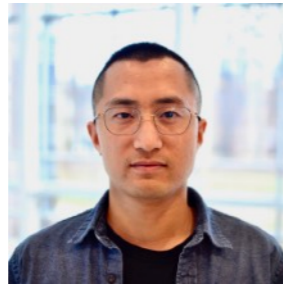


# H♥rtDown: Document Processor for Executable Linear Algebra Papers

Yong Li



Shoaib Kamil



Alec Jacobson



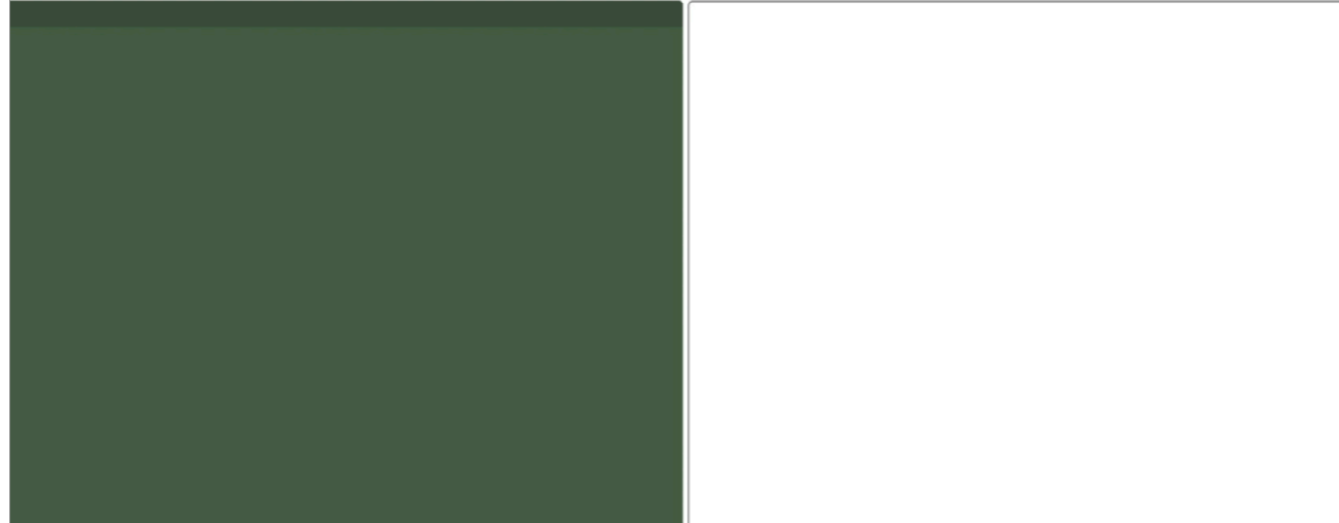
Yotam Gingold



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



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

```

4 # Surface Fairing
5 ♥: fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A simple graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the degree matrix  $D$ . Those matrices can be derived purely from the the edges of the mesh  $E$ .
8 ```iheartla
9  $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$ 
10
11  $D_{ii} = \sum_j A_{ij}$ 
12  $L = D^{-1} ( D - A )$ 
13 where
14  $E \in \{ \mathbb{Z} \times \mathbb{Z} \}$  index
15  $A \in \mathbb{R}^{(n \times n)}$ : The adjacency matrix
16  $n \in \mathbb{Z}$ : The number of mesh vertices
17 ```
18
19
20 We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write the fair mesh vertices  $V'$  directly given boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar constraint weight  $w$   $w=10^6$ , and 3D vertices for the constrained mesh  $V$ :
21 ```iheartla
22 diag from linearalgebra
23
24  $V' = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V)$ 
25 where

```

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

H♥rtDown Editor

```
1 ---
2 full_paper: False
3 ---
4 # Surface Fairing
5 w: fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A simple
8 class="def">graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the
9 class="def">degree matrix  $D$ . Those matrices can be derived purely from the
10 class="def">edges of the mesh  $E$ .
11
12  $A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases}$ 
13  $D_{ii} = \sum_j A_{ij}$ 
14  $L = D^{-1} (D - A)$ 
15 where
16  $E \in \mathbb{Z}^2$ : index
17  $A \in \mathbb{R}^{(n \times n)}$ : The adjacency matrix
18  $n \in \mathbb{Z}$ : The number of mesh vertices
19 ---
20 We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write
21 class="def">the fair mesh vertices  $V$  directly given class="def">boundary constraints
22 provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar class="def">
23 class="def">constraint weight  $w$ , and  $n$  vertices for the constrained mesh
24  $V \in \mathbb{R}^{(n \times 3)}$ .
25
26 class="def">diag from linearalgebra
27  $V = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V)$ 
28 where
29  $B \in \mathbb{R}^n$ 
30  $V \in \mathbb{R}^{(n \times 3)}$ 
31 ---
32 class="def">figure
33 class="def">python
34 from lib import *
35 import make_cylinder
36
37 # Load cylinder with n vertices
38 mesh = make_cylinder.make_cylinder( 10, 10 )
39 make_cylinder.save_obj( mesh, 'input.obj', clobber = True )
40 V = mesh.v
41 F = mesh.fv
42 n = len(V)
43
44 # Extract the mesh edges
45 edges = set()
46 for face in F:
47     for fvi in range(3):
48         v1, v2 = face[fvi], face[(fvi+1)%3]
49         edges.add( ( min(v1, v2), max(v1, v2) ) )
50
51 # The constraint vector is all vertices with  $z < 1/4$  or  $z > 3/4$ 
52 B = np.zeros( n, dtype = int )
53 B[ V[:,2] < 1/4 ] = 1
54 B[ V[:,2] > 3/4 ] = 1
55
56 # Rotate the top around the z axis by 90 degrees.
57 R = np.array([ [ 1, 0, 0 ],
```

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

```
simple <span  
x  $\$A\$$  and the <span  
span class="def">the  
  
face. We can write  
>boundary constraints  
<span  
or the constrained mesh
```

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

H♥rtDown Editor

```

1  #
2  full_paper: False
3  #
4  # Surface Fairing
5  #
6  #
7  Surface fairing given boundary constraints depends on the order of the Laplacian. A simple
8  #
9  # class="def" graph Laplacian L can be written in terms of the adjacency matrix A and the degree
10 # matrix D. Those matrices can be derived purely from the edges of the mesh mesh.edges.
11 #
12 #
13 #
14 #
15 #
16 #
17 #
18 #
19 #
20 #
21 #
22 #
23 #
24 #
25 #
26 #
27 #
28 #
29 #
30 #
31 #
32 #
33 #
34 #
35 #
36 #
37 #
38 #
39 #
40 #
41 #
42 #
43 #
44 #
45 #
46 #
47 #
48 #
49 #
50 #
51 #
52 #
53 #
54 #
55 #
56 #

```

### 1 Surface Fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A simple graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the degree matrix  $D$ . Those matrices can be derived purely from the edges of the mesh  $E$ .

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

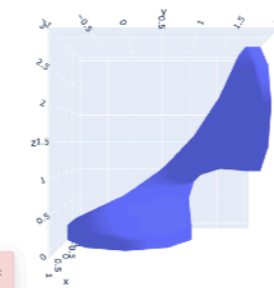
$$D_{ii} = \sum_j A_{ij}$$

$$L = D - A$$

We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write the fair mesh vertices  $V^f$  directly given boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar constraint weight  $w$ , and 3D vertices for the constrained mesh  $V^c$ .

$$V^f = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V^c)$$

Missing descriptions for symbols:  
fairing:  $D$



Fairing the middle half of a cylinder.

#### Glossary of fairing

- $A \in \mathbb{R}^{n \times n}$ : The adjacency matrix
- $B \in \mathbb{Z}^n$ : boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices
- $D \in \mathbb{R}^{n \times n}$
- $E$  set type: the edges of the mesh  $E$
- $L \in \mathbb{R}^{n \times n}$ : graph Laplacian
- $V^c \in \mathbb{R}^{n \times 3}$ : 3D vertices for the constrained mesh  $V^c$
- $V^f \in \mathbb{R}^{n \times 3}$ : the fair mesh vertices  $V^f$
- $n \in \mathbb{Z}$ : The number of mesh vertices
- $w \in \mathbb{R}$ : constraint weight

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

HotDown Editor
Glossary of fairing

```

1
2 full_paper: False
3
4 # Surface Fairing
5 # Fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A simple
8
9 class def graph Laplacian L:
10     def __init__(self, A):
11         self.A = A
12         self.D = np.zeros((A.shape[0], A.shape[0]))
13         for i, j in enumerate(A.nonzero()[0]):
14             self.D[i, i] += A[i, j]
15         self.L = self.D - self.A
16
17     def solve(self, B):
18         return np.linalg.pinv(self.L).dot(B)
19
20 We then solve a system of equations Lx = B for free vertices to obtain the fair surface. We can write
21
22 class def fair mesh vertices V:
23     def __init__(self, L, B, w):
24         self.L = L
25         self.B = B
26         self.w = w
27         self.V = (self.L + w * np.diag(B))^-1 * (self.B + w * np.diag(B) * V)
28
29
30
31
32
33
34
35 # Load cylinder with n vertices
36 mesh = make_cylinder.make_cylinder(10, 10)
37 mesh.vertices[:, 2] = mesh.vertices[:, 2] * 0.5
38 V = mesh.v
39 F = mesh.fv
40 n = len(V)
41
42 # Extract the mesh edges
43 edges = set()
44 for face in F:
45     for i, j in range(3):
46         v1, v2 = face[face[i], face[(i+1)%3]]
47         edges.add((min(v1, v2), max(v1, v2)))
48
49 # The constraint vector is all vertices with z < 1/4 or z > 3/4
50 B = np.zeros(n, dtype=int)
51 B[V[:, 2] < 1/4] = 1
52 B[V[:, 2] > 3/4] = 1
53
54 # Rotate the top around the z axis by 90 degrees.
55 B = np.array([1, 0, 0])

```

### 1 Surface Fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A simple graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the degree matrix  $D$ . These matrices can be derived purely from the edges of the mesh  $E$ .

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$D_{ii} = \sum_j A_{ij}$$

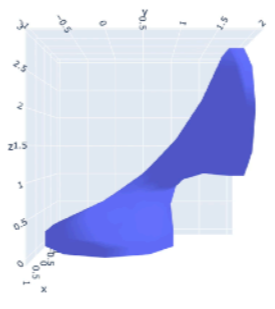
$$L = D - A$$

We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write the fair mesh vertices  $V'$  directly given boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar constraint weight  $w = 10^5$ , and 3D vertices for the constrained mesh  $V$ :

$$V' = (L + w \text{diag}(B))^{-1} (w \text{diag}(B)V) \quad (2)$$

**Glossary of fairing**

- $A \in \mathbb{R}^{n \times n}$ : The adjacency matrix
- $B \in \mathbb{Z}^n$ : boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices
- $D \in \mathbb{R}^{n \times n}$ : degree matrix
- $E$  set type: the edges of the mesh
- $L \in \mathbb{R}^{n \times n}$ : graph Laplacian
- $V \in \mathbb{R}^{n \times 3}$ : 3D vertices for the constrained mesh
- $V' \in \mathbb{R}^{n \times 3}$ : the fair mesh vertices
- $n \in \mathbb{Z}$ : The number of mesh vertices
- $w \in \mathbb{R}$ : constraint weight



Fairing the middle half of a cylinder.

The compiled math can be used to generate figures.

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

## Outline

- Related work
- Function analysis
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

Researching and disseminating scientific ideas relies on written communication



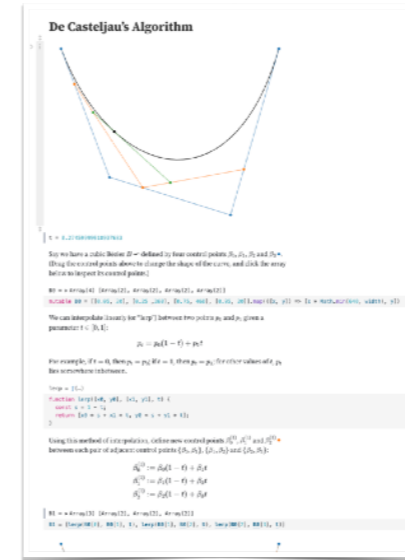
## Outline

- Related work
- Function analysis
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

Researching and disseminating scientific ideas relies on written communication

## Related Work: Literate programming environments

- Literate Programming [Knuth 1984]
- Markdown [Gruber and Swartz 2004]
- Notebooks [Amon 1988; Kery et al. 2018; Rule et al. 2018; Wolfram 1988]
- Pluto [Plas 2020]
- Observable [Bostock 2017]



Observable  
[Bostock 2017]

There are many Literate programming environments including Notebook, Pluto and Observable.

Some of these support prose-determined order. These duplicate math and code

# The prose determines the order in which the pieces of the computer program are presented, rather than the order of execution required by the programming language.

## Related Work: Reactive documents and publishing

- Idyll [Conlen and Heer 2018]
- Tangle [Victor 2011]
- ScholarPhi [Head et al. 2021]
- Distill [Team 2021]
- Authorea [Goodman et al. 2017]
- Nota [Crichton 2021]
- [Bonneel et al. 2020]



ScholarPhi  
[Head et al. 2021]

Many approaches to reactive documents such as Idyll, Tangle and ScholarPhi have been proposed.

Some papers focus on authoring and publishing scientific articles such as Distill, Authorea and Nota.

In contrast, **HeartDown** is focused on helping users correctly author, read, and experiment with mathematical formulas in scientific documents.

## Related Work: Compilable math and augmentations

- Fortress [Allen et al. 2005]
- Lean [de Moura et al. 2015]
- Julia [Bezanson et al. 2017]
- I♥LA [Li et al. 2021]
- [Alcock and Wilkinson 2011]
- [Dragunov and Herlocker 2003]
- [Head et al. 2021, 2022]
- Penrose [Ye et al. 2020]

```
given
p_i ∈ ℝ³: points on lines
d_i ∈ ℝ³: unit directions along lines

P_i = ( I₃ - d_i d_iᵀ )
q = ( ∑_i P_i )⁻¹ ( ∑_i P_i p_i )
```

I♥LA example

Languages for Compilable Math include Fortress, Lean, Julia and I♥LA. We can build H♥rtDown on top of different languages here, we choose I♥LA since it resembles the equations in paper and can generate code for different backends.

Various math augmentations such as ScholarPhi and Penrose have been proposed to facilitate understanding mathematical notation in papers.

We use some of these augmentations in our viewer.

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

We did a formative study for H♥rtDown, before that

## Design Goals

- Support **authoring, reading**, and making use of (**experimenting** with)
  - Correct and reproducible documents
  - Minimal authoring overhead
- **Ecological compatibility**
  - Don't change/restrict 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 thoroughly analyzed 156 papers from the SIGGRAPH North America 2020 Technical Papers, 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.

We found that

They all 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.



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



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]

284 • Wronski et al.

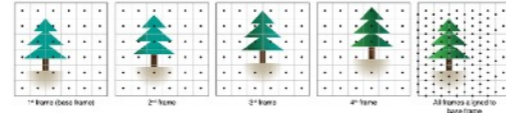


Fig. 4. Subpixel displacements from handfield motion. Illustration of a kernel of four frames with linear hand motion. Each frame is offset from the previous frame by half a pixel along the x-axis and a quarter pixel along the y-axis due to the hand motion. After alignment to the base frame, the pixel centers (black dots) uniformly cover the sampling grid (gray lines) at an increased density. In practice, the distribution is more uniform than in this simplified example.

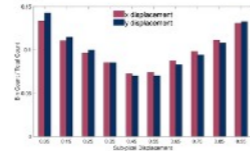


Fig. 5. Distribution of estimated subpixel displacements. Histogram of subpixel displacements as computed by the alignment algorithm (Section 3.2). While the alignment process is based on auto-correlation values, we observe a different coverage of sampled values to maintain representation. Note that displacements in x and y are not correlated.



Fig. 6. Sparse data reconstruction with anisotropic kernels. A sparse sample of very blurry (e.g., matrix  $R_{\text{down}} = 0.15$ ) kernels on a real captured local. For demonstration purposes, we represent samples corresponding to white BGR input patches instead of separate color channels. Kernel adaptation allows us to apply anisotropic kernels to different channels through the final alignment step. The original kernel aligns with the local image gradient. The anisotropic kernel aligns with the local image gradient to enhance the structure at details.

## 5.1 Kernel Reconstruction

The core of our algorithm is built on the idea of creating pixels of multiple raw Bayer frames as irregularly offset, aligned and registered measurements of three different underlying continuous pixels, one for each color channel of the Bayer mosaic. Though the color channels are often correlated, in the case of saturation colors like orange, red, green or blue which they are not. Using different spatial coverage, separate per-channel reconstruction allows us to recover the original high resolution signal even in these cases.

To produce the final output image we process all frames separately - for every output image pixel, we evaluate local reconstruction in the red, green and blue color channels from different input frames. Every input raw image pixel has a different color channel and contributes only to a specific output color channel. Local contributions are weighted, therefore, we accumulate weights, probabilities and weights. At the end of the pipeline, these probabilities are normalized. For each color channel, this can be expressed as:

$$C(x, y) = \frac{\sum_{i=1}^N w_i \cdot R_i(x, y)}{\sum_{i=1}^N w_i} \quad (1)$$

ACM Trans. Graph., Vol. 38, No. 4, Article 28. Publication date: July 2019.

where  $(x, y)$  are the pixel coordinates, the sum  $\sum_{i=1}^N$  is over all one subpixel frames,  $\sum_{i=1}^N$  is a sum over samples within a local neighborhood (in our case  $3 \times 3$ ).  $C_{i,j}$  denotes the value of the Bayer pixel at Bayer frame  $i$  and sample  $j$ ,  $w_{i,j}$  is the local sample weight and  $R_i$  is the local reconstruction (Section 3.2). In the case of the base frame,  $R$  is equal to 1 as it does not get aligned, and we have full confidence in its local sample values.

To compute the local pixel weights, we use local radial basis function kernels, similar to the non-parametric local regression framework of Takida et al. [2016, 2017]. Unlike Takida et al., we don't alternate kernel basis function parameters at sparse sample positions. Instead, we evaluate them at the final resampling grid positions. Furthermore, we always look at the nine closest samples in a  $3 \times 3$  neighborhood and use the same kernel function for all those samples. This allows for efficient parallel evaluation on a GPU. Using this "softmax" approach every output pixel is independently processed only once per frame. This is similar to work



Fig. 7. Anisotropic Kernels. Left: When isotropic kernels ( $\beta_{\text{horizontal}} = 1$ ,  $\beta_{\text{vertical}} = 1$ ) are applied to an image, small misalignments cause blurry edges. Right: Anisotropic kernels ( $\beta_{\text{horizontal}} = 0$ ,  $\beta_{\text{vertical}} = 1$ ) in the corners.

Yu and Turk [2013], developed for fluid rendering. Two steps described in the following sections are: estimation of the kernel shape (Section 3.3) and subsequent kernel sample contribution weighting (Section 5.2).

5.1.1 Local Anisotropic Merge Kernels. Given our problem formulation, kernel weights and kernel functions define the image quality of the final merged image. Kernels with wide spatial support produce noise-free and artifact-free, but blurry images, while kernels with very narrow support can produce sharp and detailed images. A natural choice for kernels used for signal reconstruction are radial Basis Function kernels - in our case anisotropic Gaussian kernels. We can adjust the kernel shape to different local properties of the input frames: amounts of detail and the presence of edges (Figure 6). This is similar to kernel selection techniques used in other sparse data reconstruction applications [Takida et al. 2016, 2017; Yu and Turk 2013].

Specifically, we use a 2D normalized anisotropic Gaussian RBF kernel:

$$k_{ij} = \exp\left(-\frac{1}{2} \mathbf{x}_i^T \mathbf{\Omega}^{-1} \mathbf{x}_j\right) \quad (2)$$

where  $\mathbf{x}_i$  is the kernel covariance matrix and  $\mathbf{\Omega}$  is the offset vector of samples in the output image ( $\mathbf{\Omega} = \mathbf{x}_i - \mathbf{x}_j$ ,  $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^2$ ).

One of the main motivations for using anisotropic kernels is that they increase the algorithm's tolerance for small misalignments and uneven coverage around edges. Edges are ambiguous in the alignment process due to the aperture problem and result in alignment errors [Robinson and Siftaris 2004] even frequently compared to non-edge regions of the image. Subpixel misalignment as well as a lack of sufficient sample coverage can manifest as zipper artifacts (Figure 9). By stretching the kernel along the edges, we can enforce the assignment of smaller weights to pixels not belonging to edges in the image.

5.1.2 Kernel Covariance Computation. We compute the kernel covariance matrix by analyzing every frame's local gradient structure. To improve runtime performance and resilience to image noise, we analyze gradients of half-resolution images formed by downsampling the original raw frames by a factor of two. To estimate a Bayer image containing different color channels, we create a single

Handfield Multi-Frame Super-Resolution • 287

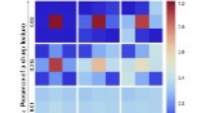


Fig. 8. Merge kernels. Heatmap of kernel weights for different  $3 \times 3$  sampling kernels as a function of local image features.

pixel from a  $2 \times 2$  Bayer quad by combining four different color channels together. This way, we can operate on single channel Bayer images and perform the computation of a quarter of the full resolution cost and with improved signal to noise ratio. To estimate local information about strength and direction of gradients, we use gradient structure tensor analysis [Bigün et al. 1994; Harris and Stephens 1988]:

$$\mathbf{\Omega} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad (3)$$

where  $I_x$  and  $I_y$  are the local image gradients in horizontal and vertical directions, respectively. The image gradients are computed by finite forward differencing the luminance in a small  $3 \times 3$  color window (going to four different horizontal and vertical gradient values). Eigenanalysis of the local structure tensor  $\mathbf{\Omega}$  gives two orthogonal direction vectors  $\mathbf{e}_1, \mathbf{e}_2$  and two associated eigenvalues:

$$\mathbf{\Omega} = \mathbf{e}_1 \lambda_1 \mathbf{e}_1^T + \mathbf{e}_2 \lambda_2 \mathbf{e}_2^T \quad (4)$$

where  $\lambda_1$  and  $\lambda_2$  contain the desired tensor variance in either edge or orthogonal direction. We convert these values to robustly estimate super-resolution and denoising. We use the magnitude of the structure tensor's dominant eigenvalue  $\lambda_1$  to derive the spatial support of the kernel and the trade-off between the super-resolution and denoising, where  $\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}$  is used to derive the desired anisotropy of the kernels (Figure 9). The specific process we use to compute the final kernel covariance can be found in the supplemental material along with the tuning values. Since  $\mathbf{\Omega}$  is computed at half of the Bayer image resolution, we upsample the kernel covariance values through bilinear sampling before computing the kernel weights.

## 5.2 Motion Robustness

Reliable alignment of an arbitrary sequence of images is extremely challenging - because of both theoretical [Robinson and Siftaris 2004] and practical (available computational power) limitations. Even assuming the existence of a perfect registration algorithm, changes in scene and cameras can result in some areas of the photographed scene being superrepresented in many frames of the

ACM Trans. Graph., Vol. 38, No. 4, Article 28. Publication date: July 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:

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

[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  $\bar{u}_e^{ij}$  generated by setting  $f = e_i$  and  $g = e_j$  for every pair  $(i, j)$ , where the vectors  $\{e_1, e_2, e_3\}$  form an orthonormal bases spanning  $\mathbb{R}^3$ . Due to superposition, we can linearly combine  $\bar{u}_e^{ij}$  with scalar coefficients  $F_{ij}$ , and obtain a matrix-driven solution of (2) of the form

$$\bar{u}_e(r) = \sum_{ij} F_{ij} e_j \cdot \nabla(\mathcal{K}_e(r) e_i) = \nabla \mathcal{K}_e(r) : F, \quad (12)$$

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

By computing the spatial derivatives of  $u_e$ , we obtain the displacement field  $\bar{u}_e(r)$  in terms of the force matrix  $F$ :

$$\bar{u}_e(r) = -a \left( \frac{1}{r_e^3} + \frac{3e^2}{2r_e^5} \right) Fr + b \left[ \frac{1}{r_e^3} (F + F^t + \text{tr}(F)I) - \frac{3}{r_e^5} (r^t F r) I \right] r. \quad (14)$$

[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  $\mathbf{u}$  while equation 14 is implementable.

## 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.

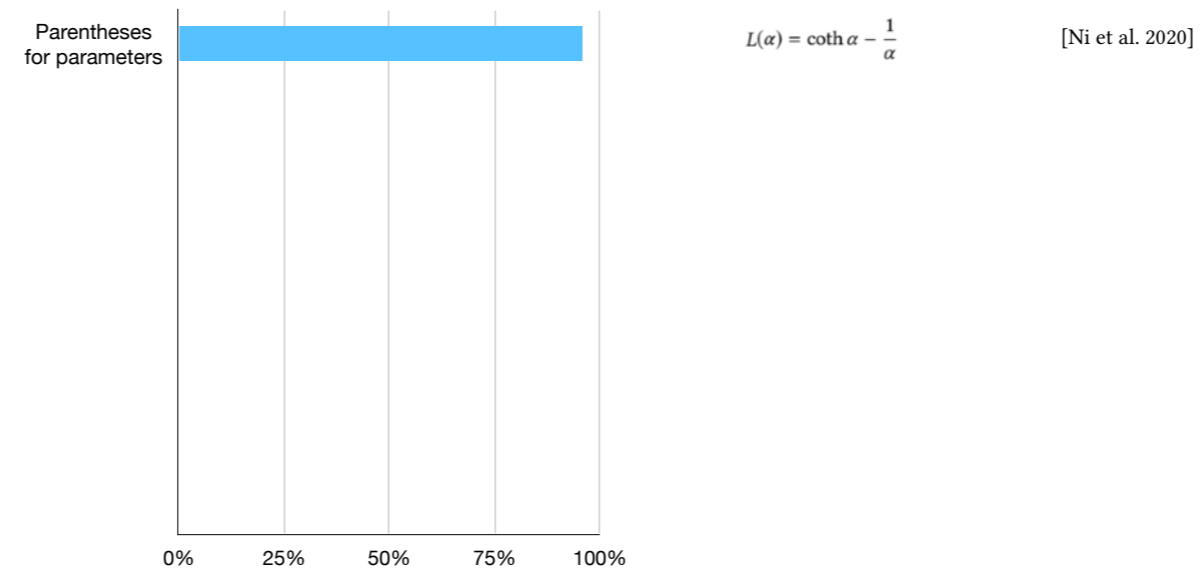
(I) Symbols and functions may be defined via conditional assignment which is a simple form of control flow

(II) Functions make use of a variety of implied semantics for parameters and pre-computed symbols.

$f_1(y) = \begin{cases} -\frac{y^2}{c^2 k^2} + \frac{2y}{c_0 k}, & y \in (0, h \epsilon_v) \\ 1, & y \geq h \epsilon_v. \end{cases} \quad (13)$	$R(\lambda) = r_{as} + \sum_{k=0}^{\infty} r_{as} r_{sa} \left( r_{1a}^2 e^{i\lambda \phi} \right)^k e^{i\lambda \phi} r_{sa} \quad (27)$	$\Psi(d) = \begin{cases} 1, & d = 0 \\ 1/n, & d > 0 \text{ and } d \leq h \\ 0, & d > h \end{cases} \quad (32)$	$E[(f)_{\text{SMS}}] = E[(f)_{\text{SMS}}]_{(t_1, x_1), \dots, (t_n, x_n)} \quad (33a)$
$\bar{V}_k(d) = \int_{\mathcal{V}^d} \gamma_k(d, \nabla d) dV, \quad (1)$	$E[L_{X_p}] = \int_0^\pi  \cos \phi_1  \frac{\sin^{n-2} \phi_1 d\phi_1}{i_{n-2}} \quad (24)$	$\log(\alpha_T) = \lambda^{-1} \int_{V(T)} w(r) \log(\alpha_T)^2 dr, \quad (3)$	$E[(f)_{\text{SMS}}] = E[E[(f)_{\text{SMS}}]_{t_1, \dots, t_n}   t_1, \dots, t_n] \quad (33b)$
$G_{\text{SMS}}(l) = \sum_i \sum_{m \in \mathcal{W}_i} \hat{f}_m^l(l) \hat{f}_m^l(l). \quad (3)$	$\bar{b}(f, g) = \begin{cases} \frac{1}{2} \  (x_g - x_f) - (r_{x_g} - r_{x_f}) \ _W^2 & \text{if }  f - g  = 1 \\ 0 & \text{otherwise.} \end{cases}$	$W_{\text{cloth}}(\lambda_1, \lambda_2) = \begin{cases} 0 & \lambda_1 < 1, \lambda_2 < 1 \\ W_{\text{SbVX}}(\lambda_1, \lambda_2, \lambda_1) & \lambda_1 \geq 1, \lambda_2 < \lambda_2(\lambda_1) \\ W_{\text{SbVX}}(\lambda_1, \lambda_2) & \text{otherwise.} \end{cases}$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} \quad (33c)$
$\pi(\mathbf{a} \mathbf{s}) = \sum_{i \in \mathcal{E}} w_i(\mathbf{s}) \pi_i(\mathbf{a} \mathbf{s}), \quad w_i(\mathbf{s}) = \frac{\exp(g_i(\mathbf{s}))}{\sum_{i \in \mathcal{E}} \exp(g_i(\mathbf{s}))} \quad (8)$	$J_{RL}(\theta) = E \left[ \sum_{t=0}^{\infty} \gamma^t r_t(s_t, a_t) \right]. \quad \Gamma_{n,b}(z) := \left( S(a, b) + z n(a, b) : z \in \left[ -\frac{h(a, b)}{2}, \frac{h(a, b)}{2} \right] \right)$	$i_{\Omega}(r) = \begin{cases} 1, & r \in \Omega \\ 1/2, & r \in \Sigma \\ 0, & r \in \Gamma \setminus (\Omega \cup \Sigma), \end{cases} \quad (9)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \int_{\mathcal{V}^d} w(x, t_i) \frac{f(x)}{\rho(x)} dx \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$D(n, m) = \sum_{k \in \mathcal{K}} d(n+k, m+k) \quad (1)$	$F_{g \rightarrow f}(X_p) = \frac{\rho(\mathbf{u}_b - \mathbf{u}_a) \cdot \mathbf{n}}{\Delta t} \quad \mathcal{L}(H(\bar{x}), \tilde{H}(\bar{x})) = \frac{1}{2} \sum_{\bar{x}} (H(\bar{x}) - \tilde{H}(\bar{x}))^2 \quad (16)$	$b(\alpha, \beta) := \begin{cases} \sum_{f=1}^n \ e_{f\alpha\beta}\  \left( 1 + \sum_{i \in \alpha \cap \beta} \ x_{f_i} - \bar{x}_i\  \right) & \text{if } q_\alpha \neq q_\beta \\ 0 & \text{otherwise,} \end{cases} \quad (3)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$d(n, m) = w_1 \frac{d_k}{ F } + w_2 \frac{d_c}{ F } + w_3 \frac{d_{cc}}{ F } + w_4 d_{\text{root}} \quad (1)$	$a^{(n)}(\mathbf{x}, t) = \int_{\mathcal{V}^d} \frac{(2\pi)^{-d/2}}{e^{-\ \mathbf{u}\ ^2/2}} H^{(n)}(\mathbf{u}) d\mathbf{u} \approx \sum_{i=0}^{q-1} \hat{f}_i(\mathbf{x}, t) H^{(n)}(\mathbf{e}_i), \quad (7)$	$c^{lm}(\mathbf{v}^q, \mathbf{v}^h, \omega_p^q, \omega^h) = \ \mathbf{v}^q - \mathbf{v}^h\ ^2 + \ \omega_p^q - \omega^h\ ^2. \quad (5)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$\Psi(F^S, J^S) = \Psi^S(F^S) + \Psi^S(J^S), \quad (9)$	$E_{T, \omega}(t) = \frac{1}{2} \frac{\  \text{Det}(Y) \ }{\sqrt{\text{Det}(Y^T Y)}} \quad (2)$	$\mathcal{L}_{\text{total}}(\theta) = \sum_d \lambda_d \ \Phi_d(U^*) - \Phi_d(f(U; \theta))\ _1$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$JSD(P  Q) = \frac{1}{2} D(P  M) + \frac{1}{2} D(Q  M) \quad (8)$	$\phi_H(\nabla \mathbf{u}) = \begin{cases} \frac{1}{2\alpha} (\nabla \mathbf{u})^2, &  \nabla \mathbf{u}  \leq \alpha \\  \nabla \mathbf{u}  - \frac{\alpha}{2}, &  \nabla \mathbf{u}  > \alpha \end{cases} \quad C_d(x) = \frac{1}{V_d(1)} \int_{-\infty}^x R_d(t) dt$	$\mathcal{L}_{\text{pos}}(\theta) = \ \mathbf{r} - f(U; \theta)\ _1$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$\pi(a_{t+1} s_t, c_t) = \frac{1}{Z(s_t, c_t)} \prod_{i=1}^k \theta_i^{a_i} \quad (3)$	$\text{corr}(x; T^c, \mathcal{P}^c) = \frac{1}{2} \sqrt{\text{Det}(X^T X + \alpha X^T X)} \quad (2)$	$\mathcal{L}(\theta) = 0.01 \times \mathcal{L}_{\text{total}}(\theta) + \mathcal{L}_{\text{pos}}(\theta), \quad (9)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$d(\mathcal{L}, \ell) = \frac{1}{ \ell } \int_{\ell} \min_{c_i \in \mathcal{L}} \text{dist}(\ell_i, p_{\theta}) d p_{\theta}, \quad (2)$	$q^f(\mathbf{x}) = \sum_c q_c^f N_c^{f-1}(\mathbf{x}), \quad (28)$	$VTV[h] = \sup_{\phi \in \mathcal{L}, \mathbf{v}, \ \phi(\mathbf{x})\ _r \leq 1} \left( \sum_{i=1}^n \int_{\Omega} h_i \nabla \cdot \phi_i \right), \quad (2)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$Q(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2 2\theta & \sin 2\theta \cos 2\theta & -\sin 2\theta \\ 0 & \sin 2\theta \cos 2\theta & \sin^2 2\theta & \cos 2\theta \\ 0 & \sin 2\theta & -\cos 2\theta & 0 \end{bmatrix}, \quad (2)$	$\mathcal{L}_{i \rightarrow j}^{\text{spatial}}(\mathbf{x}) = \ \mathbf{p}_{i \rightarrow j}(\mathbf{x}) - \mathbf{f}_{i \rightarrow j}(\mathbf{x})\ _2, \quad (11)$	$E^k(s) = \frac{\int_{\omega_k/\sqrt{2}}^{\omega_k/\sqrt{2}}  F(S)(s, \omega) ^2 d\omega}{\omega_k(\sqrt{2} - 1/\sqrt{2})} \quad \text{and } S^k(s) = 10 \log_{10} E^k(s), \quad (20)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$b_3(x_3, y_3, \lambda) = \tilde{b}_3(x_3, y_3, \lambda) d(x_3, y_3) = \frac{h(x_3, y_3, \lambda)}{(j\lambda f)^2 A} \left( \frac{x_3 + x_3}{\lambda f}, \frac{y_3 + y_3}{\lambda f} \right) \frac{1}{v_0} \sum_{k=-\infty}^{\infty} \delta \left( x_3 - \frac{k}{v_0} \right), \quad (19)$	$f_s(S_n(\mathbf{y}), \mathbf{y}) = -\frac{1}{(n+1)!} (x^{(n+1)}, \mathbb{S}^{n+1} \mathbf{y}) + o( \mathbf{y} ^{n+1}), \quad (16)$	$W_f^{(-)}(x) = \max_{\eta \in H_f^{(-)}} K_{\eta}(x), \quad (4)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$V[\ell_i] = \frac{1}{M} \sum_{p=1}^M V \left[ \frac{w_i f}{p} \right] - \frac{1}{MN} \sum_{p=1}^M V \left[ \frac{w_i f}{q} \right] + \frac{1}{N} V \left[ \frac{w_i f}{q} \right] + \left( 1 - \frac{1}{N} \right) V \left[ \frac{1}{\rho(Z_i)} \int_{\mathcal{R}} w_i(g_2) f(g_2) d\mu(g) \right] - \frac{1}{M} \left( 1 - \frac{1}{N} \right) \sum_{p=1}^M V \left[ \frac{1}{\rho(Z_i)} \int_{\mathcal{R}} w_i(g_2) f(g_2) d\mu(g) \right], \quad (9)$	$\text{Var}[\mathbf{x}^*] = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}^{-1} \cdot \frac{\partial^2 f}{\partial x \partial y} \cdot \text{Var}[y] \cdot \frac{\partial^2 f}{\partial y \partial x} \cdot \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}^{-1} = \sigma^2 \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}^{-1}, \quad (56)$	$E(\Phi) = \int_{\mathcal{R}} \ \mathbb{1}\ _2^2 dA_{\mathcal{R}} + \int_{\mathcal{A}} \ \mathbb{1}^{-1}\ _2^2 dA_{\mathcal{A}} \quad (8)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$E(\mathbf{X}) = \sum_{i=1}^K \sum_{j=0}^{N-1} \lambda_j \sum_{m=0}^K \gamma_{ij}(f_m, v) \omega_{ij}, \quad (14)$	$t(j) = \begin{cases} p_k, & \text{if } \exists k : j = \Delta_{\Sigma}(k) \\ \{t_{\Psi_k, \Psi_{k+1}}(\psi(j))\}, & \text{else } \exists k : \Delta_{\Sigma}(k) < j < \Delta_{\Sigma}(k+1) \end{cases} \quad (14)$	$K_{\eta}(x) = \alpha e^{-\left( \frac{(\theta - \theta_0)^2}{\sigma_{\theta}^2} + \frac{(\theta - \theta_0)^2}{\sigma_{\theta}^2} \right)}, \quad (4)$	$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$
$D_{\text{iso}}(\theta) = \begin{cases} a_1 \theta + b_1 & \theta_0 = 0 \leq \theta \leq \theta_1 \\ a_2 \theta + b_2 & \theta_1 \leq \theta \leq \theta_2 \\ \dots & \dots \\ a_n \theta + b_n & \theta_{n-1} \leq \theta \leq \theta_n = \frac{\pi}{2}. \end{cases} \quad (11)$			$E[(f)_{\text{SMS}}] = E \left[ \sum_{i=1}^n \sum_{m \in \mathcal{W}_i} w(x_i, t_i) \frac{f(x_i)}{\rho(x_i)} \right]_{t_1, \dots, t_n} = I. \quad (33d)$

Quantitatively, we manually observed the 916 function definitions across the 156 SIGGRAPH papers.

## Analysis of all 916 function definitions at SIGGRAPH 2020

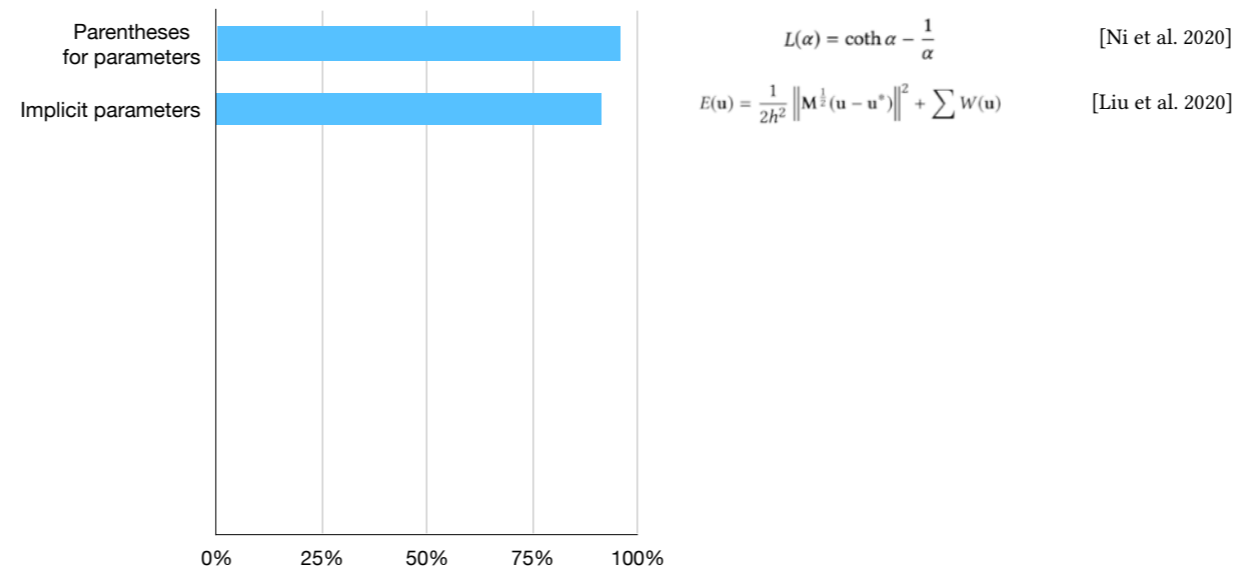


We take an empiric approach to categorize these Equations.

Here is the overview of them.

96% use parentheses for parameters,

## Analysis of all 916 function definitions at SIGGRAPH 2020



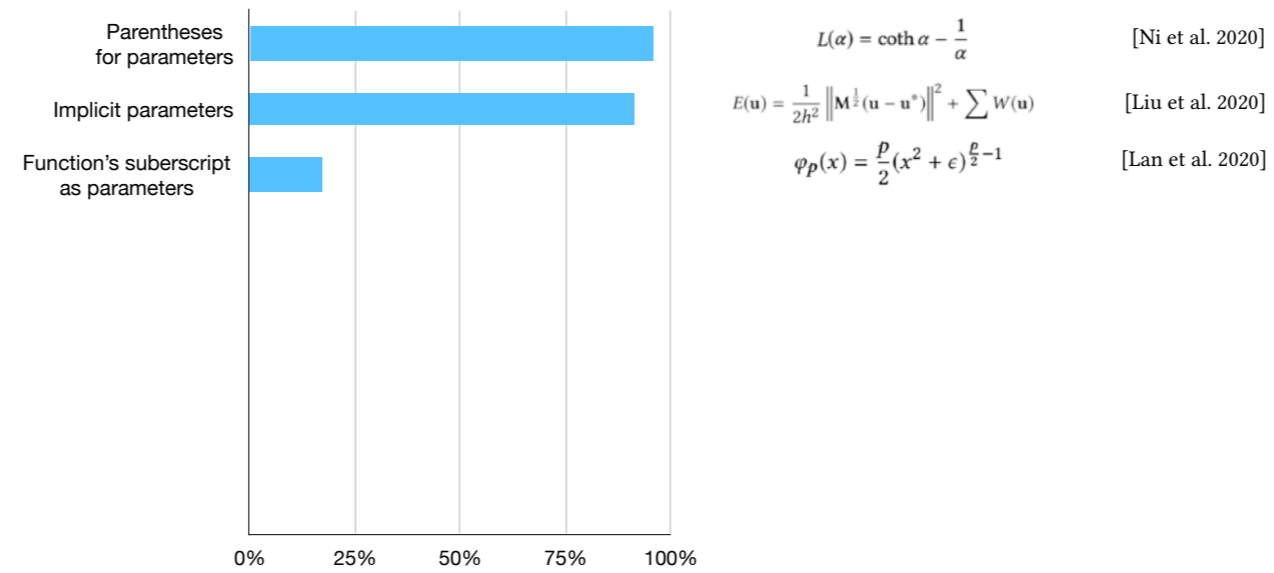
$$L(\alpha) = \coth \alpha - \frac{1}{\alpha} \quad [\text{Ni et al. 2020}]$$

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

91% rely on implicit parameters,

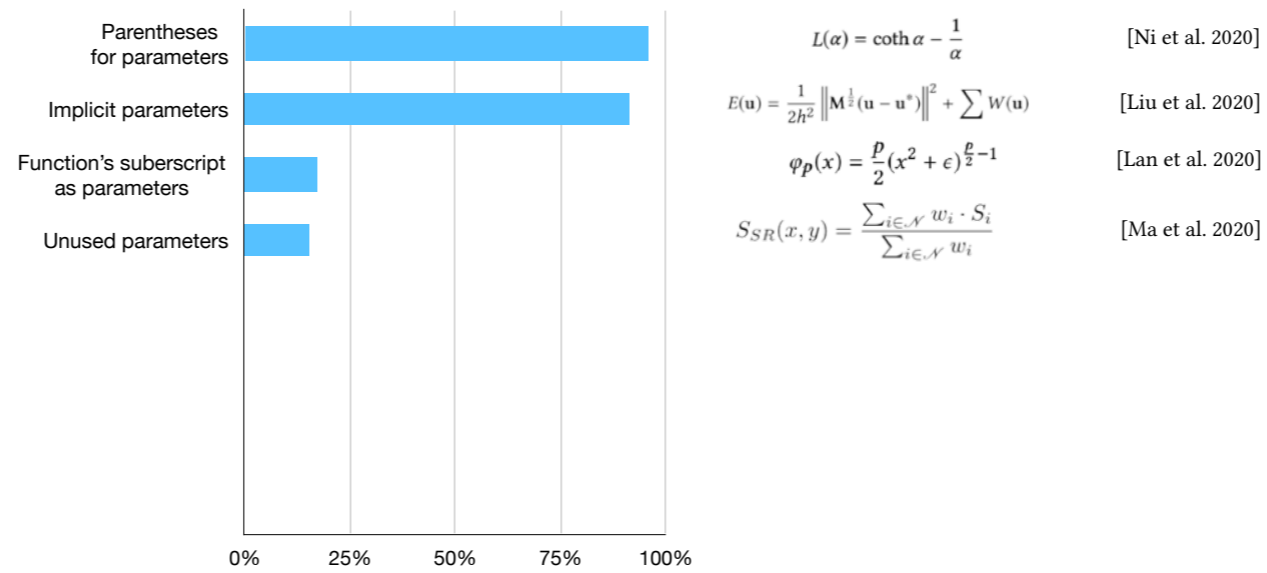


## Analysis of all 916 function definitions at SIGGRAPH 2020



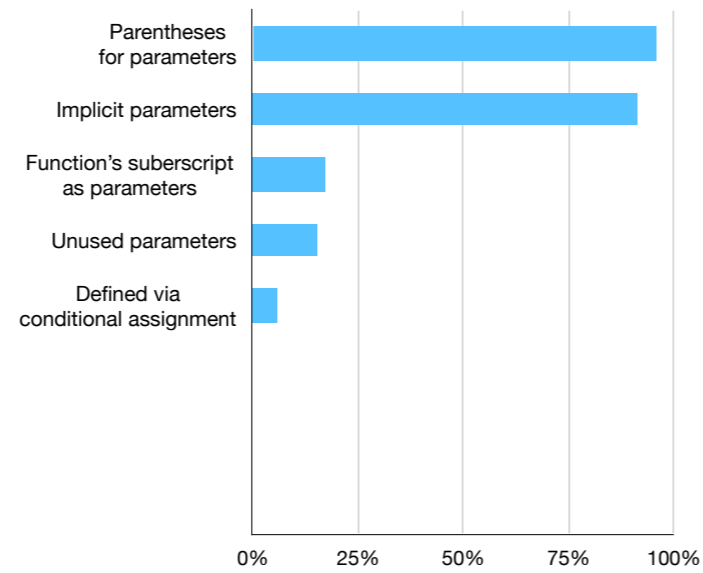
17% interpret the function's subscript as parameters,

## Analysis of all 916 function definitions at SIGGRAPH 2020



15% have seemingly unused 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]

$$\varphi_p(x) = \frac{p}{2}(x^2 + \epsilon)^{\frac{p}{2}-1}$$

[Lan et al. 2020]

$$S_{SR}(x, y) = \frac{\sum_{i \in \mathcal{N}} w_i \cdot S_i}{\sum_{i \in \mathcal{N}} w_i}$$

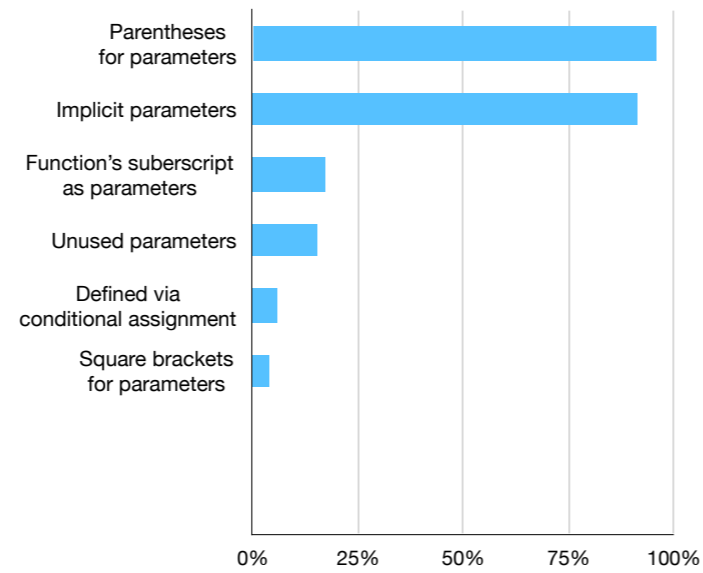
[Ma et al. 2020]

$$W(r, h)_{\text{cubic}} = \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^3, & 0 \leq r \leq 1, \\ \frac{1}{6}(2-r)^3, & 1 \leq r \leq 2, \\ 0, & r > 2. \end{cases}$$

[Kim et al. 2020]

6% are defined via conditional assignment

## Analysis of all 916 function definitions at SIGGRAPH 2020



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

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

$$\varphi_p(x) = \frac{p}{2}(x^2 + \epsilon)^{\frac{p}{2}-1} \quad [\text{Lan et al. 2020}]$$

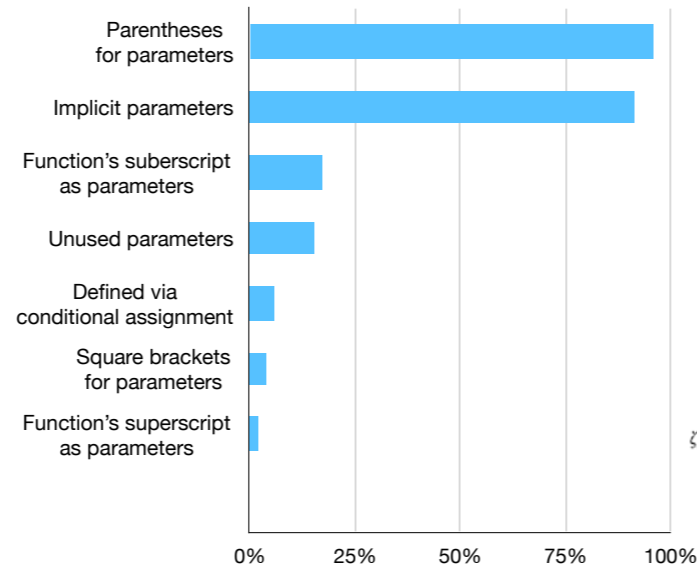
$$S_{SR}(x, y) = \frac{\sum_{i \in \mathcal{N}} w_i \cdot S_i}{\sum_{i \in \mathcal{N}} w_i} \quad [\text{Ma et al. 2020}]$$

$$W(r, h)_{\text{cubic}} = \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^3, & 0 \leq r \leq 1, \\ \frac{1}{6}(2-r)^3, & 1 \leq r \leq 2, \\ 0, & r > 2. \end{cases} \quad [\text{Kim et al. 2020}]$$

$$E[L_{x_p}] = \frac{2}{(n-1)\sqrt{\pi}} \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-1}{2}\right)} = \frac{2}{n\sqrt{\pi}} \frac{\Gamma\left(\frac{n+2}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)} \quad [\text{Chiu et al. 2020}]$$

4% use square brackets for parameters,

## Analysis of all 916 function definitions at SIGGRAPH 2020



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

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

$$\varphi_p(x) = \frac{p}{2}(x^2 + \epsilon)^{\frac{p}{2}-1} \quad [\text{Lan et al. 2020}]$$

$$S_{SR}(x, y) = \frac{\sum_{i \in \mathcal{N}} w_i \cdot S_i}{\sum_{i \in \mathcal{N}} w_i} \quad [\text{Ma et al. 2020}]$$

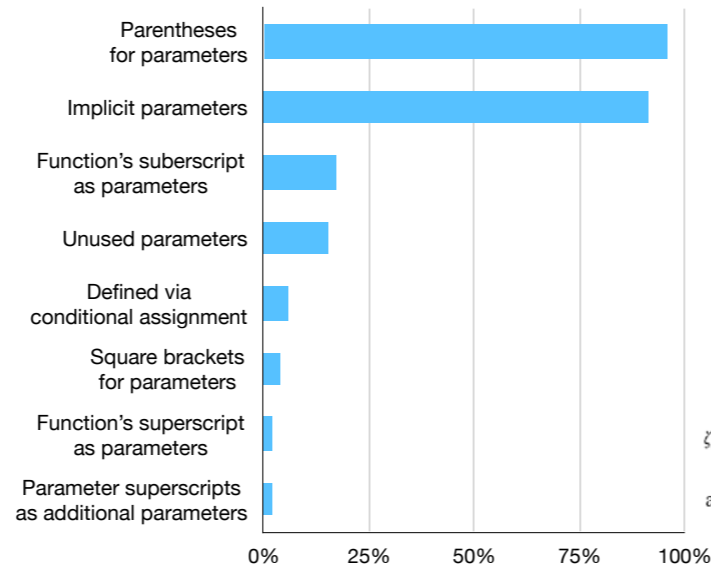
$$W(r, h)_{\text{cubic}} = \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^3, & 0 \leq r \leq 1, \\ \frac{1}{6}(2-r)^3, & 1 \leq r \leq 2, \\ 0, & r > 2. \end{cases} \quad [\text{Kim et al. 2020}]$$

$$E[L_{x_p}] = \frac{2}{(n-1)\sqrt{\pi}} \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-1}{2}\right)} = \frac{2}{n\sqrt{\pi}} \frac{\Gamma\left(\frac{n+2}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)} \quad [\text{Chiu et al. 2020}]$$

$$\zeta_s^\alpha(x) \equiv \frac{2^j \alpha^{-1}}{(2\pi)^{3/2}} \sum_n \beta_{j,n}^i \zeta_s^{\alpha,n}(x) = \int_{\mathbb{R}_u} \psi_s(S_\alpha(x, u)^T) du \quad [\text{Lessig 2020}]$$

2% interpret the function's superscript as parameters,

## Analysis of all 916 function definitions at SIGGRAPH 2020



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

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

$$\varphi_p(x) = \frac{p}{2}(x^2 + \epsilon)^{\frac{p}{2}-1} \quad [\text{Lan et al. 2020}]$$

$$S_{SR}(x, y) = \frac{\sum_{i \in \mathcal{N}} w_i \cdot S_i}{\sum_{i \in \mathcal{N}} w_i} \quad [\text{Ma et al. 2020}]$$

$$W(r, h)_{\text{cubic}} = \begin{cases} \frac{2}{3} - r^2 + \frac{1}{2}r^3, & 0 \leq r \leq 1, \\ \frac{1}{6}(2-r)^3, & 1 \leq r \leq 2, \\ 0, & r > 2. \end{cases} \quad [\text{Kim et al. 2020}]$$

$$E[L_{x_p}] = \frac{2}{(n-1)\sqrt{\pi}} \frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})} = \frac{2}{n\sqrt{\pi}} \frac{\Gamma(\frac{n+2}{2})}{\Gamma(\frac{n+1}{2})} \quad [\text{Chiu et al. 2020}]$$

$$\zeta_s^\alpha(x) \equiv \frac{2^j \alpha^{-1}}{(2\pi)^{3/2}} \sum_n \beta_{j,n}^i \zeta_s^{\alpha,n}(x) = \int_{\mathbb{R}_u} \psi_s(S_\alpha(x, u))^T du \quad [\text{Lessig 2020}]$$

$$\text{area}(f^\delta) = \text{area}(f)(1 - 2\delta H(f) + \delta^2 K(f)) \quad [\text{Jiang et al. 2020}]$$

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.

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

We design H♥rtDown based on the previous formative study



# H♥rtDown Design: Authoring

- Context definition

```
4 # Surface Fairing
5 ♥: fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A
  simple graph Laplacian  $LS$  can be written in terms of the
  adjacency matrix  $SA$  and the degree matrix  $SDS$ . Those
  matrices can be derived purely from the edges of the mesh
   $SE$ .
8 ````heartla
9  $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$ 
10
11  $D_{ii} = \sum_j A_{ij}$ 
12  $L = D^{-1} (D - A)$ 
13 where
14  $E \in \{ Z \times Z \}$  index
15  $A \in R^{(n \times n)}$ : The adjacency matrix
16  $n \in Z$ : The number of mesh vertices
17 ````
18
19
20 We then solve a system of equations  $SLx = 0$  for free vertices to obtain the fair
  surface. We can write the fair mesh vertices  $SV$  directly
  given boundary constraints provided as a binary vector  $BS$  with
  1's for boundary vertices, a large scalar constraint
  weight  $w=10^6$ , and 3D vertices for the constrained mesh
   $VS$ :
```

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. This allows better symbol and formula re-use

Later context declarations override earlier declarations.

The context disambiguates symbol reuse (and corresponds to a concept of re-usable modules).

## H♥rtDown Design: Authoring

- Prose descriptions

```
4 # Surface Fairing
5 ♥: fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A
simple graph Laplacian  $L$  can be written in terms of the
adjacency matrix  $A$  and the degree matrix  $D$ . Those
matrices can be derived purely from the edges of the mesh
 $E$ .
8 ``heartla
9  $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$ 
10
11  $D_{ii} = \sum_j A_{ij}$ 
12  $L = D^{-1} (D - A)$ 
13 where
14  $E \in \{ Z \times Z \}$  index
15  $A \in R^{(n \times n)}$ : The adjacency matrix
16  $n \in Z$ : The number of mesh vertices
17 ``
18
19
20 We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair
surface. We can write the fair mesh vertices  $V'$  directly
given boundary constraints provided as a binary vector  $B$  with
1's for boundary vertices, a large scalar  $w$  constraint
weight, and 3D vertices for the constrained mesh
 $V$ :
```

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

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

## H♥rtDown Design: Authoring

- Executable mathematical expressions

```
4 # Surface Fairing
5 ♥: fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A
  simple graph Laplacian  $SL$  can be written in terms of the
  adjacency matrix  $SA$  and the degree matrix  $SDS$ . Those
  matrices can be derived purely from the edges of the mesh
   $SE$ .
8
9  $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$ 
10
11  $D_{ii} = \sum_j A_{ij}$ 
12
13  $L = D^{-1} (D - A)$ 
14 where
15  $E \in \{ Z \times Z \}$  index
16  $A \in R^{(n \times n)}$ : The adjacency matrix
17  $n \in Z$ : The number of mesh vertices
18
19
20 We then solve a system of equations  $SLx = 0$  for free vertices to obtain the fair
  surface. We can write the fair mesh vertices  $SV$  directly
  given boundary constraints provided as a binary vector  $BS$  with
  1's for boundary vertices, a large scalar constraint
  weight  $w=10^6$ , and 3D vertices for the constrained mesh
   $VS$ :
```

Authors can write executable mathematical expressions in different I♥LA blocks and inline I♥LA formula

I♥LA requires type declarations for all symbols not appearing on the left-hand side

## H♥rtDown Design: Authoring

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

Original I♥LA can't handle multiple equations in papers, we need to add 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

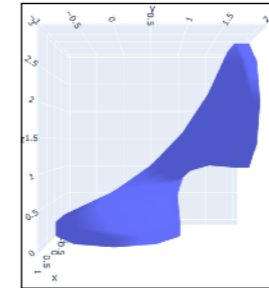
We add Modules from another I♥LA file to support different contexts

We also modify MathJax output to include the metadata

# H♥rtDown Design: Authoring

- Figures

```
30 #Figure
31 """
32 from lib import *
33 import rake_cylinder
34
35 # Load cylinder with n vertices
36 mesh = rake_cylinder.rake_cylinder(10, 10)
37 rake_cylinder.save_obj(mesh, "input.obj", clobber = True)
38 V = mesh.v
39 F = mesh.fv
40 n = len(V)
41
42 # Extract the mesh edges
43 edges = set()
44 for face in F:
45     for i in range(3):
46         vi, vj = face[Fv[i]], face[Fv[(i+1)%3]]
47         edges.add( ( min(vi,vj), max(vi,vj) ) )
48
49 # The constraint vector is all vertices with z < 3/4 or z > 3/4
50 k = rp.zeros( n, dtype = int )
51 [ k[V[:,2] < 3/4] = 1
52   k[V[:,2] > 3/4] = 1
53
54 # Rotate the top around the z axis by 90 degrees.
55 k = rp.array([[ 1, 0, 0 ],
56              [ 0, 0, 1 ],
57              [ 0, -1, 0 ]])
58 for vi in np.where(k[:,:] > 3/4)[0]: V[vi] = k * V[vi] + (0,1,0)
59
60 # Solve for new positions
61 result = fairing( E = edges, n = n, k = k, V = V )
62 mesh.v = result.V_astroptre
63 rake_cylinder.save_obj(mesh, "solved.obj", clobber = True)
64
65 import plotly.graph_objects as go
66 fig = go.Figure(data=[go.Mesh3d(
67     x=mesh.v[:,0], y=mesh.v[:,1], z=mesh.v[:,2],
68     i=mesh.fv[:,0], j=mesh.fv[:,1], k=mesh.fv[:,2]
69 )])
70 fig.update_layout(scene_camera={"eye":dict(x=2.5,y=0,z=0), "up":dict(x=0,y=0,z=1)}, margin=dict(l=0, r=0, b=0))
71 fig.write_html("cylinder.html")
72
73 
74 <figcaption>Fairing the middle half of a cylinder.</figcaption>
75 </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 Editor

```

7 Surface fairing given boundary constraints depends on the order of the Laplacian. A simple
8
9 graph Laplacian L = D - A can be written in terms of the adjacency matrix A and the
10 degree matrix D. These matrices can be derived purely from the edges of the mesh
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

```

### 1 Surface Fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A simple graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the degree matrix  $D$ . These matrices can be derived purely from the edges of the mesh  $E$ :

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$D_{ii} = \sum_j A_{ij}$$

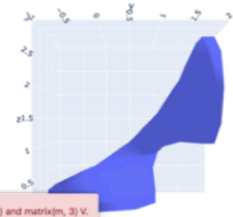
$$L = D - A$$

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

$$V' = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V) \quad (2)$$

**Glossary of fairing**

- $A \in \mathbb{R}^{n \times n}$ : The adjacency matrix
- $B \in \mathbb{R}^n$ : boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices
- $D \in \mathbb{R}^{n \times n}$ : degree matrix
- $E$ : set type: the edges of the mesh
- $L \in \mathbb{R}^{n \times n}$ : graph Laplacian
- $V \in \mathbb{R}^{n \times 3}$ : 3D vertices for the constrained mesh
- $V' \in \mathbb{R}^{n \times 3}$ : the fair mesh vertices
- $n \in \mathbb{Z}$ : The number of mesh vertices
- $w \in \mathbb{R}$ : constraint weight



Fairing the middle half of a cylinder.

Dimension mismatch. Can't multiply matrix(n, n) w diag(B) and matrix(n, 3) V.  
 $V' = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V)$

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 LaTeX compiler error message and highlights the appropriate line in the source.

# H♥rtDown Design: Author support

H♥rtDown Editor

```
1 full_paper: false
2
3 # Surface Fairing
4 # Surface Fairing
5 # Surface Fairing
6
7 Surface fairing given boundary constraints depends on the order of the Laplacian. A simple
8
9 The graph Laplacian  $L \in \mathbb{R}^{n \times n}$  can be written in terms of the adjacency matrix  $A$  and the degree
10 matrix  $D$ . Those matrices can be derived purely from the edges of the mesh  $E$ .
11
12 Inherently
13  $A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$ 
14
15  $D_{ii} = \sum_j A_{ij}$ 
16 where
17  $L = D - A$ 
18  $E \in \mathbb{Z}^2$  index
19  $A \in \mathbb{R}^{n \times n}$ : The adjacency matrix
20  $n \in \mathbb{Z}$ : The number of mesh vertices
21
22 We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write
23
24 The fair mesh vertices  $V^f \in \mathbb{R}^{n \times 3}$  directly given boundary constraints
25 provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar
26 constraint weight  $w$ , and  $n$  3D vertices for the constrained mesh
27  $V^c \in \mathbb{R}^{n \times 3}$ .
28
29 Inherently
30  $V^f = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V^c)$ 
31 where
32  $B \in \mathbb{Z}^n$ 
33  $V^c \in \mathbb{R}^{n \times 3}$ 
34
35 # Figure
36 # Python
37 from math import pi
38 import numpy as np
39 import trimesh
40
41 # Load cylinder with n vertices
42 mesh = trimesh.creation.cylinder(10, 10)
43 mesh.vertices[:, 2] = np.clip(mesh.vertices[:, 2], -1, 1)
44 V = mesh.vertices
45 n = len(V)
46
47 # Extract the mesh edges
48 edges = mesh.edges
49 for face in mesh.faces:
50     v1, v2 = face[0], face[1]
51     edges.add((min(v1, v2), max(v1, v2)))
52
53 # The constraint vector is all vertices with  $x < -1/4$  or  $x > 1/4$ 
54 B = np.zeros(n, dtype=int)
55 B[V[:, 0] < -1/4] = 1
56 B[V[:, 0] > 1/4] = 1
57
58 # Rotate the top around the z axis by 90 degrees.
59 V = np.array([0, 0, 1], dtype=float)
60
```

## 1 Surface Fairing

Surface fairing given boundary constraints depends on the order of the Laplacian. A simple graph Laplacian  $L$  can be written in terms of the adjacency matrix  $A$  and the degree matrix  $D$ . Those matrices can be derived purely from the edges of the mesh  $E$ .

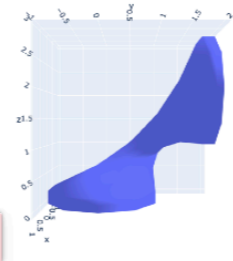
$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise} \end{cases}$$
$$D_{ii} = \sum_j A_{ij}$$
$$L = D - A$$

We then solve a system of equations  $Lx = 0$  for free vertices to obtain the fair surface. We can write the fair mesh vertices  $V^f$  directly given boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices, a large scalar constraint weight  $w$ , and 3D vertices for the constrained mesh  $V^c$ .

$$V^f = (L + w \text{diag}(B))^{-1} (w \text{diag}(B) V^c)$$

## Glossary of fairing

- $A \in \mathbb{R}^{n \times n}$ : The adjacency matrix
- $B \in \mathbb{Z}^n$ : boundary constraints provided as a binary vector  $B$  with 1's for boundary vertices
- $D \in \mathbb{R}^{n \times n}$
- $E \in \mathbb{Z}^2$ : type: the edges of the mesh  $E$
- $L \in \mathbb{R}^{n \times n}$ : graph Laplacian
- $V^c \in \mathbb{R}^{n \times 3}$ : 3D vertices for the constrained mesh  $V^c$
- $V^f \in \mathbb{R}^{n \times 3}$ : the fair mesh vertices  $V^f$
- $n \in \mathbb{Z}$ : The number of mesh vertices
- $w \in \mathbb{R}$ : constraint weight



Fairing the middle half of a cylinder.

Missing descriptions for symbols:  
fairing:  $D$

Compile

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

# H♥rtDown Design: Reading Environment

### A Symmetric Objective Function for ICP

SZYMON RUSINKIEWICZ, Princeton University

The Iterative Closest Point (ICP) algorithm, commonly used for alignment of 3D models, has previously been defined using either a point-to-point or point-to-plane objective. Alternatively, researchers have proposed computationally-expensive methods that directly minimize the distance function between surfaces. We introduce a new symmetric objective function that achieves the simplicity and computational efficiency of point-to-plane optimization, while yielding improved convergence speed and a wider convergence basin. In addition, we present a linearization of the objective that is exact in the case of exact correspondences. We experimentally demonstrate the improved speed and convergence basin of the symmetric objective, on both smooth models and challenging cases involving noise and partial overlap.

#### 1 INTRODUCTION

Registration of 3D shapes is a key step in both 3D model creation (from scanners or computer vision systems) and shape analysis. For rigid-body alignment based purely on geometry (as opposed to RGB-D), the most common methods are based on variants of the Iterative Closest Point (ICP) algorithm [Besl and McKay 1992]. In this method, points are repeatedly selected from one model, their nearest points on the other model (given the current best-estimate rigidbody alignment) are selected as correspondences, and an incremental transformation is found that minimizes distances between point pairs. The algorithm eventually converges to a local minimum of surface-to-surface distance.

Because ICP-like algorithms can be made efficient and reliable, they have become widely adopted. As a result, researchers have focused on both addressing the shortcomings of ICP and extending it to new settings such as color-based registration and non-rigid alignment. One particular class of improvements has focused on the loss function that is optimized to obtain an incremental transformation. For example, as compared to the original work of Besl and McKay, which minimized point-to-point distance, the method of [Chen and Medioni 1992] minimized the distance between a point on one mesh and a plane containing the matching point and perpendicular to its normal. This point-to-plane objective generally results in faster convergence to the correct alignment and greater ultimate accuracy, though it does not necessarily increase the basin of convergence. Work by [Fitzgibbon 2003], [Mitra et al. 2004], and [Fotran et al. 2006] showed that both point-to-point and point-to-plane minimization may be thought of as approximations to minimizing the squared Euclidean distance function of the surface, and they presented algorithms that achieved greater con-

#### Glossary of ICP

- $\bar{p} \in \mathbb{R}^3$ : the averaged coordinate of points
- $\bar{q} \in \mathbb{R}^3$ : the averaged coordinate of points
- $\epsilon_{plane} \in \mathbb{R}$ : the point-to-plane objective
- $\epsilon_{point} \in \mathbb{R}$ : the point-to-point objective
- $\epsilon_{symm-RN} \in \mathbb{R}$ : the rotated-normals ("RN") version of the symmetric objective
- $\epsilon_{symm} \in \mathbb{R}$ :  $\epsilon_{symm}$  as the symmetric objective
- $\epsilon_{two-plane} \in \mathbb{R}$ : the sum of squared distances to planes defined by both  $n_x$  and  $n_y$
- $n_x \in \text{sequence of } \mathbb{R}^3$ : the surface normals
- $n_y \in \text{sequence of } \mathbb{R}^3$ : surface normals  $n_{y,i}$
- $R \in \mathbb{R}^{3 \times 3}$ : a rigid-body transformation ( $RH$ ) such that applying the transformation to  $P$  causes it to lie on top of  $Q$
- $S \in \mathbb{R}^{3 \times 4}$
- $\alpha \in \mathbb{R}^3$ :  $\alpha$  and  $\theta$  are the axis and angle of rotation
- $n \in \text{sequence of } \mathbb{R}^3$
- $p \in \text{sequence of } \mathbb{R}^3$ : pairs of corresponding points  $(p_i, q_i)$ , where  $q_i$  is the closest point to  $p_i$  given the current transformation
- $P \in \text{sequence of } \mathbb{R}^3$
- $q \in \text{sequence of } \mathbb{R}^3$ : pairs of corresponding points  $(p_i, q_i)$ , where  $q_i$  is the closest point to  $p_i$  given the current transformation
- $Q \in \text{sequence of } \mathbb{R}^3$
- $rot \in \mathbb{R}, \mathbb{R}^3 \rightarrow \mathbb{R}^{3 \times 3}$ : the rotation function
- $t \in \mathbb{R}^3$ : a rigid-body transformation ( $RH$ ) such that applying the transformation to  $P$  causes it to lie on top of  $Q$
- $trans \in \mathbb{R}^3 \rightarrow \mathbb{R}^{3 \times 4}$ : the translation function
- $\Gamma \in \mathbb{R}^3$
- $d \in \mathbb{R}^3$
- $\theta \in \mathbb{R}$ :  $\alpha$  and  $\theta$  are the axis and angle of rotation

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

We were inspired by ScholarPhi [Head et al. 2021] but aims to support all enhanced reader environments as a paper writing environment



# 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.

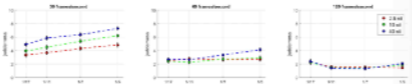


Fig. 4. Results for experiment 1 (moving edge), averaged over participants and contrasts. Vertical lines depict standard error over all samples. Results for 120 (right) and 60 FPS (mid) show little judder. Thirty FPS (left) appeared considerably distorted—judder increases almost linearly with speed,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

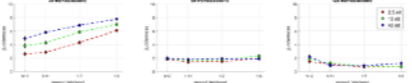


Fig. 5. Results for experiment 2 (panning complex images), averaged over participants and images. Vertical lines depict standard error over all samples. Results are similar to experiment 1, with 120 (right) and 60 FPS (mid) not showing much judder. Thirty FPS (left) continues to present a positive and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

Glossary of judder

- $F_i \in \mathbb{R}$ : Denoting  $F_i$  and  $F_i$  as the two frame rates
- $F_i \in \mathbb{R}$ : Denoting  $F_i$  and  $F_i$  as the two frame rates
- $L_i \in \mathbb{R}$ :  $L_i$ ,  $L_i$  as the luminances
- $L_i \in \mathbb{R}$ :  $L_i$ ,  $L_i$  as the luminances
- $CFP \in \mathbb{R} \rightarrow \mathbb{R}$ : the critical flicker fusion rate (CFP)
- $F$ : 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  $M$
- $P \in \mathbb{R}, \mathbb{R} \rightarrow \mathbb{R}$
- $S \in \mathbb{R}$ : speed  $S$
- $a \in \mathbb{R}$ :  $a$  and  $b$  are known constants
- $b \in \mathbb{R}$ :  $a$  and  $b$  are known constants
- $\alpha \in \mathbb{R} \rightarrow \mathbb{R}$ :  $\alpha$  the logarithm function
- $\beta \in \mathbb{R} \rightarrow \mathbb{R}$ :  $\beta$  is the multiplicative inverse

H♥rtDown displays the glossary on the right side of the window in a fixed position as the page scrolls.

The glossary in H♥rtDown is context-dependent: when the page scrolls to a different context, the glossary updates automatically with the relevant symbol list.

## H♥rtDown Design: Reading Environment

- Symbol definitions

approximate the surface around  $q_i$  as planar, which only requires evaluation of surface normals  $\vec{n}_{q_i}$ . Indeed, this approach dates back to the work of [Chen and Medioni 1992], who minimized what has come to be called the point-to-plane objective :

$$\varepsilon_{plane} = \sum_i ((Rp_i + t - q_i) \cdot \vec{n}_{q_i})^2 \quad (2)$$

It can be shown that minimizing this  $\vec{n}_q \in \text{sequence of } \mathbb{R}^3: \text{ surface normals } n_{q_i}$  minimiza-

This figure shows the visualization we provide when clicking on the symbol in an equation

# H♥rtDown Design: Reading Environment

- Equation relationships

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

$$Rv \approx v \cos \theta + (a \times v) \sin \theta = \cos \theta (v + \bar{a} \times v) \quad (8)$$

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

$$\varepsilon_{\text{approx}} \approx \sum_i (\cos \theta (p_i - q_i) \cdot n_i + \cos \theta (\bar{a} \times (p_i + q_i)) \cdot n_i + \bar{a} \cdot n_i)$$

$$\varepsilon_{\text{approx}} \approx \sum_i \cos(\theta)^2 ((p_i - q_i) \cdot n_i + ((p_i + q_i) \times n_i) \cdot \bar{a} + n_i \cdot \bar{a})^2 \quad (9)$$

where  $n_i = n_{x_i} + n_{y_i}$  and  $\bar{a} = \frac{a}{\cos(\theta)}$ . We now make the additional approximation of weighting the objective by  $1/\cos^2 \theta$ , which approaches 1 for small  $\theta$ . 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:

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

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

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

When clicking an equation, H♥rtDown highlights the terms involved in a symbol's definition as well as downstream uses of the symbol.

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)

```
1 full_paper: false
2
3 # clustering
4 # k-means
5
6 In k-means clustering, we are given a sequence of data  $x_i, i \in [n]$ . We want to cluster the data into  $k \in \mathbb{N}$ 
7 clusters. First, we initialize the cluster_centers  $c_i, i \in [k]$  arbitrarily.
8 Then we iteratively update cluster centers. The updated cluster centers are the points which minimize the
9 sum of squared distances to all cluster_centers points  $c_i$  which are closer to  $c_i$  than any other
10 cluster  $c_j, j \neq i$ .
11
12 ---
13 import numpy as np
14 from IPython.display import FigureWidget
15 from IPython.display import Image
16
17 # Generate 2D data
18 # k, n = np.random.randint(10, 20, 2) * 5 - 0.5
19 # x, y = np.random.randn(n, 2)
20 # x, y = np.random.randn(n, 2)
21 # x, y = np.random.randn(n, 2)
22 # x, y = np.random.randn(n, 2)
23 # x, y = np.random.randn(n, 2)
24 # x, y = np.random.randn(n, 2)
25 # x, y = np.random.randn(n, 2)
26 # x, y = np.random.randn(n, 2)
27 # x, y = np.random.randn(n, 2)
28 # x, y = np.random.randn(n, 2)
29 # x, y = np.random.randn(n, 2)
30 # x, y = np.random.randn(n, 2)
31
32 # Initial cluster centers
33 k = 4
34 c_i = np.random.randn(k, 2)
35
36 iterations = 0
37 while True:
38     iterations += 1
39     # All distances give us labels
40     d_i = np.sqrt([[(x - c_i)**2 + (y - c_i)**2] for x, y in zip(x, y)])
41     labels = d_i.argmin(axis=1)
42
43     # Update c_i with the minimization algorithm
44     c_i = np.mean([x[labels == i] for i in range(k)])
45
46     if np.allclose(c_i, c_i, atol=1e-6) or iterations > 100: break
47     c_i = c_i.copy()
48
49 # Plot
50 fig = FigureWidget()
51 fig.scatter(x=x, y=y, color=labels.astype('str'))
52 fig.scatter(x=c_i, y=c_i, color='black', marker='x', s=100)
53 fig.show()
54
55 # Write HTML
56 fig.write_html('clusters.html')
```

Here's a clustering example.

We generate the output by compiling the source code.

The initial source code uses L1 norm to calculate the distance, we can change the math to use L2 norm which it's the more common norm.

H♥rtDown will update both the math output 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.

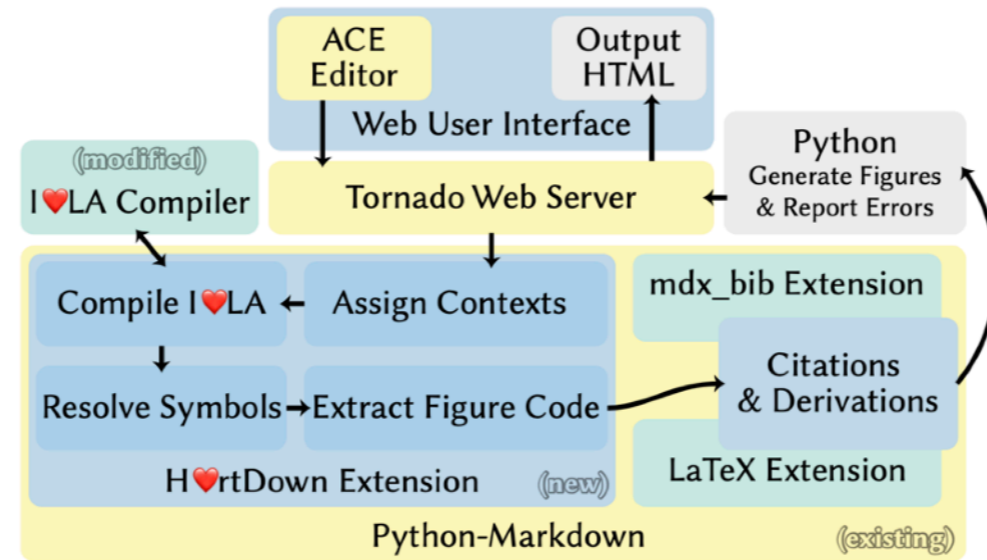
We'll show examples of that in our case studies

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

For the implementation of H♥rtDown

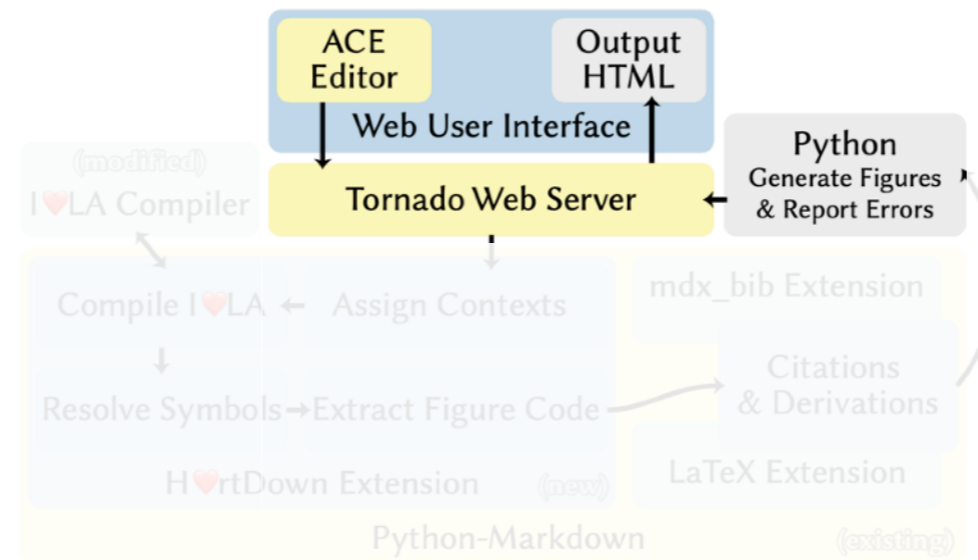
## Implementation



The figure shows the overall structure

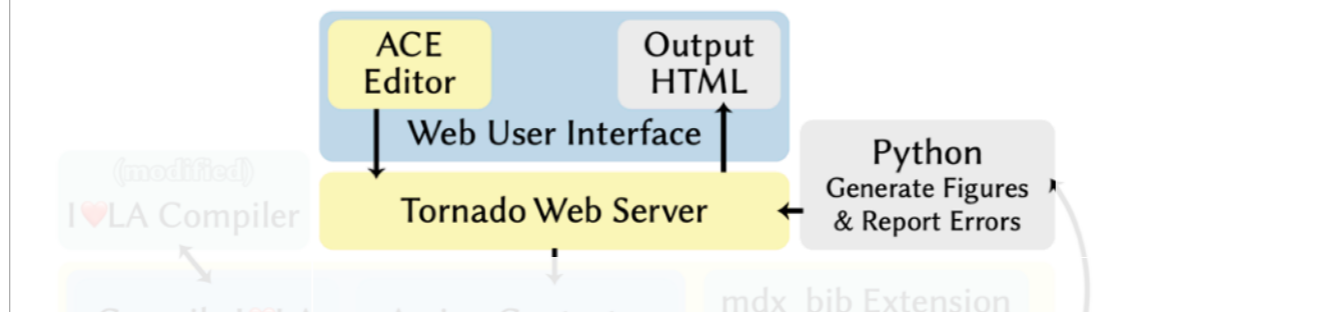
# (of the implementation of H♥rtDown)

## Implementation



The web-based authoring GUI displays the editable input source and output paper reading environment side by side, leveraging an embeddable code editor

## Implementation

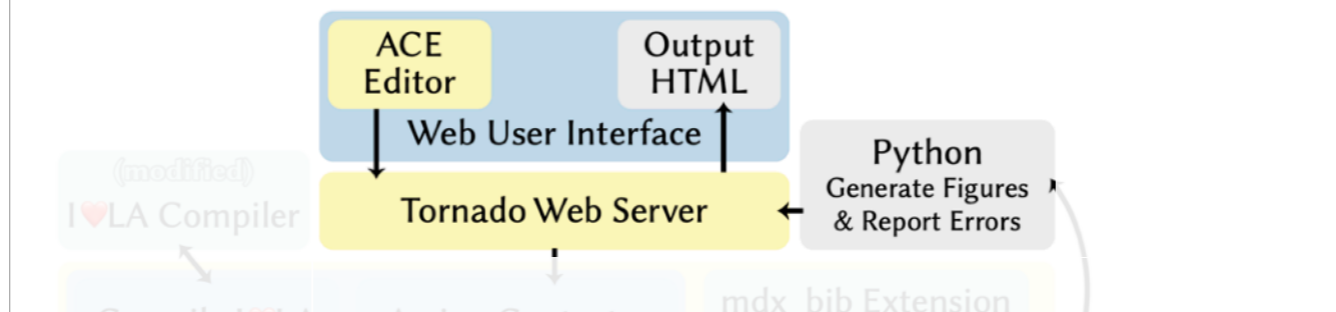


The web-based authoring GUI displays the editable input source and output paper reading environment side by side, leveraging an embeddable code editor



## Implementation: Web-based Editor

- Side-by-side source editor and output reading environment
- Communicates via POST requests with a Python-based Tornado server
- Caches I♥LA code and only re-compiles when necessary



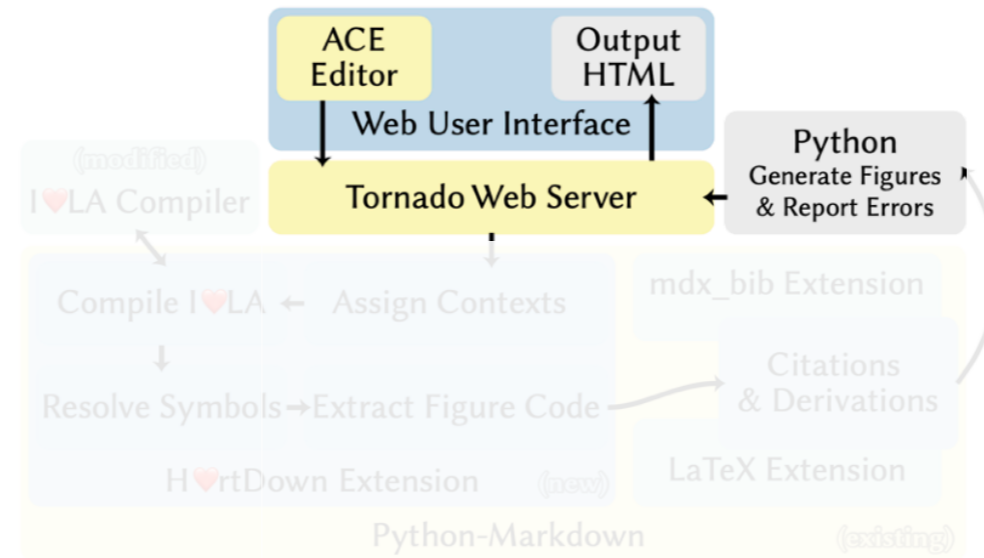
The web-based authoring GUI displays the editable input source and output paper reading environment side by side, leveraging an embeddable code editor

The GUI communicates via POST requests with a server running the Python-based Tornado web framework and asynchronous networking library to run the H♥rtDown document processing

To speed compilation, H♥rtDown caches I♥LA code and only re-compiles it when the I♥LA code has changed (determined via string comparison)

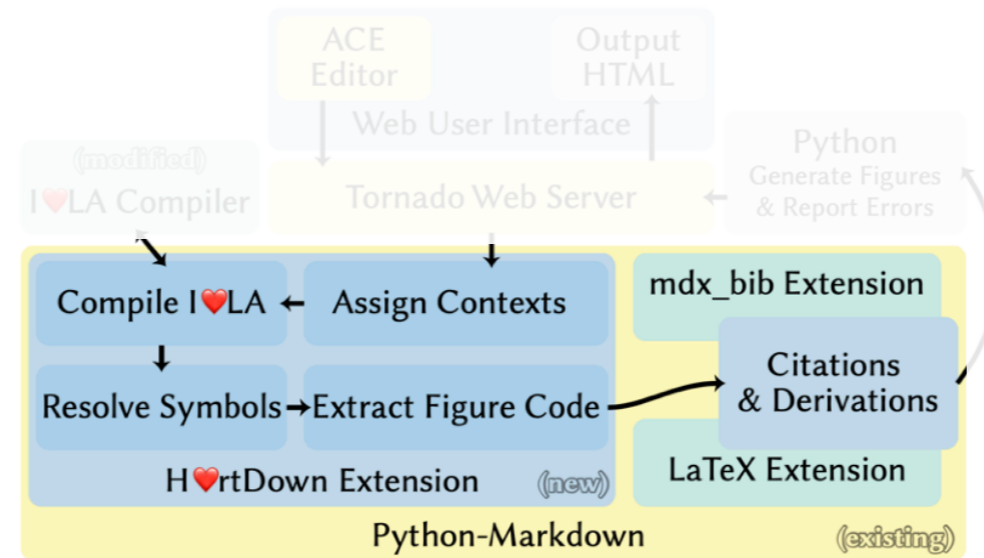
When a figure's code is changed from the viewer, H♥rtDown only runs that Python code block

## Implementation



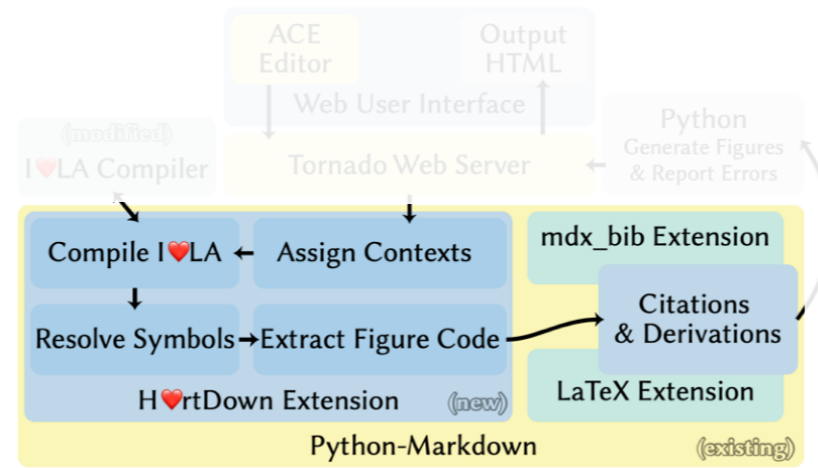
We implemented H♥rtDown as an extension to Python Markdown,

## Implementation



We implemented H♥rtDown as an extension to Python Markdown,

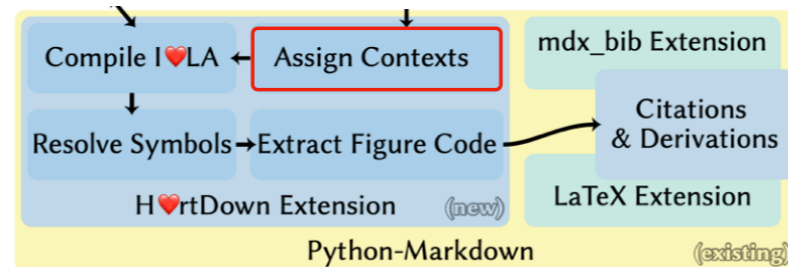
## Implementation



We implemented H♥rtDown as an extension to Python Markdown,

## Implementation: Python Markdown Extension

```
4 filters
5
6 # Image Filtering
7
8 To filter an image, we take a weighted average of pixels in a window around each pixel. We will use a quadratic
9 Gaussian approximation to create a <span class="def">filter function  $f(x)$ </span> that applies to <span
10 class="def">values  $Sx$  in  $S[-1,1]$ </span>;
11
12 <math display="block">f(x) = \begin{cases} 1-3x^2 & \text{if } |x| < 1 \\ 1.5x^2 - 3|x| + 1.5 & \text{if } |x| < 2 \\ 0 & \text{otherwise where } x \in \mathbb{R} \end{cases}
13
14 ...
15
16 <figure>
17 <code>
18 from lib import *
19 import numpy as np
20 import skimage.io
21 import scipy.signal
22
```



We implemented H♥rtDown as an extension to Python Markdown. (Given an input Markdown source file, H♥rtDown first parses all the context declarations<sup>(\*)</sup>. These are used to infer each symbol's scope, so that the prose annotations in the MathJax LaTeX output, the span tags around symbol definitions in prose, and I♥LA can all omit specifying the context. Then H♥rtDown concatenates all I♥LA code from the same context and compiles it<sup>(\*)</sup> into both executable code for the libraries and MathJax. H♥rtDown then<sup>(\*)</sup> searches for each symbol in the user's LaTeX derivations and applies the LaTeX command used by our MathJax extension so that derivation symbols can be queried uniformly.

<sup>(\*)</sup>If the source contains figure code, H♥rtDown extension will extract them in the end.)<sup>(\*)</sup>

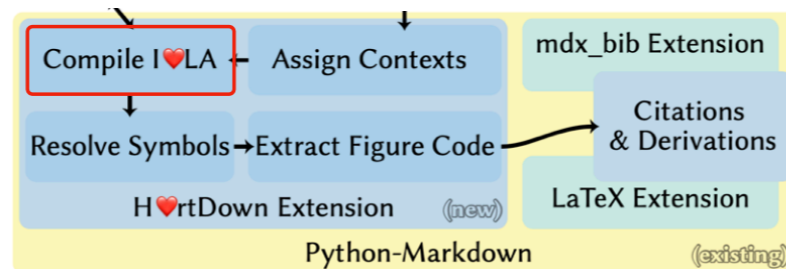
We added support for inline mathematical expressions by enclosing them in \$ (a single-line change)

We modified an bibliography extension to resemble the SIGGRAPH bibliography style.

We also used Pandoc-style YAML headers to specify metadata

## Implementation: Python Markdown Extension

```
4  # filters
5
6  # Image Filtering
7
8  To filter an image, we take a weighted average of pixels in a window around each pixel. We will use a quadratic
9  Gaussian approximation to create a <span class="def">filter function  $f(x)$ </span> that applies to <span
10  class="def">values  $x$  in  $[-1,1]$ </span>;
11
12  <math display="block">f(x) = \begin{cases} 1-3x^2 & \text{if } |x| < 1 \\ 1.5x^2 - 3|x| + 1.5 & \text{if } |x| < 1 \\ 0 & \text{otherwise where } x \in \mathbb{R} \end{cases}</code>
13  ...
14
15
16  <figure>
17  .. python
18  from lib import *
19  import numpy as np
20  import skimage.io
21  import scipy.signal
22
```



We implemented H♥rtDown as an extension to Python Markdown. (Given an input Markdown source file, H♥rtDown first parses all the context declarations<sup>(\*)</sup>. These are used to infer each symbol's scope, so that the prose annotations in the MathJax LaTeX output, the span tags around symbol definitions in prose, and I♥LA can all omit specifying the context. Then H♥rtDown concatenates all I♥LA code from the same context and compiles it<sup>(\*)</sup> into both executable code for the libraries and MathJax. H♥rtDown then<sup>(\*)</sup> searches for each symbol in the user's LaTeX derivations and applies the LaTeX command used by our MathJax extension so that derivation symbols can be queried uniformly.

<sup>(\*)</sup>If the source contains figure code, H♥rtDown extension will extract them in the end.)<sup>(\*)</sup>

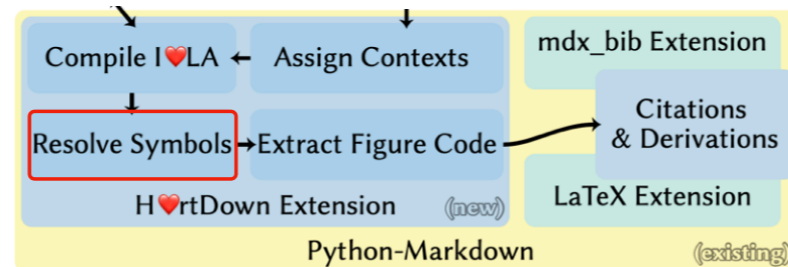
We added support for inline mathematical expressions by enclosing them in  $(a single-line change)$

We modified an bibliography extension to resemble the SIGGRAPH bibliography style.

We also used Pandoc-style YAML headers to specify metadata

## Implementation: Python Markdown Extension

```
4  # filters
5
6  # Image Filtering
7
8  To filter an image, we take a weighted average of pixels in a window around each pixel. We will use a quadratic
9  Gaussian approximation to create a <span class="def">filter function  $f(x)$ </span> that applies to <span
10  class="def">values  $x$  in  $[-1,1]$ </span>;
11
12  ..:::heartla
13  f(x) = { 1-3x^2      if |x| < 1
14         1.5x^2 - 3|x| + 1.5 if |x| < 1
15         0 otherwise where  $x \in \mathbb{R}$ 
16  ...
17
18  <figure>
19  .. python
20  from lib import *
21  import numpy as np
22  import skimage.io
23  import scipy.signal
```



We implemented H♥rtDown as an extension to Python Markdown. (Given an input Markdown source file, H♥rtDown first parses all the context declarations<sup>(\*)</sup>. These are used to infer each symbol's scope, so that the prose annotations in the MathJax LaTeX output, the span tags around symbol definitions in prose, and I♥LA can all omit specifying the context. Then H♥rtDown concatenates all I♥LA code from the same context and compiles it<sup>(\*)</sup> into both executable code for the libraries and MathJax. H♥rtDown then<sup>(\*)</sup> searches for each symbol in the user's LaTeX derivations and applies the LaTeX command used by our MathJax extension so that derivation symbols can be queried uniformly.

<sup>(\*)</sup>If the source contains figure code, H♥rtDown extension will extract them in the end.)<sup>(\*)</sup>

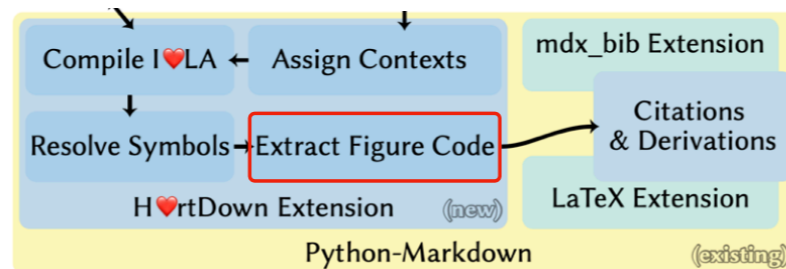
We added support for inline mathematical expressions by enclosing them in \$ (a single-line change)

We modified an bibliography extension to resemble the SIGGRAPH bibliography style.

We also used Pandoc-style YAML headers to specify metadata

## Implementation: Python Markdown Extension

```
4  * filters
5
6  # Image Filtering
7
8  To filter an image, we take a weighted average of pixels in a window around each pixel. We will use a quadratic
9  Gaussian approximation to create a <span class="def">filter function  $f(x)$ </span> that applies to <span
10  class="def">values  $x$  in  $[-1,1]$ </span>;
11
12  .. iheartla
13  f(x) = { 1-3x^2      if |x| < 1
14         1.5x^2 - 3|x| + 1.5 if |x| < 1
15         0 otherwise where  $x \in \mathbb{R}$ 
16  ..
17
18  .. figures
19  .. python
20  from lib import *
21  import numpy as np
22  import skimage.io
23  import scipy.signal
```



We implemented H♥rtDown as an extension to Python Markdown. (Given an input Markdown source file, H♥rtDown first parses all the context declarations<sup>(\*)</sup>. These are used to infer each symbol's scope, so that the prose annotations in the MathJax LaTeX output, the span tags around symbol definitions in prose, and I♥LA can all omit specifying the context. Then H♥rtDown concatenates all I♥LA code from the same context and compiles it<sup>(\*)</sup> into both executable code for the libraries and MathJax. H♥rtDown then<sup>(\*)</sup> searches for each symbol in the user's LaTeX derivations and applies the LaTeX command used by our MathJax extension so that derivation symbols can be queried uniformly.

<sup>(\*)</sup>If the source contains figure code, H♥rtDown extension will extract them in the end.)<sup>(\*)</sup>

We added support for inline mathematical expressions by enclosing them in  $(a single-line change)$

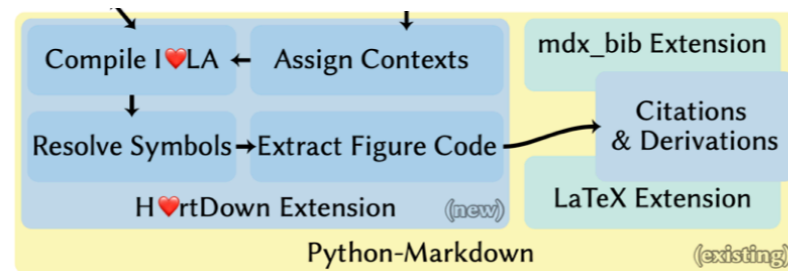
We modified an bibliography extension to resemble the SIGGRAPH bibliography style.

We also used Pandoc-style YAML headers to specify metadata



## Implementation: Python Markdown Extension

- Inline mathematical expressions enclosed by \$
- SIGGRAPH bibliography style
- Pandoc-style YAML header for metadata



We implemented H♥rtDown as an extension to Python Markdown. (Given an input Markdown source file, H♥rtDown first parses all the context declarations<sup>(\*)</sup>. These are used to infer each symbol's scope, so that the prose annotations in the MathJax LaTeX output, the span tags around symbol definitions in prose, and I♥LA can all omit specifying the context. Then H♥rtDown concatenates all I♥LA code from the same context and compiles it<sup>(\*)</sup> into both executable code for the libraries and MathJax. H♥rtDown then<sup>(\*)</sup> searches for each symbol in the user's LaTeX derivations and applies the LaTeX command used by our MathJax extension so that derivation symbols can be queried uniformly.

<sup>(\*)</sup>If the source contains figure code, H♥rtDown extension will extract them in the end.)<sup>(\*)</sup>

We added support for inline mathematical expressions by enclosing them in \$ (a single-line change)

We modified an bibliography extension to resemble the SIGGRAPH bibliography style.

We also used Pandoc-style YAML headers to specify metadata

## Implementation: Reading Environment

- HTML for document reflow
- SVG arrows for math augmentation
- JSON output by the H♥rtDown extension to visualize symbol relationships
- MathJax extensions store information for symbols and equations

The paper reading environment uses html for document-reflow

SVG arrows for math augmentations

JSON output by the H♥rtDown extension to visualize symbol relationships and enhance the paper reading experience.

We leverage MathJax extensions to store information for symbols and equations in the HTML tags generated by MathJax when displaying LaTeX math. This allowed us to access the symbols in a structured way from JavaScript, which implements the dynamic, interactive aspects of our reading environment, and styles the symbols using CSS.

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

We converted a variety of SIGGRAPH papers and paper sections to H♥rtDown.

## H♥rtDown Case Studies

### Entire papers

- An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR
- A Luminance-aware Model of Judder Perception (\*)
- A Perceptual Model for Eccentricity-dependent Spatio-temporal Flicker Fusion and its Applications to Foveated Graphics
- A Symmetric Objective Function for ICP (\*)
- Regularized Kelvinlets Sculpting Brushes based on Fundamental Solutions of Elasticity (\*)

### Paper sections

- Stable Neo-Hookean Flesh Simulation (\*)
- A perceptual model of motion quality for rendering with adaptive refresh-rate and resolution
- Anisotropic Elasticity for Inversion-Safety and Element Rehabilitation (\*)
- On Elastic Geodesic Grids and Their Planar to Spatial Deployment
- Nautilus-Recovering Regional Symmetry Transformations for Image Editing
- Computational Design of Transforming Pop-up Books
- Unmixing-Based Soft Color Segmentation for Image Manipulation (\*)
- Generic Objective Vortices for Flow Visualization (\*)
- SIERE: a hybrid semi-implicit exponential integrator for efficiently simulating stiff deformable objects

(\*) compares code to an existing implementation

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

## H♥rtDown Case Studies

Case	Source	Type	#Lines (#B/#I)
1	Schroers et al. [2018]	Paper	18 (13/4)
2	Rusinkiewicz [2019]	Paper(*)	11 (11/5)
3	Krajancich et al. [2021]	Paper	11 (9/3)
4	Chapiro et al. [2019]	Paper(*)	8 (8/0)
5	De Goes and James [2017]	Paper(*)	7 (7/0)
6	Chen et al. [2020]	Section	16 (6/0)
7	Kim et al. [2019]	Section(*)	12 (9/3)
8	Pillwein et al. [2020]	Section	5 (5/0)
9	Denes et al. [2020]	Section	5 (4/0)
10	Smith et al. [2018]	Section(*)	4 (4/2)
11	Xiao et al. [2018]	Section	4 (4/0)
12	Lukáč et al. [2017]	Section	4 (3/0)
13	Günther et al. [2017]	Section(*)	3 (3/2)
14	Aksoy et al. [2017]	Section(*)	1 (1/0)

Out of the 14 papers we reimplemented, we only found code online for 7 of them. Among these, 4 are from the paper authors and 3 are third-party implementations.

Papers with an asterisk(\*) had implementations available. The right-most column displays the total number of I♥LA equation lines(\*) and, in parentheses, the number of I♥LA blocks(\*) and the number of inline I♥LA formulas(\*).

We use the library generated by H♥rtDown to replace functions in the original code for each of these 7 examples, and use input examples to verify that the results match.

# H♥rtDown Case Studies

## A Symmetric Objective Function for ICP

Szymon Rusinkiewicz

SIGGRAPH North America 2019

- H♥rtDown source (entire paper)
- H♥rtDown-generated code libraries
- Existing implementation source code before modification and modified to call H♥rtDown-generated code

The image shows two side-by-side screenshots. The left one is the original PDF of the paper 'A Symmetric Objective Function for ICP' by Szymon Rusinkiewicz. It contains mathematical derivations for the ICP problem, including the Rodrigues rotation formula and the symmetric objective function. The right one is the H♥rtDown Paper Viewer, which displays the same content in a digital, interactive format with a table of contents on the right side.

Each example in our evaluation include 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.

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

We conducted an expert study to understand how active researchers can make use of H♥rtDown and the executable code it generates.

## Expert Study

- 3 CS PhD students
- Author an original document related to their computer graphics research
- Spent a total of 24, 7, and 6 hours, respectively, using H♥rtDown over a period of two weeks

We recruited 3 computer science PhD students

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: Expert 1

Let's say we have a hand made of five fingers and we want to know if it's intersecting a shape. Assume we can detect where the five fingertips intersect with the shape. And below we will analyse the distance of fingertips to a cuboid.

## Distance to Cuboid

Assume we have two lists of 3D points with same length, in which  $ps$  includes the start points of eight edges, and  $pe$  includes all end points of edges. The following formula  $f$  calculates the distance from one point to an edge in 3 conditions: closest to start or end point, or perpendicular to the edge.  $ps_i$  is the start point of edge  $i$ , and  $pe_i$  represents the 3D position of endpoint of edge  $i$ .  $V_j$  represents the 3D position of fingertip  $j$ .  $A$  is the matrix storing the distance between fingertips to edges  $j$ .  $f$  represents the 3D position of fingertip  $j$ .

$$f(ps_i, pe_i, V_j) = \begin{cases} \|V_j - ps_i\| & \text{if } (pe_i - ps_i) \cdot (V_j - ps_i) > 0 \\ \|V_j - pe_i\| & \text{if } (ps_i - pe_i) \cdot (V_j - pe_i) > 0 \\ \frac{(ps_i - pe_i) \cdot (V_j - ps_i)}{\|ps_i - pe_i\|} & \text{otherwise} \end{cases} \quad (1)$$

$$A_{i,j} = f(ps_i, pe_i, V_j)$$

This equation has 4 symbols:  
 $f \in \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3 \rightarrow \mathbb{R}$ :  $f$  represents the 3D position of fingertip  $j$   
 $ps_i \in \mathbb{R}^3$ :  $ps_i$  is the start point of edge  $i$   
 $pe_i \in \mathbb{R}^3$ :  $pe_i$  represents the 3D position of endpoint of edge  $i$   
 $V_j \in \mathbb{R}^3$ :  $V_j$  represents the 3D position of fingertip  $j$

## Glossary of HandToShapeDistance

$A \in \mathbb{R}^{dim_p \times dim_f}$ :  $A$  is the matrix storing the distance between fingertips to edges  $j$   
 $V \in$  sequence of  $\mathbb{R}^3$ : lists of position of five fingertips  
 $V_j \in \mathbb{R}^3$ :  $V_j$  represents the 3D position of fingertip  $j$   
 $f \in \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3 \rightarrow \mathbb{R}$ :  $f$  represents the 3D position of fingertip  $j$   
 $pe \in$  sequence of  $\mathbb{R}^3$ : lists of position of end points of line segments  
 $pe_i \in \mathbb{R}^3$ :  $pe_i$  represents the 3D position of endpoint of edge  $i$   
 $ps \in$  sequence of  $\mathbb{R}^3$ : lists of position of start points of line segments  
 $ps_i \in \mathbb{R}^3$ :  $ps_i$  is the start point of edge  $i$

The first expert wrote a function with conditionals to calculate the distance from hand to shape in 3D.

## Expert Study: Expert 2

$$E_{\text{perpendicular}}(V, a, b, p, q) = \left( \left( \frac{V_{a,b} - V_{b,a}}{\|V_{a,b} - V_{b,a}\|} \right) \cdot \left( \frac{V_{p,q} - V_{q,p}}{\|V_{p,q} - V_{q,p}\|} \right) \right)^2 \quad (3)$$

where  $E_{\text{perpendicular}}$  takes in points  $V$  and the index  $a, b, p, q$  returns perpendicular energy.

Given a set of these functions and corresponding sets of positions given as indices into an array  $V_0 \in \mathbb{R}^{n \times 3}$ , we can find new positions via optimization:

$$t = \min_{V \in \mathbb{R}^{n \times 3}} E_{\text{len}}(V_0, L) + E_{\text{par}}(V_0, P) + E_{\text{per}}(V_0, Q) \quad (4)$$

This equation has 8 symbols:

- $t \in \mathbb{R}$ :  $t$  is energy equals to the sum of  $E_{\text{len}}$ ,  $E_{\text{par}}$  and  $E_{\text{per}}$ .
- $L \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.
- $P \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.
- $E_{\text{len}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{len}}$  takes  $V_0, L$  and sums all the length energy value.
- $E_{\text{par}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{par}}$  takes  $V_0, P$  and sums all the parallel energy value.
- $V_0 \in \mathbb{R}^{n \times 3}$ ,  $V_0$  is the subset of points to be optimized.
- $Q \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.
- $E_{\text{per}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{per}}$  takes  $V_0, Q$  and sums all the perpendicular energy value.

where  $V_0$  is the subset of points to be optimized,  $V_0$  is the initial value of  $V_0$ ,  $L, P, Q$  are length, parallel and perpendicular indices, and  $t$  is energy equals to the sum of  $E_{\text{len}}$ ,  $E_{\text{par}}$  and  $E_{\text{per}}$ .

Since some vertices are fixed, function  $f$  is used to get the position of all vertices. In order to conveniently get the position for each energy, we can use several helper functions to index the full position matrix:

$$E_{\text{len}}(V_0, L) = \sum_i E_{\text{length}}(f(V_0), L_{i,1}, L_{i,2}, L_{i,3}, L_{i,4}) \quad (5)$$

where  $f$  maps  $V$  to  $V_0$ , and  $E_{\text{len}}$  takes  $V_0, L$  and sums all the length energy value.

$$E_{\text{par}}(V_0, P) = \sum_i E_{\text{parallel}}(f(V_0), P_{i,1}, P_{i,2}, P_{i,3}, P_{i,4}) \quad (6)$$

where  $E_{\text{par}}$  takes  $V_0, P$  and sums all the parallel energy value.

$$E_{\text{per}}(V_0, Q) = \sum_i E_{\text{perpendicular}}(f(V_0), Q_{i,1}, Q_{i,2}, Q_{i,3}, Q_{i,4}) \quad (7)$$

Glossary of ScaffoldSketch

$E_{\text{length}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z} \rightarrow \mathbb{R}$ :  $E_{\text{length}}$  takes in points  $V$  and the index  $a, b, p, q$  returns length energy.

$E_{\text{len}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{len}}$  takes  $V_0, L$  and sums all the length energy value.

$E_{\text{parallel}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z} \rightarrow \mathbb{R}$ :  $E_{\text{parallel}}$  takes in points  $V$  and the index  $a, b, p, q$  returns parallel energy.

$E_{\text{par}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{par}}$  takes  $V_0, P$  and sums all the parallel energy value.

$E_{\text{perpendicular}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z}, \mathbb{Z} \rightarrow \mathbb{R}$ :  $E_{\text{perpendicular}}$  takes in points  $V$  and the index  $a, b, p, q$  returns perpendicular energy.

$E_{\text{per}} \in \mathbb{R}^{n \times 3}, \mathbb{Z}^{n \times 4} \rightarrow \mathbb{R}$ :  $E_{\text{per}}$  takes  $V_0, Q$  and sums all the perpendicular energy value.

$L \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.

$P \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.

$Q \in \mathbb{Z}^{n \times 4}$ ,  $L, P, Q$  are length, parallel and perpendicular indices.

$V \in \mathbb{R}^{n \times 3}$ ,  $V$  is the points.

$V_0 \in \mathbb{R}^{n \times 3}$ ,  $V_0$  is the subset of points to be optimized.

$V_0 \in \mathbb{R}^{n \times 3}$ ,  $V_0$  is the initial value of  $V_0$ .

$a \in \mathbb{Z}$ ,  $a, b, p, q$  are the indices.

$b \in \mathbb{Z}$ ,  $a, b, p, q$  are the indices.

$f \in \mathbb{R}^{n \times 3} \rightarrow \mathbb{R}^{n \times 3}$ ,  $f$  maps  $V$  to  $V_0$ .

$m \in \mathbb{Z}$ ,  $m$  is the number of points.

$p \in \mathbb{Z}$ ,  $a, b, p, q$  are the indices.

$q \in \mathbb{Z}$ ,  $a, b, p, q$  are the indices.

$t \in \mathbb{R}$ :  $t$  is energy equals to the sum of  $E_{\text{len}}$ ,  $E_{\text{par}}$  and  $E_{\text{per}}$ .

The second expert solved an optimization problem for 3D snapping

# The objective function is divided into three sub-functions.

## Expert Study: Expert 3

**Bending Energy**

Define bending energy  $E_b$

$$E_b = \frac{1}{2} \sum_i \frac{1}{l_i} \left( B_{1,1,1} (\kappa_{2i} - \bar{\kappa}_{2i})^2 + B_{1,2,2} (\kappa_{3i} - \bar{\kappa}_{3i})^2 \right) \quad (2)$$

This equation has 7 symbols:

- $E_b \in \mathbb{R}$ : bending energy  $E_b$
- $\bar{\kappa}_2 \in \mathbb{R}^{dim}$ ,  $\kappa_1$  and  $\kappa_2$  being rest curvature vectors
- $B \in \text{sequence of } \mathbb{R}^{3 \times 3}$ ;  $B$  is the bending stiffness matrix
- $\kappa_1 \in \mathbb{R}^{dim}$ ,  $\kappa_1$  and  $\kappa_2$  being curvature vectors
- $\kappa_2 \in \mathbb{R}^{dim}$ ,  $\kappa_1$  and  $\kappa_2$  being curvature vectors
- $l \in \text{sequence of } \mathbb{R}$ ;  $l$  is the voronoi length
- $\bar{\kappa}_i \in \mathbb{R}^{dim}$ ;  $\bar{\kappa}_1$  and  $\bar{\kappa}_2$  being rest curvature vectors

where

$$\begin{aligned} \kappa_{2i} &= \frac{\kappa b_i \cdot (\bar{d}_{2i} + d_{2i})}{2} \\ \kappa_{3i} &= \frac{\kappa b_i \cdot (\bar{d}_{1i} + d_{1i})}{2} \\ \bar{\kappa}_{2i} &= \frac{\bar{\kappa} b_i \cdot (\bar{d}_{2i} + d_{2i})}{2} \\ \bar{\kappa}_{3i} &= \frac{\bar{\kappa} b_i \cdot (\bar{d}_{1i} + d_{1i})}{2} \end{aligned} \quad (3)$$

$\kappa b$  being curvature binormal,  $\bar{\kappa} b$  being rest curvature binormal,  $\kappa_1$  and  $\kappa_2$  being curvature vectors,  $\bar{\kappa}_1$  and  $\bar{\kappa}_2$  being rest curvature vectors,  $B$  is the bending stiffness matrix, which  $B_i = \frac{EA}{4} \begin{bmatrix} a_i^2 & 0 \\ 0 & b_i^2 \end{bmatrix}$ ,  $l$  is the voronoi length, and  $E$  is the Young's modulus.

**Twisting Energy**

Define twisting energy  $E_t$

**Glossary of energy**

- $A \in \mathbb{R}^{dim}$ : the area of the node cross-section  $A_i$
- $B \in \text{sequence of } \mathbb{R}^{3 \times 3}$ ;  $B$  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
- $\bar{d}_i \in \text{sequence of } \mathbb{R}^3$ ; bar tilde  $d_1$  is bar  $d_1$  shifted left by one
- $\tilde{d}_i \in \text{sequence of } \mathbb{R}^3$ ; bar tilde  $d_2$  is bar  $d_2$  shifted left by one
- $d_1 \in \text{sequence of } \mathbb{R}^3$ ; rest orthogonal directors  $d_1$  and  $d_2$
- $\tilde{d}_2 \in \text{sequence of } \mathbb{R}^3$ ; rest orthogonal directors  $\tilde{d}_1$  and  $\tilde{d}_2$
- $\bar{e} \in \text{sequence of } \mathbb{R}^3$ ;  $\bar{e}$  being the rest edge length
- $l \in \text{sequence of } \mathbb{R}$ ;  $l$  is the voronoi length
- $\bar{m} \in \text{sequence of } \mathbb{R}$ ;  $\bar{m}$  is the rest twist
- $\bar{\kappa} b \in \text{sequence of } \mathbb{R}^3$ ;  $\bar{\kappa} b$  being rest curvature binormal
- $\bar{\kappa}_1 \in \mathbb{R}^{dim}$ ;  $\bar{\kappa}_1$  and  $\bar{\kappa}_2$  being rest curvature vectors
- $\bar{\kappa}_2 \in \mathbb{R}^{dim}$ ;  $\bar{\kappa}_1$  and  $\bar{\kappa}_2$  being rest curvature vectors
- $\tilde{d}_1 \in \text{sequence of } \mathbb{R}^3$ ; tilde  $d_1$  is  $d_1$  shifted left by one
- $\tilde{d}_2 \in \text{sequence of } \mathbb{R}^3$ ; tilde  $d_2$  is  $d_2$  shifted left by one
- $a \in \text{sequence of } \mathbb{R}$ ;  $a_i$  and  $b_i$  as the two axes of the ellipse at the  $i^{\text{th}}$  segment
- $b \in \text{sequence of } \mathbb{R}$ ;  $a_i$  and  $b_i$  as the two axes of the ellipse at the  $i^{\text{th}}$  segment
- $d_i \in \text{sequence of } \mathbb{R}^3$ ;  $d_1$  and  $d_2$  are orthogonal directors of every segment on the center-line
- $\tilde{d}_2 \in \text{sequence of } \mathbb{R}^3$ ;  $\tilde{d}_1$  and  $\tilde{d}_2$  are orthogonal directors of every segment on the center-line
- $e \in \text{sequence of } \mathbb{R}$ ;  $e$  being the edge length
- $k_s \in \mathbb{R}$ ;  $k_s$  is the stretching coefficient
- $m \in \text{sequence of } \mathbb{R}$ ;  $m$  is the twist
- $\beta \in \mathbb{R}^{dim}$ ;  $\beta$  is the twisting modulus
- $\kappa_1 \in \mathbb{R}^{dim}$ ;  $\kappa_1$  and  $\kappa_2$  being curvature vectors
- $\kappa_2 \in \mathbb{R}^{dim}$ ;  $\kappa_1$  and  $\kappa_2$  being curvature vectors
- $\kappa b \in \text{sequence of } \mathbb{R}^3$ ;  $\kappa b$  being curvature binormal

The third expert wrote a tutorial for discrete elastic rods.

# The energy functions were written separately, some relied on variables defined in the inline I ❤️ LA block.

# The tutorial written by this expert became a dynamically annotated document generating with canonical executability available in multiple programming environments.

This tutorial is now more readable due to math augmentations and can be used in addition to read since it self-generates code in any programming languages I ❤️ LA supports

## Expert Study: Observations and Conclusions

- Two participants appreciated that writing in H♥rtDown is similar to writing Markdown
- Two commented that writing math in I♥LA is harder than with Markdown/LaTeX
- One commented that the generated code compensates for the additional time spent writing the equations
- All participants liked the dynamic reader features

At the conclusion of the study, we sent participants a follow-up questionnaire.

Two participants appreciated that writing in H♥rtDown is similar to writing Markdown

Two commented that writing math in I♥ LA is harder than with Markdown/LaTeX

One commented that the generated code compensates for the additional time spent writing the equations.

All participants liked the dynamic reader features

## 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.”*

One user 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.

## Expert Study: Observations and Conclusions

- I♥LA language limitations (e.g. summation ranges)
- We fixed the cosmetic usability problems raised by the participants
- User feedback guides development efforts

Most of the limitations they encountered were due to I♥LA language limitations, such as limitations around summation ranges.

We fixed the cosmetic usability problems raised by the participants like stale information being shown when an error occurs.

We plan to address limitations in I♥LA functionality. Since our goal is for H♥rtDown to be adopted by researchers, user feedback will guide development efforts.

## Outline

- Related work
- Formative Study
- H♥rtDown Design
- H♥rtDown Implementation
- Case studies
- Expert study
- Conclusion

We have demonstrated that H♥rtDown is a useful compatible document processor.

## 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.  
1: foreach particle  $i$  do  
2:   compute  $\rho_{i,h}^t$  ▷ see Subsection 3.3.2  
3:   compute  $L_i$  ▷ see Eq. (15)  
4:   compute  $\mathbf{a}_i^{\text{ther},t}$  ▷ e.g., gravity and adhesion  
5:   compute  $\mathbf{a}_i^{\text{friction},t}$  ▷ using Eq. (24)  
6: SOLVE for  $\mathbf{a}_i^{\text{a}}$  ▷ see Subsection 3.2.1  
7: SOLVE for  $\mathbf{a}_i^{\text{G}}$  ▷ see Subsection 3.2.2  
8: foreach particle  $i$  do  
9:   integrate  $\mathbf{v}_i^{t+\Delta t} = \mathbf{v}_i^t + \Delta t (\mathbf{a}_i^{\text{ther},t} + \mathbf{a}_i^{\text{friction},t} + \mathbf{a}_i^{\text{a}} + \mathbf{a}_i^{\text{G}})$   
10: foreach particle  $i$  do  
11:   integrate  $\mathbf{F}_{E,i,t}$  ▷ see Subsection 3.3.1  
12: foreach particle  $i$  do  
13:   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 or semi-automatic conversion from LaTeX to H♥rtDown
- Incorporating a proof checker could allow verification of 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

In future, we also plan to,

Support Automatic or semi-automatic conversion from LaTeX to H♥rtDown

Incorporating a proof checker could allow the verification of 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.

## Acknowledgments

- Anonymous reviewers for their suggestions
- Seth Walker for helping design the reader environment
- Zoya Bylinskii for a discussion on related research projects
- Zhecheng Wang, Xue Yu and Jialin Huang for additional feedback
  
- Sponsors:
  - Canada Research Chairs Program
  - Sloan Foundation
  - Adobe Inc.

I'd like to thank

Anonymous reviewers for their suggestions

Seth Walker for helping design the reader environment

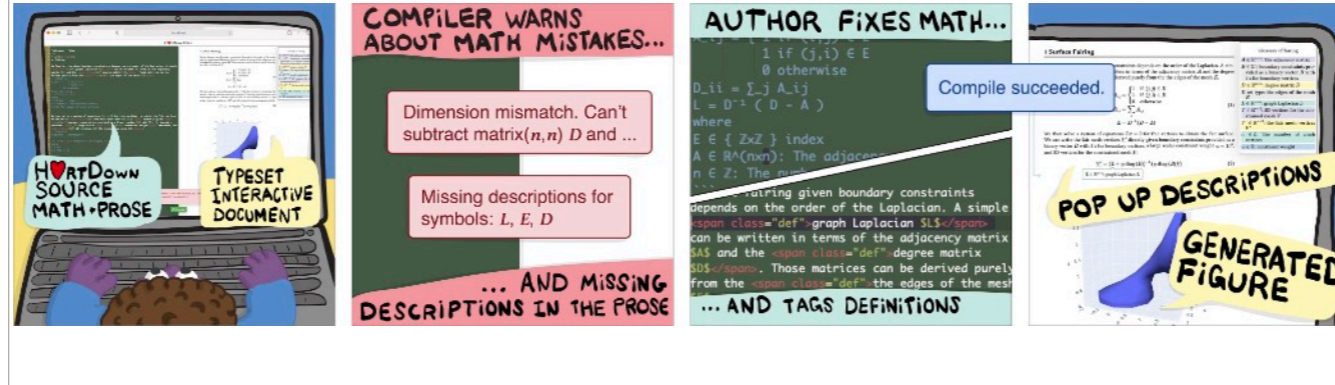
Zoya Bylinskii for a discussion on related research projects

Zhecheng Wang, Xue Yu and Jialin Huang for additional feedback

The research is supported by Canada Research Chairs Program, Sloan Foundation and Adobe

# H♥rtDown

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



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.