Thematic2.5D: A Toolkit for Evaluating 2D and 3D Feature Effects in Supervised Classification

Taqi Hamoda[1], **Hayat Rajani** [1], **Ahmed Alketbi**[2], **Arthur Gleason**[3], and **Nuno Gracias** [1]

1 University of Girona, Spain ROR **2** Technology Innovation Institute, UAE **3** University of Miami, USA

Summary

The accurate detection and classification of objects in complex environments is a critical task across various domains, requiring robust methods to process multi-modal data, such as optical imagery and 3D reconstructions. Object detection often leverages computer vision, machine learning, and 3D modeling to distinguish targets from complex backgrounds. Existing approaches relying solely on 2D image features often suffer from high false positive rates due to background variability, necessitating the integration of 3D geometric data to enhance accuracy.

Thematic2.5D is an open-source Python package designed for supervised classification using multi-modal data, including optical imagery and depth-maps derived from 3D reconstructions. Building on the methodology of Gleason et al. (2015), the package provides a modular pipeline for dataset creation, feature extraction, classification, and evaluation. Its key features include:

- **Dataset Creation:** Automated tile extraction and labeling from imagery and depth-maps for diverse applications.
- **Feature Extraction:** Extraction of 2D-derived (color, texture) and 3D-derived (curvature, rugosity) features, which can be combined into a hybrid 2.5D feature set.
- **Classification:** Support for binary (object vs. background) and multi-class classification using Support Vector Machines (SVM).
- **Inference and Evaluation:** A complete inference pipeline to generate prediction masks on new data and quantitative evaluation using mean Intersection over Union (mIoU).
- **Modularity:** A flexible, configuration-driven framework allowing integration of new features, models, and data modalities.

Thematic2.5D serves as a versatile toolkit for researchers, data scientists, and educators working in computer vision, machine learning, or thematic mapping. It integrates with popular scientific Python libraries and supports workflows for both research and practical applications, such as environmental surveys, urban mapping, or industrial inspections. The package provides a classical framework to evaluate the relative contributions of 2D and 3D feature modalities, enabling users to quantify the impact of integrating geometric data with traditional image features for improved classification performance.

Statement of Need

Object detection in complex environments is a widespread challenge across fields like environmental monitoring, urban planning, and industrial inspection, where distinguishing objects from varied backgrounds is critical. Traditional detection methods, such as those using acoustic or optical imagery, are often limited by resolution or background complexity. Optical imagery combined with depth-maps offers high-resolution data for precise object identification, as demonstrated by Gleason et al. (2015). However, existing software tools are often proprietary,

---

41 domain-specific, or lack the flexibility to handle multi-modal data in a unified framework.

42 `Thematic2.5D` addresses this gap by providing a free, open-source Python toolkit that im-
43 plements a classical supervised classification pipeline, inspired by Gleason et al. ([2015](#)), for
44 semantic segmentation and thematic mapping. The package serves as a robust framework
45 for evaluating the effectiveness of 2D-derived (e.g., color, texture) versus 3D-derived (e.g.,
46 curvature, rugosity) features, allowing researchers to assess the relative contributions of these
47 modalities to classification accuracy.

48 Its modularity enables experimentation with new features, classifiers, or data sources (e.g.,
49 sonar, stereo vision), while its accessibility supports educational use in courses on machine
50 learning, computer vision, or data science. By offering a comprehensive pipeline from data
51 ingestion to evaluation, `Thematic2.5D` lowers barriers to entry for multi-modal classification
52 research, fostering innovation across diverse applications.

53 Background

Figure 1: Inference output of the Thematic2.5D model trained to detect underwater military munitions (UWMM). The leftmost image represents the output of the model trained exclusively on 2D-derived features, the central image shows the output of the model trained exclusively on 3D-derived features, and the rightmost image illustrates the output of the model trained on 2.5D extracted features.

54 Object detection in complex scenes involves processing optical imagery to identify targets
55 against varied backgrounds (e.g., natural landscapes, urban settings). The core challenge
56 lies in distinguishing objects from similarly shaped or textured background elements. It was
57 demonstrated that while 2D-derived features (color, texture) achieve moderate accuracy ($>$80%
58 for binary classification), they suffer from high false positives due to background complexity
59 ([Gleason et al., 2015](#)). Incorporating 3D-derived features (e.g., curvature, rugosity) from depth-
60 maps significantly improves accuracy (89-95%) and reduces false positives by capturing distinct
61 geometric properties. This combined, hybrid approach, referred to as the 2.5D approach,
62 enhances classification performance by leveraging both visual and geometric information.

63 The `Thematic2.5D` pipeline provides a classical framework to systematically evaluate the
64 contributions of 2D and 3D features. The methodology is broken down into a multi-stage
65 pipeline. First, large survey images are processed into smaller, manageable tiles. This approach
66 allows the model to learn local, high-resolution features and creates a large, diverse dataset
67 suitable for training supervised machine learning models. The general workflow is as follows:

68 1. **Dataset Creation:** Optical images, depth maps, and ground-truth masks are used to
69   generate a labeled dataset of image tiles.
70 2. **Feature Extraction:** A comprehensive set of 2D-derived and 3D-derived features is
71   extracted from each tile.
72 3. **Model Training:** A classifier, such as an SVM, is trained on the extracted features.

4. **Inference:** The trained model is used to predict the locations of objects in new, unseen images.
5. **Evaluation:** The model's predictions are compared against ground-truth data to assess performance quantitatively.

This project implements this entire workflow in a configurable and automated pipeline, building on the foundational work of Gleason et al. (2015), to provide an accessible software tool for evaluating feature modalities in supervised classification.
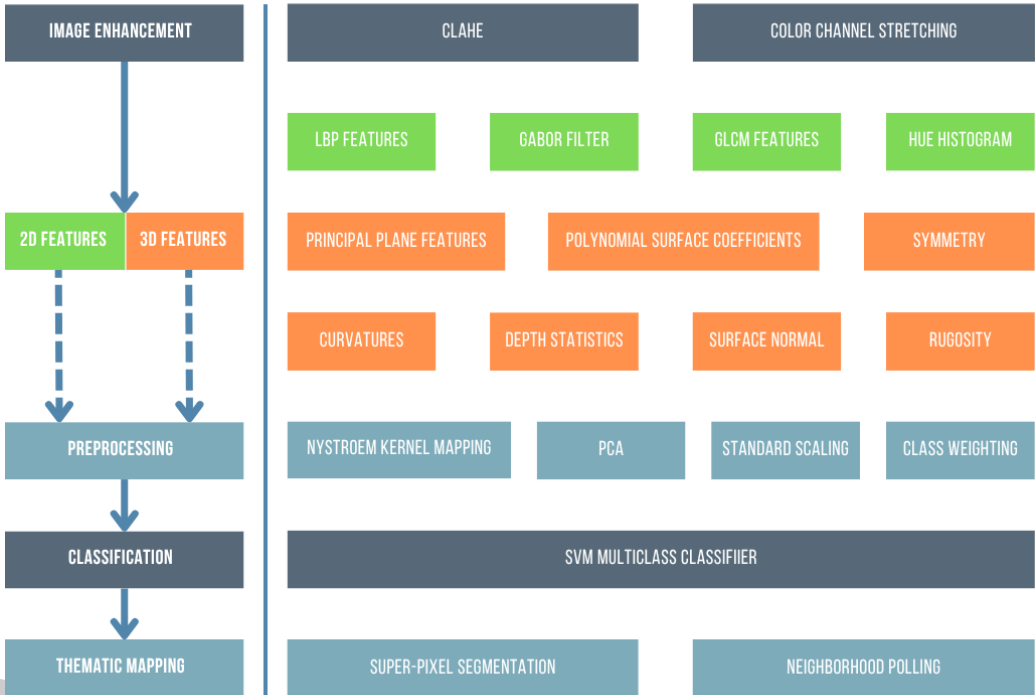
Methodology

Figure 2: The implemented classification framework is depicted in the above flowchart. The left section outlines the primary stages of the pipeline, while the right section details the specific components and features utilized in each stage. The dashed arrows represent the path being optional.

The `Thematic2.5D` pipeline processes images, depth-maps, and masks to generate a labeled dataset of uniformly sized tiles for training and testing. It identifies potential object and background locations using masks, extracting corresponding square tiles from image and depth data. To address class imbalance, a fixed number of background tiles are sampled per image, and only a subset of available object pixels are used as tile centers. Data augmentation is applied by rotating object tiles at multiple angles.

From the processed tiles, 2D-derived and 3D-derived features are extracted, extending the feature set proposed by Shihavuddin et al. (2014) and Gleason et al. (2015):

- **2D-derived Features (from optical images):**
  - **Color Histograms:** HSV color distributions to capture background and object appearance (Shihavuddin et al., 2013).
  - **Local Binary Patterns (LBP):** Texture descriptors robust to illumination changes (Ojala et al., 1994).

94 - **Gray Level Co-occurrence Matrix (GLCM):** Texture metrics (e.g., contrast, dissim-
95   ilarity) for background characterization (Haralick et al., 1973).
96 - **Gabor Filters:** Edge and texture detection across multiple scales and orientations
97   (Granlund, 1978).
98 • **3D-derived Features (from depth maps):**
99 - **Principal Plane Features:** Statistics of depth values, polynomial coefficients from a
100   fitted surface, and rugosity (ratio of surface area to planar area).
101 - **Curvatures and Normals:** Mean and Gaussian curvatures, shape index, curvedness,
102   and surface normal vector statistics.
103 - **Symmetry Features:** Gabor filters applied to depth maps to capture structural
104   symmetry.

105 The classifier is implemented using a SVM (Cortes & Vapnik, 1995) with a radial basis function
106 (RBF) kernel, approximated via the Nystroem method (Williams & Seeger, 2000). The model
107 supports training on 2D-derived, 3D-derived, or combined 2.5D feature sets. Prior to inference,
108 superpixel segmentation (Achanta et al., 2012) identifies homogeneous regions in the input
109 image. For each identified superpixel, a sliding window centered on its centroid is used to
110 extract a series of overlapping tiles. These tiles are individually processed through the trained
111 model to predict class labels. A neighborhood poll determines object presence by evaluating
112 whether the number of positive tile predictions within a region exceeds a predefined threshold.
113 The final output is a segmentation mask indicating detected objects locations (see Figure 1).

114 This implementation leverages standard Python libraries, including NumPy (Harris et al., 2020),
115 OpenCV (Bradski, 2000), scikit-learn (Pedregosa et al., 2011), scikit-image (Walt et al., 2014),
116 and SciPy (Virtanen et al., 2020) for feature extraction, model training, and data processing.

Model Training and 3D-derived Feature Extraction

118 The classification pipeline in `Thematic2.5D` relies on feature extraction and SVM-based classi-
119 fication. For a tile $\mathbf{x} \in \mathbb{R}^{m \times n}$ (image) and corresponding depth-map $\mathbf{d} \in \mathbb{R}^{m \times n}$, the feature
120 vector $\mathbf{f}_{2.5D}$ combines 2D-derived and 3D-derived features:

$$\mathbf{f}_{2.5D} = [\mathbf{f}_{2D}, \mathbf{f}_{3D}]$$

121 The 3D-derived features, $\mathbf{f}_{3D}$, are extracted from the depth-map of each tile. This process
122 involves several calculations to describe the geometry of the surface.

123 A 2D-derived polynomial surface of the third degree is fitted to the depth-map of a tile to
124 model its shape (Shihavuddin et al., 2014). The equation for this surface is:

$$f(x, y) = p_1 + p_2 x + p_3 y + p_4 x^2 + p_5 xy + p_6 y^2 + p_7 x^2 y + p_8 xy^2 + p_9 y^3$$

125 The nine coefficients of this polynomial $(p_1, ..., p_9)$ are extracted through least-squares fitting
126 and used as features (Shihavuddin et al., 2014).

127 Additional statistical features are calculated from the depth values (z_i) within a tile, including
128 the standard deviation, skewness, and kurtosis (Shihavuddin et al., 2014). With z_m as the
129 mean depth, S as the standard deviation, and N as the number of data points, the skewness
130 and kurtosis are calculated as:

$$\text{Skewness} = \frac{\sum (z_i - z_m)^3}{(N-1)S^3}$$

$$\text{Kurtosis} = \frac{\sum_{i=1}^{N} (z_i - z_m)^4}{(N-1)S^4}$$

With regards to surface curvature, the Gaussian (G) and mean (M) surface curvatures are first calculated from the partial derivatives of the depth map where the two principal curvatures, k_1 and k_2, can then be derived (Shihavuddin et al., 2014):

$$k_1 = M + \sqrt{M^2 - G}$$

$$k_2 = M - \sqrt{M^2 - G}$$

These principal curvatures are then used to calculate a shape index (S) and a curvedness index (C):

$$S = \frac{2}{\pi} \arctan\left(\frac{k_2 + k_1}{k_2 - k_1}\right), \quad C = \sqrt{\frac{k_1^2 + k_2^2}{2}}$$

Finally, rugosity (r) is computed to characterize the roughness of the seafloor habitat (Shihavuddin et al., 2014). It is calculated by dividing the contoured surface area of the tile (A_s) by the area of its orthogonal projection onto the principal plane (A_p):

$$r = \frac{A_s}{A_p}$$

Once the full feature vector is assembled, the SVM classifier solves the following optimization problem:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i$$

subject to:

$$y_i(\mathbf{w}^T\phi(\mathbf{f}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, N$$

where $\mathbf{w}$ is the weight vector, b is the bias, ξ_i are slack variables, C is the regularization parameter, ϕ is the kernel mapping (RBF), and y_i are the class labels (Cortes & Vapnik, 1995).

Acknowledgements

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(11), 2274–2282. https://doi.org/10.1109/TPAMI.2012.120

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

154 Gleason, A. C. R., Shihavuddin, A., Gracias, N., Schultz, G., & Gintert, B. E. (2015).
155 Improved supervised classification of underwater military munitions using height features
156 derived from optical imagery. *OCEANS 2015 - MTS/IEEE Washington*, 1–5. https:
157 //doi.org/10.23919/OCEANS.2015.7404580

158 Granlund, G. H. (1978). In search of a general picture processing operator. *Computer Graphics*
159 *and Image Processing*, *8*(2), 155–173. https://doi.org/10.1016/0146-664X(78)90047-3

160 Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). Textural features for image clas-
161 sification. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-3*(6), 610–621.
162 https://doi.org/10.1109/TSMC.1973.4309314

163 Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D.,
164 Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk,
165 M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant,
166 T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https:
167 //doi.org/10.1038/s41586-020-2649-2

168 Ojala, T., Pietikainen, M., & Harwood, D. (1994). Performance evaluation of texture measures
169 with classification based on kullback discrimination of distributions. *Proceedings of 12th*
170 *International Conference on Pattern Recognition*, *1*, 582–585 vol.1. https://doi.org/10.
171 1109/ICPR.1994.576366

172 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
173 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,
174 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.
175 *Journal of Machine Learning Research*, *12*, 2825–2830.

176 Shihavuddin, A. S. M., Gracias, N., Garcia, R., Campos, R., Gleason, A. C. R., & Gintert,
177 B. (2014). Automated detection of underwater military munitions using fusion of 2D and
178 2.5D features from optical imagery. *Marine Technology Society Journal*, *48*(4), 61–71.
179 https://doi.org/doi:10.4031/MTSJ.48.4.7

180 Shihavuddin, A. S. M., Gracias, N., Garcia, R., Gleason, A. C. R., & Gintert, B. (2013). Image-
181 based coral reef classification and thematic mapping. *Remote Sensing*, *5*(4), 1809–1841.
182 https://doi.org/10.3390/rs5041809

183 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D.,
184 Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson,
185 J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy
186 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in
187 Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

188 Walt, S. van der, Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager,
189 N., Gouillart, E., Yu, T., & contributors, the scikit-image. (2014). Scikit-image: Image
190 processing in Python. *PeerJ*, *2*, e453. https://doi.org/10.7717/peerj.453

191 Williams, C. K. I., & Seeger, M. (2000). Using the nyström method to speed up kernel machines.
192 *Proceedings of the 14th International Conference on Neural Information Processing Systems*,
193 661–667.