightness, geometric, mixed, and enhance group. Each group consists of 200 images generated from our ground truth. The description of the configurations used for the generation of the groups would be too exhausting because of the large number of parameters being set. The example in Appendix B is the configuration used for the mixed dataset group.

We also generated twelve groups consisting of 50 images each. These images differ from the ground truth images only in their resolution as described at the end of Section 4.3.

5 Experiments

In this chapter, we briefly discuss the setup of our experiment and the results obtained.

We test the methods described in Section 2. Before we begin, let us explain why we chose the methods.

SIFT was chosen as a representative method for the use of the Gaussian scale-space and the use of the image measurements to store the information about the image (e.g.: gradients). While being computationally slower than the alternatives, it still shows results comparable to the other methods as shown in [22].

AKAZE was chosen as a representative method for non-linear scale-space construction. KAZE is omitted from the testing because AKAZE is both computationally faster and produces features of higher quality than its predecessor as was stated in [22].

ORB is a method that is able to produce higher quantities of features while still being faster in detection and description. It is representative of binary feature description.

LATCH is another method based on the binary feature description. It also introduces a novel concept of using triplets of patches for the descriptor construction. The descriptor takes the geometrical arrangement of the keypoints into account.

5.1 Experimental setup

First, we generate the dataset groups described in Section 4.3 and 4.4. We save the information about the source image for each generated image. This information is used later on.

After that, we use all of the mentioned methods to produce descriptors for each image. We use the OpenCV library [17] implementation of the methods. SIFT and AKAZE are used with the default OpenCV settings. The number of descriptors for ORB was chosen as an average number of descriptors produced by SIFT and AKAZE for each dataset group. We are using the average per group to acknowledge the amount of information contained in the images of said group. We base the decision on the assumption that if the group contains images with a smaller number of discriminative features, the extra features detected by ORB could bring in unsolicited noise. The decision is further supported for the other extreme case where some group of images exhibits much greater than the average number of features across all of the groups. If we used a global average for this particular group, ORB could underperform because it would lack the information that the other methods were able to extract. LATCH is a method that produces only the descriptors of the detected keypoints. Therefore as a detector for LATCH we use SIFT.

Having the image descriptors, we can now use the OpenCV FLANN based matcher to compare the descriptions. The matcher has the same setting for all the methods to eliminate any bias that could be brought in. The settings of the matcher are taken from [23]. The matcher then tries to match the descriptors of the source image and the descriptors of all the output images produced by the same method using the course of action described in [23]. For each pair (source image - output image) we retain only a number of matched descriptors. This number of matched descriptors is used for further evaluation. We also sort the number of matches in descending order.

We conduct two separate evaluations. The first evaluation regards strictly the robustness of the methods to the image resolution changes. For this task we generated the resolution groups mentioned in Sections 4.3 and 4.4. The second evaluation tests the methods against the remaining groups defined in the same Sections.

For the first evaluation task, we use the ranked matched descriptor count for each resolution group to calculate mean average precision as per Equations (40),(42),(43) within each method. We then plot the mAP value of the methods against the corresponding resolution rate and evaluate the methods' success rate.

For the second evaluation task, we also use the ranked numbers of matches for each of the remaining groups and calculate the mAP value for each method.

Before we continue, let us define a true positive case and a false positive case of an image match. A true positive case is when the source image is correctly matched with its output image. A false positive case is when the source image is incorrectly matched with an output of another source image. For the decision of whether the image is matched correctly or not the information about the source images saved during the dataset generation is used.

We can now use the number of descriptors for the source image U denoted as D_U to calculate threshold t_i for each output image Y_i as

$$t_i = \frac{M_{Y_i}}{D_U}, \tag{47}$$

where M_{Y_i} is the number of matched descriptors between the source image and the output image Y_i. For each method we calculate its own set of thresholds for each source image. We then group all of the thresholds corresponding to one method together.

Within each method, we iterate over the thresholds to get the true positive and positive rates per image within the threshold. This is done by calculating the true positive rate for each image as per Equation (44), where the number of true positive cases is the count of matches that have the number of matched descriptors greater than $t_i \cdot M_{Y_i}$. We also know the number of real positive cases in the data, because we generated the dataset groups ourselves. False positive rate fpr is calculated in the same way using the number of false positive cases found as the numerator and the number of real negative cases as the denominator. The number of real negative cases is equal to the difference between the number of output images and the number of real positive cases.

We then take the rates tpr_i, corresponding to source images X_i, $i = 1, ..., N$, where $N = 50$ is the number of our source images. We can then calculate average

true positive rate *atpr* for each threshold within the method as

$$atpr = \frac{1}{N} \sum_{i=1}^{N} tpr_i. \tag{48}$$

The average false positive rate *afpr* per threshold is calculated in the same way.

We then use the calculated average rates for the thresholds within the methods to plot the ROC curves as per Section 3.2. This calculation is done for every group leading to eight graphs.

5.2 Experimental results

Having specified the setup and approaches used during the experiment, let us now present the results.

5.2.1 Robustness in change of resolution

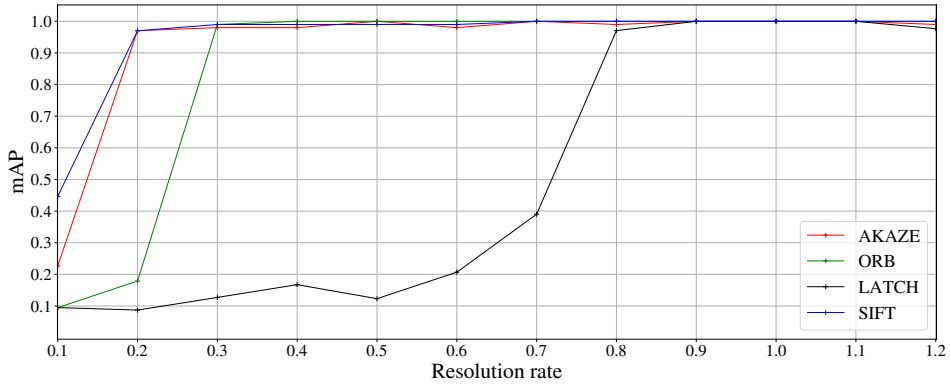
As we can observe in Figure 5, all of the methods are somewhat prone to a big resolution change. SIFT is the method that was the least affected by the resolution drop to 0.1 starting at mAP value of 0.446. AKAZE, ORB, and LATCH were all greatly affected by this change of resolution. Generally speaking, this lowest resolution hindered the scale-space based methods much less than it did the binary methods but all of the methods showed very poor performance.

The superiority of the scale-space based methods at low resolution rates got confirmed at the resolution rate of 0.2, where the scale-space based methods quickly rose in their precision reaching mAP values of 0.97 for both SIFT and AKAZE. ORB recorded this precision rise later than SIFT and AKAZE at the resolution rate of 0.3.

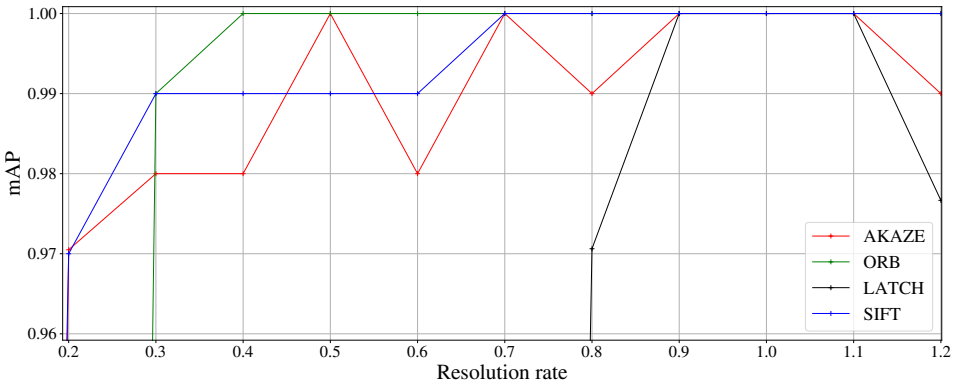
LATCH is the most affected method by the changes of resolution. Its mAP values started dropping at only 0.8 resolution rate and recorded a huge drop at lower rates.

At the resolution rates ranging from 0.3 to 1.2 the AKAZE, ORB, and SIFT methods are comparable in their results which show their robustness to image size changes. LATCH on the other hand shows these comparable results to AKAZE, ORB, and SIFT only at rates ranging from 0.8 to 1.2 meaning that it is very prone to changes in the image size.

If we look at the Figure 5b, we can see that while being more robust to bigger changes in resolution, SIFT does not reach the 1.0 mAP value until 0.7 resolution rate. AKAZE is also somewhat "unstable" in this regard, jumping from mAP value 1.0 back to 0.98 and 0.99 respectively. From the resolution rate of 0.3, ORB outperforms both of these methods and shows very stable results.



(a) The overall mAP values.



(b) The mAP values zoomed at the mAP interval (0.96, 1.00).

Figure 5: The mAP value of the methods for different resolutions of the ground truth images.

5.2.2 ROC curves across the dataset groups

As stated in Section 5.1 for each of the eight groups we calculated the average true and false positive rates for different threshold settings. We used these values for the ROC curves construction.

Let us start by evaluating the dataset groups corresponding to the transformation categories first.

Noise dataset ROC curve can be seen in Figure 6. All of the methods show large robustness to noise. LATCH is the worst-performing method in this regard but still shows great results reaching the true positive rate of 0.98 while maintaining the false positive rate of 0.

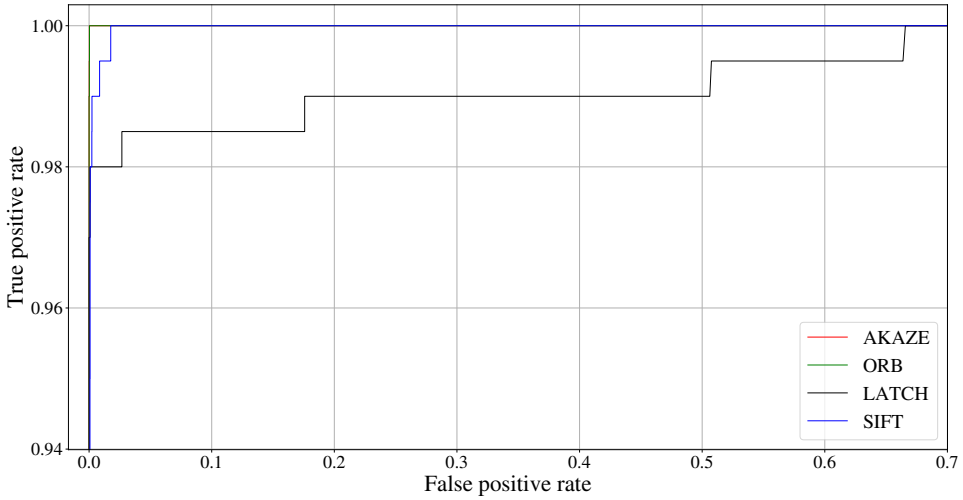


Figure 6: ROC curve for the noise dataset group.

While still performing very well, SIFT and LATCH have both shown slight susceptibility to image blurring. This is visible especially when compared to AKAZE and ORB that are very robust to blurring. This can be seen in Figure 7 depicting the ROC curve for the blur dataset group.

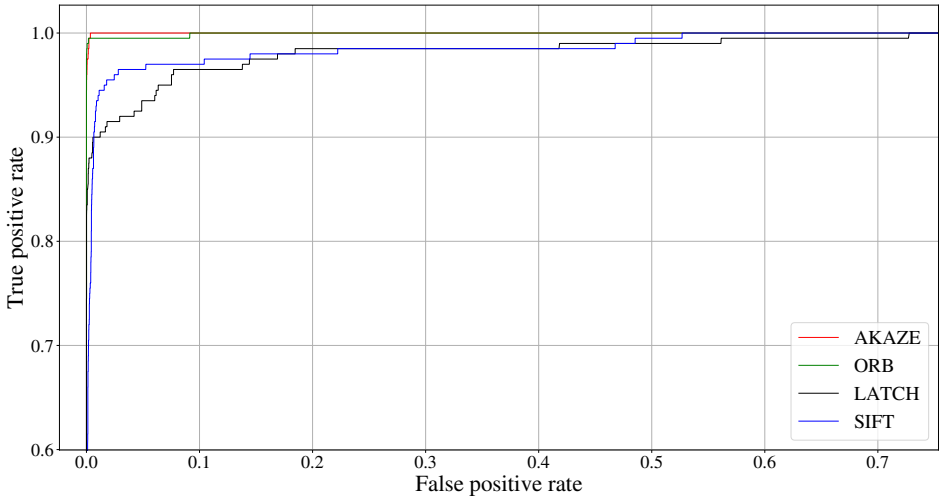
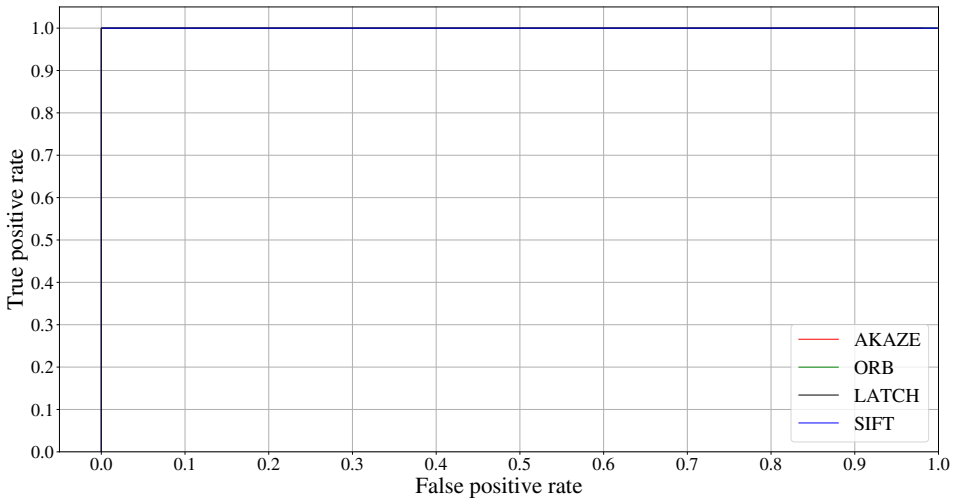


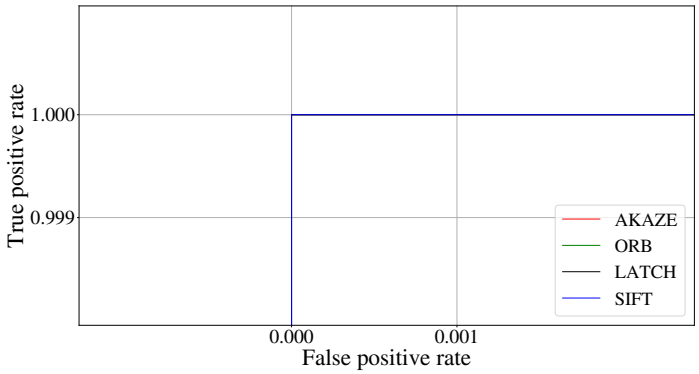
Figure 7: ROC curve for the blur dataset group.

Figure 8 shows the methods ROC curves for the histogram dataset group. All of the tested methods are completely robust to the used kinds of histogram equalization. This robustness can be seen in the ROC curves. They achieve the true positive rate 1 while maintaining the false positive rate 0 as per Section 3.2. This is further depicted on the zoomed view in Figure 8b.

This is the expected result because histogram transformations do not change the order of the pixel brightness in the images. Hence for the methods, the resulting image is the same as the source image.



(a) ROC curve for the histogram dataset group.



(b) Zoomed ROC curve for the histogram dataset group.

Figure 8: ROC curves for the histogram dataset group.

All of the methods except LATCH are mildly susceptible to color shifts and image-wide brightness changes. This can be seen in Figure 9 which depicts the ROC curves of the methods for the color dataset group.

LATCH shows nearly perfect results and a nearly perfect curve, this could mean that LATCH is very robust when there are no changes in the geometry of the image. Because of this, LATCH slightly outperforms the other methods. Other methods show very comparable and good results. AKAZE is the better performing method that reaches the true positive rate of 1.0 before reaching the false positive rate of 0.8. ORB on the other hand is not able to reach the true positive rate of 1.0 before reaching false positive rate of 1.0.

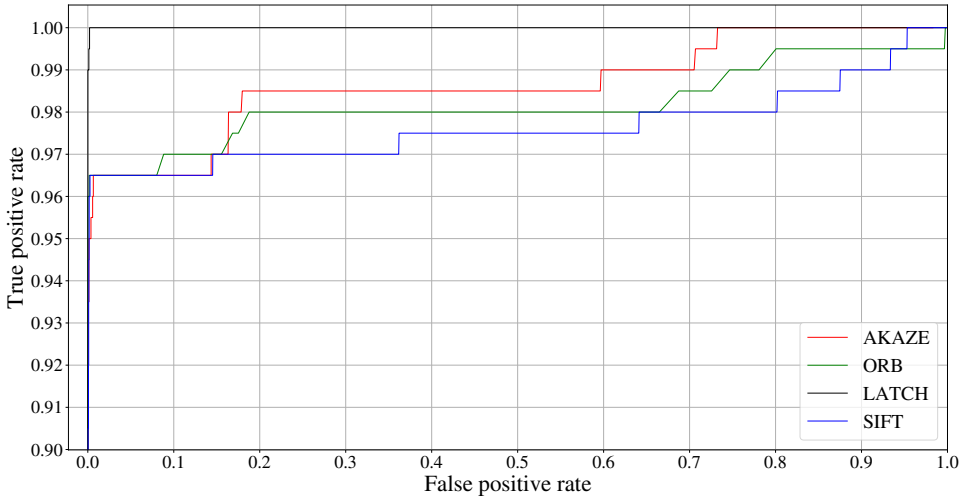
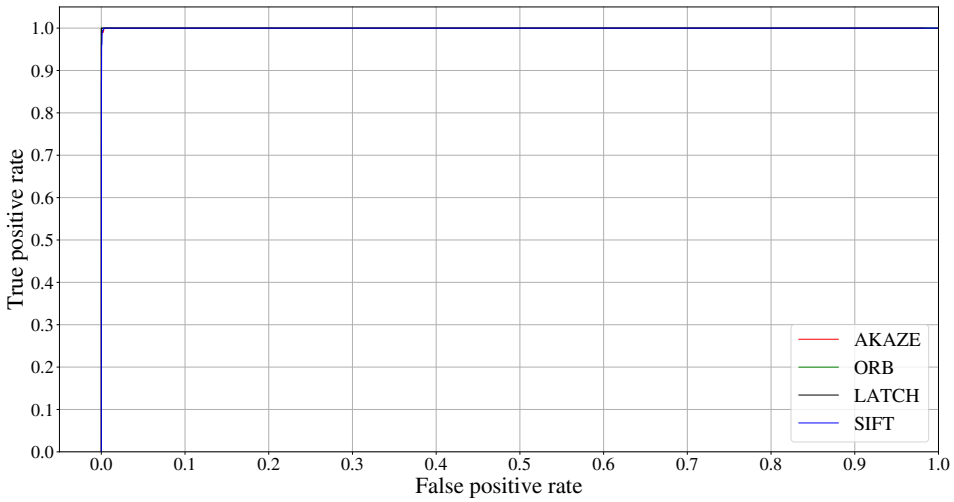


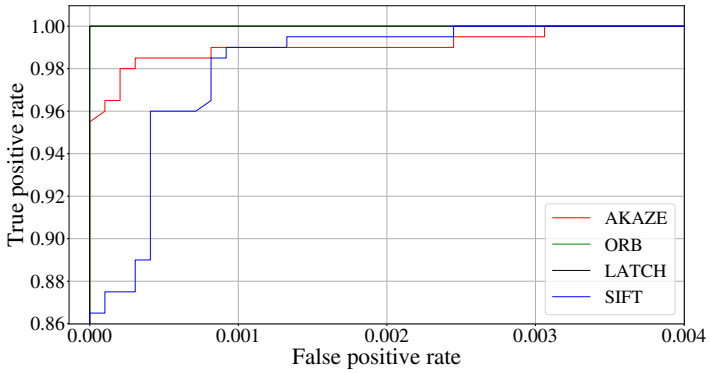
Figure 9: ROC curve for the color dataset group.

The methods' response to local brightness adjustments is comparable to the response to histogram equalization. All of the methods yield an almost perfect ROC curve as shown in Figure 10.

If we look at the zoomed curve in Figure 10b, we can see that ORB and LATCH are the methods that yield the perfect curve. AKAZE shows slightly worse performance and SIFT is the worst performing method. But all of the methods reach the true positive rate of 1.0 before even reaching the false positive rate of 0.004 which is a great performance.



(a) ROC curve for the local brightness dataset group.



(b) Zoomed ROC curve for the local brightness dataset group.

Figure 10: ROC curves for the local brightness dataset group.

The geometric transformations category seems to be the most disruptive transformation category. If we focused on the lowest possible false positive rate while maintaining a solid (around 0.8) true positive rate, SIFT is the method that surpasses the other methods. If we allow higher false positive rates than 0.1 then AKAZE gets to the point where it performs better than SIFT. ORB is slightly worse than the other two methods and LATCH's ability to correctly match images is impaired greatly. This is depicted in Figure 11.

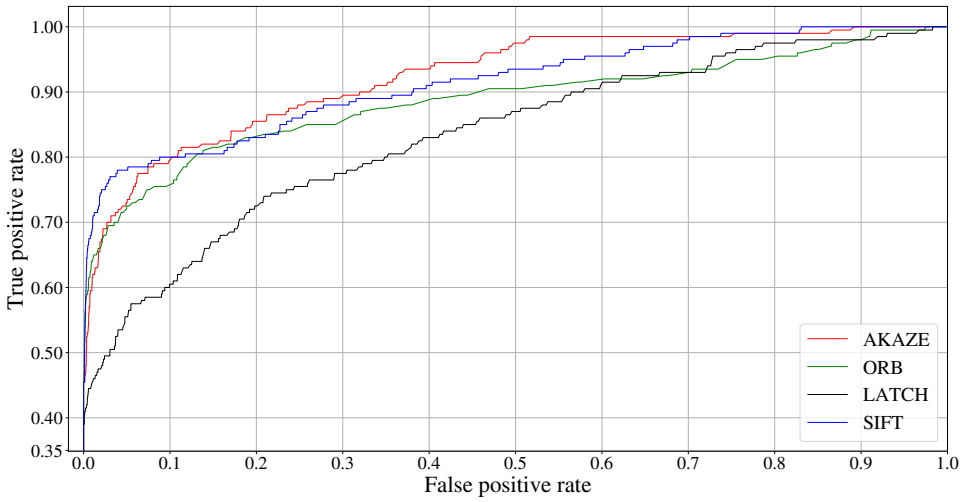


Figure 11: ROC curve for the geometric dataset group.

Let us now look at the two dataset groups that employ multiple transformation categories at once - the mixed dataset group and the enhance dataset group.

The course of the ROC curves for the mixed dataset group depicted in Figure 12 is very similar for all of the methods. LATCH is the worst performing method and it is the only method that is significantly worse than the others. This is likely caused by the employment of the geometric transformations. The best performing method is AKAZE followed very closely by ORB.

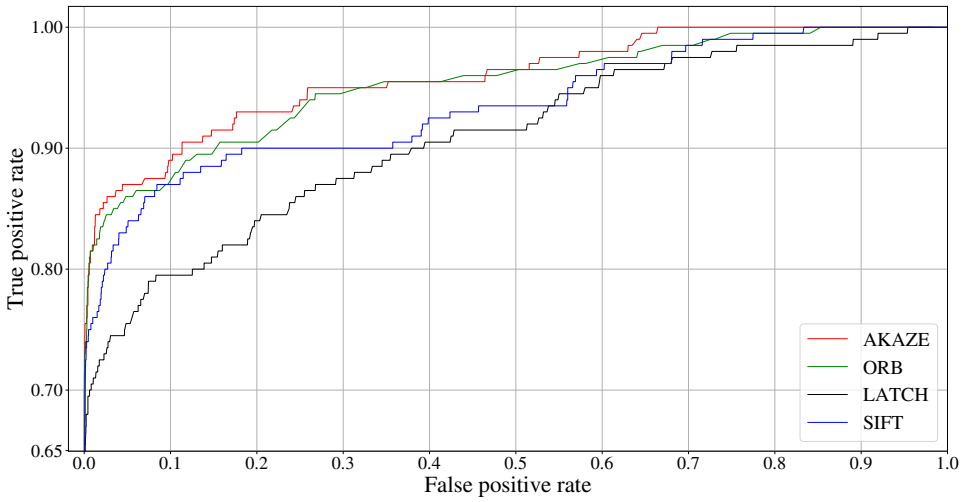


Figure 12: ROC curve for the mixed dataset group.

A similar curve shape can be observed for the enhance dataset group in Figure 13. The geometric transformations used in this group are not deforming the image as in the other groups. This leads to improved performance of LATCH compared to the previous group. It is still the worst-performing method. The other methods are comparable with AKAZE pulling ahead for greater false positive rates. At the false positive rate of 0.5 AKAZE shows approximately 0.05 improvement in the true positive rate over SIFT and ORB.

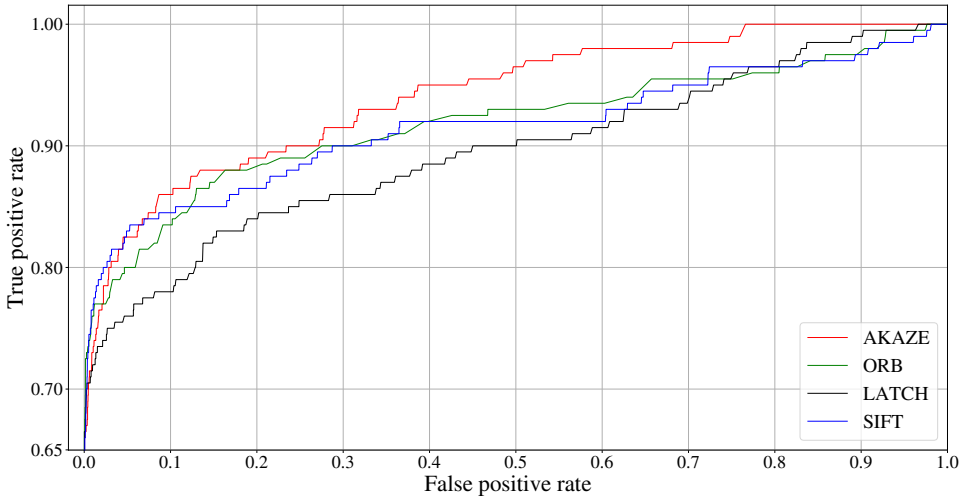


Figure 13: ROC curve for the enhance dataset group.

5.2.3 mAP values accross the dataset groups

For each of the groups mean average precision score over 50 queries was calculated for every method. These mAP values can be seen in Table 1.

The mAP values give us a closer look at the efficiency of the methods. As we can see, the results correspond to the results obtained by the means of ROC curve analysis. The methods show perfect or almost perfect success rates for the noise, blur, histogram, and local brightness transformation categories. This further supports our previous statement of the methods being robust to these transformations.

For the color category, the methods show very solid results but we can see they are somewhat affected by the transformations. LATCH is the only method that reaches the perfect mAP value. This shows the strength of the method in the field of transformations where geometrical properties of the keypoint triplets are left untouched.

The slight weakness of SIFT and LATCH to blur transformations that we observed in the previous Section is further confirmed by their mAP values. While still reaching good mean average precision they are performing worse than AKAZE and ORB. This may be caused by the further smoothing employed by SIFT that further blurs already blurred edges and corners. LATCH is affected by this too, probably because we used SIFT for the detection of the keypoints.

All of the methods suffer from the change of geometric properties of the images. SIFT outperforms the other methods regarding the mean average precision value. While the values of AKAZE and ORB are comparable, LATCH shows very poor performance. We have already shown multiple times that LATCH is not at all robust to any change of the geometric properties.

AKAZE has the best mean average precision of all the methods in the mixed dataset group, and SIFT has the best mean average precision in the enhance dataset group. LATCH is the worst performer in both of these groups, presumably because

of the geometric transformations present. What we want to point out is the stable and very good performance of ORB in both of these groups. ORB shows results very close to the best methods for the particular group nearing the results of AKAZE for the mixed dataset group and results of SIFT for the enhance dataset group.

mAP score	AKAZE	ORB	LATCH	SIFT
Noise group	1.0	1.0	0.9880	0.9857
Blur group	0.9980	0.9978	0.9699	0.9475
Histogram group	1.0	1.0	1.0	1.0
Color group	0.9683	0.9746	1.0	0.9636
Local brightness group	0.9979	1.0	1.0	1.0
Geometric group	0.7533	0.7411	0.5695	0.8615
Mixed group	0.8706	0.8592	0.7650	0.8181
Enhance group	0.7975	0.8146	0.7546	0.8260

Table 1: Table of mAP scores for the individual groups and methods.

5.2.4 Speed of the methods

Now that we have presented the results regarding the efficiency of the methods, let us take a look at their speed. We have not measured the time spent on detecting the keypoints so let us have a look at Levis LATCH paper [4]. If we take a look at Table 2, we can see that AKAZE is one order of magnitude faster than the binary methods ORB and LATCH and two orders of magnitude faster than SIFT. Note that this measurement regards only the descriptor extraction and not the detection of the keypoint.

Method	Running time (ms)
AKAZE	0.069
ORB	0.486
LATCH	0.616
SIFT	3.29

Table 2: Time measured in miliseconds for extracting a single local patch descriptor (presented in [4]).

We also measured the time taken to match the ground truth image set ($n = 50$) with the mixed dataset group ($n = 200$). This measurement was conducted on a machine with a processor Intel Core i5-4460 3.20Ghz, 16GB usable RAM, and running Windows 7 64-bit. As we can see in Table 3, ORB is slightly faster than AKAZE and LATCH. SIFT is by far the slowest descriptor type to match.

Method	Running time (s)
AKAZE	11611.3
ORB	9176.6
LATCH	13689.8
SIFT	59436.2

Table 3: Time measured in seconds for matching the ground truth image set with the mixed dataset group.

5.2.5 Evaluation of the results

Let us now evaluate the performances of the methods across the datasets.

AKAZE has shown great experimental results. As we showed in Section 5.2.1, it is very robust to change in image resolution. It also shows great results regarding the most transformation categories and good results if there are multiple categories applied as per Sections 5.2.2, 5.2.3. Where it may be lacking in the terms of performance it compensates with its speed.

ORB, similarly to AKAZE, shows solid and stable performance across the board. It is slightly less robust to changes in image resolution, catching up to SIFT or AKAZE later as shown in Section 5.2.1. We have also shown, that while not always being the best performing method, it is almost always among the top two performing methods. This can be seen in Section 5.2.3. In the combination with the computational cost and running time, this stability makes ORB a great method that should perform well in environments with diverse transformations applied.

LATCH shows great performance regarding the transformations that change the brightness values and color of the image. It has been shown that LATCH is very prone to any sort of geometrical transformations and resolution changes greater than 0.8 times the original resolution make LATCH practically unusable. The running time of LATCH is comparable with that of ORB making LATCH one of the faster methods.

SIFT behaves similarly to AKAZE in all of the experiments. It is the most robust method regarding the image resolution change having acceptable performance even at 0.1 times the original resolution. It is a very robust method and while not always the best in certain categories, its mean average precision never dropped below 0.8 in any of our experiments making it a very stable descriptor (as shown in Section 5.2.3). The big downside of SIFT is the computational cost and the running times, which are making the method inapplicable in any sort of real-time application.

6 Conclusion

The goal of our work was to study and describe methods of keypoint detection and description that could be used in the task of finding an image in a database of authorized images and to test them against each other regarding this task.

In our work we have chosen four methods to test and described them thoroughly - SIFT, AKAZE, ORB, and LATCH. We have also chosen and described two evalu-

ation methods - the mean average precision and the receiver operating characteristic curve.

We constructed our custom dataset for the purpose of the testing. We have chosen a suitable set of ground truth images. These ground truth images were then used to construct eight dataset groups using various image transformations we implemented and 12 dataset groups containing different resolutions of the ground truth images. The implementation of the transformation and the dataset creation was an extensive part of our work as we created the methods and the generation to be fully configurable.

The images in the dataset were described by the chosen methods and the descriptors were matched. The number of the matches was evaluated by the chosen evaluation methods and the obtained results were discussed in the last Section.

After evaluating the experimental results, we have concluded that the best method for our task is ORB followed closely by AKAZE. ORB was shown to be a faster method regarding the finding of the matches. For the mixed image group the matching run-time was 9176 seconds. It also has more stable results across the board. Out of the eight groups, the mAP value dropped below 0.8 only in the geometric group. The geometric group was also the only group where ORB was worse than both AKAZE and SIFT. AKAZE is faster in the keypoint description but slower in matching. The matching run-time was 11611 seconds which is 40 minutes slower than ORB. It also experiences more performance fluctuations than ORB, especially regarding the resolution change as was observed in Section 5.2.1. SIFT was proven to also be a solid method with good results, but it is very slow to compute and match. The matching took 16 hours and 30 minutes for one group only matching 50 images into 200. LATCH is the worst performing method in our task because we found it very susceptible to any sort of geometric transformation dropping to 0.57 mAP value for the geometric transformations group.

We could also discuss if ORB could provide even better results if we better optimized the number of keypoints it is detecting, likely by increasing the number. This could improve its results and show it to be the best method for our task but it would slow down the method regarding matching. It could also bring in unsolicited noise as discussed in Section 5.1.

6.1 Future work

This bachelor's thesis and the results obtained will serve as a foundation for our master's thesis. Several tasks have come up during our experiment that are worth further study.

We want to further improve the dataset that was created in regard to choosing better ground truth images. During our study of the methods of image description, it has occurred to us that the datasets used as the benchmark for most of the methods are compact. This compactness is good to quickly test the methods of image description but does not provide any in-depth information. We will focus on finding out, whether or not it gives any recent datasets. If we won't find any reasonable datasets, we will try to construct our benchmark dataset.

We also want to focus on the aspect of computer vision that has been on the rise recently and that is the neural network based methods of image description like SuperPoint [24] and SuperGlue [25]. Neural networks are rapidly gaining popularity and have become state-of-the-art approaches in many fields of artificial intelligence and we would like to test them against the non-neural network methods and compare their performance.

Recently, multimodal neural networks rose in popularity and are extensively researched regarding the task of classification. It would also be interesting to test the performance of these networks regarding image matching.

References

[1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[2] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell*, 34(7):1281–1298, 2011.

[3] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[4] Gil Levi and Tal Hassner. Latch: learned arrangements of three patch codes. In *2016 IEEE winter conference on applications of computer vision (WACV)*, pages 1–9. IEEE, 2016.

[5] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European conference on computer vision*, pages 214–227. Springer, 2012.

[6] Joachim Weickert, BM Ter Haar Romeny, and Max A Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE transactions on image processing*, 7(3):398–410, 1998.

[7] Matthew Brown and David G Lowe. Invariant features from interest point groups. In *BMVC*, volume 4. Citeseer, 2002.

[8] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[9] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[11] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[12] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In *International Conference on Advances in Pattern Recognition*, pages 399–408. Springer, 2001.

[13] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[14] Lior Wolf, Tal Hassner, and Yaniv Taigman. Descriptor based methods in the wild. In *Workshop on faces in'real-life'images: Detection, alignment, and recognition*, 2008.

[15] Lior Wolf, Tal Hassner, and Yaniv Taigman. Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *IEEE transactions on pattern analysis and machine intelligence*, 33(10):1978–1990, 2010.

[16] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):43–57, 2010.

[17] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[18] Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6, 2004.

[19] Andrew Turpin and Falk Scholer. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 11–18, 2006.

[20] M. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39 4:561–77, 1993.

[21] John A Swets. *Signal detection theory and ROC analysis in psychology and diagnostics: Collected papers*. Psychology Press, 2014.

[22] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International conference on computing, mathematics and engineering technologies (iCoMET)*, pages 1–10. IEEE, 2018.

[23] Alexander Mordvintsev and K. Abid. Feature matching, 2013.

[24] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[25] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

Appendix A

This appendix shows the effect of the transformation groups on the images and an example of an image from each dataset group.

Figure 14: An example of the ground truth image.[4]

[4]Image in Figure 14 by alex.ch published in 2008 on Flickr.

(a) Noise dataset group example.

(b) Blur dataset group example.

Figure 15

(a) Histogram dataset group example.

(b) Color dataset group example.

Figure 16

(a) Local brightness dataset group example.

(b) Geometric dataset group example.

Figure 17

(a) Mixed dataset group example.

(b) Enhance dataset group example.

Figure 18

Appendix B

Following list is the list of global options for the configuration:

- settings: contains setting for random ordering of the transformations
  - group_random_order: shuffles the order of the transformation groups if set to True, geometric group is always the last
  - transformations_random_order: shuffles the transformations inside the group if set to True
- transformation_group: contains the transformation categories and the max_transformations_allowed parameter that sets the number of transformations used from the category
- transformation: contains the list of the transformations, each transformation has the following setting
  - group: the category of the transformation
  - enabled: if set to True, transformation can be applied
  - probability: the probability of the application
  - params (optional): if the transformations has any parameters, they are further specified here

Let us now look at the parameters params for the transformations:

- white_noise:
  - deviation: interval $[a, b]$ from which a standard deviation is chosen randomly
- salt_and_pepper:

  - amount: the number of pixels in the image being transformed given as a fraction of total image pixels
  - ratio: ratio of salt vs pepper (0 for pure pepper, 1 for pure salt)

- iso_noise:
  - color_shift: interval $[a, b]$ for color hue change, measured as a fraction of 360 degrees Hue angle in HLS colorspace
  - intensity: interval $[a, b]$ for choice of multiplicative factor that controls the strength of color and luminance noise

- image_compression:
  - quality_lower: lower quality boundary of the compression
  - quality_upper: upper quality boundary of the compression
  - comp_type: compression type (jpeg or webp)

- sharpness:
  - sharp_const: interval $[a, b]$ from which the sharpness shift is chosen

- posterize:
  - num_bits: number of bits to which each color channel is reduced

- averaging_blur:
  - max_mask_size: specifies interval $[3, max\_mask\_size]$ from which the blurring mask size is chosen

- gaussian_blur:
  - max_mask_size: specifies interval $[3, max\_mask\_size]$ from which the blurring mask size is chosen

- median_blur:
  - max_mask_size: specifies interval $[3, max\_mask\_size]$ from which the blurring mask size is chosen

- motion_blur:
  - max_mask_size: specifies interval $[3, max\_mask\_size]$ from which the blurring mask size is chosen

- glass_blur:
  - max_delta: max distance between pixels which are swapped
  - sigma: standard deviation for the Gaussian kernel

  - iterations: number of repeats

- clahe:
  - clip_limit: upper threshold value for contrast limiting

- equalization:
  - mode: mode of equalization, either cv or pil for OpenCV or Pillow method

- channel_drop:
  - fill_value: the value that replaces the dropped channel

- channel_shuffle

- color_jitter:
  - brightness: specifies interval $[1 - brightness, 1 + brightness]$ from which the brightness change is chosen
  - contrast: specifies interval $[1 - contrast, 1 + contrast]$ from which the contrast change is chosen
  - saturation: specifies interval $[1 - saturation, 1 + saturation]$ from which the saturation change is chosen
  - hue: specifies interval $[-hue, hue]$ from which the hue change is chosen

- rand_gamma:
  - gamma_limit: specifies interval $[a, b]$ from which a c is chosen and gamma calculated as $c/100.0$

- sepia

- black_point:
  - threshold: specifies interval $[a, b]$ from which a threshold for black point is chosen

- solarize:
  - threshold: specifies interval $[a, b]$ from which a threshold for inverting is chosen

- vignette:
  - factor: specifies factor from which a gaussian kernel sigma value is calculated as $factor \cdot image\_shape$

- shadowing:

  - start: specifies from which point the image can be shadowed (0 for whole image, 1 for no shadow)

- rand_sunflare:
  - flare_roi: region of interest where sunflare can appear
  - num_circles_low: lower boundary for the number of circles around the center
  - num_circles_up: upper boundary for the number of circles around the center
  - radius: radius of the center

- light_point:
  - lowest_scale: lower boundary for the size of the light point
  - highest_scale: upper boundary for the size of the light point

- flip:
  - d: direction of the flip (0 vertical, 1 horizontal, -1 both)

- grid_dist:
  - num_steps: count of the grid cells
  - distort_limit: interval $[-distort\_limit, distort\_limit]$ for distortion of each cell

- rotate:
  - angle: interval $[a, b]$ from which the rotation angle is chosen
  - lossless: True for the lossless rotation

- scale:
  - ratio: interval $[a, b]$ from which the ratio of the resolution change is chosen

- crop:
  - crop_val: interval $[a, b]$ from which the ratio of the crop change is chosen

- perspective

The YAML file for the transformation configurations looks as follows:

```
settings:
  group_random_order: True
  transformations_random_order: True

transformation_group:
```

```
    noise:
      max_transformations_allowed: 3
    blur:
      max_transformations_allowed: 2
    histogram:
      max_transformations_allowed: 1
    color:
      max_transformations_allowed: 3
    local_brightness:
      max_transformations_allowed: 2
    geometric:
      max_transformations_allowed: 4

transformation:
    white_noise:
      group: noise
      enabled: True
      probability: 0.25
      params:
        deviation: [25, 45]
    salt_and_pepper:
      group: noise
      enabled: True
      probability: 0.1
      params:
        amount: 0.0075
        ratio: 0.5
    iso_noise:
      group: noise
      enabled: True
      probability: 0.2
      params:
        color_shift: [0.01, 0.05]
        intensity: [0.1, 0.25]
    image_compression:
      group: noise
      enabled: True
      probability: 0.2
      params:
        quality_lower: 40
        quality_upper: 70
        comp_type: 'jpeg'
    sharpness:
      group: noise
      enabled: True
```

```
    probability: 0.3
    params:
      sharp_const: [0.75, 1.25]
posterize:
    group: noise
    enabled: False
    probability: 0.75
    params:
      num_bits: 4

averaging_blur:
    group: blur
    enabled: True
    probability: 0.5
    params:
      max_mask_size: 9
gaussian_blur:
    group: blur
    enabled: True
    probability: 0.5
    params:
      max_mask_size: 9
median_blur:
    group: blur
    enabled: True
    probability: 0.5
    params:
      max_mask_size: 9
motion_blur:
    group: blur
    enabled: False
    probability: 0.75
    params:
      max_mask_size: 5
glass_blur:
    group: blur
    enabled: False
    probability: 0.75
    params:
      max_delta: 4
      sigma: 0.1
      iterations: 1

clahe:
    group: histogram
```

```
    enabled: True
    probability: 0.3
    mutex: equalization
    params:
      clip_limit: 6
equalization:
    group: histogram
    enabled: True
    probability: 0.3
    mutex: clahe
    params:
      mode: 'cv'

channel_drop:
    group: color
    enabled: True
    probability: 0.008
    params:
      fill_value: 0
channel_shuffle:
    group: color
    enabled: True
    probability: 0.008
color_jitter:
    group: color
    enabled: True
    probability: 0.4
    params:
      brightness: 0.25
      contrast: 0.3
      saturation: 0.18
      hue: 0.09
rand_gamma:
    group: color
    enabled: True
    probability: 0.4
    params:
      gamma_limit: [75, 125]
sepia:
    group: color
    enabled: True
    probability: 0.08
black_point:
    group: color
    enabled: True
```

```
    probability: 0.25
    params:
      threshold: [5, 20]
  solarize:
    group: color
    enabled: False
    probability: 0.75
    params:
      threshold: 0

  vignette:
    group: local_brightness
    enabled: True
    probability: 0.2
    params:
      factor: 0.5
  shadowing:
    group: local_brightness
    enabled: True
    probability: 0.2
    params:
      start: 0.4
  rand_sunflare:
    group: local_brightness
    enabled: True
    probability: 0.2
    mutex: light_point
    params:
      flare_roi: [0, 0, 1, 0.5]
      num_circles_low: 3
      num_circles_up: 6
      radius: 200
  light_point:
    group: local_brightness
    enabled: True
    probability: 0.2
    mutex: rand_sunflare
    params:
      lowest_scale: 0.2
      highest_scale: 0.4

  flip:
    group: geometric
    enabled: True
    probability: 0.35
```

```
    params:
      d: 1
grid_dist:
    group: geometric
    enabled: True
    probability: 0.15
    params:
      num_steps: 10
      distort_limit: 0.2
rotate:
    group: geometric
    enabled: True
    probability: 0.25
    params:
      angle: [1, 4]
      lossless: True
scale:
    group: geometric
    enabled: True
    probability: 0.25
    params:
      ratio: [0.8, 1.2]
crop:
    group: geometric
    enabled: True
    probability: 0.25
    params:
      crop_val: [0.9, 0.99]
perspective:
    group: geometric
    enabled: True
    probability: 0.15
```