3DCV Homework 1

Department: CSIE Student ID: d11922016 Name: Jia-Wei, Liao
YouTube: https://youtu.be/roQgFH-6i_A

October 2, 2023

Problem 1: Homography Estimation

Given three color images A, B, and C, please follow the instruction to compute the homographies that warps the anchor image A to target image B and C.

(a) Image A

(b) Image B

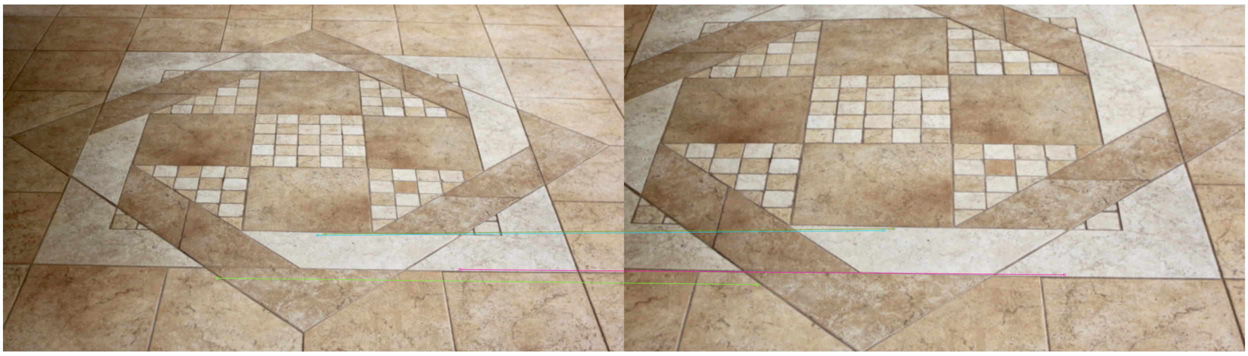
(c) Image C

Figure 1: Images A, B, C

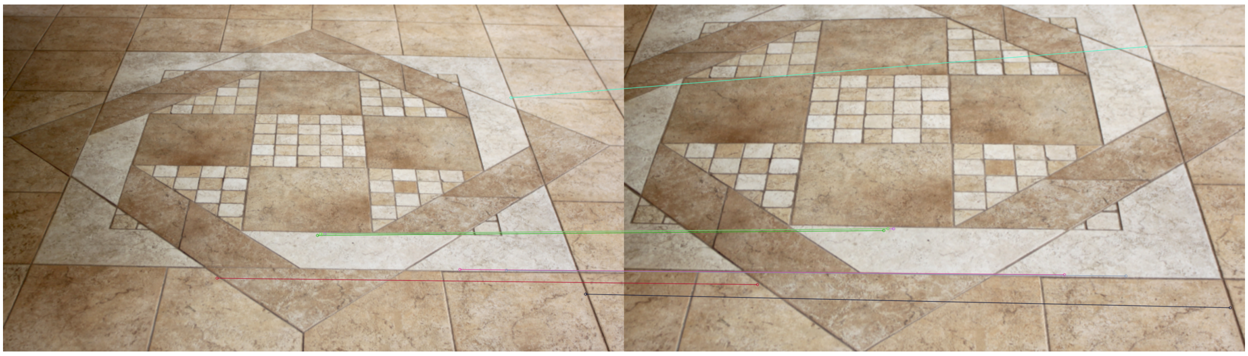
1.1 Feature Matching

- Perfom local feature detection on each image.
- Find the correspondence between anchor image and target images by descriptor matching.
- Reject the outliers by ratio test or manual comparison and select top k pairs from the matching result, where $k = 4, 8, 20$.
- (BONUS) Try other local features, e.g., SuperPoint.

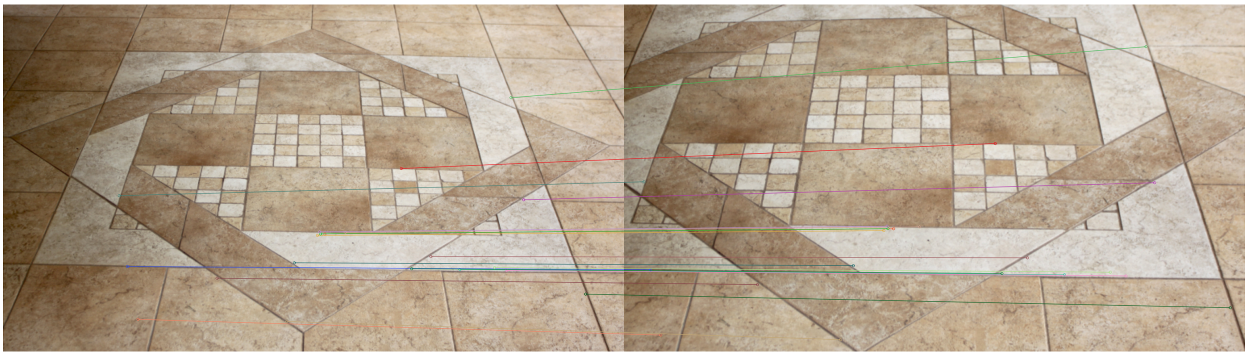
Solution.



(a) Top 4 pairs feature matching between image A and image B.



(b) Top 8 pairs feature matching between image A and image B.

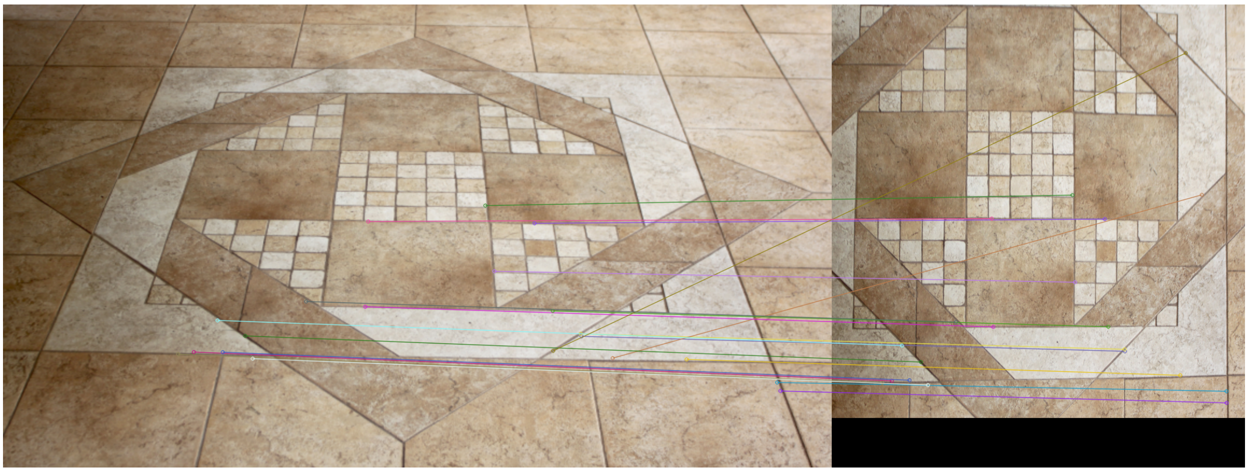


(c) Top 20 pairs feature matching between image A and image B.

Figure 2: Feature matching between image A and image B.

(a) Top 4 pairs feature matching between image A and image C.

(b) Top 8 pairs feature matching between image A and image C.



(c) Top 20 pairs feature matching between image A and image C.

Figure 3: Feature matching between image A and image C.

1.2 Direct Linear Transform

- For each k value, estimate homography between anchor image and target images with direct linear transform.
- Compute the reprojection error with the ground truth matching pairs.

Solution.

Let $\{(x_i, y_i)\}_{i=1}^n$, $\{(u_i, v_i)\}_{i=1}^n$ be the source points and target points. The primary objective is to determine the homography matrix that maps the source points to their corresponding target points. The homogeneous coordinates of the source points and target points, when transformed using the homography matrix, can be represented as follows:

$$\lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ h_{2,1} & h_{2,2} & h_{2,3} \\ h_{3,1} & h_{3,2} & h_{3,3} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, i = 1, 2, ..., n \tag{1}$$

This representation can express the transformation as a system of linear equations. Moreover, we can organize these equations into a $2n$ by 9 linear system as follows:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 & -u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 & -v_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nu_n & -y_nu_n & -u_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nv_n & -y_nv_n & -v_n \end{bmatrix} \begin{bmatrix} h_{1,1} \\ h_{1,2} \\ h_{1,3} \\ h_{2,1} \\ h_{2,2} \\ h_{2,3} \\ h_{3,1} \\ h_{3,2} \\ h_{3,3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2}$$

The linear system (2) can be reformulated as a constraint optimization problem:

$$h^* = \underset{\|h\|_2=1}{\operatorname{argmin}} \|Ah\|_2 = \underset{h\in\mathbb{R}^9}{\operatorname{argmin}} \frac{\|Ah\|_2}{\|h\|_2} = \underset{h\in\mathbb{R}^9}{\operatorname{argmin}} \frac{\|Ah\|_2}{\|h\|_2} = \underset{h\in\mathbb{R}^9}{\operatorname{argmin}} \frac{h^\top A^\top Ah}{h^\top h}$$

According to the principle of the Rayleigh quotient, h^* corresponds to the eigenvector associated with the smallest eigenvalue of the matrix $A^\top A$. Furthermore, we can obtain h^* by extracting the vector from the last column of the matrix V in the Singular Value Decomposition (SVD) of A.

Image pair	$k = 4$	$k = 8$	$k = 20$
(A, B)	138.8286	1.5032	0.2896
(A, C)	344.9664	530.8090	552.7130

Table 1: Reprojection error of DLT.

1.3 Normalized Direct Linear Transform

- Similar to 1.2, but use normalized direct linear transform instead.
- Compute the reprojection error and compare the results with 1.2.

$$\text{error}(P_1, P_2) = \frac{1}{n}\sum_{i=1}^{n} \|P_1^{(i)} - P_2^{(i)}\|_2$$

- (BONUS) Try other methods or tricks that may improve DLT or Normalized DLT.

Solution.

The normalization transformation, denoted as T, is defined by the following matrix:

$$T = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{bmatrix}.$$

Where the parameters are computed as:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \quad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \bar{d} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{n}$$
$$s = \frac{\sqrt{2}}{\bar{d}}, \quad t_x = -s\bar{x}, \quad t_y = -s\bar{y},$$

Now, let H represent the homography matrix of $T{P_1^{(i)}}_{i=1}^{n}$. The homography obtained through normalized Direct Linear Transformation (DLT) can be expressed as:

$$\widetilde{H} = T^{-1}HT$$

Table 2 demonstrates that the Normalized DLT method consistently exhibits superior performance compared to the conventional DLT method for the image pair (A, B). However, for the image pair (B, C), Normalized DLT appears to perform less favorably, which could potentially be attributed to the influence of outliers.

Image pair	Methods	$k = 4$	$k = 8$	$k = 20$
(A, B)	DLT	138.8286	1.5032	0.2896
(A, B)	Normalized DLT	**138.8172**	**1.4362**	**0.2799**
(A, C)	DLT	**344.9664**	**530.8090**	**552.7130**
(A, C)	Normalized DLT	344.9710	538.8167	565.7884

Table 2: Reprojection error of DLT and Normalized DLT.

Ablation Study (BONUS).

To address the influence of outliers, we introduce Algorithm 1, designed to identify and eliminate outliers within corresponding source and target point sets, denoted as P_1 and P_2. This algorithm computes the vector P_{12} and subsequently derives the slopes m_{12} associated with these vectors. It utilizes statistical characteristics to detect outliers by quantifying their deviation from the median slope, normalized by the standard deviation, with the aid of a user-defined threshold parameter τ. This algorithm can exclude outlier points from consideration, ultimately enhancing the precision and stability of geometric transformations. We shows the distributions of m_{12} before and after the removal of outliers in Figure 4.

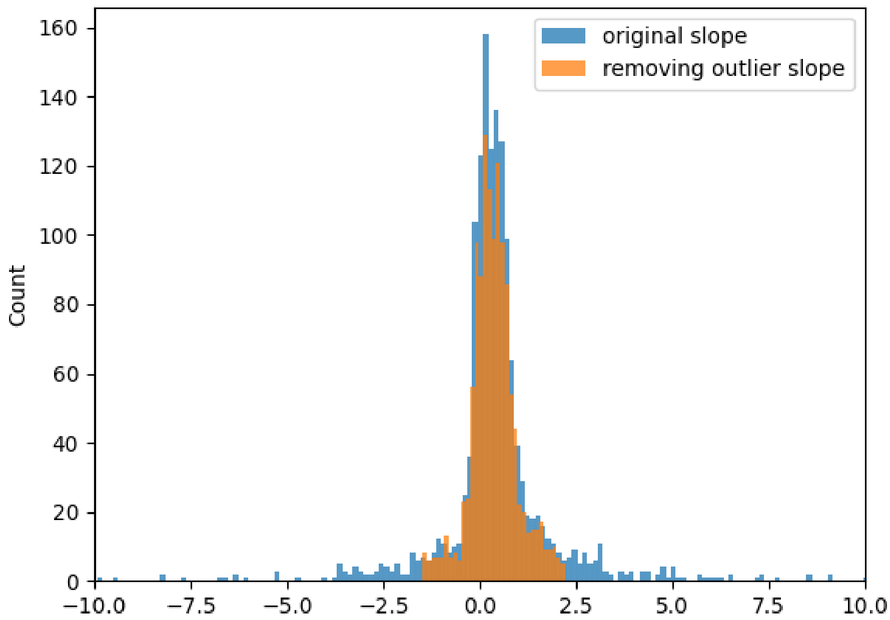


Figure 4: Distribution of m_{12}.

Algorithm 1 Outlier Removal

1: **Input:** source points P_1, target points P_2, and threshold $\tau > 0$.
2: Compute the vector P_1 to P_2 $P_{12} = P_2 - P_1$.
3: Compute the P_{12} slope $m_{12} = \Delta_y P_{12} / \Delta_x P_{12}$.
4: Find the inlier index $I = \text{Index}\left(\left|\frac{m_{12} - \text{median}(m_{12})}{\text{std}(m_{12})}\right| < \tau\right)$
5: **return** selected source points and target points $P_1(I)$, $P_2(I)$

Table 3 shows that when the original reprojection error is relatively high, employing the outlier removal algorithm with the DLT and Normalized DLT methods at $k = 8$ and $k = 20$ results in lower reprojection errors. However, with fewer points at $k = 4$, the outcomes become less stable. Hence, in situations with a sufficient number of points, this algorithm effectively detects and removes outliers within image point pairs, improving the precision and stability of geometric transformations or estimations.

Additionally, we attempted to employ the RANSAC Algorithm 2 to identify the optimal homography matrix. The RANSAC algorithm utilizes random sampling to effectively eliminate outliers, thereby enhancing accuracy.

Image pair	Methods	Outlier removal	$k = 4$	$k = 8$	$k = 20$
(A, B)	DLT		138.8286	1.5032	**0.2896**
(A, B)	DLT	✓	**138.8186**	**0.4691**	0.4167
(A, B)	Normalized DLT		**138.8172**	1.4362	**0.2799**
(A, B)	Normalized DLT	✓	**138.8172**	**0.5217**	0.4254
(A, C)	DLT		**344.9664**	530.8090	552.7130
(A, C)	DLT	✓	826.0086	**151.5762**	**8.9089**
(A, C)	Normalized DLT		**344.9710**	538.8167	565.7884
(A, C)	Normalized DLT	✓	826.0152	**19.3519**	**5.8534**

Table 3: Reprojection error of DLT and Normalized DLT with outlier removal.

Algorithm 2 Estimate homography matrix with RANSAC

Input: source points P_1, target points P_2, and maximum iteration M.
Initial: best error $\epsilon^* = \infty$.
for $i = 1, 2, ..., M$ **do**
    Random sample source points and target points $\widetilde{P}_1$, $\widetilde{P}_2$.
    Compute homography matrix H with $\widetilde{P}_1$, $\widetilde{P}_2$.
    Compute inlier reproject error ϵ between P_1 and P_2.
    if $\epsilon < \epsilon^*$ **then**
        $\epsilon^* \leftarrow \epsilon$.
        $H^* \leftarrow H$.
    end if
end for
return best homography matrix H^*.

Table 4 demonstrates that, in the case of image pair (A, B), both the DLT and Normalized DLT methods significant improvements with the use of RANSAC when the number of inliers is set to $k = 4$ and $k = 8$, leading to a substantial reduction in reprojection error. However, at $k = 20$, where the reprojection error is already quite low, a slight increase is observed. For image pair (A, C), both the DLT and Normalized DLT methods consistently exhibit a notable decrease in reprojection error across all inlier counts ($k = 4$, $k = 8$, and $k = 20$) when RANSAC is applied. This experiment shows that the RANSAC effectively enhances the accuracy of geometric transformation estimation.

Image pair	Methods	RANSAC	$k = 4$	$k = 8$	$k = 20$
(A, B)	DLT		138.8286	1.5032	**0.2896**
(A, B)	DLT	✓	**0.8332**	**0.5454**	0.3599
(A, B)	Normalized DLT		138.8172	1.4362	**0.2799**
(A, B)	Normalized DLT	✓	**0.8346**	**0.3171**	0.3277
(A, C)	DLT		344.9664	530.8090	552.7130
(A, C)	DLT	✓	**0.5427**	**0.6828**	**1.6684**
(A, C)	Normalized DLT		344.9710	538.8167	565.7884
(A, C)	Normalized DLT	✓	**0.5426**	**0.7569**	**1.6443**

Table 4: Reprojection error of DLT and Normalized DLT with RANSAC.

Discussion.

The condition number of a matrix A is a vital metric in scientific computing, providing a fundamental gauge of matrix stability. It is calculated as the ratio of the maximum singular value $\sigma_{\max}(A)$ to the minimum singular value $\sigma_{\min}(A)$ of the matrix. A lower condition number indicates increased stability, while a higher value implies potential numerical instability. It's defined as

$$\mathrm{cond}(A) := \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}$$

In this study, we employ the condition number as a metric to assess the stability of the homography matrix. We investigate the effect of different methods on the condition number. The Normalized Direct Linear Transform (DLT) method, as shown in Table 5, exhibits a significant reduction in the condition number compared to the standard DLT method for various image pairs.

Image pair	Methods	$k = 4$	$k = 8$	$k = 20$
(A, B)	DLT	434668.4932	84677.4291	82068.5854
(A, B)	Normalized DLT	2.3454	1.5212	1.7042
(A, C)	DLT	5978840.2303	107130381.5828	28327399.8783
(A, C)	Normalized DLT	10.7620	118.82915	54.7303

Table 5: Condition number of homography matrix H of DLT and Normalized DLT.

From the ablation study, we observe that removing outliers can reduce the reprojection error in certain cases, and adopting the RANSAC algorithm consistently leads to a substantial decrease in reprojection error. However, when combining outlier removal and the RANSAC algorithm, as shown in Table 6, the results unfortunately do not surpass the performance achieved by directly employing the RANSAC algorithm.

Image pair	$k = 4$	$k = 8$	$k = 20$
(A, B)	1.6160	0.7586	0.3218
(A, C)	3.3402	2.8648	1.8754

Table 6: Reprojection error of Normalized DLT with outlier removal and RANSAC.

Problem 2: Document Rectification

Rectification is one of the most fundamental techniques when digitizing documents. Given an image of a document captured by the camera, please recover its original geometric property which is lost after perspective transformation.

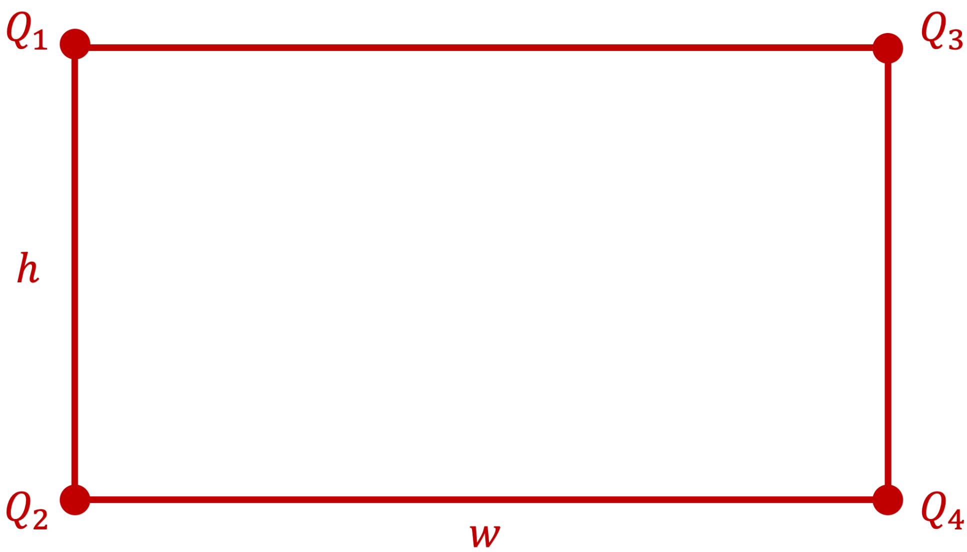
2.1 Capture Document and Mark 4 Corner Points

- Find an interesting document that your want to rectify. Note that the image must be captured by yourself.
- Automatically or manually mark the corner points on the image.

Solution.

We detect the four corners, denoted as P_1, P_2, P_3, P_4, using the `mouse_click.py` program. Subsequently, we calculate the height h and width w as $(h, w) = P_4 - P_1$. The coordinates of the four corners in the source image and their corresponding target corner coordinates are presented in Table 7.

(a) Original image with 4 corner.



(b) Target corners.

Figure 5: Comparison of original image and image rectification.

2.2 Homography Estimation and Warp Image

- Compute the homography for image transformation.
- Implement bilinear interpolation for image warping.

	left-up	left-down	right-up	right-down
source	$P_1(17, 3)$	$P_2(579, 2)$	$P_3(92, 813)$	$P_4(483, 821)$
target	$Q_1(0, 0)$	$Q_2(h, 0)$	$Q_3(0, w)$	$Q_4(h, w)$

Table 7: Corner coordinates.

Solution.

Let $f(x, y)$ be the grayscale of source image. The biliner interpolation can be expressed as follows for $x \in (x_1, x_2)$ and $y \in (y_1, y_2)$,

$$f(x, y) = w_{1,1} f(x_1, y_1) + w_{1,2} f(x_1, y_2) + w_{2,1} f(x_2, y_1) + w_{2,2} f(x_2, y_2),$$

where the interpolation weights are defined as:

$$w_{1,1} = \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)}, \quad w_{1,2} = \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)},$$
$$w_{2,1} = \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)}, \quad w_{2,2} = \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)}.$$

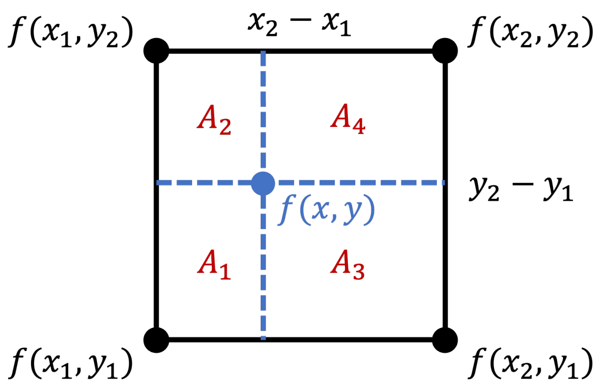


Figure 6: An illustration of bilinear interpolation.

We efficiently implemented bilinear interpolation using vectorized calculations with the NumPy library. The entire process took 0.06917 seconds.

(a) original image.

(b) image rectification.

Figure 7: Comparison of original image and image rectification.

Interesting Finding.

The interpolation weights can be expressed as the area ratios of four cubes:

$$w_{1,1} = \frac{A_4}{A_1 + A_2 + A_3 + A_4}, \quad w_{1,2} = \frac{A_3}{A_1 + A_2 + A_3 + A_4},$$
$$w_{2,1} = \frac{A_2}{A_1 + A_2 + A_3 + A_4}, \quad w_{2,2} = \frac{A_1}{A_1 + A_2 + A_3 + A_4}.$$

Appendix

Environment

To set the environment, you can run this command:

```
pip install -r configs/requirements.txt
```

The packages and their respective versions are displayed as follows:

```
numpy==1.24.4
opencv-python==4.8.0.76
seaborn==0.12.2
```

Notice that we installed version 4.8.0.76 of opencv-python as we were unable to install version 4.5.1.48.

Reproduce

- To reproduce problem 1 result of image 1-0 and 1-1, you can run the command as follows:

```
bash scripts/run_1.1.sh
```

- To reproduce problem 1 result of image 1-0 and 1-2, you can run the command as follows:

```
bash scripts/run_1.2.sh
```

- To reproduce problem 2 result, you can run the command as follows:

```
bash scripts/run_2.sh
```

- To reproduce all the results, you can run the command as follows:

```
bash scripts/run_all.sh
```