Supplemental Material: Shape-Biased Domain Generalization via Shock Graph Embeddings

Maruthi Narayanan, Vickram Rajendran
Johns Hopkins University Applied Physics Laboratory
Laurel, MD 20723
{Maruthi.Narayanan, Vickram.Rajendran}@jhuapl.edu

Benjamin Kimia
Brown University
School of Engineering
benjamin_kimia@brown.edu

1. Shock Graph Computational Algorithm

Formally, the computation of shocks from an unorganized set of curve fragments relies on two key ideas. First, since the shocks arising from points and lines can be analytically computed, the curve fragments can be first approximated by a polyline, namely, a set of line segments that share end points; there are well-known algorithms to achieve this by specifying the expected error [1]. It is well-known that the bisector of two points is a line, the bisector of two lines is a line, and the bisector of a point and a line is a parabola, Figure 1. The dynamics of shocks moving on this bisector are less trivial but they can also be calculated analytically. The geometry and dynamics of shocks arising from any isolated pair from a collection of point and line sources is therefore readily available in analytic form and this saves in computation time and provides numerical stability robustness.

The second key idea is the use of a wave propagation and shock propagation to piece together shocks from multiple pairs of sources in a highly efficient way. Observe that the union of all shock-graph of isolated pairs of sources is a superset of the shock graph of the union of boundary sources, *e.g.*, as in the bisectors shown in grey and green which is a superset of the shock graph shown in green in Figure 2(h). Consider first the shock points on a bisector of a pair of boundary sources: a shock point is valid if no other waves from sources other than its own sources reach there. In other words, a shock is valid if its time of formation (distance to its sources) is earlier than the time of propagation (distance) to any earlier other source. Since the shock path on any bisector has an initial shock point (shock source), the extent of the valid shock path can be determined by first examining the validity of the shock source itself, the first shock point to form in time. Since a valid shock path can only be potentially terminated at the intersection of two shock paths, these are the only points that need to be examined. This is a key distinction since it discretizes the continuous propagation into a discrete propagation. Thus, the algorithm

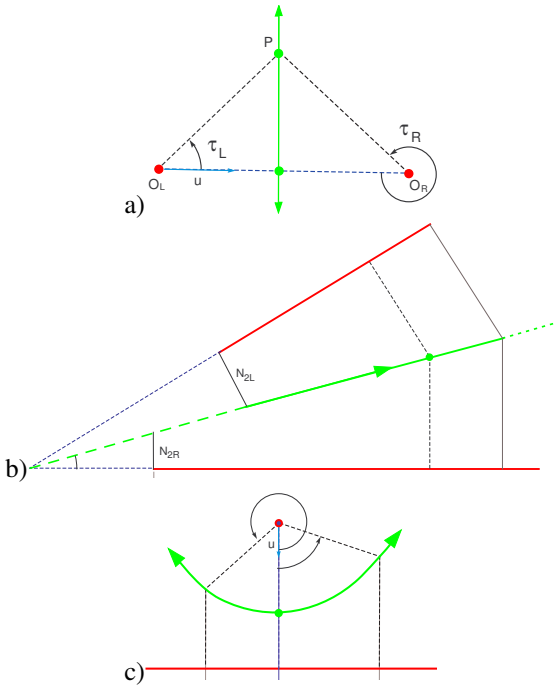

Figure 1. Analytic computation of shocks from polylines requires shocks from (a) point-point (b) line-line and (c) point-line pairs. Boundary sources are shown in red, shocks are shown in green, and the rays connecting boundary sources to respective shocks are shown in dashed lines. These computations assume only a pair of sources with no interaction from other sources so all shocks go off to infinity.

first computes all shock sources, an N^2 computation for N boundary sources, and considers shock paths for the sources in order of time.

Figure 2 illustrates this process. Consider two fragments, C_1 and C_2 whose potential shock path is shown in cyan in (a). For the purpose of illustrating the second key idea, namely using propagation to compute the shock graph, the contour fragments are not restricted to be polylines and can take any form. Also, these are sketches and not accurate

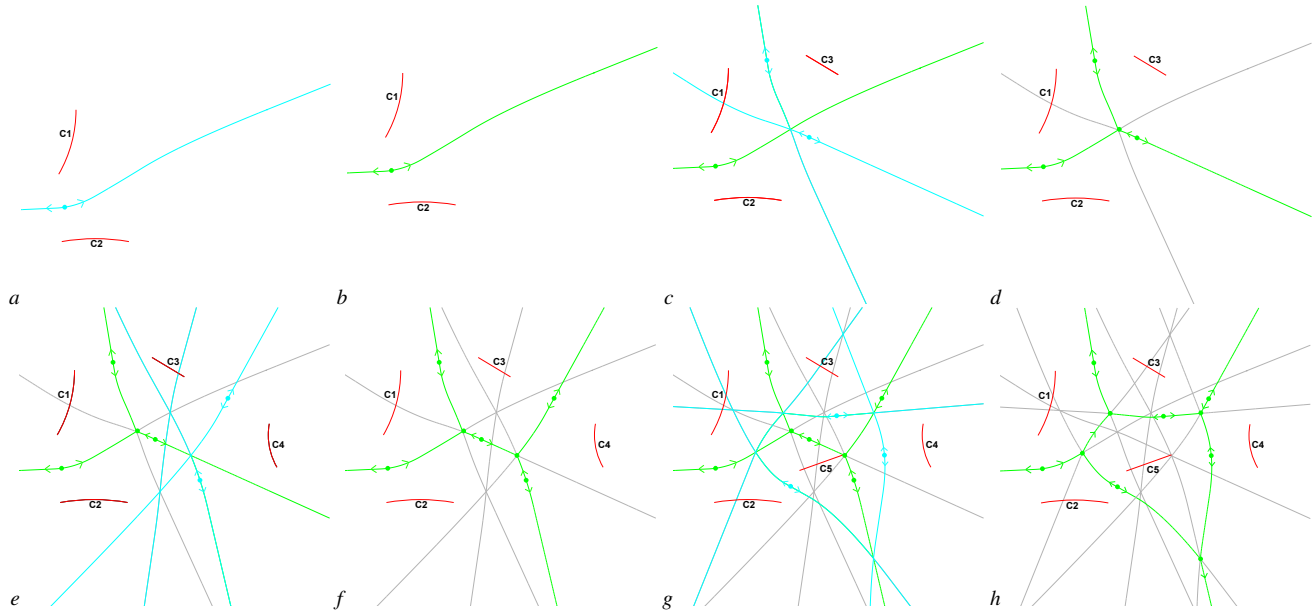

Figure 2. This figure illustrates the incremental steps of shock computation for a set of red input curves. At each step infinite length cyan bisector curves between a new boundary element and each of the previous boundary elements are computed. The finite true bisectors after the limiting process are shown in green with the removed shock branches shown in gray. The green shock links represent the valid portion of bisector curves.

in scale. Since there are no other interacting shock paths, the entire shock path is validated and shown in green. Next, suppose that a third contour fragment C_3 is added to the pool of curve fragments. The interaction of the new-comer C_3 with C_1 and C_2 generates two potential shock paths shown in cyan in (c). The intersection of these shock paths with existing shock paths create shock junctions, which are the only potential terminators of a shock path. In this particular case there is only one shock junction, with each of the three shock paths flowing into the shock junction. Since the continuation of each shock path has a time exceeding the time of the waves reaching there from another source, all three shock paths are terminated, as shown in (d). In other cases it can happen that two shock branches flow into a junction and one shock branch flows out. In this case the two branches flowing into the junction are terminated in which the shock branch flowing out only begins at the junction as its source. It can also happen that the entire shock path is invalid, say if the two sources on the opposite sides of an image with many sources in between. Consider now N boundary sources. The process of shock computation can be implemented by first computing the shock path for a pair of boundary sources, and then considering a third, a fourth, *etc.* , Figure 2. While this can be done in any order, the most efficient order is one that considers the earliest forming shocks first.

The practical implementation of these two ideas is realized through the construction and visitation of two time-ordered lists. The first list contains the *candidate* sources that are initially pre-computed and has size N^2 and is a superset of all possible source nodes in the shock-graph. The second list is the *active* shock list, the list of shocks under propagation, and is initially empty. The algorithm proceeds by visiting the first element in the *candidate* source list and initializing a pair of child shocks (outgoing flow) where each child has a start time equivalent to the time of formation of the source node and an end time initially at infinity. This initial pair of shocks are inserted into the active elements list, and the source node is removed from the *candidate* list. Each *active* shock is then propagated to the nearest junction on it, thus forming a valid shock link of the shock graph, with a specific start and a specific end-time, and this active shock is now removed from the list. The process is iterated by visiting the next active shock in the list. As the algorithm alternates between visiting the two lists it prioritizes visiting *active* shock elements first over candidate sources. If there are no more active shocks the algorithm will visit the next element in the candidate source list, initialize a new pair of child shocks, and subsequently propagate those active shocks to determine junctions with the existing shocks. This process will terminate when both lists are empty. As candidate sources are visited the algorithm determines which sources are valid or invalid given the current state of the simulation, before initializing a new pair of child shocks. It is formally shown in [6] that the number of discarded candidate sources in the second stage (propagation of shocks) leads to a logarithmic dependency on the number of contours, N, resulting in an overall run-

time of $N^2 log(N)$. Note that during this process, shocks, shock nodes, and shock links are augmented with continuous attributes such as its intrinsic geometry (time of formation, length, curvature, *etc*.) and also labeled with discrete attributes signifying the *node type*, namely, source, sink, or junction and *link type*, namely, degenerate, semi-degenerate, or degenerate [3].

Finally, there are two practical considerations in utilizing the shock computation algorithm of [6]. First, many shock paths extend infinitely beyond the image borders. The addition of a bounding box to the initial set of contour fragments contains the shock paths to a finite, well-defined area. In practice, this bounding box is twice the size of the image, Figure 3. Second, the approximation of each contour fragment as a polyline generates numerous artifactual shocks at the points of convex discontinuity. The regularization algorithm in [6] recovers the "true" shock graph, Figure 3, *i.e*., the shock graph of the contour before the polyline approximation, by scoring and removing shock links according to some user-defined threshold. The saliency computed for each shock link measures the amount of deformation of the boundary that is needed for the link to be removed. It is closely related to the "splice cost" in [5]. In all subsequent figures, unless otherwise noted, the shock computation is depicted with a bounding box twice the size of the image and with $\lambda = 1.0$ regularization.

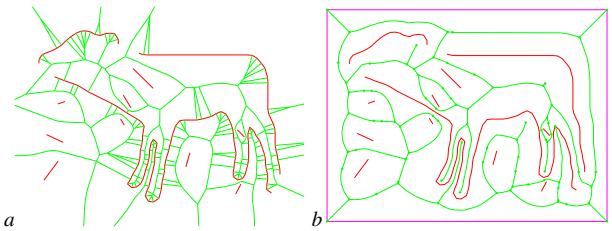

Figure 3. a) The set of red contours and its corresponding shock graph in green. b) If we augment the contour set with a magenta bounding box and apply regularization ($\lambda = 1.0$) we recover the shock graph corresponding to the initial contour, before the polyline approximation.

2. Construction of ColoredMNIST

The ColoredMNIST [2, 4] is constructed as follows: First, take the MNIST dataset and label each image with a 0 if it corresponds to a digit less than 5, and a 1 if it corresponds to a digit greater than 5. Flip these labels with probability 0.25. This ensures that there is "at most" a correlation of 0.75 between the shape of the digit and the actual label. Now we introduce color. If the label is 0, color it red, otherwise color it green. With probability ρ, flip the color of the image. This ensures that there is a $1 - \rho$ correlation between the color of the image and the true label. We use different values of ρ to create different domains, and explore how well models trained on multiple domains generalize to the others. Following [2], we use $\rho = 0.1$, 0.2, and 0.9 for our three domains. Note that by construction, color is more highly correlated with the label than shape.

References

[1] http://vxl.sourceforge.net/. 1

[2] Martin Arjovsky, Lon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020. 3

[3] Peter. J. Giblin and Benjamin. B. Kimia. On the local form and transitions of symmetry sets, medial axes, and shocks. *International Journal of Computer Vision*, 54(Issue 1-3):143–157, August 2003. 3

[4] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization, 2020. 3

[5] Thomas Sebastian, Philip Klein, and Benjamin Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26:551–571, May 2004. 3

[6] Amir Tamrakar and Benjamin B. Kimia. Intrinsic shock representation and shock computation from points, lines and circular arcs in an eulerian-lagrangian framework. Technical Report LEMS-199, Laboratory for Engineering Man/Machine Systems, Brown University, 2008. 2, 3