



Advanced Computation:
Computational Electromagnetics

Surface Propagation Methods

1

Outline

- Introduction to Surface Propagation
- Cell-Volume Method
- String Method
- Level Set Method
- Fast Marching Method

2

Introduction to Surface Propagation

Slide 3

3

Comparison of Methods

- Cell Volume Method
 - Rigorous and stable
 - Very slow!!
- String Method
 - Extremely fast and efficient
 - Restricted to surfaces that are mostly smooth and continuous
 - Inherently unstable
 - Rarely extended to 3D
- Level Set Method
 - Most rigorous
 - Stable
 - Excellent for 3D
 - Slower than FMM
- Fast Marching Method
 - Requires
 - Unidirectional progression of surface
 - Rate function depends only on position
 - Stable
 - Excellent for 3D
 - Very fast and efficient!

Slide 4

4

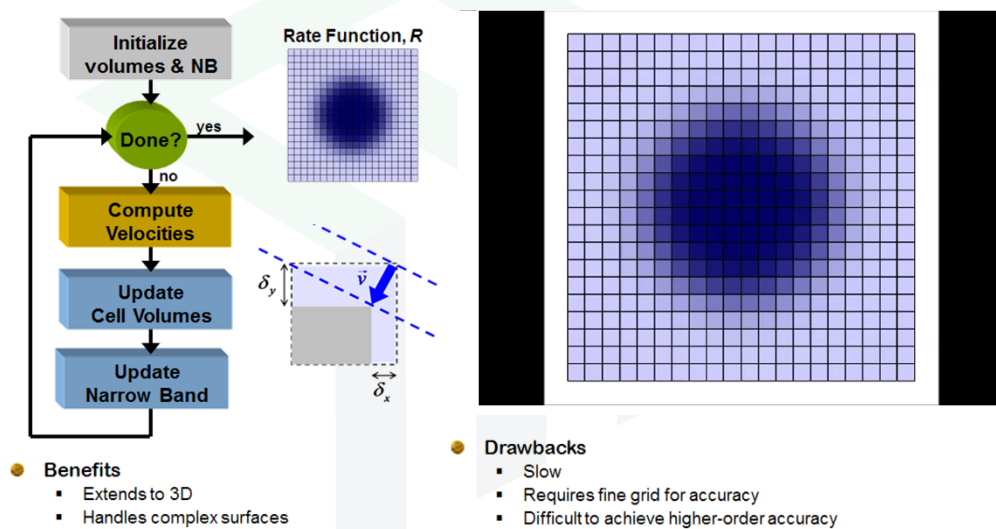
Cell-Volume Method

Hirt, Cyril W., and Billy D. Nichols. "Volume of fluid (VOF) method for the dynamics of free boundaries." Journal of computational physics 39.1 (1981): 201-225.

Slide 5

5

Algorithm for Cell-Volume Method

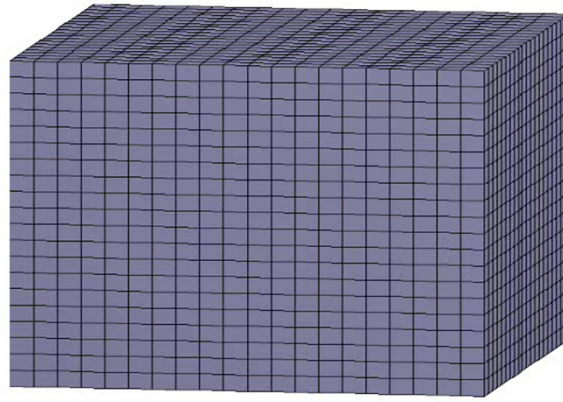


R. C. Rumpf, E. G. Johnson, "Modeling the formation of photonic crystals by holographic lithography," Proceedings of SPIE Micromachining Technology for Microoptics and Nanooptics III, Vol. 5720, pp. 18-26, Jan. 2005.

Slide 6

6

3D Cell Volume Method



7

String Method

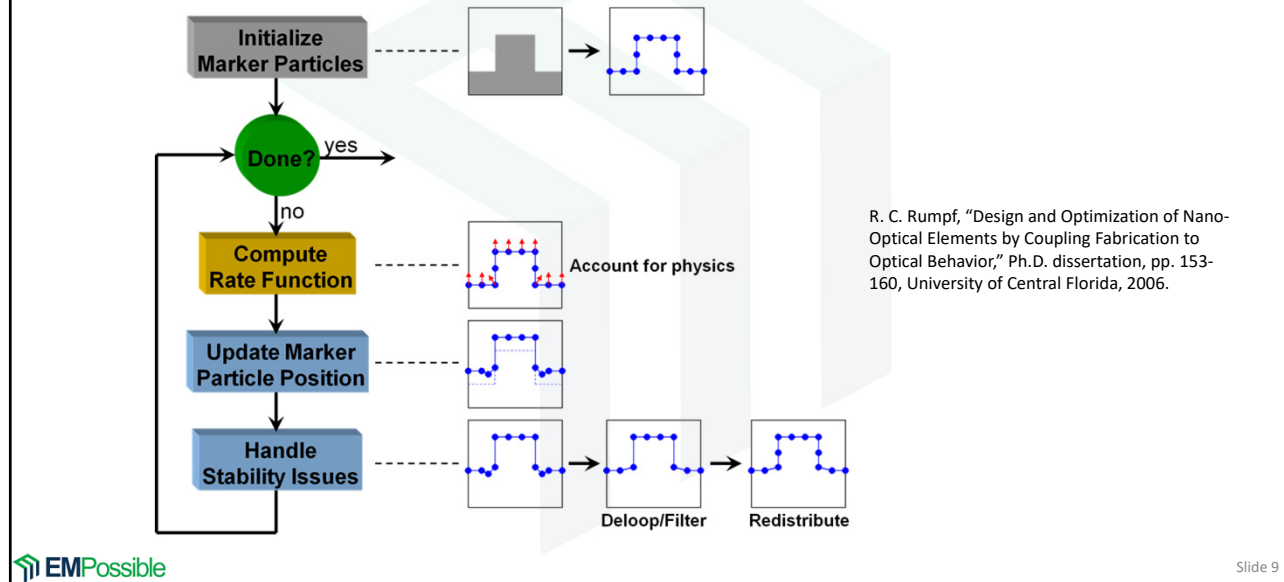
Kawakami, Shojiro, Takayuki Kawashima, and Takashi Sato. "Mechanism of shape formation of three-dimensional periodic nanostructures by bias sputtering." *Applied Physics Letters* 74.3 (1999): 463-465.

Sethian, James A. "Curvature and the evolution of fronts." *Communications in Mathematical Physics* 101.4 (1985): 487-499.

Tazawa, Satoshi, Seitaro Matsuo, and Kazuyuki Saito. "A general characterization and simulation method for deposition and etching technology." *IEEE transactions on semiconductor manufacturing* 5.1 (1992): 27-33.

8

Algorithm for the String Method



9

The Math

The list of points \vec{x}_i describing the surface are stored in an array.

$$\mathbf{X} = [\vec{x}_1 \quad \vec{x}_2 \quad \cdots \quad \vec{x}_N]$$

The *rate function* \mathbf{R} quantifies how fast and in what direction each point on the surface must move to best model the physics of the process being simulated.

$$\mathbf{R} = [\vec{r}_1 \quad \vec{r}_2 \quad \cdots \quad \vec{r}_N] \quad \leftarrow \text{This is the most difficult part of the method to formulate.}$$

Given the rate function, the positions of the points are updated given a suitable time step Δt . Δt must be chosen that is small enough to prevent particles from crossing paths.

$$\mathbf{X}_{t+\Delta t} = \mathbf{X}_t + \Delta t \mathbf{R}_t$$

10

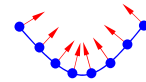
Sources of Instability and Fixes

There are two main reasons the string method is inherently unstable.

1. Points Converging

As points converge, they reduce the accuracy of the rate function.
They may also cross paths and form unstable loops in the string

This is mitigated using a "smoothing" function and by removing points from the string.

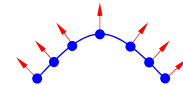


1. Points Diverging

As points diverge, the distribution of points becomes too sparse to adequately resolve the surface.

This is mitigated by adding additional points.

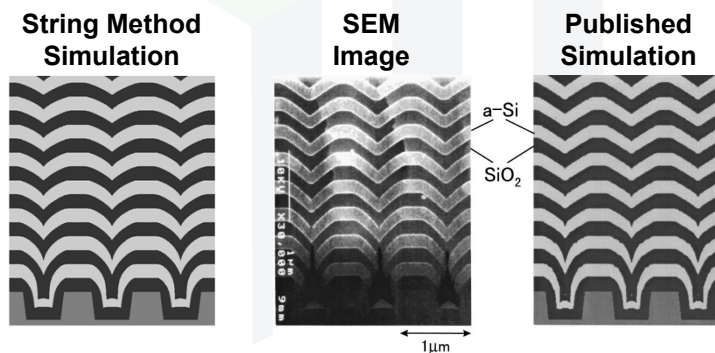
Care must be taken to correctly interpolate where the points are to be added. Performing a filtering function on the position vectors after the update can help maintain stability.



Example #1: Autocloning

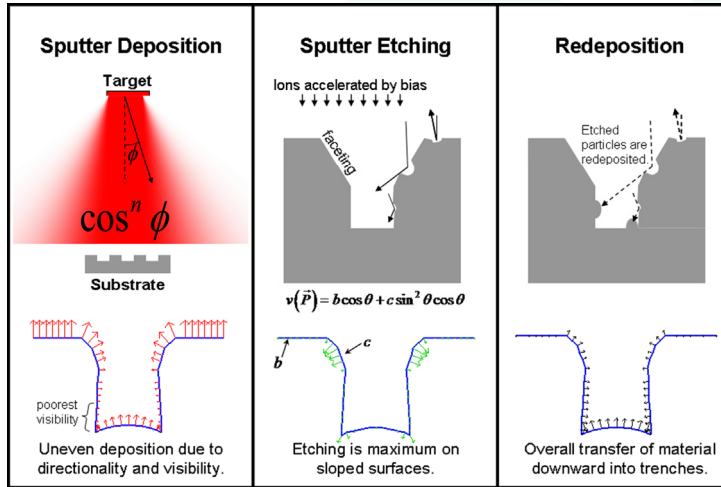
What is Autocloning?

Autocloning is a nanofabrication process where alternating layers of dielectric are deposited onto a corrugated surface. When the process parameters are just right, a self-shaping effect causes the surface topology to quickly converge to a profile that is maintained over a virtually unlimited number of layers.



Example #1: Autocloning

Physics of Autocloning?



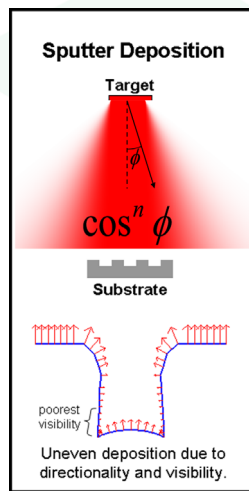
Three physical mechanisms produce autocloning.

R. C. Rumpf, "Design and Optimization of Nano-Optical Elements by Coupling Fabrication to Optical Behavior," Ph.D. dissertation, pp. 153-160, University of Central Florida, 2006.

Example #1: Autocloning

Sputter Deposition

During sputter deposition, neutral particles are deposited from a target to the substrate with an angular dependence $\cos^n \phi$.



Rate Function for Deposition

$$R_{\text{dep}}(\vec{x}_i) = d \frac{\int_{-\pi/2}^{\pi/2} V_T(\vec{x}_i, \phi) \cos^n(\phi) \cos(\phi - \theta) d\phi}{\int_{-\pi/2}^{\pi/2} \cos^{n+1}(\phi) d\phi}$$

Deposition can occur on vertical sidewalls!

Simplified Rate Function

$$R_{\text{dep}}(\vec{x}_i) = d \frac{\int_{\phi_{\text{min}}}^{\phi_{\text{max}}} \cos^n(\phi) \cos(\phi - \theta) d\phi}{\int_{-\pi/2}^{\pi/2} \cos^{n+1}(\phi) d\phi}$$

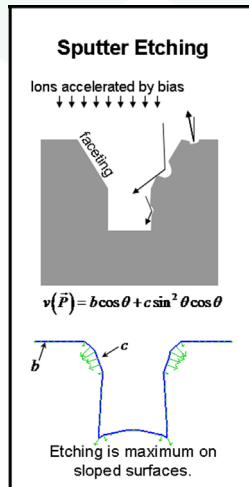
We only integrate over the range of angles visible to the target.

The angular dependence on deposition leads to horizontal surfaces growing faster than vertical surfaces. Visibility leads to a shadowing effect that produces "kinks" near the bottom of the vertical sidewalls.

- d \equiv flux of particles
- θ \equiv surface slope
- V_T \equiv visibility of target
- $V_T = 0$ not visible
- $V_T = 1$ completely visible
- n \equiv diffusion profile
- $n = 0$ isotropic deposition
- $n = \infty$ deposition normal to surface

Example #1: Autocloning *Sputter Etching*

Charged ions accelerated by the field during bias sputtering leads to etching of the substrate. This opposed deposition.



Rate Function for Etching

$$R_{\text{etch}}(\vec{x}_i) = b \cos \theta + c \sin^2 \theta \cos \theta \quad \text{for } b, c < 0$$

$b \equiv$ flux etch rate of a horizontal surface
 $c \equiv$ angular selectivity

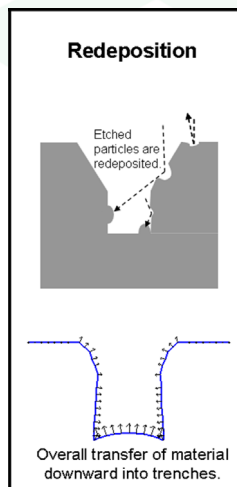
The function $\sin^2 \theta \cos \theta$ has a maximum value around $\theta = 55^\circ$.

Etching is maximum at this angle.

It is this mechanism that leads to regions of the surface profile converging to the constant slope near 55° .

Example #1: Autocloning *Redeposition*

Etched particles can be redeposited back onto the surface. The path of these particles is extremely complex and difficult to model. The tendency is to move particles from the upper part of trenches to lower parts.



Rate Function for Redeposition

$$R_{\text{redep}}(\vec{x}_i) = \frac{e}{2} \int_{-\pi/2}^{\pi/2} V_s(\vec{x}_i, \phi) \cos(\phi - \theta) d\phi$$

$e \equiv$ redeposition rate
 $V_s \equiv$ visibility of surface

We have assumed that particles emerge from all surfaces uniformly. This is a good approximation, especially when the surface profile has converged to V-shaped grooves and the surface slope is more constant.

Points at deeper parts of a groove will receive greater redeposition because greater surface area is visible from which to receive particles.

It is this mechanism that that enables lower portions of the V-grooves to keep pace with the top portions despite shadowing effects inhibiting sputter deposition in the lower portions.

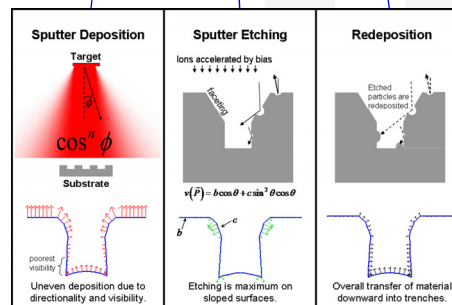
Example #1: Autocloning

The Overall Rate Function

The overall rate function is the sum of the three competing mechanisms. Deposition is in the direction normal the surface.

$$\vec{R}(\vec{x}_i) = \left[R_{\text{dep}}(\vec{x}_i) + R_{\text{etch}}(\vec{x}_i) + R_{\text{redep}}(\vec{x}_i) \right] \hat{n}_i$$

$\hat{n}_i \equiv$ surface normal
at point \vec{x}_i



17

Example #1: Autocloning

Benchmark Simulation

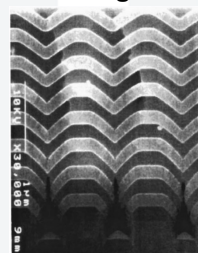
SIMULATION PARAMETERS		
Parameter	a-Si	SiO ₂
d	1	1
n	0.4	1
b	0	-0.1
c	0	-0.4
e	0.15	0.5

PROCESS CONDITIONS		
Parameter	a-Si	SiO ₂
Target	poly-Si	fused silica
Sputtering Gas	Ar+H ₂	Ar+H ₂
Gas Pressure	3.5 mTorr	2.0 mTorr
RF Power	400 W	400 W
Bias	0	50 W

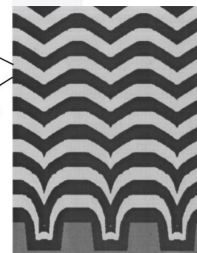
String Method
Simulation



SEM
Image



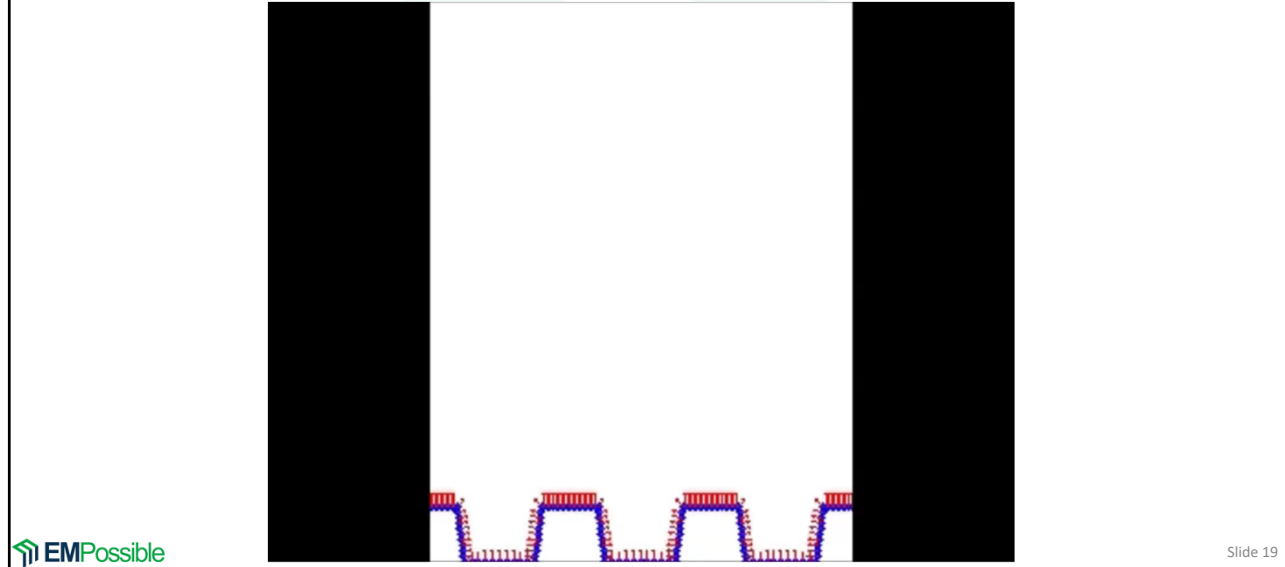
Published
Simulation



18

Example #1: Autocloning

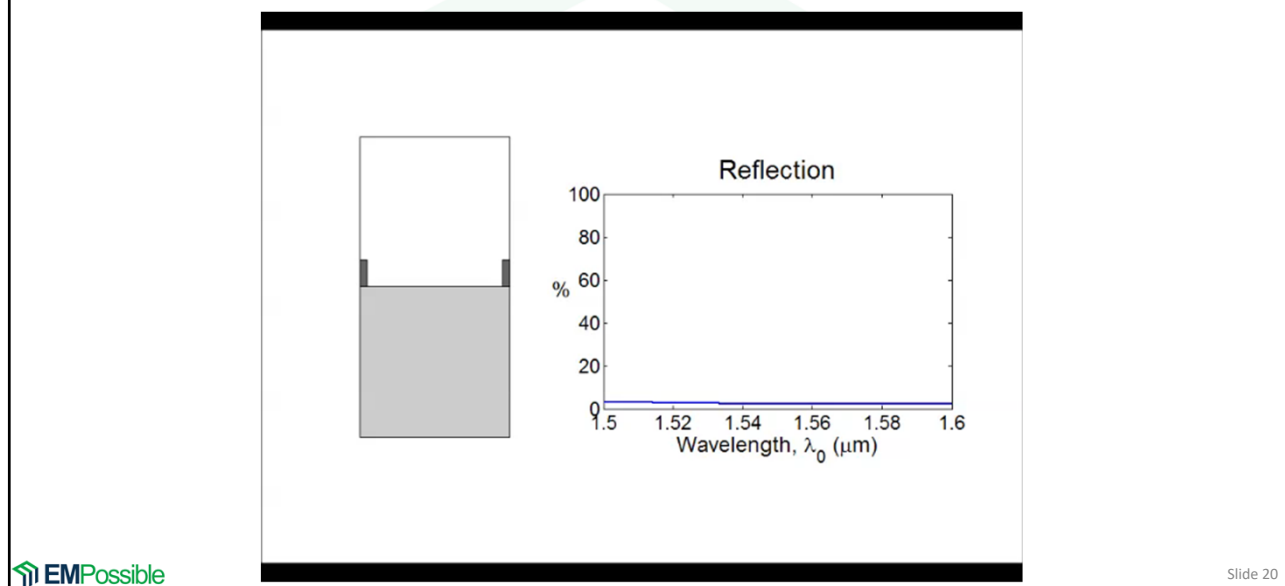
Movie of Autocloning Simulation



19

Example #2: Tuning GMR Filters

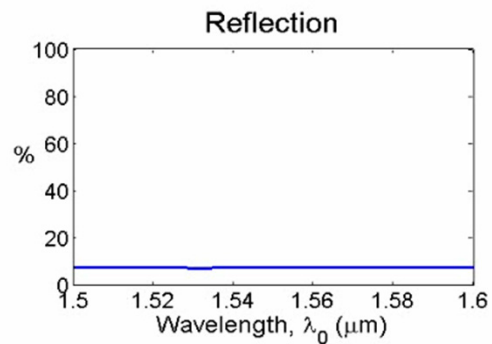
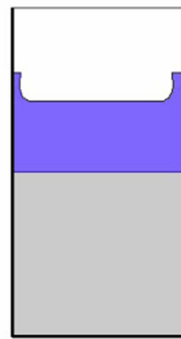
Tune by Deposition



20

Example #2: Tuning GMR Filters

Tune by Etching



Level Set Method

Fedkiw, Stanley Osher Ronald, and Stanley Osher. "Level set methods and dynamic implicit surfaces." *Surfaces* 44 (2002): 77.

Sethian, James Albert. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Vol. 3. Cambridge university press, 1999.

Osher, Stanley, and Ronald P. Fedkiw. "Level set methods: an overview and some recent results." *Journal of Computational physics* 169.2 (2001): 463-502.

A Different Way to Describe Surfaces

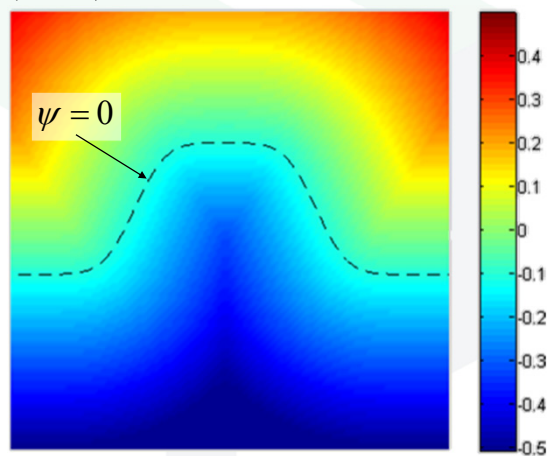
It is most obvious to consider surfaces from a geometry perspective where shapes of structures and positions of interfaces change over time. It is tempting to track the motion of a surface explicitly leading to techniques like the string method and cell-volume method.

The level set method and the fast marching method approach the problem completely differently. They solve differential equations to compute distance “level sets” and “time of arrival” functions. The surfaces as a function of time are interpolated from this data.

This enables robust schemes based on well understood methods to be formulated that are more rigorous and easier to adapt to different conditions.

Level Set Data

$\psi(x, y; t) \equiv$ distance from the surface



The surface at time t is the isometric surface at $\psi = 0$. One set of data only describes the surface at time t .

Level Set Equation for Surface Propagation

The level set equation is an initial value problem in the form of a partial differential equation that must be solved for the level set function ψ .

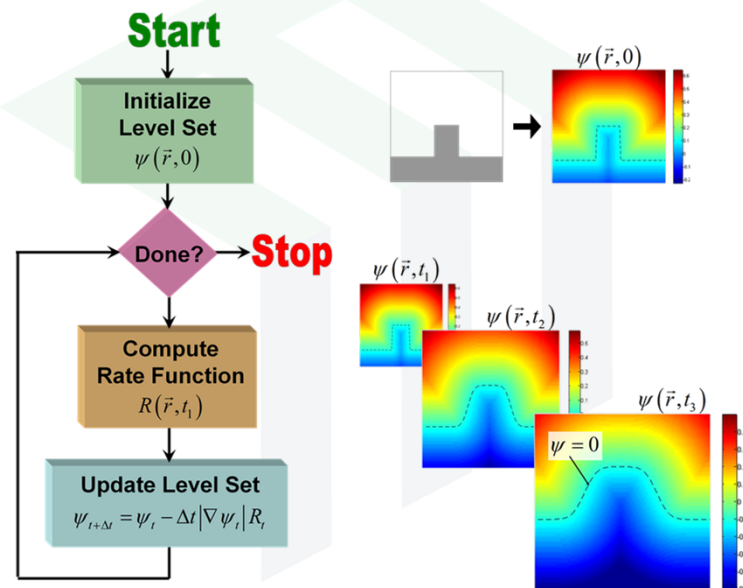
$$\frac{\partial \psi(\vec{r}, t)}{\partial t} + |\nabla \psi(\vec{r}, t)| \cdot R(\vec{r}, t) = 0 \quad \text{given } \psi(\vec{r}, 0)$$

The rate function R is where the physics guiding the surface propagation is incorporated.

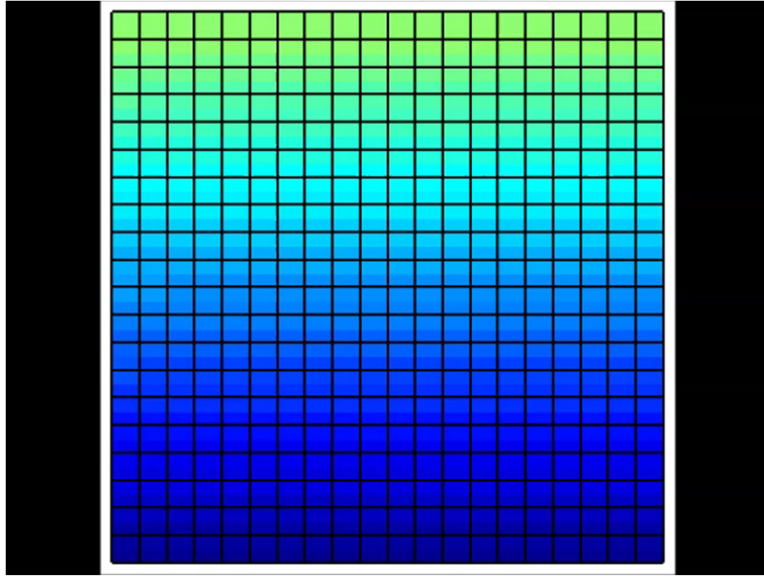
The level set method solved in the time-domain by approximating the time-derivative with a finite-difference and solving for ψ at the future time step.

$$\psi(t + \Delta t) = \psi(t) - \Delta t \cdot |\nabla \psi(t)| \cdot R(t)$$

Algorithm for the Level Set Method



Movie of the Level Set Method



Fast Marching Method

Sethian, James Albert. Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Vol. 3. Cambridge university press, 1999.

Sethian, James A. "A fast marching level set method for monotonically advancing fronts." Proceedings of the National Academy of Sciences 93.4 (1996): 1591-1595.

Special Case

What if: (1) surface does not move backward, and (2) rate function depends only on position?

The level set formulation can be dramatically simplified!!!

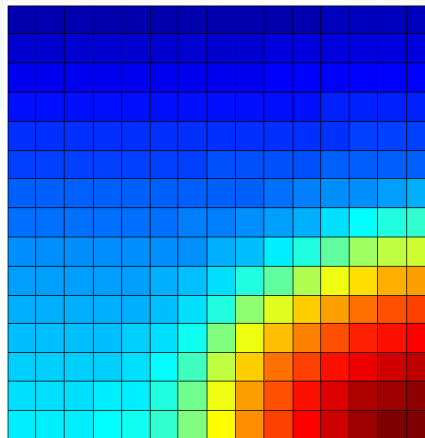
This makes the rate function always positive and it becomes easier to describe the problem in terms of a time-of-arrival function $T(\vec{r})$.

The governing equation is a boundary value problem of the form

$$|\nabla T(\vec{r})| \cdot R(\vec{r}) = 1 \quad \text{initial surface } T(\vec{r}) = 0$$

Time-of-Arrival Function

$T(\vec{r}) \equiv$ time when surface arrives at position \vec{r}



The surface at time t is the isometric surface at $T = t$.
One set of data describes entire progression of the surface.

The Governing Equation

$$|\nabla T(\vec{r})| \cdot R(\vec{r}) = 1 \quad \text{initial surface } T(\vec{r}) = 0$$

This equation is squared and rearranged to obtain

$$|\nabla T(\vec{r})| \cdot R(\vec{r}) = 1$$

$$\left| \frac{\partial T}{\partial x} \hat{x} + \frac{\partial T}{\partial y} \hat{y} + \frac{\partial T}{\partial z} \hat{z} \right| = \frac{1}{R}$$

$$\sqrt{\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2 + \left(\frac{\partial T}{\partial z}\right)^2} = \frac{1}{R}$$

Observe that since we are squaring the spatial derivatives, their sign will not matter.

$$\boxed{\left(\frac{\partial T}{\partial x}\right)^2 + \left(\frac{\partial T}{\partial y}\right)^2 + \left(\frac{\partial T}{\partial z}\right)^2 = \frac{1}{R^2}}$$

We will approximate these derivatives with finite-differences by fitting T to a discrete grid.

A Problem

We have assumed that $R > 0$.

The values of T are only known on one side of the surface front.

We call this the “upwind” side of the front and we must use upwind finite-differences to solve this problem.

Upwind Finite-Differences (1 of 3)

We can approximate T using either a forward or backward finite-difference, depending which way is upwind.

$$\frac{\partial T}{\partial x} \Big|_{i,j,k} \approx \begin{cases} D_{x-} T_{i,j,k} = \frac{T_{i,j,k} - T_{i-1,j,k}}{\Delta x} & \text{backward finite-difference} \\ D_{x+} T_{i,j,k} = \frac{T_{i+1,j,k} - T_{i,j,k}}{\Delta x} & \text{forward finite-difference} \end{cases}$$

Which one do we use?

Upwind Finite-Differences (2 of 3)

We select the finite-difference that utilizes the side from which the front is approaching most quickly. To determine that, we identify and examine four cases.

Case 1: $D_{x-} T_{i,j,k} < 0$ Front moving away on left side

Case 2: $D_{x-} T_{i,j,k} > 0$ Front approaching from left side

Case 3: $D_{x+} T_{i,j,k} < 0$ Front approaching from right side

Case 4: $D_{x+} T_{i,j,k} > 0$ Front moving away on right side

$$\frac{\partial T}{\partial x} \Big|_{i,j,k} \approx \begin{cases} D_{x-} T_{i,j,k} & \text{when } D_{x-} T_{i,j,k} > -D_{x+} T_{i,j,k} \\ D_{x+} T_{i,j,k} & \text{when } D_{x-} T_{i,j,k} < -D_{x+} T_{i,j,k} \end{cases}$$

Upwind Finite-Differences (3 of 3)

$$\left. \frac{\partial T}{\partial x} \right|_{i,j,k} \approx \begin{cases} D_{x-} T_{i,j,k} & \text{when } D_{x-} T_{i,j,k} > -D_{x+} T_{i,j,k} \\ D_{x+} T_{i,j,k} & \text{when } D_{x-} T_{i,j,k} < -D_{x+} T_{i,j,k} \end{cases}$$

Recall that we do not care about the sign of the finite-difference when solving the differential equation. That let's us evaluate the above according to

$$\left. \frac{\partial T}{\partial x} \right|_{i,j,k} \approx \max [D_{x-} T_{i,j,k}, -D_{x+} T_{i,j,k}]$$

What if the front is moving away on both sides? In this case we need to ignore both finite-differences.

$$\left. \frac{\partial T}{\partial x} \right|_{i,j,k} \approx \max [D_{x-} T_{i,j,k}, -D_{x+} T_{i,j,k}, 0]$$

Approximation of Governing Equation

Our governing equation was

$$\left(\frac{\partial T}{\partial x} \right)^2 + \left(\frac{\partial T}{\partial y} \right)^2 + \left(\frac{\partial T}{\partial z} \right)^2 = \frac{1}{R^2}$$

Approximating this equation with our upwind finite-differences, we get

$$\left(\max [D_{x-} T, -D_{x+} T, 0] \right)^2 + \left(\max [D_{y-} T, -D_{y+} T, 0] \right)^2 + \left(\max [D_{z-} T, -D_{z+} T, 0] \right)^2 = \frac{1}{R^2}$$

Update Equation (1 of 3)

We need to solve our finite-difference equation for $T_{i,j,k}$.

We must pull $T_{i,j,k}$ out of the max[] functions.

$$\begin{aligned} \frac{\partial T}{\partial x} \Big|_{i,j,k} &\approx \max[D_x T_{i,j,k}, -D_x T_{i,j,k}, 0] \\ &\approx \max\left[\frac{T_{i,j,k} - T_{i-1,j,k}}{\Delta x}, -\frac{T_{i+1,j,k} - T_{i,j,k}}{\Delta x}, 0\right] \\ &\approx \frac{\max[T_{i,j,k} - T_{i-1,j,k}, -T_{i+1,j,k} + T_{i,j,k}, 0]}{\Delta x} \\ &\approx \frac{T_{i,j,k} + \max[-T_{i-1,j,k}, -T_{i+1,j,k}, -T_{i,j,k}]}{\Delta x} \\ &\approx \frac{T_{i,j,k} - \min[T_{i-1,j,k}, T_{i+1,j,k}, T_{i,j,k}]}{\Delta x} \end{aligned}$$

$$\frac{\partial T}{\partial x} \Big|_{i,j,k} \approx \frac{T_{i,j,k}^{\text{new}} - \min[T_{i-1,j,k}^{\text{old}}, T_{i+1,j,k}^{\text{old}}, T_{i,j,k}^{\text{old}}]}{\Delta x}$$

37

Update Equation (2 of 3)

Our governing equation can now be written as

$$\left(\frac{T_{i,j,k}^{\text{new}} - \min[T_{i-1,j,k}^{\text{old}}, T_{i+1,j,k}^{\text{old}}, T_{i,j,k}^{\text{old}}]}{\Delta x}\right)^2 + \left(\frac{T_{i,j,k}^{\text{new}} - \min[T_{i,j,k-1}^{\text{old}}, T_{i,j,k+1}^{\text{old}}, T_{i,j,k}^{\text{old}}]}{\Delta y}\right)^2 + \left(\frac{T_{i,j,k}^{\text{new}} - \min[T_{i,j,k-1}^{\text{old}}, T_{i,j,k+1}^{\text{old}}, T_{i,j,k}^{\text{old}}]}{\Delta z}\right)^2 = \frac{1}{(R_{i,j,k})^2}$$

After some algebra, we write this express as a quadratic equation.

$$a(T_{i,j,k}^{\text{new}})^2 + b(T_{i,j,k}^{\text{new}}) + c = 0$$

$$a = \frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}$$

$$b = -2 \left[\frac{m_x|_{i,j,k}}{(\Delta x)^2} + \frac{m_y|_{i,j,k}}{(\Delta y)^2} + \frac{m_z|_{i,j,k}}{(\Delta z)^2} \right]$$

$$c = \left(\frac{m_x|_{i,j,k}}{\Delta x}\right)^2 + \left(\frac{m_y|_{i,j,k}}{\Delta y}\right)^2 + \left(\frac{m_z|_{i,j,k}}{\Delta z}\right)^2 - \frac{1}{(R_{i,j,k})^2}$$

$$m_x|_{i,j,k} = \min[T_{i-1,j,k}^{\text{old}}, T_{i+1,j,k}^{\text{old}}, T_{i,j,k}^{\text{old}}]$$

$$m_y|_{i,j,k} = \min[T_{i,j,k-1}^{\text{old}}, T_{i,j,k+1}^{\text{old}}, T_{i,j,k}^{\text{old}}]$$

$$m_z|_{i,j,k} = \min[T_{i,j,k-1}^{\text{old}}, T_{i,j,k+1}^{\text{old}}, T_{i,j,k}^{\text{old}}]$$

38

Update Equation (3 of 3)

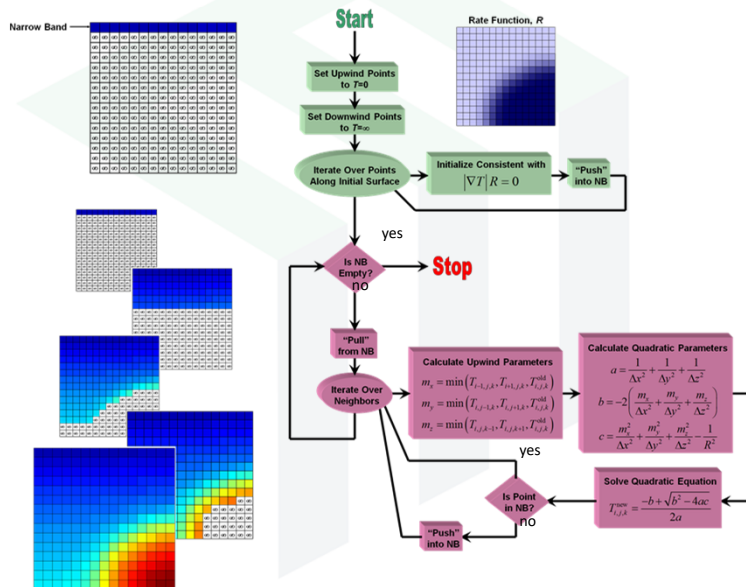
The solution to the quadratic equation is

$$T_{i,j,k}^{new} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

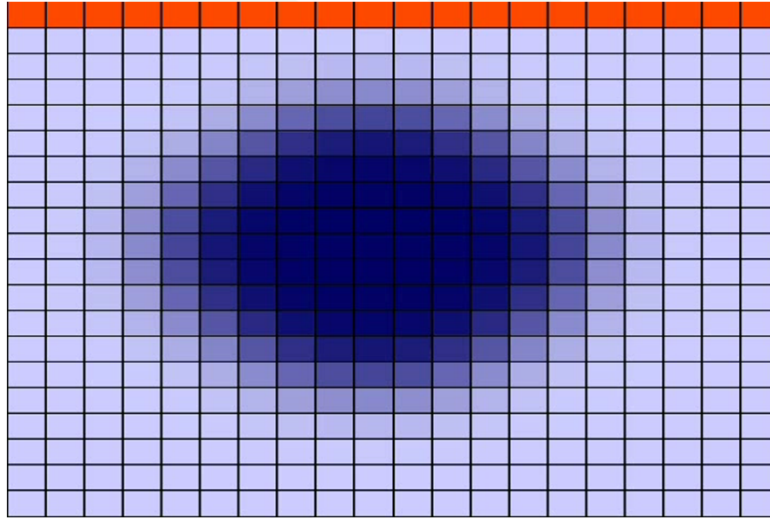
We choose the positive root to ensure that our time values are always positive.

$$T_{i,j,k}^{new} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Algorithm for the Fast Marching Method



Movie of Fast Marching Method



41

Min-Heap Data Structure

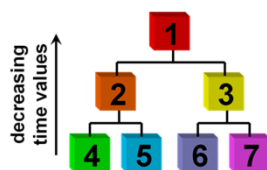
By far, the most time-consuming task is finding the grid point in the “narrow band” with the smallest associated time value.

The key to a fast and efficient FMM code is to quickly find this point.

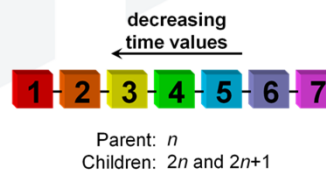
A good approach is to store the narrow band grid points in a min-heap data structure, which is a form of a binary tree, but it can be stored in a linear array without the need of data pointers.

It is ordered so that the children of any element always have an equal or greater value associated with it.

Binary Tree Interpretation

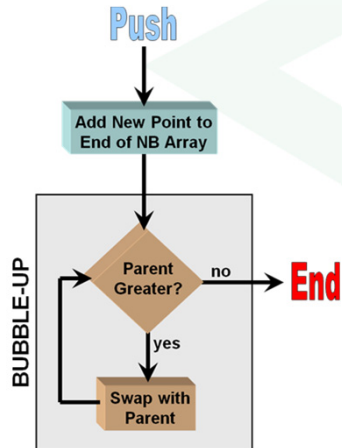


Linear Array (Actual Implementation)



42

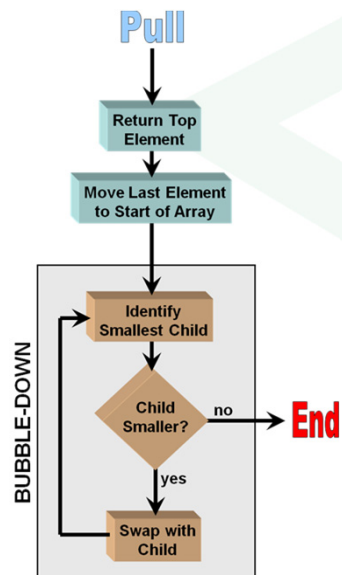
Push Operation



The push algorithm is used to add an element to the tree.

This algorithm ensures that element #1 at the top of the tree will have the smallest value. No searching for it is even necessary!

Pull Operation



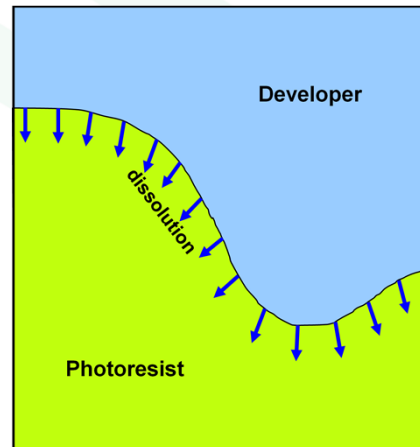
The pull algorithm removes the top element from the tree and adjusts the remaining elements.

Example – Developing Photoresists

What is Developing?

The developing process dissolves soluble photoresist into a developer solution leaving behind the insoluble portions.

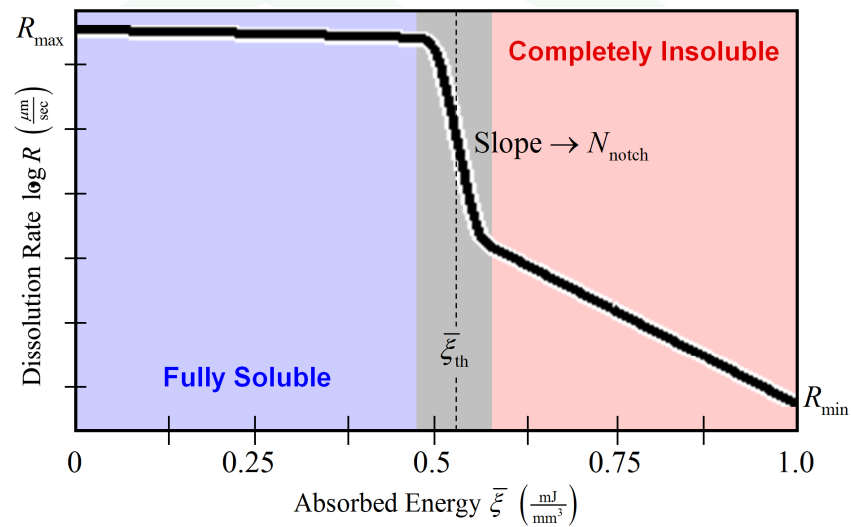
It is the partially soluble photoresist that must be treated correctly to accurately predict geometry.



45

Example – Developing Photoresists

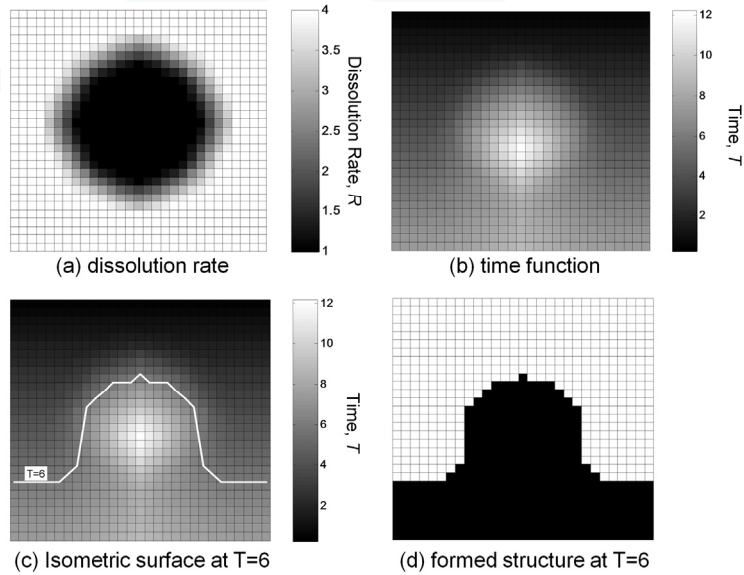
Solubility of a Photoresist



46

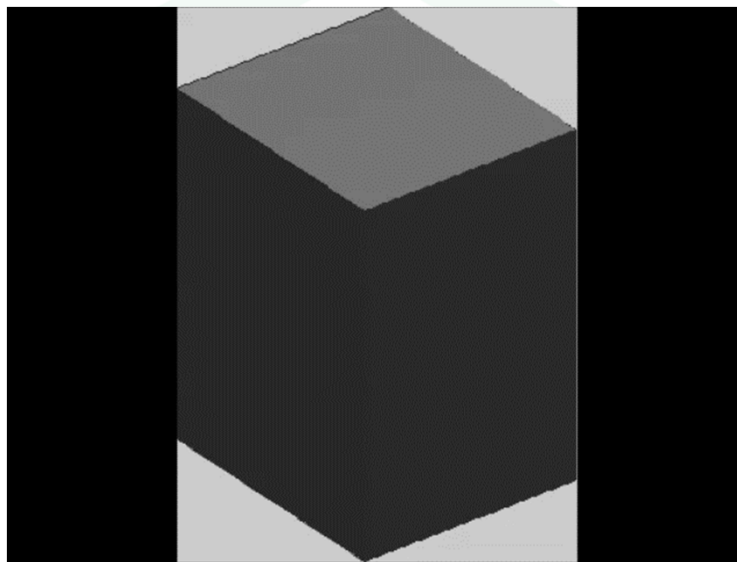
Example – Developing Photoresists

Simulating Developing with FMM

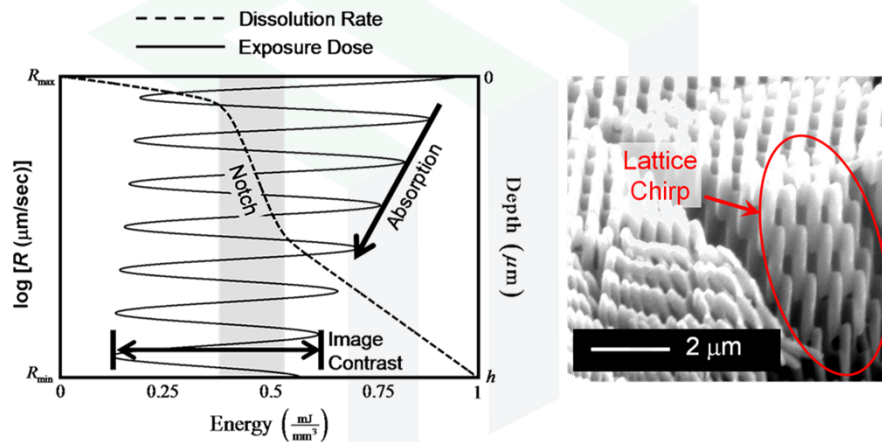


Example – Developing Photoresists

3D Movie of Developing

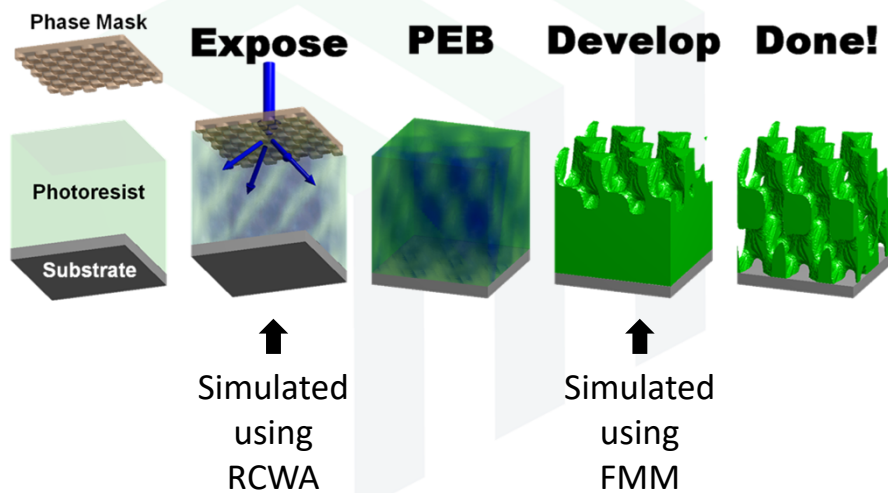


Example – Developing Photoresists *Simulating Holographic Lithography*



49

Example – Developing Photoresists *Simulating Near-Field Nano-Patterning*



50

Example – Developing Photoresists

Movie of Near-Field Nano-Patterning

