ROS Bag Processing Guide

Processing Image Pairs with Trajectory Data

Overview

This guide provides step-by-step instructions for processing ROS bag files containing camera images and trajectory data. The project extracts image pairs from the bag file, processes trajectory data from TUM format files, and matches them together to create a dataset for visual odometry research.

Project Components

1. **Docker Environment**: A containerized environment with ROS Noetic and all dependencies
2. **Scripts**:
   - `traj_reader.py`: Processes TUM trajectory files with quaternion to Euler conversion
   - `image_pair.py`: Extracts consecutive image pairs from ROS bags
   - `pairs_with_trajectory.py`: Matches image pairs with ground truth trajectory data
   - `visualization.py`: Visualizes image pairs and trajectory data

File Structure

```
SECOND TRIAL/
├── scripts/              # Python scripts
│   ├── traj_reader.py
│   ├── image_pair.py
│   ├── pairs_with_trajectory.py
│   └── visualization.py
├── data/                 # Data files (ROS bags and trajectory)
│   ├── ariel_2023-12-21-14-26-32_2.bag
│   └── qualisys_ariel_odom_traj_8_id6.tum
├── output/               # Generated output
│   ├── image_pairs/      # Extracted image pairs
│   ├── visualizations/   # Visualizations of image pairs
│   └── plots/            # Trajectory plots
├── Dockerfile            # Docker configuration
└── run.sh                # Complete pipeline script
```

Step-by-Step Instructions

Step 1: Set Up the Project

1. Create the project directory structure:

```bash
mkdir -p "SECOND TRIAL/scripts" "SECOND TRIAL/data"
cd "SECOND TRIAL"
```

2. Copy the provided scripts to the `scripts/` directory:
   - `traj_reader.py`: Processes TUM trajectory files
   - `image_pair.py`: Extracts image pairs from ROS bags
   - `pairs_with_trajectory.py`: Matches images with trajectory data
   - `visualization.py`: Visualizes the matched data
   - `run.sh`: Complete pipeline script
3. Copy your data files to the `data/` directory:
   - ROS bag file (`ariel_2023-12-21-14-26-32_2.bag`)
   - TUM trajectory file (`qualisys_ariel_odom_traj_8_id6.tum`)
4. Create the Dockerfile with the provided content.

Step 2: Build the Docker Image

Open PowerShell or Command Prompt in the project directory and run:

```powershell
# Windows PowerShell
docker build -t ros-trajectory-analysis .
```

This builds a Docker image with all necessary dependencies:

- ROS Noetic
- OpenCV
- Python dependencies (numpy, pandas, pyquaternion, etc.)

Step 3: Run the Docker Container

In Windows PowerShell:

```powershell
# Create output directory if it doesn't exist
mkdir -p output

# Run the container with volume mounting
docker run -it --rm `
  -v "${PWD}\data:/app/data" `
  -v "${PWD}\output:/app/output" `
  ros-trajectory-analysis
```

Step 4: Process Trajectory Data

Inside the Docker container:

```bash
python3 /app/scripts/traj_reader.py \
  --input /app/data/qualisys_ariel_odom_traj_8_id6.tum \
  --output /app/output/processed_trajectory.csv \
  --plot /app/output/plots/trajectory_plot.png
```

This command:

- Reads the TUM format trajectory file
- Converts quaternions to Euler angles (yaw, pitch, roll)
- Calculates deltas between consecutive poses
- Saves processed data as CSV
- Generates a 2D plot of the trajectory

Step 5: Extract Image Pairs

Based on the bag file inspection, we know the camera topics are `/alphasense_driver_ros/cam0` through `/alphasense_driver_ros/cam4`. Use the appropriate topic:

```bash
python3 /app/scripts/image_pair.py \
  --bag /app/data/ariel_2023-12-21-14-26-32_2.bag \
  --output /app/output/image_pairs_cam0 \
  --topic "/alphasense_driver_ros/cam0" \
  --time_diff 0.1
```

This command:

- Reads images from the specified ROS topic
- Extracts consecutive image pairs (prev/current)
- Saves image pairs with timestamps
- Creates a diff image between consecutive frames
- The `time_diff` parameter controls how far apart frames should be (0.1 = 100ms)

To process multiple cameras, repeat for each camera topic:

```bash
python3 /app/scripts/image_pair.py \
  --bag /app/data/ariel_2023-12-21-14-26-32_2.bag \
  --output /app/output/image_pairs_cam1 \
  --topic "/alphasense_driver_ros/cam1" \
  --time_diff 0.1
```

Step 6: Match Image Pairs with Trajectory Data

```bash
python3 /app/scripts/pairs_with_trajectory.py \
  --pairs /app/output/image_pairs_cam0 \
  --trajectory /app/output/processed_trajectory.csv \
  --output /app/output/image_pairs_with_gt_cam0.csv
```

This command:

- Reads timestamps from each image pair
- Finds the closest trajectory point in time
- Associates motion data (delta_x, delta_y, delta_z, delta_yaw) with each image pair
- Saves the matched data as CSV

Step 7: Visualize Results

bash

```
python3 /app/scripts/visualization.py \
  --input /app/output/image_pairs_with_gt_cam0.csv \
  --output_dir /app/output/visualizations_cam0 \
  --num_pairs 5 \
  --plot_trajectory \
  --trajectory_plot /app/output/plots/deltas_plot_cam0.png
```

This command:

- Visualizes image pairs with their corresponding motion data
- Creates side-by-side views of consecutive frames
- Shows delta values (changes in x, y, z, yaw)
- Optionally plots trajectory data over time
- Saves visualizations to the specified directory

Step 8: Run the Complete Pipeline

For convenience, you can run all steps together using the provided run.sh script:

bash

```
# Inside the Docker container
ROS_TOPIC="/alphasense_driver_ros/cam0" bash /app/scripts/run.sh
```

The script handles all the steps automatically:

- Processing the trajectory data
- Extracting image pairs
- Matching with trajectory data
- Generating visualizations

Understanding the Output

After running the pipeline, you'll have:

1. **Processed Trajectory Data** (`output/processed_trajectory.csv`):
   - Timestamp, x, y, z positions
   - Quaternion values (qx, qy, qz, qw)
   - Euler angles (yaw, pitch, roll)

- Deltas between consecutive poses

2. **Image Pairs** (`output/image_pairs_cam0/`):
   - Directories for each pair (pair_0000, pair_0001, etc.)
   - Previous and current frame images
   - Diff image showing changes
   - Timestamp file with frame timestamps
3. **Matched Data** (`output/image_pairs_with_gt_cam0.csv`):
   - Image pair ID and path
   - Timestamps for both frames
   - Closest trajectory timestamp
   - Motion data (delta_x, delta_y, delta_z, delta_yaw)
   - Orientation (yaw, pitch, roll)
4. **Visualizations** (`output/visualizations_cam0/`):
   - Side-by-side views of image pairs
   - Overlay of motion data
   - Trajectory plots

Troubleshooting

No image pairs are extracted

- Verify the ROS topic is correct using the bag inspection
- Check if the bag file contains image messages
- Try different time thresholds with `--time_diff`

Trajectory matching issues

- Ensure timestamps in the TUM file and ROS bag are compatible
- Adjust the time threshold in the matching script
- Check the TUM file format

Docker mounting issues

- Use absolute paths for volume mounting
- Ensure proper permissions on host directories
- For Windows, use the correct path format in PowerShell

Visualization problems

- For interactive visualization, set up X11 forwarding
- Save visualizations to files and view outside the container

Advanced Usage

Processing Multiple Cameras

Create a loop to process all cameras:

bash

```bash
for CAM_NUM in {0..4}; do
  TOPIC="/alphasense_driver_ros/cam${CAM_NUM}"
  OUTPUT_DIR="/app/output/image_pairs_cam${CAM_NUM}"

  python3 /app/scripts/image_pair.py \
    --bag /app/data/ariel_2023-12-21-14-26-32_2.bag \
    --output ${OUTPUT_DIR} \
    --topic ${TOPIC}

  python3 /app/scripts/pairs_with_trajectory.py \
    --pairs ${OUTPUT_DIR} \
    --trajectory /app/output/processed_trajectory.csv \
    --output /app/output/image_pairs_with_gt_cam${CAM_NUM}.csv

  python3 /app/scripts/visualization.py \
    --input /app/output/image_pairs_with_gt_cam${CAM_NUM}.csv \
    --output_dir /app/output/visualizations_cam${CAM_NUM} \
    --num_pairs 2
done
```

Customizing Time Intervals

To extract frames at specific intervals:

bash

```bash
python3 /app/scripts/image_pair.py \
  --bag /app/data/ariel_2023-12-21-14-26-32_2.bag \
  --output /app/output/image_pairs_cam0_fast \
  --topic "/alphasense_driver_ros/cam0" \
  --time_diff 0.05  # 50ms between frames (faster)
```

Creating a Dataset for Machine Learning

After generating the matched data, you can create a structured dataset:

```bash
# Structure: image_pairs, deltas.csv
mkdir -p /app/output/dataset/{images,labels}

# Extract data into ML-friendly format (would require a custom script)
python3 /app/scripts/create_ml_dataset.py \
  --input /app/output/image_pairs_with_gt_cam0.csv \
  --output_dir /app/output/dataset
```

Conclusion

This pipeline provides a complete workflow for:

1. Processing trajectory data from TUM files
2. Extracting image pairs from ROS bags
3. Matching images with ground truth motion data
4. Visualizing the results

The resulting dataset can be used for visual odometry research, machine learning training, or other robotics applications that require matched visual and motion data.