PIV Project

Group: 38, Diana Coelho, João Silveira, Tomás Esteves, and José Afonso
(Dated: February 22, 2025)

PART 1.2

I. A - PROBLEM DEFINITION

This problem involves computing a reliable coordinate transformation (homography) H_i^R for each frame F_i in a video sequence relative to a reference frame F_R. This task can be divided into several subproblems:

A, B) How should video frames and their interrelationships be represented as a graph, and what criteria should guide the creation of edges for effective transformations?

C) In what manner should edges be weighted to reflect their significance and reliability in accurately estimating coordinate transformation?

C, D) What approach should be utilized to identify the optimal homography within the graph and compute the composite homography between each frame and the reference frame?

II. B - SOLUTION

Our proposed solution intends to address the questions raised above systematically. We will examine each pipeline step in detail and justify our choices.

A. Graph Construction

We define a graph $G = (V, E)$ where:

- Vertices V: Each node $v_i \in V$ represents a frame F_i in the video
- Edges E: Each edge $e_{ij} \in E$ connects two frames F_i and F_j

Using this type of graph allows us to model and manage the complex relationships between video frames, facilitate efficient pathfinding for consistent transformations, and effectively convert the problem into an optimization problem.

B. Edge Types

For creating the edges of the graph, which consists of modeling the type of connections that can be made between the nodes (frames), we will use the following guiding principles:

1. Temporal Connections

We connect consecutive frames:

$$E_{temp} = \{e_{i,i+1} | \forall i \in [0, N-1]\} \tag{1}$$

We use this edge to maintain sequential consistency in video frames, as many consecutive frames show minor variations, allowing homography estimation to benefit from smooth transitions.

2. Direct Connections

We include this type of connection to enhance connectivity, bridging the gap caused by rapid movements and improving the graph's robustness.

$$E_{direct} = \{e_{i,0} | \forall i \in [1, N]\} \tag{2}$$

where we assume that the reference frame has $i = 0$

3. Similarity-Based Connections

We also used these connections to ensure we capture content-based connections using the K-nearest neighbors (KNN) algorithm. Our similarity measure was based on the centroid similarity of the key points weighted by the variance of the normalized descriptors of each frame. This way, we connect similar visual frames, identifying recurring patterns and reducing cumulative errors. Furthermore, it enhances the robustness of scene changes, as similar frames throughout the sequence offer alternative pathways for accurate homography computation.

$$E_{KNN} = \{e_{i,j} | F_j \text{ is among nearest neighbors of } F_i\} \tag{3}$$

The number of neighbors in the K-nearest neighbors algorithm, N_{KNN}, is a hyperparameter that we need to fine-tune.

C. Edge Weighting

An essential part of defining the graph is calculating the weights for each connection. We assume that nodes have no weights, so the weight for each connection should reflect the residual error from applying the homography between the nodes.

For our algorithm, we assume that the cumulative errors of homographies are additive and independent, which holds under the condition that there exists a reliable path within the frame graph where small, Gaussian-distributed and independent errors, consistent motion patterns, and dependable feature correspondences characterize homography estimations.

1. Feature Matching

To estimate the homography associated with each transition, we start by doing Feature Matching between F_i and F_j $\forall e_{ij} \in E$, applying the **Lowe's ratio** with hyperparameter τ to the descriptor's vectors, which worked as a pre-filtering and significantly reduced the number of outliers entering MSAC.

MSAC (M-estimatoe SAmple Consensus) followed this procedure, which was chosen to improve the model selection beyond just the inlier count. It considers

the following score function:

$$\text{Score}_{MSAC}(H_{ij}) = \sum_{n=1}^{N} \min(r_n^2, \delta^2) \tag{4}$$

Where r_n is a residual error of the n matched keypoint between frames i and j, δ is the MSAC threshold that distinguishes between inliers and outliers.

In this way, MSAC tries to minimize the score function, balancing inlier fit quality and outlier mitigation and rewarding better fitting inliers.

This algorithm also contains a set of hyperparameters we must fine-tune: the confidence value, p_{MSAC}, score function threshold, δ_{MSAC}, and the maximum number of iterations, N_{MSAC}.

2. Homography Estimation

In this way, we can effectively obtain the inliers set, $\mathcal{I}_{ij}$, which we use to compute the homographies, the forward and inverse ones, between the two frames.

We used the SVD method to solve the linear system $\mathbf{A}.\mathbf{h} = 0$ for homography estimation. This method was widely discussed in our classes.

3. Residual Calculation

As a result, we compute the residuals, the reprojection errors, for $\mathcal{I}$:

$$\mathbf{r}_{ij} = ||p_j^k - H_{ij} p_i^k||_2^2 \tag{5}$$

where p_i^k and p_j^k are corresponding keypoints in F_i and F_j, respectively, that belong to $\mathcal{I}_{ij}$

Edge Weights:

Given the set of residuals, denoted as $\mathbf{r}_{ij}$, we initially considered a linear combination of its mean, its variance, and the inlier ratio as the edge weight. However, through our experiments, we found that the most effective approach was to use only the mean, as given by the following expression:

$$\omega_{ij} = \frac{1}{|\mathbf{r}_{ij}|} \sum_k r_{ij}^k \tag{6}$$

This choice has the following good properties:

- Non-negativity: $\omega_{ij} \geq 0$ since residuals are norms
- Normalization: The mean residual is independent of $|\mathcal{I}_{ij}|$, allowing fair comparison across edges.

Furthermore, we impose that each connection has a sufficient number of reliable correspondences to ensure statistical significance and robustness by using the following equation:

$$\omega_{ij} = \begin{cases} \omega_{ij} \text{ if } |\mathcal{I}_{ij}| \geq N_{min} \\ +\infty \text{ if } |\mathcal{I}_{ij}| < N_{min} \end{cases} \tag{7}$$

where N_{min} is a hyperparameter that must be fine-tuned.

D. Graph Search for Homography Estimation

Now, the goal is to find a sequence of homographies that best align each frame for the reference frame. It corresponds to finding the *shortest* path in the graph where the cumulative edge weights (residuals) are minimized, converting the problem into an optimization one.

A path corresponds to $\mathcal{P}_i = \{v_{i_1}, v_{i_1}, \ldots, v_R\}$ where $v_{i_1} = F_i$ and $v_R = F_R$, and the shortest one,$\mathcal{P}_i^*$, is given by:

$$\mathcal{P}_i^* = \arg\min_{\mathcal{P}_i} \sum_{k=1}^{|\mathcal{P}|-1} \omega_{ik} \tag{8}$$

To tackle this problem, we used the *Dijkstra's* Algorithm, as it ensures we find the shortest path. Thus, for each $F_i \in V$:

$$\mathcal{P}_i^* = \text{Dijkstra}(G, F_i, F_R) \tag{9}$$

E. Homography Computation

Once the shortest path $\mathcal{P}_i^*$ is found for a frame F_i, compute the cumulative homography H_i^R by chaining the homographies along the path:

$$H_i^R = \prod_{e \in \mathcal{P}_i^*} H_e = H_{i_1 i_2}.H_{i_2 i_3}.\ldots.H_{i_{n-1} R} \tag{10}$$

III. C - EXPERIMENTS AND ANALYSIS

After implementing and defining the entire pipeline for Part 1.2, we conducted our analysis and experiments using mainly the provided volley dataset.

Initially, we fine-tuned the model's hyperparameters by visually inspecting the results of the estimated homographies for all video frames in the dataset. Although this process involves subjective elements, it enables us to understand how well the frames were aligned with the reference frame. Additionally, we aimed to minimize the path length $\mathcal{P}_i$ and its total cost for each frame F_i.

The table below shows the optimal values we identified for the hyperparameters:

Hyperparameter	Value
N_{KNN}	3
τ_{Lowes}	0.25
p_{MSAC}	0.99
N_{MSAC}	1000
δ_{MSAC}	4
N_{min}	18

TABLE I: Hyperparameters of the Model

The graph 1 illustrates the interrelationships among video frames, with nodes representing frames and edges denoting direct, temporal, and KNN-based content similarity. Dense clusters signify visually analogous frames,

whereas sparse regions indicate transitions or low correlation. The high degree of connectivity facilitates numerous reliable pathways for homography computation, thereby minimizing cumulative errors.

We conclude that both plots are highly correlated by analyzing both Figures 2 and 3. The different types of connections between nodes influence the variations in path cost and length. KNN-based content connections mainly drive quick changes, while a path length of one indicates direct connections. In contrast, smoother behavior demonstrates the stability of temporal connections. This distinction emphasizes the complementary roles of these connection types in achieving reliable homography estimation.

We also applied the model with the same hyperparameters to the dataset ISRWall and Pplant and obtained reasonable results from the visual inspection.

A. *Failure modes*

The choice of the hyperparameters is well-suited for the volley dataset. However, applying the program to a different dataset, we observe multiple possible "Failure modes."

Case 1: τ_{Lowes} is too small

A very small τ threshold makes the *Lowe's* ratio test overly strict, filtering out many potential matches. This results in an inlier set size ,$|\mathcal{I}_{ij}|$, that falls below the minimum N_{min} required to compute a valid homography.

Since the homography estimation requires at least four non-collinear points (or, more robustly, a larger N_{min} to account for noise and outliers), with insufficient inliers, the algorithm cannot calculate the transformation matrix, causing the pipeline to fail.

Case 2: N_min is too large

In this case, there are edges for which there is no homography estimation. Eventually, it could not be possible to estimate the composite homography, as the graph could fail to be fully connected.

Case 3: threshold MSAC is too small

This could imply that the best inlier set is smaller than the N_{min}, and it is not possible again to estimate the homography.

To mitigate the risk of program crashes when a path cannot be computed from a specific frame F_i to the reference frame, we implemented a method that first calculates the homography from frame F_i to frame F_j, where $j < i$. Subsequently, we leverage the established homography from frame j to the reference frame. This strategy enables us to estimate the homography from frame i to the reference frame, albeit with the understanding that this estimation may not be optimal.

PART 2

IV. A - PROBLEM DEFINTION

In this problem, we face a structure similar to that in . However, in this case, we must estimate a rigid transformation using 3D point clouds instead of a homography.

V. B - SOLUTION

We apply the same strategy here regarding all the solutions presented in the section about graph construction and edge weighting. However, we also include the following features:

A. Depth Map:

Only the key points and descriptors whose corresponding depth coordinates have confidence above the following threshold were considered.

B. Feature Matching: Geometric Consistency

We found that feature matching based solely on the descriptors led to inaccurate similarities due to local features and repetitive patterns. To resolve this issue, we employed the 'pydegensac.findFundamentalMatrix' function, ensuring that the previously matched features conform to the epipolar constraint. This approach guarantees that the correspondences respect the fundamental geometric relationship between the point clouds, improving the alignment accuracy of the subsequent transformation estimation.

1. Rigid Transformation Estimation

Kabsch algorithm:

In this case, to estimate the rigid transformation that relates Frame i to Frame j, we need to compute the rotation matrix $\mathbf{R}_{ij}$ and the translation vector $\vec{t}_{ij}$. We utilized the Kabsch algorithm, which employs Singular Value Decomposition (SVD) to determine the optimal rotation and translation that align the point clouds. This is achieved by computing the SVD of the covariance matrix of the centered coordinates, which allows us to calculate the rotation matrix and determine the translation vector.

ICP for Refinement

After obtaining initial estimates for rotation (R) and translation ($\vec{t}$) using the Kabsch algorithm, we refined these transformations with the Iterative Closest Point (ICP) algorithm. This algorithm minimizes the distance between the moving and reference point clouds by finding the closest point correspondences, minimizing alignment error, and updating the transformation until convergence, as explained in class. This refinement improves accuracy by correcting minor misalignments in the fused point cloud.

C. Composite Transformation

In this case, having found in the graph the shortest paths P_i^* for each frame F_i, we have the series of transformations $T_1, T_2, \ldots, T_n$ and the composite $T_{composite}$ is given by:

$$T_{composite} = T_n \circ T_{n-1} \circ \cdots \circ T_2 \circ T_1 \tag{11}$$

We assume that the reference frame is the initial frame by default and $\circ$ is the composition of transformations. Moreover, each transformation T_i can be represented as a 4×4 homogeneous transformation matrix:

$$T_i = \begin{pmatrix} \mathbf{R}_i & \vec{t}_i \\ \vec{0}^T & 1 \end{pmatrix} \tag{12}$$

where $\mathbf{R}_i$ is the rotation matrix and $\vec{t}_i$ is the translation vector for the i-th transformation.

VI. C - ANALYSIS AND EXPERIMENTS

The pipeline for Part 2 builds upon the logic developed in Part 1.2, with the addition of 3D point filtering and transformations. We conducted experiments primarily on the *office* dataset to evaluate the performance of the updated pipeline, focusing on the impact of parameter tuning and algorithm selection on the final results.

A. Parameter Tuning and Algorithm Testing

The pipeline in Part 2 involved several critical parameters and algorithmic choices, which we refined through experimentation. The primary focus was on matching accuracy, graph connectivity, and point cloud alignment. Below, we describe the parameters tested and their impact on performance.

Feature Matching Algorithms: We evaluated multiple matching approaches:

Hybrid Matching: Combined Lowe's Ratio Test and Mutual Nearest Neighbors for robust initial correspondences.

Cross-Check Matching: Enforced bidirectional consistency without additional filtering.

Optional Matching: Focused purely on nearest neighbor distances, sacrificing robustness for simplicity.

The hybrid matching approach consistently produced the most reliable matches across frames.

Geometric Consistency Filtering: We applied `pydegensac.findFundamentalMatrix` to enforce the epipolar constraint on 2D matches, refining correspondences and improving geometric consistency before proceeding to 3D inlier detection.

3D Inlier Detection (τ_{inliers}): Using MSAC, we filtered matches based on 3D consistency. The algorithm iteratively sampled random matches to compute an affine transformation via SVD and evaluated its quality by minimizing the residuals between transformed and actual 3D points. Matches with residuals below a threshold ($\delta_{\text{MSAC}} = 3.0$) were considered inliers. After testing various thresholds, we found that setting $\tau_{\text{inliers}} = 17$ provided the best balance between retaining valid matches and discarding outliers, ensuring robust and accurate transformation estimation.

Transformation Refinement: To address residual errors in transformations, we applied Iterative Closest Point (ICP) as a post-alignment refinement step. ICP improved the alignment quality, particularly for frames with partial overlaps. Key parameters included a maximum of 70 iterations and a tolerance of 10^{-5}.

The table below shows the optimal values we identified for the hyperparameters:

Hyperparameter	Value
δ_{MSAC}	3.0
N_{MSAC}	1900
τ_{inliers}	17
N_{ICP}	70
ICP Tolerance	10^{-5}

TABLE II: Hyperparameters of the Model for Part 2

B. Graph Structure and Results

The graph was constructed with nodes representing frames and edges representing transformations. Matches were initially filtered in 2D using epipolar constraints, and the inliers were further refined in 3D. Figure 11 illustrates the graph, with high-connectivity nodes indicating reliable matches and sparse connections representing transitions or low-overlap frames.

C. Point Cloud Alignment

The final merged point cloud was constructed by applying composite transformations to align all frames with the reference. Figure 12 shows the final result, which demonstrates the alignment with some distortions we could not avoid on this dataset.

D. Failure Modes

Despite overall good performance, the pipeline exhibited the following key failure modes:

Excessive Outliers in Matches: Outliers in the initial matches often led to incorrect transformations during MSAC or Kabsch, propagating errors in the final composite transformations.

Sparse Keypoints: Frames with insufficient keypoints resulted in unreliable matches, causing weak graph connections.

Error Propagation: Errors in initial transformations were amplified when computing composite transformations across multiple frames, leading to noticeable distortions in the merged point cloud.

Repetitive Patterns: Frames with repetitive structures produced incorrect matches that bypassed filtering, leading to misalignments in the affected regions.

APPENDIXES

1. Images Part 1.2

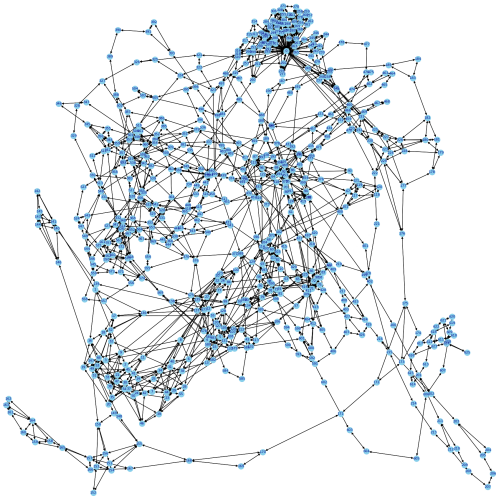

FIG. 1: Graph obtained for the volley dataset

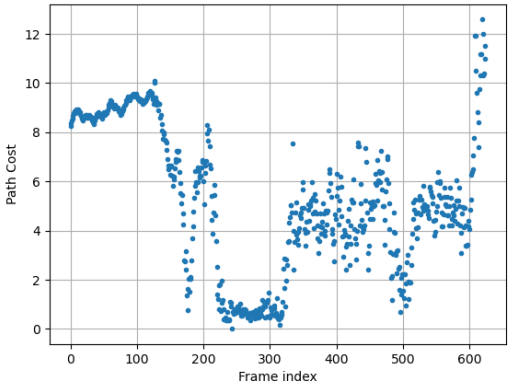

FIG. 2: Path cost associated with the composite homography for each frame for the volley dataset

2. Images Part 2

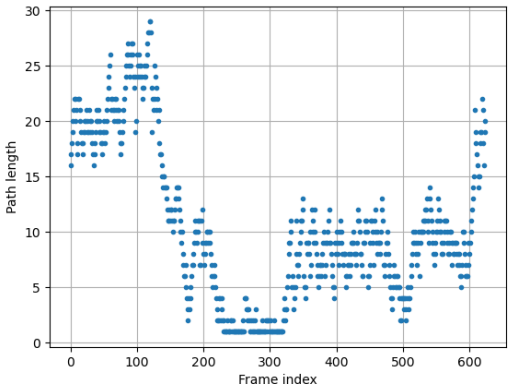

FIG. 3: Path lentgh associated with the composite homography for each frame for the volley dataset

FIG. 4: Coordinate tranformation for the Frame 3

FIG. 5: Coordinate tranformation for the Frame 187

FIG. 6: Coordinate tranformation for the Frame 487

FIG. 7: Coordinate tranformation for the Frame 618

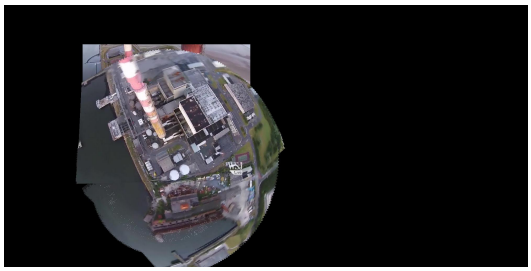

FIG. 8: Coordinate tranformation for the Frame 155

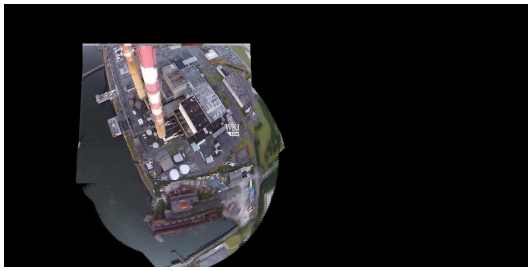

FIG. 9: Coordinate tranformation for the Frame 487

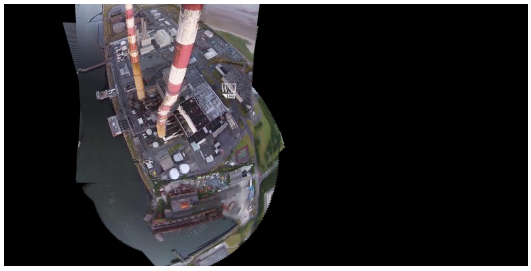

FIG. 10: Coordinate tranformation for the Frame 618

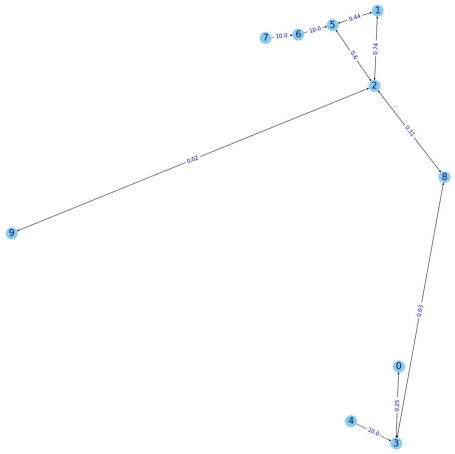

FIG. 11: Graph obtained for the office dataset

FIG. 12: Merged pointclouds for the office dataset