Cubist-style image effects with oblique decision trees

Edric Chan
Great Oak High School
edric1106@gmail.com

Magzhan Gabidolla
Dept. CSE, UC Merced
mgabidolla@ucmerced.edu

Miguel Á. Carreira-Perpiñán
Dept. CSE, UC Merced
mcarreira-perpinan@ucmerced.edu

Image effects and restyling There is a range of computer-based image effects which can be used for various tasks, such as reconstruction, enhancement, noise removal, synthesis—or simply aesthetic effects, on which we focus. Among the simpler methods are non-adaptive local convolutional filters for blurring or edge detection [8], which are implemented in image editing packages such as PhotoShop or the Gimp in Linux. Recently, machine learning approaches based on neural networks have had a wild success. Neural style transfer [7] learns from a single pair of images to transform one using the visual "style" of the other. Generative AI methods, such as diffusion models [10], learn to generate new, realistic images from a large dataset of images. Here we show how decision trees, trained on a single image, are able to rerender it in a style reminiscent of cubist paintings. We provide a brief summary of the approach and a representative selection of the effects it can produce.

Image modeling as a regression problem with forests An image can be considered as a dataset of pixels in a 2D grid, thus as a sample of a mapping from $\mathbb{R}^2$ to a feature space (intensity, color, texture, etc.). We can learn a parametric model by using the image to define a least-squares regression problem. We consider oblique regression trees with constant leaves, which threshold a hyperplane (a line in 2D) in the decision nodes and output a constant vector (3D for color) at each leaf (fig. 1). For a forest with T trees, the output is the average of the trees' outputs. A regression forest with complete trees of depth Δ defines a piecewise constant function with up to $2^{T\Delta}$ pieces, which are convex polygons. On an image, this creates a peculiar, cubist-like effect (although our procedure is closer to that of pixelation or mosaics). Axis-aligned trees (not shown here) use vertical or horizontal splits which produce blocky, Mondrian-like images, since every polygon is a rectangle.

Optimization We use the Tree Alternating Optimization (TAO) algorithm [2] to optimize the squared error over a single tree. This optimizes the parameters of each decision node or leaf given all other nodes are fixed and decreases monotonically the error over the whole tree. This implies that the optimal value at a leaf equals the average of the pixels it contains. The cost of each iteration is $\mathcal{O}(\Delta N)$, where N is the number of pixels. Further, all nodes at the same depth can be updated in parallel. TAO learns trees that are much smaller and more accurate than the traditional greedy recursive partitioning algorithms such as CART [1] or C5.0 [9], and it has made it possible to learn trees for tasks such as clustering, dimensionality reduction, semi-supervised learning or imbalanced classification [5, 6, 11, 12]. For forests, we use the Forest Alternating Optimization algorithm (FAO) [3], which applies TAO one tree at a time (as well as using steps that optimize all leaves jointly). FAO also reduces the error monotonically and consistently results in smaller, more accurate forests than state-of-the-art forests such as XGBoost [4].

Model parameters and aesthetic effect Given an image, we control the "cubist" effect via the tree depth Δ, number of trees T, number of TAO/FAO iterations and choice of random seed. The initial tree is a median tree, which sets the decision hyperplane to a random direction that splits the data into two halves top-down recursively. This already produces a nice effect, but further iterations better adapt it to the input image. Small Δ results in smoother images, while increasing Δ and/or T produces more detailed, noisier images. We show a selection of effects on paintings, photographs, cartoons and calligraphy. **Zoom into the images to see individual polygons.** We used a C++/Python implementation using OpenMP for parallel processing. Our experiments were ran in a Linux laptop with processor Intel Core i9 2.2 GHz with 24 cores and 32 GB of RAM (using only CPUs, not GPU). We reduced the original images to a maximum size of 100×100 pixels. The time to run 10 TAO iterations (which is usually more than enough to reach convergence) ranges from less than one second to a few seconds (e.g. 20 seconds for $\Delta = 11$ and $T = 2$).

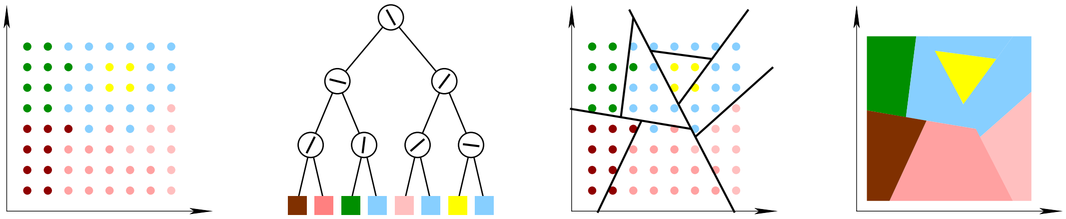

Figure 1: Learning an oblique regression tree to represent a single image. *Plot 1*: an input image, as a grid of points $\mathbf{x}_n \in \mathbb{R}^2$ (input features) each with a color $\mathbf{y}_n \in \mathbb{R}^3$ (output labels). *Plot 2*: an oblique regression tree of depth 3 learned on this dataset. *Plot 3*: the partition of the 2D space induced by the tree. *Plot 4*: the partition with each leaf polygon colored by the leaf label.

Leonardo | Picasso (cubist style) | $T = 1, \Delta = 7$ | $T = 1, \Delta = 13$ | $T = 4, \Delta = 6$

Figure 2: *Images 1, 2*: actual paintings. *Rest*: tree outputs for image 1. Zoom in for details.

original | $\Delta = 1, T = 1$ | $\Delta = 5, T = 1$ | $\Delta = 12, T = 1$ | $\Delta = 7, T = 2$ | $\Delta = 6, T = 5$

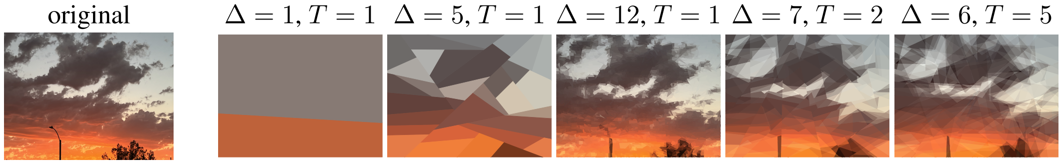

Figure 3: *Left*: photograph. *Rest*: tree outputs using different depths Δ and number of trees T.

original 1 2 3 8 | original 1 2 3 8

Figure 4: Tree outputs over training TAO iterations (*left*: $\Delta = 5, T = 1$; *right*: $\Delta = 6, T = 1$).

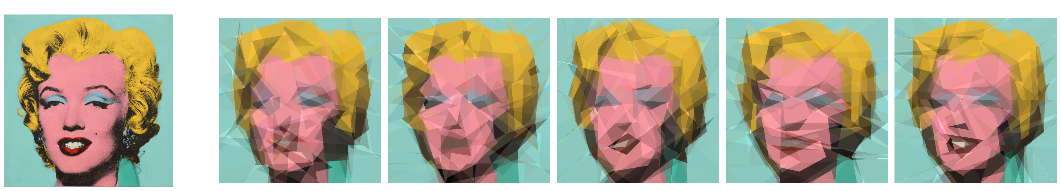

Figure 5: *Left*: original image. *Rest*: tree outputs using different seeds (for $\Delta = 6$, $T = 3$). Combining these images into a video produces an jittery effect reminiscent of rotoscopic animation. See https://youtu.be/TXPmOmw4a_A.

$\Delta = 14, T = 1$ | $\Delta = 6, T = 1$ | $\Delta = 8, T = 1$ | $\Delta = 10, T = 1$ | $\Delta = 9, T = 1$

Figure 6: Can you guess the original paintings, drawings or photographs?

Acknowledgments

Work partially supported by NSF award IIS–2007147. Edric Chan is a rising senior at Great Oak High School, Temecula, California. His contribution to this work took place during an internship at UC Merced in summer 2024. We thank Kuat Gazizov for help with the Python setup. The text of this paper was wholly written by humans, without the use of any large language model or other AI tool.

Image credits:

- Fig. 2: https://wikipedia.org/wiki/Mona_Lisa.
- Fig. 2: https://wikipedia.org/wiki/Les_Demoiselles_d%27Avignon.
- Fig. 3: photograph © 2022 Edric Chan.
- Fig. 4: https://www.kyotoanimation.co.jp/en/works/k-on02 and https://wikipedia.org/wiki/Taoism.
- Fig. 5: https://wikipedia.org/wiki/Shot_Marilyns.
- Original images (not shown) in fig. 6: the first three are from, respectively, https://wikipedia.org/wiki/The_Starry_Night, NASA/JPL-Caltech's https://www.spitzer.caltech.edu/image/ssc2008-10a-a-roadmap-to-the-milky-way and https://www.kyotoanimation.co.jp/en/works/k-on02. The last two are © 2022–2023 Edric Chan.

More examples are available online at the authors' web pages:

- https://edric-chan.github.io
- http://faculty.ucmerced.edu/mcarreira-perpinan/research/TAO.html

References

[1] L. J. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[2] M. Á. Carreira-Perpiñán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *NeurIPS*, pages 1211–1221, 2018.

[3] M. Á. Carreira-Perpiñán, M. Gabidolla, and A. Zharmagambetov. Towards better decision forests: Forest Alternating Optimization. *CVPR*, pages 7589–7598, 2023.

[4] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. *KDD*, pages 785–794, 2016.

[5] M. Gabidolla and M. Á. Carreira-Perpiñán. Optimal interpretable clustering using oblique decision trees. *KDD*, pages 400–410, 2022.

[6] M. Gabidolla, A. Zharmagambetov, and M. Á. Carreira-Perpiñán. Beyond the ROC curve: Classification trees using Cost-Optimal Curves, with application to imbalanced datasets. *ICML*, 2024.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. *CVPR*, pages 2414–2423, 2016.

[8] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, second edition, 2002.

[9] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[10] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):105:1–39, Apr. 2024.

[11] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Learning interpretable, tree-based projection mappings for nonlinear embeddings. *AISTATS*, pages 9550–9570, 2022.

[12] A. Zharmagambetov and M. Á. Carreira-Perpiñán. Semi-supervised learning with decision trees: Graph Laplacian tree alternating optimization. *NeurIPS*, pages 2392–2405, 2022.