CVGlobal

Leonardo Ruso
leonardo.russo@inria.fr
Diego Marcos
diego.marcos@inria.fr

INRIA
Evergreen Team
Montpellier, France

Abstract

This document demonstrates the format requirements for papers submitted to the British Machine Vision Conference. The format is designed for easy on-screen reading, and to print well at one or two pages per sheet. Additional features include: pop-up annotations for citations [1, 2]; a margin ruler for reviewing; and a greatly simplified way of entering multiple authors and institutions.

All authors are encouraged to read this document, even if you have written many papers before. As well as a description of the format, the document contains many instructions relating to formatting problems and errors that are common even in the work of authors who *have* written many papers before.

1 Introduction

...

2 Dataset Generation Method

In this section, we present our methodology for constructing CVGlobal, a large-scale multi-modal dataset that pairs satellite and street-view imagery across diverse global regions. Our approach systematically samples locations from five continents while ensuring geographical diversity and balanced representation between urban and rural environments.

2.1 Dataset Design and Sampling Strategy

Our dataset construction methodology is guided by three key principles: *geographical diversity*, *balanced representation*, and *multi-modal consistency*. We define sampling regions across five major continents (North America, Europe, Asia, South America, and Africa), with each continent contributing equally to the final dataset to prevent geographical bias.

For each continent, we establish two distinct sampling regions:

- **Urban regions**: Areas with high population density and significant urban infrastructure
- **Rural regions**: Areas with low population density and predominantly natural or agricultural landscapes

Table 1: Geographical sampling regions defined for each continent and environment type.

Continent	Type	Location	Lat Range	Lon Range
North America	Urban	New York City	40.71°–40.81°N	74.01°–73.91°W
	Rural	California Farmland	36.78°–36.88°N	119.42°–119.32°W
Europe	Urban	Paris	48.86°–48.96°N	2.35°–2.45°E
	Rural	French Countryside	46.23°–46.33°N	2.21°–2.31°E
Asia	Urban	Tokyo	35.69°–35.79°N	139.69°–139.79°E
	Rural	Rural India (Agra)	27.18°–27.28°N	78.04°–78.14°E
South America	Urban	São Paulo	23.55°–23.45°S	46.63°–46.53°W
	Rural	Brazilian Rainforest	14.24°–14.13°S	51.93°–51.83°W
Africa	Urban	Nairobi	1.29°–1.19°S	36.82°–36.92°E
	Rural	Kenyan Savanna	2.15°–2.05°S	37.31°–37.41°E

The sampling regions are carefully selected to represent diverse climatic, cultural, and developmental contexts within each continent. Table 1 details the specific geographical boundaries for each region.

2.2 Urban-Rural Classification

To ensure accurate labeling of locations as urban or rural, we employ the Global Urban Areas dataset [**?**], which provides comprehensive polygon boundaries for urban areas worldwide. For each randomly generated coordinate, we perform a spatial intersection test to determine its classification:

$$\text{Urban}(p) = \begin{cases} 1 & \text{if } p \in \bigcup_i U_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where p represents a coordinate point and U_i denotes the i-th urban area polygon from the Global Urban Areas dataset. This automated classification ensures consistent and objective urban-rural labeling across all geographical regions.

2.3 Multi-Modal Data Acquisition

Our data acquisition pipeline consists of three main components: *coordinate generation*, *outdoor location validation*, and *multi-modal image retrieval*.

2.3.1 Coordinate Generation and Validation

For each sampling region, we generate random coordinates within the specified geographical boundaries using uniform sampling. Each coordinate undergoes a validation process to ensure data quality:

1. **Urban-Rural Consistency**: Verify that the generated coordinate matches the intended environment type (urban/rural) using the spatial intersection described above.

Algorithm 1 Outdoor Location Detection

Require: Street View metadata M, Google Maps client G
Ensure: Boolean indicating outdoor location

```
if M.status ≠ OK then
    return False
end if
if M.place_id is undefined then
    return True                          ▷ Assume outdoor for street-level locations
end if
details ← G.place(M.place_id)
types ← details.result.types
indoor_types ← {shopping_mall, store, restaurant, hospital, ...}
if types ∩ indoor_types ≠ ∅ then
    return False
else
    return True
end if
```

2. **Street View Availability**: Query the Google Street View Metadata API to confirm image availability at the location.

3. **Outdoor Location Filtering**: Apply our outdoor detection algorithm to exclude indoor environments.

2.3.2 Outdoor Detection Algorithm

To ensure our dataset captures genuine outdoor environments, we implement a robust filtering mechanism that leverages Google Places API data. Our algorithm evaluates each location using the following criteria:

This approach is more nuanced than simple keyword filtering, as it distinguishes between genuinely indoor locations (e.g., shopping malls, restaurants) and outdoor points of interest (e.g., parks, monuments) that may also carry establishment tags.

2.3.3 Image Acquisition and Processing

For each validated coordinate, we acquire two types of imagery:

Satellite Imagery We retrieve high-resolution satellite images using the Google Static Maps API with the following specifications:

- Resolution: 640x640 pixels
- Zoom level: 18 (approximately 1.19 meters/pixel)
- Map type: Satellite view
- Format: JPEG

Street View Imagery We collect street-view images from four cardinal directions (0°, 90°, 180°, 270°) to provide comprehensive ground-level perspective. Each image has:

- Resolution: 640x640 pixels
- Field of view: Default Google Street View settings
- Format: JPEG

The four directional images are horizontally concatenated to create a panoramic representation, resulting in a 2560x640 pixel stitched image that captures the complete ground-level environment.

2.4 Quality Assurance and Error Handling

Our data acquisition pipeline implements robust error handling and quality assurance mechanisms:

2.4.1 Network Resilience

We employ an exponential backoff retry strategy for API requests, with up to 3 retry attempts for failed connections. This approach handles temporary network issues and API rate limiting gracefully.

2.4.2 Coordinate Correction

The Google Street View API may return imagery from coordinates slightly different from the requested location due to road network constraints. We handle this by:

1. Recording both original and corrected coordinates
2. Using corrected coordinates for file naming and deduplication
3. Ensuring consistent satellite-street view pairing

2.4.3 Resume Capability

Our pipeline supports interruption and resumption, checking for existing complete image sets before processing each location. A complete set consists of:

- One satellite image
- Four directional street view images (0°, 90°, 180°, 270°)
- One stitched panoramic image

2.5 Dataset Statistics and Validation

Throughout the data collection process, we maintain comprehensive statistics including:

- Success and failure rates per continent and environment type
- API call counts and timing information
- Error categorization (metadata failures, download failures, indoor rejections)
- Coordinate generation efficiency metrics

These statistics are automatically compiled into detailed reports (both human-readable and machine-readable formats) that facilitate dataset validation and quality assessment.

The resulting CVGlobal dataset provides a balanced, geographically diverse collection of paired satellite and street-view imagery suitable for training and evaluating computer vision models across varied global contexts.

3 CroDino Method

We propose CroDINO, a novel approach for cross-view orientation estimation that leverages the robust feature representations of DINOv2 with orientation-aware token aggregation strategies. Our method addresses the fundamental challenge of aligning ground-level panoramic images with aerial satellite views by exploiting both spatial structure and depth information.

3.1 Problem Formulation

Given a ground-level panoramic image I_g and an aerial satellite image I_a of the same geographic location, our goal is to estimate the relative orientation θ between the two views. The ground image is extracted from a 360° panorama using a field-of-view (FOV) window defined by parameters (f_x, f_y, ψ, ϕ), where f_x and f_y represent the horizontal and vertical FOV angles, ψ is the yaw (rotation around the vertical axis), and ϕ is the pitch (elevation angle).

3.2 Architecture Overview

3.2.1 Dual-Stream Feature Extraction

CroDINO builds upon the DINOv2 Vision Transformer architecture, which provides rich, self-supervised feature representations. We modify the standard DINOv2 model to process both ground and aerial images simultaneously while maintaining separate positional embeddings and class tokens for each modality.

The model consists of:

- **Shared Backbone**: We utilize the pre-trained DINOv2-ViT-B/14 as our feature extractor, freezing its parameters to preserve the learned representations.
- **Dual Positional Embeddings**: Separate positional embeddings $\mathbf{E}^g_{pos}$ and $\mathbf{E}^a_{pos}$ for ground and aerial images to account for their different spatial characteristics.
- **Cross-Modal Attention**: A final single-head attention layer that enables interaction between ground and aerial features.

3.2.2 Token Processing Pipeline

For each input image pair, the model generates patch embeddings of size 14×14 pixels, resulting in a 16×16 grid of 768-dimensional feature vectors. The forward pass can be formulated as:

$$\mathbf{F}_g = \text{DINOv2}(I_g; \mathbf{E}^g_{pos}) \tag{2}$$

$$\mathbf{F}_a = \text{DINOv2}(I_a; \mathbf{E}^a_{pos}) \tag{3}$$

$$\mathbf{F}_{combined} = \text{Attention}([\mathbf{F}_g; \mathbf{F}_a]) \tag{4}$$

where $\mathbf{F}_g, \mathbf{F}_a \in \mathbb{R}^{16\times16\times768}$ are the ground and aerial feature matrices, respectively.

3.3 Orientation-Aware Token Aggregation

3.3.1 Sky Filtering and Depth Estimation

To improve orientation estimation, we incorporate semantic and geometric priors:

Sky Segmentation: We employ a lightweight CNN-based sky filter to identify and mask sky regions in ground images. The sky mask M_{sky} is computed at the patch level using majority voting within each 16×16 grid cell.

Depth Estimation: We utilize the Depth-Anything model to generate depth maps D for ground images. The depth information is downsampled to match the patch grid, providing depth values $d_{i,j}$ for each spatial location (i, j).

3.3.2 Multi-Layer Token Aggregation

We introduce a novel aggregation strategy that separates tokens into three depth layers: foreground, middleground, and background. This approach captures the multi-scale nature of visual features in cross-view matching.

Vertical Column Analysis: For each vertical column j in the ground image feature grid, we compute depth-weighted averages:

$$\mathbf{t}^{fore}_j = \frac{\sum_i w^{fore}_i \cdot \mathbf{f}^g_{i,j} \cdot M_{sky}(i,j)}{\sum_i w^{fore}_i \cdot M_{sky}(i,j)} \tag{5}$$

$$\mathbf{t}^{mid}_j = \frac{\sum_i w^{mid}_i \cdot \mathbf{f}^g_{i,j} \cdot M_{sky}(i,j)}{\sum_i w^{mid}_i \cdot M_{sky}(i,j)} \tag{6}$$

$$\mathbf{t}^{back}_j = \frac{\sum_i w^{back}_i \cdot \mathbf{f}^g_{i,j} \cdot M_{sky}(i,j)}{\sum_i w^{back}_i \cdot M_{sky}(i,j)} \tag{7}$$

where the depth-dependent weights are defined as:

$$w^{fore}_i = d_{i,j} \tag{8}$$

$$w^{mid}_i = \begin{cases} \frac{d_{i,j}}{\tau} & \text{if } d_{i,j} \leq 0.5 \\ \frac{1-d_{i,j}}{d_{i,j}} & \text{otherwise} \end{cases} \tag{9}$$

$$w^{back}_i = 1 - d_{i,j} \tag{10}$$

with threshold $\tau = 0.5$.

Radial Direction Analysis: For aerial images, we extract features along radial directions from the center:

$$\mathbf{r}_{\beta}^{fore} = \frac{\sum_r w_r^{fore} \cdot \mathbf{f}_{\beta,r}^{a}}{\sum_r w_r^{fore}} \tag{11}$$

$$\mathbf{r}_{\beta}^{mid} = \frac{\sum_r w_r^{mid} \cdot \mathbf{f}_{\beta,r}^{a}}{\sum_r w_r^{mid}} \tag{12}$$

$$\mathbf{r}_{\beta}^{back} = \frac{\sum_r w_r^{back} \cdot \mathbf{f}_{\beta,r}^{a}}{\sum_r w_r^{back}} \tag{13}$$

where β represents the angle direction, r is the radial distance, and the weights follow a linear progression: $w_r^{fore} = 1 - r/R$, w_r^{mid} follows a triangular pattern, and $w_r^{back} = r/R$.

3.4 Orientation Estimation

3.4.1 Cross-Modal Alignment

We estimate orientation by finding the angular offset that minimizes the cosine distance between corresponding vertical and radial feature aggregations. For each candidate orientation θ, we compute:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} \left\| 1 - \begin{bmatrix} \mathbf{t}_{N-1-i}^{fore} \\ \mathbf{t}_{N-1-i}^{mid} \\ \mathbf{t}_{N-1-i}^{back} \end{bmatrix}^T \begin{bmatrix} \mathbf{r}_{\phi(\theta,i)}^{fore} \\ \mathbf{r}_{\phi(\theta,i)}^{mid} \\ \mathbf{r}_{\phi(\theta,i)}^{back} \end{bmatrix} \right\| \tag{14}$$

where $\phi(\theta, i) = (\lfloor \theta/\Delta\theta \rfloor + i - N/2) \bmod |\mathcal{R}|$ maps vertical columns to radial directions, and $\Delta\theta$ is the angular step size.

3.4.2 Confidence Estimation

To assess the reliability of orientation estimates, we compute a confidence score based on the Z-score of the minimum distance:

$$\text{confidence} = \frac{\mu(\mathcal{L}) - \min(\mathcal{L})}{\sigma(\mathcal{L})} \tag{15}$$

where $\mu(\mathcal{L})$ and $\sigma(\mathcal{L})$ are the mean and standard deviation of the loss values across all candidate orientations.

3.5 Training Strategy

Our approach operates in a largely unsupervised manner, leveraging the pre-trained DINOv2 features without requiring orientation labels during training. The model learns to align cross-view features through the geometric constraints imposed by the aggregation strategy and the cosine similarity objective.

3.5.1 Data Preprocessing

We extract random FOV windows from panoramic images with parameters:

- Horizontal FOV: 90°
- Vertical FOV: 180°
- Random yaw: $\psi \sim \text{Uniform}(0^\circ, 360^\circ)$
- Fixed pitch: $\phi = 90^\circ$

Aerial images undergo center cropping and optional polar transformation to align with the ground view geometry.

3.6 Implementation Details

The complete pipeline processes image pairs through the following stages:

1. Feature extraction using frozen DINOv2-ViT-B/14
2. Sky segmentation using guided filter refinement
3. Depth estimation with Depth-Anything model
4. Multi-layer token aggregation with depth weighting
5. Cross-modal orientation search with cosine similarity

All models are implemented in PyTorch and can operate on both CPU and GPU. The orientation search space is discretized with angular steps of $\Delta\theta = 90/16 = 5.625$ for a 16×16 patch grid.

References

[1] Authors. The frobnicatable foo filter, 2006. ECCV06 submission ID 324. Supplied as additional material `eccv06.pdf`.

[2] N. David Mermin. What's wrong with these equations? *Physics Today*, October 1989. http://www.cvpr.org/doc/mermin.pdf.