Computer Vision for International Border Legibility

Trevor Ortega[1], Thomas Nelson[1], Skyler Crane[1], Josh Myers-Dean[2] and Scott Wehrwein[1]

[1]Western Washington University {ortegat, nelso343, cranes2, wehrwes}@wwu.edu
[2]University of Colorado Boulder josh.myers-dean@colorado.edu

1. Data Collection

1.1. Aerial Imagery Collection

Our border imagery dataset is comprised of of 612,374 overhead aerial images covering the entire world's international land borders, each with metadata describing the image space coordinates of the border. This section provides details on the collection and processing of the global borders dataset. As discussed in the main paper, the image tiles in our dataset are each 256x256 RGB pixels and cover a geographical area of 400m by 400m, leading to a total worldwide land coverage of over 90 billion square meters.

We start with the ground truth for political border locations from the *International-Borders-2* dataset from Simmons and Kenwick [6], a Shapefile in which each border between two countries is given as one or more linestrings of GPS coordinates. Our high-level approach is to walk along each border and query the Bing Maps API for image tiles at intervals that achieve close to full image coverage of the border. For tiles with geographic side length L (400 meters, in our case), we sample points spaced at intervals of L along the border using the Haversine distance formula. The geographic extent of a tile for a given zoom level varies due to the map projection. As a result, we dynamically select the zoom level (in practice, 16 or 17) that results in a side length as close as possible to our target dimension of 400 meters.

The Bing Maps API provides two mechanisms for retrieving image tiles; one (`https://dev.virtualearth.net/REST/v1/Imagery/Map`) yields a custom-cropped image with the requested center coordinates, zoom level, and spatial extent, while the other (`https://dev.virtualearth.net/REST/v1/Metadata/Map`) returns metadata including links to fixed-resolution map tiles stored internally by Bing that contain the query point. While centering each tile around a point on the border would provide consistent context surrounding the border, the former method has the drawback that the returned image may be composited from multiple different underlying tiles that were not necessarily taken at the same time. While not very common, we found that the custom cropped images sometimes contained clear boundaries between composited images taken under different conditions. The resulting seams are easily mistaken for border features, and have the potential to corrupt both the human annotations and fool our automated methods. As a result, we use the imagery metadata API, which returns metadata and URLs for tiles containing the query point. This means that the border location may not go through the center of the image.

Saved alongside each image in our dataset is an ordered list of pixel indices corresponding to the physical location of the GPS coordinates captured by the image. To determine where exactly the border lies within the image's geographical area, we use Shapely [3] to calculate the intersection between the image's GPS bounding box and the border's line string. The calculated intersection, along with any other border coordinate contained within the bounding box, are collected as data points associated with the given tile. Before saving, we convert the data points from GPS coordinates within a geographical bounding box to pixel coordinates within the image boundaries. The conversion between GPS coordinates, (x, y), and pixel coordinates, (i, j) given a GPS bounding box, $(x_{min}, y_{min}, x_{max}, y_{max})$ is described by:

$$i = \frac{S}{x_{max} - x_{min}} * (x - x_{min}) \quad (1)$$

$$j = \frac{S}{y_{max} - y_{min}} * (y - y_{min}) \quad (2)$$

Where S is the side length in pixels of each side of the image returned from the Bing Maps API (256 in our case). The saved pixel coordinates can be turned into segmentation masks when necessary by drawing interpolated lines between each pair of coordinates in pixel space. The saved coordinates and segmentation masks comprise the ground truth for border locations.

While the Bing Maps API terms of use prevent us

from releasing the dataset, we will make publicly available upon publication data and code necessary to reproduce our dataset. This will include the query location and border metadata for each tile, as well as the scraping code we used to generate the query locations and download the tiles.

1.2. Ground Truth Legibility Comparisons

We collected a dataset of ground truth pairwise legibility comparisons among 1000 images chosen from the global imagery dataset. We selected 1000 random tiles; to avoid tiles where the border just barely nicks the corner of the image, for example, we filtered out tiles with negligible border length and/or negligible image content on either side of the border. Carlson and Montgomery's framework [2] suggests collecting approximately 20 comparisons per datum. To account for the possibility of bad annotators, we added a 20% buffer and generated 12,030 pairs such that a given image should, on average, be compared to about 24 others.

We collected ground truth annotations using Amazon Mechanical Turk (AMT), asking workers to choose one of two tiles presented with the more legible border. The location of the border was drawn on the tiles, but could be toggled off. Tasks were grouped in batches of 30 pairs. Before performing the task, annotators were given detailed instructions (included as a separate document in the supplemental material) and required to complete a qualification test with six test pairs to demonstrate their understanding of the task.

1.3. Annotator Quality and Problem Ambiguity

To validate the quality of the human annotations, we calculated each annotator's accuracy with respect to the majority-vote winner for all pairs of images they annotated. Figure 1 plots each annotator's accuracy (red) alongside four points of reference (blue). All annotators landed well above the Random Simulated Worker, suggesting that none of our annotators were making entirely random or uninformed guesses. The ranking framework helps to control noise due to low-quality annotations by simultaneously modeling annotator skill and putting tiles on a single, globally consistent scale.

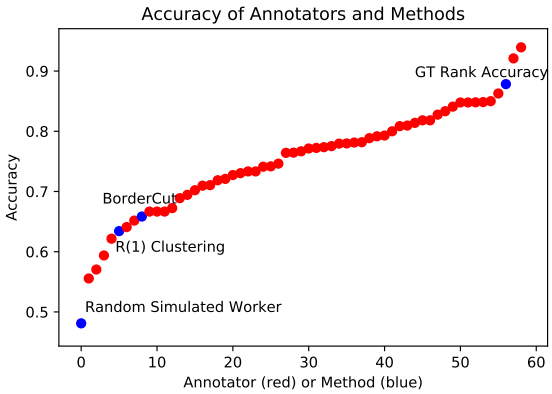

Figure 1: Accuracy (compared to the final ground-truth ranking) of annotators, a simulated random worker, two of our methods, and a majority-vote oracle.

To help quantify the underlying problem ambiguity that remains after tiles are globally ranked, we computed an "oracle" upper-bound accuracy from the ranking: the rate at which the rank order of a pair agreed with the majority-vote annotations for a given pair was 87.83%. This suggests that even a perfectly designed method may only score up to about 88% on the accuracy metric for the ground truth set. In future work, we plan to more rigorously quantify sources of error and devise additional metrics that take underlying problem ambiguity into account.

2. Additional Baseline Methods

We experimented with several baseline methods in addition to those given in Section 4.1 of the main paper. This section describes a class of methods using Gaussian Mixture Models. Results including these methods are given in Table 1, which is an expanded version of the main paper's Table 1.

Gaussian Mixture Models. Given the performance of our clustering baseline, we experimented with Gaussian Mixture Models as a similar but more probabilistically principled approach. Our high-level idea was to fit GMMs for each segment and use the likelihood of pixels from one segment given the model fit from another as an indication of similarity or differences among segments.

We fit three Gaussian Mixture Models, one trained on the features in each of F_A, F_B, F_C. The dissimilarity between each pair of segments is then calculated as the difference in likelihood between another segment and a segment with itself. After fitting GMM_A, GMM_B, and GMM_C to the features in each segment, $D(F_A, F_B)$ is the difference in per-sample average log likelihood of B to GMM_A and A to GMM_A. In other words,

$$D(F_A, F_B) = p_{GMM_A}(B) - p_{GMM_A}(A)$$

We compute $D(\cdot, \cdot)$ for all (ordered) pairs of feature collections, taking the maximum score as the legibility score. In our experiments, we fit mixture models with three components. Though arguably more principled, this approach performed worse the corresponding clustering baseline with pixels as input features.

Cosine Distance GMM. Fitting GMMs on CNN and transformer features gave unstable results due to their higher dimensionality. To get around this issue, we experimented with computing GMMs of *pairwise feature distances*, with the idea that these 1D quantities would still

Method	$\phi^{\mathbf{P}}$			$\phi^{\mathbf{R(1)}}$			$\phi^{\mathbf{R(2)}}$			$\phi^{\mathbf{R(3)}}$			$\phi^{\mathbf{T}}$		
	Acc.	τ	**Footrule**	**Acc.**	τ	**Footrule**	**Acc.**	τ	**Footrule**	**Acc.**	τ	**Footrule**	**Acc.**	τ	**Footrule**
Distance	62.40	0.151	277.91	60.28	0.084	301.38	58.17	0.059	313.26	56.19	0.261	244.82	51.63	-0.027	332.82
Clustering	61.28	0.075	305.33	**63.42**	0.116	290.92	60.19	0.209	262.53	58.80	**0.449**	**186.68**	49.86	0.032	314.49
Pixel GMM	60.33	0.065	310.69												
Distance GMM				60.51	0.061	294.48	57.91	0.081	254.33	52.36	-0.016	335.02	55.20	-0.004	320.01

Table 1: An expanded version of Table 1 from the main paper giving baseline results on the annotated validation set. Distance: Average Pairwise Feature Distance; Cluster: Cluster Assignment Distributions; Pixel GMM: Gaussian Mixture Models; Distance GMM: Cosine Distance GMM. Feature extractors ϕ are as described in the main paper. Higher is better for accuracy and τ, while lower is better for Footrule.

be informative and might outperform the simple averaging approach of our Distance baseline. We perform the same cosine distance operation as in our Distance method, but then fit a unique GMM for each pair of segments. In other words, instead of averaging as in Equation 2 of the main paper, we fit $GMM_{1,2}$ to the collection of pairwise feature distances $D_{12} = \{d(f_1, f_2) : f_1 \in F_1, f_2 \in F_2\}$. We build one GMM for each pair of segments AB, AC, BC, and define the pairwise dissimilarity $D(D_{AB}, D_{BC})$ analogously to above:

$$D(D_{AB}, D_{AC}) = p_{GMM_{AB}}(D_AC) - p_{GMM_{AB}}(D_AB)$$

The overall legibility score is once again the maximum such dissimilarity over all (ordered) pairs of AB, AC, AB. This method did not turn out to achieve a clear improvement over the distance averaging, and also underperformed the Clustering method even at its best, using $\phi^{R(1)}$ features.

3. Other Methods Considered

Early on in our experimentation, we considered various alternative methods unrelated to those discussed in Section 4 of the main paper. While we ultimately decided not to pursue these methods further, we briefly discuss them here.

Direct Prediction We considered using border location prediction as a proxy task, with the idea that a model trained to predict the border would do well on highly legible borders and poorly on illegible borders. In this context, we considered multiple deep and low-level approaches: Semantic Segmentation[4], Edge Detection [1], Deep Line Detection [7, 8, 9], and directly regressing an angle and offset to represent the border as a single line.

While many of these methods are capable at locating image features which may be considered border-like, they are not equipped to deal with the context and properties unique to political borders. In particular, border-like features are common even when they don't correspond to the border (*e.g.*, two vibrant cities divided by a wall will have many edge features in addition to the wall itself). Meanwhile, many other (illegible) tiles have no features whatsoever (*e.g.*, a featureless desert), and thus providing no training signal in the proxy task. As mentioned in the main paper, this makes for a challenging learning task and our attempts did not result in models that outperformed our simpler baselines.

Inpainting We considered framing border legibility as the ability for a model to reconstruct a border tile given a missing segment of the border. The intuition is that certain segments of an image that are crucial for border identification, such as a fence along the length border, will have substantially worse reconstruction scores when removed. Using the Masked AutoEncoder [5], we find impressive decoding performance on masked portions of images from our dataset. For straightforward scenarios like a fence along the border, we found that this framing of the task works. However, for images where features do not follow the border exactly (*e.g.*, a single road that is partially, but not entirely, covered by the border mask) it is possible for a model to reconstruct the missing features with high accuracy despite partial masking. In practice we found that a large number of tiles had features that did not align perfectly with the border, so the overall performance remained poor.

References

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] David Carlson and Jacob M Montgomery. A pairwise comparison framework for fast, flexible, and reliable human coding of political texts. *American Political Science Review*, 111(4):835–843, 2017.

[3] Sean Gillies et al. Shapely: Manipulation and analysis of geometric objects. `https://github.com/Toblerity/Shapely`, 2007–.

[4] Eric Guérin, Killian Oechslin, Christian Wolf, and Benoît Martinez. Satellite image semantic segmentation, 2021.

[5] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[6] Beth A. Simmons and Michael R. Kenwick. Border orientation in a globalizing world. *American Journal of Political Science*, n/a(n/a), 2022.

[7] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4257–4266, June 2021.

[8] Nan Xue, Song Bai, Fudong Wang, Gui-Song Xia, Tianfu Wu, and Liangpei Zhang. Learning attraction field representation for robust line segment detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] Kai Zhao, Qi Han, Chang bin Zhang, Jun Xu, and Ming ming Cheng. Deep hough transform for semantic line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021.