# Surface Networks via General Covers - Supplementary Material

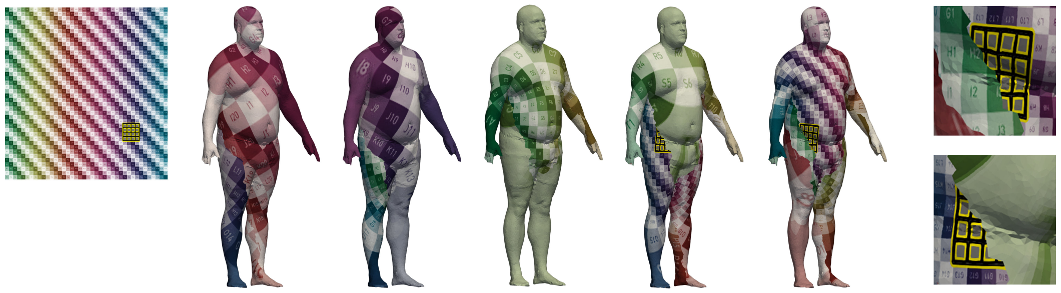

Figure 1: A topological torus (textured image, left) covers a human model 5 times (middle). This cover is constructed out of 5 copies of the model stitched to form a topological torus (middle). The map from the flat torus (left) to the cover is visualized using a colored checkerboard. Since the 5 copies form a torus, up/down/right/left translations are well-defined everywhere on the cover. The standard image convolution is invariant to these translations over the surface (see right insets where the convolution kernel moves seamlessly between copies of the model).

## 1. Convolution on a spherical mesh

In Figure 1 we depict a cover map from the torus (texture square image on the left) to a human surface (middle); this map covers the human $5$ times. We further show how standard convolution stencil (in yellow) translates to a seamless convolution on the surface. Note that the texture seams on the human models are pretty arbitrary and just indicate when moving to a different copy of the surface.

## 2. Guidelines on Choosing parameters

Adding branch points helps reducing the local distortion in protruding parts, therefore we recommend to choose as many branch points as there are protruding parts common in the dataset (*e.g.* $5$ for humans, $8$ for octopuses etc.). As we mentioned in section 4.1.1 we choose a ramification type of the form $[[r, 1^{d-r}]]$ for each branch point.

As noted in Section 4.1.1, higher ramification ($r$) also improves area distortion of protruding parts. However, in that case, we are limited by the RH formula ((6)). So we would recommend choosing the highest $r$ possible (*e.g.* as appears in Table 1) and taking $d$ (number of copies) to satisfy (6). Also note that higher $r$ implies higher $d$ (number of copies). Therefore, for a fixed image resolution we would like the highest number of branch points for which all relevant parts are still visible in the image.

## 3. Gluing Instructions

As mentioned in Section 4.1.1, for each choice of number of branch points $k$ , degree $d$ and ramification type $\rho$ satisfying Equations (5) and (6) We need to compute a product one tuple of permutations satisfying the conditions of theorem 1. We note that this computation can be done in an offline step, before using Algorithm 1 to compute the toric parameterization. In Table 1 We provide gluing instructions corresponding to each valid choice of $k \leq 6$, $d \leq 10$ and $\rho$ that complies with Equations (5) and (6). Each of the gluing instructions in Table 1 can be used as input to Algorithm (1).

## 4. Implementation Details

**Learning.** We use Pytorch [5] for learning. All the experiments are done with toric images generated by our algorithm and off-the-shelf CNN architectures with a single change: we replace the standard zero padding with periodic padding.

**Data generation.** For the surface segmentation task we use a cover of the type $\rho = \left[[1^5, 5]^5\right]$, that is, $\rho_i = [1^5, 5], i \in [5]$. For the spherical learning tasks (shape retrieval and classification) we use a cover of type $\rho_i = [1, 2], i \in [6]$. The locations of the branch points are chosen using farthest points sampling. We use the shortest

| $k$ | $d$ | $\rho$ | Gluing instructions |
|---|---|---|---|
| 3 | 3 | $\left[[3]^3\right]$ | $(1,2,3),(1,2,3),(1,2,3)$ |
| 3 | 6 | $\left[[1,5]^3\right]$ | $(1,2,3,4,5),(1,3,4,6,2),(2,6,3,5,4)$ |
| 3 | 9 | $\left[[1^2,7]^3\right]$ | $(1,2,3,4,5,6,7),(1,7,6,2,3,8,9),(1,9,8,2,5,4,3)$ |
| 4 | 2 | $\left[[2]^4\right]$ | $(1,2),(1,2),(1,2),(1,2)$ |
| 4 | 4 | $\left[[1,3]^4\right]$ | $(1,2,3),(2,3,4),(1,2,4),(1,2,3)$ |
| 4 | 6 | $\left[[1^2,4]^4\right]$ | $(1,2,3,4),(2,5,4,3),(1,5,6,4),(1,5,4,6)$ |
| 4 | 8 | $\left[[1^3,5]^4\right]$ | $(1,2,3,4,5),(3,6,8,5,4),(1,5,4,7,2),(2,7,4,8,6)$ |
| 4 | 10 | $\left[[1^4,6]^4\right]$ | $(1,2,3,4,5,6),(1,7,8,3,5,9),(2,10,8,7,6,5),(1,9,4,3,8,10)$ |
| 5 | 5 | $\left[[1^2,3]^5\right]$ | $(3,4,5),(2,3,5),(1,5,2),(1,2,5),(2,4,3)$ |
| 5 | 10 | $\left[[1^5,5]^5\right]$ | $(6,7,8,9,10),(1,7,3,4,9),(1,8,4,3,7),(2,5,4,7,6),(2,10,9,4,5)$ |
| 6 | 6 | $\left[[1^3,3]^6\right]$ | $(1,2,3),(2,5,3),(3,6,5),(3,5,6),(1,4,5),(3,5,4)$ |
| 6 | 9 | $\left[[1^5,4]^6\right]$ | $(1,9,3,5),(1,7,8,4),(3,7,5,6),(4,8,7,9),(1,3,6,2)$ |

Table 1: Gluing instructions for choices of $k, d, \rho$

paths from an arbitrary base point to all branch points in order to cut the mesh. When the mesh does not allow such a path we subdivide it locally (without changing its geometry). This pre-processing step is implemented in Matlab. It takes $\sim 22$ seconds in average (relatively long running time due to a non-optimized mesh cutting code in Matlab) to generate a periodic (toric) image for a mesh with 6890 vertices on a single CPU core in an Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz machine.

### 4.1. Segmentation Task

**Prediction.** The network outputs per-pixel labels. In order to obtain a label for each face in the original mesh $M$, we first transfer the per-pixel logits to the faces $F_T$ of the toric mesh using bilinear interpolation sampled at the faces' centers. Since each face $f$ in $M$ has $d$ duplicated faces in the toric mesh $T$ ($|\Psi^{-1}(f)| = d$), each face $f$ in $M$ has $d$ sets of logits. We use a weighted average of the $d$ sets of logits, where the weights are the area scales of the faces $\Psi^{-1}(f)$. The label of $f$ is the argmax of this weighted-average of logits. This means that better scaled faces (in the toric mesh) receive more weight when deciding how to label a face in the original mesh $M$.

**Architecture.** We use a version of a U-Net [7]. The feature-channels sizes are given in Table 2. After each convolution we use ReLU with a Batch-Normalization layer [4]. Each UpSample layer is a nearest-neighbour interpolation with scale-factor 2.

## 5. Proofs

### 5.1. Riemann-Hurwitz formula

Consider a branched covering map $E : T \to M$ of degree $d$ and $k$ branch points, from a toric mesh $T = (V_T, E_T, F_T)$ to a spherical mesh $M = (V_M, E_M, F_M)$. We prove that the ramification type of $E$ must satisfy the Riemann-Hurwitz formula (3).

*Proof of Riemann-Hurwitz formula.* First, we note that the set of branch points $B = \{b_1, \ldots b_k\}$ can always be chosen from $V_M$.

Every node $v \in V_M \setminus B$ has $d$ pre-images in $V_T$. However, a branch point $b_i$ has $l_i < d$ pre-images in $V_T$. Every edge $e \in E_M$ has exactly $d$ pre-images in $E_T$, that is $|E_T| = d|E_M|$. Similarly, $|F_T| = d|F_M|$.

By computing the Euler characteristic for a toric surface:

$$\begin{aligned} 0 = \chi(T) &= |V_T| - |E_T| + |F_T| \\ &= d\underbrace{(|V_M| - |E_M| + |F_M|)}_{\chi(M)} - \sum_{i=1}^{k}(d - l_i) \qquad (1) \\ &= 2d - \sum_{i=1}^{k}(d - l_i) \end{aligned}$$

Using
$$\sum_{j=1}^{l_i} r_{i,j} = d \tag{2}$$
and rewriting we obtain the Riemann-Hurwitz formula (RH), in its version for a map from a toric surface to a spherical surface:
$$\sum_{i=1}^{k}\sum_{j=1}^{l_i}(r_{i,j}-1) = 2d \tag{3}$$
□

## 5.2. Proof of Theorem 1

We recall the following topological facts. A degree $d$ branched covering map $E : T \to M$ from a torus to a genus 0 surface induces a group homomorphism, called the monodromy representation, from $\pi_1$, the fundamental group of $M \setminus \{b_1, \ldots, b_k\}$ to $S_d$.

The homomorphism is given as follows: We take each loop $l \in \pi_1$, based at a point $p$, and lift it to $T$ starting from a preimage of $p$. This lift has to end at another preimage of $p$. Due to properties of the lifting, this induces a permutation on the preimages of $p$ in $T$, referred to as the *fiber* of $p$.

The group $\pi_1$ has $k$ generators and a single relation. The generators, $l_1 \ldots, l_k$, are the $k$ loops around each of the branch points. The relation is $l_1 * \ldots * l_k = 1$.

Our gluing instructions, $\sigma_1, \ldots, \sigma_k$, will be the images of $l_1, \ldots, l_k$ under the monodromy representation. We shall now give a proof of Theorem 1 . Namely, that our algorithm produces a cover $T \to M$ with ramification $\rho$ if and only if the gluing instructions are a tuple of permutations satisfying the conditions of Theorem 1.

*Proof of Theorem 1.* First we prove that the conditions in the theorem are necessary.

For $(i)$, we note that a lift of a loop around a branch point $l_i$ with a particular ramification structure induces a permutation with the same cycle structure.
For $(ii)$, the fact that $l_1 * \ldots * l_k = 1$, implies (using group homomorphism) that $\sigma_1 \cdot \ldots \cdot \sigma_k = I_d$.
For $(iii)$, fix $p_1, p_2$ in the fiber of $p$. Since $T$ is connected, there exists a path $\gamma$ connecting $p_1$ and $p_2$. The loop $E \circ \gamma$ is a loop starting and ending at $p$ whose lift takes $p_1$ to $p_2$. Thus, the action of group generated by $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is transitive.

Conversely, suppose we have a product one tuple $\sigma_1, \ldots, \sigma_k$ satisfying the conditions of the theorem and $k$ branch points $b_1, \ldots, b_k$. Then condition (i) allows us to define an action of the group $H := \langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle$ on $[d]$. Following the construction in [3] pg 68-70 the space $U \times [d] / \pi_1 \times H$ is a covering space of $M$, where $U$ is the universal cover of $M$. The transitivity of $H$ implies that this covering space $C$ is connected. Condition $(iii)$ implies by the Riemann-Hurwitz formula that $C$ is topologically a torus.

Let $D$ be the space produced from Algorithm 1. Note that the construction in Algorithm 1 implies that lifting a loop circling each branch point $b_i$ induces the permutation $\sigma_i$ on the fiber of a generic point. Thus, the action of $\pi_1$ on $D$ coincides with the action of $\pi_1$ on $C$. Since every action of $\pi_1$ on $[d]$ (up to conjugation) produces a unique (up to homeomorphism) covering space, we deduce that $D$ is homeomorphic to $C$.

□

**Comment:** The equivalence between branched covering maps and tuples of permutations satisfying the conditions of Theorem 1 is well known. This equivalence is commonly referred to as Riemann's existence theorem (RET). However, to the best of our knowledge, it was previously not known how to practically construct any given branched covering map (our Algorithm 1).

## 6. Gluing Instructions

We now turn to describing an algorithm that finds tuples of permutations $\sigma_1, \ldots, \sigma_k \in S_d$, corresponding to a prescribed ramification structure $\rho$, up to simultaneous conjugation (relabeling of the branch points). We call such a tuple a product one tuple. We implement our algorithm using Magma computational algebra system [2].

We denote the conjugacy class in $S_d$ associated with the cycle structure of $\rho_i$ by $C_i$. In the algorithm construction we use the following:

**Claim 1.** *1. $\langle \sigma_1, \sigma_2, \ldots, \sigma_{k-1} \rangle$ is a transitive permutation group and $\Pi_{i=1}^{k-1}\sigma_i \in C_n$, if and only if $\sigma_1, \sigma_2, \ldots, \sigma_k$, where $\sigma_k = (\sigma_1\sigma_2 \cdots \sigma_{k-1})^{-1}$ is a transitive product one tuple with $\sigma_k \in C_k$.*

*2. The set $\{\sigma_1, \ldots, \sigma_i\}$ can be completed to a transitive product one tuple compatible with a ramification structure $\rho$ if and only if $\{\sigma_1, \ldots, \sigma_{i-1}, g\sigma_i g^{-1}\}$, for any $g \in Z(\sigma_1, \ldots, \sigma_{i-1})$ ($Z$ denotes the centralizer), can be completed to a transitive product one tuple compatible with $\rho$.*

*Proof.* (1) follows from the observations that adding elements to a transitive generator set keeps the set transitive, and that for $g \in S_d$ the cycle structure of $g$ and $g^{-1}$ are the same. For (2), note that for any $g \in Z(\sigma_1, \ldots, \sigma_{i-1})$ and $j \in [i-1]$ it holds that $g\sigma_j g^{-1} = \sigma_j$. Thus, for any $g \in Z(\sigma_1, \ldots, \sigma_{i-1})$, we have that any tuple with $\sigma_1, \ldots, \sigma_i$ is the same as a tuple with $g\sigma_1 g^{-1}, \ldots, g\sigma_i g^{-1}$, up to simultaneous conjugation. □

The main idea in the algorithm for finding all gluing instructions corresponding to a ramification type $\rho$ is to exhaustively go over all tuples $\sigma_i \in C_i$ and check whether they form a product one tuple. We use the claim above to prune this exhaustive search, as described in Algorithm 2. Note that this computation is done once for a given cover ramification type and is reused for all models using this type of cover.

```
Data: a ramification structure $\rho = (\rho_1, \dots, \rho_k)$
Result: all gluing instructions $\Sigma$ compatible with $\rho$
Pick $\sigma_1 \in C_1$
Call the recursive function tuplesFinder($\rho$,$\sigma_1$)

tuplesFinder($\rho$, $\{\sigma_1, \dots, \sigma_i\}$)
while $C_i \neq \emptyset$ do
    pick $\sigma_i \in C_i$
    update $C_i = C_i \setminus$
      $Z(\sigma_1, \sigma_2, \dots, \sigma_{i-1})\sigma_i Z(\sigma_1, \sigma_2, \dots, \sigma_{i-1})^{-1}$
    if $i = n - 1$ then
        if $\Pi_{i=1}^{n-1}\sigma_i$ and $\sigma_n$ are conjugates and
          $\langle \sigma_1, \sigma_2, \dots, \sigma_{n-1} \rangle$ is transitive then
            add $\sigma_1, \sigma_2, \dots, \sigma_k$ to list of product one
              tuples
        end
    else
        call tuplesFinder($\rho$, $\{\sigma_1, \sigma_2, \dots, \sigma_i\}$ )
    end
end
```
**Algorithm 2:** Finding gluing instructions.

## 7. Orbifold-Tutte embedding of $T$

We compute $x$ by solving a sparse linear system following [1]:

$$Ax = b \tag{4}$$

Here $A \in \mathbb{R}^{m\times m}$ and $x, b \in \mathbb{R}^{m\times 2}$, where $m$ is the number of vertices in the disk-like mesh $T_{disk}$. The linear system (4) is constructed by putting together four sets of linear equations as follows:

First, for all interior vertices we set the discrete harmonic equation:

$$\sum_{u\in N_v} w_{vu}\,(x_v - x_u) = 0 \tag{5}$$

where $N_v$ is the set of vertices in $V_{T_{disk}}$ adjacent to $v$ and $w_{uv}$ are the cotangent weights [6].

Let $L_1$ and $L_2$ be the generators of the homotopy group of $T$. Denote by $v_0 \in V_T$ the intersection of the two loops $L_1$ and $L_2$ . In $T_{disk}$, the vertex $v_0$ has four copies $v_1', v_2', v_3', v_4'$. next, we ensure that these four copies are mapped to the four corners of the unit square $[0,1]^2$. Explicitly,

$$v_1' = [0,0]^T, v_2' = [1,0]^T, v_3' = [1,1]^T, v_4' = [1,0]^T \tag{6}$$

Each vertex $v \in \partial V_{T_{disk}} \setminus \{v_1', v_2', v_3', v_4'\}$ has a twin vertex $\tilde{v}$ such that $v$ and $\tilde{v}$ correspond to the same vertex in the uncut mesh $T$. Moreover, each such vertex $v$ has its origin in $V_T$ either in $L_1$ or in $L_2$.

We set the vertices whose origin is in $L_1$ to be different by a constant translation in $[0,1]^T$ and the vertices whose origin is in $L_2$ to be different by a constant translation in $[1,0]^T$. Namely:

$$\tilde{v} - v = a \tag{7}$$

where $v$ and $\tilde{v}$ are twins, and $a$ is either $[0,1]^T$ or $[1,0]^T$, depending on whether the origin of $v$ belongs to $L_1$ or $L_2$.

Finally we set each vertex $v \in \partial V_{T_{disk}} \setminus \{v_1', v_2', v_3', v_4'\}$ to be the weighted average of both its neighbors and the translated neighbors of its twin.

$$\sum_{u\in N(v)} w_{uv}(x_v - x_u) + \sum_{u\in N(\tilde{v})} w_{u\tilde{v}}(x_{\tilde{v}} - x_u + a) = 0 \tag{8}$$

with $a$ as before.

## References

[1] Noam Aigerman and Yaron Lipman. Orbifold tutte embeddings. *ACM Trans. Graph.*, 34(6):190–1, 2015.

[2] MAGMA Group et al. Magma computational algebra system, version 2.22-7, sydney. 2016.

[3] Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[5] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[6] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Table 2: Channel sizes of our U-Net architecture for surface segmentation

| Spatial Dimensions | Layer | kernel size | # input channels | # output channels |
|---|---|---|---|---|
| 512 x 512 | Conv2d | 5 | 3 | 128 |
| | Conv2d | 3 | 128 | 128 |
| | MaxPool2d | 2 | | |
| 256 x 256 | Conv2d | 3 | 128 | 128 |
| | Conv2d | 3 | 128 | 128 |
| | MaxPool2d | 2 | | |
| 128 x 128 | Conv2d | 3 | 128 | 128 |
| | MaxPool2d | 2 | | |
| 64 x 64 | Conv2d | 3 | 128 | 256 |
| | MaxPool2d | 2 | | |
| 32 x 32 | Conv2d | 3 | 256 | 512 |
| | MaxPool2d | 2 | | |
| 16 x 16 | Conv2d | 3 | 512 | 512 |
| | Conv2d | 3 | 512 | 512 |
| | UpSample | | | |
| 32 x 32 | Conv2d | 3 | 1024 | 256 |
| | Conv2d | 3 | 256 | 256 |
| | UpSample | | | |
| 64 x 64 | Conv2d | 3 | 512 | 128 |
| | UpSample | | | |
| 128 x 128 | Conv2d | 3 | 256 | 128 |
| | UpSample | | | |
| 256 x 256 | Conv2d | 3 | 256 | 128 |
| | UpSample | | | |
| | Conv2d | 3 | 256 | 128 |
| 512 x 512 | Conv2d | 1 | 128 | 8 |