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
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:

$$\forall t \in (0, T), x \in \mathbb{R}^d, \begin{cases} 
\phi(t, x) < 0 & \text{if } x \in \Omega(t) \\
\phi(t, x) = 0 & \text{if } x \in \Gamma(t) \\
\phi(t, x) > 0 & \text{if } x \in \Omega(t)^c
\end{cases}$$  \hspace{1cm} (1)
Let \( x_0 \in \Gamma(0) \) be fixed. By the intuitive definition of an evolving domain, it comes:

\[
\forall t \in (0, T), \phi(t, \chi(x_0, t, 0)) = 0
\]

Differentiating and using the chain rule yields:

\[
\frac{\partial \phi}{\partial t}(t, \chi(x_0, t, 0)) + \frac{d}{dt}(\chi(x_0, t, 0)) \cdot \nabla \phi(t, \chi(x_0, t, 0)) = 0
\]

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

\[
\forall t \in (0, T), \forall x \in \mathbb{R}^d, \frac{\partial \phi}{\partial t} + V(t, x) \cdot \nabla \phi = 0
\]
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*. 
A domain $\Omega \subset \mathbb{R}^d$ is equivalently defined by a function $\phi: \mathbb{R}^d \rightarrow \mathbb{R}$ such that:

$$\phi(x) < 0 \quad \text{if } x \in \Omega \quad ; \quad \phi(x) = 0 \quad \text{if } x \in \Gamma \quad ; \quad \phi(x) > 0 \quad \text{if } x \in \overline{c \Omega}$$  \hspace{1cm} (2)

Here the zero level set of the surface is a square.
Merging and splitting are here handled naturally by the surface motion.

The evolving front in red is known by taking the zero level set of a surface $\phi$. 

(a) $\phi_0$  
(b) $\phi_6$  
(c) $\phi_{10}$  
(d) $\phi_T$
Merging and splitting are here handled naturally by the surface motion.

The evolving front in red is known by taking the zero level set of a surface $\phi$.

**Question**

The question is now: what is the function $\phi$?
Math definition

\[ \phi(x(t), t) = 0 \]

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 \( \phi \) at \( t=0 \), it would be possible to know \( \phi \) at any time \( t \) with the motion equation \( \frac{\partial \phi}{\partial t} \).
The chain rule gives us:

\[
\frac{\partial \phi(x(t), t)}{\partial t} = 0
\]

\[
\frac{\partial \phi}{\partial x(t)} \frac{\partial x(t)}{\partial t} + \frac{\partial \phi t}{t} = 0
\]

\[
\frac{\partial \phi}{\partial x(t)} x_t + \phi_t = 0
\]

(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 $x_t = V(x(t)) \ n$ where $n = \frac{\nabla \phi}{|\nabla \phi|}$. The previous motion equation can be rewritten with:

$$
\phi_t + \nabla \phi x_t = 0 \\
\phi_t + \nabla \phi F n = 0 \\
\phi_t + V \nabla \phi \frac{\nabla \phi}{|\nabla \phi|} = 0 \\
\phi_t + V |\nabla \phi| = 0
$$

(4)
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 \(\Delta t\) is a time step.

3. The space is discretized by a **Cartesian grid** with steps \(\Delta x, \Delta y\).
From there, updating the surface $\phi(i, j)$ is done with:

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

(5)

Where:

$$
\nabla^+x(i, j) = \max[0, \Delta^-x\phi(i, j)]^2 + \min[0, \Delta^+x\phi(i, j)]^2, \text{ when } V > 0, \text{ or }
$$

$$
\nabla^-x(i, j) = \max[0, \Delta^+x\phi(i, j)]^2 + \min[0, \Delta^-x\phi(i, j)]^2, \text{ when } V < 0
$$

(6)

Denote the normal vector of speed $V$ and $\nabla(i, j)$ at the same direction.

The eq. (5) can be simplified to:

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

(7)
Design target
Review of mask optimization

Design target without OPC

Mask

Wafer
Review of mask optimization

Design target

Mask

Wafer

without OPC

with OPC
ILT flow

Forward simulation process

Input $\{m\}$ → Convolution → Aerial Image $\{Hm\}$ → Sigmoid → Output $\{z = \text{sig}(Hm)\}$

"approximates the aerial image formation process"  "approximates the hard thresholding (resist effect)"  "close to binary"

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

\[ E = ||Z_t - Z||^2_2, \quad (8) \]

where \( Z_t \) is the target and \( Z \) is the wafer image of a given mask. Apply translated sigmoid functions to make the pixel values close to either 0 or 1.

\[ Z = \frac{1}{1 + \exp[-\alpha \times (I - I_{th})]}, \quad (9) \]

\[ M_b = \frac{1}{1 + \exp(-\beta \times M)}. \quad (10) \]

\[ \frac{\partial E}{\partial M} = 2\alpha \beta \times M_b \odot (1 - M_b) \odot \]

\[ (((Z - Z_t) \odot Z \odot (1 - Z) \odot (M_b \otimes H^*)) \otimes H + \]

\[ ((Z - Z_t) \odot Z \odot (1 - Z) \odot (M_b \otimes H)) \otimes H^*). \quad (11) \]
\[
\frac{\partial E}{\partial M} = 2\alpha\beta \times M_b \odot (1 - M_b) \odot
\]

\[
(((Z - Z_t) \odot Z \odot (1 - Z) \odot (M_b \otimes H^*)) \otimes H^+
\]

\[
((Z - Z_t) \odot Z \odot (1 - Z) \odot (M_b \otimes H)) \otimes H^*).
\]

(12)

The gradient can be got from the litho-simulation.
Recall that, in the level set function:

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

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

So we can use level set to solve the mask optimization problem.
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;
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.
The gradient can be got from the litho-simulation.
The level set function is defined as: min distance of each point to the boundary.

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

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

The drawbacks of conventional level sets OPC.

1. \( V \): the conventional litho simulator needs \(~40\)s to calculate the gradient.

2. \( \nabla^+ x(i, j) \) and \( \nabla^- x(i, j) \): \(~5\)s for a \(1280 \times 1280\) 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 \( \sim 40 \)s to calculate the gradient.

2. \( \nabla^+_x(i, j) \) and \( \nabla^-_x(i, j) \): \( \sim 5 \)s 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.
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:
\[
\phi_{\text{SDF}}(x, y) = \begin{cases} 
-d(x, y), & \text{if } (x, y) \in \text{inside}(C), \\
0, & \text{if } (x, y) \in C, \\
d(x, y), & \text{if } (x, y) \in \text{outside}(C), 
\end{cases} 
\]  
(13)

TSDF:
\[
\phi_{\text{TSDF}} = \begin{cases} 
D_u, & \text{if } \phi_{\text{SDF}} > D_u, \\
\phi_{\text{SDF}}, & \text{if } D_l \leq \phi_{\text{SDF}} \leq D_u, \\
D_l, & \text{if } \phi_{\text{SDF}} < D_l.
\end{cases} 
\]  
(14)
DSO: 3. Curvature term

\[ \kappa = \lambda m_\theta |\nabla \phi_i| \text{div} \left( \frac{\nabla \phi_i}{|\nabla \phi_i|} \right), \]  
\[ (15) \]

\[ \frac{\partial \phi_i}{\partial t} = - \left( \alpha \frac{\partial L_{ilt}}{\partial M} + \beta \frac{\partial L_{pvb}}{\partial M} \right) |\nabla \phi_i| + \lambda m_\theta |\nabla \phi_i| \text{div} \left( \frac{\nabla \phi_i}{|\nabla \phi_i|} \right). \]  
\[ (16) \]
Visualizations for ablation study of the curvature term.

<table>
<thead>
<tr>
<th>w/o. curv.</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Visualization" /></td>
<td><img src="image2.png" alt="Visualization" /></td>
<td><img src="image3.png" alt="Visualization" /></td>
<td><img src="image4.png" alt="Visualization" /></td>
<td><img src="image5.png" alt="Visualization" /></td>
<td><img src="image6.png" alt="Visualization" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>w. curv.</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image7.png" alt="Visualization" /></td>
<td><img src="image8.png" alt="Visualization" /></td>
<td><img src="image9.png" alt="Visualization" /></td>
<td><img src="image10.png" alt="Visualization" /></td>
<td><img src="image11.png" alt="Visualization" /></td>
<td><img src="image12.png" alt="Visualization" /></td>
</tr>
</tbody>
</table>
\[ L_0(\theta) = \sum_{(x,y)} (\phi_{0,\theta}(x,y) - \phi_{gt}(x,y))^2, \]  

(17)
\[
\frac{\partial \phi_i}{\partial t} = - \left( \alpha \frac{\partial L_{\text{ilt}}}{\partial \mathbf{M}} + \beta \frac{\partial L_{\text{pvb}}}{\partial \mathbf{M}} \right) |\nabla \phi_i| \\
+ \lambda m_\theta |\nabla \phi_i| \text{div} \left( \frac{\nabla \phi_i}{|\nabla \phi_i|} \right). 
\]

(18)

\[
L_m(\theta) = \sum_{(x,y)} \left( H_\varepsilon(\phi_{m,\theta}(x,y)) - m_{\text{gt}}(x,y) \right)^2,
\]

(19)
Results

(a) PGAN
(b) GLS
(c) NILT
(d) DSO
(e) DevelSet

Mask visualizations.
## Table: Mask Printability, Complexity Comparison with SOTA.

<table>
<thead>
<tr>
<th>Bench</th>
<th>Area($nm^2$)</th>
<th>ILT (DAC13) $L_2$</th>
<th>PVB</th>
<th>#shots</th>
<th>Level-Set (DATE20) $L_2$</th>
<th>PVB</th>
<th>#shots</th>
<th>GAN-OPC (DAC18) $L_2$</th>
<th>PVB</th>
<th>#shots</th>
<th>Neural-ILT (ICCAD20) $L_2$</th>
<th>PVB</th>
<th>#shots</th>
<th>DevelSet $L_2$</th>
<th>PVB</th>
<th>#shots</th>
</tr>
</thead>
<tbody>
<tr>
<td>case1</td>
<td>215344</td>
<td>49893</td>
<td>65534</td>
<td>2478</td>
<td>46032</td>
<td>62693</td>
<td>1476</td>
<td>52570</td>
<td>56267</td>
<td>931</td>
<td>50795</td>
<td>63695</td>
<td>743</td>
<td>49142</td>
<td>59607</td>
<td>969</td>
</tr>
<tr>
<td>case2</td>
<td>169280</td>
<td>50369</td>
<td>48230</td>
<td>704</td>
<td>36177</td>
<td>50642</td>
<td>861</td>
<td>42253</td>
<td>50822</td>
<td>692</td>
<td>36969</td>
<td>60232</td>
<td>571</td>
<td>34489</td>
<td>52012</td>
<td>743</td>
</tr>
<tr>
<td>case3</td>
<td>213504</td>
<td>81007</td>
<td>108608</td>
<td>2319</td>
<td>71178</td>
<td>100945</td>
<td>2811</td>
<td>83663</td>
<td>94498</td>
<td>1048</td>
<td>94447</td>
<td>85358</td>
<td>791</td>
<td>93498</td>
<td>76558</td>
<td>889</td>
</tr>
<tr>
<td>case4</td>
<td>82560</td>
<td>20044</td>
<td>28285</td>
<td>1165</td>
<td>16345</td>
<td>29831</td>
<td>432</td>
<td>19965</td>
<td>28957</td>
<td>386</td>
<td>17420</td>
<td>32287</td>
<td>209</td>
<td>18682</td>
<td>29047</td>
<td>376</td>
</tr>
<tr>
<td>case5</td>
<td>281958</td>
<td>44656</td>
<td>58835</td>
<td>1836</td>
<td>47103</td>
<td>56328</td>
<td>963</td>
<td>44733</td>
<td>59328</td>
<td>950</td>
<td>42337</td>
<td>65536</td>
<td>631</td>
<td>44256</td>
<td>58085</td>
<td>902</td>
</tr>
<tr>
<td>case6</td>
<td>286234</td>
<td>57375</td>
<td>48739</td>
<td>993</td>
<td>46205</td>
<td>51033</td>
<td>942</td>
<td>46062</td>
<td>52845</td>
<td>836</td>
<td>39601</td>
<td>59247</td>
<td>745</td>
<td>41730</td>
<td>53410</td>
<td>774</td>
</tr>
<tr>
<td>case7</td>
<td>229149</td>
<td>37221</td>
<td>43490</td>
<td>577</td>
<td>28609</td>
<td>44953</td>
<td>548</td>
<td>26438</td>
<td>47981</td>
<td>515</td>
<td>25424</td>
<td>50109</td>
<td>354</td>
<td>25797</td>
<td>46606</td>
<td>527</td>
</tr>
<tr>
<td>case8</td>
<td>128544</td>
<td>19782</td>
<td>22846</td>
<td>504</td>
<td>19477</td>
<td>22541</td>
<td>439</td>
<td>17690</td>
<td>23564</td>
<td>286</td>
<td>15588</td>
<td>25826</td>
<td>467</td>
<td>15460</td>
<td>24836</td>
<td>493</td>
</tr>
<tr>
<td>case9</td>
<td>317581</td>
<td>55399</td>
<td>66331</td>
<td>2045</td>
<td>52613</td>
<td>62568</td>
<td>881</td>
<td>56125</td>
<td>65417</td>
<td>1087</td>
<td>52304</td>
<td>68650</td>
<td>653</td>
<td>50834</td>
<td>64950</td>
<td>932</td>
</tr>
<tr>
<td>case10</td>
<td>102400</td>
<td>24381</td>
<td>18097</td>
<td>380</td>
<td>22415</td>
<td>18769</td>
<td>333</td>
<td>9990</td>
<td>19893</td>
<td>338</td>
<td>10153</td>
<td>22443</td>
<td>423</td>
<td>10140</td>
<td>21619</td>
<td>393</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>44012.750899.513000.1</td>
<td>38615.450030.3</td>
<td>968.6</td>
<td>39948.949957.2</td>
<td>706.9</td>
<td>38503.853338.3</td>
<td>558.7</td>
<td>38402.8</td>
<td>48673.0</td>
<td>699.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td></td>
<td>1.146</td>
<td>1.046</td>
<td>1.858</td>
<td>1.006</td>
<td>1.028</td>
<td>1.384</td>
<td>1.040</td>
<td>1.026</td>
<td>1.010</td>
<td>1.003</td>
<td>1.096</td>
<td>0.798</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

$^\dagger$L$_2$ and PVB unit: nm$^2$. 

Results 35/36
THANK YOU!