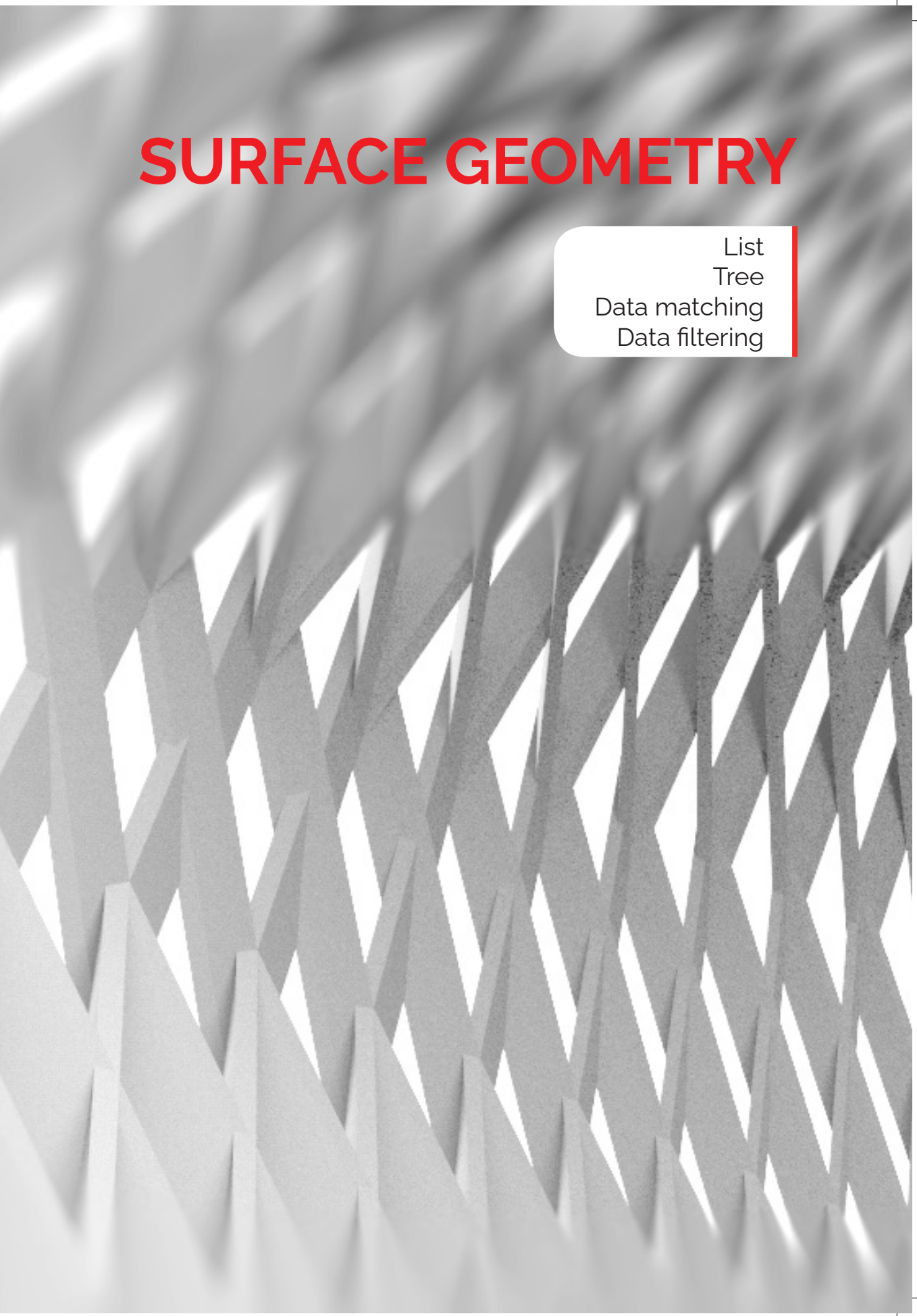


SURFACE GEOMETRY

List
Tree
Data matching
Data filtering

6. SURFACE GEOMETRY

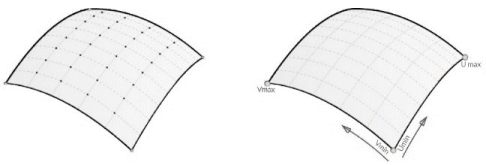
File latihan di bab ini dapat diunduh di:
https://github.com/aswinindra/eskacangmerah

NURBS (Non-Uniform Rational B-splines) adalah representasi matematik yang secara akurat mendeskripsikan obyek baik 2D maupun 3D, baik polygon maupun organic dan kurvalinier. Karena kapabilitas dan fleksibilitas ini, obyek dan model berbasis NURBS digunakan dalam berbagai industri: ilustrasi, visualisasi, animasi hingga manufaktur.

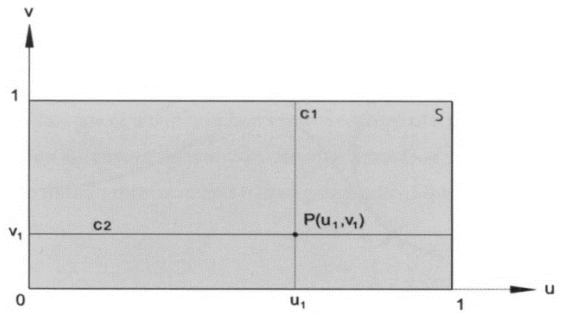
Seperti halnya NUBRS Curve, hampir semua obyek dapat dibuat model menggunakan NURBS Surface.

6.1. NURBS Surface

Karakteristik NURBS Surface hampir sama dengan NURBS Curve misalnya kelengkungan (curvature), normal, tangent dan lainnya. Perbedaan utama antara keduanya terletak pada dua aspek: **Curve** memiliki **tangent vector** dan **normal planes**, namun Surface memiliki **normal vector** dan **tangent planes.** Apa artinya? Curve memiliki kekurangan di orientasi sedangkan Surface memiliki kekurangan di arah/direction. Mengapa? Surface memiliki **dua arah vector yang dihasilkan dari Rectangular Grid** yang membentuk sebuah Surface. Rectangular Grid ini memiliki arah {u} dan {v} dan merupakan kumpulan Curve pada arah {u} dan {v}.



Anda dapat membayangkan NURBS Surface adalah grid dari NURBS Curve pada dua arah. Bentuk SURBS Surface ditentukan dari banyaknya control point pada setiap perpotongan curve tersebut.



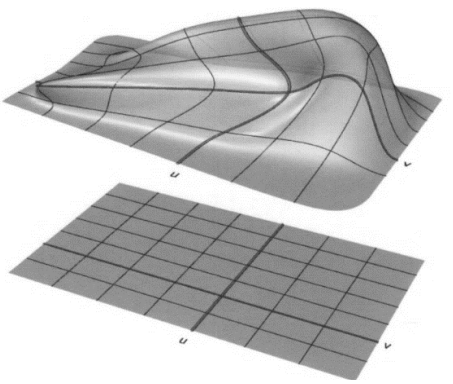
Gambar 141. NURBS Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

Seperti halnya Curve, Representasi Surface ditentukan menggunakan LCS (Local Coordinate System). Setiap titik *pada* suatu Surface dipetakan menggunakan parameter **u** dan **v**, dimana nilai **u** dan **v** berada diantara 0 dan 1. Ini mirip dengan parameter **t** pada Curve.

Pada sebuah Surface, pada setiap nilai di {u} dan {v}, titik $P(u_1,v_1)$ merupakan perpotongan (Sectional Curves) antara Curve C_1 dan Curve C_2 seperti terlihat pada gambar di atas. Curve C1 dan C2 ini dinamakan **Isocurve**s atau **Isoparametric Curves**. Jadi, pada contoh di atas, C_1 dan C_2 adalah Isocurves untuk Surface S pada titik P (u_1, v_1). Isocurves adalah sebuah Curve dangan nulai **u** atau **v** yang konstan.

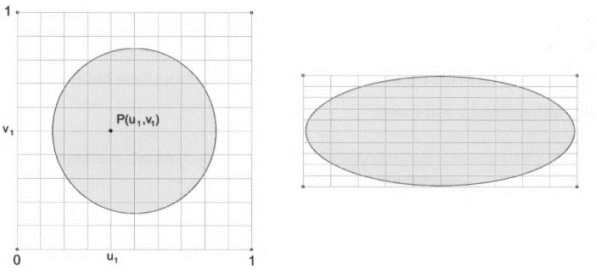
Isocurves membuat Rectangular Grid yan mendefinisikan grid Cartesian pada sebuah permukaan/ Surface baik planar maupun non-planar. Konsep ini berlaku untuk setiap *freeform surface* karena sebuah NURBS Surface dapat dibayangkan sebagai produk deformasi dari sebuah NURBS Surface planar/datar. Pada suatu Surface, terdapat dua buah domain/ **bidimensional/bidirectional domains** (**domain²**) yang menentukan range nilai **u** dan **v**.

Pada contoh di bawah, sebuah Surface yang kompleks diproduksi oleh Surface yang **flat** dan **rectangular**.



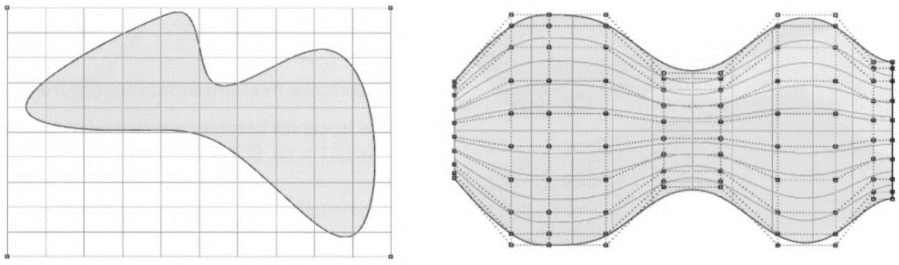
Gambar 142. Complex dan Flat Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

Pada Surface yang non-rectangular, misalnya berbentuk lingkaran, ellipse atau yang lain, basisnya tetap Rectangular Grid namun Rhino akan menyembunyikan (*hide*) area yang berada di luar bentuk tersebut. Sehingga dalam kasus ini, **domain²** tetap berbentuk Rectangular.



Gambar 143. Trimmed Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

Sebuah Surface yang tidak memiliki domain dinamakan **Trimmed Surface** dan Surface yang memiliki domain dinamakan **Untrimmed Surface**, seperti contoh berikut.

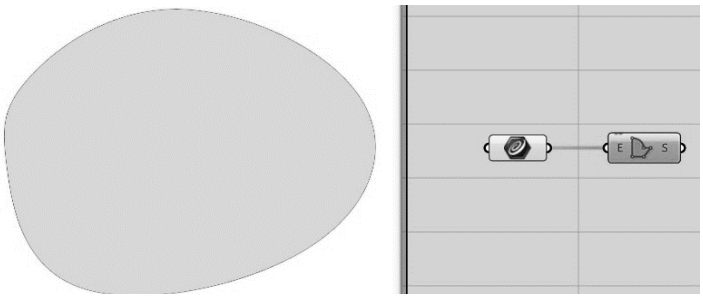


Gambar 144. Untrimmed Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

6.2. Membuat Surface

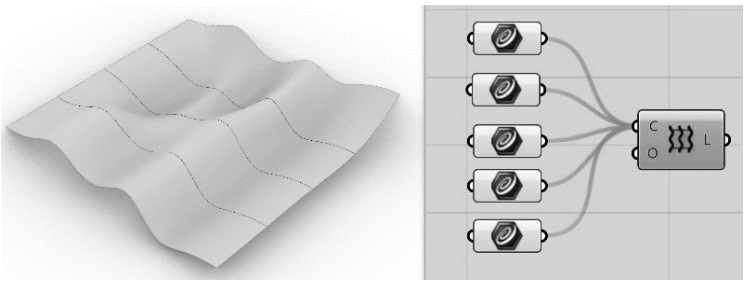
Surface dapat dibuat melalui fitur-fitur yang ada di Rhinoceros (seperti sudah dituliskan di bagian awal) maupun dari Grasshopper. Beberapa metode dalam membuat Surface di Grasshopper antara lain:

1. **Extrude**: membuat surface berupa bidang vertical dari sebuah profil kurva.
2. **Boundary Surface**: membuat surface planar baik trimmed maupun untrimmed dari kurva tertutup.



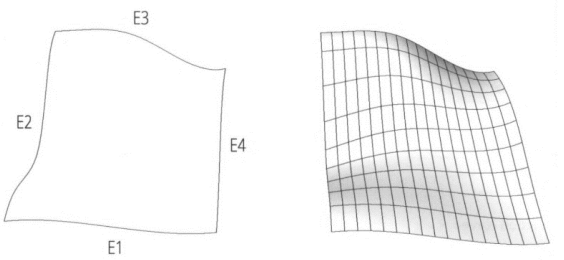
Gambar 145. Boundary Surface

3. **Loft**: membuat surface dari beberapa profil kurva yang disusun berurutan. Loft memerlukan setidaknya dua profil kurva. **Loft Option** digunakan untuk menentukan jenis loft surface dan beberapa seeting lain.



Gambar 146. Loft

4. **Surface from Point**: membuat surface dari titik-titik suatu grid. Komponen ini bias digunakan untuk membuat surface dari proyeksi titik-titik surface lain, atau hasil dari suatu fungsi tertentu.
5. **Edge Surface**: membuat surface dari 2,3, atau 4 Curve. Perhatikan bahwa assignment untuk setiap Curve harus berurutan.

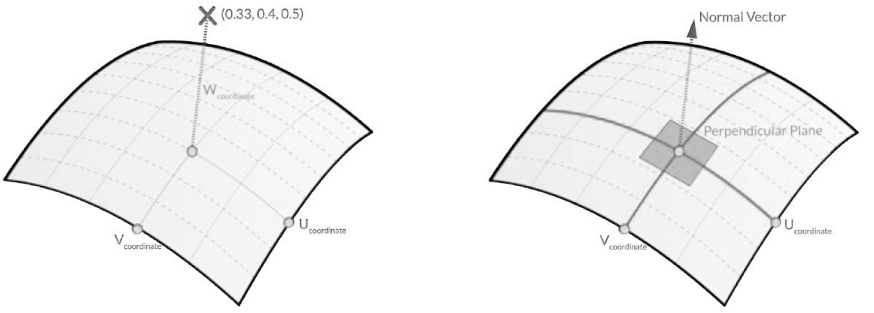


Gambar 147. Edge Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

6. **Metode lain** yang bisa dicoba sendiri adalah: **Patch, Network Surface, Sweep 1 & Sweep 2**.

6.3. Evaluasi Surface

Seperti halnya NURBS Curve, evaluasi Surface adalah menentukan posisi suatu titik **pada**/*on* Surface Domain. Harap diingat bahwa titik tengah pada domain (mid-u, mid-v) tidak selalu berarti berada di tengah-tengah/centroid suatu Surface.



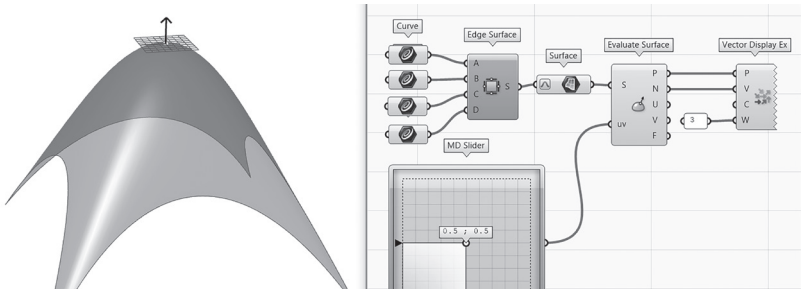
Gambar 148. Koordinat Lokal dan Vektor Normal Pada Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

6.4. Vektor Normal dan Bidang Tangensial (*Normal Vector & Tangent Planes*)

Sebuah bidang tangensial (*Tangent Planes*) pada suatu Surface pada suatu titik P adalah bidang yang *menyentuh* Surface pada titik P tersebut. Arah sumbu Z pada bidang tangensial menunjukkan arah **normal**/ ***Normal Direction***–Arah yang **tegak lurus** Surface pada titik tersebut.

6.5. Menemukan Titik pada Suatu Surface

Karena konsep NURBS Surface merupakan representasi dari Surface pada bidang datar, maka pada prinsipnya, bagaimanapun bentuk permukaannya, menggunakan representasi parametrik, kita dapat menemukan Point/ titik pada suatu Surface. Komponen **Evaluate Surface** digunakan untuk ini. Komponen ini memerlukan Surface yang akan dianalisis dan dua koordinat local (u dan v) agar setiap titik dalam Surface itu dapat dipetakan dalam u dan v. Untuk memetakan u dan v dalam Local Coordinate System, kita dapat menggunakan komponen **MD Slider**. Selain MD Slider, secara sederhana, input untuk uv misalnya: 0.5,0.5 (anda bisa gunakan Panel untuk ini).

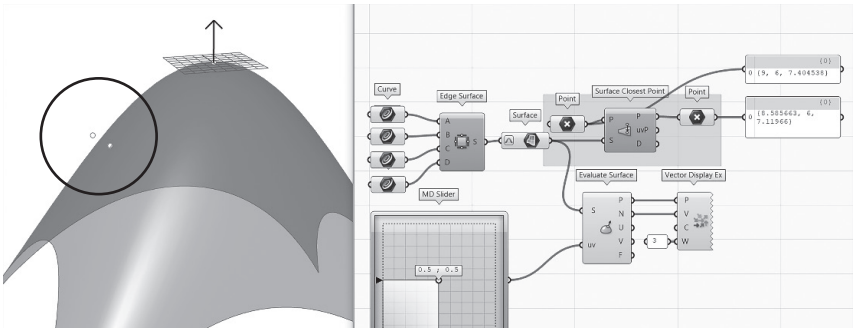


Gambar 149. Evaluate Surface

- Evaluate Surface akan menghasilkan beberapa titik termasuk: koordinat local titik sesuai dengan titik pada MD Slider, vector normal pada titik tersebut, serta informasi koordinat local untuk u dan v.
- *Pertanyaan untuk dijawab: Mengapa musti memasukkan komponen Surface sebagai input pada komponen Evaluate Surface? Toh keluaran dari Edge Surface juga Surface?*

6.6. Menemukan Titik pada Suatu Surface yang Tegak Lurus dengan Titik Di Luar Surface

Komponen **Surface Closest Point (Surface CP)** adalah komponen yang memproyeksikan Point yang berada di luar Surface (artinya posisinya menggunakan *World Coordinate System*) ke suatu Point pada Surface yang **tegak lurus** dengan titik di luar Surface tersebut. (Titik pada Surface menggunakan *Local Coordinate System*).

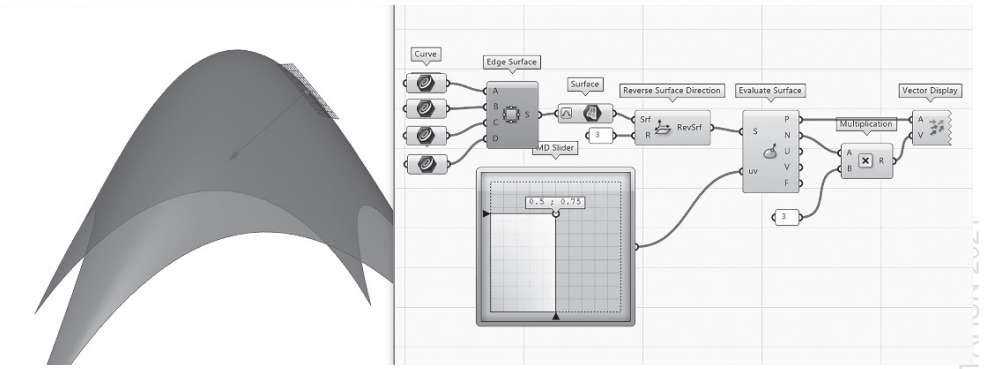


Gambar 150. Surface Closest Point

- Titik yang dihasilkan komponen Surface CP adalah titik pada Surface yang tegak lurus titik yang berada di luar Surface. Perhatikan bahwa titik ini memiliki DUA koordinat, satu adalah koordinat WCS, yang kedua adalah koordinat LCS.

6.7. Membalikkan Arah Vektor dari Suatu Surface

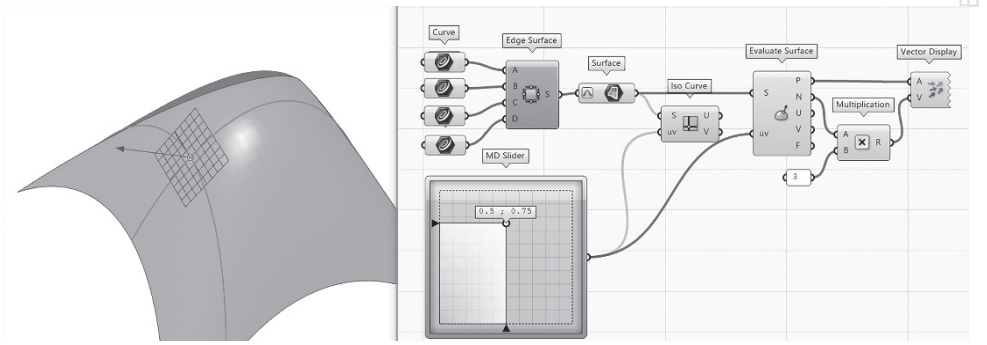
Jika komponen **Evaluate Surface** menginformasikan arah vector normal suatu Surface, maka ada komponen yang berfungsi untuk membalikkan arah vector normal ini. Komponen ini adalah: **Reverse Surface Direction**. Sayangnya komponen ini secara default tidak tersedia dan anda harus install **Lunchbox** (Gratis di Food4Rhino- Lunchbox).



Gambar 151. Reverse Surface Direction (Lunchbox)

6.8. Mengekstrak Komponen Pembentuk Surface (IsoCurve)

IsoCurve adalah sepasang kurva/ Curve yang memetakan setiap titik pada suatu Surface. Isocurve ini selalu memiliki dua nilai: u dan v.

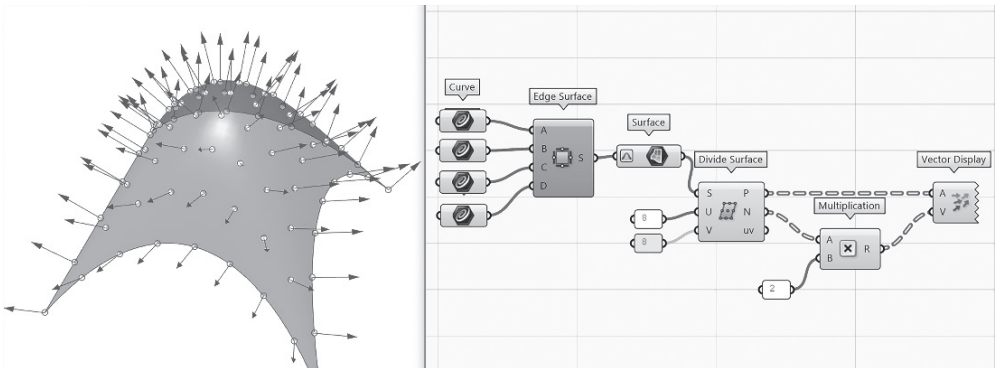


Gambar 152. IsoCurve

6.9. Membagi Surface

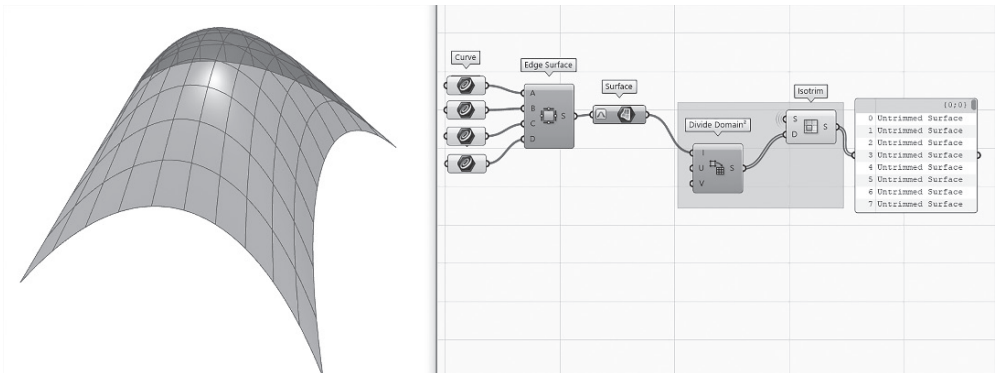
Jika komponen Divide Curve membagi suatu Curve menjadi beberapa segmen, maka komponen **Divide Surface** membagi Surface menjadi titik-titik grid. Komponen ini

memerlukan input Surface, banyaknya segmen u dan segmen v dan menghasilkan titik-titik P yang merupakan **titik potong IsoCurve pada setiap Grid**, vector normal (n) untuk setiap titip P dan informasi lokasi (u dan v) untuk setiap titik P. Perhatikan bahwa **Divide Surface TIDAK** menghasilkan grid, tetapi menghasilkan **TITIK POTONG dari grid**.



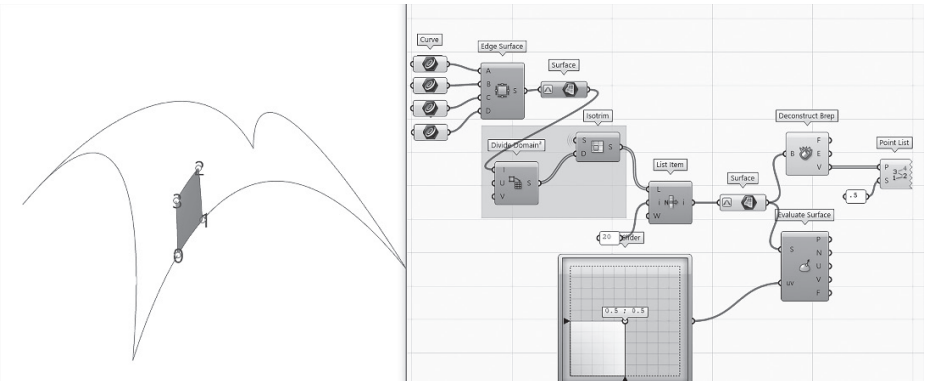
Gambar 153. Divide Surface

Komponen **Isotrim (SubSurface)** berfungsi untuk **membuat GRID** atau membuat **Sub Surface** pada suatu surface berdasarkan grid yang ditentukan (IsoCurve). Perhatikan bahwa Komponen Isotrim tidak memerlukan komponen Divide Surface untuk membuat SubSurface, tetapi memerlukan komponen **Divide Domain²** : komponen yang berfungsi membagi domain dua dimensi dari suatu domain.



Gambar 154. Divide Domain dan Isotrim

- Membagi Surface menjadi Sub Surface selalu diawali dengan membagi domain surface tersebut (**Divide Domain²**), setelah itu hasil domain menjadi input untuk komponen **Isotrim/ Subsurface** yang akan membagi surface menjadi grid sesuai dengan input dari Divide Domain².
- Hasil dari komponen Isotrim adalah Untrimmed Surface (face/patch) berupa **lis dari grid**.
- Karena berupa lis dari grid, maka kita bisa mengekstrak informasi dari surface grid tersebut.



Gambar 155. Subsurface

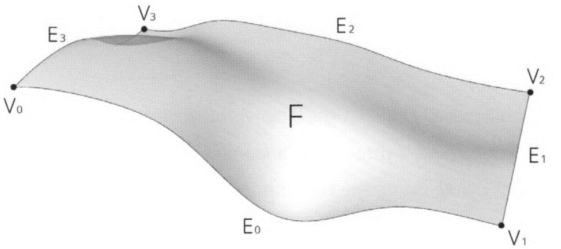
- Pada contoh di atas, setelah komponen **Isotrim** menghasilkan beberapa **SubSurface**, kita bisa ekstrak salah satu **SubSurface** dengan **List Item**.
- Kita bisa lihat bahwa pada setiap SubSurface, aturan posisi titik sama dengan Surface keseluruhan, kita bias gunakan Evaluate Surface untuk cek titik tengah di setiap SubSurface (SubSurface di-reparameterized).
- Pada setiap Subsurface, kita dapat cek **URUTAN** INDEX dari setiap titik Isocurve menggunakan komponen **PointList.**

6.10. Mengekstrak Komponen Surface

Jika komponen **IsoCurve** mengekstrak informasi u dan v dari suatu titik maka komponen **Deconstruct Brep** adalah komponen yang digunakan ungtuk mendekonstruksi suatu obyek polysurface, termasuk Surface. Sebuah Surface berbasis NURBS dapat dibayangkan merupakan obyek geometri yang dibangun dari beberapa entitas: Faces, Edges dan Vertices.

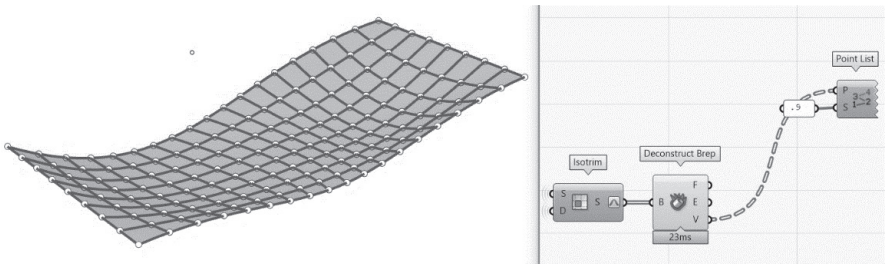
Untuk setiap SubSurface atau Surface, tergantung obyek yang akan diekstrak dengan komponen Deconstruct Brep, berlaku sebagai berikut:

- Sebuah Face (F_0) adalah bagian dari suatu Surface (yang terdiri dari beberapa Face).
- Empat buah Edges (E_0, E_1, E_2, E_3) adalah pembentuk Face dan merupakan empat buah Curves
- Empat buah Vertices (V_0, V_1, V_2, V_3) adalah keempat titik sudut yang membentuk face.



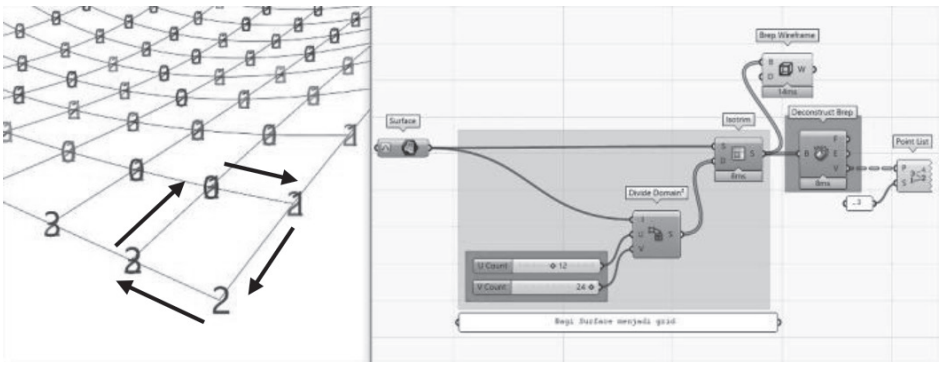
Gambar 156. Nonplanar Surface. (Sumber: Gil Akos & Ronie Parsons, 2017)

Jadi komponen Deconstruct Brep (DeBrep) akan menghasilkan lis berupa Face, Cuve dan Points.



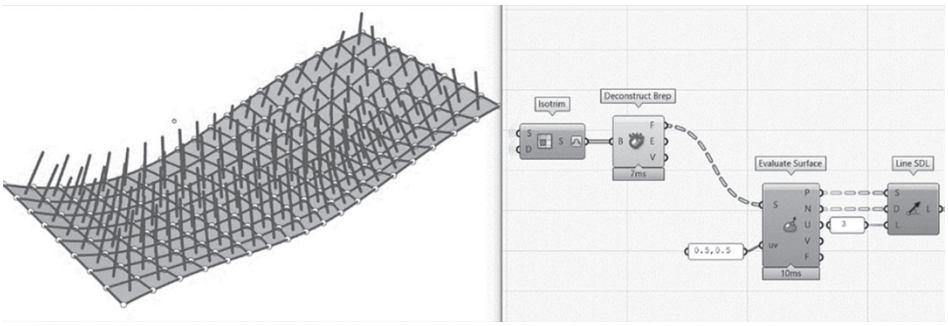
Gambar 157. Mengekstrak Surface

- Surface dapat diekstrak menjadi tiga elemen: Gridface (F), Edge (E)- Rusuk dan Vertices (V)- Titik Sudut. Masing-masing elemen ini merupakan daftar dalam bentuk Tree.
- Perhatikan bahwa sebuah Surface disusun oleh sekumpulan Gridface dimana setiap Gridface disusun oleh empat Vertices dan empat Edges. Susunan atau urutan Vertices ini juga seragam untuk semua Gridface. Kita dapat menampilkan urutan Vertices menggunakan **Point List.**



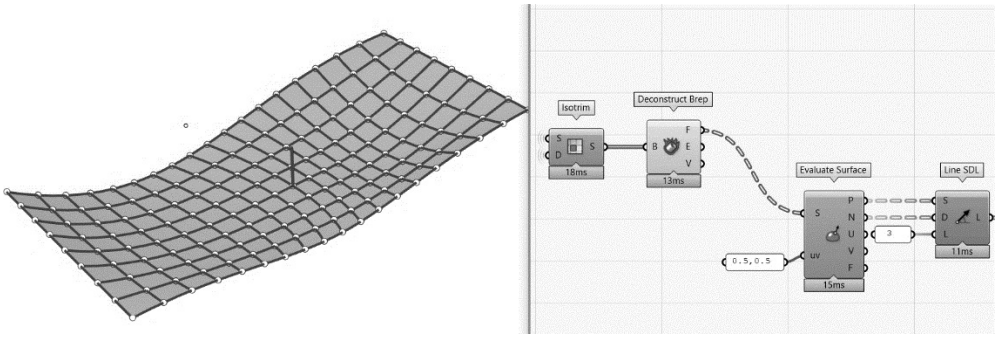
Gambar 158. Point List untuk Menampilkan Indeks Setiap Vertices

- Perhatikan bahwa urutan indeks di GridFace adalah sama, mulai dari 0 hingga 3.



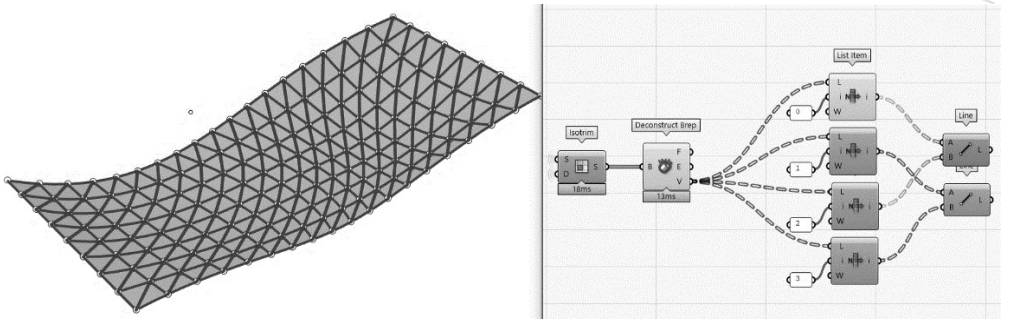
Gambar 159. Evaluate Surface pada Reparameterized Surface

- Jika sebuah surface dibagi menggunakan **Isotrim** dan hasilnya di-Reparameterized, maka nilai u dan v merupakan nilai unit dari setiap Gridface (nilainya masing-masing adalah 1), sehingga ketika dilakukan **Evaluate Surface** dengan nilai uv=0.5,0.5, maka titik tersebut adalah titik tengah di setiap Gridface.



Gambar 160. Evaluate Surface pada Sebuah Surface (Tidak di-Reparameterized)

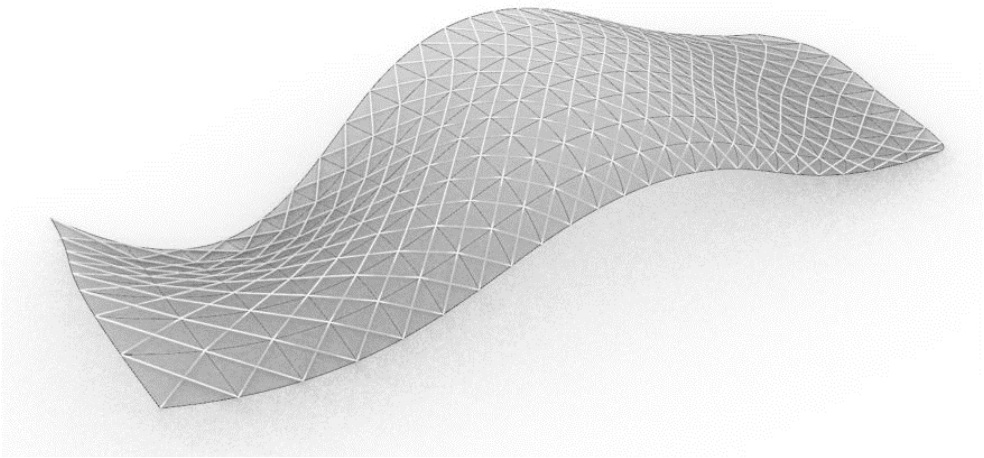
- Bandingkan dengan gambar di atas dimana nilai uv=0.5,0.5 untuk sebuah Surface berarti titik tengah dari Surface tersebut.



Gambar 161. Mengekstrak Vertices dengan Nomor Indeks

- Menggunakan **List Item,** kita dapat mengekstrak titik-titik Vertices sesuai dengan nomor indeksnya.

Latihan 11: *Surface Geometry*

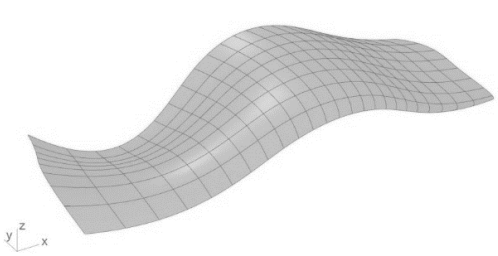


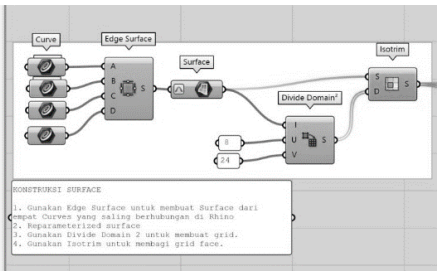
Gambar 162. Pola Diagrid Pada Permukaan Nonplanar

Misalnya kita akan membuat pola diagrid pada suatu permukaan (*Surface*). Kira-kira bagaimana konstruksi algoritmanya?

Prosedur atau algoritma:
1. Buat Surface, bagi ke dalam grid (parameter u dan v sebagai parameter untuk menentukan banyak grid)
2. Pada masing-masing grid surface (face), tentukan nomor indeks masing-masing titik vertices (indeks dimulai dari 0 dan maksimal 3).
3. Buat garis yang menghubungkan antara titik-titik pojok sedemikian rupa sehingga bersilangan.
4. Buat Pipe dari masing-masing garis tersebut.

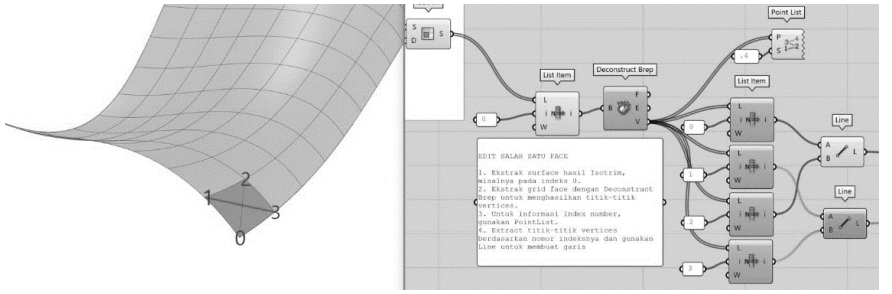
Langkah 1





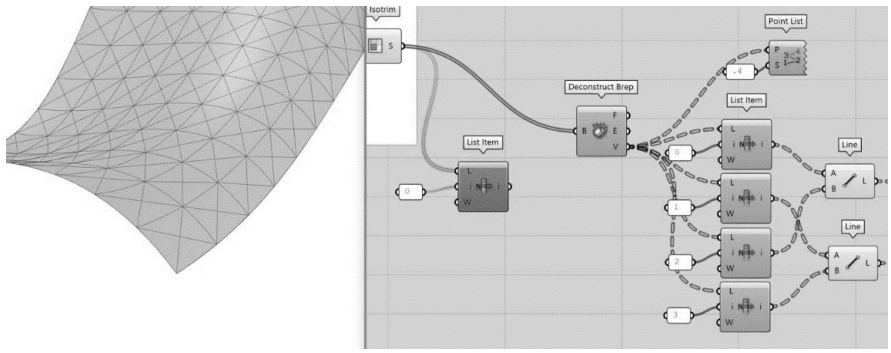
- Tentukan surface dari empat buah Curve yang dibuat di RH dengan Edge Surface.
- Reparameterized hasil surface.
- Buat grid dengan Divide Domain2
- Buat gride face dengan Isotrim.

Langkah 2



- Hasil dari Isotrim, ekstrak salah satu grid face dengan ListItem, misalnya index=0
- Akan terlihat mana grid face yang nomor indexnya=0.
- Gunakan komponen Deconstruct Brep pada gridface tersebut untuk mengekstrak titik-titik vertices. Gunakan komponen PointList untuk mengetahui mana titik dengan nomor indeks tertentu.
- Seperti contoh di atas, garis yang kita buat adalah dari vertices indeks 1 ke 3 dan vertices dari indeks 0 ke 2. Gunakan ListItem untuk mengekstrak ini.
- Gunakan Line untuk membuat garis.

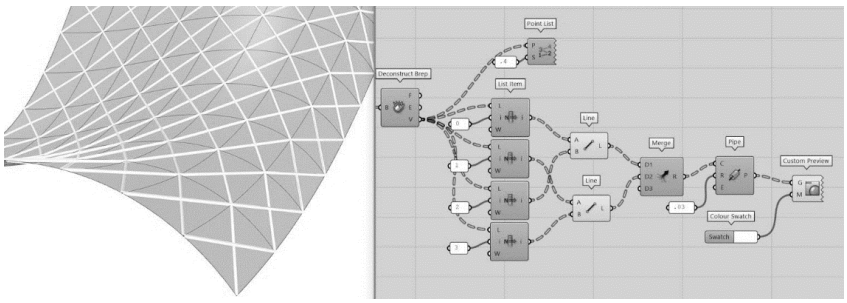
Langkah 3



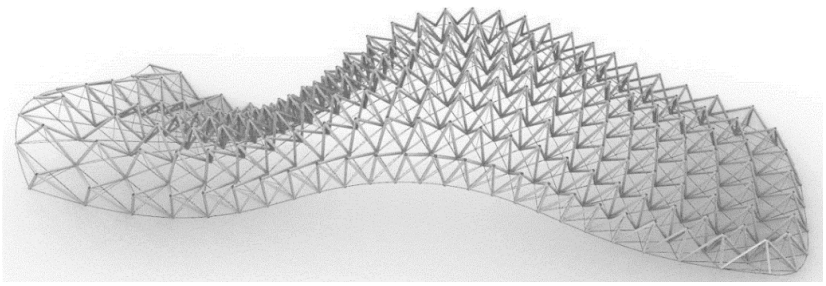
- Langkah 2 dilakukan pada salah satu grid face untuk mempermudah kita memahami bagaimana memanipulasi vertices. Karena proses ini dilakukan pada seluruh grid face, maka langkah selanjutnya adalah tidak lagi menggunakan ListItem untuk mengekstrak salah satu grid face tapi seluruh lis grid diekstrak menggunakan Deconstruct Brep.
- Anda akan lihat tidak ada komponen yang ditambahkan karena semua gridface memiliki aturan yang sama dengan salah satu gridface sehingga pola yang dilakukan pada salah satu grid face bias direpetisi.

Langkah 4

- Gunakan Pipe pada Line untuk membuat obyek pipa.



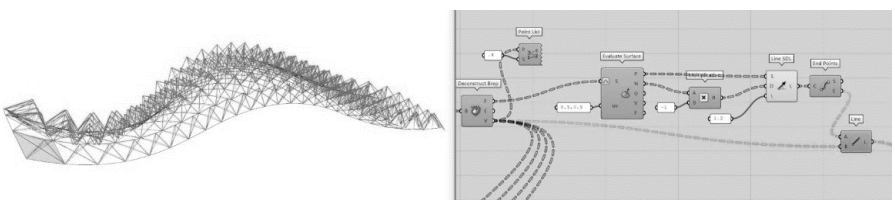
Dengan mengembangkan algoritma yang sama, bagaimana kita membuat obyek Space Frame seperti berikut?



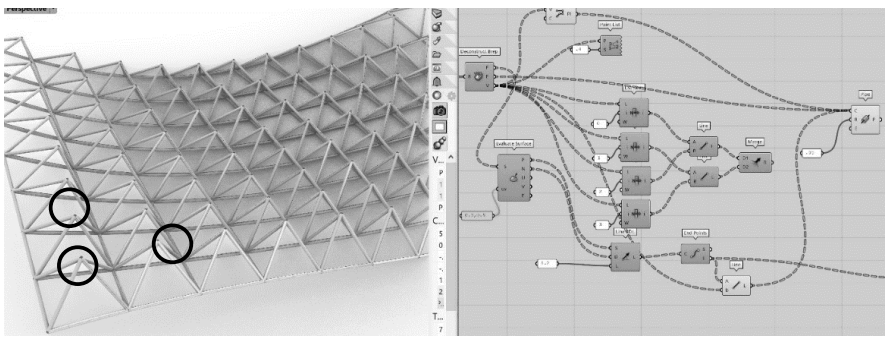
Gambar 163. Konstruksi Space Frame

Prosedur atau algoritma:

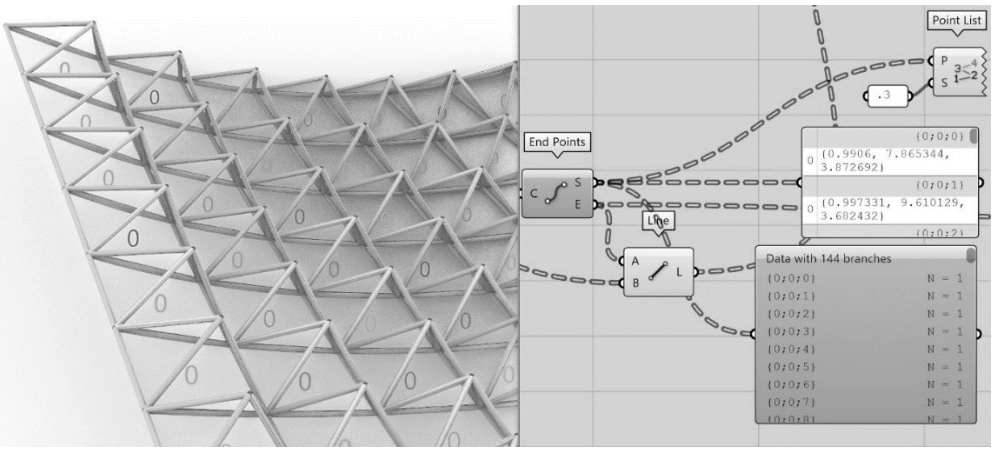
1. Evaluate Surface untuk mengetahui titik tengah dari setiap gridface.
2. Buat garis dari titik tengah tersebut tegak lurus permukaan.
3. Pada setiap titik Endpoint dari garis tersebut, buat garis dengan keempat titik sudut pada setiap gridface.



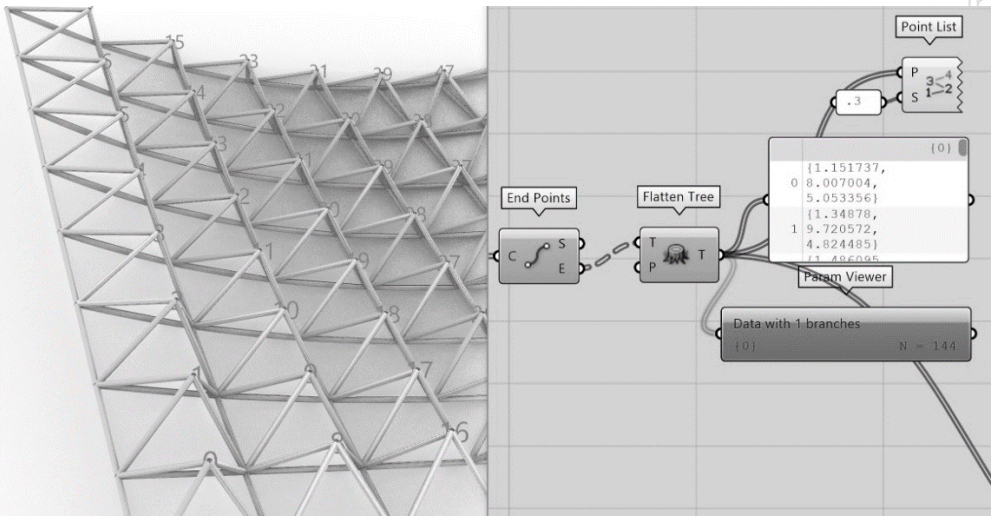
Silakan dipelajari dan dikembangkan dari definisi membuat pola diagrid sebelumhya.



Setelah bentuk seperti ini didapatkan akan ada pertanyaan, bagaimana menyatukan titik-titik puncak dalam sebuah garis-garis grid?

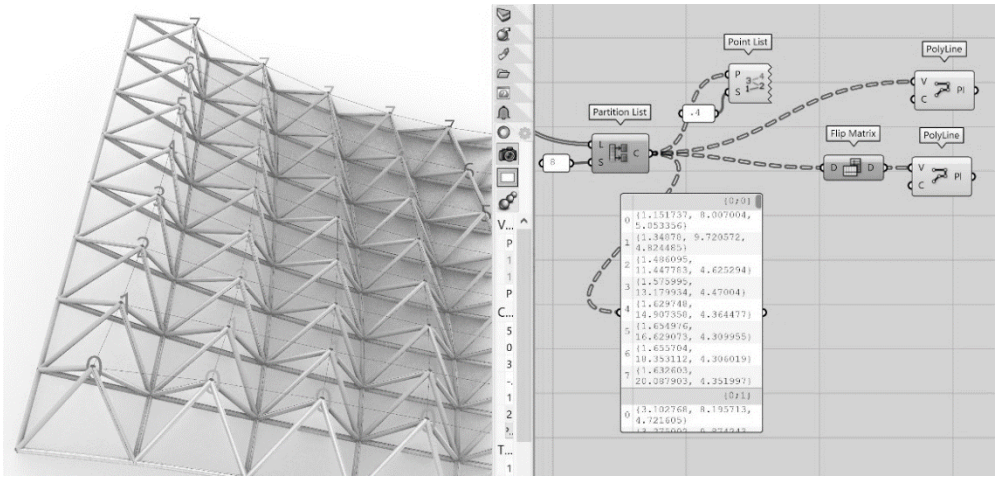


Dengan **PointList**, **ParamViewer** dan **Panel**, kita tahu bahwa semua nomor indeks di titik-titik tersebut adalah nol dan setiap titik merupakan cabang/path sendiri (lihat di ParamViewer, jumlah cabang=144 cabang). Kita tidak dapat membuat **Polyline** atau **Line** jika semua indeks nilainya sama. Karena itu, ini harus diubah, semua cabang dihilangkan sehingga kita hanya punya satu cabang utama dengan data berupa List. Komponen yang digunakan untuk menghilangkan cabang dan meletakkan semua data di cabang utama Namanya **Flatten Tree**.

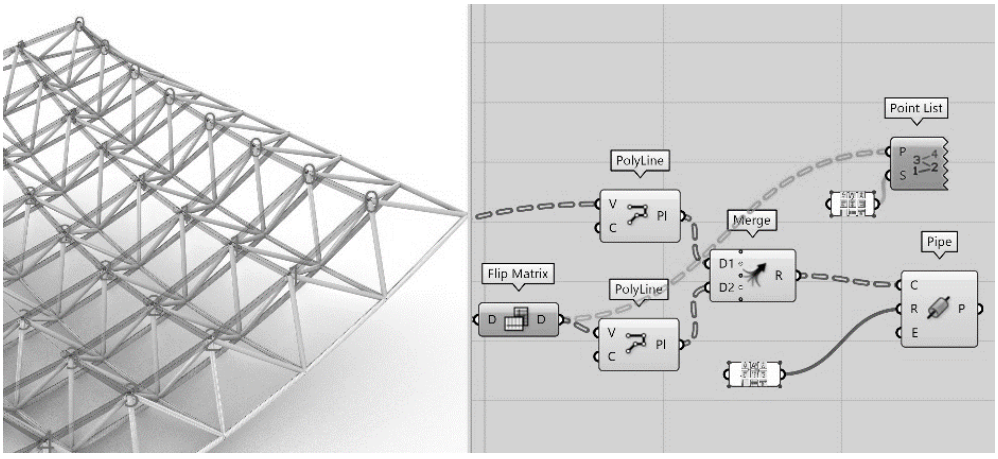


- Flatten Tree membuat semua data pada cabang/path dimasukkan ke cabang utama {0}.
- Akibatnya, setiap titik memiliki nomor indeks sendiri dari nol hingga n dengan urutan seperti pada gambar.

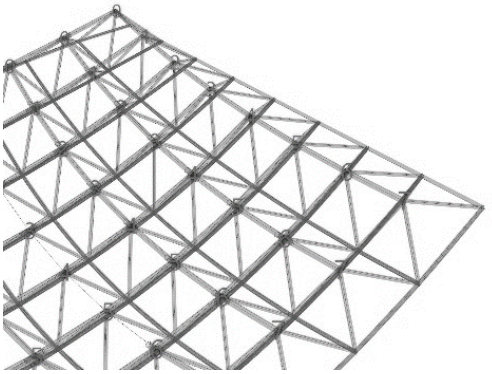
Cara 1

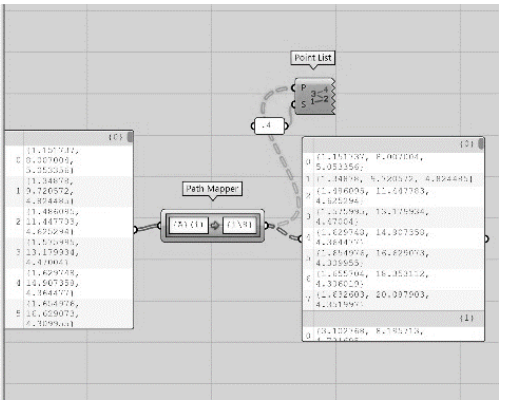


- Gunakan komponen **Partition List** untuk membuat partisi/membuat cabang baru dari setiap n list dalam setiap partisinya. Seperti terlihat di atas, kita menginginkan setiap partisi berisi 8 titik, atau 8 data, maka susunan struktur data berubah (lihat gambar), dimana setiap setiap kolom merupakan satu partisi yang berisi 8 titik/data dengan indeks dari 0-7.
- Dengan demikian, jika kita ingin menghubungan setiap partisi, dimana yang dihubungkan adalah index 0 ke 1, index 1 ke 2 dan seterusnya, kita gunakan komponen **Polyline**.
- Untuk membuat garis yang sebaliknya, maka struktur data tadi harus dibalik/di-flip menggunakan kompnen **Flip Matrix** dimana yang menjadi partisi/path adalah baris dan menjadi data/nomor indeks adalah kolom.



Cara 2





- Cara kedua menggunakan komponen **Path Mapper**. Komponen ini berfungsi mengubah struktur tree (path dan data di dalamnya) melalui aturan Lexicon: penggunaan karakter sebagai variable. PathMapper ini komponen yang penting namun pengoperasiannya cukup rumit. Prinsip kerja dari komponen ini adalah definisikan struktur data eksisting kemudian tentukan transformasi yang diinginkan (melibatkan jumlah path, jumlah data per-path, dan lainnya). Komponen Path Mapper memiliki kapabilitas yang lebih luas dibanding Partition List.
- Pada contoh di atas, struktur data eksisting hanya memiliki satu path dengan semua data: {A}(i), akan diubah dengan cara membuat grup cabang dimana masing-masing cabang memiliki 8 data {i\8}. Grouping adalah salah satu fungsi dari komponen **Path Mapper**.
- Langkah selanjutnya sama dengan setelah anda menggunakan **Partition List**.