DevelSet: Deep Neural Level Set for Instant Mask Optimization

Guojin Chen, Ziyagn Yu, Hongduo Liu, Yuzhe Ma, Bei Yu

Chinese University of Hong Kong
{gjchen21,byu}@cse.cuhk.edu.hk

Nov. 1, 2021

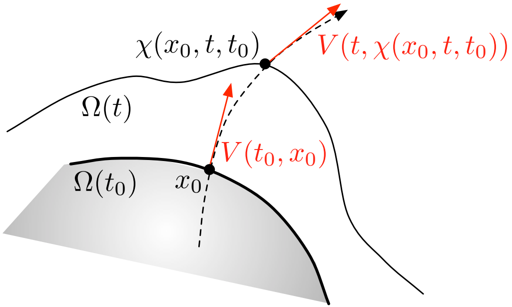
Outline

1. Level set introduction.
2. Level set for mask optimization
3. Deep level sets

Intuitive notion of an evolving domain.

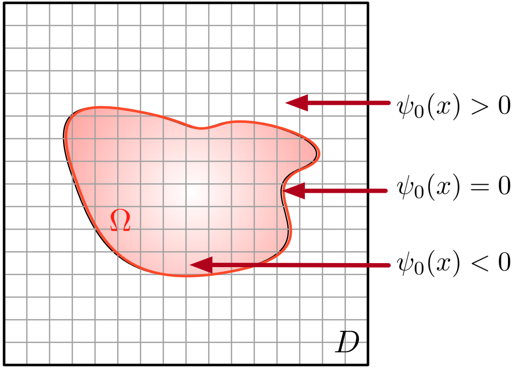
A domain $\Omega(t)$ evolves according to a velocity field $V(t,x)$ from an initial position $\Omega(t_0)$ if it is obtained by transporting its points along V :

$$\Omega(t) = \{\chi(x_0, t, t_0), x_0 \in \Omega(t_0)\}$$



- Let $\Omega(t)$ be a (smooth) domain, moving over $(0,T)$ along the (smooth) velocity field $V(t,x)$. Let $\phi(t,x)$ be a smooth Level Set function, i.e:

$$\forall t \in (0,T), x \in \mathbb{R}^d, \begin{cases} \phi(t,x) < 0 & \text{if } x \in \Omega(t) \\ \phi(t,x) = 0 & \text{if } x \in \Gamma(t) \\ \phi(t,x) > 0 & \text{if } x \in {}^c\overline{\Omega(t)} \end{cases} \tag{1}$$



Intuitive defination of an evolving domains. (II)

- Let $x_0 \in \Gamma(0)$ be fixed. By the intuitive definition of an evolving domain, it comes:

$$\forall t \in (0,T), \phi\left(t, \chi\left(x_0, t, 0\right)\right) = 0$$

- Differentiating and using the chain rule yields:

$$\frac{\partial \phi}{\partial t}\left(t, \chi\left(x_0, t, 0\right)\right) + \frac{d}{dt}\left(\chi\left(x_0, t, 0\right)\right) \cdot \nabla \phi\left(t, \chi\left(x_0, t, 0\right)\right) = 0$$

- Since this holds for any point $x_0 \in \Gamma(0)$, we obtain the Level Set advection equation:

$$\forall t \in (0,T), \forall x \in \mathbb{R}^d, \frac{\partial \phi}{\partial t} + V(t,x) \cdot \nabla \phi = 0$$

Intuitive defination of an evolving domains. (III)

- If, in addition, the velocity is consistently oriented along the normal vector $n_t(x)$ to $\Omega(t)$:

  $V(t,x) = v(t,x)\frac{\nabla\phi(t,x)}{|\nabla\phi(t,x)|}$, for some scalar $v(t,x)$

  the equation rewrites as the Level Set Hamilton-Jacobi equation:

$$\forall t \in (0,T), \forall x \in \mathbb{R}^d, \frac{\partial \phi}{\partial t} + v(t,x)|\nabla\phi| = 0$$

Level set method is all about:

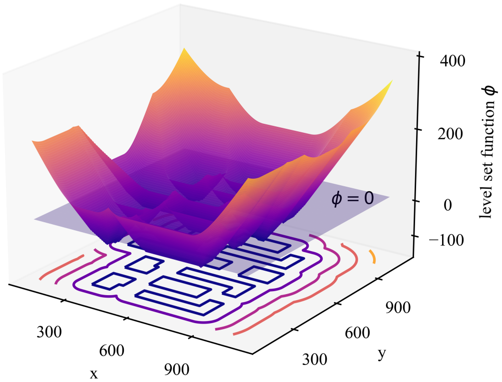
evolving a **surface**,

instead of the real **contour**.

Level set method: Explanation

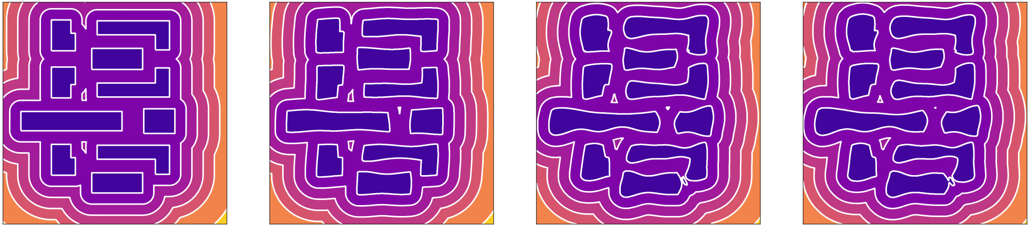
A domain $\Omega \subset \mathbb{R}^d$ is equivalently defined by a function $\phi\colon \mathbb{R}^d \to \mathbb{R}$ such that:

$$\phi(x) < 0 \quad \text{if } x \in \Omega \quad ; \quad \phi(x) = 0 \quad \text{if } x \in \Gamma \quad ; \quad \phi(x) > 0 \quad \text{if } x \in {}^c\bar{\Omega} \qquad (2)$$



Here the zero level set of the surface is a square

Merging and spliting are here handled naturally by the surface motion.

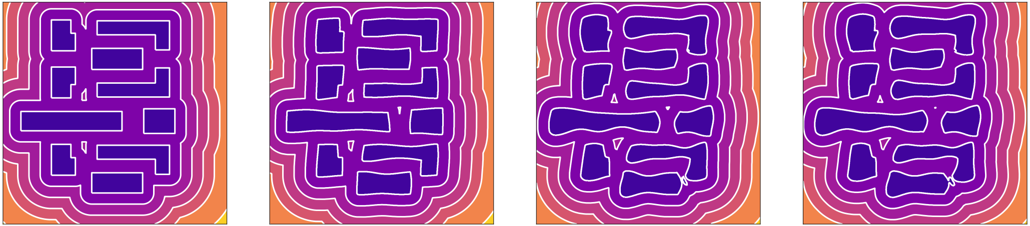


(a) ϕ_0 (b) ϕ_6 (c) ϕ_{10} (d) ϕ_T

The evolving front in red is known by taking the zero level set of a surface ϕ

Level set surface evolves

Merging and spliting are here handled naturally by the surface motion.



(a) ϕ_0 (b) ϕ_6 (c) ϕ_{10} (d) ϕ_T

The evolving front in red is known by taking the zero level set of a surface ϕ

Question

The question is now: what is the function ϕ?

Math defination

$$\phi(x(t), t) = 0$$

The question still remains: what is the function $\phi(x(t), t)$? It can actually be anything we want as long as its zero level set gives us the contour.

Given an initial ϕ at t=0, it would be possible to know ϕ at any time t with the motion equation $\frac{\partial \phi}{\partial t}$.

The chain rule gives us:

$$\begin{aligned}\frac{\partial \phi(x(t), t)}{\partial t} &= 0 \\ \frac{\partial \phi}{\partial x(t)} \frac{\partial x(t)}{\partial t} + \frac{\partial \phi}{t} \frac{t}{t} &= 0 \\ \frac{\partial \phi}{\partial x(t)} x_t + \phi_t &= 0\end{aligned} \tag{3}$$

Here, recall that $\frac{\partial \phi}{\partial x} = \nabla \phi$. Also, the speed x_t is given by a force F normal to the surface, so $\boldsymbol{x_t} = \boldsymbol{V}\,(\boldsymbol{x}(\boldsymbol{t}))\,\boldsymbol{n}$ where $n = \frac{\nabla \phi}{|\nabla \phi|}$. The previous motion equation can be rewriten with:

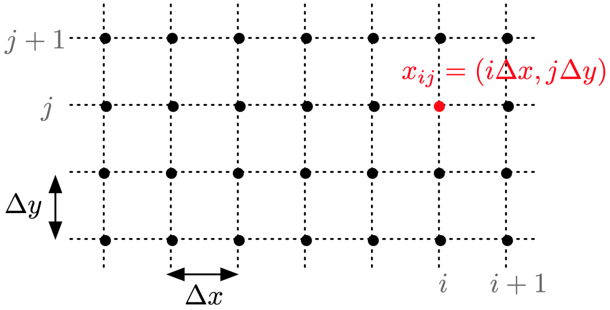
$$\begin{aligned} \phi_t + \nabla \phi x_t &= 0 \\ \phi_t + \nabla \phi F n &= 0 \\ \phi_t + V \nabla \phi \frac{\nabla \phi}{|\nabla \phi|} &= 0 \\ \phi_t + V |\nabla \phi| &= 0 \end{aligned} \tag{4}$$

Numerical setting

1. In the 2d computer world, images are pixels.
2. The time interval $(0, T)$ is split into $N = T/\Delta t$ subintervals:

$$(t^n, t^{n+1}), \text{ where } t^n = n\Delta t, \quad n = 0, \ldots, N$$

   and Δt is a time step.
3. The space is discretized by a Cartesian grid with steps Δx, Δy.



From there, updating the surface $\phi(i,j)$ is done with:

$$\phi(i,j,t+\Delta t) = \phi(i,j,t) - \Delta t \left[max[V,0]\nabla^{+x}(i,j) + min[V,0]\nabla^{-x}(i,j) \right] \tag{5}$$

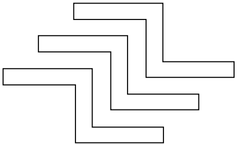
Where:

$$\begin{gathered} \nabla^{+x}(i,j) = \max\left[0, \Delta^{-x}\phi(i,j)\right]^2 + \min\left[0, \Delta^{+x}\phi(i,j)\right]^2, \text{ when } V > 0, \text{ or} \\ \nabla^{-x}(i,j) = \max\left[0, \Delta^{+x}\phi(i,j)\right]^2 + \min\left[0, \Delta^{-x}\phi(i,j)\right]^2, \text{ when } V < 0 \end{gathered} \tag{6}$$

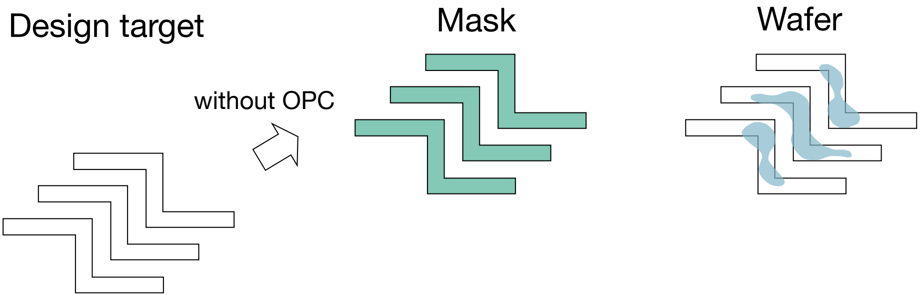
Denote the normal vector of speed V and $\nabla(i,j)$ at the same direction.
The eq. (5) can be simplified to:

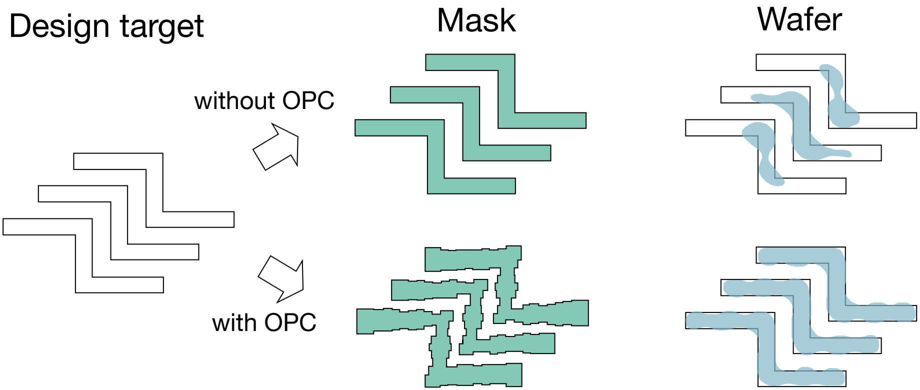
$$\phi(i,j,t+\Delta t) = \phi(i,j,t) - \Delta t \cdot V \cdot \nabla(i,j) \tag{7}$$

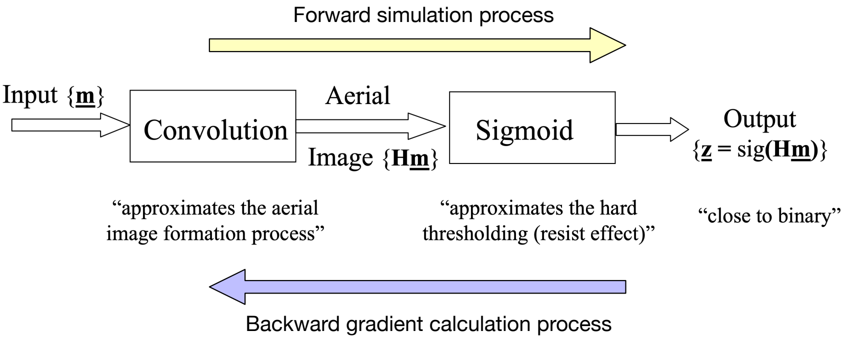
Design target



Review of mask optimization




Design target
Mask
Wafer
without OPC
with OPC



Forward simulation process
Input {m}
Convolution
Aerial
Image {Hm}
Sigmoid
Output
{z = sig(Hm)}
"approximates the aerial image formation process"
"approximates the hard thresholding (resist effect)"
"close to binary"
Backward gradient calculation process

Inverse Lithography Technique (ILT)

The main objective in ILT is minimizing the lithography error through gradient descent.

$$E = ||\mathbf{Z}_t - \mathbf{Z}||_2^2, \tag{8}$$

where $\mathbf{Z}_t$ is the target and $\mathbf{Z}$ is the wafer image of a given mask.
Apply translated sigmoid functions to make the pixel values close to either 0 or 1.

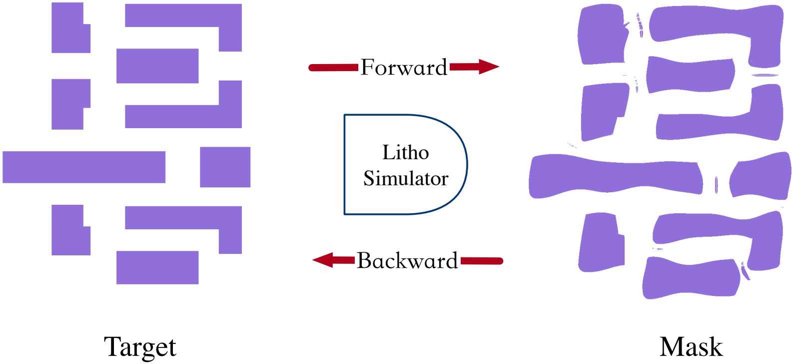
$$\mathbf{Z} = \frac{1}{1 + \exp[-\alpha \times (\mathbf{I} - \mathbf{I}_{th})]}, \tag{9}$$

$$\mathbf{M}_b = \frac{1}{1 + \exp(-\beta \times \mathbf{M})}. \tag{10}$$

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{M}} =& 2\alpha\beta \times \mathbf{M}_b \odot (1 - \mathbf{M}_b) \odot \\ & (((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H}^*)) \otimes \mathbf{H} + \\ & ((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H})) \otimes \mathbf{H}^*).\end{aligned} \tag{11}$$

ILT with visulization

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{M}} =& 2\alpha\beta \times \mathbf{M}_b \odot (1-\mathbf{M}_b) \odot \\
&(((\mathbf{Z}-\mathbf{Z}_t) \odot \mathbf{Z} \odot (1-\mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H}^*)) \otimes \mathbf{H} + \\
&((\mathbf{Z}-\mathbf{Z}_t) \odot \mathbf{Z} \odot (1-\mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H})) \otimes \mathbf{H}^*).
\end{aligned} \tag{12}
$$



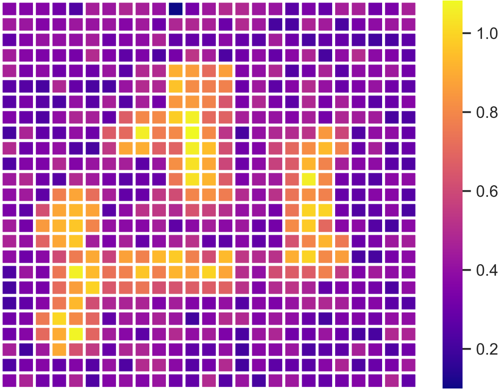
The gradient can be got from the litho-simulation.

Recall that, in the level set function:

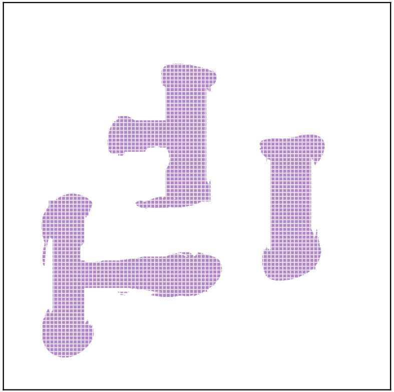
$$\phi(i,j,t+\Delta t) \quad = \quad \phi(i,j,t) - \Delta t \left[max[V,0]\nabla^{+x}(i,j) + min[V,0]\nabla^{-x}(i,j) \right]$$

1. Here, the V is given by the gradient.
2. The $\nabla^{+x}(i,j)$ and $\nabla^{-x}(i,j)$ is given by the level set function.

So we can use level set to solve the mask optimization problem.

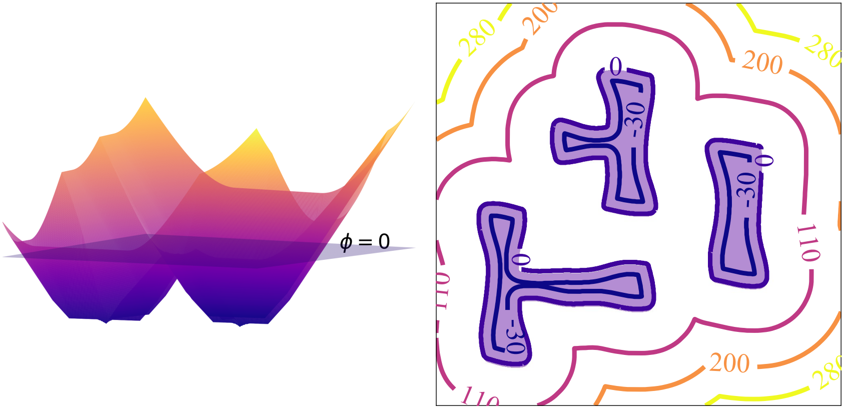


(a)



(b)

Comparison of pixel-based ILT and level set-based ILT. (a) Intensity matrix of pixel-based ILT; (b) Mask generated by pixel-wise intensity threshold;

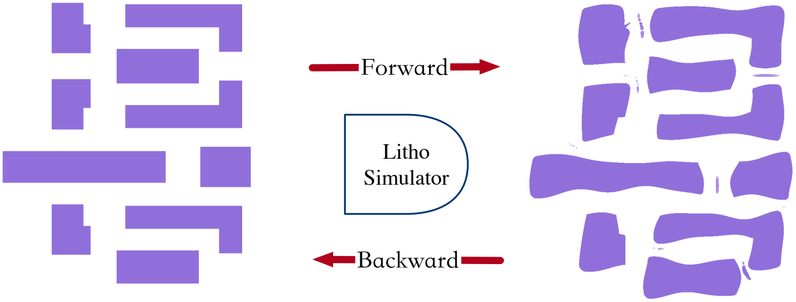


(a) (b)

Comparison of pixel-based ILT and level set-based ILT. (a) Level set-based ILT; (b) Mask generated by zero level set.

1. V : gradient on the image.
2. $\nabla(i, j)$: self defined level set function.

V: gradient on the image.



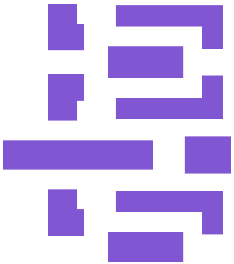
Target

Mask

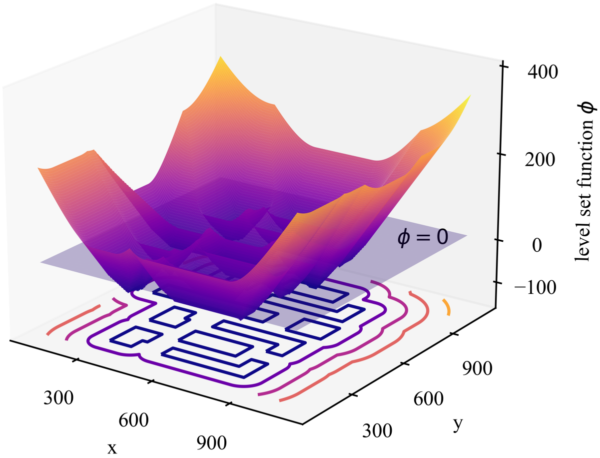
The gradient can be got from the litho-simulation.

$\nabla(i,j)$: self defined level set function.

The level set function is defined as: min distance of each point to the boundary.



(a) target



(b) levelset function

$$\phi(i, j, t + \Delta t) = \phi(i, j, t) - \Delta t \cdot V \cdot \nabla(i, j)$$

$$\phi(i,j,t+\Delta t) = \phi(i,j,t) - \Delta t \cdot V \cdot \nabla(i,j)$$

The drawbacks of conventional level sets OPC.

1. V : the conventional litho simulator needs ~40s to calculate the gradient.
2. $\nabla^{+x}(i,j)$ and $\nabla^{-x}(i,j)$: ~5s for a 1280×1280 image.

Usually, We need more than 20 iterations. More than 1000 seconds totally.

$$\phi(i,j,t+\Delta t) = \phi(i,j,t) - \Delta t \cdot V \cdot \nabla(i,j)$$

The drawbacks of conventional level sets OPC.

1. V : the conventional litho simulator needs ~40s to calculate the gradient.
2. $\nabla^{+x}(i,j)$ and $\nabla^{-x}(i,j)$: ~5s for a 1280 $\times$ 1280 image.

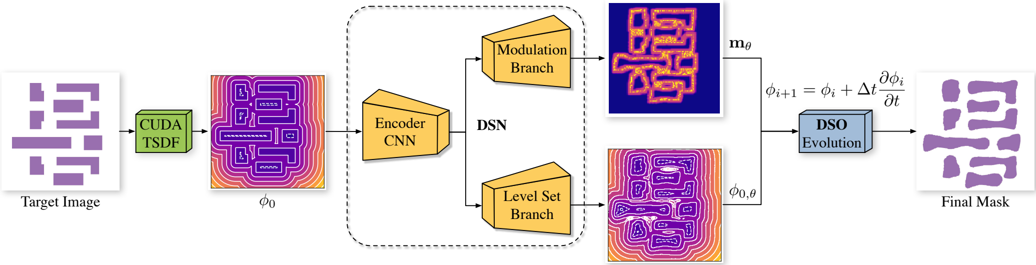
Usually, We need more than 20 iterations. More than 1000 seconds totally.

Warning

Too Slow.

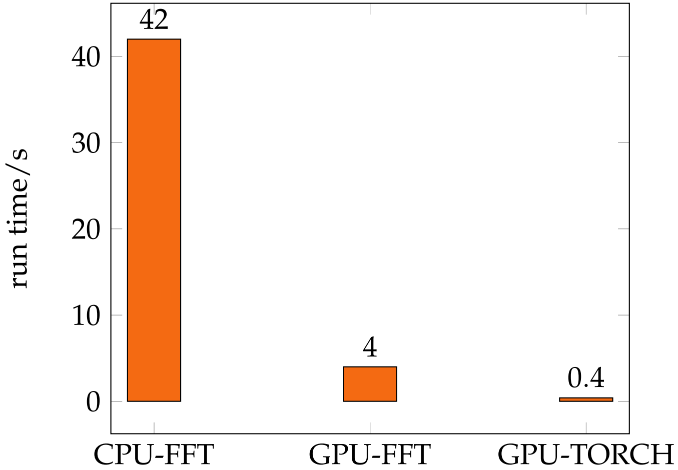
Deep Level sets

Standing on the shoulders of the giants



Overview of DevelSet framework with the end-to-end joint optimization flow of DSN and DSO.

DSO: 1. GPU acceleration



GPU-TORCH can achieve nearly 40 times speed up each epoch.
Total runtime can be reduced from more than 400s to 3-10 s.

SDF:

$$\phi_{\text{SDF}}(x,y) = \begin{cases} -d(x,y), & \text{if}(x,y) \in \text{inside}(\mathcal{C}), \\ 0, & \text{if}(x,y) \in \mathcal{C}, \\ d(x,y), & \text{if}(x,y) \in \text{outside}(\mathcal{C}), \end{cases} \tag{13}$$

TSDF:

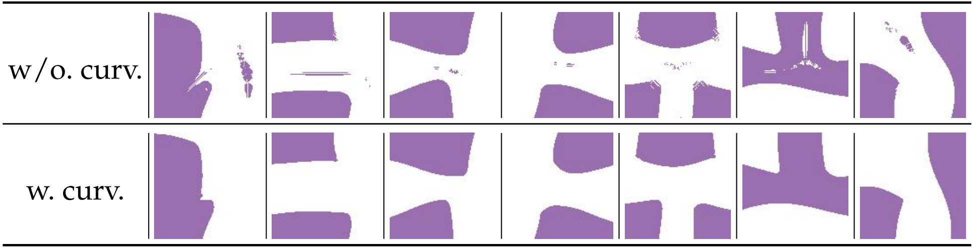
$$\phi_{\text{TSDF}} = \begin{cases} D_u, & \text{if } \phi_{\text{SDF}} > D_u, \\ \phi_{\text{SDF}}, & \text{if } D_l \le \phi_{\text{SDF}} \le D_u, \\ D_l, & \text{if } \phi_{\text{SDF}} < D_l. \end{cases} \tag{14}$$

DSO: 3. Curvature term

$$\kappa = \lambda \mathbf{m}_\theta \left|\nabla \phi_i\right| \operatorname{div}\left(\frac{\nabla \phi_i}{\left|\nabla \phi_i\right|}\right), \tag{15}$$

$$\frac{\partial \phi_i}{\partial t} = -\left(\alpha \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}}\right)\left|\nabla \phi_i\right| + \underline{\lambda \mathbf{m}_\theta \left|\nabla \phi_i\right| \operatorname{div}\left(\frac{\nabla \phi_i}{\left|\nabla \phi_i\right|}\right)}. \tag{16}$$

Curvature term: visulization



Visualizations for ablation study of the curvature term.

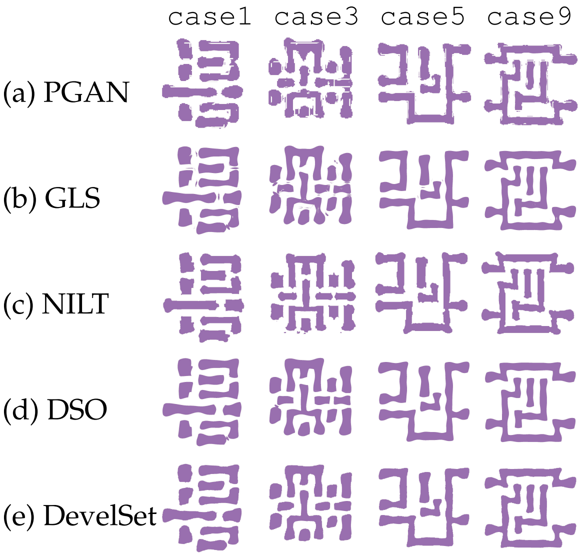
$$L_0(\theta) = \sum_{(x,y)} \left(\phi_{0,\theta}(x,y) - \phi_{\mathrm{gt}}(x,y)\right)^2 , \tag{17}$$

DSN: 2. Modulation branch

$$\begin{aligned}\frac{\partial \phi_i}{\partial t} =& -(\alpha\frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta\frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}})|\nabla \phi_i| \\ &+ \lambda \mathbf{m}_\theta \, |\nabla \phi_i| \, \text{div}\left(\frac{\nabla \phi_i}{|\nabla \phi_i|}\right).\end{aligned} \tag{18}$$

$$L_m(\theta) = \sum_{(x,y)} \big(H_\varepsilon(\phi_{m,\theta}(x,y)) - m_{\text{gt}}(x,y)\big)^2, \tag{19}$$

Results



Mask visualizations.

Table: Mask Printability, Complexity Comparison with SOTA.

Bench	Area(nm^2)	ILT (DAC13)			Level-Set (DATE20)			GAN-OPC (DAC18)			Neural-ILT (ICCAD20)			**DevelSet**		
		L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots	L_2	PVB	#shots
`case1`	215344	49893	65534	2478	46032	62693	1476	52570	56267	931	50795	63695	743	49142	59607	969
`case2`	169280	50369	48230	704	36177	50642	861	42253	50822	692	36969	60232	571	34489	52012	743
`case3`	213504	81007	108608	2319	71178	100945	2811	83663	94498	1048	94447	85358	791	93498	76558	889
`case4`	82560	20044	28285	1165	16345	29831	432	19965	28957	386	17420	32287	209	18682	29047	376
`case5`	281958	44656	58835	1836	47103	56328	963	44733	59328	950	42337	65536	631	44256	58085	902
`case6`	286234	57375	48739	993	46205	51033	942	46062	52845	836	39601	59247	745	41730	53410	774
`case7`	229149	37221	43490	577	28609	44953	548	26438	47981	515	25424	50109	354	25797	46606	527
`case8`	128544	19782	22846	504	19477	22541	439	17690	23564	286	15588	25826	467	15460	24836	493
`case9`	317581	55399	66331	2045	52613	62568	881	56125	65417	1087	52304	68650	653	50834	64950	932
`case10`	102400	24381	18097	380	22415	18769	333	9990	19893	338	10153	22443	423	10140	21619	393
Average		44012.7	50899.5	1300.1	38615.4	50030.3	968.6	39948.9	49957.2	706.9	38503.8	53338.3	558.7	**38402.8**	**48673.0**	699.8
Ratio		1.146	1.046	1.858	1.006	1.028	1.384	1.040	1.026	1.010	1.003	1.096	0.798	**1.000**	**1.000**	1.000

† *L_2 and PVB unit: nm^2.*

THANK YOU!