GT-Loc: Unifying When and Where in Images Through a Joint Embedding Space

Supplementary Material

7. Model inference

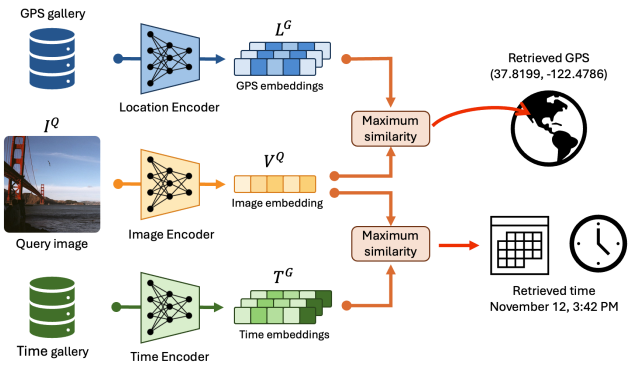

Figure 6. GT-Loc during inference.

Our framework consists of a model that can predict both the location and capture-time of an image at the same time using a retrieval approach. Given a query image I^Q, a gallery of GPS coordinates and a gallery of timestamps, GT-Loc maps the three modalities into a shared feature space using an image, location and time encoder. The query image embedding V^Q is compared against a set of location embeddings L^G and time embeddings T^G. The GPS and timestamp with the highest cosine similarity are selected as the predictions of our model. Figure 6 represents the overview of our approach.

8. Implementation details

Following GeoCLIP, the backbone of the image encoder is a pretrained ViT-L/14 from CLIP and the MLP consists of two fully connected layers with the ReLU activation function and dimensions 768 and 512 respectively. We use the same architecture for the time and location encoders as GeoCLIP. Both employ three RFF positional encoding layers, mapping the 2-dimensional GPS to a vector with 512 dimensions. The standard deviation values used to sample the RFF are $\sigma_i \in \{2^0, 2^4, 2^8\}$. The MLPs from the time and location encoder have three hidden layers with 1024 dimensions and a projection layer to map the final embeddings into a feature space of 512 dimensions. In the location encoder, we use a dynamic queue that stores the last 4096 seen locations, but we don't use it for time. The GPS coordinates and times are augmented by adding Gaussian noise with standard deviation of 150 meters for the in-batch GPS, 1500 meters for the GPS queue, 0.15 months and 0.15 hours for time. We perform two augmentations for each image in the training set using random resized crops of size 224, random horizontal flipping and image normalization.

9. Training protocol

GT-Loc is trained for 20 epochs using a cosine decay scheduler, with learning rate values ranging from $\alpha_{max} = 3 \times 10^{-5}$ to $\alpha_{min} = 3 \times 10^{-7}$. We use Adam optimizer with coefficients $\beta_1 = 0.9$, $\beta_2 = 0.999$ and ℓ_2 penalty of 1×10^{-6}. For the contrastive losses, we use two learnable temperature parameters that are optimized during training. The batch size B is set to 512 for all experiments, and the models are trained on a machine with 12 CPU cores and a NVIDIA RTX A6000 GPU.

10. Additional qualitative results

We show additional qualitative results of our method in figures 7 and 8. We include a failure case, on the last row of figure 8, where the time error is high because of the presence of fog in the image.

11. Time-of-capture prediction histograms

The time prediction histograms, shown in figures 4, 5, 7, and 8, are computed using the following equation:

$$C_i = \sum_{j=1}^{N_G} \mathbb{1}_{[j \in \mathcal{B}_i]} \cdot I^Q \cdot T_j^G, \tag{12}$$

where $\mathcal{B}_i$ is the set of gallery embeddings that correspond to the ith bin, C_i is the bin count, N_G is the gallery size, I^Q is the query image embedding, T_j^G is the jth time embedding from the gallery, and $\mathbb{1}$ is an indicator variable.

12. Dataset details

We apply two filters to remove samples from the CVT that don't provide meaningful temporal information. In particular, we remove all night-time and indoor images, since they often have inconsistent temporal cues. To remove night images, we estimate the sunrise and sunset times from the date, latitude and longitude using the General Solar Position algorithm, and remove all samples before sunrise or after sunset. Then, for indoor images, we leverage a CNN model pretrained on the Places365 Dataset [46]. In general, night images often have inconsistent artificial lighting, more noise or specialized cameras such as night vision. Indoor images also have artificial lighting and controlled temperature, making it difficult to estimate the time or date.

Regarding the levels of noise in the dataset, the SkyFinder subset consists of images with accurate time estimates, since

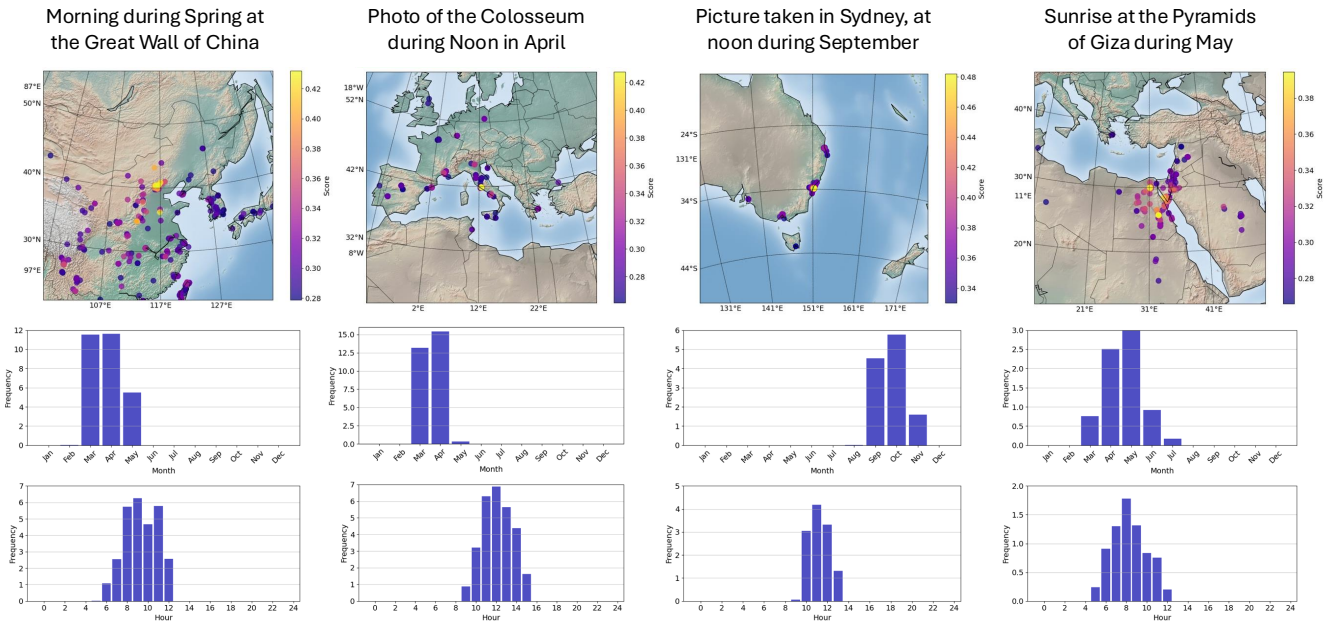

Figure 7. Additional qualitative examples of geo-localization and time-of-capture prediction using text queries. Top: prompt passed to CLIP's text encoder. Middle: spatial distribution of the predicted geo-locations with the highest cosine similarity. Bottom: histogram of the predicted months and hours with the highest cosine similarity.

they were collected from calibrated outdoor webcams. However, we observed that CVT has a moderate amount of noisy labels. Thus in order to train a model that can accurately predict the time, we need both datasets in the training set.

For evaluating the models, we employ a subset of unseen SkyFinder cameras, as well as two geo-localization datasets used by other state-of-the-art methods for evaluation: Im2GPS3k and GWS15k. Similar to GeoCLIP, we create a 100k GPS gallery to evaluate the model on Im2GPS3k, a 500k GPS gallery for GWS15k, and a 100k time gallery for the SkyFinder test set. The GPS galleries are created by sampling GPS coordinates from the MP-16 dataset, while the time gallery is created by sampling times from the combined CVT and SkyFinder training sets.

13. Additional ablations

13.1. Image backbones

To evaluate the impact of different image embeddings on time prediction performance, we conducted ablation studies using three backbones: DINOv2-L [21], OpenCLIP ViT-G [7], and OpenAI's original CLIP ViT-L [24]. For these experiments, we used the TimeLoc variant of our model, which incorporates only the image and time encoders. The results, summarized in Table 7, indicate that OpenAI's CLIP ViT-L achieves the lowest errors for both hours and months, as well as the highest Time Prediction Score (TPS).

Backbone	Param.	Month Error	Hour Error	TPS
DINOv2-L	0.3B	2.10	3.25	68.71
OpenCLIP-G	1.8B	1.57	2.94	74.65
CLIP-L	0.2B	1.52	2.84	75.49

Table 7. Comparison of time prediction performance using different frozen image backbones.

13.2. Time representation

Motivated by the cyclical nature of time, Mac Aodha et al. 17 used **circular decomposition** to wrap the temporal input to their geographical prior encoder, resulting in similar embeddings for dates that are close to the start and end of the year, such as December 31^{st} and January 1^{st}. To achieve this, for each dimension l of the temporal input $\vec{x}$, they perform the mapping $[\sin(\pi x^l), \cos(\pi x^l)]$, resulting in two numbers for each dimension.

Time2Vec [10] is a method for encoding time that captures both periodic and non-periodic patterns. It transforms scalar time values into a vector of size $k+1$. The first element models linear, non-periodic trends, while the remaining elements are defined by a periodic activation function (e.g., sine), capturing repeating temporal behaviors like daily or weekly cycles. The representation is defined as:

$$t2v(\tau)[i] = \begin{cases} \omega_i \tau + \phi_i & \text{if } i = 0, \\ F(\omega_i \tau + \phi_i) & \text{if } 1 \leq i \leq k, \end{cases}$$

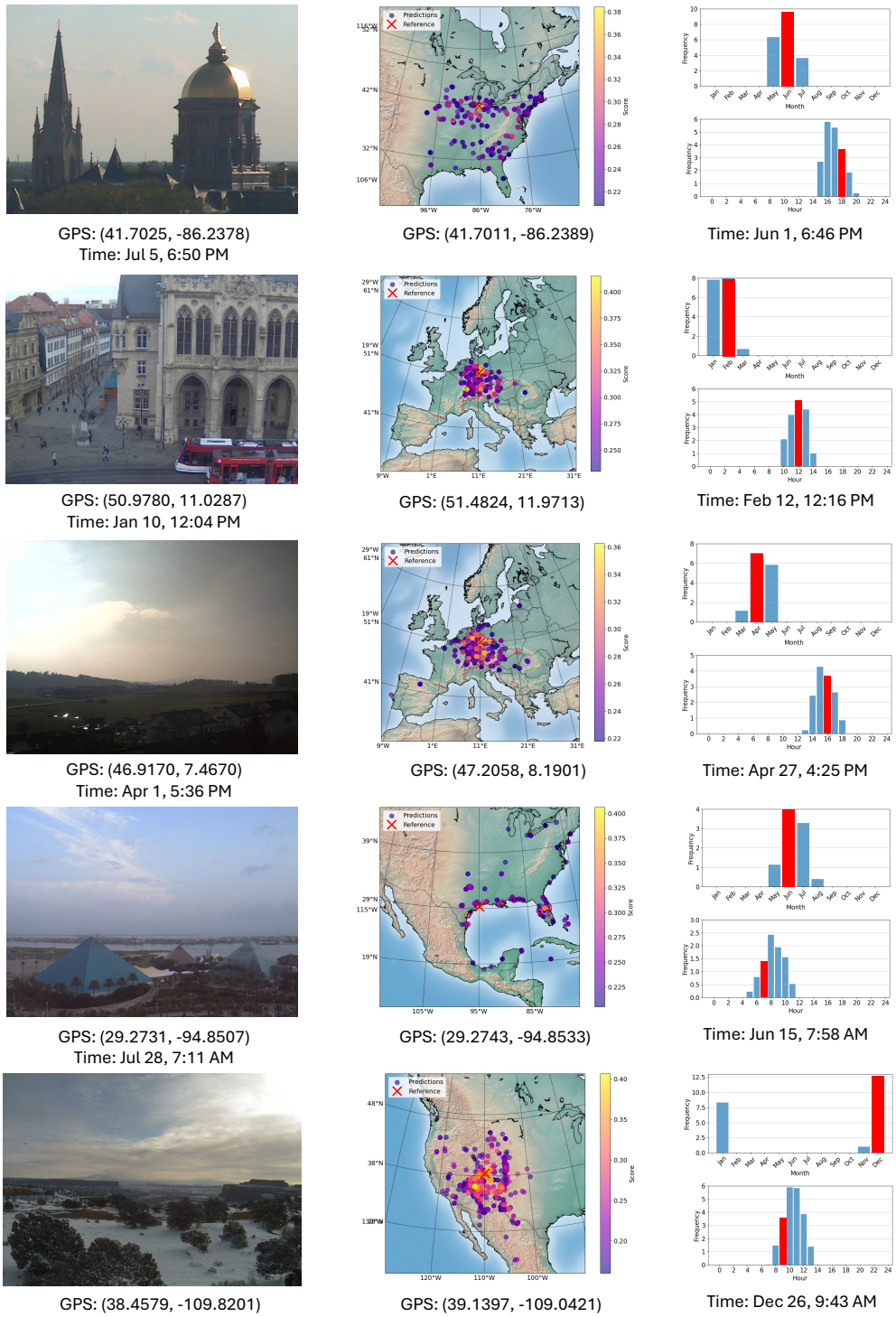

Figure 8. (a) Additional sample predictions for three cameras of the SkyFinder test set with the ground truth location and capture time. (b) Spatial distribution of the predicted GPS coordinates colored by the cosine similarity between the location and image embeddings. (c) Temporal distribution of the predicted month and hour, weighted by the cosine similarity between the time and image embeddings. The red bin contains the top-1 predicted time.

where F is a periodic activation function, typically $\sin$, and ω_i and ϕ_i are learnable parameters representing the frequency and phase shift, respectively.

Table 8. **Ablations** for time prediction performance using different time encoders.

Time encoder	Month Error	Hour Error	TPS
Circular decomp.	1.59	2.86	74.80
Time2Vec	1.56	2.62	75.99
RFF	1.40	2.72	77.00

13.3. Time-of-year scale

Time-of-year (ToY) can be represented at either a monthly or daily scale. In practice, the choice of time scale should not significantly affect the results, as the value is normalized before being passed to the time encoder, $\mathcal{T}(\cdot)$. However, two approaches are available. The first approach converts the integer month m_i and day d_i into a real-valued month, normalized over a 12-month period, as shown in Equations 3 and 4. The second approach represents ToY as the number of days elapsed since the start of the year, normalized over 365 days (assuming no leap years in our dataset). This representation is defined as:

$$\theta_i = \frac{1}{365}\left(d_i - 1 + \sum_{k=1}^{i} \mathcal{D}(m_k - 1)\right),$$

where $\mathcal{D}(m_k)$ is the number of days in month m_k, and it is assumed that $\mathcal{D}(0) = 0$. In Table 9, we empirically show that using a monthly scale for ToY representation results in slightly better performance. However, we attribute this improvement to statistical noise rather than the time representation method itself.

Table 9. **Ablations** for time prediction performance using monthly and daily scales.

ToY Scale	Month Error	Hour Error	TPS
daily	1.45	2.71	76.61
monthly	1.40	2.72	77.00

14. Compositional image retrieval details

To compare GT-Loc against a suitable baseline, we repurpose the model proposed by Zhai et al. [45] for compositional retrieval. In its original form, the model comprises an image encoder $C_I(I)$, a time encoder $C_T(t)$, and a location encoder $C_L(L)$. It concatenates the image and time embeddings to predict location via $P(l \mid C_I, C_T)$, and similarly concatenates the image and location embeddings to predict time via $P(t \mid C_I, C_L)$.

To adapt it for compositional image retrieval, we take a query time T^Q and location L^Q along with an image gallery I^G. First, we concatenate each image embedding $C_I(I^G)$ with $C_T(T^Q)$ or $C_L(L^Q)$ and feed these pairs to the respective classification heads. This yields probability distributions $P_i(l \mid C_I, C_T)$ and $P_i(t \mid C_I, C_L)$ for each image i in the gallery, indicating how well that image matches the queried location and time. Since we already know the desired location and time, we extract the corresponding probabilities from the distributions and average them to produce a final similarity score. Ranking by this score allows us to retrieve the top-k gallery images most likely to match (T^Q, L^Q). This adaptation of the Zhai et al. [45] model provides a direct, fair comparison to GT-Loc's performance on compositional retrieval tasks.

Figure 9 shows two qualitative results of our compositional retrieval model.

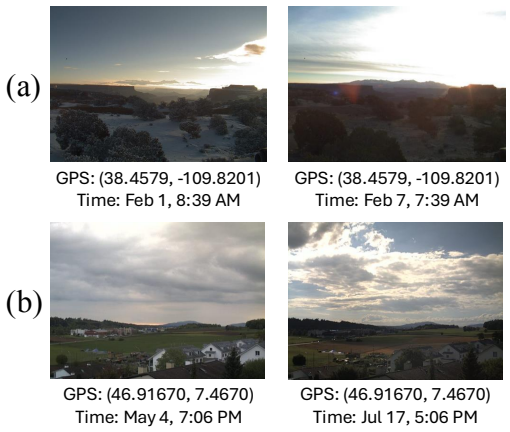

Figure 9. Illustrating *compositional* $L + T \rightarrow I$ retrieval with GT-Loc. Each example Left: query location and query time, showing actual image. Right: retrieved image for given query location and time.

15. Additional geo-localization analysis

Figure 10 presents the cumulative geolocation error evaluated over a range of distance thresholds. In addition to our main results, we include evaluations on two extra datasets: YFCC26k and OSV-5M. These plots allow for a more comprehensive comparison of model performance across diverse data distributions. We also include a comparison against the hybrid method proposed by Astruc et al. [1], which was not shown in the main paper. We observe that our method, GT-Loc, consistently outperforms Astruc et al. [1] on all datasets with the exception of OSV-5M. We attribute this to the fact that OSV-5M contains imagery that is in-domain for their model, whereas it is out-of-domain for ours.

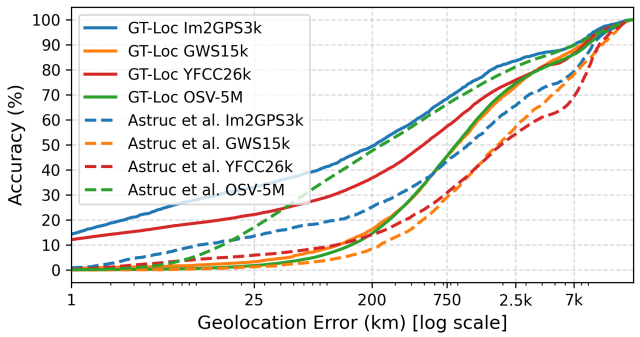

Figure 10. Cumulative geolocation error at different thresholds.

16. Effect of population density on geo-localization

Geo-localization datasets, particularly those collected via web-scraping or social media platforms, are inherently biased toward regions with higher population density. This raises a natural question: how does the performance of geo-localization algorithms vary between densely and sparsely populated areas? To investigate this, we conduct an additional analysis. For each image in our dataset, we obtain the corresponding population density from the Gridded Population of the World version 4 (GPWv4) dataset [39]. We then group the data into ranges based on population density (measured in pop/m^2) and compute the median geo-localization error within each group. This stratified evaluation allows us to assess model performance across regions with varying levels of human activity. Consistent with the findings of Astruc et al. [1], we observe that geo-localization errors tend to be higher in areas with lower population density, as summarized in Table 10.

Table 10. Median geo-localization error by population density group.

Pop. density	Im2GPS3k (km)	GWS15k (km)
< 100	266.66	1244.22
$[100, 1k)$	246.12	818.65
$[1k, 10k)$	197.99	639.73
$\geq 10k$	38.32	1035.81

17. Model performance vs. gallery size

Figure 11 shows how the size of the gallery affects geo-localization and time prediction performance. We observe that increasing the size of the gallery leads to improved results, but the gains tend to saturate quickly. In particular, a gallery containing 100,000 samples is already sufficient to capture the majority of the performance improvements for geo-localization. Interestingly, for time-of-capture prediction, we find that even smaller archives, such as those with only 4,000 samples, can still yield strong performance.

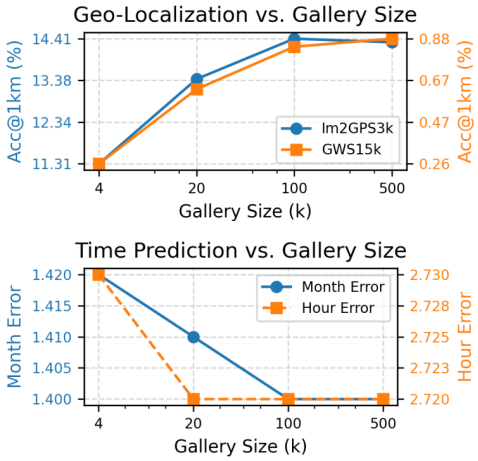

Figure 11. Effect of gallery size on geo-localization and time prediction.

18. Analysis of the learned embedding space

One of the key motivations for GT-Loc is to align images, time, and location in a shared multimodal embedding space. This approach is inspired by prior works like GeoCLIP [37], SatCLIP [11], and CSP [18], which embed images and GPS coordinates in shared spaces, as well as methods like ImageBind [4], LanguageBind [48], Everything At Once [31], and Preserving Modality [32], which align multiple modalities such as images, text, videos, and audio. Our work extends this idea to include temporal information, showing in Table 1 that aligning these three modalities leads to improved time prediction performance compared to using only images and time.

To explore the relationships between these modalities in the learned embedding space, we performed Principal Component Analysis (PCA) on the embeddings. While PCA has limitations in fully capturing the underlying structure of high-dimensional spaces, the results provide interesting qualitative insights. Figure 12(a) presents the distributions of image, time, and location embeddings, appearing in different subspaces. Figures 12(b) and 12(c) show more details about the relationship between image and time embeddings. The image embeddings are clustered in the center, surrounded by time embeddings. Notably, the directions of hours and months are well-defined: months are radially distributed, while hours are linearly distributed in a perpendicular direction. For location embeddings (Figures 12(d-e)), even though the patterns are less pronounced, the embeddings at different latitudes and longitudes still form distinct clusters in the feature space.

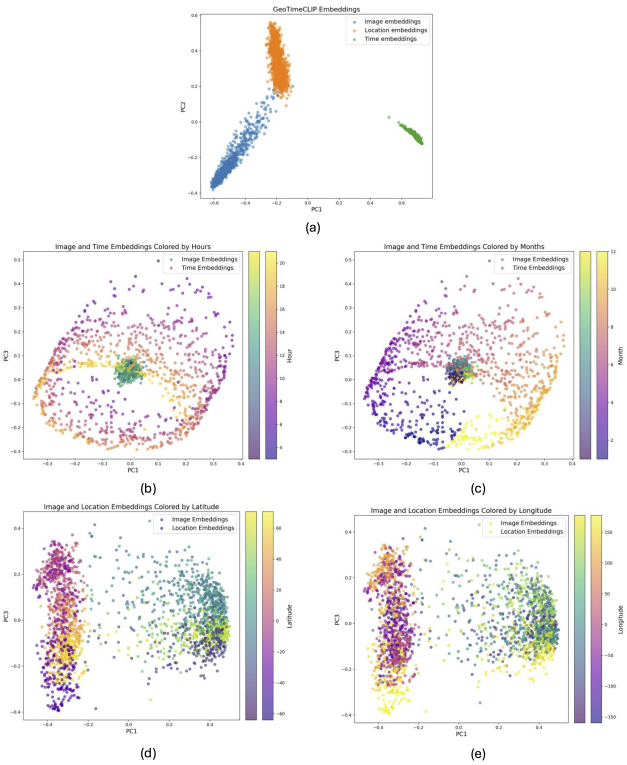

Figure 12. PCA plots of the embedding spaces in GT-Loc. (a) Distribution of the image, time and location embeddings. (b)-(c) Distribution of the image and time embeddings, colored by the time-of-day and time-of-year respectively. (c)-(d) Distribution of the image and location embeddings, colored by the latitude and longitude respectively.

19. Embedding distribution of non-overlapping panorama crops

To further analyze the structure of our learned embedding space, we conduct a qualitative visualization using t-SNE (Figure 13). We begin by randomly selecting 20 panoramas from the CVUSA dataset [42]. From each panorama, we extract four non-overlapping 90° crops, resulting in a total of 80 image embeddings. Since all four crops from a single panorama share the same capture time and location by definition, they serve as a natural test of spatial and temporal consistency in the embedding space. In the resulting t-SNE plot, we observe that embeddings from the same panorama (shown using the same color) form tight and coherent clusters, suggesting that our model effectively encodes shared contextual information across views from the same scene.

20. Scalability of the retrieval approach

In Table 11, we present a comparison of the memory usage and computational cost (measured in FLOPs) between classification and regression baselines and our retrieval-based approach, across gallery sizes ranging from 4,000 to 500,000 samples. At the upper end of this range (500k), we observe that memory usage increases by approximately 950 MB. However, the additional computational cost introduced by the retrieval operations remains minimal, with only a +3.32% increase in FLOPs. It is important to note that the gallery embeddings used for retrieval are precomputed offline in a single forward pass (requiring 9.47 TFLOPs) and are therefore not included in the runtime FLOPs reported in the table.

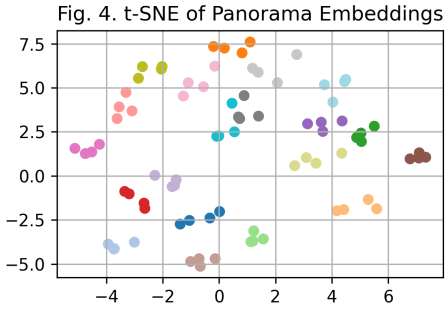

Figure 13. t-SNE visualization of image embeddings from non-overlapping 90° crops of 20 panoramas sampled from the CVUSA dataset. Each color represents the four crops from a single panorama, which share the same time and location.

Table 11. Memory usage and compute cost.

Method	Memory (GB)			TFLOPs		
(gallery size)	Params	Gallery	Total	Forward	Retrieval	Total
CLIP + cls	1.63	–	1.63	159.41	–	159.41
CLIP + reg	1.63	–	1.63	159.41	–	159.41
GT-Loc (4k)	1.67	0.01	1.68	159.41	0.004	159.41
GT-Loc (20k)	1.67	0.04	1.71	159.41	0.021	159.43
GT-Loc (100k)	1.67	0.19	1.86	159.41	0.105	159.52
GT-Loc (500k)	1.67	0.95	2.62	159.41	0.524	159.94

21. Limitations and reproducibility challenges of existing time prediction methods

Most previous time prediction methods suffer from a lack of standardization in their training and evaluation protocols. For instance, Zhai et al. [45] use subsets of the AMOS [8, 9] and YFCC100M [34] datasets, without providing the source code or exact dataset splits necessary to replicate their experiments. Additionally, their time prediction evaluation relies on cumulative error plots, but they do not provide a single numerical value summarizing the performance of their model. Similarly, while Salem et al. [26] and Padilha et al. [22] offer datasets and code, they do not include the cross-camera split for zero-shot time prediction—a more challenging and informative evaluation protocol that we adopt. Moreover, their results are presented only qualitatively, though they can be adapted for obtaining quantitative results. Salem et al.

[27] also omit critical details such as dataset splits for the SkyFinder dataset, do not clarify whether their results corresponds to same- or cross-camera evaluation, and fail to provide the source code for replication. Their use of top-k accuracy as an evaluation metric further complicates direct comparisons. Notably, none of these methods, with the exception of Salem et al. [26] and Padilha et al. [22], compare their time prediction performances against each other, and even these comparisons are only qualitative. Other time prediction approaches, including those by Tsai et al. [36], Li et al. [16], Lalonde et al. [14], and Hold-Geoffroy et al. [6], also face similar challenges, such as a lack of available source code, missing datasets, or datasets that are no longer hosted online. This lack of standardization prevents consistent benchmarking across different methods.

22. Qualitative time-of-day results near sunrise and sunset

Predicting the time of day during periods close to sunrise and sunset is particularly challenging due to the visual similarity of scenes captured around these times. The task is further complicated by its strong dependence on geographic location (latitude) and the time of year (month), both of which have a direct influence on the time when these events occur. To illustrate these challenges, Figure 14 presents two qualitative examples where GT-Loc makes accurate predictions, as well as two examples where it fails.

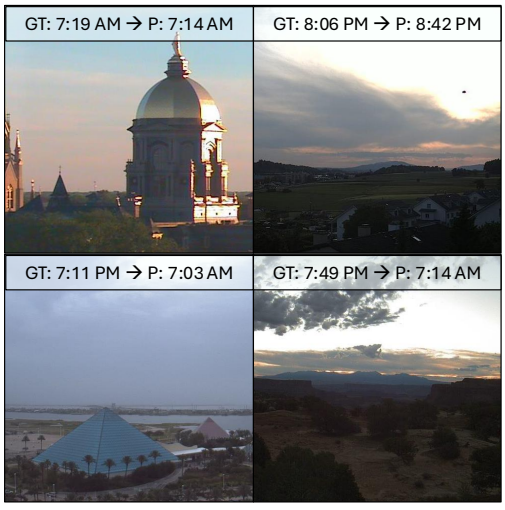

Figure 14. Qualitative time-of-day results near sunrise and sunset. Top row: correct predictions where the prediction (P) is within one hour of the ground truth (GT). Bottom row: failure cases.

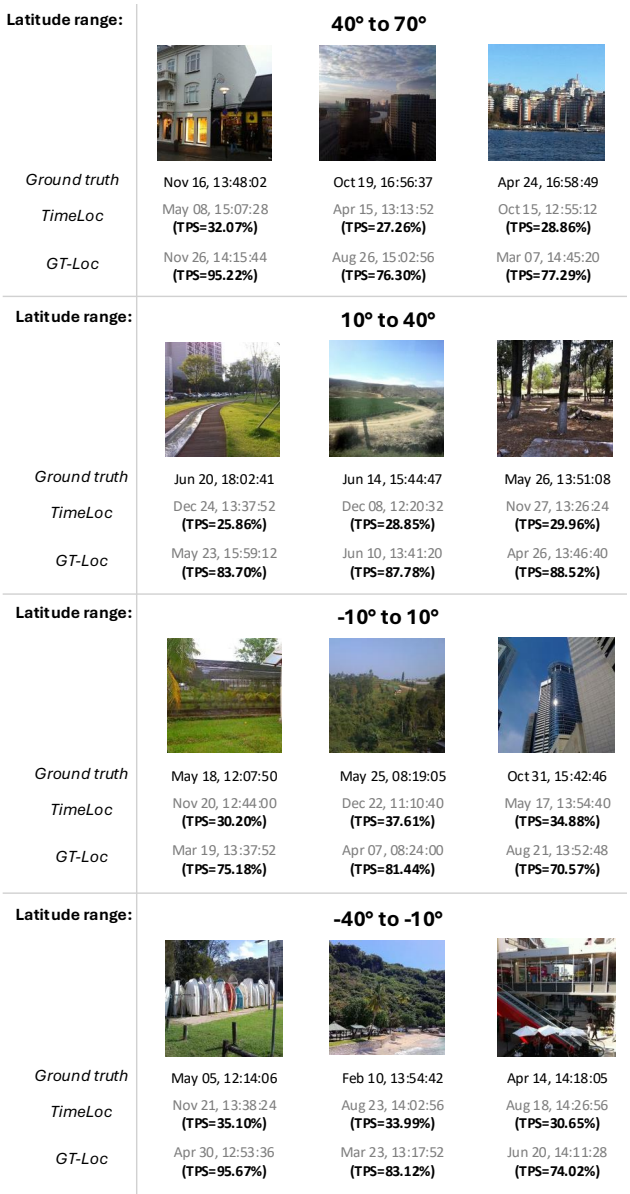

Figure 15. Sample predictions where GT-Loc outperforms the TimeLoc baseline across different latitudes.

23. GT-Loc predictions across different latitudes

Figure 15 compares time prediction examples from GT-Loc and TimeLoc, a baseline model trained solely with the visual ($\mathcal{V}$) and temporal ($\mathcal{T}$) encoders. The results suggest that TimeLoc struggles more with hour predictions at higher latitudes (40° to 70°) compared to GT-Loc. In contrast, at moderate latitudes (-40° to 40°), both models exhibit more consistent hour prediction errors, though GT-Loc demonstrates superior performance in month prediction.