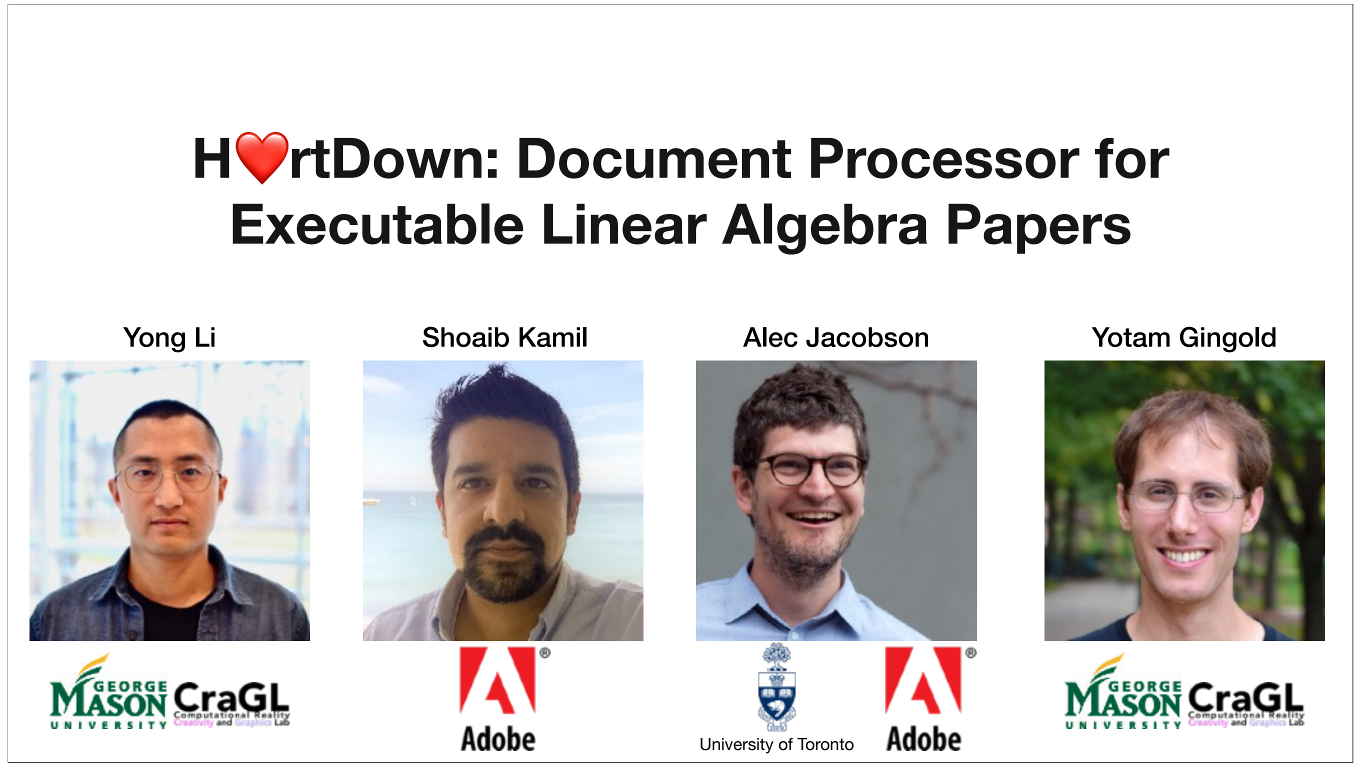


Thank you. This talk should be given by my PhD student Yong Li. He couldn't be here today for visa reasons.

%%%

Welcome to the talk, my name is Yong Li, I'm a PhD student in George Mason University.
We create a document processor called H❤️rtDown for executable linear algebra papers.

This is a joint work with
Dr. Shoaib Kamil from Adobe Research,
Prof. Alec Jacobson from University of Toronto and Adobe Research,
And my advisor Prof. Yotam Gingold from George Mason University.

H❤️rtDown is an environment for reading and writing scientific documents. Instead of writing formulas in latex, you write them in I❤️LA.

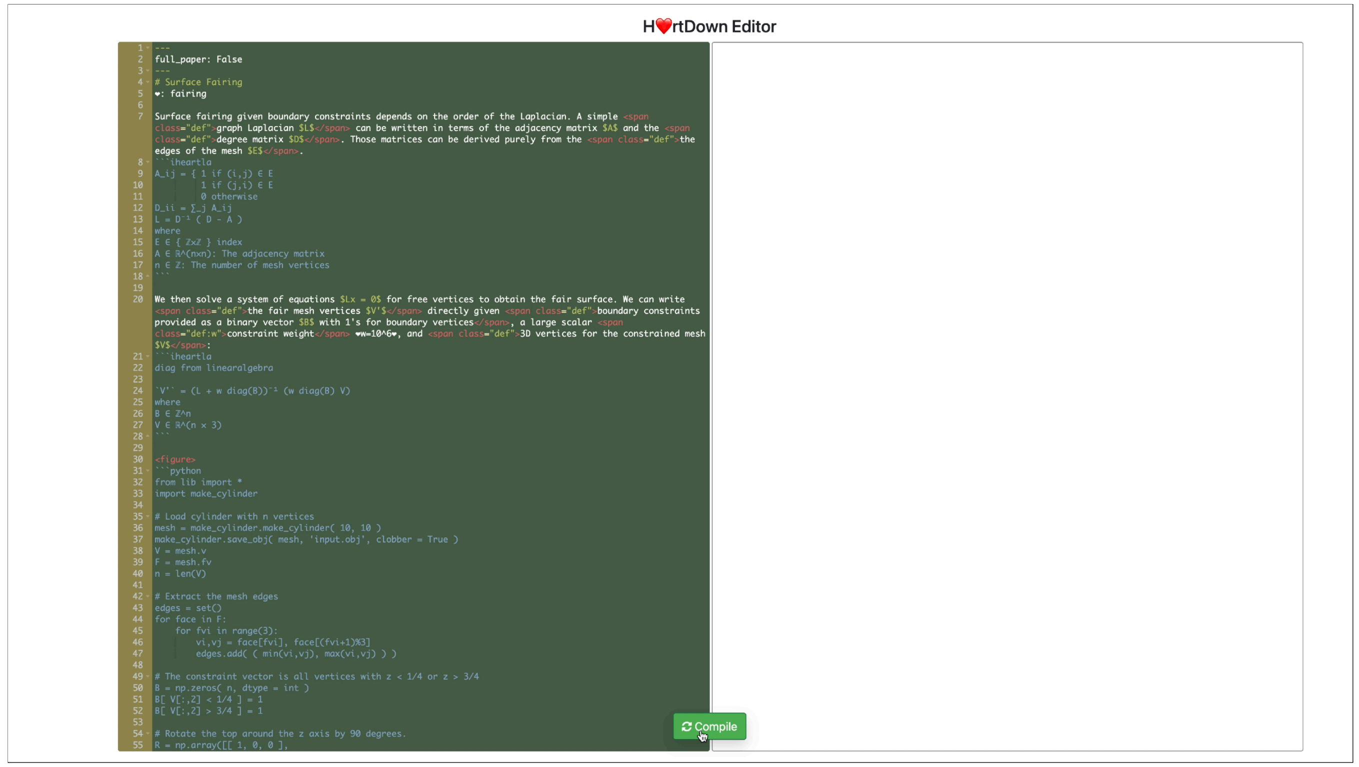
````
# Surface Fairing
❤: fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A simple <span
class="def">graph Laplacian $L$</span> can be written in terms of the adjacency matrix $A$ and the <span
class="def">degree matrix $D$</span>. Those matrices can be derived purely from the <span class="def">the
edges of the mesh $E$</span>.
```iheartla
A_ij = { 1 if (i,j) ∈ E
         1 if (j,i) ∈ E
         0 otherwise
D_ii = ∑_j A_ij
L = D⁻¹ ( D - A )
where
E ∈ { ℤxℤ } index
A ∈ ℝ^(nxn): The adjacency matrix
n ∈ ℤ: The number of mesh vertices
```

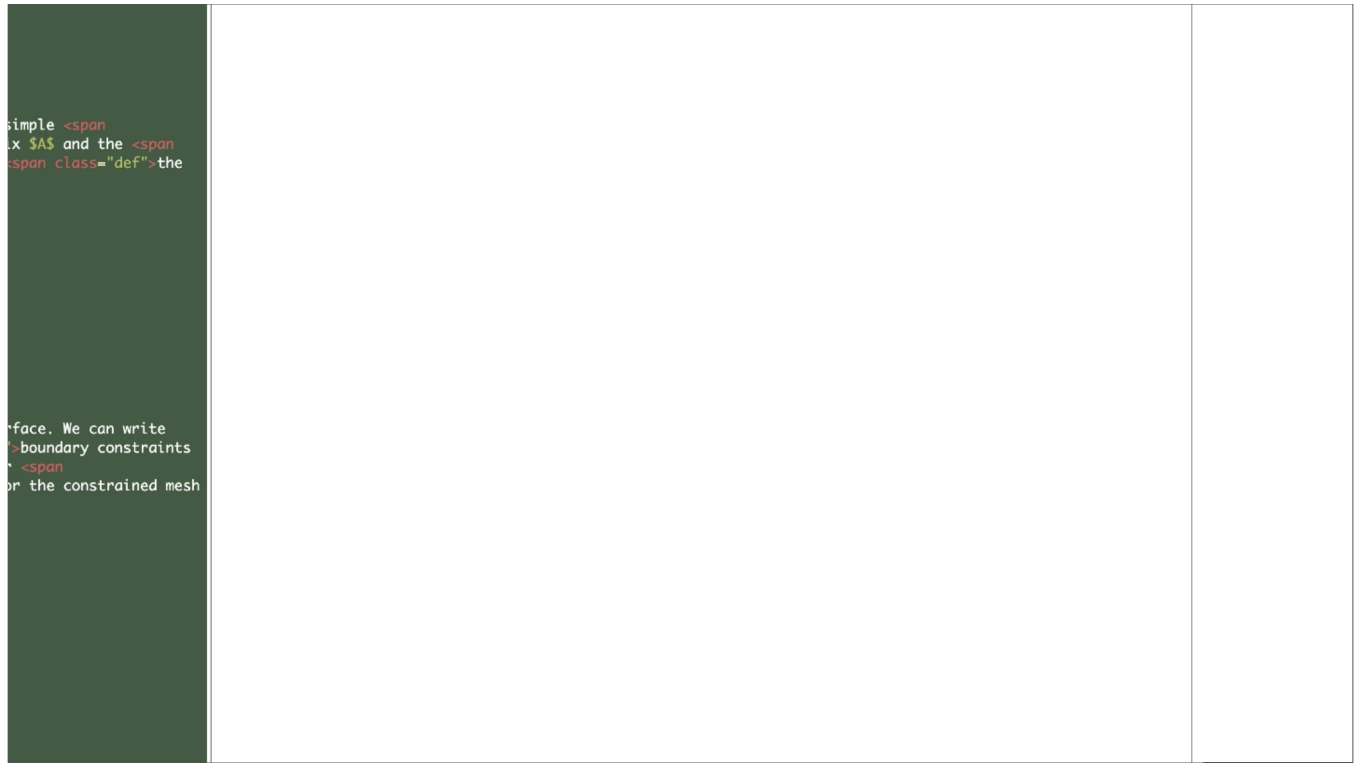
We then solve a system of equations $Lx = 0$ for free vertices to obtain the fair surface. We can write
<span class="def">the fair mesh vertices $V'$</span> directly given <span class="def">boundary constraints
provided as a binary vector $B$ with 1's for boundary vertices</span>, a large scalar <span
class="def:w">constraint weight</span> ❤w=10^6❤, and <span class="def">3D vertices for the constrained mesh
$V$</span>:
```iheartla
diag from linearalgebra

`V'` = (L + w diag(B))⁻¹ (w diag(B) V)
where
````

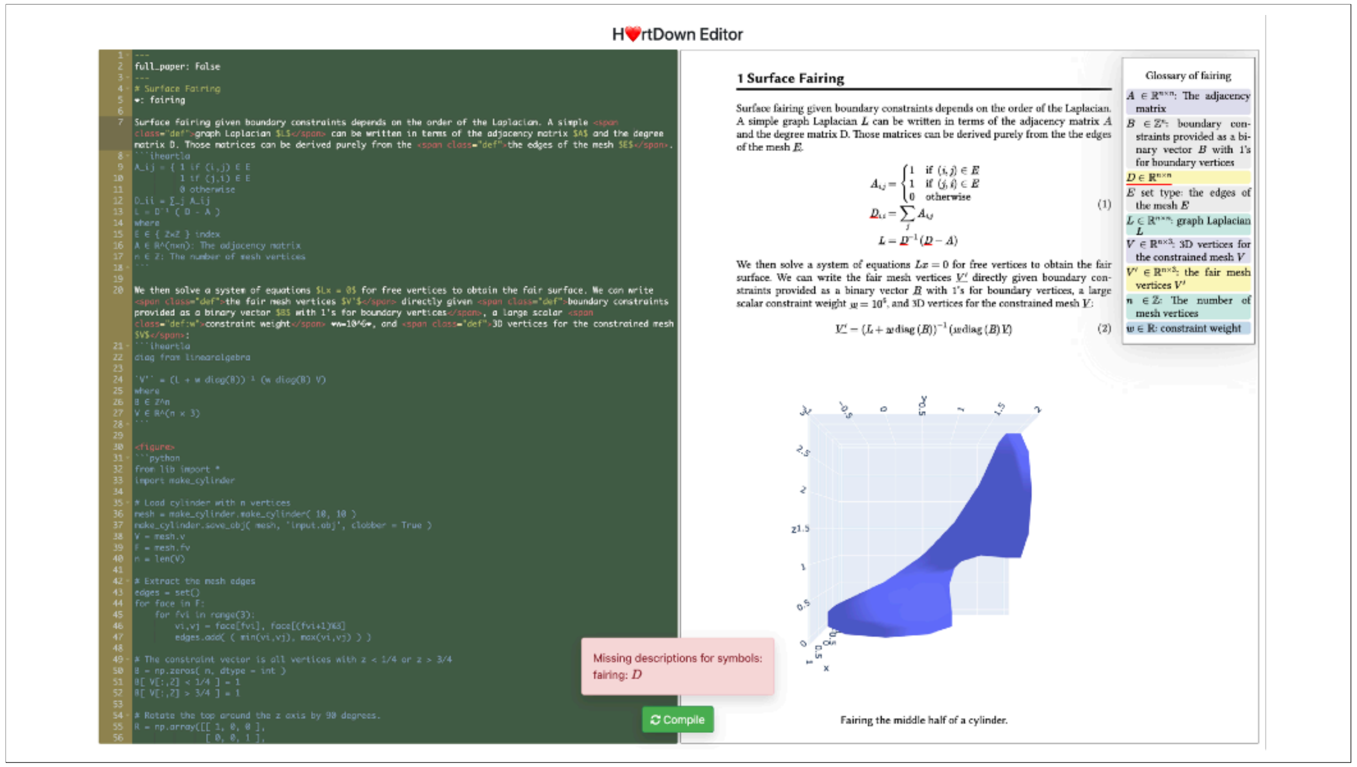
H❤️rtDown is an environment for reading and writing scientific documents. Instead of writing formulas in latex, you write them in I❤️LA.
````



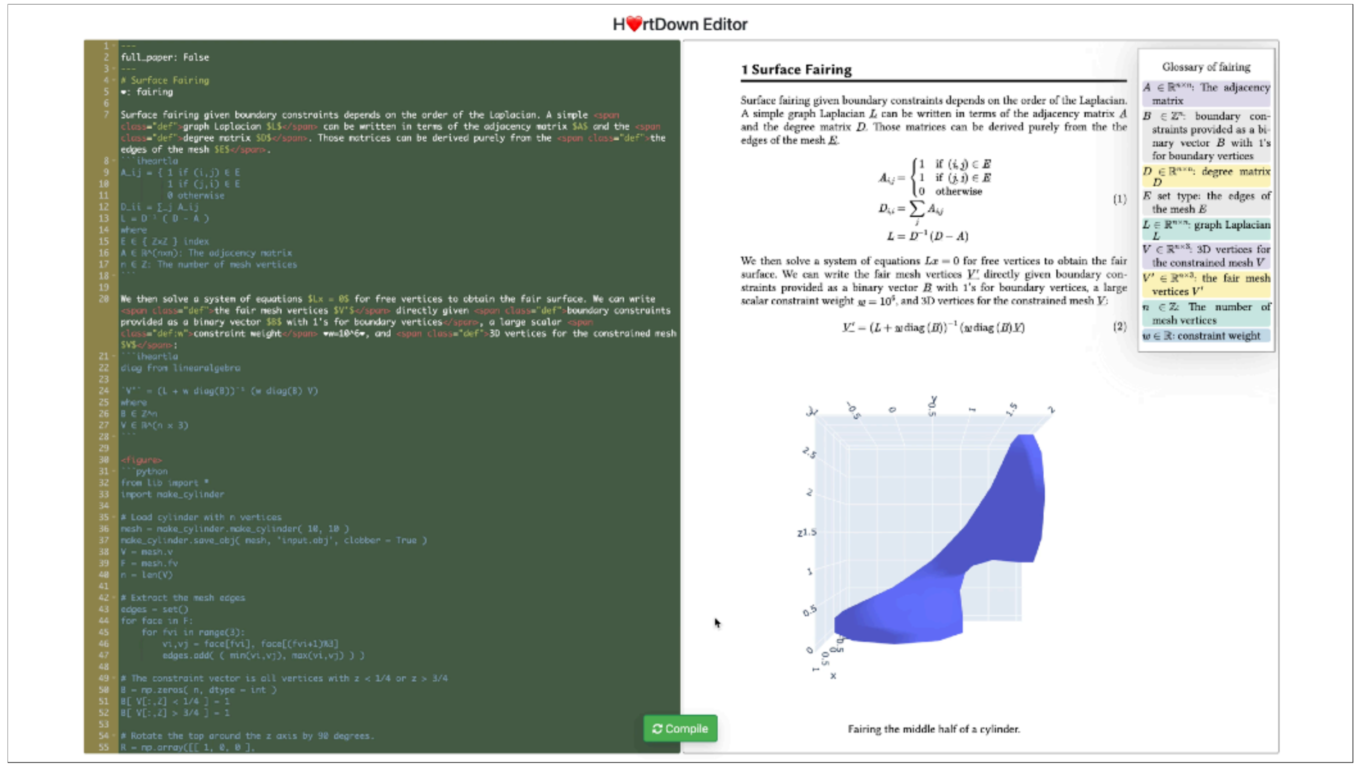
By compiling the math, H❤️rtDown augments the paper with clickable definitions



By compiling the math, H❤️rtDown augments the paper with clickable definitions

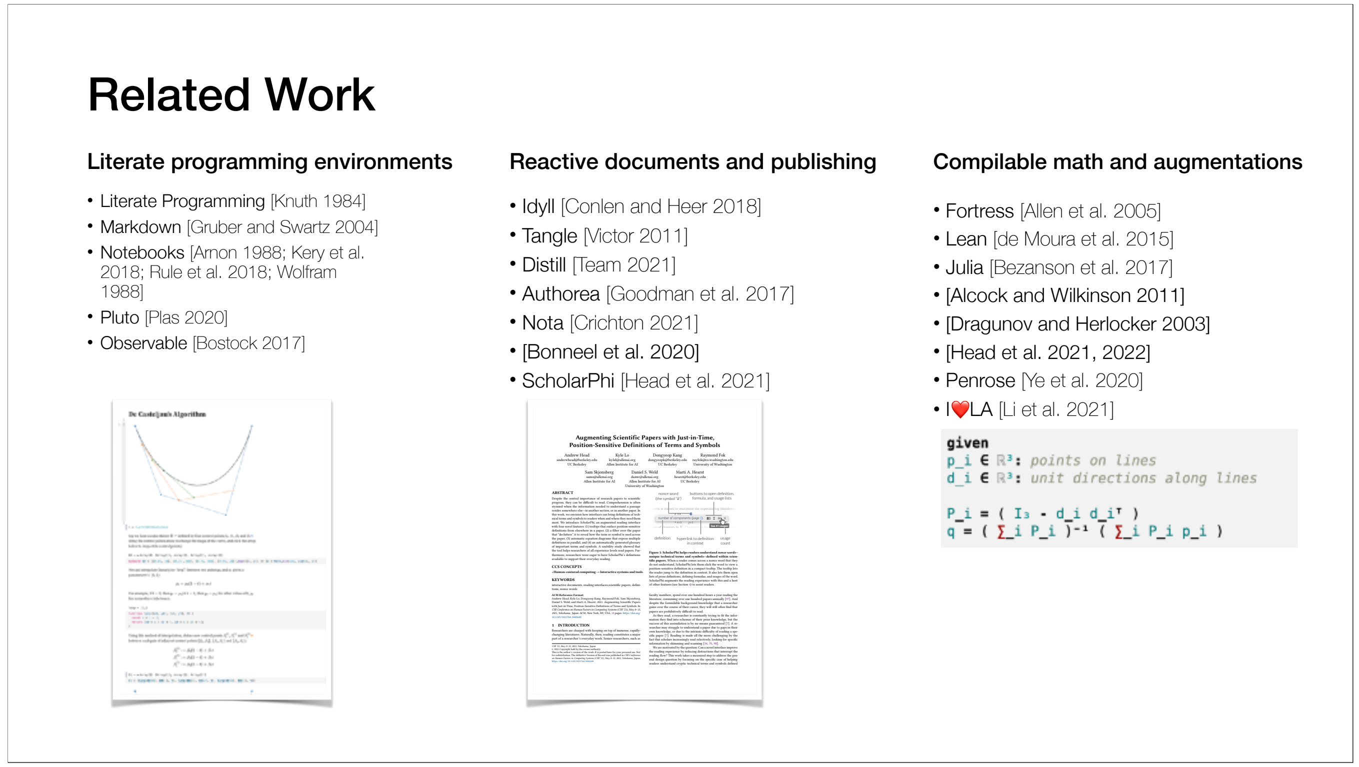


and warns you when you've forgotten to describe a variable



The compiled math can be used to generate figures.

Updating the formula will change both the typeset math and the figure.



There is related work on …

* Literate programming environments, which duplicate math and code. H❤️rtDown avoids this duplication.

* H❤️rtDown is related to reactive documents and publishing. H❤️rtDown focuses on helping scientific document users correctly author, read, and experiment with mathematical formulas.

* We make use of languages for compiling math and ideas for augmenting math.

Design Goals

- Support **authoring**, **reading**, and making use of (**experimenting** with)
  - Correct and reproducible documents
  - Minimal authoring overhead
- **Ecological compatibility**
  - Don’t change **what** authors put in papers (prose, math, figures, tables)
  - Minimal changes to **how** they write
    - Plain text documents

We have two design goals.
The first is to support authoring, reading, and making use of (experimenting with)
Correct and reproducible documents
With Minimal authoring overhead

The second is to provide ecological compatibility which means
We don’t want to change/restrict what authors put in papers (prose, math, figures, tables)
And we want minimal changes to how they write, e.g.: We prefer plain text documents

To inform our design, we analyzed 156 papers from the SIGGRAPH North America 2020 Technical Papers program.

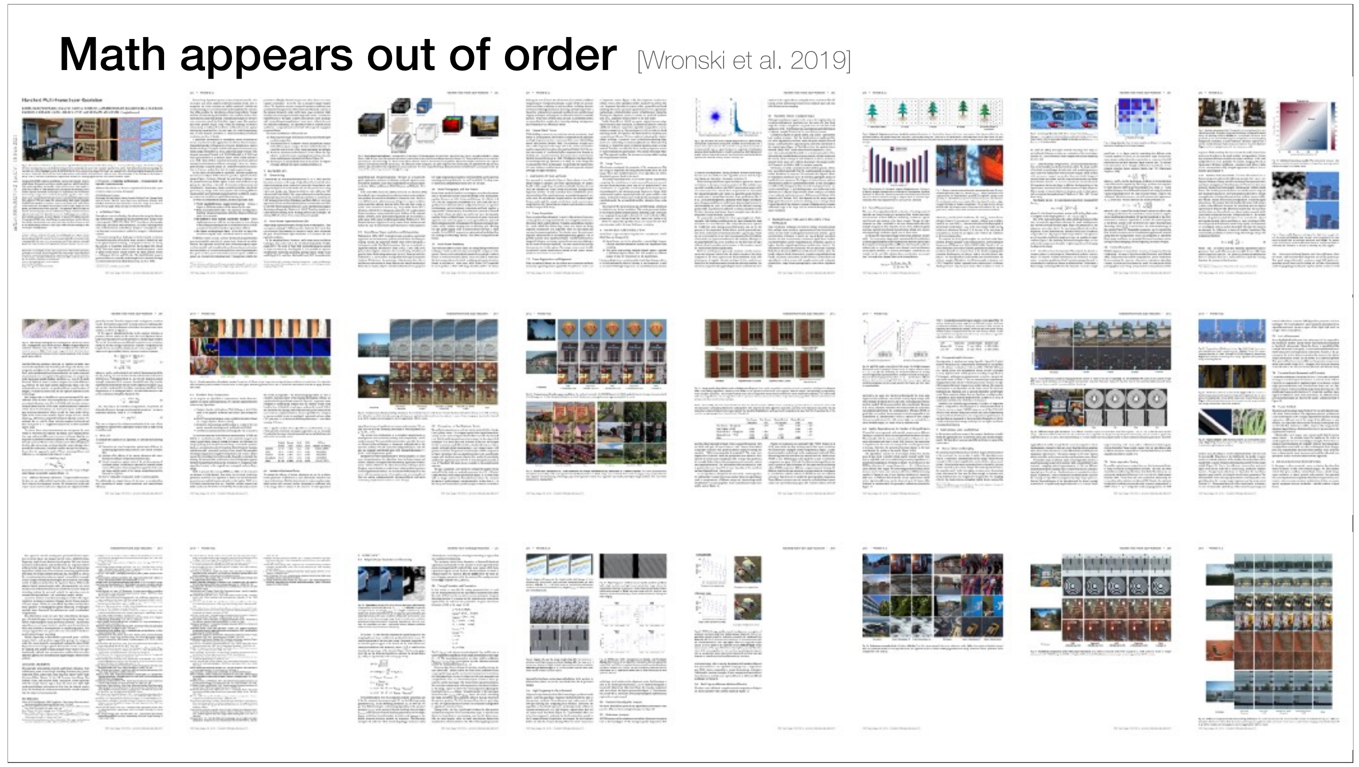
% collecting both quantitative and qualitative observations.

Formative Study

- All appear to be written using LaTeX.
- Observations:
  I. Prose organizes the document, interleaved with math.
  II. Math appears out of order. Symbols used before defined.

Our formative study found that
All papers appear to be written using LaTeX.
Other Observations include
(I) Prose organizes the document. Mathematical expressions appear between paragraphs of prose or inline.
(II) Math symbols are often used before they are defined, as determined by the prose.

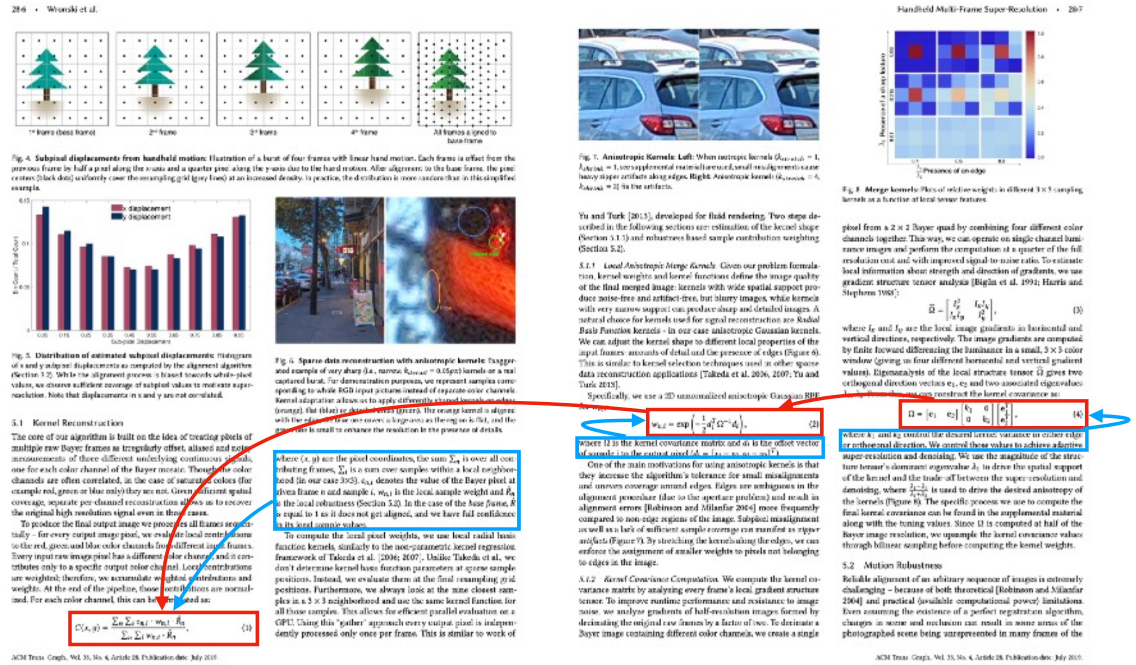
Equations in papers often depend on each other.



Let’s see a typical example from this paper. (# this paper is not from SIGGRAPH 2020, but it better demonstrates the dependence)

If we zoom in the six and seventh pages.

Math appears out of order [Wronski et al. 2019]

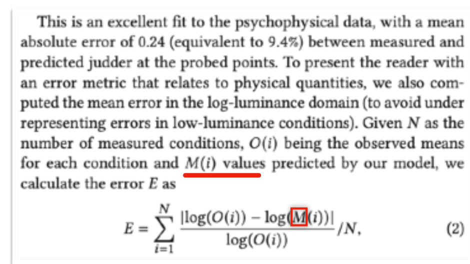


We can see that **(*)** equation one defines a function C that uses w which is **(*)** defined in the second equation, meanwhile, omega in the second equation is defined in **(*)** the fourth equation.

For each equation, there’s a prose block **(*) after** the equation describing all the symbols in that equation

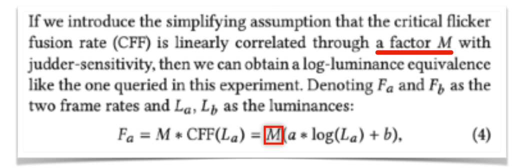
Formative Study

- All appear to be written using LaTeX.
- Observations:
  - I. Prose organizes the document, interleaved with math.
  - II. Math appears out of order. Symbols used before defined.
  - III. Symbols re-used in different contexts.



This is an excellent fit to the psychophysical data, with a mean absolute error of 0.24 (equivalent to 9.4%) between measured and predicted judder at the probed points. To present the reader with an error metric that relates to physical quantities, we also computed the mean error in the log-luminance domain (to avoid under representing errors in low-luminance conditions). Given N as the number of measured conditions, $O(i)$ being the observed means for each condition and $M(i)$ values predicted by our model, we calculate the error E as

$$E = \sum_{i=1}^{N} \frac{|\log(O(i)) - \log(M(i))|}{\log(O(i))} / N, \quad (2)$$



If we introduce the simplifying assumption that the critical flicker fusion rate (CFF) is linearly correlated through a factor M with judder-sensitivity, then we can obtain a log-luminance equivalence like the one queried in this experiment. Denoting F_a and F_b as the two frame rates and L_a, L_b as the luminances:

$$F_a = M * \mathrm{CFF}(L_a) = M(a * \log(L_a) + b), \quad (4)$$

[Chapiro et al. 2019]

We also found that

(I) Symbols may be re-used, but the different context is clear to the reader.

For example, the M symbols have different meanings in these equations.

Formative Study

- All appear to be written using LaTeX.
- Observations:
  I. Prose organizes the document, interleaved with math.
  II. Math appears out of order. Symbols used before defined.
  III. Symbols re-used in different contexts.
  IV. Symbol appears in executable formulas and non-executable derivations.

We now consider the nine possible deformations $\tilde{\boldsymbol{u}}_\varepsilon^{ij}$ generated by setting $\boldsymbol{f} = \boldsymbol{e}_i$ and $\boldsymbol{g} = \boldsymbol{e}_j$ for every pair (i, j), where the vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ form an orthonormal bases spanning $\mathbb{R}^3$. Due to superposition, we can linearly combine $\tilde{\boldsymbol{u}}_\varepsilon^{ij}$ with scalar coefficients F_{ij}, and obtain a matrix-driven solution of (2) of the form

$$\tilde{\boldsymbol{u}}_\varepsilon(\boldsymbol{r}) = \sum_{ij} F_{ij}\, \boldsymbol{e}_j \cdot \nabla(\mathcal{K}_\varepsilon(\boldsymbol{r})\, \boldsymbol{e}_i) = \nabla \mathcal{K}_\varepsilon(\boldsymbol{r}) : \boldsymbol{F}, \tag{12}$$

where $\boldsymbol{F} = [F_{ij}]$ is a 3×3 force matrix, and the symbol $:$ indicates the double contraction of $\boldsymbol{F}$ to the third-order tensor $\nabla\mathcal{K}_\varepsilon(\boldsymbol{r})$, thus returning a vector. Similarly, we can write the body load that generates

By computing the spatial derivatives of $\boldsymbol{u}_\varepsilon$, we obtain the displacement field $\tilde{\boldsymbol{u}}_\varepsilon(\boldsymbol{r})$ in terms of the force matrix $\boldsymbol{F}$:

$$\tilde{\boldsymbol{u}}_\varepsilon(\boldsymbol{r}) = -a\left(\frac{1}{r_\varepsilon^3} + \frac{3\varepsilon^2}{2r_\varepsilon^5}\right)\boldsymbol{F}\boldsymbol{r} + b\left[\frac{1}{r_\varepsilon^3}(\boldsymbol{F} + \boldsymbol{F}^t + \mathrm{tr}(\boldsymbol{F})\boldsymbol{I}) - \frac{3}{r_\varepsilon^5}(\boldsymbol{r}^t\boldsymbol{F}\boldsymbol{r})\,\boldsymbol{I}\right]\boldsymbol{r}. \tag{14}$$

[De Goes and James 2017]

(I) A symbol may appear in both derivations and executable formulas.

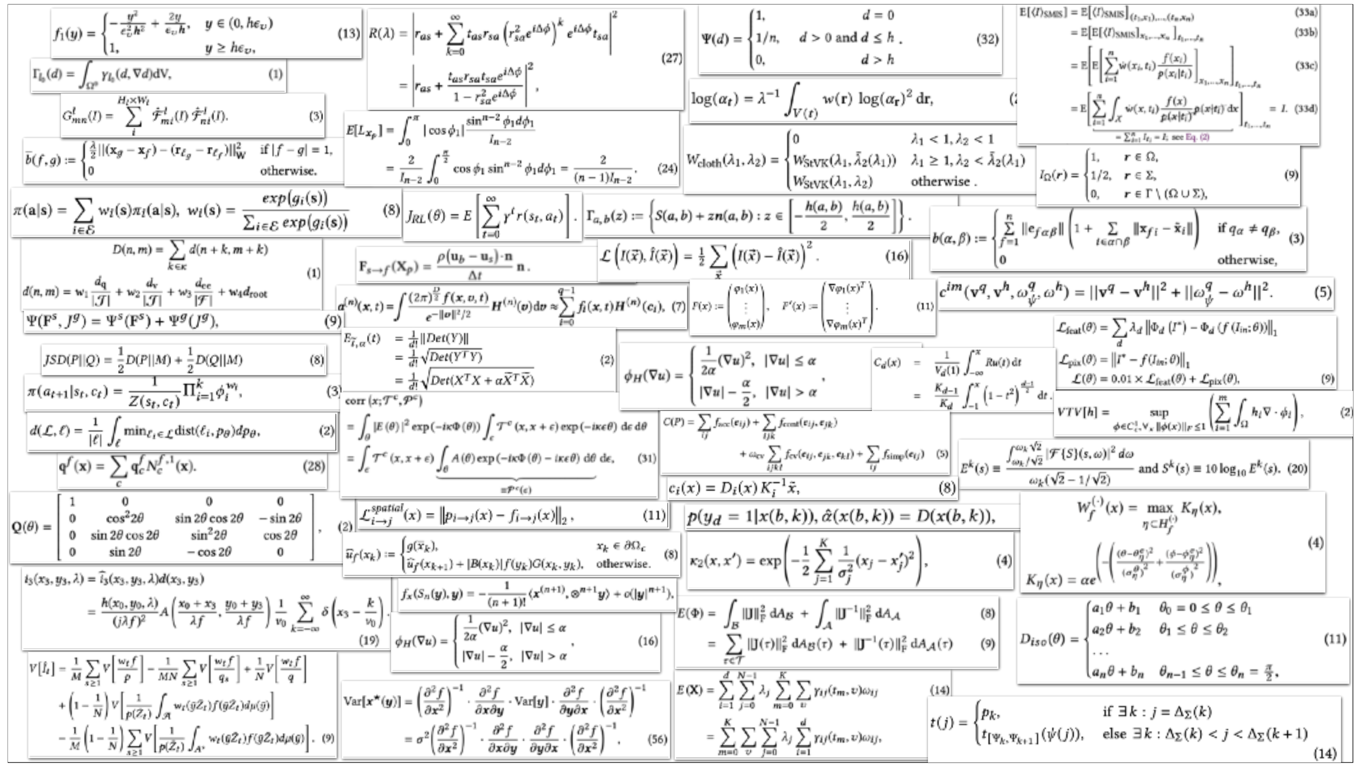
For example, the equation 12 is the derivation for the function **u** while equation 14 is executable.

Formative Study

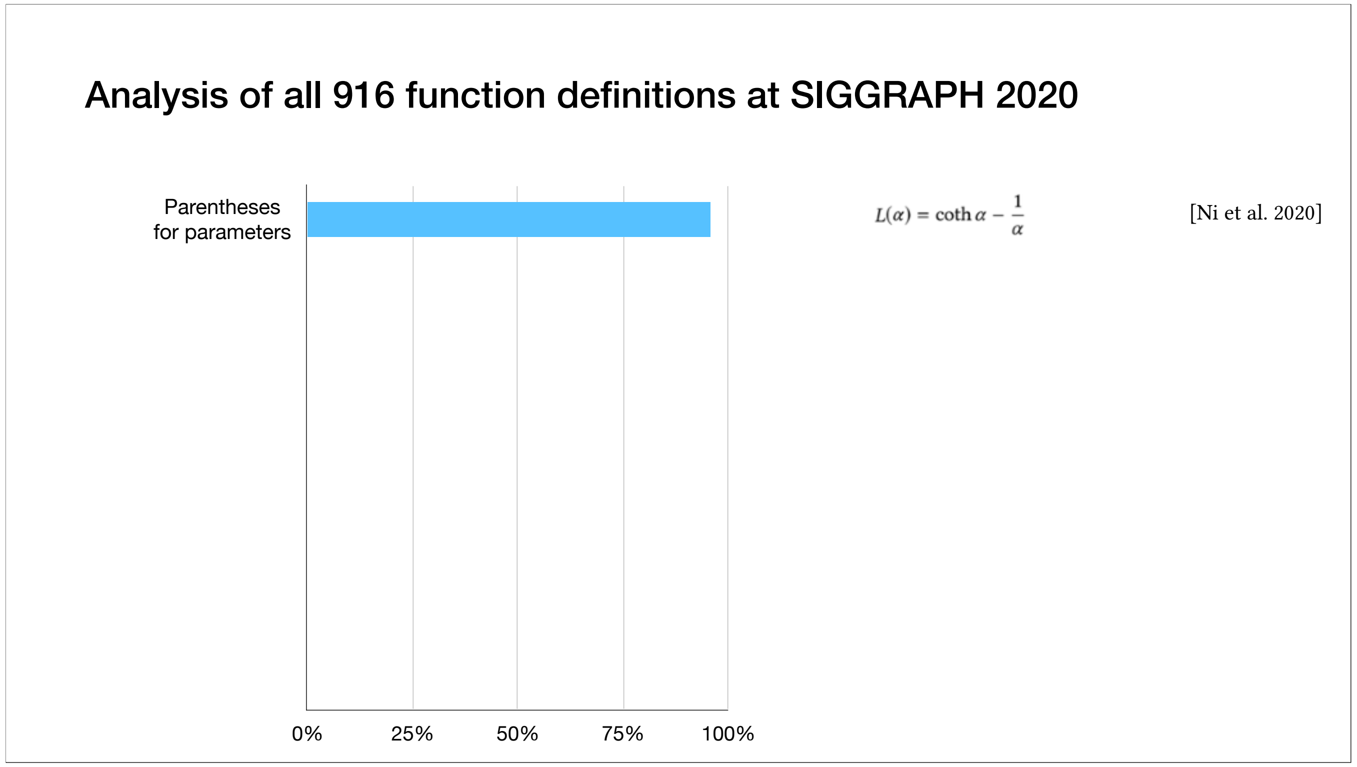
- All appear to be written using LaTeX.
- Observations:
  - I. Prose organizes the document, interleaved with math.
  - II. Math appears out of order. Symbols used before defined.
  - III. Symbols re-used in different contexts.
  - IV. Symbol appears in executable formulas and non-executable derivations.
  - V. Symbols and functions appear with conditional assignment.
  - VI. Functions have a variety of implied semantics for parameters and pre-computed symbols.

(I) Symbols and functions may be defined via conditional assignment, a simple form of control flow

(II) Functions make use of a variety of implied semantics for parameters



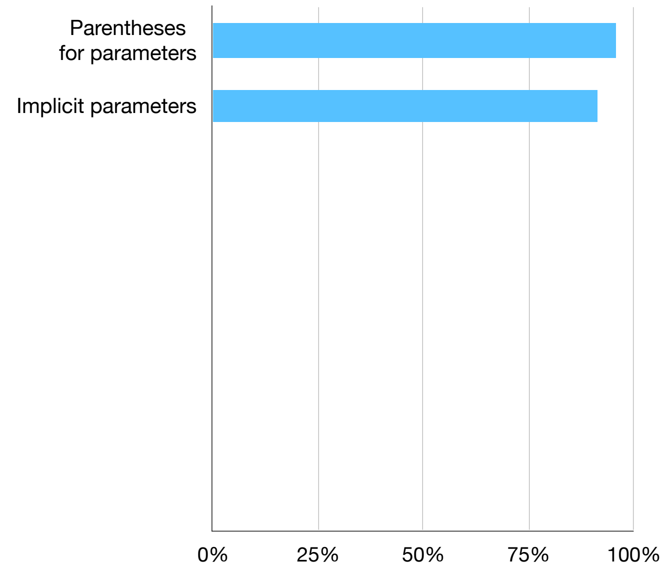
We quantitatively analyzed the 916 function definitions across the 156 SIGGRAPH papers.



We take an empiric approach to categorize these Equations.

Here is the overview of them.
96% use parentheses for parameters,

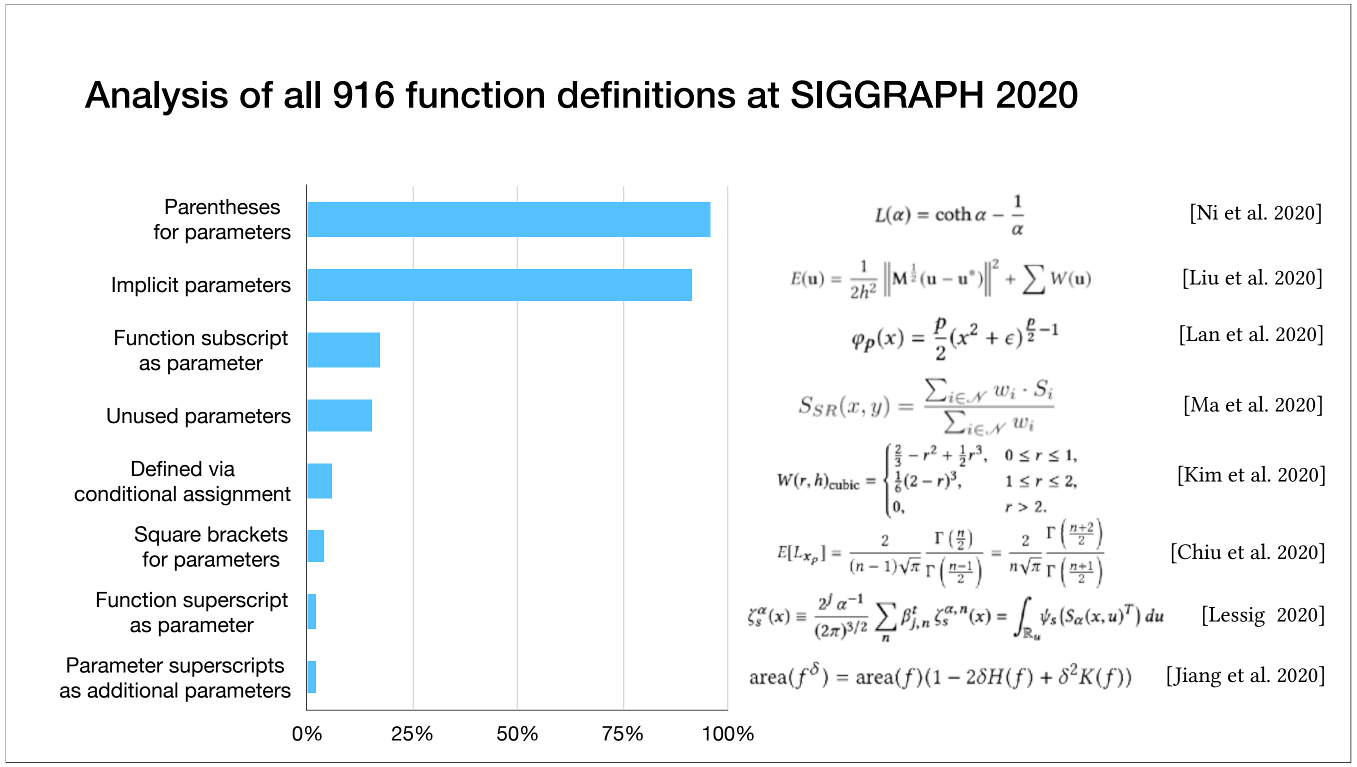
Analysis of all 916 function definitions at SIGGRAPH 2020



$$L(\alpha) = \coth\alpha - \frac{1}{\alpha}$$ [Ni et al. 2020]

$$E(\mathbf{u}) = \frac{1}{2h^2}\left\|\mathbf{M}^{\frac{1}{2}}(\mathbf{u} - \mathbf{u}^*)\right\|^2 + \sum W(\mathbf{u})$$ [Liu et al. 2020]

91% rely on implicit parameters,



We observed a variety of other less common semantics related to function parameters.

% 2% interpret the parameter superscripts as additional parameters,

Based on these findings, we extend the grammar and implementation of I❤️LA to include support for local functions

Formative Study

- All appear to be written using LaTeX.
- Observations:
  I. Prose organizes the document, interleaved with math.
  II. Math appears out of order. Symbols used before defined.
  III. Symbols re-used in different contexts.
  IV. Symbol appears in executable formulas and non-executable derivations.
  V. Symbols and functions appear with conditional assignment.
  VI. Functions have a variety of implied semantics for parameters and pre-computed symbols.
- Pseudocode sometimes present, compilable code isn't. No literate programs.

In addition, Pseudocode sometimes present while compilable code isn't. There's No literate programs.

H❤️rtDown Design: Authoring

- Context definition

~~~
# Surface Fairing
❤: fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A
simple <span class="def">graph Laplacian $L$</span> can be written in terms of the
adjacency matrix $A$ and the <span class="def">degree matrix $D$</span>. Those
matrices can be derived purely from the <span class="def">the edges of the mesh
$E$</span>.
```iheartla
A_ij = { 1 if (i,j) ∈ E
         1 if (j,i) ∈ E
         0 otherwise
D_ii = ∑_j A_ij
L = D⁻¹ ( D - A )
where
E ∈ { ℤxℤ } index
A ∈ ℝ^(nxn): The adjacency matrix
n ∈ ℤ: The number of mesh vertices
```

We then solve a system of equations $Lx = 0$ for free vertices to obtain the fair
surface. We can write <span class="def">the fair mesh vertices $V'$</span> directly
given <span class="def">boundary constraints provided as a binary vector $B$ with
1's for boundary vertices</span>, a large scalar <span class="def:w">constraint
weight</span> ❤w=10^6❤, and <span class="def">3D vertices for the constrained mesh
$V$</span>:
~~~

We design H❤️rtDown based on our formative study.

Just as in LaTeX or many other Markdown formats, the prose is written as plain text with occasional markup commands

(*) Authors must declare a context for their symbols.
The context disambiguates symbol reuse (and corresponds to our concept of modules).

###

This allows better symbol and formula re-use

Later context declarations override earlier declarations.

H❤️rtDown Design: Authoring

- Prose descriptions

````
# Surface Fairing
❤: fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A
simple <span class="def">graph Laplacian $L$</span> can be written in terms of the
adjacency matrix $A$ and the <span class="def">degree matrix $D$</span>. Those
matrices can be derived purely from the <span class="def">the edges of the mesh
$E$</span>.
```iheartla
A_ij = { 1 if (i,j) ∈ E
         1 if (j,i) ∈ E
         0 otherwise
D_ii = ∑_j A_ij
L = D⁻¹ ( D - A )
where
E ∈ { ℤxℤ } index
A ∈ ℝ^(nxn): The adjacency matrix
n ∈ ℤ: The number of mesh vertices
```

We then solve a system of equations $Lx = 0$ for free vertices to obtain the fair
surface. We can write <span class="def">the fair mesh vertices $V'$</span> directly
given <span class="def">boundary constraints provided as a binary vector $B$ with
1's for boundary vertices</span>, a large scalar <span class="def:w">constraint
weight</span> ❤w=10^6❤, and <span class="def">3D vertices for the constrained mesh
$V$</span>:
````

One appearance of a symbol in the prose deserves special attention: the text describing the symbol.

Detecting the span of this description cannot be accurately automated, so we require authors to annotate such spans.

H❤️rtDown Design: Authoring

- Executable mathematical expressions

````
# Surface Fairing
❤: fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A
simple <span class="def">graph Laplacian $L$</span> can be written in terms of the
adjacency matrix $A$ and the <span class="def">degree matrix $D$</span>. Those
matrices can be derived purely from the <span class="def">the edges of the mesh
$E$</span>.
```iheartla
A_ij = { 1 if (i,j) ∈ E
         1 if (j,i) ∈ E
         0 otherwise
D_ii = ∑_j A_ij
L = D⁻¹ ( D - A )
where
E ∈ { ℤxℤ } index
A ∈ ℝ^(nxn): The adjacency matrix
n ∈ ℤ: The number of mesh vertices
```

We then solve a system of equations $Lx = 0$ for free vertices to obtain the fair
surface. We can write <span class="def">the fair mesh vertices $V'$</span> directly
given <span class="def">boundary constraints provided as a binary vector $B$ with
1's for boundary vertices</span>, a large scalar <span class="def:w">constraint
weight</span> ❤w=10^6❤, and <span class="def">3D vertices for the constrained mesh
$V$</span>:
````

Authors can write executable mathematical expressions in different I❤️LA blocks and inline I❤️LA formula. We chose I❤️LA since it resembles equations in papers and can generate latex and code for different backends.

% I❤️LA requires type declarations for all symbols not appearing on the left-hand side of an equals sign

H❤️rtDown Design: Authoring

- I❤️LA extensions
  - Local function support
  - Symbol def-use analysis
  - Modules
  - MathJax output includes metadata

We extended I❤️LA with new language features.

We add local function support based on the formative study.
In order to handle Math appearing out of order, we add symbol def-use analysis
We add Modules to support different contexts
We also modify MathJax output to include metadata for our viewing environment

H❤️rtDown Design: Authoring

- Figures

~~~~
<figure>
```python
from lib import *
import make_cylinder

# Load cylinder with n vertices
mesh = make_cylinder.make_cylinder( 10, 10 )
make_cylinder.save_obj( mesh, 'input.obj', clobber = True )
V = mesh.v
F = mesh.fv
n = len(V)

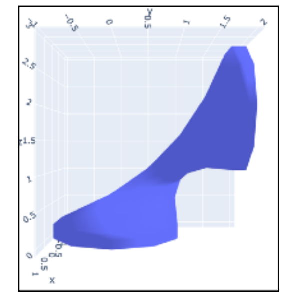
# Extract the mesh edges
edges = set()
for face in F:
    for fvi in range(3):
        vi,vj = face[fvi], face[(fvi+1)%3]
        edges.add( ( min(vi,vj), max(vi,vj) ) )

# The constraint vector is all vertices with z < 1/4 or z > 3/4
B = np.zeros( n, dtype = int )
B[ V[:,2] < 1/4 ] = 1
B[ V[:,2] > 3/4 ] = 1

# Rotate the top around the z axis by 90 degrees.
R = np.array([[ 1, 0, 0 ],
              [ 0, 0, 1 ],
              [ 0, -1, 0 ]])
for vi in np.where(V[:,2] > 3/4)[0]: V[vi] = R @ V[vi] + (0,1,2)

# Solve for new positions
result = fairing( E = edges, n = n, B = B, V = V )
mesh.v = result.V_apostrophe
make_cylinder.save_obj( mesh, 'solved.obj', clobber = True )

import plotly.graph_objects as go
fig = go.Figure(data=[go.Mesh3d(
    x=mesh.v[:,0], y=mesh.v[:,1], z=mesh.v[:,2],
    i=mesh.fv[:,0], j=mesh.fv[:,1], k=mesh.fv[:,2]
    )])
fig.update_layout( scene_camera={'eye':dict(x=2.5,y=0,z=0), 'up':dict(x=0,y=0,z=1)}, margin=dict(t=0, r=0, l=0, b=0) )
fig.write_html( 'cylinder.html' )
```
<img src="cylinder.html" alt="a fair cylinder surface">
<figcaption>Fairing the middle half of a cylinder.</figcaption>
</figure>
~~~~

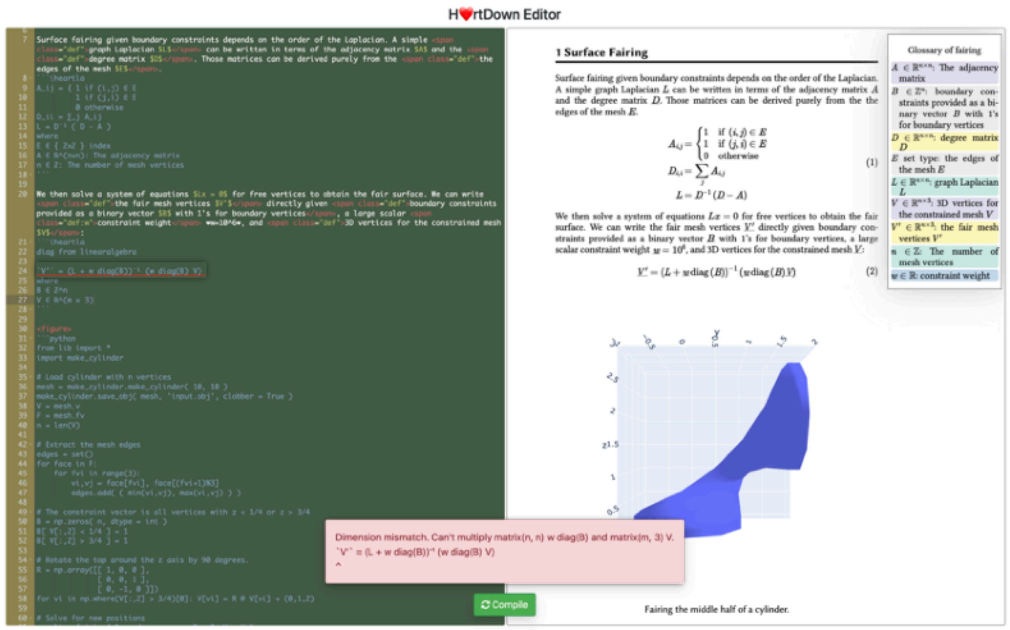


H❤️rtDown executes Python code blocks, which allows authors to generate figures programmatically

The Python code can access the compiled functionality of the document as a module.

% Authors can also edit I❤️LA formulas and Python code for figures directly in the viewer-side of the authoring environment.

H❤️rtDown Design: Author support

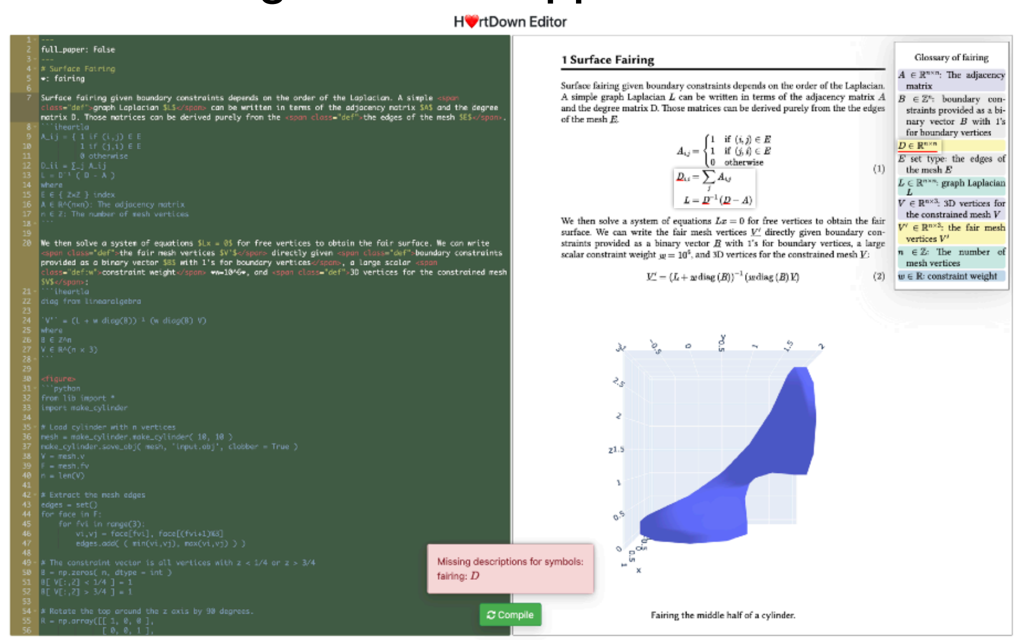


H❤️rtDown helps authors write correct math and complete prose.

Error messages appear whenever the user's formulas contain incompatible indices, dimensions, types or erroneous syntax.

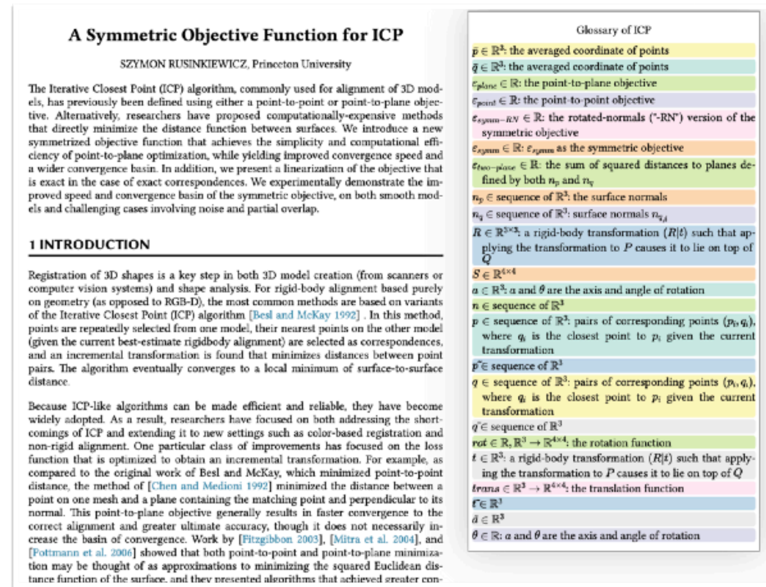
(*) The editor displays the I❤️LA compiler's error message and highlights the appropriate line in the source.

H❤️rtDown Design: Author support



When symbols are not described anywhere in the prose, they**(*)** appear with red underlines in the viewer.

H❤️rtDown Design: Reading Environment

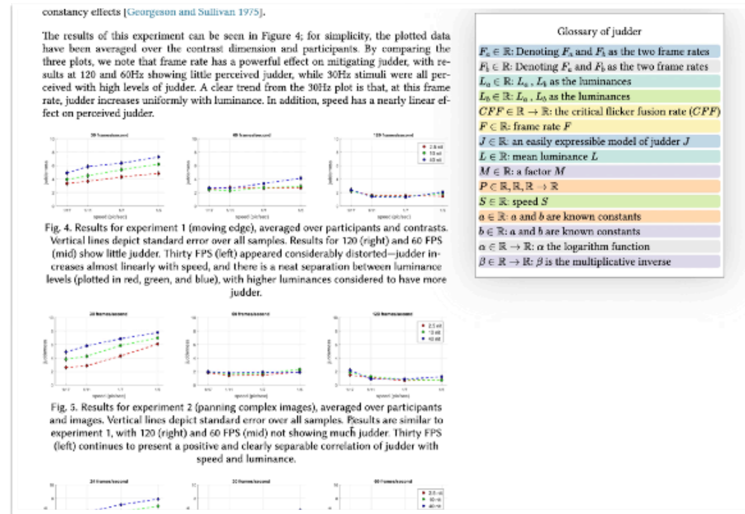


H❤️rtDown's paper reading environment provides several useful interactions that use the metadata.

Other enhanced reading environments could be created using the metadata H❤️rtDown generates. In fact, our augmentations were inspired by the ScholarPhi reading environment [Head et al. 2021].

H❤️rtDown Design: Reading Environment

- Glossary



constancy effects [Georgeson and Sullivan 1975].

The results of this experiment can be seen in Figure 4; for simplicity, the plotted data have been averaged over the contrast dimension and participants. By comparing the three plots, we note that frame rate has a powerful effect on mitigating judder, with results at 120 and 60Hz showing little perceived judder, while 30Hz stimuli were all perceived with high levels of judder. A clear trend from the 30Hz plot is that, at this frame rate, judder increases uniformly with luminance. In addition, speed has a nearly linear effect on perceived judder.

Fig. 4. Results for experiment 1 (moving edge), averaged over participants and contrasts. Vertical lines depict standard error over all samples. Results for 120 (right) and 60 FPS (mid) show little judder. Thirty FPS (left) appeared considerably distorted—judder increases almost linearly with speed, and there is a neat separation between luminance levels (plotted in red, green, and blue), with higher luminances considered to have more judder.

Fig. 5. Results for experiment 2 (panning complex images), averaged over participants and images. Vertical lines depict standard error over all samples. Results are similar to experiment 1, with 120 (right) and 60 FPS (mid) not showing much judder. Thirty FPS (left) continues to present a positive and clearly separable correlation of judder with speed and luminance.

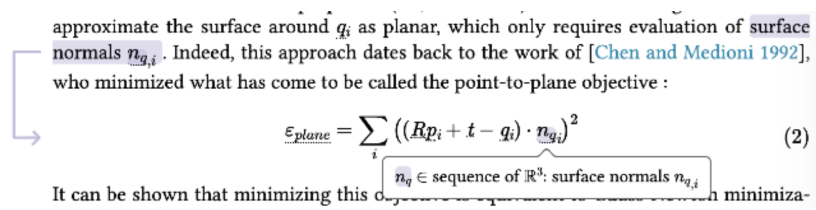
Glossary of judder

- $F_a \in \mathbb{R}$: Denoting F_a and F_b as the two frame rates
- $F_b \in \mathbb{R}$: Denoting F_a and F_b as the two frame rates
- $L_a \in \mathbb{R}$: L_a , L_b as the luminances
- $L_b \in \mathbb{R}$: L_a , L_b as the luminances
- $CFF \in \mathbb{R} \to \mathbb{R}$: the critical flicker fusion rate (CFF)
- $F \in \mathbb{R}$: frame rate F
- $J \in \mathbb{R}$: an easily expressible model of judder J
- $L \in \mathbb{R}$: mean luminance L
- $M \in \mathbb{R}$: a factor M
- $P \in \mathbb{R}, \mathbb{R}, \mathbb{R} \to \mathbb{R}$
- $S \in \mathbb{R}$: speed S
- $a \in \mathbb{R}$: a and b are known constants
- $b \in \mathbb{R}$: a and b are known constants
- $\alpha \in \mathbb{R} \to \mathbb{R}$: α the logarithm function
- $\beta \in \mathbb{R} \to \mathbb{R}$: β is the multiplicative inverse

(*)H❤️rtDown displays a context-dependent glossary in a fixed position as the page scrolls. The glossary updates automatically with the relevant symbol list.

H❤️rtDown Design: Reading Environment

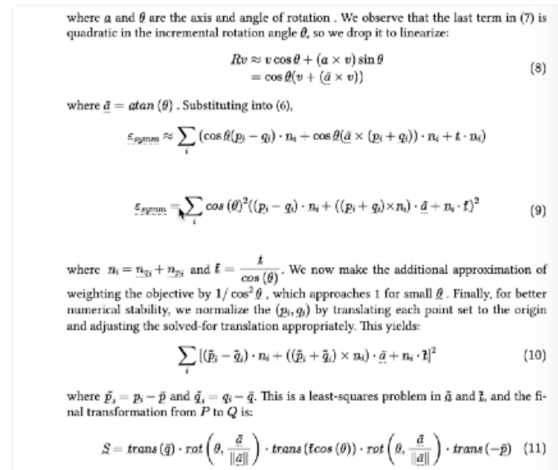
- Symbol definitions



When clicking on a symbol, H❤️rtDown shows its description and an arrow to the prose that described it.

H❤️rtDown Design: Reading Environment

- Equation relationships


where a and θ are the axis and angle of rotation . We observe that the last term in (7) is quadratic in the incremental rotation angle θ, so we drop it to linearize:

$$\begin{aligned} Rv &\approx v\cos\theta + (a \times v)\sin\theta \\ &= \cos\theta(v + (\tilde{a} \times v)) \end{aligned} \quad (8)$$

where $\tilde{a} = a tan(\theta)$. Substituting into (6),

$$\varepsilon_{symm} \approx \sum_i (\cos\theta(p_i - q_i) \cdot n_i + \cos\theta(\tilde{a} \times (p_i + q_i)) \cdot n_i + t \cdot n_i)$$

$$\varepsilon_{symm} = \sum_i cos(\theta)^2((p_i - q_i) \cdot n_i + ((p_i + q_i) \times n_i) \cdot \tilde{a} + n_i \cdot \tilde{t})^2 \quad (9)$$

where $n_i = n_{q_i} + n_{p_i}$ and $\tilde{t} = \frac{t}{cos(\theta)}$. We now make the additional approximation of weighting the objective by $1/\cos^2\theta$, which approaches 1 for small θ . Finally, for better numerical stability, we normalize the (p_i, q_i) by translating each point set to the origin and adjusting the solved-for translation appropriately. This yields:

$$\sum_i [(\tilde{p}_i - \tilde{q}_i) \cdot n_i + ((\tilde{p}_i + \tilde{q}_i) \times n_i) \cdot \tilde{a} + n_i \cdot \tilde{t}]^2 \quad (10)$$

where $\tilde{p}_i = p_i - \bar{p}$ and $\tilde{q}_i = q_i - \bar{q}$. This is a least-squares problem in $\tilde{a}$ and $\tilde{t}$, and the final transformation from P to Q is:

$$S = trans(\bar{q}) \cdot rot\left(\theta, \frac{\tilde{a}}{\|\tilde{a}\|}\right) \cdot trans(\tilde{t}cos(\theta)) \cdot rot\left(\theta, \frac{\tilde{a}}{\|\tilde{a}\|}\right) \cdot trans(-\bar{p}) \quad (11)$$

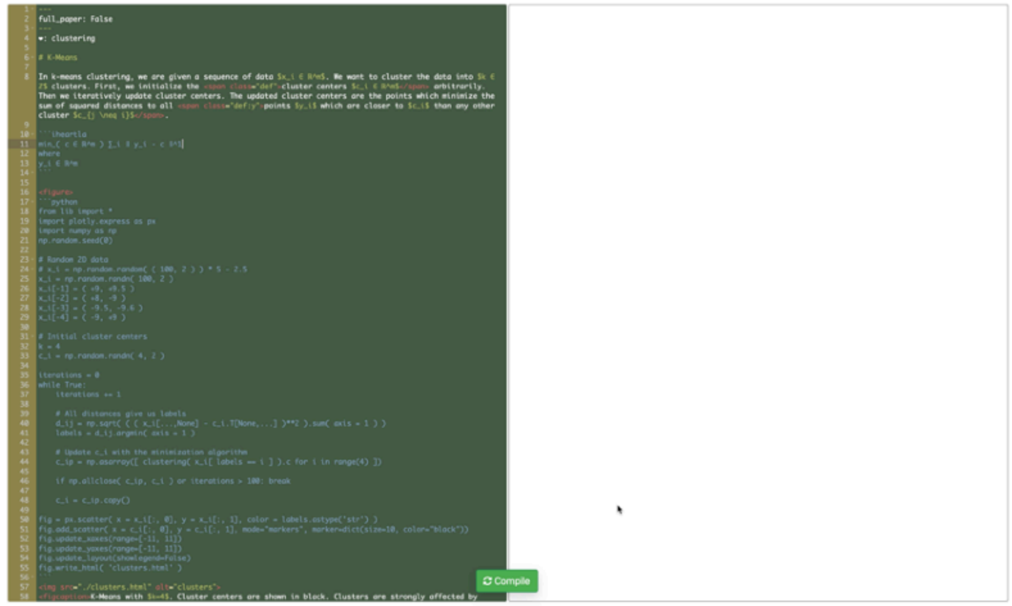
(*)When clicking an equation, H❤️rtDown highlights the terms involved in the formula as well as downstream uses of the symbol.

H❤️rtDown solves a graph coloring problem to color symbols distinctly.

% H❤️rtDown solves a graph coloring problem using a greedy technique [Liu et al. 2021] to ensure symbols in the same equation have different colors.

H❤️rtDown Design: **Experimenter** (making use of)

H❤️rtDown Editor



(*)Here's an example use of H❤️rtDown as an experimenter. This example describes a clustering algorithm.

We generate the output by compiling the source code.

The paper uses the L1 norm to calculate distances. The user changes the math to use the more common L2 norm.

H❤️rtDown updates both the typeset math and the figure that relies on the generated code library.

The new cluster centers have changed and are now heavily influenced by the outliers.

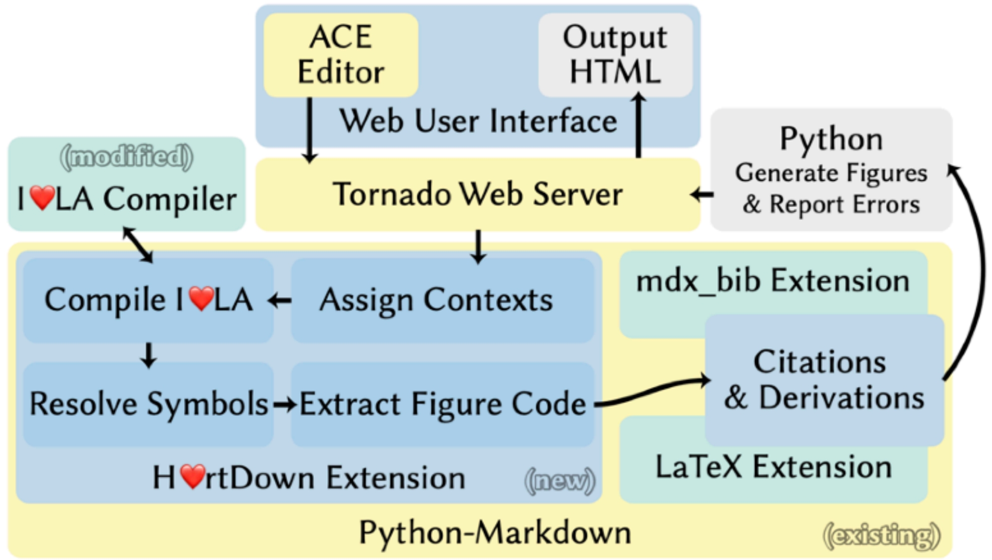
We can also click the figure to update the figure code, in this case, H❤️rtDown will only rerun the figure code.

###

The generated code libraries are saved into files and can be used outside of the H❤️rtDown reading/authoring environment.

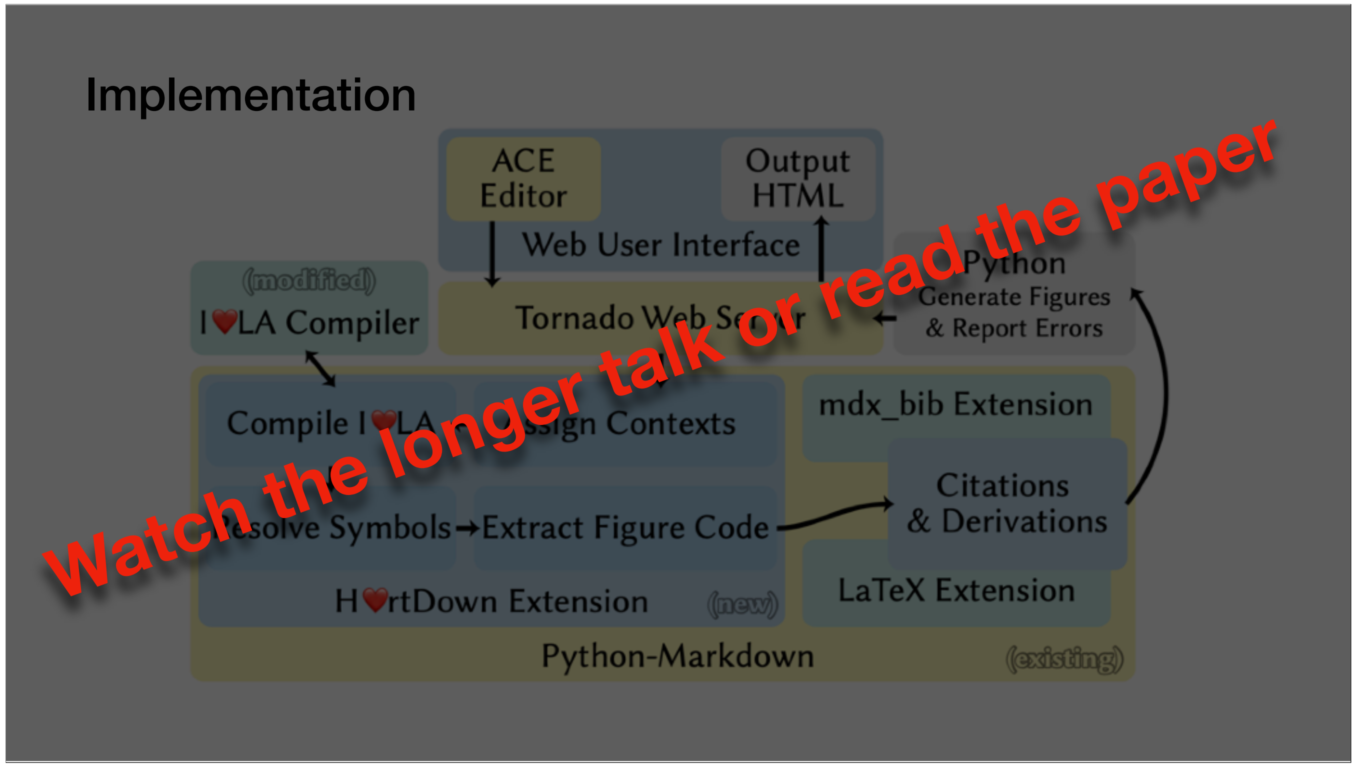
We'll show examples of that in our case studies.

Implementation

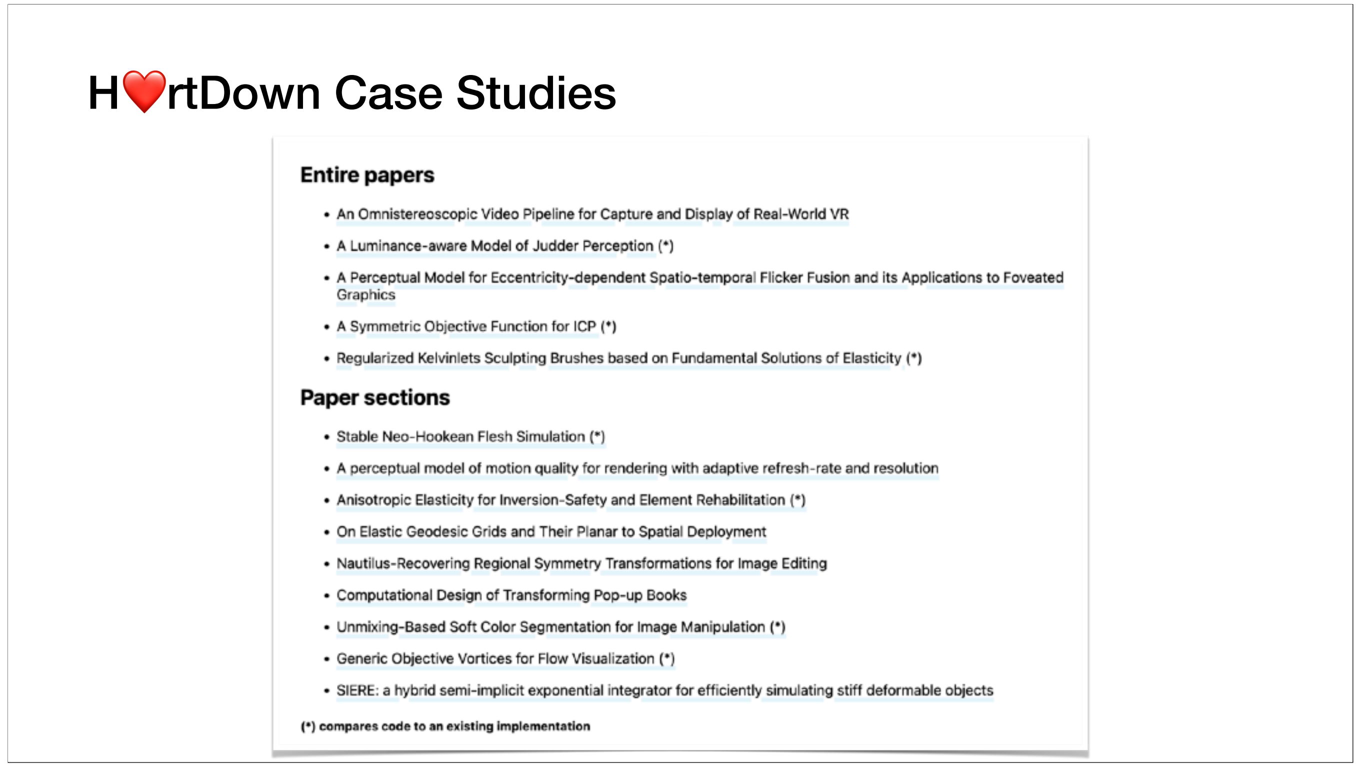


To learn about H❤️rtDown's implementation,

% This figure shows the overall structure for H❤️rtDown's implementation.



please watch Yong's longer talk or read the paper.

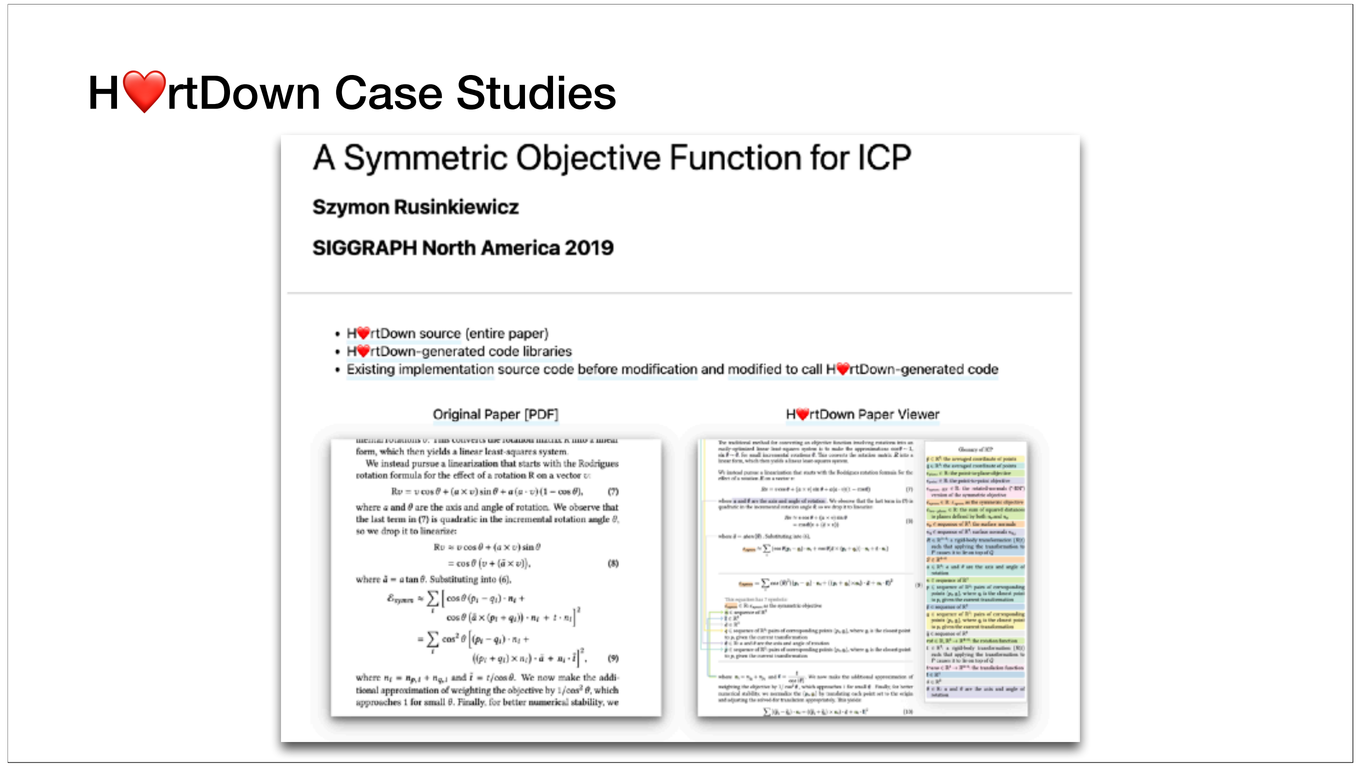


We converted a variety of SIGGRAPH papers and paper sections to H❤️rtDown as case studies.

########
Our criteria for selecting papers were that they use linear algebra implementable by I❤️LA

The papers are from the past five years (2017–2021) of SIGGRAPH and span geometry processing, image processing, visualization, simulation, and rendering.

It include 5 full papers and 9 papers for which we implemented single subsections

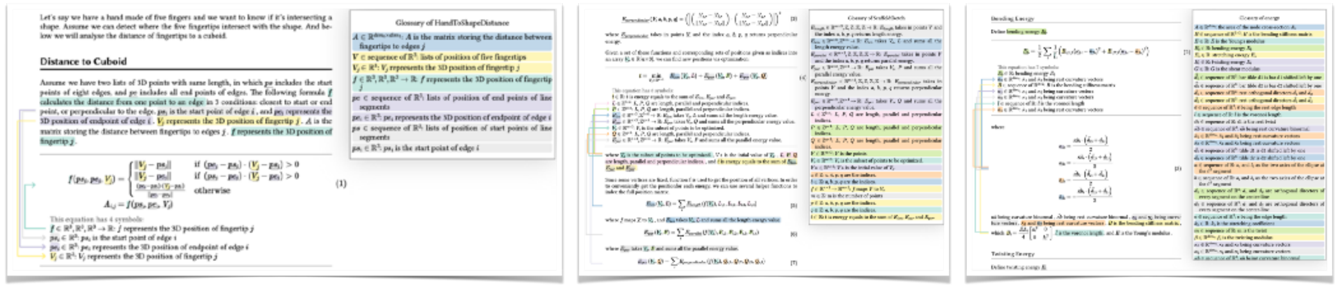


Each case study includes the H❤️rtDown source file, H❤️rtDown's generated paper reading environment, and H❤️rtDown's generated code library for C++, Python and MATLAB.

We also provide a link to the original paper for comparison and side-by-side screenshots.

Expert Study

- 3 CS PhD students
- Author an original document related to their computer graphics research



- Spent 24, 7, and 6 hours, respectively, using H❤️rtDown over a period of two weeks

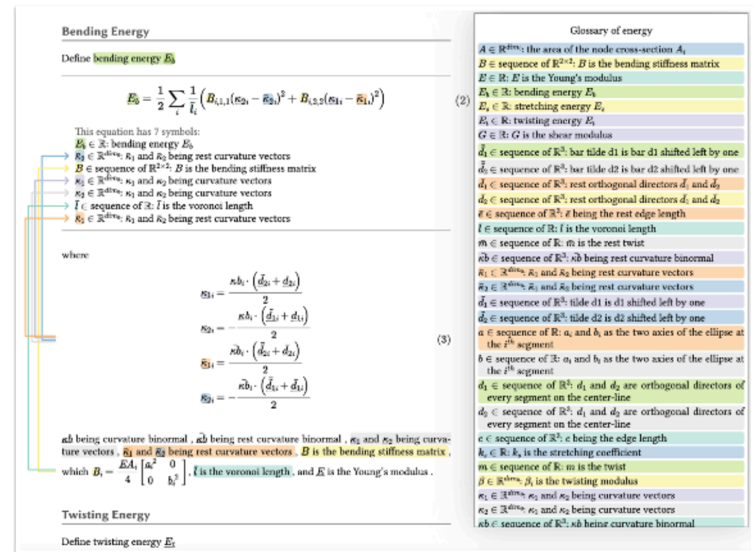
We recruited 3 computer science PhD students for an expert study.

In our experiment, participants were given initial and follow-up questionnaires to understand their current practices and share their thoughts about H❤️rtDown.

They spent a total of 24, 7, and 6 hours, respectively, using H❤️rtDown over a period of two weeks.
For the tasks in our expert study,

Expert Study: Observations and Conclusions

“H❤️rtDown is an excellent tool to share tutorial[s] online—it highlights the vector dimension and variable meaning…following all the vectors/matrices/their dims is ***the hardest part*** *of reproducing a paper.”*



Bending Energy

Define bending energy E_b

$$E_b = \frac{1}{2}\sum_i \frac{1}{\bar{l}_i}\left(B_{i,1,1}(\kappa_{2_i} - \bar{\kappa}_{2_i})^2 + B_{i,2,2}(\kappa_{1_i} - \bar{\kappa}_{1_i})^2\right) \quad (2)$$

This equation has 7 symbols:
- $E_b \in \mathbb{R}$: bending energy E_b
- $\bar{\kappa}_2 \in \mathbb{R}^{dims}$: $\bar{\kappa}_1$ and $\bar{\kappa}_2$ being rest curvature vectors
- $B \in$ sequence of $\mathbb{R}^{2\times2}$: B is the bending stiffness matrix
- $\kappa_1 \in \mathbb{R}^{dims}$: κ_1 and κ_2 being curvature vectors
- $\kappa_2 \in \mathbb{R}^{dims}$: κ_1 and κ_2 being curvature vectors
- $\bar{l} \in$ sequence of $\mathbb{R}$: $\bar{l}$ is the voronoi length
- $\bar{\kappa}_1 \in \mathbb{R}^{dims}$: $\bar{\kappa}_1$ and $\bar{\kappa}_2$ being rest curvature vectors

where

$$\kappa_{1_i} = \frac{\kappa b_i \cdot (\tilde{d}_{2_i} + d_{2_i})}{2}$$
$$\kappa_{2_i} = -\frac{\kappa b_i \cdot (\tilde{d}_{1_i} + d_{1_i})}{2}$$
$$\bar{\kappa}_{1_i} = \frac{\bar{\kappa b}_i \cdot (\bar{\tilde{d}}_{2_i} + \bar{d}_{2_i})}{2}$$
$$\bar{\kappa}_{2_i} = -\frac{\bar{\kappa b}_i \cdot (\bar{\tilde{d}}_{1_i} + \bar{d}_{1_i})}{2} \quad (3)$$

κb being curvature binormal, $\bar{\kappa b}$ being rest curvature binormal, κ_1 and κ_2 being curvature vectors, $\bar{\kappa}_1$ and $\bar{\kappa}_2$ being rest curvature vectors, B is the bending stiffness matrix, which $B_i = \frac{EA_i}{4}\begin{bmatrix} a_i^2 & 0 \\ 0 & b_i^2 \end{bmatrix}$, $\bar{l}$ is the voronoi length, and E is the Young's modulus.

Twisting Energy

Define twisting energy E_t

Glossary of energy
- $A \in \mathbb{R}^{dims}$: the area of the node cross-section A_i
- $B \in$ sequence of $\mathbb{R}^{2\times2}$: B is the bending stiffness matrix
- $E \in \mathbb{R}$: E is the Young's modulus
- $E_b \in \mathbb{R}$: bending energy E_b
- $E_s \in \mathbb{R}$: stretching energy E_s
- $E_t \in \mathbb{R}$: twisting energy E_t
- $G \in \mathbb{R}$: G is the shear modulus
- $\bar{\tilde{d}}_1 \in$ sequence of $\mathbb{R}^3$: bar tilde d1 is bar d1 shifted left by one
- $\bar{\tilde{d}}_2 \in$ sequence of $\mathbb{R}^3$: bar tilde d2 is bar d2 shifted left by one
- $\bar{d}_1 \in$ sequence of $\mathbb{R}^3$: rest orthogonal directors $\bar{d}_1$ and $\bar{d}_2$
- $\bar{d}_2 \in$ sequence of $\mathbb{R}^3$: rest orthogonal directors $\bar{d}_1$ and $\bar{d}_2$
- $\bar{e} \in$ sequence of $\mathbb{R}^3$: $\bar{e}$ being the rest edge length
- $\bar{l} \in$ sequence of $\mathbb{R}$: $\bar{l}$ is the voronoi length
- $\bar{m} \in$ sequence of $\mathbb{R}$: $\bar{m}$ is the rest twist
- $\bar{\kappa b} \in$ sequence of $\mathbb{R}^3$: $\bar{\kappa b}$ being rest curvature binormal
- $\bar{\kappa}_1 \in \mathbb{R}^{dims}$: $\bar{\kappa}_1$ and $\bar{\kappa}_2$ being rest curvature vectors
- $\bar{\kappa}_2 \in \mathbb{R}^{dims}$: $\bar{\kappa}_1$ and $\bar{\kappa}_2$ being rest curvature vectors
- $\tilde{d}_1 \in$ sequence of $\mathbb{R}^3$: tilde d1 is d1 shifted left by one
- $\tilde{d}_2 \in$ sequence of $\mathbb{R}^3$: tilde d2 is d2 shifted left by one
- $a \in$ sequence of $\mathbb{R}$: a_i and b_i as the two axies of the ellipse at the i^{th} segment
- $b \in$ sequence of $\mathbb{R}$: a_i and b_i as the two axies of the ellipse at the i^{th} segment
- $d_1 \in$ sequence of $\mathbb{R}^3$: d_1 and d_2 are orthogonal directors of every segment on the center-line
- $d_2 \in$ sequence of $\mathbb{R}^3$: d_1 and d_2 are orthogonal directors of every segment on the center-line
- $e \in$ sequence of $\mathbb{R}^3$: e being the edge length
- $k_s \in \mathbb{R}$: k_s is the stretching coefficient
- $m \in$ sequence of $\mathbb{R}$: m is the twist
- $\beta \in \mathbb{R}^{dims}$: β_i is the twisting modulus
- $\kappa_1 \in \mathbb{R}^{dims}$: κ_1 and κ_2 being curvature vectors
- $\kappa_2 \in \mathbb{R}^{dims}$: κ_1 and κ_2 being curvature vectors
- $\kappa b \in$ sequence of $\mathbb{R}^3$: κb being curvature binormal

(*) One expert wrote a tutorial for discrete elastic rods. This tutorial is now more readable due to math augmentations and can be used in addition to reading since it self-generates code in any programming languages I❤️LA supports.

That expert said: “H❤️rtDown is an excellent tool to share tutorial[s] online—it highlights the vector dimension and variable meaning…following all the vectors/matrices/their dims is the hardest part of reproducing a paper.”

Please see the longer talk or the paper for an in-depth discussion.

Limitations

- H❤️rtDown does not consider pseudocode or algorithmic steps described in prose

Algorithm 1 A single simulation step of our proposed SPH-based snow solver.

foreach particle i **do**
  compute $\rho_{0,i}^{t}$ ▷ see Subsection 3.3.2
  compute L_i ▷ see Eq. (15)
  compute $\mathbf{a}_i^{\text{other},t}$ ▷ e.g., gravity and adhesion
  compute $\mathbf{a}_i^{\text{friction},t}$ ▷ using Eq. (24)
SOLVE for $\mathbf{a}_i^{\lambda}$ ▷ see Subsection 3.2.1
SOLVE for $\mathbf{a}_i^{G}$ ▷ see Subsection 3.2.2
foreach particle i **do**
  integrate $\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t(\mathbf{a}_i^{\text{other},t} + \mathbf{a}_i^{\text{friction},t} + \mathbf{a}_i^{\lambda} + \mathbf{a}_i^{G})$
foreach particle i **do**
  integrate $\mathbf{F}_{E,i}$ ▷ see Subsection 3.3.1
foreach particle i **do**
  integrate $\mathbf{x}_i^{t+\Delta t} = \mathbf{x}_i^t + \Delta t \mathbf{v}_i^{t+\Delta t}$

[Gissler et al. 2020]

- The space of executable math and potential application domains for H❤️rtDown is much broader than linear algebra

One limitation of H❤️rtDown is that it does not consider pseudocode, literate programming, or algorithmic steps described in prose. Algorithms are often needed to make formulas useful.

Another limitation stems from the kinds of formulas that our extended version of I❤️LA can handle.

Future Work

- Automatic conversion from LaTeX to H❤️rtDown
- A proof checker to verify derivations
- Callbacks and delegates for expanding the abilities of the generated code
- Support for active reading (e.g. annotating and comparing)

There are a lot of directions we'd like to explore in the future.

Automatic or semi-automatic conversion from LaTeX to H❤️rtDown

Incorporating a proof checker to verify derivations

Explore callbacks and delegates for expanding the abilities of the generated code

Improve our reading environment to support active reading activities such as annotating and comparing

Conclusions

- H❤️rtDown is a low-overhead, ecologically compatible document processor
- H❤️rtDown supports authors and improves replicability, readability, and experimentation
- Participants in our expert study found uses for H❤️rtDown in their research practice.

In conclusion,

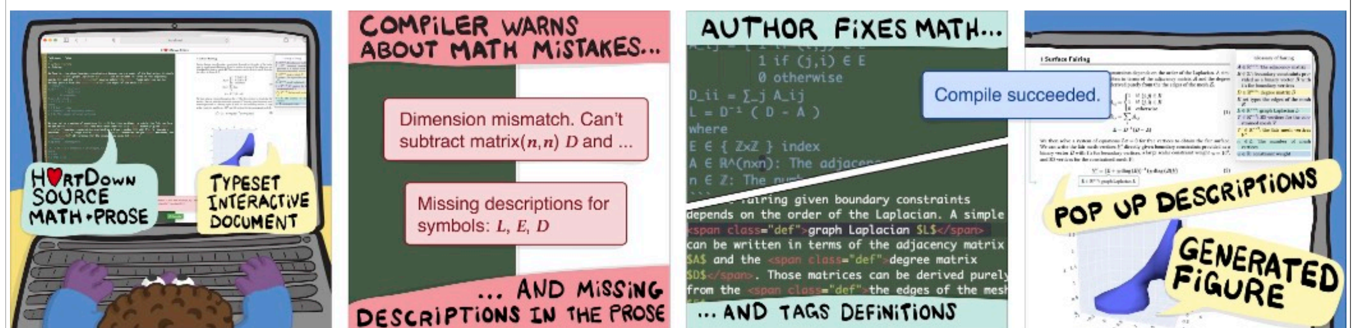
H❤️rtDown is a low-overhead, ecologically compatible document processor

H❤️rtDown supports authors and improves replicability, readability, and experimentation

Participants In our expert study found uses for H❤️rtDown in their research practice.

H❤️rtDown

https://iheartla.github.io/heartdown/



Acknowledgements: Anonymous reviewers, Seth Walker, Zoya Bylinskii, Zhecheng Wang, Xue Yu, Jialin Huang
Sponsors: Canada Research Chairs Program, Sloan Foundation, Adobe Inc.

H❤️rtDown can be used at all stages of research
(from experimenting with the seed of an idea, to writing the final paper)

Thanks for listening!

Please try our language.

You are welcome to contact us in the future.