3D Model Surface Mesh Parametrization

Xuhui Wang

November 2021

Introduction

The computing process of the bijective mapping between two surfaces with the similar topology is referred to as mesh parameterization, which is widely used in a variety of applications including mesh morphing, mesh editing, mesh completion, remeshing, mesh compression, mesh fitting, shape analysis, detail mapping, detail-transfer and so on. In this project, I introduce and implement a sequence of methods that leverage vector field design on surfaces of 3D objects, create scalar field whose gradients align with the given vector fields as closely as possible, experiment with harmonic and least-square conformal parameterizations and visually demonstrate the working flow of parameterization editing with vector fields.

Tangent vector fields

(Note that, in this report, the wordings that describe the algorithms are all from the course material by Dr. Schneider and I will give the appropriate citation at the end of each corresponding paragraph. On the other hand, all resulted figures, methods implementation, statistics, analysis, explanations and conclusions are all from my personal work.) Firstly, we design smooth tangent vector fields on a surface; these will be "integrated" later to define a scalar field. A (piecewise constant) vector field on a triangle mesh is defined as an assignment of a single vector to each triangle such that each vector lies in the tangent plane containing the triangle. We will design fields to follow a set of alignment constraints provided by the user: the user specifies the field vectors at a subset of the mesh triangles (the constraints) and those constraints are interpolated smoothly throughout the surface to define a field. [1]

As the first step we need to define an assignment data that contains a selection of faces and vector constraints at such faces. Since each field vector u_f lies in the plane of its respective triangle f, it can be decomposed into the triangle's local basis and represented with two real coefficients: $u_f = (x_f, y_f)$ in R^2. Alternatively, we can identify the triangle plane with the complex plane, expressing the vector as a single complex number $u_f = x_f + iy_f$ in C. [1]

The interpolation produces a smooth vector field from the constraints by trying to make each vector as similar as possible to the adjacent triangles' vectors. However, since vectors u_f, u_g at adjacent triangles f, g are expressed with respect to the triangles' local bases, and the triangles generally have different bases, their complex expressions cannot be compared directly as shown in Figure 1. So we must first express the vectors with respect to a common basis. [1]

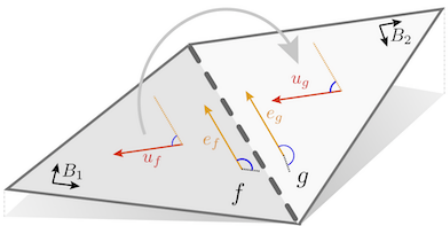


Figure 1: triangles' local basis [1]

A simple way to choose a common basis is to use the shared (directed) edge vector as the new real axis for both triangles. This implicitly also chooses the 90°-counterclockwise-rotated edge vector as the imaginary axis, forming a complete basis. In this new common basis, the two triangles' vectors can be written as $\tilde{u}_f = u_f e_f^{-1} = u_f \bar{e_f}$ and $u_g = u_g \bar{e_g}$ where e_f, e_g are the shared edge vector expressed in each local basis. Here the overline denotes the complex conjugate of a complex number, and we assume e_f, e_g are normalized so that the conjugate is actually the inverse. The difference ("non-smoothness") of the two vectors across the edge (f, g) can now be computed as $E_{fg}(u_f, u_g) = |u_f \bar{e_f} - u_g \bar{e_g}^2|$. By recalling that $||z||^2 = z$ and z's conjugate is a complex number z's squared magnitude, this is a quadratic form on the complex variables u_f, u_g and can be manipulated into the form $E_{fg}(u_f, u_g) = [\bar{u_f} \quad \bar{u_g}] Q_{fg} [u_f \quad u_g]^T$ for a particular complex matrix Q_{fg} [1]. The full field's smoothness can then be written as the sum of all these per-edge energies: $\sum_{edge(f,g)} E_{fg}(u_f, u_g)$. This yields a sparse quadratic form $u^* Q u$ where the complex column vector u encodes each each per-face vector. [1]

The row vector u^* is u's adjoint (conjugate transpose), and Q is the appropriate combination of the matrices Q_{fg}. Thus, finding the smoothest field under the prescribed constraints is equivalent solving $min\{u^* Q u \quad u|_{cf}\} = c$ where cf are the constrained face indices and c are the prescribed vectors at those faces. We can differentiate the smoothness energy to find its minimum, thus obtaining a (complex) linear system $Qu = 0 \quad u|_{cf} = c$. This system's solution describes the smoothly interpolated vector field. [1]

Through the construction of the complex matrix Q, solving the system under the prescribed constraints, we display the constraints and the interpolated field in Figure 2. We also print the interpolated field for the given mesh and the input constraints in the given constraints file in Figure 3.

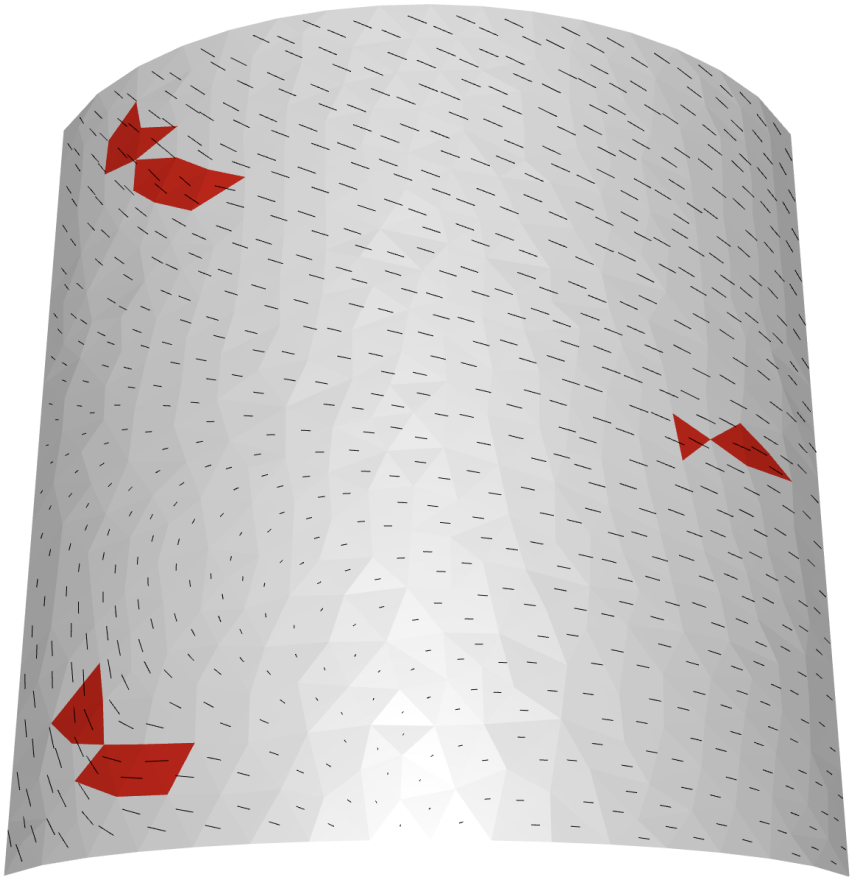


Figure 2: the constraints and the interpolated field

```
the interpolated field is
[[ 0.23591958 -0.2272005   0.39047189]
 [-0.42277571  0.13051135 -0.14899519]
 [ 0.68709978 -0.24579441 -0.29458214]
 ...
 [ 0.44584006 -0.09987585  0.05341989]
 [ 0.34755951 -0.66810857  0.60711695]
 [ 0.4087346  -0.67078033  0.56895471]]
the shape is (756, 3)
the constraints is
[525  93 527 650 652 649 312 562 558 679 337 362 678 276  63 277 466  10
 505 686]
[[-0.86136    0.050168  -0.505512 ]
 [-0.822122   0.13914   -0.552047 ]
 [-0.761541   0.286908  -0.581154 ]
 [-0.527336   0.718255  -0.453902 ]
 [-0.426421   0.812406  -0.397695 ]
 [-0.173632   0.94519   -0.276528 ]
 [-0.12095    0.987557  -0.100515 ]
 [ 0.528064  -0.685298   0.501512 ]
 [ 0.485416  -0.738721   0.467614 ]
 [ 0.499676  -0.705184   0.50303  ]
 [ 0.566063  -0.704697   0.427757 ]
 [ 0.540817  -0.730976   0.416162 ]
 [ 0.562468  -0.68217    0.467198 ]
 [ 0.604154  -0.712722   0.356406 ]
 [ 0.631683  -0.639429   0.438301 ]
 [ 0.818211  -0.414128   0.398783 ]
 [ 0.817361  -0.21155   -0.53588  ]
 [ 0.772265  -0.0814812 -0.630054 ]
 [ 0.708944  -0.0922904 -0.699201 ]
 [ 0.659804  -0.0752781 -0.747658 ]]
the shape is (20, 3)
```

Figure 3: the interpolated field for the given mesh and the input constraints in the given constraints

Reconstructing a scalar field from a vector field

We need to find a scalar function $S(x)$ defined over the surface whose gradient fits a given vector field as closely as possible. The scalar field is defined by values on the mesh vertices that are linearly interpolated over each triangle's interior: for given vertex values s_i, the function $S(x)$ inside a triangle t is computed as $S(t) = \sum_{\text{vertex } i \in t}^{3} s_i \phi_i^t(x)$ where $\phi_i^t(x)$ are the linear "hat" functions associated with each triangle vertex (i.e. $\phi_i^t(x)$ is linear and takes the value 1 at vertex i and 0 at all other vertices). Then the scalar function's (vector-valued) gradient is $g_t = \nabla S_t = \sum_{\text{vertex } i \in t}^{3} s_i \nabla \phi_i^t$. [1]

Since the "hat" functions are piecewise linear, their gradients are constant within each triangle, and so is g_t (the full scalar function's gradient). Specifically, g_t is a linear combination of the constant hat function gradients with the (unknown) values s_i as coefficients, meaning that we can write an expression of the form $g = Gs$, where s is a $\#V \times 1$ column vector holding each s_i, g is a column vector of size $3\#F \times 1$ consisting of the vectors g_t "flattened" and vertically stacked, and G is the so-called "gradient matrix" of size $3\#F \times \#V$. [1]

Since there is no guarantee that our interpolated face-based field is actually the gradient of some function, we cannot attempt to integrate it directly. Instead, we will try to find $S(x)$ by asking its gradient to approximate the vector field u in the least-squares sense: $min \sum_{\text{face } t} A_t ||g_t - u_t||^2$ where A_t is triangle t's area, g_t is the (unknown) function's gradient on the triangle, and u_t is the triangle's vector assigned by the guiding vector field. Using the linear relationship $g = Gs$, we can write this least-squares error as a (real) quadratic form: $\frac{1}{2} s^T K s + s^T b + c$ and minimize it by solving a linear system for the unknown s [1]. We determine the matrix K and vector b in the above minimization (by expanding the least-squares error expression).[1]

Firstly we write the sum of error term in matrix form:

$$min\{(Gs - u)^T A (Gs - u)\}$$

where A is a sparse diagonal matrix of which the entries are $A_1, ..., A_n, A_1, ..., A_n, A_1, ..., A_n$. Specifically, A_i is the area of the triangle i. Expanding the sum of error term, we get

$$min\{s^T G^T A G s - 2 s^T G^T A u + u^T A u\}.$$

Here $K = G^T AG$ and $b = -2G^T Au$. Differentiating the sum of error term and setting it to 0, we get the equation

$$G^T AGs = G^T Au.$$

Then we could solve the equation for s.

Here, as shown in Figure 4, we display the plots of scalar function on the surface using a color map, overlay its gradient vectors and deviation between the input vector field and the solution scalar function's gradient (the "Poisson reconstruction error"), where the white line denotes the gradient and black line denotes the scalar function.

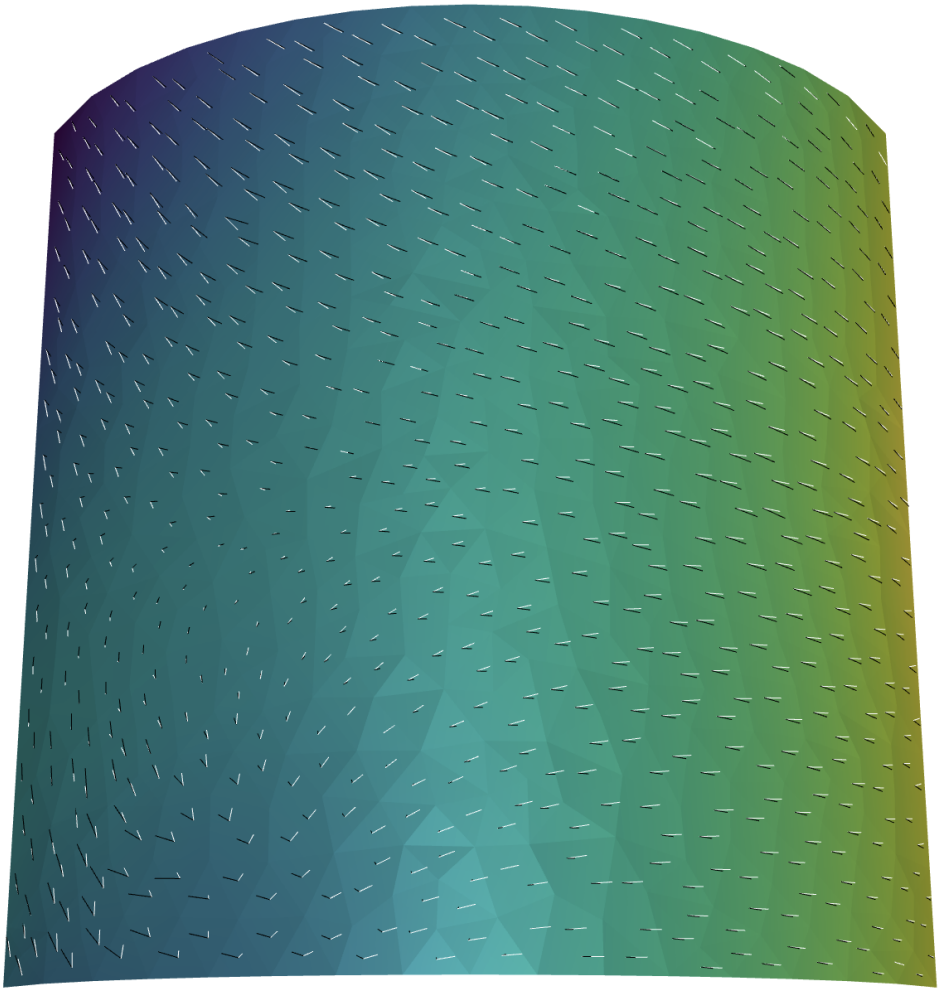


Figure 4: scalar function on the surface using a color map, overlay its gradient vectors and the "Poisson reconstruction error"

Harmonic and LSCM parameterizations

For this section, we experiment with flattening a mesh with a boundary onto the plane using two parameterization methods: Harmonic and Least Squares Conformal (LSCM) parameterization. In both cases, two scalar fields, U and V, are computed over the mesh. The per-vertex (u, v) scalars defining these coordinate functions determine the vertices' flattened positions in the plane (the flattening is linearly interpolated within each triangle).

Harmonic parametrization is a single patch, fixed boundary parametrization algorithm that computes the 2D coordinates of the flattened mesh as two harmonic functions. The algorithm is divided in 3 steps: Detect of the boundary vertices. Map the boundary vertices to a circle. Compute two harmonic functions (one for u and one for the v coordinate). The harmonic functions use the fixed vertices on the circle as boundary constraints. [1]

For the harmonic parametrization example, we first map the mesh boundary to a unit circle in the UV plane centered at the origin. The boundary U and V coordinates are then "harmonically interpolated" into the interior by solving the Laplace equation with Dirichlet boundary conditions (setting the Laplacian of U equal to zero at each interior vertex, then doing the same for V). This involves two separate linear system solves (each with the same system matrix). [1]

Instead, least squares conformal maps parametrization minimizes the conformal (angular) distortion of the parametrization. Differently from harmonic parametrization, it does not need to have a fixed boundary. Specifically, the energy LSCM minimizes is

$$E(u, v) = \int_X \frac{1}{2} |\nabla u^{\perp} - \nabla v|^2 \mathrm{d}A.$$

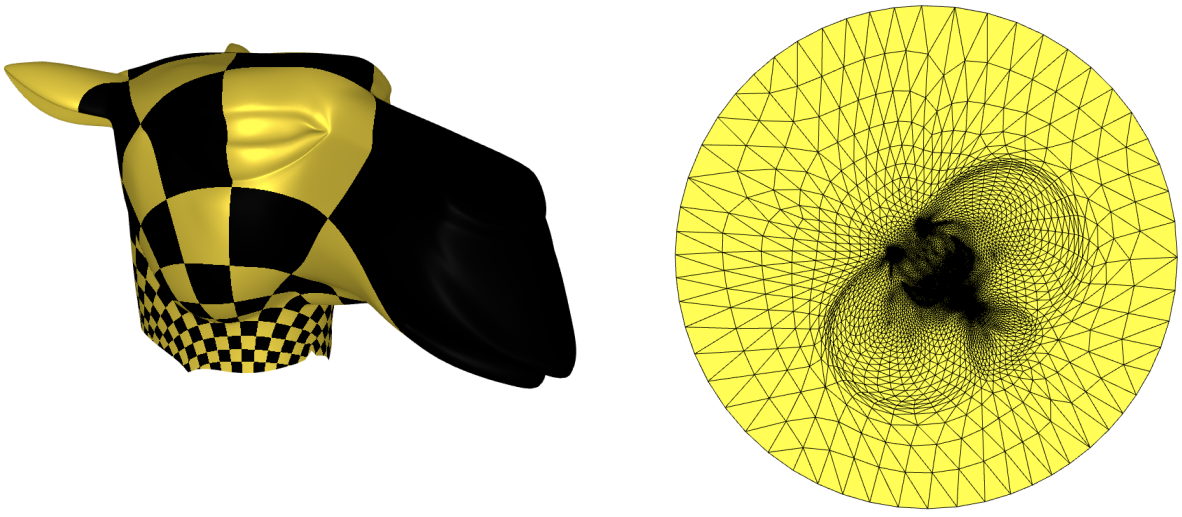
We could rewrite this in a matrix from as it follows:

$$E(u, v) = \frac{1}{2}[u, v]^t (L_c - 2A)[u, v]$$

where L_c is the cotangent Laplacian matrix and A is a matrix such that $[u, v]^t A[u, v]$ is equal to the vector area of the mesh. [1]

In LSCM, the boundary is free, with the exception of two vertices that must be fixed at two different locations in the UV-plane (to pin down a global position, rotation, and scaling factor). These vertices can be chosen arbitrarily. The process is again a linear system solve, but in this case the U and V functions are entwined into a single linear system. [1]

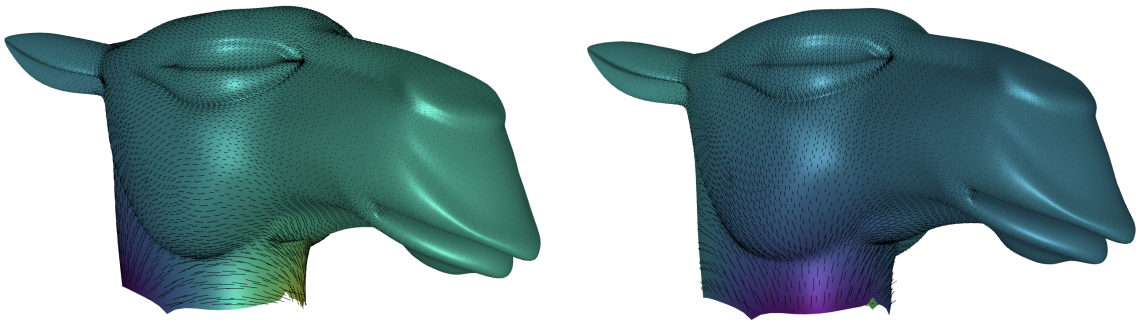
Firstly, we display the computed mapping functions for Harmonic parametrization in Figure 5.



(a) (b)

Figure 5: plots of the uv mapping using harmonic parameterization and the uv plane

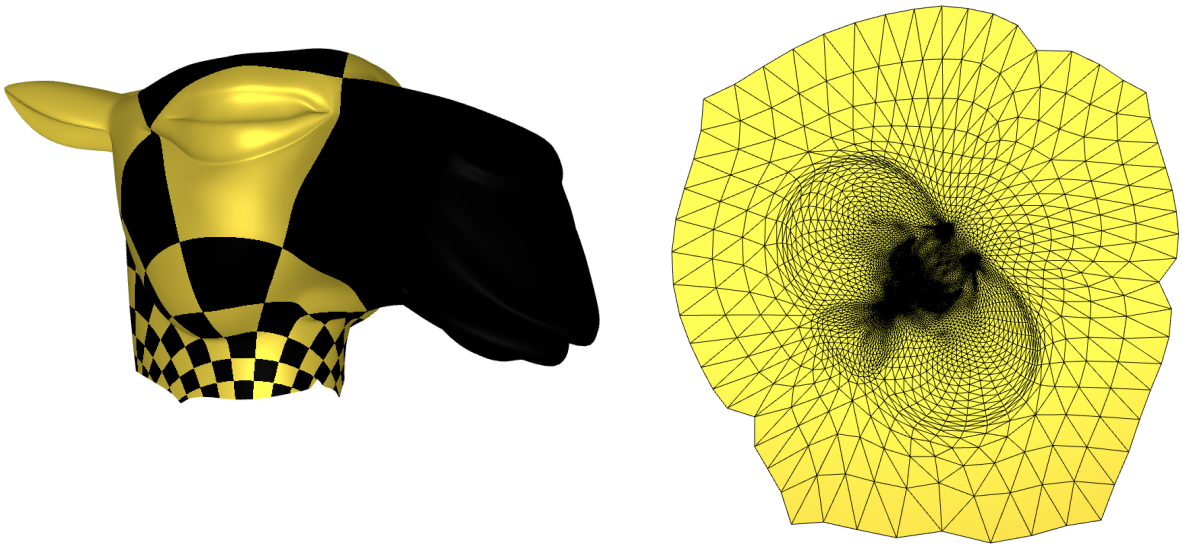
Then, we display the gradients for Harmonic parametrization in Figure 6.



(a) the gradient lines for u function, u as color map (b) the gradient lines for v function, v as color map

Figure 6: plots of gradients using harmonic parameterization

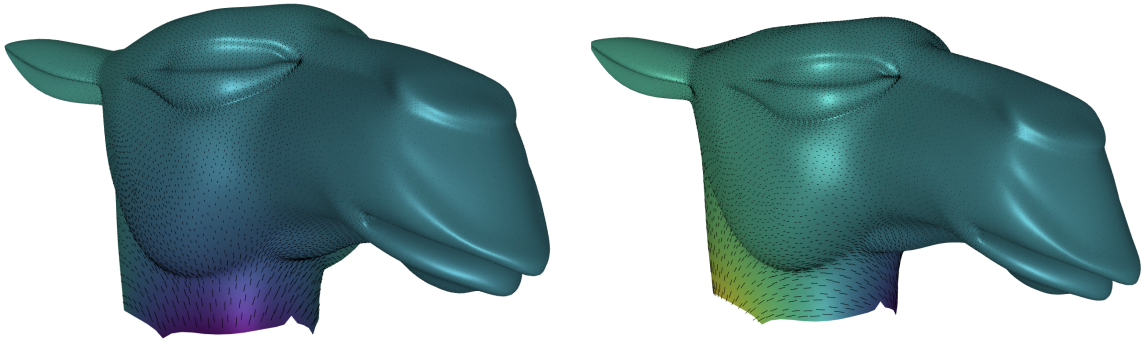
Then, we display the computed mapping functions for LSCM Harmonic parametrization in Figure 7.



(a) (b)

Figure 7: plots of the uv mapping using LSCM parameterization and the uv plane

Then, we display the gradients for LSCM parametrization in Figure 8.



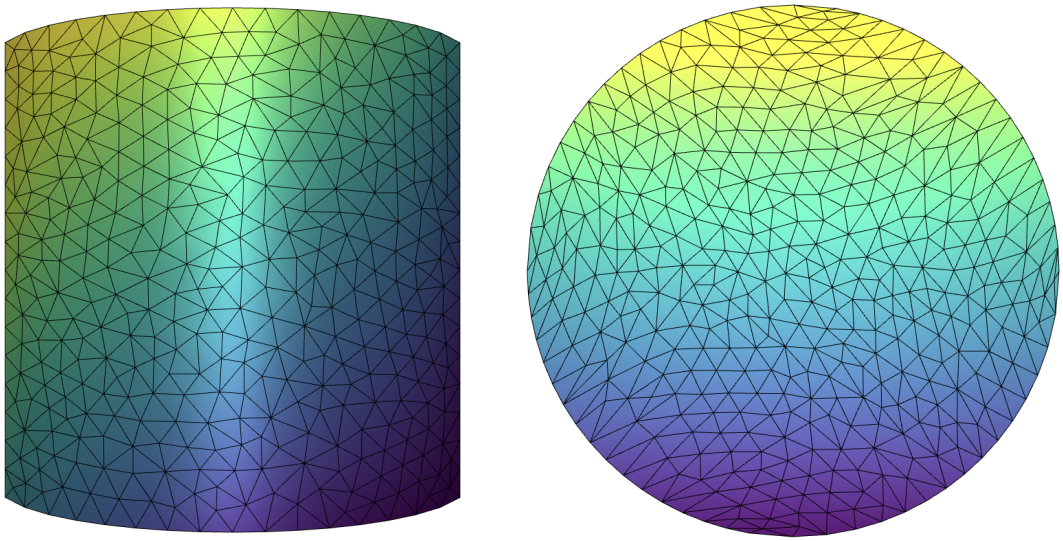
(a) the gradient lines for u function, u as color map (b) the gradient lines for v function, v as color map

Figure 8: plots of gradients using LCSM parameterization

Editing a parameterization with vector fields

A parameterization consists of two scalar coordinate functions on the surface. As such, we can use vector fields to guide the parameterization: we can design a vector field and fit one of the coordinate functions' gradients to it. Starting with Harmonic parameterization, we use the results of the previous steps to replace one of the U, V functions with a function obtained from a smooth user-guided vector field [1]. Then visualize the resulting U or V replacement function and its gradient atop the mesh, and texture the mesh with the new parameterization.

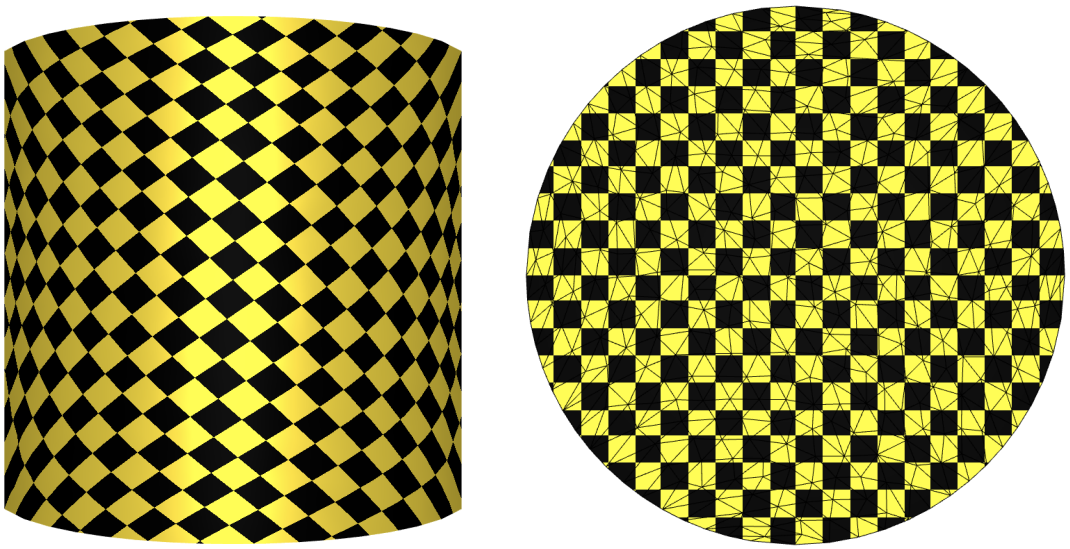
For editing the parameterization, we display the initial harmonic parameterization colored with v and the mapping uv plane with the color map as v function in Figure 9.



(a) (b)

Figure 9: initial harmonic parameterization colored with v and the mapping uv plane with the color map as v function

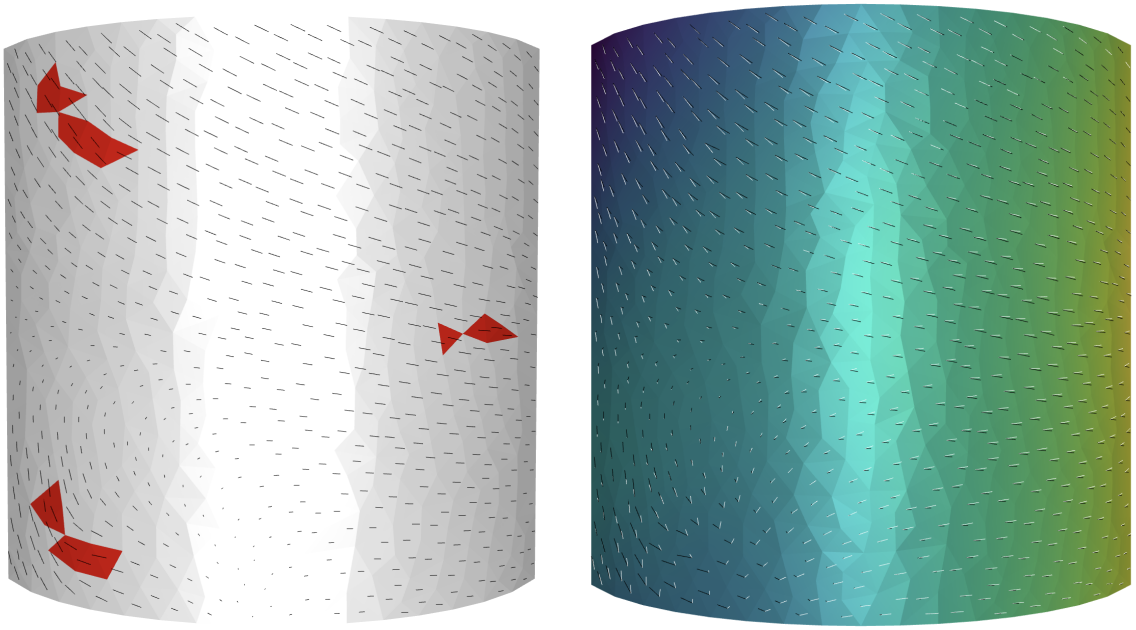
Then we display the checkerboard: initial harmonic parameterization and the mapping uv plane in Figure 10.



(a) (b)

Figure 10: checkerboard: initial harmonic parameterization and the mapping uv plane

Then the user-provided constraints are first interpolated, and a scalar function is reconstructed as shown in Figure 11.



(a) user-provided constraints (b) scalar function

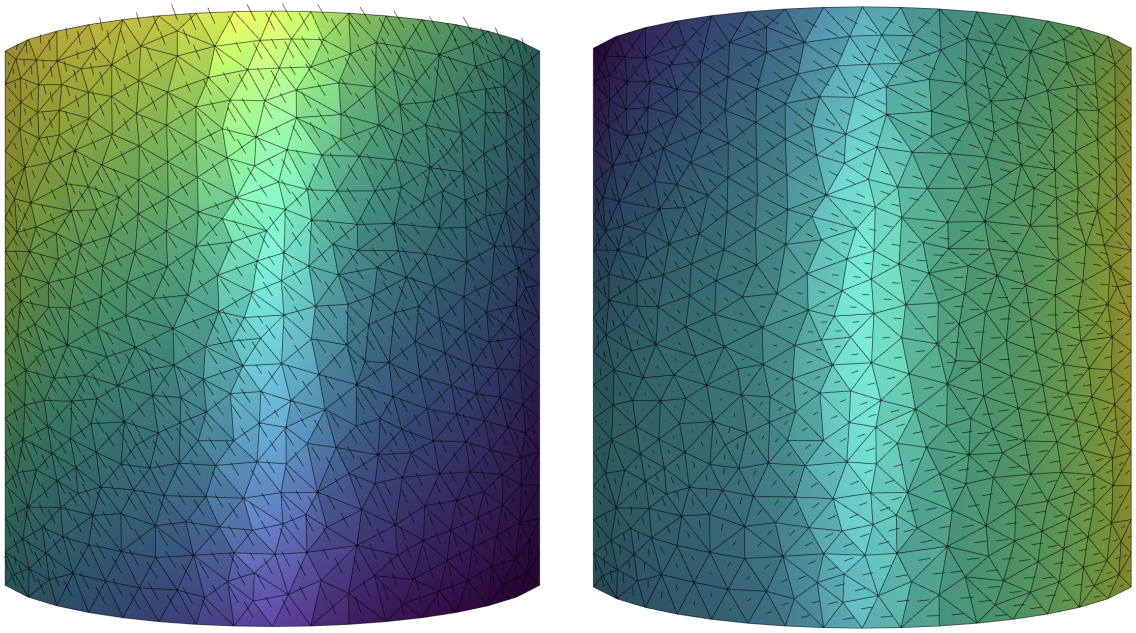
Figure 11: user-provided constraints and scalar function

The resulting mapping with v function replaced with the scalar function s with its gradient atop the mesh, and texture the mesh with the new parameterization as shown in Figure 12.

(a) (b)

Figure 12: resulting mapping with v function replaced with the scalar function s with its gradient atop the mesh, and texture the mesh with the new parameterization

Then we plot the gradients of the v function as the color map with the gradient of v and plot the new scalar function s and its gradient with s being the color map as shown in Fugure 13.



(a) (b)

Figure 13: scalar functions and gradients

It is possible for a parameterization created in this way to cause triangles to flip over as they are mapped into the UV plane. We determine a reliable criterion for detecting flipped triangles and visualize the planar mapped mesh with the flipped faces highlighted in red.

Detecting the flipped triangles due to the update in us from uv parameterization, we solve it by detecting if two edges of a triangle are arranged in different order in uv and us planes. If the cross product of two edges have different signs in two different cases, then the edges or vertices in the us mapping are clockwise, so that we know that this triangle is flipped. Triangles that are red are flipped.

Firstly, we plot the visualization of flipped elements in the original object as shown in Figure 14.

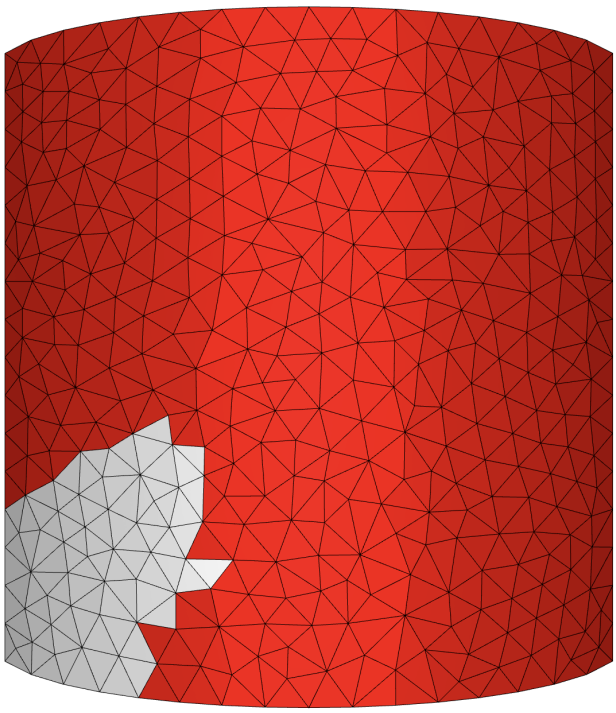


Figure 14: visualization of flipped elements in the original object

Then, we plot the visualization of flipped elements in the planar mapped mesh uv as shown in Figure 15.

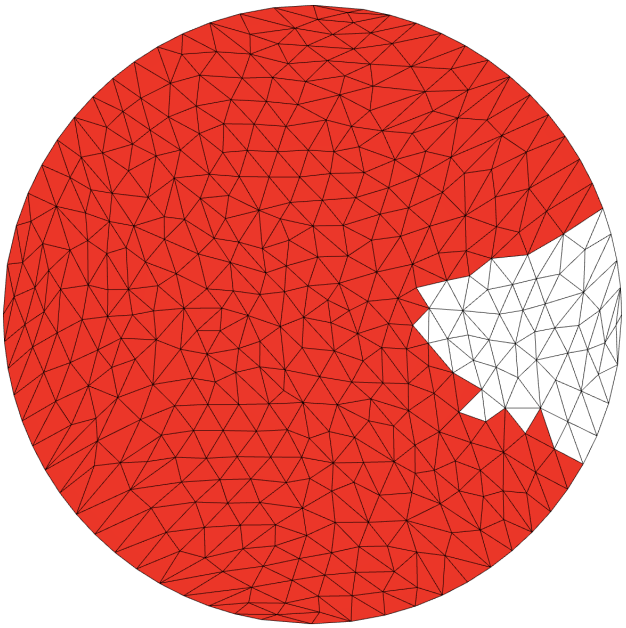


Figure 15: visualization of flipped elements in the planar mapped mesh uv

Last but not least, we plot the visualization of flipped elements in the planar mapped mesh *us* as shown in Figure 16.

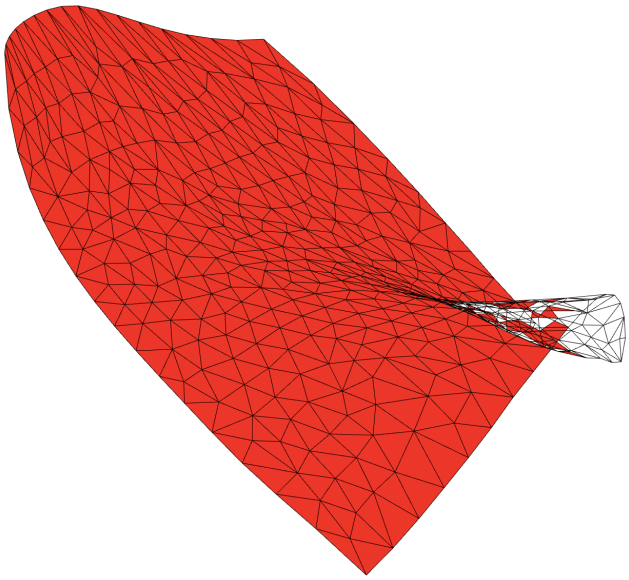


Figure 16: visualization of flipped elements in the planar mapped mesh *us*

Finally we give the flipped triangle indices as shown in Figure 17.

```
the flipped triangle indices:
[  0   1   2   3   4   5   7   8   9  10  11  12  13  14  15  16  17  18
  19  20  21  22  23  26  27  28  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  59  60  61  62  63  64  65  66  67  68  69  70  72  73  74  75  76
  77  78  79  80  81  82  83  84  85  86  90  95  96  97  98  99 100 101
 102 103 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121
 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193
 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211
 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229
 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247
 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265
 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283
 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
 303 305 306 307 320 321 322 324 326 327 328 329 330 331 332 334 335 337
 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355
 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373
 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391
 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409
 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427
 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463
 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481
 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499
 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
 518 519 520 522 523 530 531 532 533 534 535 536 539 541 542 543 544 545
 546 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 569
 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587
 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605
 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623
 624 625 626 627 628 629 630 631 632 633 634 635 636 639 665 666 667 668
 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688
 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706
 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724
 725 726 727 728 729 730 731 733 734 735 736 737 738 739 746 747 748 749
 750 751 752 753 754 755]
```

Figure 17: the flipped triangle indices

Conclusion

In this project, I introduced and implemented a sequence of methods that leverage vector field design on surfaces of 3D objects, create scalar field whose gradients align with the given vector fields as closely as possible, experiment with harmonic and least-square conformal parameterizations and visually demonstrate the working flow of parameterization editing with vector fields. In addition, I used different models as examples to visually demonstrate the effectiveness of the methods and show that these methods are efficient and effective to accomplish the tasks of computer model 3D surface parametrization in application.

References

[1] T. Schneider, *Csc486b: Geometric modelling course materials.*