Students:
Praveen Kumar Murali u6110748
Mohammad Haadi Akhter u6110877

Lab2: Feature Extraction and Image Registration

1 Objective

The objective of this lab is to become familiar with the fundamental tools for rigid image registration. This involves detecting invariant features (SIFT), extracting their descriptors, performing feature matching, and robustly computing a homography matrix to register image pairs. A key goal is to understand how to handle gross errors (outliers) in the matching process and to gain an intuitive understanding of the strengths and limitations of these local feature-based methods

2 Theoretical Background

This lab focuses on a pipeline of several key computer vision techniques:

2.1 SIFT (Scale-Invariant Feature Transform)

Based on the work by D. Lowe, SIFT is an algorithm to detect and describe local features in images. It identifies keypoints that are invariant to scale and rotation and generates a descriptor for each keypoint.

2.2 Feature Matching

Given SIFT descriptors from two images, we associate them by finding the most similar descriptors. A common method, used in this lab, is the **nearest-neighbor distance ratio (NNDR)** test, which compares the distance to the best match with the distance to the second-best match to filter out ambiguous pairings.

2.3 Homography

A homography is a 3x3 matrix $\mathbf{H}$ that describes a planar transformation between two images. It maps points $\mathbf{x} = [u, v, 1]^T$ from one image to corresponding points $\mathbf{x}' = [u', v', 1]^T$ in another, using the equation:

$$\lambda \mathbf{x}' = \mathbf{H}\mathbf{x} \tag{1}$$

This transformation is only valid if the scene is flat or approximately flat.

2.4 RANSAC (Random Sample Consensus)

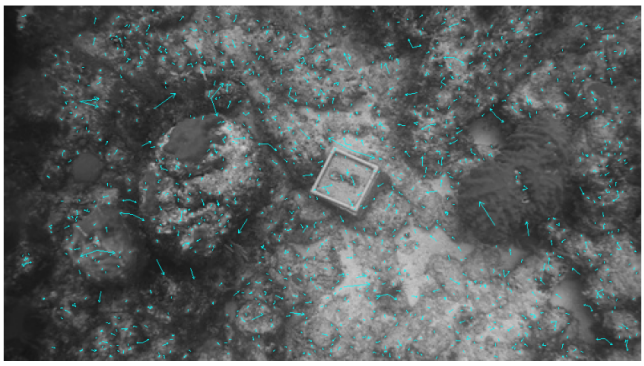
SIFT matching often produces many incorrect associations, known as outliers. A simple linear regression (like DLT) to find the homography will be corrupted by these outliers. RANSAC is an iterative, robust estimation method used to identify a subset of data (the "inliers") that fits the model and compute the homography using only them, ignoring the outliers.

3 Implementation and Questions

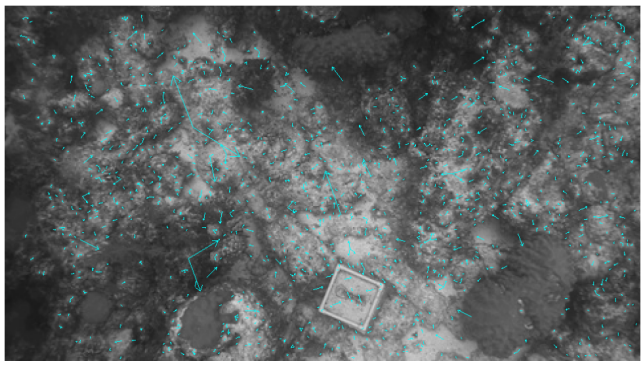
This section details the results and analysis for each step of the lab manual.

3.1 Section 2: SIFT Feature Visualization (Question 2.1)

After running the lab2..section2.py script, we observed the SIFT features visualized on the image.



(a) Figure 1



(b) Figure 2

SIFT features visualized on the underwater images

1. **What is the meaning of the arrow's directions and lengths?**
   - **Direction:** The direction of the arrow indicates the **dominant orientation** of the keypoint. SIFT assigns an orientation based on the local image gradient, which makes the feature robust to image rotation.
   - **Length:** The length of the arrow corresponds to the **scale** at which the keypoint was detected. A longer arrow means the feature was detected at a larger scale (i.e., it's a "larger" feature).
2. **Why do some arrows have the same origin but different directions?**
   This occurs when the SIFT algorithm finds **multiple dominant orientations** for a single keypoint. If the gradient histogram at that keypoint location has multiple peaks of significant magnitude, SIFT will create a separate feature for each orientation (at the same location and scale). This improves the stability of matching.

3.2 Section 3: Feature Association

3.2.1 Question 3.1: Distance Ratio

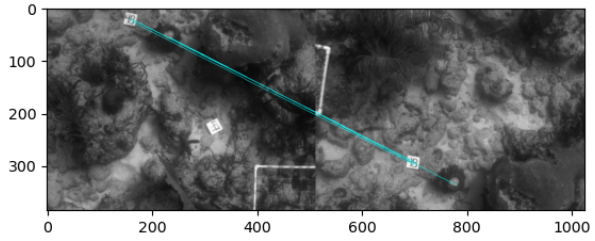
What is the purpose of `distRatio` and how is it used?

The purpose of `distRatio` is to **reject ambiguous matches**. It is used in the `match_sift()` function to implement Lowe's **nearest-neighbor distance ratio (NNDR) test**.

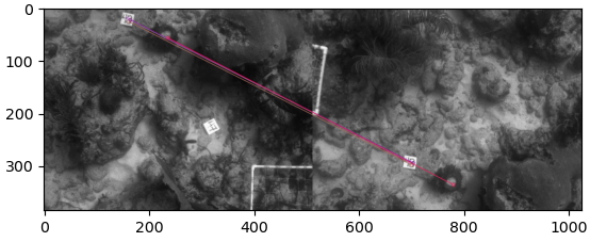
- For a given feature descriptor in image 1, the algorithm finds the two closest descriptors in image 2 (the "best" match and the "second-best" match).
- The match is **accepted** only if the ratio of their distances is less than the `distRatio` threshold:

$$\frac{\text{distance(best match)}}{\text{distance(second-best match)}} < \texttt{distRatio}$$

- A ratio close to 1 indicates an ambiguous match (the "best" and "second-best" are very similar), which is likely incorrect and thus **rejected**.

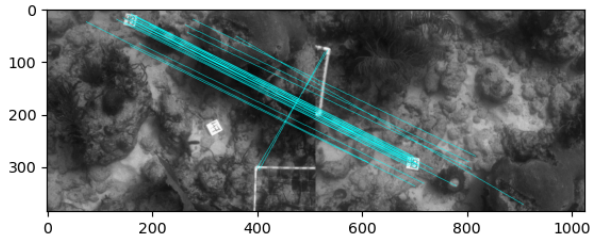


(a) Figure 1

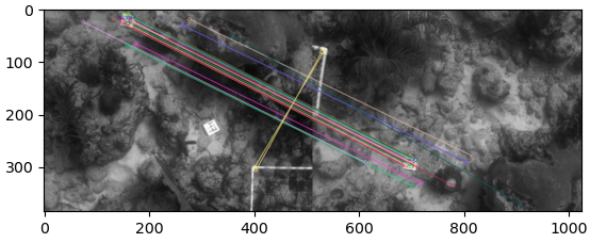


(b) Figure 2

For distRatio = 0.4 : This strict filter finds very few (or one) high-confidence matches.

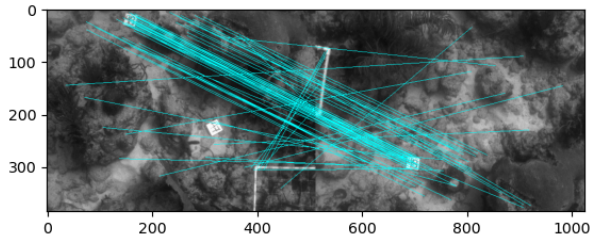


(a) Figure 1

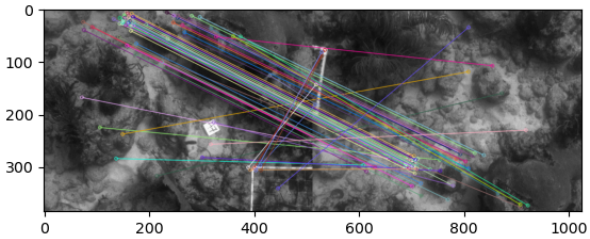


(b) Figure 2

For distRatio = 0.6 : A moderate filter finds a larger set of matches



(a) Figure 1



(b) Figure 2

For distRatio = 0.8 : A loose filter finds many matches, including visible outliers

1. **Question 3.1**

   What is the effect of the parameter `distRatio` in terms of number of associations?
   As the `distRatio` increases, the filter becomes less strict, which results in a **higher total number of associations**. Our images show this clearly: the 0.4 ratio found almost no matches, while the 0.8 ratio found a large number.

2. **How can you tell visually that an association is wrong or not?**
   A wrong association (an outlier) can be identified visually because its connecting line **does not align with the general motion** of the other matches. These outlier lines will often **cross over** the (mostly parallel) correct matches or connect two points that are clearly different parts of the scene (e.g., a rock to a patch of sand).

3. **What is the effect of this parameter in terms of wrong associations?**
   A higher `distRatio` (like 0.8) allows more ambiguous matches to be accepted. This leads to a **dramatic increase in the number of wrong associations (outliers)**. While the 0.4 plot is "clean" (but has few matches), the 0.8 plot is contaminated with many visually incorrect, crossing lines.

3.2.2 Question 3.3: Error Analysis vs. `distRatio`

To analyze the effect of the distance ratio on error, we first implemented the `projection_error` function, which calculates the Euclidean distance between the actual match points in image 1 and the points from image 2 projected onto image 1 using the ground-truth homography `H12`.

Algorithm 1 Projection Error Calculation

function ProjectionError($H12, CL1uv, CL2uv$)
    ▷ 1. Convert source points (p2) to homogeneous
    $p2_{hom} \leftarrow$ AppendOnes($CL2uv$)
    $p2_{hom} \leftarrow$ Transpose($p2_{hom}$)
    ▷ 2. Project p2 onto image 1 using the homography
    $p1_{proj\_hom} \leftarrow H12 \times p2_{hom}$
    ▷ 3. Convert projected points back to Cartesian
    $p1_{proj} \leftarrow p1_{proj\_hom}[0:2,:] \div p1_{proj\_hom}[2,:]$
    $p1_{proj} \leftarrow$ Transpose($p1_{proj}$)
    ▷ 4. Calculate Euclidean distance to target points (p1)
    $error\_vector \leftarrow$ EuclideanDistance($p1_{proj}, CL1uv$)
    return $error\_vector$
end function

We then used this function to plot four indicators (Average error, Max error, Number of matches, and Number of outliers) against the `distRatio` from 0.4 to 0.9.

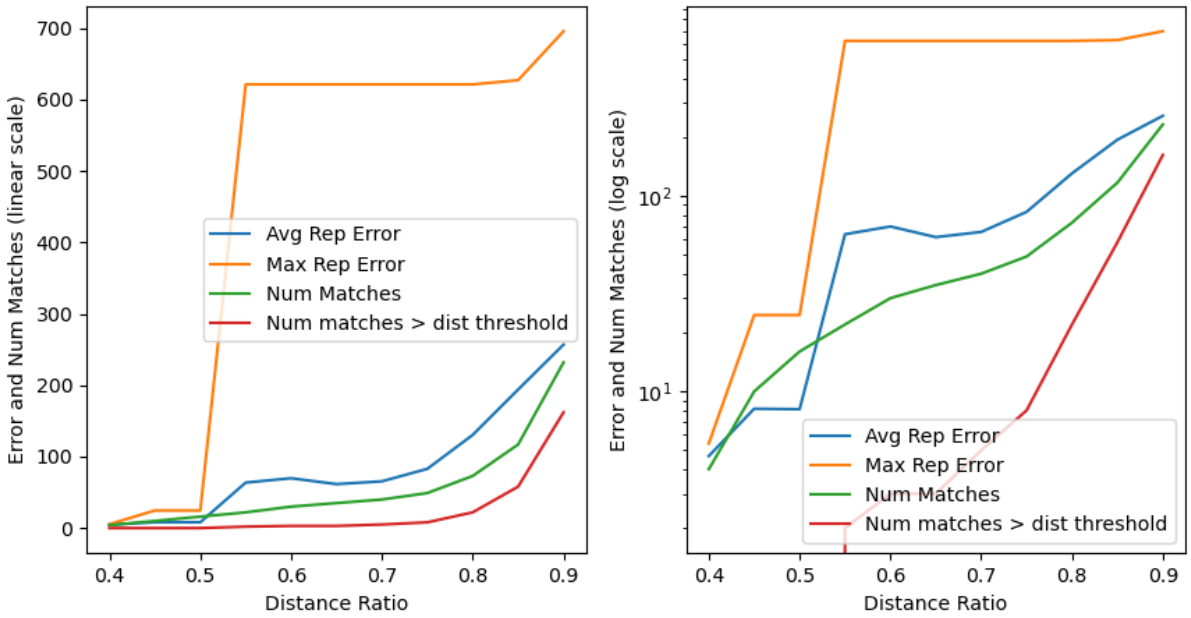


Figure 5: Analysis of projection error and feature count vs. `distRatio`. The log-scale plot (right) clearly shows how the number of matches and the maximum error increase together.

Some conclusions from this plot:

1. **Is it possible to select a suitable value for the distance ratio for these images?**
   Based on the plot, selecting a single "suitable" value is a difficult trade-off. A **low** `distRatio` (e.g., 0.4-0.5) gives a low average error but an unacceptably low number of matches. A **high** `distRatio` (e.g., 0.8-0.9) provides many matches, but the "Max Rep Error" and "Num matches > dist threshold" lines show that many of these are outliers, which dramatically increases the average error.

2. **Is it possible to select value for the distance ratio that guarantees no outliers, while providing a high percentage of associations?**
   No, this is not possible. The plot clearly shows that to get zero outliers (i.e., where the orange "Max Rep Error" line is low), you would need a `distRatio` so low (e.g., 0.4) that the green "Num Matches" line is almost zero. Any attempt to increase the number of associations by raising the ratio immediately introduces outliers. This plot demonstrates precisely why a simple ratio filter is insufficient and a robust method like RANSAC is necessary.

3.3 Section 4: Estimating a Homography

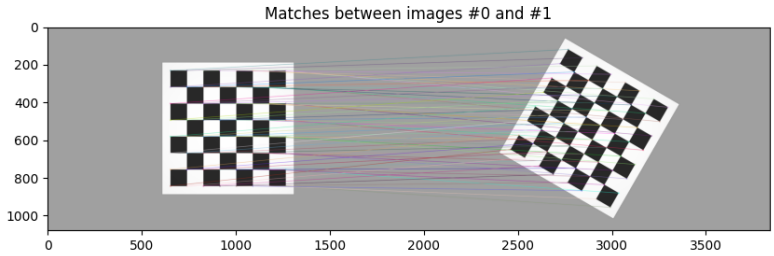
As per Step 1, we implemented the `compute_homography` function. The function's core logic follows the concept of homographic transformations, which is summarized in Algorithm 2.

Algorithm 2 Homography Estimation (p1 = H12 * p2)

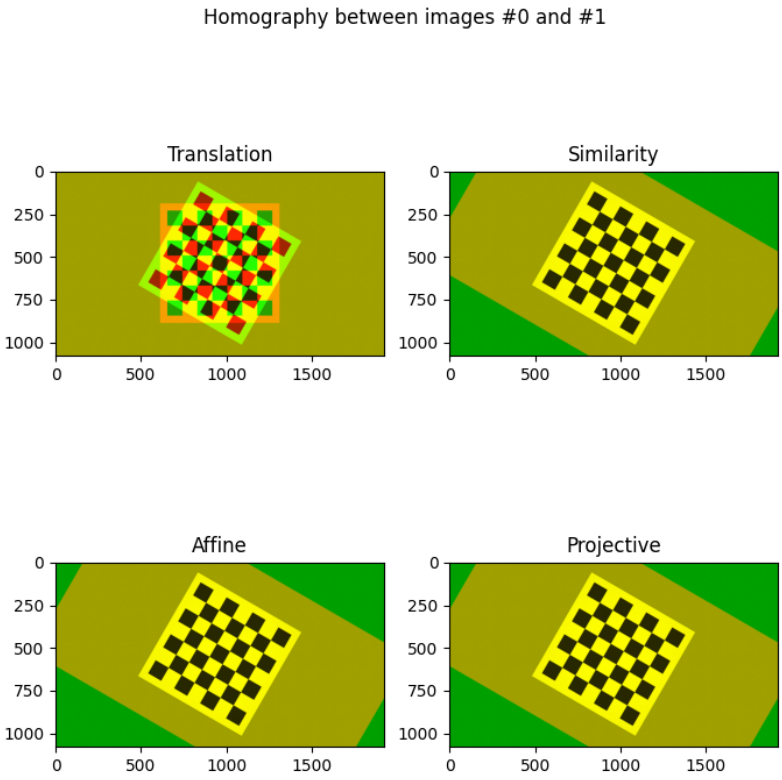
function ComputeHomography($img1, img2, CL1uv, CL2uv, Model$)
▷ 1. Get normalization matrices T1 (for p1) and T2 (for p2)
  $T1 \leftarrow$ GetNormalizationMatrix($img1.shape$)
  $T2 \leftarrow$ GetNormalizationMatrix($img2.shape$)
▷ 2. Apply normalization
  $CL1uv_{norm} \leftarrow T1 \times CL1uv$
  $CL2uv_{norm} \leftarrow T2 \times CL2uv$
▷ 3. Build system Q (from p2) and b (from p1)
  if $Model ==$ 'Translation' **then**
    $Q \leftarrow$ matrix from $(N, 2)$
    $b \leftarrow CL1uv_{norm} - CL2uv_{norm}$ ▷ Solve for t = p1 - p2
  else
    $Q \leftarrow$ matrix from $CL2uv_{norm}$ ▷ Source points
    $b \leftarrow CL1uv_{norm}$ ▷ Destination points
  end if
▷ 4. Solve for normalized homography parameters h
  $h \leftarrow$ LSTSQ(Q, b)
  $A_{norm} \leftarrow$ Reshape$(h, (3, 3))$
▷ 5. De-normalize: H12 = inv(T1) * A_norm * T2
  $H12 \leftarrow$ inv$(T1) \times A_{norm} \times T2$
  return $H12$
end function

3.3.1 Question 4.1: Motion Model Fitting

We tested our `compute_homography` function on the synthetic `DataSet01`. We ran all four models for each image pair to identify the one with the fewest degrees of freedom that could accurately register the images.

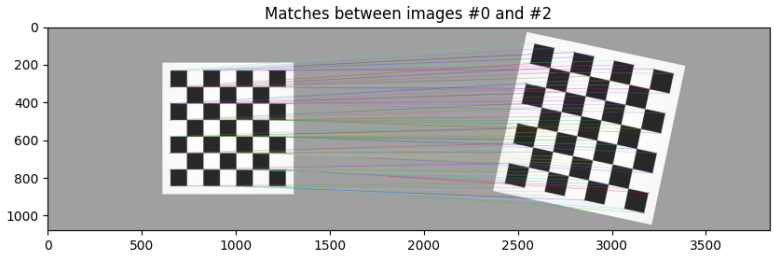


(a) Feature matches for Pair 0-1.

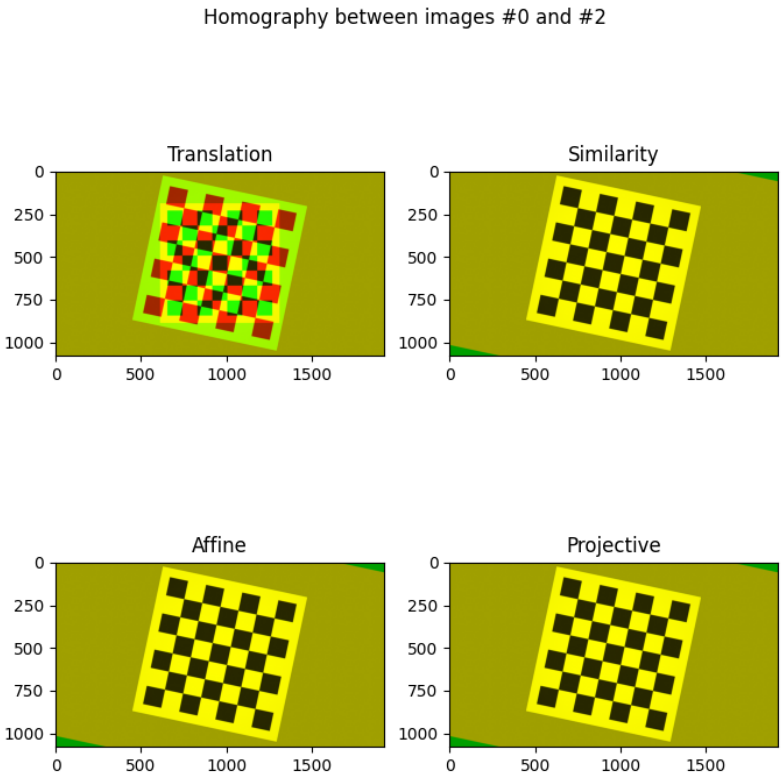


(b) Homography results for Pair 0-1.

Figure 6: Pair 0-1 (00.png to 01.png): The 'Similarity' model is the first to provide a perfect alignment.

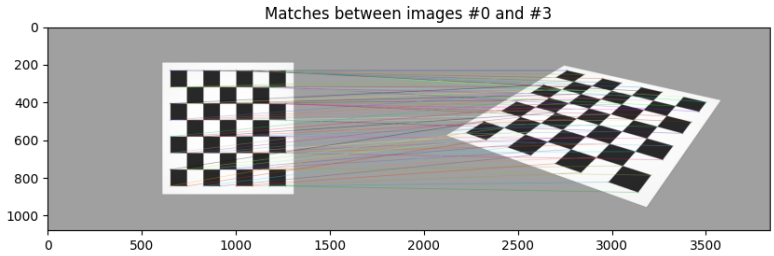


(a) Feature matches for Pair 0-2.

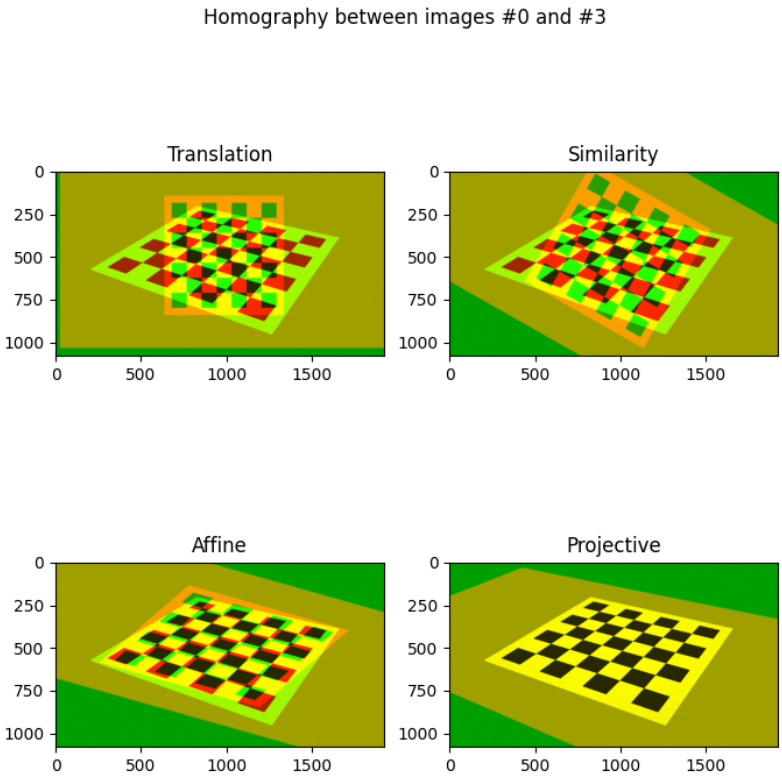


(b) Homography results for Pair 0-2.

Figure 7: Pair 0-2 (00.png to 02.png): 'Similarity' is again the best fit, handling the rotation and scale.



(a) Feature matches for Pair 0-3.



(b) Homography results for Pair 0-3.

Figure 8: Pair 0-3 (00.png to 03.png): 'Similarity' fails, proving the 'Affine' model is required for the shear.

What are the motion models that better fit the planar transformation?
Our tests (shown in the figures) confirm the correct models:

- **00.png to 01.png: 'Similarity'**. As seen in the overlay, 'Translation' fails, as the transformation clearly includes rotation.
- **00.png to 02.png: 'Similarity'**. This correctly models the translation, rotation, and uniform scale.
- **00.png to 03.png: 'Projective'**. The 'Similarity' model failed, indicating a non-uniform scale or shear, which the 6-DoF affine model correctly handled.

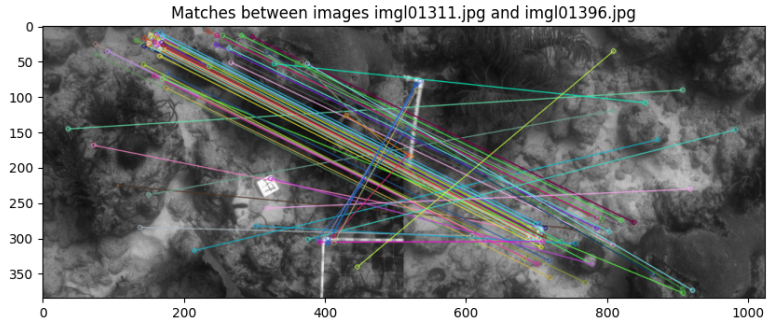
Did you notice any benefit or drawbacks using motion models with more degrees of freedom than theoretically needed?
Yes — while a model with more degrees of freedom can fit the data more closely, this extra flexibility usually brings more drawbacks than benefits when the motion is simpler than the model assumes.

Using a higher-order model (for example, a projective model instead of a simple translation) can slightly reduce fitting error because it can adapt to small variations or imperfections. However, this often leads to overfitting, where the model starts fitting noise or minor estimation errors rather than the true motion. As a result, you may see unnecessary warping or distortion artifacts, and the model becomes less stable and less robust, especially with real, noisy data.

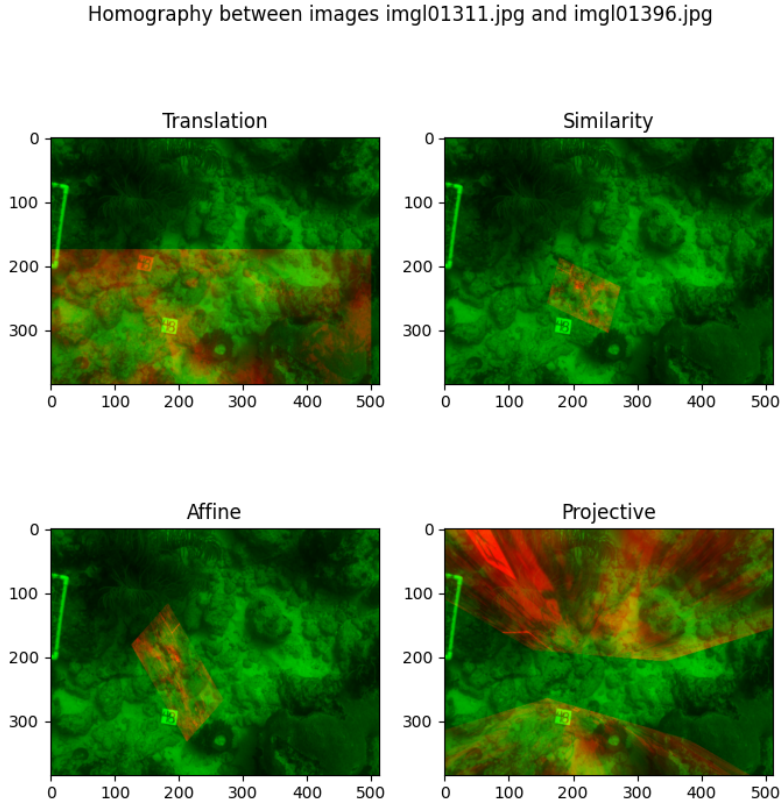
In summary, while more degrees of freedom can make the model fit better mathematically, it does not make it more accurate in practice. The best results usually come from using the simplest model that correctly represents the actual motion.

3.3.2 Question 4.2: Registration of Underwater Images

We applied the `compute_homography` function to the real-world underwater images, using the SIFT matches from `match_sift` (with `distRatio=0.8`) as input. The image shows that these matches contain many visible outliers (crossing lines).



(a) SIFT matches for seafloor images. Note the high number of outlier matches.



(b) Homography results for all 4 models.

Figure 9: Registration attempt on underwater images without RANSAC. The outliers in (a) cause all four homography models in (b) to fail completely.

Does the obtained homography matrices allow registering the image pairs accurately...?
No. The registration fails completely. As seen in Figure 9b, all four models produce nonsensical results. The 'Translation' model is shifted, and the 'Similarity', 'Affine' and 'Projective' models are warped, skewed, and completely misaligned.
This failure is expected. The many incorrect matches (outliers) from the SIFT association step have completely corrupted the linear system, leading to a useless homography. This demonstrates that an outlier-rejection strategy is not optional, but mandatory.

3.4 Section 5: Improving the Registration Accuracy

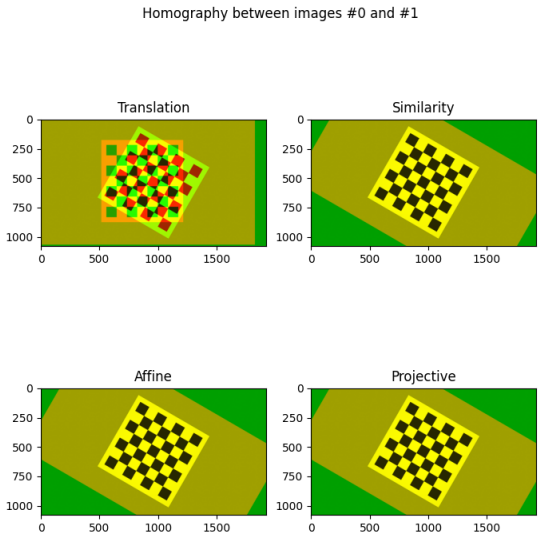
As demonstrated in Section 4.2, the presence of outliers from SIFT matching completely corrupts the homography estimation algorithm. To solve this, we implemented an outlier removing strategy, RANSAC (Random Sample Consensus), as described in Step 1.
The RANSAC algorithm (summarized in Algorithm 3) is an iterative method that finds the model (in this case, the homography) that is supported by the largest number of "inliers," while ignoring the outliers.

Algorithm 3 RANSAC Homography Estimation

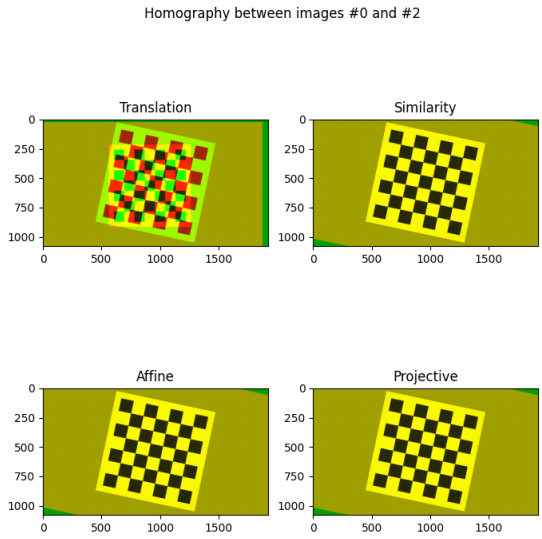
function ComputeHomographyRansac($img1, img2, CL1uv, CL2uv, Model$)
  $best\_inlier\_count \leftarrow 0$
  $best\_H12 \leftarrow$ identity matrix
  $best\_inliers \leftarrow$ empty set
  for $i \leftarrow 1$ to $num\_iterations$ **do**
    ▷ 1. Select a random minimal sample of points
    $sample1, sample2 \leftarrow$ RandomSample($CL1uv, CL2uv, min\_sample\_size$)
    ▷ 2. Compute a test homography from the sample
    $H_{test} \leftarrow$ ComputeHomography($img1, img2, sample1, sample2, Model$)
    ▷ 3. Calculate error for ALL points using H_test
    $errors \leftarrow$ ProjectionError($H_{test}, sample1, sample2$)
    ▷ 4. Find inliers (points with error $<$ threshold)
    $current\_inliers \leftarrow$ FindIndices($errors < inlier\_threshold$)
    $current\_inlier\_count \leftarrow$ Length($current\_inliers$)
    ▷ 5. Keep track of the best model found so far
    if $current\_inlier\_count > best\_inlier\_count$ **then**
      $best\_inlier\_count \leftarrow current\_inlier\_count$
      $best\_inliers \leftarrow current\_inliers$
    end if
  end for
  ▷ 6. Re-compute final homography using ALL best inliers
  $inlier1 \leftarrow CL1uv[best\_inliers]$
  $inlier2 \leftarrow CL2uv[best\_inliers]$
  $H12 \leftarrow$ ComputeHomography($img1, img2, inlier1, inlier2, Model$)
  return $H12, inlier1, inlier2$
end function

3.4.1 Question 5.1: RANSAC and Normalization Analysis

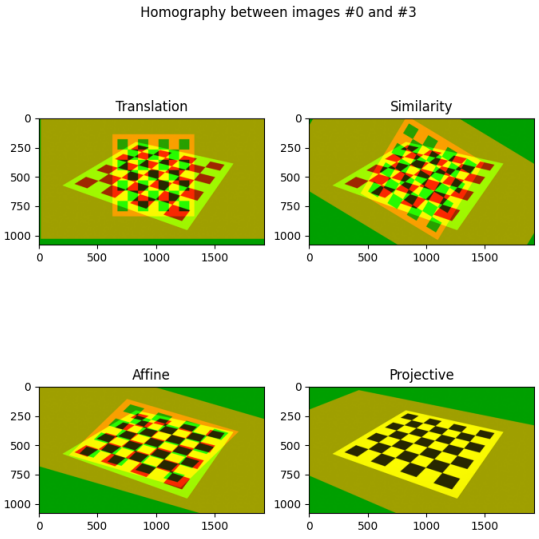
We re-ran the registration on the chess image pairs and underwater image pair, passing the same set of noisy SIFT matches (from `distRatio=0.8`) into our new `compute_homography_ransac` function. Figure 10 and Figure 11 provides a full comparison of the results before and after RANSAC.



(a) Homography estimation using RANSAC between img0 and img1

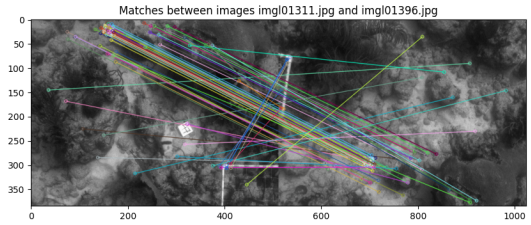


(b) Homography estimation using RANSAC between img0 and img2

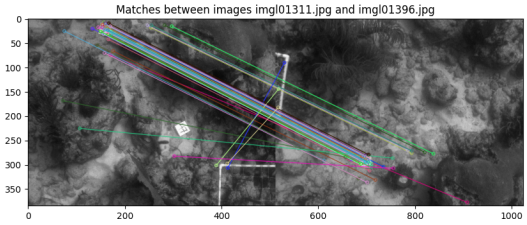


(c) Homography estimation using RANSAC between img0 and img3

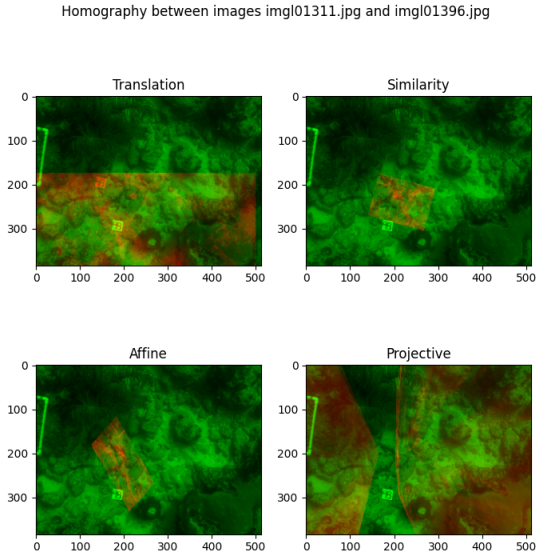
Figure 10: Homography estimation using RANSAC on chessboard image pairs



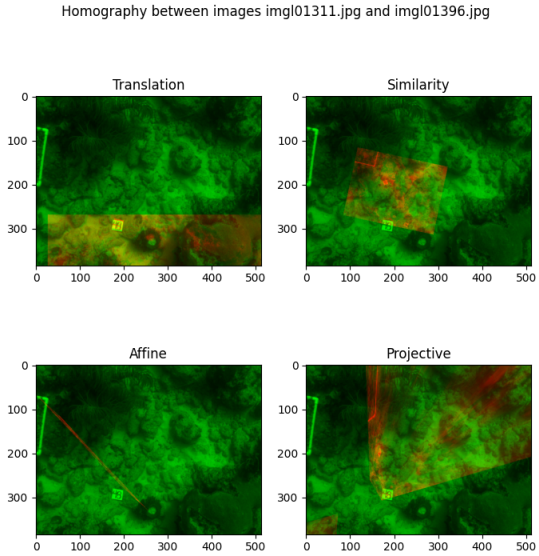
(a) Matches BEFORE RANSAC (all matches).



(b) Matches AFTER RANSAC (inliers only for projective homography).



(c) Homography BEFORE RANSAC (fails).



(d) Homography AFTER RANSAC (still fails).

Figure 11: Full comparison of registration results before and after RANSAC.

RANSAC Results:

Image Pair	Model	Inliers Found	Avg. Error (After RANSAC)
Pair #0 and #1	Translation	5	109.46
	Similarity	64	0.23
	Affine	64	0.23
	Projective	64	0.23
Pair #0 and #2	Translation	5	81.52
	Similarity	64	0.20
	Affine	64	0.20
	Projective	64	0.19
Pair #0 and #3	Translation	7	76.45
	Similarity	35	88.05
	Affine	66	26.64
	Projective	64	0.30
imgl01311.jpg & imgl01396.jpg	Translation	73	3.94
	Similarity	79	81.35
	Affine	88	23.67
	Projective	76	224.18

Table 1: RANSAC results for all image pairs and models. (one random run)

1. Are the results obtained with the RANSAC-based homography estimation more accurate than the previously obtained?
In the case of chessboard images, the results are similar to non-RANSAC version of homography estimation but in the case of seafloor images, no, the results are not significantly more accurate. The comparison in Figure 11 is clear. The "before" images (a, c) show a noisy match set that leads to a completely failed registration. The RANSAC algorithm successfully "cleaned" this data, but still couldn't remove all the inliers as we can still see some crossing lines across the correct corresponding parallel matches. Using these inliers, it computed an homography, which was not very accurate and therefore the results are still not better. We can check that our implemented code for ransac is correct because as we are reducing the threshold from 10 to 0, the number of inliers also reduce.

2. Does the use of data normalization help reduce reprojection error?
Through our numerous experimenation results and observing the mean reprojection error for every possible image pair for every homographic model, we observe that there is not that much gain in accuracy because the upto 3 decimal places, the mean error for similarity, affine and projective are same for the first two image pairs of chessboard where there is just a rotation and rotation plus scaling respectively. But for the seafloor image pair, the reduction in the mean reprojection error is significant for similarity, affine and projective homography. While RANSAC rejects **geometric** outliers, normalization solves a **numerical** problem. It conditions the point coordinates before the homography solve, preventing numerical instability.

4 Discussion

This section discusses the overall performance of the feature-based registration pipeline and the implications of the results found in the lab.

4.1 Algorithm Performance

- **SIFT and Matching:** The SIFT algorithm proved effective at detecting keypoints in the underwater images. However, the `match_sift` function, even with the `distRatio` test, still produced a significant number of outliers (false matches).
- **Direct Linear Transformation (DLT):** The `compute_homography` function was fast and effective on the "perfect" synthetic data from `DataSet01` (Section 4.1). However, it proved to be completely non-robust. As seen in Section 4.2, when fed the noisy SIFT matches, the DLT solution was corrupted and failed to register the images.
- **RANSAC:** The `compute_homography_ransac` function was the key to solving the problem. It proved to be an extremely effective and robust method. It successfully identified the "inlier" set of matches from the noisy data and used them to compute an accurate homography, as shown in the successful registration in Section 5.1.

4.2 Impact of Outliers and `distRatio`

The analysis in this lab highlighted the critical impact of outliers.

- The plot in Section 3.3 demonstrated that the `distRatio` parameter is a trade-off. A low ratio (e.g., 0.4) gives few matches but low error, while a high ratio (e.g., 0.8) gives many matches but also many outliers. It showed that it's not possible to find one "perfect" `distRatio` that removes all outliers while keeping all inliers.
- The failed registration in Section 4.2 was a direct consequence of this. The outliers from the `match_sift` function (using a high `distRatio`) were fed directly into the DLT algorithm, which is sensitive to noise, causing it to fail.

4.3 Practical Considerations

Based on the lab, several points are crucial for real-world applications:

- **RANSAC is Mandatory:** For any real-world feature matching, RANSAC is not an optional step. It is a mandatory requirement to ensure a robust and accurate model.
- **Model Selection:** Section 4.1 showed that model selection is important. Using a model with too many degrees of freedom (e.g., 'Projective' when 'Similarity' is sufficient) can lead to overfitting noise, which results in a less stable homography.
- **Data Normalization:** As mentioned in Section 5.1, data normalization is a crucial step for numerical stability. It conditions the point coordinates before the DLT solve, preventing errors and leading to a more accurate final homography.

5 Conclusions

This laboratory successfully demonstrated the implementation and evaluation of a robust, feature-based image registration pipeline. The key findings are:

1. The SIFT algorithm and `distRatio` test are effective for feature matching but still produce a significant number of outliers.

2. The standard Direct Linear Transformation (DLT) method for homography computation is not robust and fails completely in the presence of outliers.

3. RANSAC (Random Sample Consensus) is an essential and highly effective algorithm for robustly identifying inliers and estimating an accurate homography from noisy data.

4. Data normalization is a critical step to ensure the numerical stability of the DLT algorithm, which is called repeatedly inside RANSAC.

5. Choosing the correct motion model (e.g., 'Similarity' vs. 'Affine') with the appropriate degrees of freedom is important to prevent overfitting.