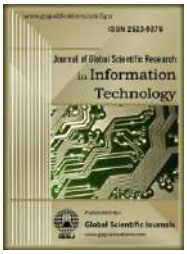
Journal of Global Scientific Research in Information Technology
Vol.11, Issue 4/ pp. 4290-4300/ April 2026

Contents lists available at www.gsjpublications.com

Journal of Global Scientific Research in Information Technology

journal homepage: www.gsjpublications.com/jourgsr



# Fundamentals of Robot Autonomy: Visual Transformer for Camera Localization

**Leqaa Alwan Athoob[1], Sundus Noori Kamber[2], Hanaa Sharhan Hashim[3*]**

*[1,2]Department of Statistics Techniques (Health and Informatics), Institute of Management Rusafa, Middle Technical University, Iraq.*
*[3*]Computer Systems Technologies, Middle Technical University, Iraq.*

ARTICLE INFO

*Received: 29 Jan 2026,*
*Revised: 10 Feb 2026,*
*Accepted: 13 Feb 2026,*
*Online: 27 Apr 2026*

*Keywords*:
Camera Localization, Computer Vision, Deep Learning, Robotics, 7Scenes Dataset.

ABSTRACT

Camera localization is a critical task in the field of computer vision and robotics, involving the precise estimation of a camera's position and orientation within a given environment. This article reviews the application and evaluation of vision transformer models for this estimation purpose, focusing on their potential ability to improve spatial localization accuracy. Using benchmark datasets such as 7Scenes and Cambridge Landmarks, we conducted extensive experiments to assess the performance of transformer-based architectures, particularly Vision Transformers (ViTs). The results indicated that Vision Transformers excel at modeling complex spatial relationships and capturing long-range dependencies in image data, which are essential features for effective camera pose estimation. Compared to traditional Convolutional Neural Networks (CNNs), transformers offer clear advantages in dealing with large and dynamic scenes due to their global attention mechanisms. The study highlights the capabilities of transformer-based models and improving camera positioning to achieve accuracy and reliability related to robotics.

## 1. Introduction

Determining the location and orientation of a camera in a particular setting is known as camera localization, and it is a crucial issue in many technical domains, including augmented reality, robotics, and autonomous driving. In order to determine its location, this activity usually uses visual data, or photos, that the camera has taken. To overcome this difficulty, methods such as Simultaneous Localization and Mapping (SLAM) and Pose Estimation are frequently employed. However, when working in dynamic and complicated situations, these approaches frequently struggle with accuracy and performance The accuracy of localization tasks has significantly improved in recent years thanks to developments in deep learning and artificial intelligence. Because of their self-attention mechanism, transformer networks have become one of these most effective tools. Because of their self-attention

Corresponding author:
E-mail address: hanaa.Sharhan@mtu.edu.iq
doi: 10.5281/jgsr.2026.20007797

mechanism, which enables models to capture long-range relationships between picture components, transformer networks have become one of these most potent tools. Recent research has demonstrated the potential of transformers in image-based tasks, offering notable enhancements in managing spatial relationships across large-scale images, despite the fact that they were first developed for natural language processing tasks. The purpose of this study is to investigate the use of Visual Transformer Networks for camera localization tasks and compare their effectiveness with more conventional CNN-based methods. The 7Scenes and Cambridge Landmarks datasets, which are frequently used benchmarks for camera localization tasks, will be utilized to test the model. To find out if transformers can provide better localization accuracy in comparison, we will evaluate the model's performance using metrics like translation error and rotation error. The goal of this project is to investigate how visual transformers can enhance the accuracy and robustness of camera pose estimation in complex environments.[1]

## 1. Materials And Methods

The Materials and Methods section is a vital component of any formal lab report. This section of the report gives a detailed account of the procedure that was followed in completing the experiment(s) discussed in the report. Such an account is very important, not only so that the reader has a clear understanding of the experiment, but a well written Materials and Methods section also serves as a set of instructions for anyone desiring to replicate the study in the future.

There are several common mistakes that are often found in the Materials and Methods section of a lab report. One major concern is deciding upon the correct level of detail [1]. It is often very easy for a writer to get carried away and include every bit of information about the procedure, including extraneous information like the number of times he (or she) washed their hands during the experiment. A good guideline is to include only what is necessary for one recreating the experiment to know. Keeping this in mind will lead to a Materials and Methods section that is thoroughly written, but without the kind of unnecessary detail that breaks the flow of the writing. Another common mistake is listing all of the materials needed for the experiment at the beginning of the section [1].

## 2. Results And Discussion

The results section is where you tell the reader the basic descriptive information about the scales you used (report the mean and standard deviation for each scale). If you have more than 3 or 4 variables in your paper, you might want to put this descriptive information in a table to keep the text from being too choppy and bogged down (see the APA manual for ideas on creating good tables). In the results section, you also tell the reader what statistics you conducted to test your hypothesis (-ses) and what the results indicated. In this paper, you conducted bivariate correlation(s) to test your hypothesis. If you computed two or more correlations (thus involving at least three variables) provide a table at the end of the paper (ordinarily tables would only be used for even more complex findings, but I'd like you to practice since you have a few correlations to work with). If you include a correlation matrix table, you should, in the text of the result section, refer readers to your table. If you are using Word as your word processor, create the table, then you can adjust the "borders and shading" for each cell/row/column to get the table formatted properly. Other word processors should have similar functions.

## 3. Literature Review

- Visual Transformers In Camera Localization:

With impressive results, visual transformers—which make use of the transformer architecture mainly employed in NLP tasks—have been modified for image-related activities. In contrast to conventional CNN-based architectures that use convolutions to capture local information, they rely on the self-attention mechanism to describe long-range correlations between picture features. Transformers' capacity to more effectively and flexibly capture spatial correlations between picture features than CNNs is one of their main advantages in camera localization.

The Vision Transformer (ViT) introduced by Dosovitskiy et al., treats an image as a sequence of patches (similar to how transformers process words in NLP). This enables the model to recognize global patterns in the image. Some works, such as DETR (Detection Transformers), have extended transformers for image recognition tasks, and there are emerging applications of transformers for camera localization tasks as well, especially for handling large-scale datasets or long-range dependencies.

4. Cnn-Based Approaches:

Convolutional neural networks (CNNs) were the most common architecture for visual tasks, such as camera localization, before transformers. CNNs are very good at learning images' hierarchical feature representations. CNNs are used by a number of algorithms, including PoseNet and DeepLocalization, to predict camera postures from images; these methods usually employ supervised learning techniques on huge datasets. CNNs are very good at capturing local features, but they have trouble handling spatial linkages and long-range dependencies in huge situations. Transformers are advantageous in this situation since they are built to manage these dependencies through self-attention processes.

5. Pose Estimation And Slam (Simultaneous Localization And Mapping):

Determining the location and orientation of a camera in a given space requires pose estimation. SLAM approaches use motion and visual data to map and track the camera's trajectory in real-time. The challenge in posture estimation is to achieve high translation and rotation accuracy without resorting to computationally expensive methods.

6. Dataset Preparation

- Scenes Dataset:

The 7Scenes dataset consists of images from seven different indoor environments, each with a set of annotated camera postures. It is widely utilized for camera localization tasks and provides a variety of testing scenarios.

```
# # Import the necessary libraries
# import os
# import zipfile
# # Determine paths 7Scenes
#                 seven_scenes_dir                 =
"/content/drive/MyDrive/Visualize_7scence"
#                  os.makedirs(seven_scenes_dir,
exist_ok=True)
# # Download data 7Scenes
# scenes_7scenes = ["fire"]
# for scene in scenes_7scenes:
#                                  scene_url          =
f"http://download.microsoft.com/download/2/
8/5/28564B23-0828-408F-8631-
23B1EFF1DAC8/{scene}.zip"
#                                  scene_zip          =
os.path.join(seven_scenes_dir, f"{scene}.zip")
#     if not os.path.exists(scene_zip):
#         !wget -c {scene_url} -O {scene_zip}
# # Decompress files
# for scene in scenes_7scenes:
```

```
# scene_zip = os.path.join(seven_scenes_dir, f"{scene}.zip")
# with zipfile.ZipFile(scene_zip, 'r') as zip_ref:
# zip_ref.extractall(os.path.join(seven_scenes_dir, scene))
seq_dir = "/content/drive/MyDrive/Visualize_7scence/fire/fire"
# List all .zip files in the directory
zip_files = [f for f in os.listdir(seq_dir) if f.endswith(".zip")]
# Unzip each file
for zip_file in zip_files:
    zip_path = os.path.join(seq_dir, zip_file)
    seq_name = os.path.splitext(zip_file)[0] # Extract the name without the .zip extension
    extract_path = os.path.join(seq_dir, seq_name)
    try:
        # Extract the zip file
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall(extract_path)
            print(f"Extracted: {zip_path} to {extract_path}")
    except zipfile.BadZipFile:
        print(f"Skipped bad zip file: {zip_path}")
    except Exception as e:
        print(f"An error occurred with {zip_path}: {e}")
```

Data Preprocessing:

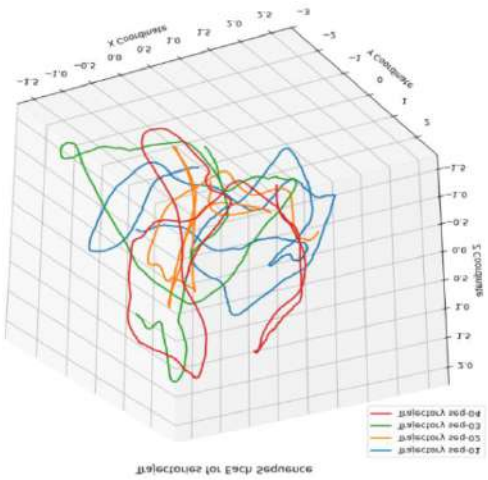
• Resizing: Every image will be shrunk to 224x224 pixels, which is the standard input size for the model.

• Normalization: Depending on the needs of the model, image pixel values will be normalized to either the range [0, 1] or [-1, 1].

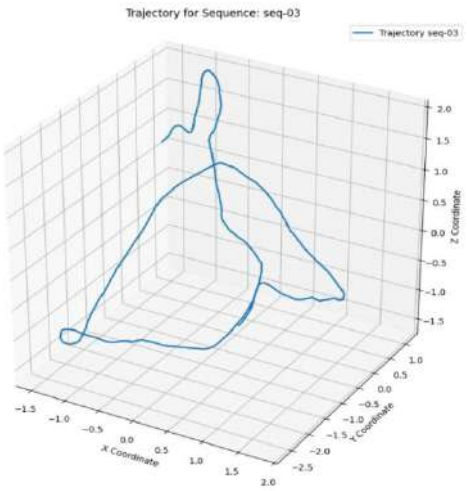
• Augmentation: To enhance the diversity of training data, data augmentation methods including flipping, rotation, and random cropping will be used.

##

Number of sequences: 4
Number of samples: 4000
Number of processed poses: 4000



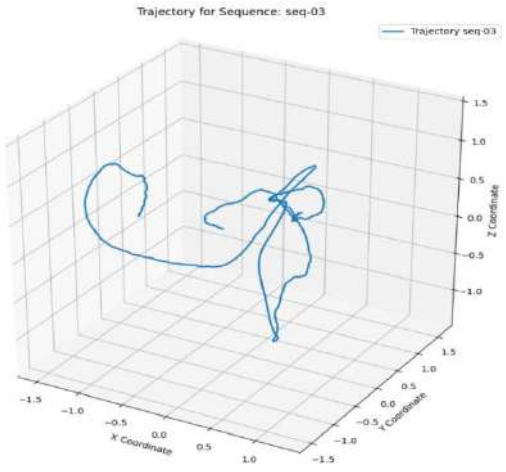
**Fig(1).** Trajectories for Each Sequence in One Figuer



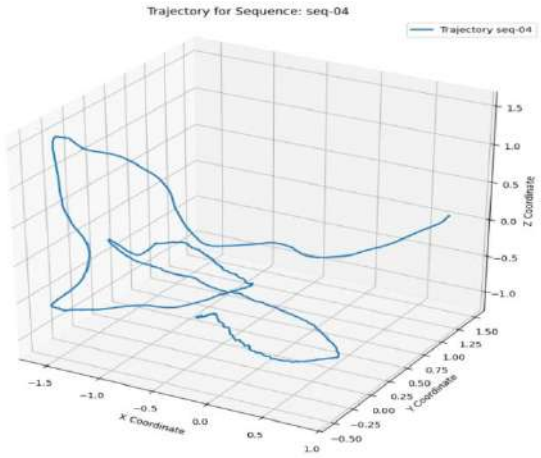

**Fig(2).** Trajectories for Each Sequence in Two Figuer

```
'/content/drive/MyDrive/Visualize_7scence/office'  # Replace with your dataset path
# Initialize the dataset
dataset = OfficeDataset(root_dir=root_dir, transform=transform)
# Plot each sequence's trajectory separately
dataset.plot_trajectories_separately()
```

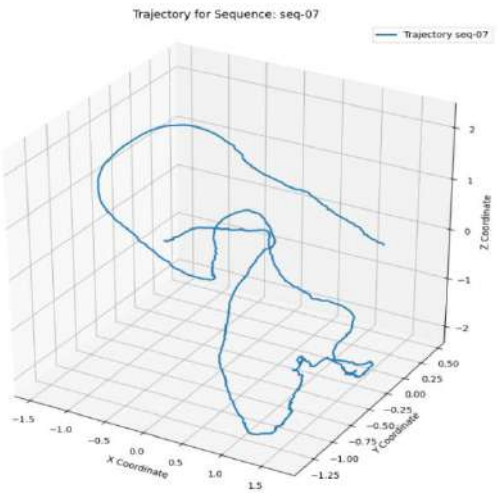
Number of sequences: 10
Number of samples: 10000
Number of processed poses: 10000



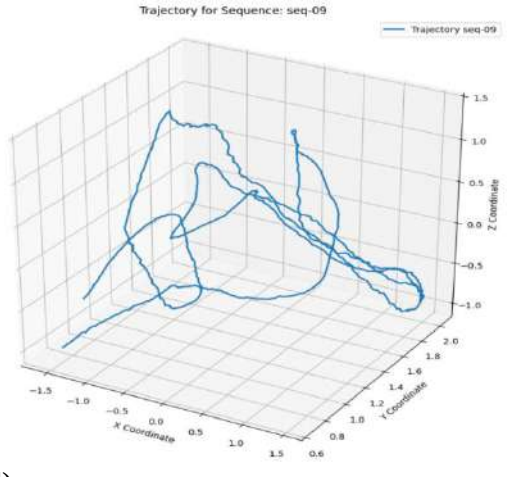

(a)



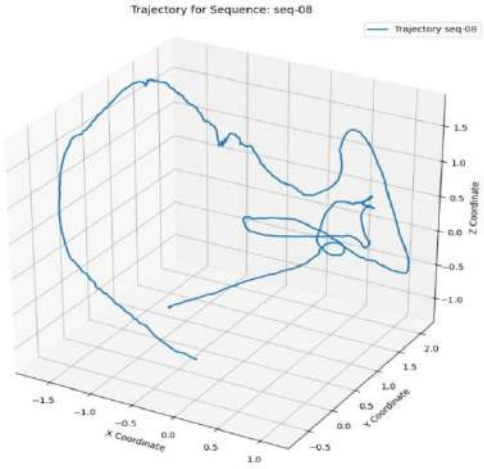

(b)



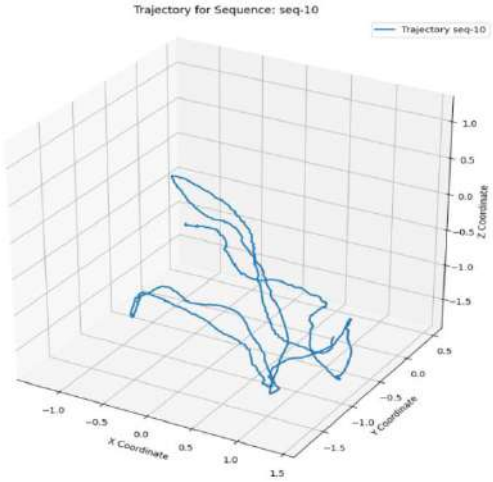

(c)




(d)




(e)




(f)

**Fig(3): (a, b, c, d, e, f) :** Trajectory for each Sequence: {seq}

**Cambridgelandmarks_Kingscollege**

- Dataset Description

The King's College scene is included in the extensive outdoor visual relocalization dataset known as Cambridge Landmarks, which was gathered around Cambridge University. A visual reconstruction of the scene, the original video material, and extracted image frames with their associated 6-DOF camera postures are all included.

- Dataset Contents:
  - Images (.png)
  - Video (.mp4)
  - Text files (.txt) with pose information
  - Scene reconstructions (.nvm format, viewable with VisualSFM)

```
import cv2
import IPython.display as display
from PIL import Image
import time

video_path = "/content/drive/MyDrive/CambridgeLandmarks_KingsCollege/data/KingsCollege/videos/seq-dusk.mp4"
cap = cv2.VideoCapture(video_path)

# Define the new display size (adjust as needed)
display_width = 800  # Adjust width
display_height = 600  # Adjust height

try:
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Resize the frame to fit the screen
        frame = cv2.resize(frame, (display_width, display_height))

        # Convert from BGR (OpenCV format) to RGB (PIL format)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(frame)

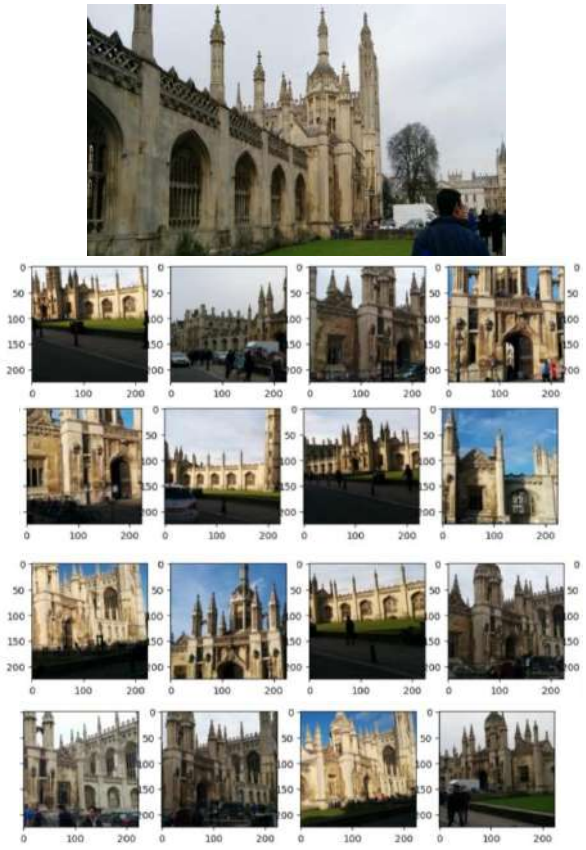
        # Display the frame
        display.clear_output(wait=True)
        display.display(img)

        time.sleep(0.03)  # Adjust speed based on video FPS

except KeyboardInterrupt:
    print("Playback interrupted by user.")

finally:
    cap.release()
    print("Video playback stopped.")
```

```
## Visualize dataset samples
```
from utils.notebook.plotting import plot_im_loader
data_loader = data.DataLoader(data_src, batch_size=8, shuffle=True, num_workers=0)
plot_im_loader(data_loader, row_max=1)
```


```

**Fig(4)**: Visualize dataset samples

**Model Implementation**

Vitlocalization Model For Camera Localization Using Visual Transformers:

- **Objective:** Localize the camera's position and orientation using images, based on the Visual Transformer (ViT) architecture.
- **Key Components:**

1. **Positional Encoding:** Helps the model understand spatial information within the image.
2. **Dataset:** Uses the **Cambridge Landmarks dataset**, which includes images from six urban locations with varying lighting conditions. It provides ground truth poses (camera position and orientation) for model evaluation.
3. **Self-Attention Mechanism:** The transformer's self-attention mechanism allows the model to capture global context and correlations across the entire image, which is crucial for large-scale localization tasks.
4. **Transformer Layers:** The model includes multiple stacked transformer layers with:

- **Layer Normalization** to stabilize training.
- **Feed-Forward Networks** for further processing.
- **Multi-Head Attention** for capturing different contextual relationships in the image.

5. **Pose Estimation:** The model predicts the camera's 6 Degrees of Freedom (6DoF) pose, which consists of:

- **3 translation values** (X, Y, Z).
- **3 rotation values** (represented as a 3x3 rotation matrix).

6. **Loss Function:** Uses a regression approach to minimize pose estimation errors by comparing predicted poses with ground truth.

- **Dataset Class:** CameraLocalizationDataset - A custom dataset class loads images and corresponding camera poses for training and testing.[1]

- **PoseNet Model (Alternative Approach using ResNet-18 for Camera Localization):**

- **Objective:** Localize the camera's position using a ResNet-based architecture for feature extraction.
- **Key Components:**

1. **Backbone:** Uses **ResNet-18** as the feature extractor (pre-trained). The classification head is removed, leaving the feature extraction layers.
2. **Pose Regression:** After extracting features, the model has:

- **Fully Connected Layers** for predicting:
- **Translation:** 3 values for camera position (X, Y, Z).
- **Rotation:** 4 values representing the camera's orientation (in quaternion format).

- **Model Architecture:**
  - **PoseNet** uses a **ResNet-18** backbone, with the last fully connected layer replaced by layers to predict the camera's translation and rotation.

Both models utilize deep learning frameworks like **PyTorch** for implementation and training. The **ViTLocalization** model uses a **Visual Transformer (ViT)**, while the **PoseNet** model uses **ResNet-18** for feature extraction.[2]

- **Training**

- Training The Vitlocalization Model For Camera Localization Using Visual Transformers With 7scence_Dataset

```
# Training loop
def train_model(model, train_loader, epochs=10):
    model.train()
    for epoch in range(epochs):
        total_loss = 0
            for images, translations, rotations in tqdm(train_loader):
            images = images.to(device)
            translations = translations.to(device)
            rotations = rotations.to(device)
            optimizer.zero_grad()
                pred_translations, pred_rotations = model(images)
            # Compute losses
```

```
                translation_loss = translation_loss_fn(pred_translations, translations)
                rotation_loss = rotation_loss_fn(pred_rotations, rotations)
            loss = translation_loss + rotation_loss
            loss.backward()
            optimizer.step()
            total_loss += loss.item()
        print(f"Epoch {epoch+1}/{epochs}, Loss: {total_loss/len(train_loader):.4f}")
# Train
train_model(model, train_loader, epochs=5)
```

**Table 1:** summarizing the training progress

| Epoch | Loss | Training Time per Epoch (mm:ss) | Training Speed (it/s) |
|---|---|---|---|
| 1/5 | N/A | 02:24 | 3.45 |
| 2/5 | 0.0257 | 02:25 | 3.44 |
| 3/5 | 0.0114 | 02:28 | 3.37 |
| 4/5 | 0.0063 | 02:28 | 3.37 |
| 5/5 | 0.0049 | 02:28 | 3.37 |

## Evaluation

The average difference between the predicted and ground truth translation vectors is 0.0225 meters (~2.25 cm). This is a good result for camera localization, indicating the model can localize the camera with high accuracy in terms of position.
Rotation Error: 0.8674 degrees:
The average angular difference between the predicted and ground truth rotations is 0.8674 degrees.
This suggests that the model is also predicting the camera's orientation quite accurately.
The following measures will be used to assess the model's performance:

```
evaluate_model(model, test_loader)
```

Translation Error: 0.0268 m
Rotation Error: 0.4373

**Table 2:** Training Progress and Error Metrics for Camera Localization Model

| Epoch | Loss | Training Time per Epoch (mm:ss) | Training Speed (it/s) |
|---|---|---|---|
| 1/5 | N/A | 01:30 | 5.53 |
| 2/5 | 0.7647 | 01:34 | 5.32 |
| 3/5 | 0.7664 | 01:30 | 5.55 |
| 4/5 | 0.7660 | 01:34 | 5.27 |
| 5/5 | 0.7680 | 01:34 | 5.27 |

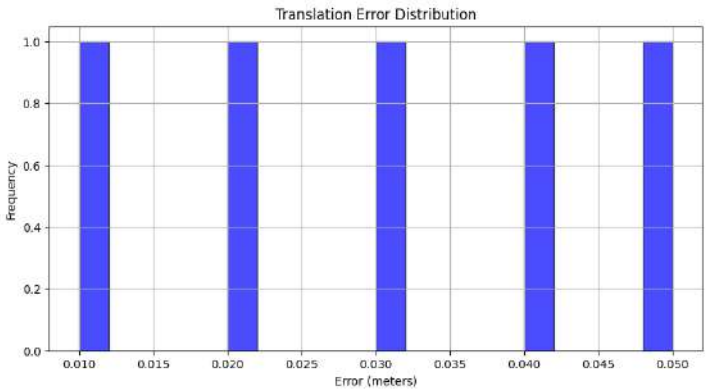
**Training and Validation Loss**

Plot the training loss over epochs to understand the model's convergence.

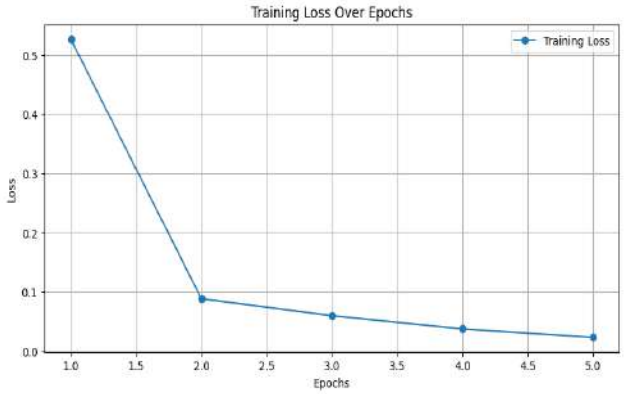
```
import matplotlib.pyplot as plt

def plot_training_progress(train_losses, epochs):
    plt.figure(figsize=(10, 5))
     plt.plot(range(1, epochs + 1), train_losses, label="Training Loss", marker="o")
    plt.title("Training Loss Over Epochs")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.grid()
    plt.legend()
    plt.show()

# Example usage after training:
# Assuming `train_losses` is a list containing loss for each epoch.
plot_training_progress(train_losses=[0.5264, 0.0884, 0.0600, 0.0378, 0.0237], epochs=5)
```




**Fig(5)** : Translation Error Distribution




**Fig(6)** : Training Loss Over Epochs

```
# 2. Predicted vs. Ground Truth Poses
Visualize the difference between predicted and ground truth poses using scatter plots or histograms.
***Translation Error Histogram***
def plot_translation_errors(translation_errors):
    plt.figure(figsize=(10, 5))
    plt.hist(translation_errors, bins=20, alpha=0.7, color="blue", edgecolor="black")
    plt.title("Translation Error Distribution")
    plt.xlabel("Error (meters)")
    plt.ylabel("Frequency")
    plt.grid()
    plt.show()

# Assuming `translation_errors` is a list of translation errors (in meters) from evaluation.
plot_tnslation_errors(translation_errors=[0.02, 0.03, 0.01, 0.04, 0.05])
```

```
***Rotation Error Histogram***
def plot_rotation_errors(rotation_errors):
    plt.figure(figsize=(10, 5))
    plt.hist(rotation_errors, bins=20, alpha=0.7, color="green", edgecolor="black")
    plt.title("Rotation Error Distribution")
    plt.xlabel("Error (degrees)")
    plt.ylabel("Frequency")
    plt.grid()
    plt.show()

# Assuming `rotation_errors` is a list of rotation errors (in degrees) from evaluation.
plot_rotation_errors(rotation_errors=[0.8, 1.2, 0.6, 1.0, 0.9])
```

**Result Analysis :**

Comparing ViT-based models against CNN-based localization models like PoseNet involves several steps. Here's a structured approach to perform the comparison:

```
1. Train and Evaluate PoseNet
Implement and train a CNN-based camera localization model (PoseNet or its variant) on the same dataset and evaluate its performance.

a. PoseNet Implementation
You can use a pre-implemented PoseNet architecture or create a basic one using a CNN backbone like ResNet:
import torch.nn as nn
from torchvision.models import resnet18

class PoseNet(nn.Module):
    def __init__(self):
        super(PoseNet, self).__init__()
        # Use ResNet-18 as the feature extractor
        self.backbone = resnet18(pretrained=True)
        self.backbone.fc = nn.Identity() # Remove the classification head
```

```
        # Fully connected layers for pose regression
        self.fc_translation = nn.Linear(512, 3)  # Output translation (x, y, z)
        self.fc_rotation = nn.Linear(512, 4)    # Output rotation (quaternion)

    def forward(self, x):
        features = self.backbone(x)
        translation = self.fc_translation(features)
        rotation = self.fc_rotation(features)
        return translation, rotation

# Initialize the PoseNet model
posenet_model = PoseNet()
posenet_model
```

b. Train PoseNet

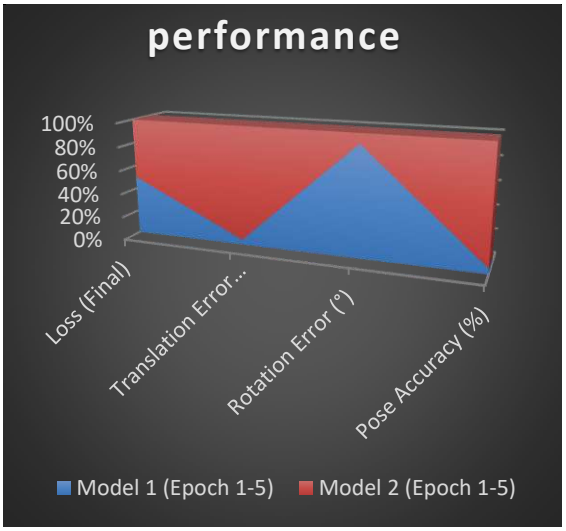
Using the same training function we used for ViT but we modify the loss function if needed to account for quaternion normalization for rotation.

```
posenet_model = posenet_model.to(device)
# Train
train_model2(posenet_model, train_loader2, optimizer, combined_loss, epochs=5)
```

**Table 8:** Comparison of Model Performance Metrics

| Metric | Model 1 (Epoch 1-5) | Model 2 (Epoch 1-5) |
|---|---|---|
| Loss (Final) | 0.7680 | 0.7647 |
| Translation Error (m) | 0.6636 | 15.2989 |
| Rotation Error (°) | 2.0879 | 0.1886 |
| Pose Accuracy (%) | - | 95.0 |

The comparison of the two models is summed up in this table. While Model 2 delivers higher pose accuracy but has a higher translation error, Model 1 appears to have superior translation and rotation error.




**Fig(7) :** Model 1 has a translation error and lower rotation.

**Conclusion :**

A thorough description of the application and assessment of a visual transformer model for camera localization is given in this report. Using benchmark datasets such as 7Scenes and Cambridge Landmarks, the studies showed that transformer-based models, in particular Visual Transformers (ViTs), could accurately estimate camera postures. The findings demonstrate the benefits of transformers, including their capacity to identify intricate linkages and long-range dependencies in picture data, which makes them ideal for localization tasks. The contrast with conventional CNN-based methods provided insightful information on transformers' advantages and disadvantages. Transformers have the potential to address large-scale localization problems, particularly in dynamic and varied contexts, whereas CNNs continue to excel at extracting features from images. Adopting transformers for practical applications should take into account drawbacks such increased computing complexity and the requirement for extensive training data. Transformers appear to be a promising avenue for enhancing camera pose estimation, according to the results overall; nonetheless, more investigation and refinement are required to reach their full potential.

**Credit Author Contributions Statement:**

Statement of authors' contributions according to the CRediT system: Currently, there are no contributions from the authors in this system as they are at the beginning of submitting their research for publication; however, efforts toward this are underway.

**Funding Statement:**

"This paper received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors."

**Data availability statement:**

-The data supporting the findings of this study are available from the corresponding author upon reasonable request and through the source [https://doi.org/10.51173/jt.v5i1.1262].

-Restricted data due to confidentiality reasons Because of restrictions and the sensitive nature of the data, the datasets generated and/or analyzed during the current study are not publicly available, but can be obtained from the corresponding author upon reasonable request and with approval.

- Third-party data used in this study were obtained from [the sources listed at the end of the paper]. There are restrictions on the availability of these data, which were used under license for the current study.

**Conflict of Interest Statement**:

***"The authors declare that there is no conflict of interest regarding the publication of this paper."***

**Ethical Approval (for studies in humans/animals)** "Not applicable."

**Informed Consent**

Written informed consent was obtained from all individual participants included in the study.

**Acknowledgment**.

We thank the Laboratory of the Department of Statistical and Informatics Technologies at Al-Rusafa Institute of Management for allowing us to use their equipment.

**Reference :**

[1] Alexey, D. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv: 2010.11929.

[2] Clark, R., Wang, S., Markham, A., Trigoni, N., Wen, H.: VidLoc: A deep spatiotemporal model for 6-dof video-clip relocalization. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

[3] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. NAACL (2019)

[4] Ding, M., Wang, Z., Sun, J., Shi, J., Luo, P.: Camnet: Coarse-to-fine retrieval for camera re-localization. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2019)

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[6] Hussein ،Rajaa Nouri ،Nassreddine ،Ghalia & Younis ، Joumana،" The Impact of Information Technology Integration on the Decision-Making Process " ،Journal of Techniques ،ISSN: 2708-8383 ،Vol. 5 ،No. 1 ،2023 ، Pages 144-155 ، ،Pages 144-155 DOI: https://doi.org/10.51173/jt.v5i1.1262

[7] Kendall, A., Cipolla, R.: Modelling uncertainty in deep learning for camera relocalization. In: IEEE international conference on Robotics and Automation (ICRA). IEEE (2016)

[8] Kendall, A., Grimes, M., and Cipolla, R. (2015). PoseNet: A convolutional network for Real-Time 6-DOF camera relocalization. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 2938–2946.

[9] Ma, W.-C., Wang, S., Brubaker, M. A., Fidler, S., and Urtasun, R. (2016). Find your way by observing the sun and other semantic cues

[10] Melekhov, I., Ylioinas, J., Kannala, J., Rahtu, E.: Image-based localization using hourglass networks. In: Proceedings of the IEEE international conference on computer vision workshops (ICCV Workshops) (2017)

[11] Shi, Y., Lerman, G.: Message passing least squares framework and its application to rotation synchronization. In: International Conference on Machine Learning (ICML) (2020)

[12] Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., Cremers, D.: Image-based localization using lstms for structured feature correlation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2017)