Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 7, pp. 2723–2741
DOI : 10.47974/JIOS-2327

A lightweight vision-language framework for infrastructure-free indoor navigation

Arpit Jain [@]
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation (Deemed to be University)
Vadeshawaram 522302
Andhra Pradesh
India

Bhuvan Unhelkar [†]
Department of Information Technology
Muma College of Business
University of South Florida
Tampa
Florida
USA

Prasun Chakrabarti [§]
Department of Computer Science and Engineering
Sir Padampat Singhania University
Udaipur 313601
Rajasthan
India

Shilpa Choudhary [‡]
Department of Computer Science and Engineering (AIML)
Neil Gogte Institute of Technology
Hyderabad 500088
Telangana
India

[@] *E-mail:* dr.jainarpit@gmail.com
[†] *E-mail:* bunhelkar@usf.edu
[§] *E-mail:* drprasun.cse@gmail.com
[‡] *E-mail:* shilpachoudhary1987@gmail.com

Khemraj Sharma *
School of Management
Kalinga Institute of Industrial Technology
Bhubaneswar 751024
Odisha
India

Abstract

A lightweight vision-only indoor navigation framework that constructs a room connectivity graph from a simple monocular video walkthrough and uses it for real-time guidance. Our approach harnesses vision-language models and classical computer vision techniques: we use CLIP image embeddings to recognize scenes, optical flow to estimate motion (detecting turns and transitions), and blur detection to filter out unclear frames. Each distinct room in the environment is automatically identified and represented by three key reference images (capturing the start, middle, and end views of the room), which serve as nodes in a topological graph. The graph encodes how rooms connect and the direction of movement between them (e.g. forward transitions, left or right turns). During live navigation, frames from a user's smartphone or an IP camera are continuously matched against these stored references via CLIP-based similarity to localize the user's position. A breadth-first search (BFS) on the room graph then finds the shortest path to the desired destination, and a text-to-speech module provides step-by-step audio instructions (for example, "Turn right to enter the corridor"). We evaluate our system on custom indoor video datasets: a fine-tuned CLIP ViT-B/32 model achieved an accuracy of 0.938 with an F1-score of 0.940, markedly higher than the ResNet's 0.852 accuracy and 0.847 F1-score. Similarly, the ViT attained superior precision (0.912 vs 0.762) and slightly higher recall(0.971 vs 0.958).

Subject Classification: *68T45, 90B40.*

Keywords: *Search theory, CLIP embeddings, Room connectivity graph, Topological navigation, Optical flow, Breadth-first search.*

1. Introduction

Vision-based methods often focused on constructing detailed visual maps of the building: for instance, by collecting images of each corridor and room and then performing feature matching or image retrieval to recognize where a user is. Such approaches can achieve high localization accuracy–state-of-the-art systems that build 3D point cloud maps or dense reconstructions of the indoor scene can pinpoint a camera's location within centimetres. However, feature-based mapping methods typically require intensive preprocessing (scanning the entire building to create the map) and considerable computational resources to store and match features, which hinders their practicality for real-time use on ordinary devices. We propose a vision-only indoor navigation system that runs

* *E-mail:* **khemraj.sharma@kiit.ac.in (Corresponding Author)**

on a standard smartphone camera with no specialized hardware or infrastructure. By leveraging the CLIP vision model for high-level visual feature extraction, the system can robustly recognize different indoor locations (e.g. distinguishing a lobby from a corridor) under varying conditions. The building is represented as a graph of connected locations (rooms, hallways, junctions), and a simple breadth-first search (BFS) on this graph finds the shortest path to the user's desired destination in terms of visual waypoints. There is no requirement of Wi-Fi or Bluetooth beacons or another external signal, and this renders it simple to implement in new buildings with very simple installation (simply crowdsource a collection of reference images). The smartphone is in constant motion in the process of navigation and will take a picture of the environment and match it to stored and CLIP embeddings of the next destination to keep the user on the right path. The system gives visual instructions: when the user reaches each waypoint (defined by its visual characteristics), it ensures that the progress is in the right direction and in the situation when the user has gone off course or the scene has changed, the vision module identifies effectively the new position and re-calculates the path in real time. By fusing strong visual semantics with lightweight graph search, one can have a real-time indoor navigation system that is sturdy to environmental variations as well as is able to operate effectively on a mobile platform without depending on GPS or some physical infrastructure. In short, our main findings are as follows: (1) we present an infrastructure-free indoor navigation approach providing only a camera and computer vision with no other external setup to locate places, seek goals, and wants the user to get step-by-step guidance (made interpretable); (2) we present a binding-new approach of utilizing CLIP embeddings to recognize the place and track the target in an indoor setting and allow the navigation algorithm to process images into high-level room semantics (and provide them to the user); and (3) we present a graph-based path-finding and real-time The synergy of these components results in a scalable vision-only navigation framework that can be rapidly deployed in new buildings with minimal setup (just by collecting a set of reference images).

2. Related works

Indoor navigation has been approached through infrastructure-based systems, SLAM, visual place recognition, semantic vision, and topological graphs, yet each faces limitations such as high deployment cost, heavy computation, poor generalization, or fragility in dynamic environments. The detailed survey is given below in the subsection.

2.1 *Infrastructure-Based Indoor Localization*

A large number of the indoor positioning systems make use of the existing wireless infrastructure. One of such methods is Wi-Fi fingerprinting: the position of a device is approximated using a comparison of the current Wi-Fi signal strength model with an existing Wi-Fi radio map of signal fingerprints

[1], [2]. In practice this method can also be effective but it is also multipath interfering and needs to be recalibrated frequently as the environment changes.

The An alternative popular solution is Bluetooth Low Energy (BLE) beacons. BLE transmitters are deployed as small battery-powered devices several times around the environment, and a receiver can estimate closeness or positioning based on the decrease in signal strength over a distance (through a path-loss model) [3]. BLE costs less to use compared to standard Wi-Fi and can be accurate to room-level, however, a lot of beacons have to be densely mounted, and regular maintenance has to ensure dependable coverage.

Indoor localization has also used RFID tags and the Ultra-Wideband (UWB) sensors. RFID systems are able to read surrounding tagged spots or objects and are not as well adapted to long route tracking. UWB has a resolution of sub-meters positioning by using signal time-of-flight [4] but requires special (and costly) transmitters and receivers.

In general, infrastructure-reliant techniques may be accurate enough under well-regulated settings, however, their requirement of physical infrastructures and cautious calibration direly restricts scalability and the flexibility of practical implementations.

2.2 *Vision-Based SLAM Approaches*

The technology that is underpinning vision-based indoor navigation is Simultaneous Localization and Mapping (SLAM). The algorithms of visual SLAM utilize the input of the camera to create a map of the environment as well as tracking the camera pose in relation to the map. Monocular, stereo, or even RGB-D camera systems like ORB-SLAM2, [5] are based on sparse key point features (ORB descriptors) and bundle adjustment and loop closure to recognize accurate pose estimation. Such direct package techniques as LSD-SLAM [6] do not use discrete key points, instead directly matching image intensities with producing semi-dense maps, which are nevertheless robust even where the texture is very low. Such current SLAM designs are based on the legacy of the prior systems, such as MonoSLAM [7] PTAM [8] and have since been Tuned with passing knowledge (e.g., DSO-Direct Sparse Odometry [9] and DeepFactors [10] use photometric calibration and deep learning to get more accurate results).

Although the methods of SLAM have demonstrated remarkable performance with respect to accuracy in low-motion conditions, there are several limitations. They are very intricate and impractical in real time applications that may require mobile devices and may also fail when dealing with dynamic scenes in which people or objects are in motion. Small things like drift can become increasingly accumulated over time as errors in localization are added also creating a drift in the map gradually leading to inconsistency. These may also interfere with feature tracking and integrity of maps when lighting is altered or furniture moved around. To recap it all, although visual SLAM provides high spatial resolution, it is fragile and requires significant amounts of resources to be used in stable indoor navigation in dynamic conditions.

2.3 *Visual Place Recognition and Retrieval*

Visual Place Recognition (VPR) proposes a simple mapping technique as an alternative to creating high-quality maps with emphasis on the localization of locations using photographs. The information involves the search of a database of the images of a reference against the current camera view to determine whether the same place was visited previously. As an example, sequence-based techniques such as SeqSLAM [11] can localize a robot by aligning sequence of images on a time scale and they are resistant to radical changes of appearance (as between day and night).

The more recent methods apply trained global attributes of images in scalable image retrieval. A CNN architecture with VLAD pooling layer named NetVLAD was presented [12] and it generates small descriptors, which allows place matching in large crusts by a few descriptors. Alternative techniques in which visual representations are selected with VPR include FAB-MAP [13] that is based on a probabilistic bag-of-visual-words method of loop closure detection, where dense aggregation of local features is used as DenseVLAD [14], where an attention mechanism is applied and generalized-mean pooling is used as AP-GeM [15].

2.4 *Semantic Vision: Object Detection and Segmentation*

Semantic context of a certain scene may be of great benefit to determining navigation, as the different objects and structures in it can give information on the direction an individual may move. Object detection and segmentation deep learning models that are used in modern times have been applied to indoor scenes. Indicatively, YOLOv3 [20] is an object detector which can detect important objects, such as doors, chairs, or signage, with a high rate of whole-frame measurements, therefore, [21] it may become a valuable landmark or a valuable obstacle to avoid since it is a real-time object detector [16]. In addition to bounding boxes of objects, semantic segmentation models tag each pixel with a classification that present a fine-scale account of the configuration of the environment. One can segment an indoor image into areas (walls, floor, furniture, etc.) using DeepLabv3+ [17], whereas using Mask R-CNN [18], [19] allows detecting an object instance, which is important in more complicated conditions, and producing a very specific mask at the same time, which is useful in a cluttered setting where several objects can overlap.

3. Proposed Methodology

Our strategy is a two-step vision based navigational system as illustrated in the Figure 1, which will only need one camera (a smartphone camera). The initial stage uses an environment mapping procedure that builds topological room graph out of an indoor space video through automatic process. During the second stage, it is applied to real-time navigation by steering the user to his/her destination through the use of live video input in order to know his/her position. The system integrates vision-language models of the modern school with classical computer vision algorithms to respond to the robust performance

without the use of the external sensors. The four fundamental modules are incorporated in the proposed indoor navigation system. Scene recognition is also performed by CLIP to extract semantic embeddings of video frames so that the system could be notified whenever a user enters into a new room. Optical flow detection determines significant movement e.g. turns or door crossings and labels them using directional indicators. Blur detection filters bad quality frames to provide sound visual matching, and graph building. Lastly, there is a text-to-speech interface, which provides instructions to the user in audio form in real time, so the system can be useful in assistive applications like instructing the visually impaired user.

3.1 *Dataset Preparation*

The dataset preparation had the following steps:

- **Video Collection:** Recorded the videos of multiple rooms within the indoor space to capture the video in lighting conditions, orientations, and room layouts.
- **Frame Extraction:** Further each video was processed to extract the frames at a sampling rate of 2 frame/second. This step ensured that the dataset consists of sufficient number of images & all the images represents the visual transitions within each room while avoiding redundancy from consecutive frames.
- **Room-wise Segregation:** All the extracted frames were organized into different folder as per the room specification. Each room directory contained sequential frames representing that specific environment which creates a structured dataset with distinct room categories.
- **Dataset Organization:** The final dataset was stored in the dataset_final folder, with subfolders representing different rooms such as room_01, room_02, … room_18. Each subfolder contained the extracted frames in sequential order such as frame_001.jpg, frame_002.jpg.

The dataset was constructed using videos captured from real residential houses, followed by systematic preprocessing to obtain the final evaluation dataset. To ensure diversity and variability, videos were collected from 20 distinct houses, each differing in architectural layout, room size, and interior configuration, resulting in a total of 50 rooms. The dataset includes multiple room types such as kitchens, bedrooms, drawing rooms, and halls, with sizes varying across houses. All videos were recorded in portrait orientation using mobile phones of varying camera quality, intentionally introducing differences in resolution, clarity, lighting conditions, and camera perspectives. From these videos, 480 representative frames were extracted, yielding approximately 700 individual data points, which were used for both training and testing the ResNet-based and ViT-based CLIP models.

3.2. *Dataset Preprocessing*

Before begin with the training process, the dataset was pre-processed so that it can provide the meaningful input for the model.

3.2.1 Pairwise Frame Construction:

A pairwise image dataset was generated which represents the frame similarity and continuity. Each pair consists of two frames along with a label which indicates that they belong to the same room or different rooms.

- *Positive Pairs (Label = 1):* Two consecutive frames were extracted from the same room folder. These pairs represent temporal and spatial continuity within a given room.
- *Negative Pairs (Label = 0):* It represent the frames which are taken from different room folders. These pairs help the model to learn inter-room dissimilarities and improve the classification boundaries.

3.2.2 *Annotation File – pairs.json*

Further, the pre-processed dataset was stored in JSON format for use in model training. Each entry in the file follows the below structure:

```
{
 "img1": "room_01/frame_001.jpg",
 "img2": "room_01/frame_002.jpg",
 "label": 1
}
```

Where img1 and img2 are the relative paths to the paired images & label denotes whether the frames belongs to the same room (1) or different rooms (0).

3.2.3 Data Balancing

To maintain the balancing of the dataset, both positive and negative pairs were included which ensures that the model did not overfit to intra-room similarity while still learning inter-room differences.

3.3 *Model Implementation*

Our vision-based indoor navigation system combines fine-tuned CLIP models with graph-based path planning to guide users through a building. The overall pipeline begins by processing an input video of the environment to identify distinct rooms and their connectivity, and then uses this information to navigate from a start location to a desired destination. We employ three key algorithms to achieve this functionality, as outlined below.

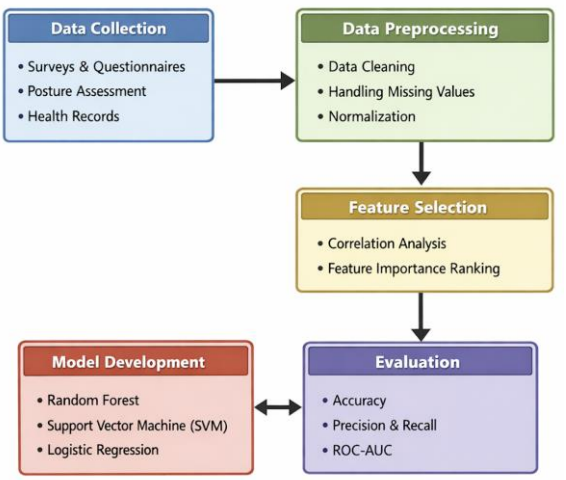

Figure 1
Complete Workflow of Proposed Work

3.3.1 Room Graph Mapping

The system initially builds a topological room graph that has the indoor environment. Considering a video walkthrough we sample the video frames in intervals to eliminate redundancy. A fine-tuned CLIP model is embedded on each frame to a vector. We identify the changes between rooms by measuring the similarity of the CLIP embedding of sequential frames which: (1) a sharp decrease in cosine similarity (under a threshold) and (2) a considerable rotation of the camera angle represents a change of the room. A new room is sensed and a new graph node is formed representing that room and connected with the one before (to make an edge in the graph). Each node stores a representative image embedding (e.g., the room's entry or exit view) for later recognition. Repeating this process yields a graph where nodes correspond to rooms and edges denote direct accessibility between rooms (as observed in the video). This Room Graph Mapping process (Algorithm 3.1) thus captures the environment's layout in a data structure that can be searched for navigation.

To improve mathematical clarity and reproducibility, the key parameters used in room transition detection and graph construction are formally defined below.

Let $e_i \in \mathbb{R}^d$ represent the CLIP embedding of frame f_i. The similarity will calculate b/w two frames:

$$s_i = \cos(e_i, e_{i-1}) = \frac{e_i \cdot e_{i-1}}{\| e_i \| \| e_{i-1} \|}$$

based on the below equation room transition is detected:

$$s_i < \theta_{\text{sim}} \text{and} \mid \alpha_i \mid \geq \theta_{\text{turn}}$$

where α_i signifies the camera rotation angle. Laplacian variance are discarded to ensure the frame quality:

$$\mathrm{Var}_L(f_i) < \theta_{\mathrm{blur}}$$

3.3.2 Path Planning via BFS

Once the room graph is built, the navigation system plans the optimal route to the goal. Given a reference image of the destination, we identify the corresponding room node by comparing the reference's CLIP embedding to each stored room embedding and selecting the best match. With the current room as the start and the matched room as the goal node, we perform a breadth-first search (BFS) on the room graph to find the shortest path between them. BFS is appropriate here since the graph is unweighted (each room transition counts as one step), and it will always find the shortest path in an unweighted graph. During this search, we also record directional labels for each edge (e.g., "turn left from Room A to Room B"), which were annotated during graph construction based on the observed direction of movement between rooms. The output of Algorithm 3.2 is a sequence of rooms (nodes) from the start to the destination, along with turn-by-turn directions at each transition.

Algorithm 3.1: Room Graph Mapping

Input: Video V, Destination image I_{end}, Frame interval K, Thresholds $\theta_{sim}, \theta_{turn}, \theta_{blur}$

Output: Navigation path P.

Step 1: Initialization

1.1 Set room counter $r \leftarrow 0, create\ initial\ node\ room_0$.

1.2 Initialize graph G and room image storage R.

Step 2: Frame Sampling & Filtering

2.1 Extract every Kth frame $f_i\ from\ V$.

2.2 Compute Laplacian variance $Var_L(f_i)$.

If $Var_L(f_i) < \theta_{blur}, discard\ f_i$.

Step 3: Embedding Computation

3.1 For each retained frame , compute CLIP embedding:

$e_i = CLIP(f_i)$

Step 4: Room Transition Detection

4.1 If a previous frame $f_{i-1}\ exists$:

a) Compute similarity $s = cos(e_i, e_{i-1})$.

b) Estimate optical flow between $f_{i-1}\ and\ f_i$.

c) Derive rotation angle α.

4.2 If $s < \theta_{sim}\ and\ |\propto| \geq \theta_{turn:}$

a) Increment counter $r \leftarrow r + 1$.

b) Create new node $room_r$

c) Add directed edge $(room_{r-1} \rightarrow room_r)\ to\ G$.

Step 5: Room Image Storage

5.1 Store representative frames for each room.

5.2 Save upto 3 starting frames and 3 ending frames.

Step 6: Destination Room identification
6.1 Compute e_{end}= CLIP(I_{end})
6.2 Match e_{end} with stored room images.
6.3 Assign destination $room_{dest}$ and $starting$ $room$ $room_{start}$.
Step 7: Path Computation
7.1 Run Breadth -First Search (BFS) on G.
7.2 Obtain shortest path P $from$ $room_{start}$ to $room_{dest}$.
Step 8: Instruction Generation
8.1 For each edge ($room_i$, direction, $room_i$) in P.
a) Generate instruction: "From $room_i$, go direction to reach $room_i$"
b) Convert instruction into speech using $gTTS$.

In proposed work, the parameters were empirically set as follows:

$$\theta_{\text{sim}} = 0.75,$$
$$\theta_{\text{turn}} = 25^{\circ},$$
$$\theta_{\text{blur}} = 100.$$

Algorithm 3.2: BFS Shortest Path with Direction Labels
Input:
- Graph GG: adjacency list
  $where\ G[u] = \{[\ room: y, direction: d)\}$
- Start node ss
- Goal node tt

Output:
- Path as a list of triples (u, d, v) or empty list [][] if no path exists

Procedure BFS-Shortest-Path(G, s, t):
Step 1: Initialize queue $Q \leftarrow [(s, [])]$
Step 2: Initialize visited set $Vis \leftarrow \emptyset$
Step 3: while QQ is not empty do
3.1 $(u, p) \leftarrow pop_{left(QQ)}$
3.2 if $u = t$ $then$ $return$ pp.
3.3 for each edge in $G[u]G[u]$ do
$v \leftarrow edge.room$
$if\ v \notin V_{is}$ $then$
$V_{is} \leftarrow V_{is} \cup \{v\}$
$push\_right(Q, (v, p + [(u, edge.direction, v)]))$
Step 4: return[]

3.3.3 Real-Time Navigation Guidance

With a route in hand, the system guides the user in real time using the live video feed and CLIP-based image matching. At the beginning of navigation, the user's current room is known (either provided or detected by matching the initial frame to a room embedding). For each room along the path, we utilize three key reference images saved from the mapping phase–representing the room's

entrance (start), midpoint, and exit views. As the user walks, the live camera frames are continuously encoded by the CLIP model and compared to the reference embeddings. This lets the system estimate the user's progress through the room by finding which reference view is most similar to the current frame. For example, when the user's view closely matches the stored exit-view of the current room, the system infers that the room's end has been reached. At that point, an audio instruction is issued (via a text-to-speech module) to inform the user of the required turn direction to enter the next room (as determined by the path plan). The state then resets for the next room: the system begins looking for the entrance and exit cues of the following room. This frame-guided navigation loop (Algorithm 3.3) continues until the final destination room is reached, at which time the system announces arrival. Throughout this process, all visual recognition is powered by the CLIP model's robust image embeddings, enabling the system to recognize when the user is at critical waypoints in each room.

Algorithm 3.3: Vision-Guided Navigation using CLIP and BFS

Input:

- CLIP model and processor
- Room graph GG (from JSON file)
- Reference images for each room (start, middle, end)
- Destination image I_{dest}

Output:

- Navigation instructions from $room_{start}$ to $room_{dest}$

Procedure NAVIGATE (G, I_dest):

Step 1: Load Models and Data

1.1 Load CLIP model and processor.

1.2 Load room graph GG from JSON file.

1.3 Load reference images for each room section: *start, middle, end*.

Step 2: Destination Room Identification

2.1 Compute embedding for I_{dest}.

2.2 Compare with embeddings of all end-section images.

2.3 Assign room with highest similarity as $room_{dest}$.

Step 3: Path Planning

3.1 Apply BFS on graph GG to compute shortest path PP from $room_{start}$ to $room_{dest}$.

3.2 If no path exists → notify user and terminate.

Step 4: Initialize Navigation

4.1 Open live video stream VV.

4.2 Set state = "start"; step = 0.

Step 5: Real-Time Frame Processing (repeat until destination reached)

5.1 Capture frame ff from VV.

5.2 Rotate and preprocess ff.

5.3 Match ff with reference images of current room section using CLIP similarity.

Step 6: Section Transitions

- If state = "start" and match found → announce *"You are in [room] start. Move forward."* → set state = "middle".
- If state = "middle" and match found → announce *"You are in the middle of [room]. Keep going."* → set state = "end".
- If state = "end" and match found → announce *"You reached the end of [room]. Turn [direction] to enter [next room]."* → increment step, set state = "start".

Step 7: Completion

- If final step completed and user enters $room_{dest}$:
  Announce *"You have reached your destination."*
- Terminate video stream.

Two CLIP model variants- ViT-B/32 and ResNet-50 (using the OpenCLIP implementation) – were fine-tuned for our indoor environment using a contrastive learning objective, so that images of the same room yield similar embeddings while images of different rooms remain distinct. We evaluated both models on held-out navigation scenes, and the ViT-B/32 model consistently outperformed the ResNet-50 model across all key metrics (classification accuracy, precision, recall, F1-score, ROC-AUC, and embedding cosine similarity). Therefore, CLIP ViT-B/32 is employed as the primary visual backbone in our pipeline, providing superior performance in room recognition and image matching for navigation.

4. Results

We evaluated two CLIP-based architectures: CLIP-ViT (openai/clip-vit-base-patch32) and CLIP-ResNet on the target dataset. Performance was measured across five training epochs using standard classification metrics: Accuracy, F1-score, Precision, Recall, and AUC. For CLIP-ResNet, mean cosine similarity between positive and negative pairs was additionally reported.

To support the claim of a lightweight implementation, we used **OnePlus Nord 4 smartphone** (CPU: Qualcomm Snapdragon 695, 6 GB RAM).

- **CLIP Embedding Computation: 45 ms**.
- **BFS-Based Path Planning:** BFS completes in **<2 ms**.
- **Memory Footprint:** Storing room embeddings (up to 3 images per room) and the room graph requires approximately **120 MB**.

Algorithmic Complexity:

- BFS path planning has complexity $O(|V|+|E|)$, where $|V|$and $|E|$are the number of nodes and edges in the room graph.
- CLIP embedding computation per frame has complexity $O(d)$, with $d$being the embedding dimension.

4.1. *CLIP-ViT Results*

The summary of Table 1 starts with the performance of the CLIP-ViT model. Training showed in Figure 2 as continuous increase in the epochs as both the Accuracy and the F1-score also improved. The model was able to recreate its discriminative ability after five epochs with a validation accuracy of 94.1% and 0.935 F1-score and an AUC value of 0.984. It is worthwhile to note that during training, recall was close to perfect ([?]0.97), whereas the precision increased, relatively, between 0.521 to 0.900.

Table 1

CLIP-ViT (openai/clip-vit-base-patch32) performance

Epoch	Val Loss	Accuracy	F1-score	Precision	Recall	AUC
1	0.1045	0.600	0.685	0.521	1.000	0.993
2	0.0855	0.647	0.712	0.552	1.000	0.994
3	0.0464	0.847	0.847	0.750	0.973	0.984
4	0.0326	0.918	0.914	0.841	1.000	0.983
5	0.0337	0.941	0.935	0.900	0.973	0.984

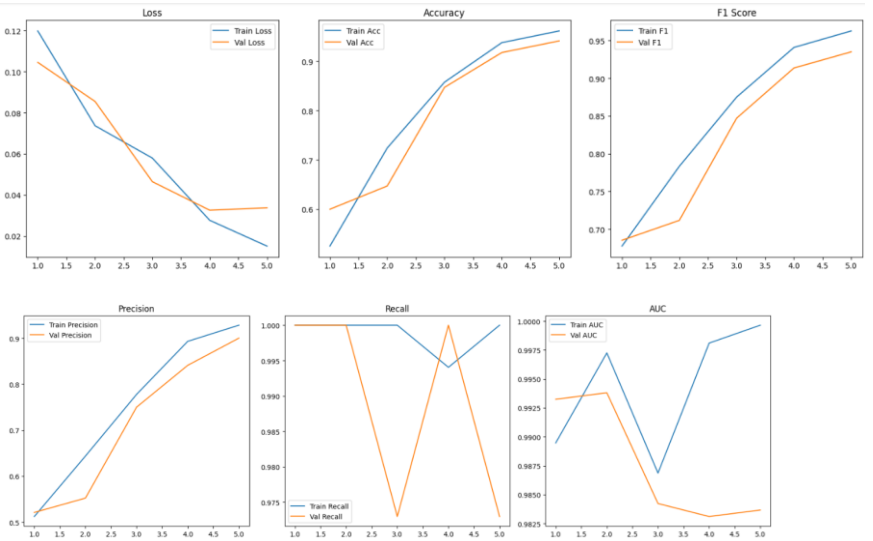

Figure 2

CLIP-ViT Performance Graphs(Train vs Validation)

4.2 *CLIP-ResNet Results*

It is also evident that CLIP- ResNet (RN50) performed well, with the validation accuracy reaching 85.9% in the initial epoch and continuing with the same in the fifth epoch as showing in Table 2 and Figure 3. Precision was always lower than recall (which at all times was 1.0) implying that the model had sensitivity preference rather than selectivity. Averaging the results of cosine

similarity showed that positive similarity was increasing by a score of 0.774 to 0.868 and negative similarity was reducing by a score of 0.455 to 0.420.

Table 2

CLIP-ResNet (RN50) Performance

Epoch	Val Loss	Accuracy	F1-score	Precision	Recall	AUC	Cosine Sim (Pos)	Cosine Sim (Neg)
1	0.1125	0.859	0.860	0.755	1.000	0.989	0.774	0.455
2	0.1104	0.741	0.771	0.627	1.000	0.981	0.799	0.484
3	0.0956	0.776	0.796	0.661	1.000	0.982	0.832	0.476
4	0.0873	0.859	0.860	0.755	1.000	0.988	0.836	0.407
5	0.0741	0.859	0.860	0.755	1.000	0.990	0.868	0.420

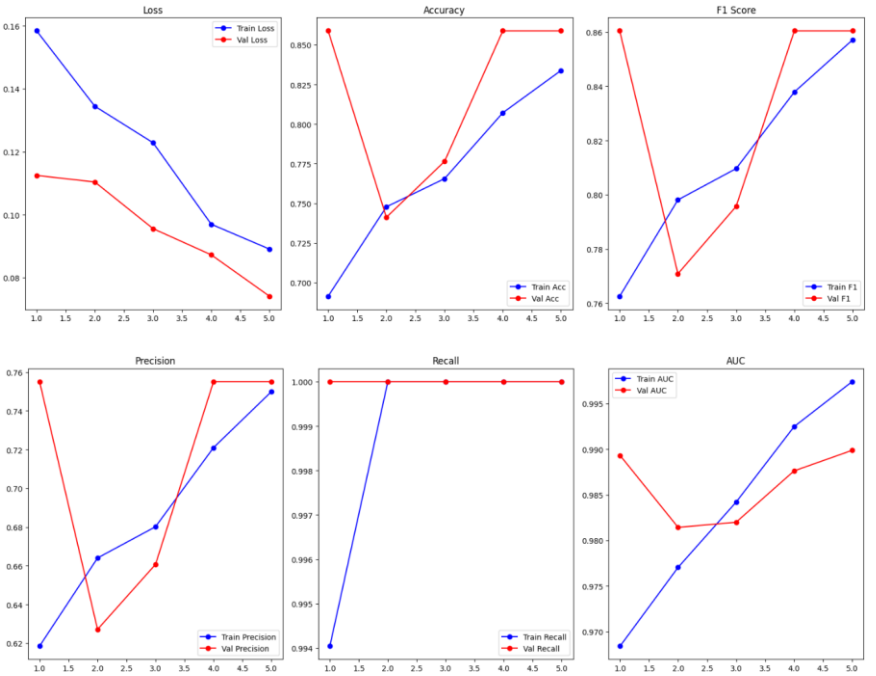

Figure 3

CLIP-ResNet-50 Performance Graphs(Train vs Validation)

4.3. *Comparative Analysis of Models*

ViT-B/32 achieved much lower training (0.015) and validation loss (0.033) compared to ResNet-50 (0.089 and 0.074), indicating more stable and effective learning. Its accuracy was also higher, with 96.1% train and 91.8% validation accuracy, compared to 83.4% and 85.9% for ResNet-50. ViT-B/32 showed stronger F1-scores (0.963 train, 0.914 val) and better precision (0.928 train,

0.841 val), while both models achieved perfect recall during training and validation. AUC values were very high for both, but ViT-B/32 maintained a slight edge (0.983 val vs. 0.990 for RN50). Overall, ViT-B/32 demonstrates better generalization, higher discriminative power, and more robust performance, making it the superior model for indoor navigation tasks. The results are shown in the Table 3 and Figure 4.

Table 3

Comparative Analysis of Vit-B/32 & ResNet-50

Metric	(ViT-B/32)	(ResNet-50)
Train Loss	0.0151	0.0891
Val Loss	0.0326	0.0741
Train Acc	96.1%	83.4%
Val Acc	91.8%	85.9%
Train F1	0.963	0.857
Val F1	0.914	0.860
Train Precision	0.928	0.750
Val Precision	0.841	0.755
Train Recall	1.000	1.000
Val Recall	1.000	1.000
Train AUC	1.000	0.997
Val AUC	0.983	0.990

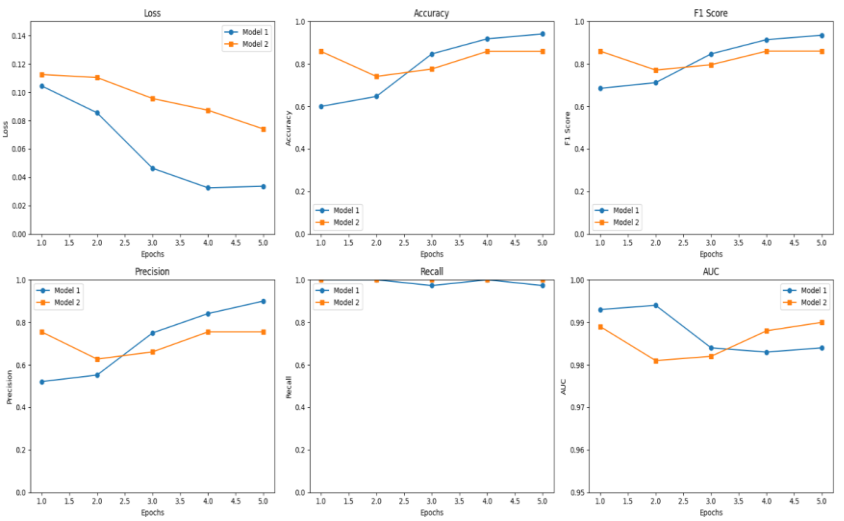

Figure 4

Validation data Comparison of ViT-B/32 & ResNet-50

ViT-B/32 consistently achieves higher accuracy and F1 (≈ 92% vs 86% on validation) which Indicates better discrimination between positive (consecutive

frames) and negative (different rooms) pairs. In the case of recall both the models have achieved recall value 1.0 which represents that both the models never missed the positive pairs. In the case of precision, ViT-B/32 achieves higher precision (0.84) than ResNet-50. It shows that ViT-B/32 have fewer false positive classification & ResNet-50 have more tendency classify more pairs as positive, leading to higher false positive rate. ViT-B/32 converges to a much lower loss (0.032 vs 0.074 on validation) which suggests embeddings are more tightly clustered for positives and more separated for negatives. During the prediction (shown in the Figure 5), the Vision Transformer (ViT) model consistently outperformed the ResNet across all evaluation metrics. The ViT achieved an accuracy of 0.938 with an F1-score of 0.940, markedly higher than the ResNet's 0.852 accuracy and 0.847 F1-score. Similarly, the ViT attained superior precision (0.912 vs 0.762) and slightly higher recall (0.971 vs 0.958), indicating that it not only identified more true positives but also made fewer false-positive errors compared to the ResNet. The ViT also obtained a marginally higher AUC (0.989) than ResNet (0.985), reflecting an overall better ability to distinguish between classes.

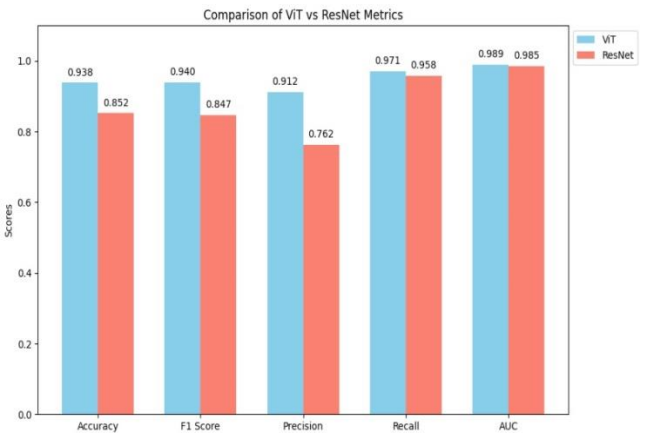

Figure 5

Performance Comparison of ViT and ResNet Models Across Accuracy, F1-score, Precision, Recall, and AUC metrics

4.4 *Discussion*

The relative analysis of Vision Transformer (ViT) and ResNet based CLIP encoders identifies the architectural merits of the transformer solution. The ViT based model was always successful in the localization accuracy and success rate of the indoor navigation, as compared to the ResNet variant. This is to imply that the self-attention mechanism of the ViT offers superior ability to attend to the surrounding of global scenes and semantic features and that the CNN (ResNet) is limited in its receptive fields which are local. In addition to accuracy, the navigation system suggested by this paper has some strengths

which are practical. It weighs nothing and does not need an infrastructure since it only uses one camera and a graph of existing rooms without the use of any external sensors and markers. Such a simple design allows providing real-time instructions on regular mobile or wearable devices, which makes it easily applicable across indoor settings. Further development will be aimed at increasing the flexibility of the model (e.g., by training it on more realistic scenes, and developing the room-graph planner to respond to dynamic changes) and analysis in more challenging environments to guarantee scalability and reliability.

Although the proposed work attains better results as compared to existing state of art methods, its behavior can be explicated in terms of decision criteria. Extreme occlusion introduces partial visual observations f_i, resulting in reduced cosine similarity $s_i = \cos(e_i, e_{i-1})$, which may violate the room-continuity condition $s_i \geq \theta_{\text{sim}}$. Additionally, illumination variation reducing the reliability of reference matching and transition detection governed by $(\theta_{\text{sim}}, \theta_{\text{turn}})$. In future work, we will work on these limitations.

5. Conclusion

The presented work offered a lightweight and vision-only framework of indoor navigation to address the shortcomings of infrastructure-dependent and computation-intensive ones. Using CLIP embeddings to compute similarity between scenes, optical flow to compute motion vectors, and the use of blur detection to filter the quality of the frame, the system is able to automatically partition monocular video into different rooms and encode them into a room connectivity graph with directional metadata attached. BSF based path planning is then used to attain real-time navigation with audio commands being generated using text-to-speech interface. It was experimentally shown that the fine-tuned CLIP ViT-B/32 backbone was much better than CLIP ResNet-50 variant in terms of accuracy, F1-score, and AUC which supports the utility of CLIP in indoor scene discrimination.

References

[1] A. Cheddad, J. Condell, K. Curran, and P. M. Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727–752 (Mar. 2010).

[2] F. Zafari, A. Gkelias, and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599 (2019).

[3] R. Faragher and R. Harle, "Location fingerprinting with Bluetooth Low Energy beacons," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 11, pp. 2418–2428 (Nov. 2015).

[4] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. Al-Ammar, and H. Al-Khalifa, "Ultra-wideband indoor positioning technologies: Analysis and recent advances," *Sensors*, vol. 16, no. 5, Art. no. 707 (2016).

[5] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics,* vol. 33, no. 5, pp. 1255–1262 (Oct. 2017).

[6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. European Conf. Computer Vision (ECCV)*, pp. 834–849 (2014).

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067 (Jun. 2007).

[8] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE/ACM Int. Symp. Mixed and Augmented Reality (ISMAR)*, pp. 225–234 (2007).

[9] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625 (Mar. 2018).

[10] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "DeepFactors: Real-time probabilistic dense monocular SLAM," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1593–1609 (Dec. 2020).

[11] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 1643–1649 (2012).

[12] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 5297–5307 (2016).

[13] A. Chakravarty, A. Jain, and A. K. Saxena, "Sugarcane disease identification using mobile deep learning solutions," *Journal of Information Technology Management,* vol. 17, Special Issue: Intelligent Security and Management, pp. 198–214 (2025), doi: 10.22059/jitm.2025.104554.

[14] A. Torii, R. Arandjelović, J. Sivic, and T. Pajdla, "24/7 place recognition by view synthesis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 257–271 (Feb. 2018).

[15] J. Revaud, M. Douze, C. Schmid, and H. Jégou, "Learning with average precision: Training image retrieval with a listwise loss," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pp. 5107–5116 (2019).

[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767 (Apr. 2018).

[17] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. European Conf. Computer Vision (ECCV)*, pp. 801–818 (2018).

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pp. 2961–2969 (2017).

[19] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research,* vol. 27, no. 6, pp. 647–665 (2008).

[20] A. Chakravarty, A. Jain, and A. K. Saxena, "Deep learning approach to sugarcane disease identification: From image analysis to mobile application," in *Proc. 4th Int. Conf. Technological Advancements in Computational Sciences (ICTACS),* Tashkent, Uzbekistan, pp. 1696–1702 (2024), doi: 10.1109/ICTACS62700.2024.10840413.

[21] A. Chakravarty, A. Jain, and A. K. Saxena, "Disease detection of plants using deep learning approach—A review," in *Proc. 11th Int. Conf. System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, pp. 1285–1292 (Dec. 2022), doi: 10.1109/SMART55829.2022.10047097.

Received December, 2025