CUDA for Generalizations of Earth Mover's Distance

Ernest K. Ryu

Joint work with W. Li, Y. Chen, P. Yin, W. Gangbo, W. Yin, and S. Osher

January 23, 2018

Outline

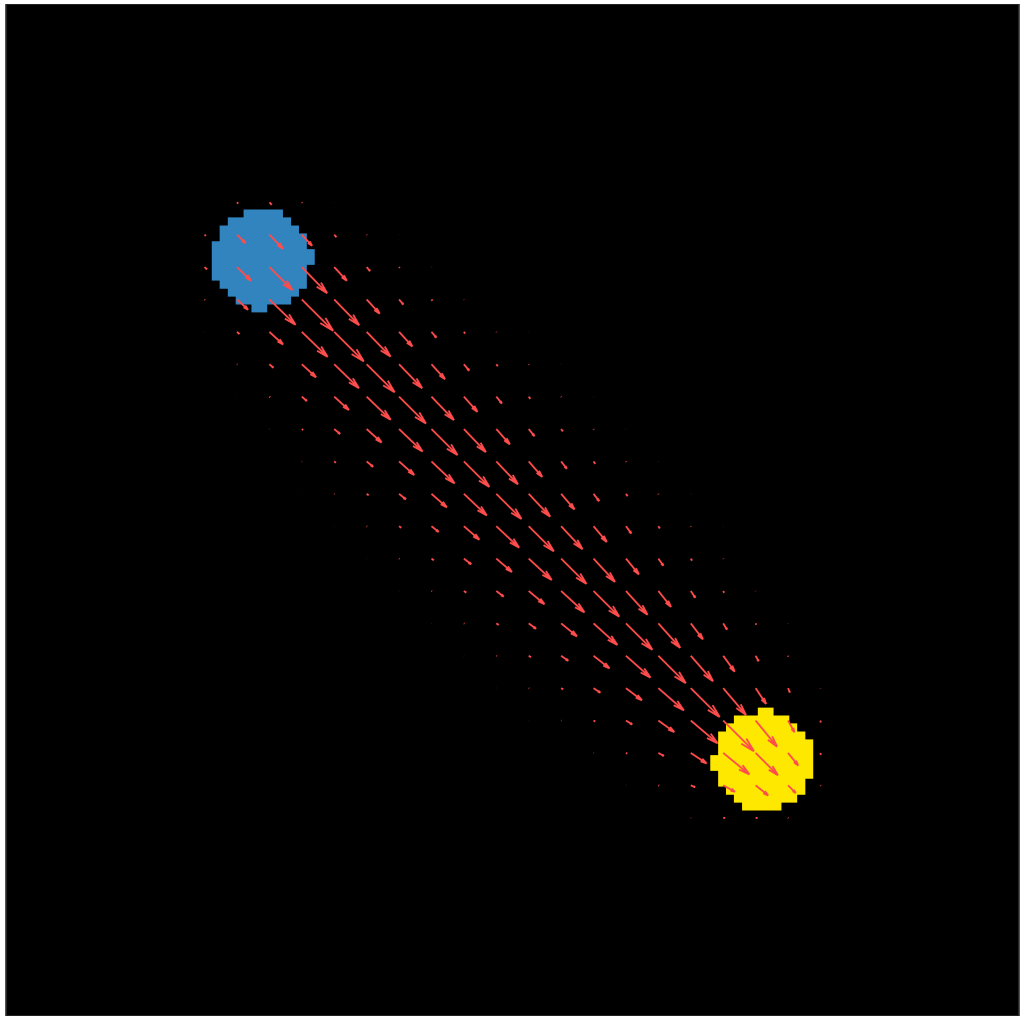
Earth mover's distance

Earth mover's distance (EMD) is

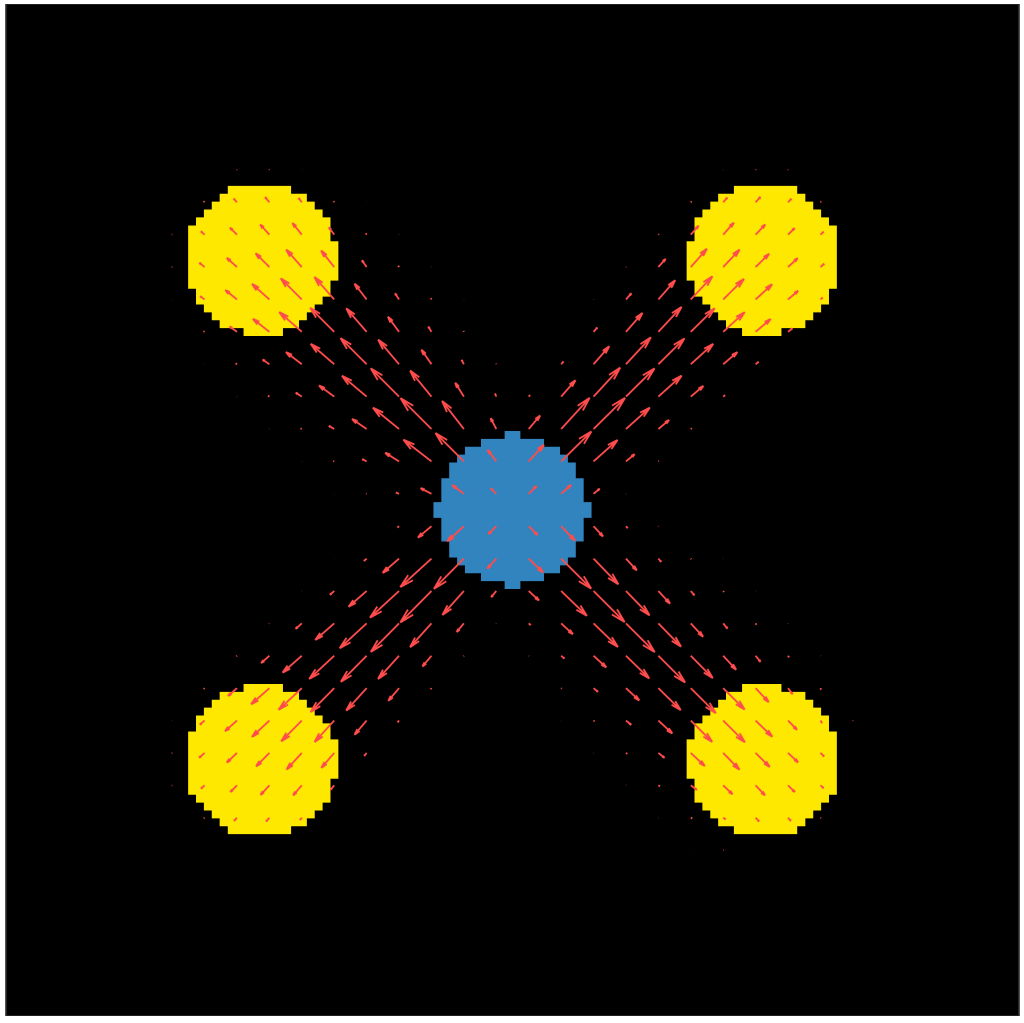
$$W(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & \int_\Omega \|\mathbf{m}(\mathbf{x})\|_2 \, d\mathbf{x} \\ \text{subject to} & \operatorname{div}(\mathbf{m}(\mathbf{x})) + \rho^1(\mathbf{x}) - \rho^0(\mathbf{x}) = 0 \\ & \mathbf{m}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = 0, \quad \begin{array}{l} \mathbf{x} \in \partial\Omega, \\ \mathbf{n}(\mathbf{x}) \text{ normal to } \partial\Omega \end{array} \end{array} \right)$$

Intepretation: Optimally move a pile of sand in ρ^0 into ρ^1. The cost per grain is the distance moved.

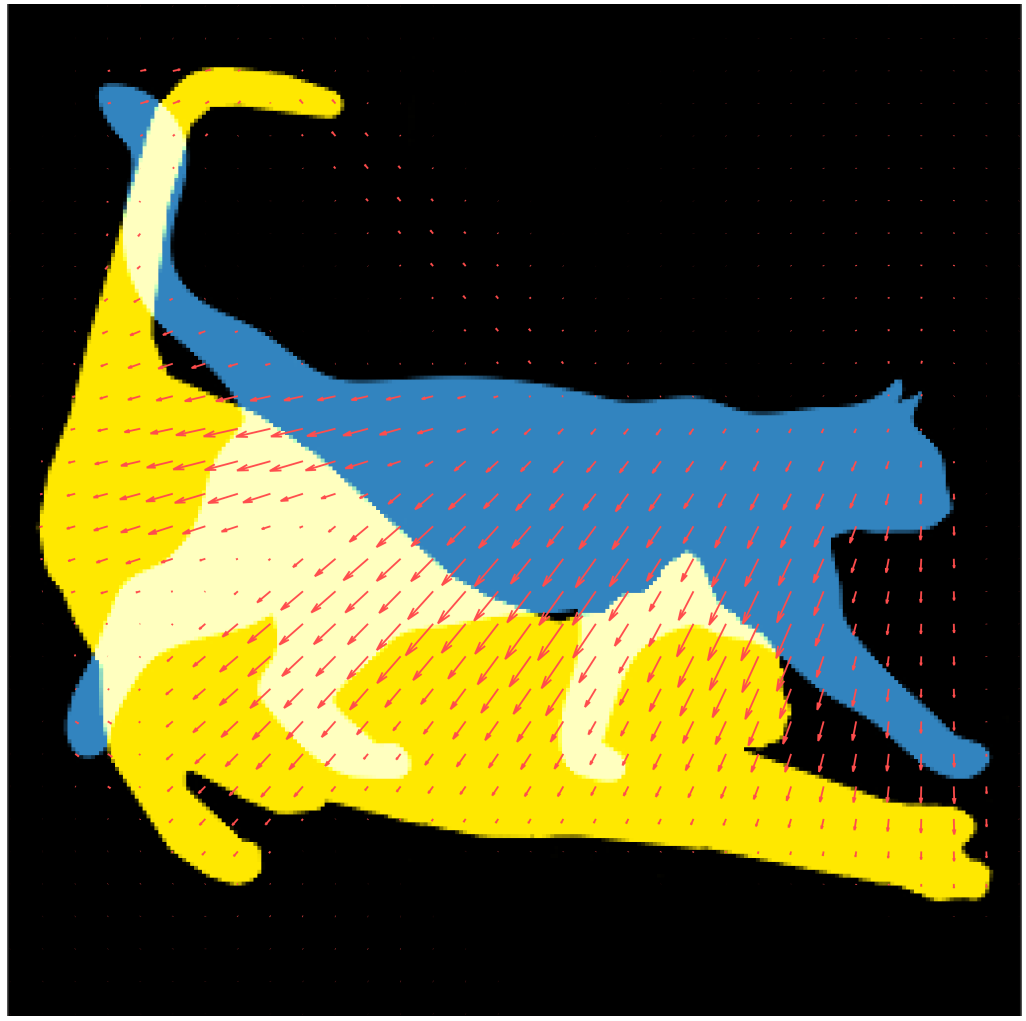
Example: EMD



Example: EMD



Example: EMD



Unbalanced EMD

Assume ρ^0 has less mass than ρ^1, i.e.,

$$\int_\Omega \rho^0(\mathbf{x})\, d\mathbf{x} \leq \int_\Omega \rho^1(\mathbf{x})\, d\mathbf{x}$$

Unbalanced EMD is

$$U(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & W(\rho^0, \tilde{\rho}^1) \\ \text{subject to} & 0 \leq \tilde{\rho}^1(\mathbf{x}) \leq \rho^1(\mathbf{x}) \\ & \int_\Omega \rho^0(\mathbf{x})\, d\mathbf{x} = \int_\Omega \tilde{\rho}^1(\mathbf{x})\, d\mathbf{x} \end{array} \right)$$

Intepretation: Optimally move all of the mass in ρ^0 to match part of the mass of ρ^1.

Example: Unbalanced EMD

Plot of flux. $\int \rho^0 = 1$ and $\int \rho^1 = 0.8$. 128×128 image.

Example: Unbalanced EMD

Plot of $\tilde{\rho}^0$. (The ink spill on the top-left is ignored.)

Partial EMD

Assume we wish to move just γ units of mass

$$0 < \gamma \leq \min \left\{ \int_\Omega \rho^0(\mathbf{x})\, d\mathbf{x}, \int_\Omega \rho^1(\mathbf{x})\, d\mathbf{x} \right\}$$

Partial EMD is

$$P(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & W(\tilde{\rho}^0, \tilde{\rho}^1) \\ \text{subject to} & 0 \leq \tilde{\rho}^0(\mathbf{x}) \leq \rho^0(\mathbf{x}) \\ & 0 \leq \tilde{\rho}^1(\mathbf{x}) \leq \rho^1(\mathbf{x}) \\ & \gamma = \int_\Omega \tilde{\rho}^0(\mathbf{x})\, d\mathbf{x} = \int_\Omega \tilde{\rho}^1(\mathbf{x})\, d\mathbf{x} \end{array} \right)$$

Intepretation: Optimally move part of the mass in ρ^0 to match part of the mass of ρ^1.

Example: Partial EMD

Plot of flux with $\gamma = 20\%$. 256×256 image.

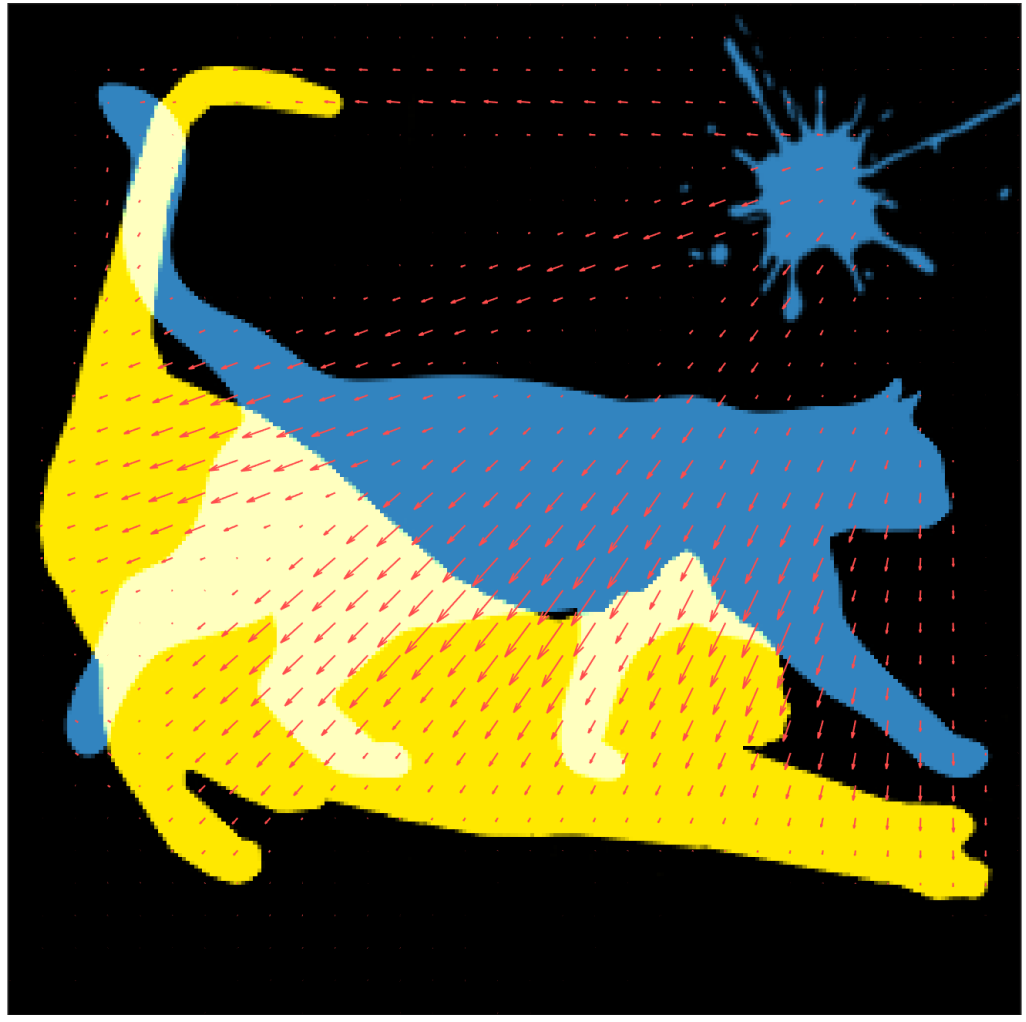
Example: Partial EMD

Plot of flux with $\gamma = 50\%$. 256×256 image.

Example: Partial EMD

Plot of flux with $\gamma = 80\%$. 256×256 image.

Example: Partial EMD

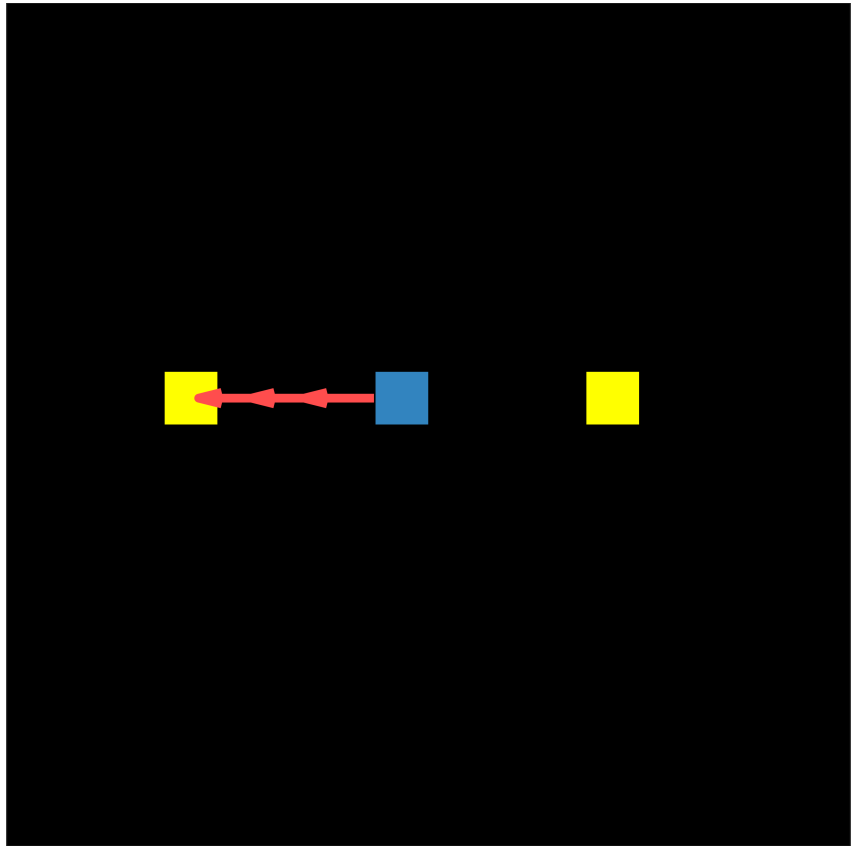


Plot of flux with $\gamma = 100\%$. 256×256 image.

Theory

- Problem generality: EMD $\subset$ Unbalanced EMD $\subset$ Partial EMD.
- $W(\rho^0, \rho^1)$ is a metric. $U(\rho^0, \rho^1)$ and $P(\rho^0, \rho^1)$ are not metrics.
- Solution exists.
- Solution not unique for unbalanced and partial EMD.
- All problems are convex optimization problems.

Why not unique?



Unbalanced EMD with ρ^0 (blue) of mass 0.5 and ρ^1 (yellow) of mass 1. This solution is not unique.

Nested optimization problem

Basic technique: Collapse a nested optimization into one.

Into

$$U(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & W(\rho^0, \tilde{\rho}^1) \\ \text{subject to} & 0 \leq \tilde{\rho}^1(\mathbf{x}) \leq \rho^1(\mathbf{x}) \\ & \int_\Omega \rho^0(\mathbf{x})\, d\mathbf{x} = \int_\Omega \tilde{\rho}^1(\mathbf{x})\, d\mathbf{x} \end{array} \right)$$

substitute

$$W(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & \int_\Omega \|\mathbf{m}(\mathbf{x})\|_2\, d\mathbf{x} \\ \text{subject to} & \operatorname{div}(\mathbf{m}(\mathbf{x})) + \rho^1(\mathbf{x}) - \rho^0(\mathbf{x}) = 0 \\ & \mathbf{m} \text{ satisfies b.c.} \end{array} \right)$$

Nested optimization problem

and we get

$$U(\rho^0, \rho^1) = \left(\begin{array}{ll} \text{minimize} & \|\mathbf{m}(\mathbf{x})\|_2 \, d\mathbf{x} \\ \text{subject to} & \operatorname{div}(\mathbf{m}(\mathbf{x})) + \tilde{\rho}(\mathbf{x}) - \rho^0(\mathbf{x}) = 0 \\ & \mathbf{m} \text{ satisfies b.c.} \\ & 0 \leq \tilde{\rho}^1(\mathbf{x}) \leq \rho^1(\mathbf{x}) \\ & \int_\Omega \rho^0(\mathbf{x}) \, d\mathbf{x} = \int_\Omega \tilde{\rho}^1(\mathbf{x}) \, d\mathbf{x} \end{array} \right)$$

The same works for the partial EMD.

Outline

Ideal parallel computer

An ideal parallel computer allows:

- Entirely independent threads.
- All threads can talk to each other. (No thread hierarchy.)
- Threads freely access shared memory.
- Each thread has faster local memory.

Shared memory machines, i.e., machines with multiple CPU cores and lots of memory, are almost like this.

Restrictions on CUDA

However, CUDA GPU has some limitations.

- 32 threads are grouped and run as a *warp*. A single warp must "do the same thing".
- Threads (warps) form a block. Threads within the same block can communicate. Threads in different blocks must communicate through the CPU.
- A warp (but not its individual threads) freely access shared memory.
- Each thread has faster local memory.

If you ignore these hardware limitations, you code will be correct but slow.

For example, you must consider the thread and memory architecture to effectively compute a vector inner product.

CUDA operates in SIMT

Not all parallel things are appropriate for CUDA.

CUDA operates in a Single instruction multiple data (SIMD) or Single instruction multiple thread (SIMT) fashion: A group of threads (a warp) can do only one thing at a time.

- If thread 1 and 2 needs to do the same thing with different data, they can work in parallel.
- If thread 1 and 2 needs to do different things, thread 2 needs to wait for thread 1 to finish and then thread 1 needs to wait for thread 2 to finish.

PDHG method

To solve

$$\underset{x}{\text{minimize}} \quad f(x) + g(Lx)$$

we use PDHG

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left\{ f(x) + \langle u^k, Lx \rangle + \frac{1}{2\mu} \|x - x^k\|_2^2 \right\}$$

$$u^{k+1} = \underset{u}{\operatorname{argmax}} \left\{ -g^*(u) + \langle u, L(2x^{k+1} - x^k) \rangle - \frac{1}{2\nu} \|u - u^k\|_2^2 \right\}$$

(A.K.A. Chambolle-Pock)

M. Zhu and T. Chan, UCLA CAM Report, 2008.
E. Esser, X. Zhang, and T. F. Chan, SIAM J. Imaging Sci., 2010.
A. Chambolle and T. Pock, J. Math. Imaging Vis., 2011.
T. Pock and A. Chambolle, IEEE Intern. Conf. Comput. Vis., 2011.

PDHG for EMD

Applying PDHG to (the discretization of) EMD

$$\begin{array}{ll} \underset{\mathbf{m}}{\text{minimize}} & \sum \|\mathbf{m}_{ij}\|_2 \\ \text{subject to} & \operatorname{div}(\mathbf{m}) + \rho^1(\mathbf{x}) - \rho^0(\mathbf{x}) = 0 \end{array}$$

we get

$$\begin{aligned} \mathbf{m}_{ij}^{k+1} &= \operatorname{shrink}(\mathbf{m}_{ij}^k + \mu(\nabla\Phi^k)_{ij}, \mu) \\ \Phi_{ij}^{k+1} &= \Phi_{ij}^k + \tau((\operatorname{div}(2\mathbf{m}^{k+1} - \mathbf{m}^k))_{ij} + \rho_{ij}^1 - \rho_{ij}^0) \end{aligned}$$

Algorithmic structure of primal-dual EMD

```
m_temp[i,j] = m[i,j]
m[i,j] = shrink(m[i,j]+mu/dx*(Phi[i+1,j]+Phi[i,j+1]-2Phi[i,j]))
m_temp[i,j] = 2*m[i,j]-m_temp[i,j]
-----------------------------------------------------------
Synchronize over all i,j
-----------------------------------------------------------
divm[i,j] = m_temp_x[i,j]-m_temp_x[i-1,j]
           +m_temp_y[i,j]-m_temp_y[i,j-1]
Phi[i,j] = Phi[i,j] + tau*(divm[i,j]/dx+rho1[i,j]-rho0[i,j]);
-----------------------------------------------------------
Synchronize over all i,j
-----------------------------------------------------------
```

Algorithmic structure of primal-dual EMD

This algorithmic structure is great for CUDA.

1. Computation splits pixel-by-pixel.
2. All threads do exactly the same thing with different data, except at the boundary. (Minimal branch divergence)
3. A group of threads, a warp, accesses a block of consecutive memory. (Coalesced memory access)

Advantage 1 is inherent to PDHG. Advantages 2 and 3 is due to the regularity of the linear operators ∇ and div.

Compared to sequential CPU code, GPU code is 100x faster.

PDHG for partial EMD

Cast

$$\begin{array}{ll} \underset{\mathbf{m},\tilde{\rho}^0,\tilde{\rho}^1}{\text{minimize}} & \sum_{ij} \|\mathbf{m}_{ij}\|_2 \\ \text{subject to} & \operatorname{div}(\mathbf{m}) = \tilde{\rho}^0 - \tilde{\rho}^1 \\ & 0 \le \tilde{\rho}^0 \le \rho^0 \\ & 0 \le \tilde{\rho}^1 \le \rho^1 \\ & \gamma = \langle \mathbf{1}, \tilde{\rho}^0 \rangle = \langle \mathbf{1}, \tilde{\rho}^1 \rangle \end{array}$$

into the standard PDHG form

$$\underset{x}{\text{minimize}} \quad f(x) + g(Lx)$$

PDHG for partial EMD

as

$$\text{minimize} \quad \underbrace{\sum_{ij} \|\mathbf{m}_{ij}\|_2 + \delta_{S(\rho^0,\gamma)}(\tilde{\rho}^0) + \delta_{S(\rho^1,\gamma)}(\tilde{\rho}^1)}_{=f(x)} + \underbrace{\delta_{\{0\}}(\operatorname{div}(\mathbf{m}) - \tilde{\rho}^0 + \tilde{\rho}^1)}_{=g(Lx)}$$

with the (convex) constraint set

$$S(\rho,\gamma) = \{\tilde{\rho} \in \mathbb{R}^{n\times n} \,|\, 0 \le \tilde{\rho} \le \rho,\, \langle \mathbf{1}, \rho\rangle = \gamma\}$$

PDHG for partial EMD

PDHG applied to the partial EMD:

$$\mathbf{m}_{ij}^{k+1} = \mathrm{shrink}_2(\mathbf{m}_{ij}^k + \mu(\nabla\Phi^k)_{ij}, \mu)$$
$$\tilde{\rho}^{0,k+1} = P_{S(\rho^0,\gamma)}(\tilde{\rho}^{0,k} + \nu\Phi^k)$$
$$\tilde{\rho}^{1,k+1} = P_{S(\rho^1,\gamma)}(\tilde{\rho}^{1,k} - \nu\Phi^k)$$
$$\Phi_{ij}^{k+1} = \Phi_{ij}^k + \tau(\mathrm{div}(2\mathbf{m}^{k+1} - \mathbf{m}^k)_{ij} + 2\tilde{\rho}_{ij}^{1,k+1} - \tilde{\rho}_{ij}^{1,k} - 2\tilde{\rho}_{ij}^{0,k+1} + \tilde{\rho}_{ij}^{0,k})$$

$\mathbf{m}$ and Φ-updates are efficient on CUDA. What about the projection?

Parallel computing primitive: Reduce

Given an array

$$\mathbf{s} = (s_1, s_2, \ldots, s_n)$$

and an associative binary operation $\bullet$, reduce is

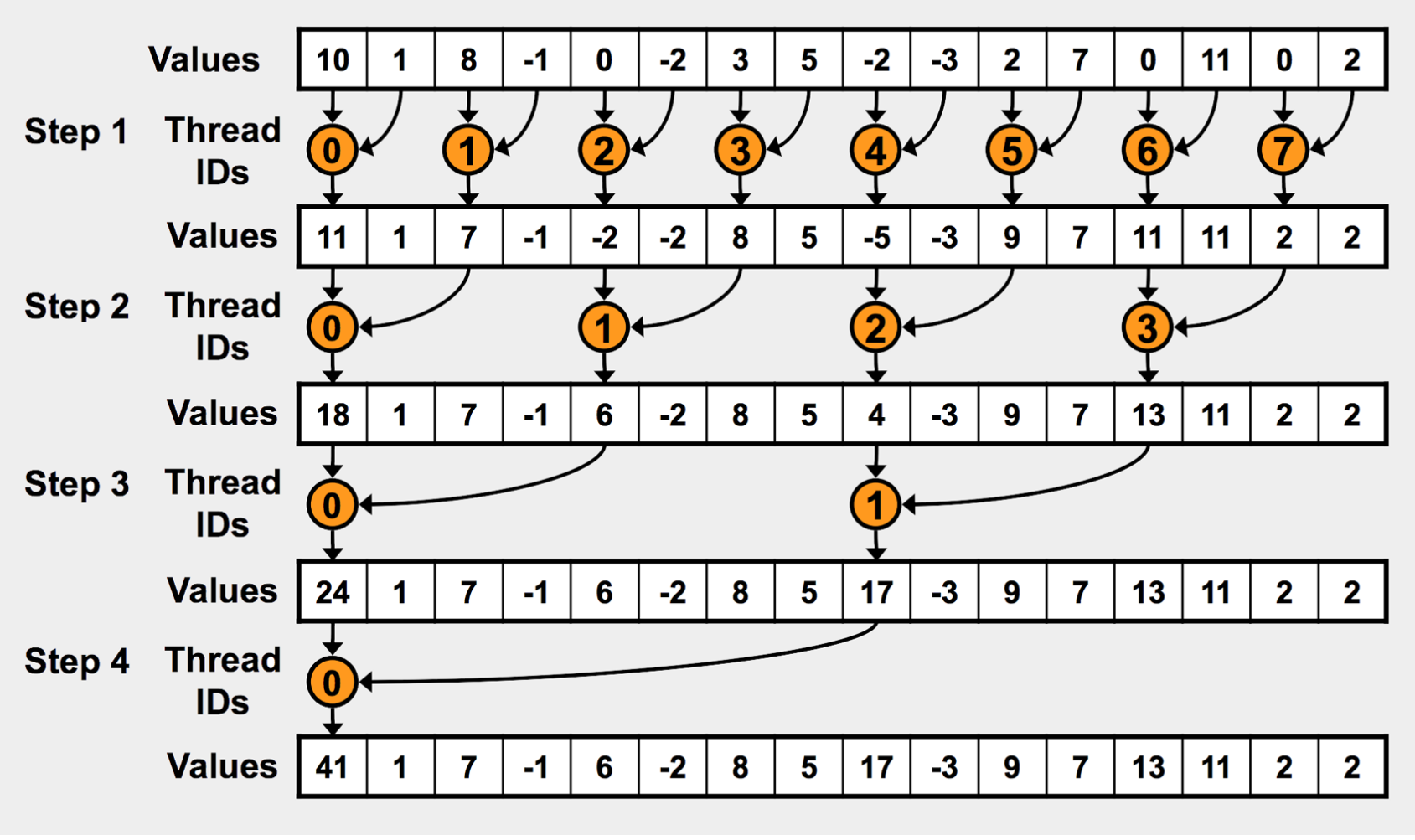
$$\mathrm{Reduce}(\mathbf{s}, \bullet) = s_1 \bullet s_2 \bullet \cdots \bullet s_n.$$

Examples of $\bullet$: scalar max, addition, and multiplication.

In parallel computing and in CUDA, reduce takes $\log n$ steps.

Parallel computing primitive: Reduce

Reduce is efficient on CUDA.



Projection algorithm

Theorem

The projection has a semi-closed-form solution:

$$P_{S(\rho,\gamma)}(\sigma) = \tilde{\rho}(\theta)$$

where $\tilde{\rho}(\theta) = \min\{\max\{\sigma - \theta\mathbf{1}, 0\}, \rho\}$. *The parameter* θ *satisfies* $\langle \mathbf{1}, \tilde{\rho}(\theta) \rangle = \gamma$, *and we can compute it with bisection since* $\langle \mathbf{1}, \tilde{\rho}(\theta) \rangle$ *is a non-increasing function.*

Projection algorithm

Initialize $\theta_{\min}$ and $\theta_{\max}$
//Perform bisection
while $\theta_{\max} - \theta_{\min} > \varepsilon$
    $\theta_{\text{mid}} = (\theta_{\max} + \theta_{\min})/2$
    if $\gamma < \langle \mathbf{1}, \tilde{\rho}(\theta_{\text{mid}}) \rangle$ //Reduction with CUDA
        $\theta_{\min} = \theta_{\text{mid}}$
    else
        $\theta_{\max} = \theta_{\text{mid}}$
    end
end
$P_{S(\rho,\gamma)}(\sigma) = \min\{\max\{\sigma - \theta_{\text{mid}}\mathbf{1}, 0\}, \rho\}$

Computation cost

(a) EMD
256×256, 10s

(b) Unbalanced EMD
128×128, 80s

(c) Partial EMD
256×256, 300s

Experiments on Titan Xp GPU.

Conclusion

EMD, unbalanced EMD, and partial EMD are tools of applied mathematics with rich theory and interesting applications.

With GPU acceleration, these are computationally feasible.

The code is available as a "mex"ed Matlab function.

Outline

Vector EMD

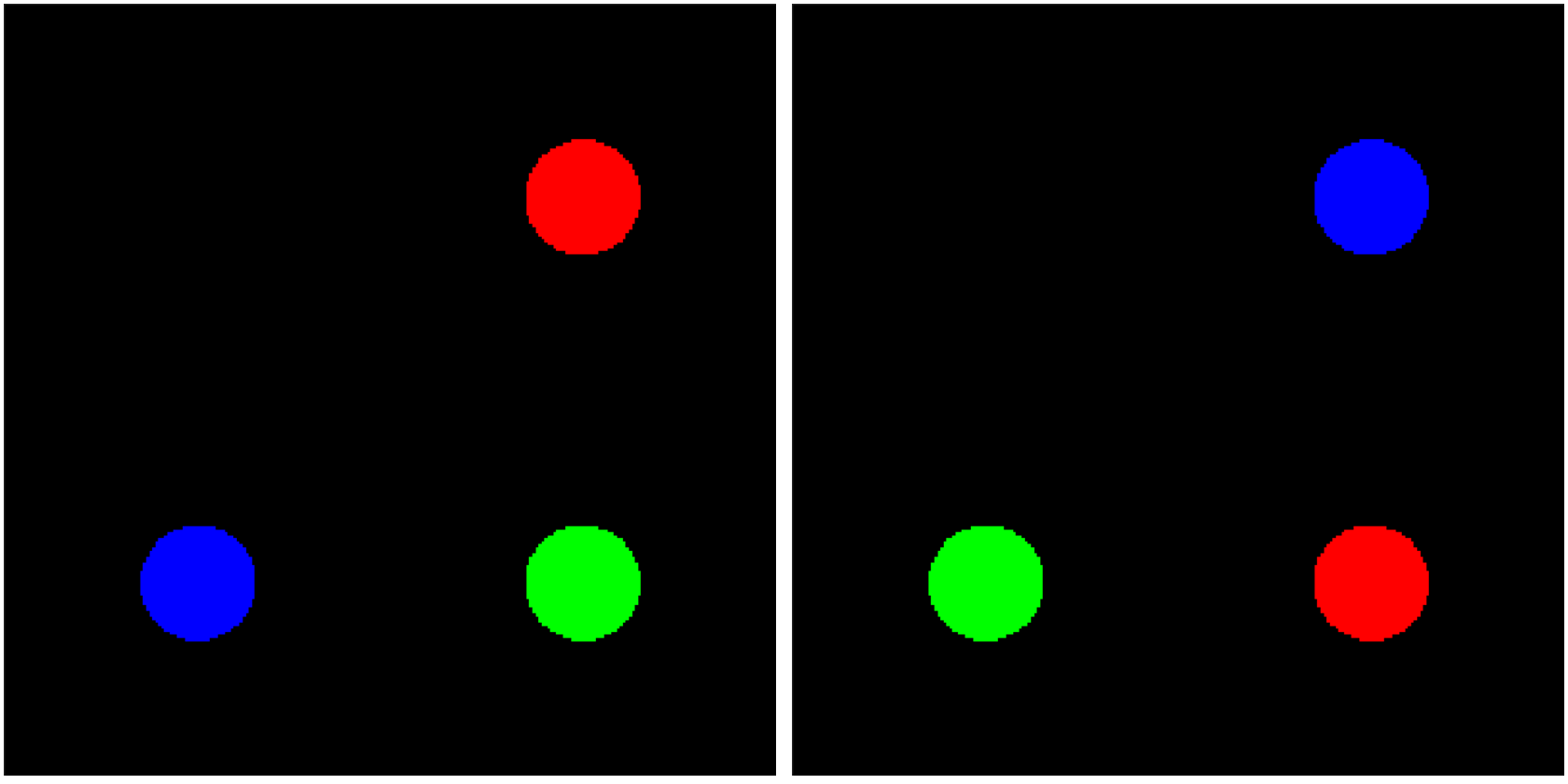
Sometimes, data has more than one piece of information associated with each spatial coordinate.

We generalize EMD to vector-valued densities

$$V(\vec{\rho}^0, \vec{\rho}^1) = \left(\begin{array}{ll} \underset{\vec{\mathbf{u}}, \vec{w}}{\text{minimize}} & \int_\Omega \|\vec{\mathbf{u}}(\mathbf{x})\|_u + \alpha \|\vec{w}(\mathbf{x})\|_w \, d\mathbf{x} \\ \text{subject to} & \operatorname{div}_{\mathbf{x}}(\vec{\mathbf{u}})(\mathbf{x}) + \operatorname{div}_{\mathcal{G}}(\vec{w}(\mathbf{x})) = \vec{\rho}^0(\mathbf{x}) - \vec{\rho}^1(\mathbf{x}) \\ & \vec{\mathbf{u}} \text{ satisfies zero-flux b.c.} \end{array} \right)$$

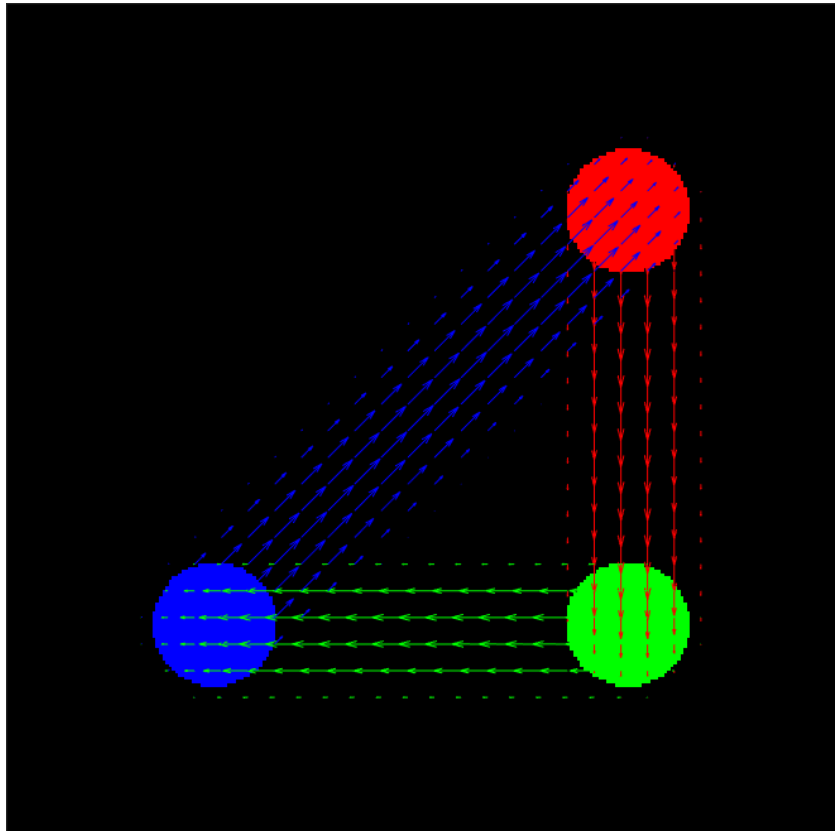
$\vec{\mathbf{u}}$ represents spatially transporting mass.
$\vec{w}$ represents changing channels (color).

Example: vector EMD



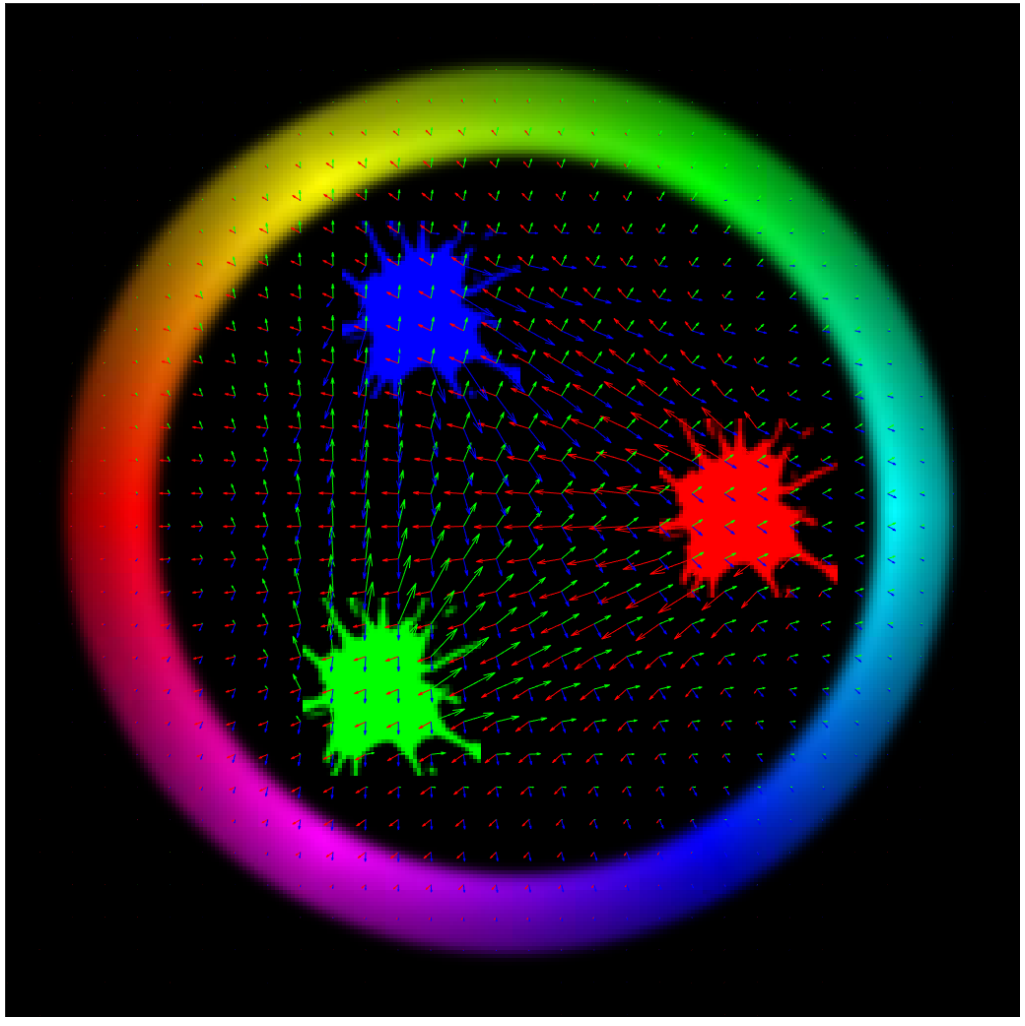
(a) $\vec{\rho}^0$ (b) $\vec{\rho}^1$

Example: vector EMD



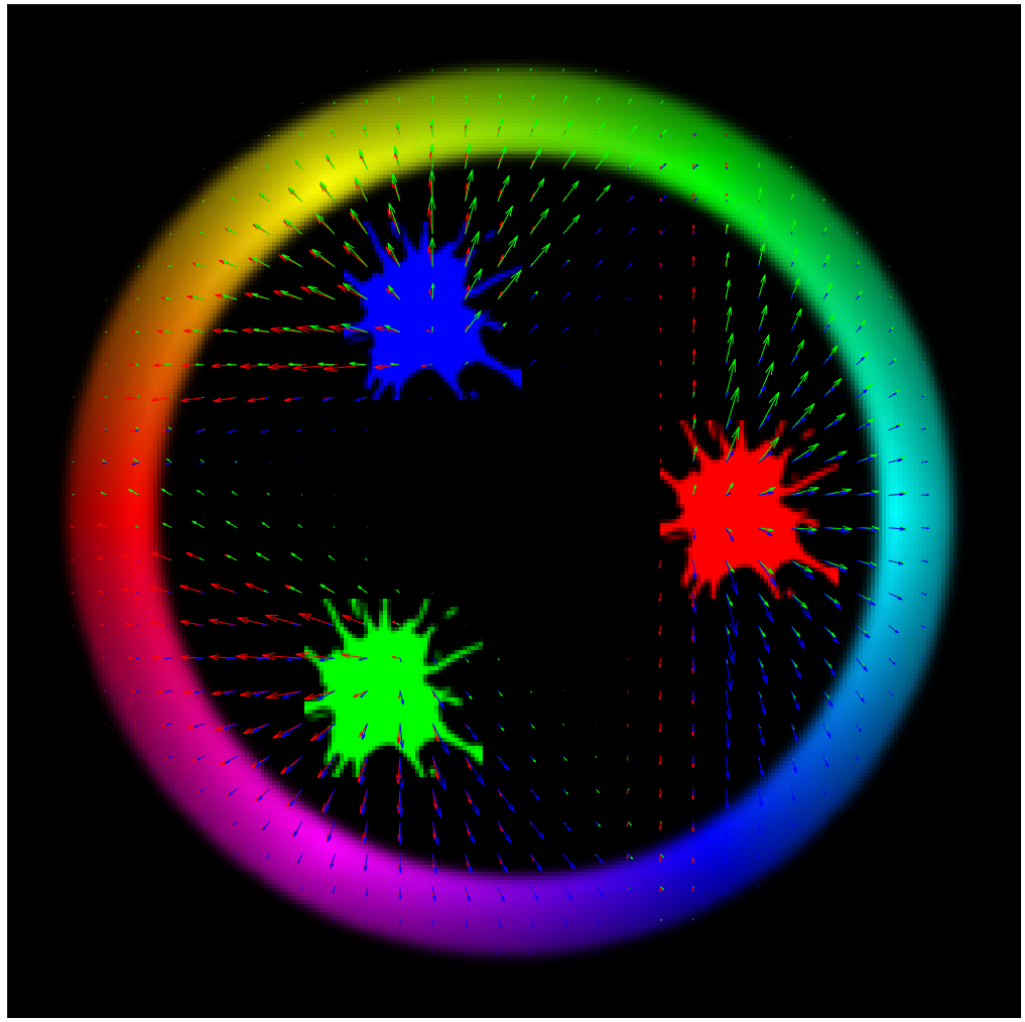
(c) Plot of flux.

Example: vector EMD



(a) $\alpha = 10$, moving mass is optimal.

Example: vector EMD



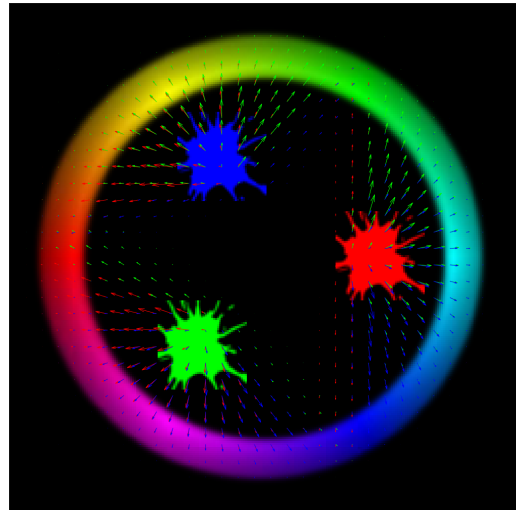
(b) $\alpha = 0.1$, changing color is optimal.

CUDA for Vector EMD

PDHG for vector EMD is analogous to PDHG for scalar EMD.

For the same reasons, vector EMD can utilize a CUDA GPU very effectively.

Computation cost



(a) Vector EMD
256×256, 20s

Experiment on Titan Xp GPU.

Outline

Dual problem

Vector EMD has the following dual problem.

$$V(\vec{\rho}^0, \vec{\rho}^1) = \left(\begin{array}{ll} \underset{\vec{\phi}}{\text{maximize}} & \int_\Omega \langle \vec{\phi}(\mathbf{x}), \vec{\rho}^1(\mathbf{x}) - \vec{\rho}^0(\mathbf{x}) \rangle \, d\mathbf{x} \\ \text{subject to} & \|\nabla_{\mathbf{x}} \vec{\phi}(\mathbf{x})\|_{u*} \leq 1 \\ & \|\nabla_{\mathcal{G}} \vec{\phi}(\mathbf{x})\|_{w*} \leq \alpha \quad \text{for all } \mathbf{x} \in \Omega \end{array} \right)$$

The optimization variable $\vec{\phi} : \Omega \to \mathbb{R}^k$ is a function.

Theoretical properties

1. $V(\vec{\rho}^0, \vec{\rho}^1)$ is a metric.
2. Solution exists.
3. Dual solution exists.
4. Strong duality holds.

1 and 3 are relatively simple to verify.

2 and 4 requires more work. Standard techniques for proving scalar EMD duality do not simply apply to our setup, since we do not have a LP formulation.

Fenchel-Rockafellar duality

$L : X \to Y$ continuous linear map between locally convex topological vector spaces X and Y.
$f : X \to \mathbb{R} \cup \{\infty\}$ and $g : Y \to \mathbb{R} \cup \{\infty\}$ lower-semicontinuous convex functions.

$$d^\star = \sup_{x \in X} \{-f(x) - g(Lx)\} \qquad p^\star = \inf_{y^* \in Y^*} \{f^*(L^* y^*) + g^*(-y^*)\}$$

Theorem

If there is an $x \in X$ such that $f(x) < \infty$ and g is bounded above in a neighborhood of Lx, then $p^\star = d^\star$. Furthermore, if $p^\star = d^\star < \infty$, the infimum of $\inf_{y^* \in Y^*} \{f^*(L^* y^*) + g^*(-y^*)\}$ is attained.

Duality proof sketch

Define $\nabla_{\mathbf{x}} : C^1 \to C$. This makes $\nabla_{\mathbf{x}}$ a bounded linear operator.

Define the dual (adjoint) operator $\nabla^*_{\mathbf{x}} : \mathcal{M} \to (C^1)^*$

Duality proof sketch

We formalize the dual problem as

$$\begin{aligned} \underset{\vec{\phi} \in C^1}{\text{maximize}} \quad & \int_\Omega \langle \vec{\phi}(\mathbf{x}), \vec{\rho}^1(\mathbf{x}) - \vec{\rho}^0(\mathbf{x}) \rangle \, d\mathbf{x} \\ \text{subject to} \quad & \|\nabla_{\mathbf{x}} \vec{\phi}(\mathbf{x})\|_{u*} \leq 1 \\ & \|\nabla_{\mathcal{G}} \vec{\phi}(\mathbf{x})\|_{w*} \leq \alpha \quad \text{for all } \mathbf{x} \in \Omega, \end{aligned}$$

Assumptions of the theorem are met.

Duality proof sketch

The Fenchel-Rockafellar dual is

$$\begin{array}{ll} \underset{\vec{\mathbf{u}}\in\mathcal{M},\,\vec{w}\in\mathcal{M}}{\text{minimize}} & \int_\Omega \|\vec{\mathbf{u}}(\mathbf{x})\|_u + \alpha\|\vec{w}(\mathbf{x})\|_w \, d\mathbf{x} \\ \text{subject to} & -\nabla^*_{\mathbf{x}}\vec{\mathbf{u}} - \nabla^*_{\mathcal{G}}\vec{w} = \vec{\rho}^0 - \vec{\rho}^1 \text{ as members of } (C^1)^*. \end{array}$$

(We view the primal problem as the dual of the dual problem, because $C^* = \mathcal{M}$ is known, but $\mathcal{M}^*$ is complicated.)

Zero-flux boundary condition

When $\vec{\mathbf{m}}$ is a smooth function,

$$\int_\Omega \langle \nabla_{\mathbf{x}} \vec{\varphi}(\mathbf{x}), \vec{\mathbf{m}}(\mathbf{x}) \rangle \, d\mathbf{x} = - \int_\Omega \langle \vec{\varphi}(\mathbf{x}), \mathrm{div}_{\mathbf{x}} \vec{\mathbf{m}}(\mathbf{x}) \rangle \, d\mathbf{x}$$

$\forall$ smooth $\vec{\varphi}$ if and only if $\vec{\mathbf{m}}$ satisfies the zero-flux b.c. I.e., $\nabla_{\mathbf{x}}^* = -\mathrm{div}_{\mathbf{x}}$ holds when the zero-flux b.c. holds.

As a generalization, $\vec{\mathbf{u}} \in \mathcal{M}$ satisfies the zero-flux b.c. in the weak sense if there is a $\vec{g} \in \mathcal{M}$ such that

$$\int_\Omega \langle \nabla_{\mathbf{x}} \vec{\phi}(\mathbf{x}), \vec{\mathbf{u}}(d\mathbf{x}) \rangle = - \int_\Omega \langle \vec{\phi}(\mathbf{x}), \vec{g}(d\mathbf{x}) \rangle$$

$\forall$ $\vec{\phi}$. In other words, $\vec{\mathbf{u}}$ satisfies the zero-flux b.c. if $\nabla_{\mathbf{x}}^* \vec{\mathbf{u}} \in \mathcal{M}$.

Duality proof sketch

With the weak definition of the boundary condition, we get

$$\begin{array}{ll} \underset{\vec{\mathbf{u}}\in\mathcal{M},\,\vec{w}\in\mathcal{M}}{\text{minimize}} & \int_\Omega \|\vec{\mathbf{u}}(\mathbf{x})\|_u + \alpha\|\vec{w}(\mathbf{x})\|_w \, d\mathbf{x} \\ \text{subject to} & -\nabla^*_{\mathbf{x}}\vec{\mathbf{u}} - \nabla^*_{\mathcal{G}}\vec{w} = \vec{\rho}^0 - \vec{\rho}^1 \text{ as members of } \mathcal{M} \\ & \vec{\mathbf{u}} \text{ satisfies zero-flux b.c} \end{array}$$

By the duality theorem, a solution exists and strong duality holds.

Conclusion

Vector EMD are tools of applied mathematics with interesting applications.

We establish the theory and provide a GPU accelerated algorithm for vector EMD.

The code is available as a "mex"ed Matlab function.