Simplicity is Complicated:

On the Balance of Performance and Knobs

Changkun Ou
LMU Munich

IDC 2019 Autumn
Vienna, Austria
Oct. 9, 2019

🇨🇳 Changkun Ou (欧长坤)
🇩🇪 Tschang-Kwën Ou
🇬🇧 / t͡ʂʰɑŋkuən ɤʊ /

https://changkun.de

- PhD student / Prof. Butz
- Expertise: Web Techs , Machine Learning, Distributed Systems

- Contributor of **Tensorflow**, **Go**, **etcd**, **redis**, … (100k+ 🌟 projects)
- Author of C++ and Go books (6k+ 🌟 projects)
- Many many other open source contributions...

Mesh simplification (MS)

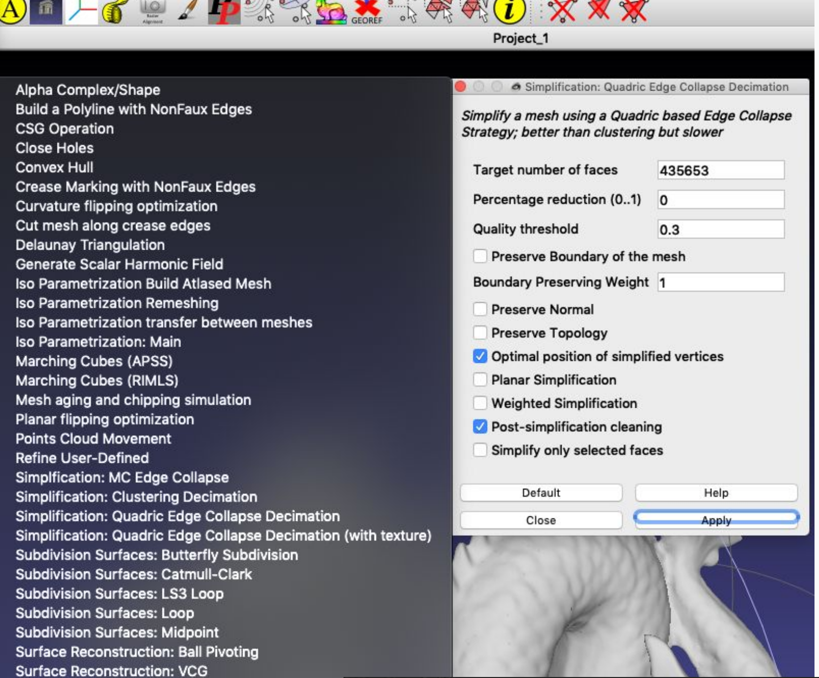
Mesh simplification (MS)

MS is largely applied

Mesh simplification (MS)

MS is largely applied

MS has been "solved" 20 years ago [Hoppe'96] [Garland'97]



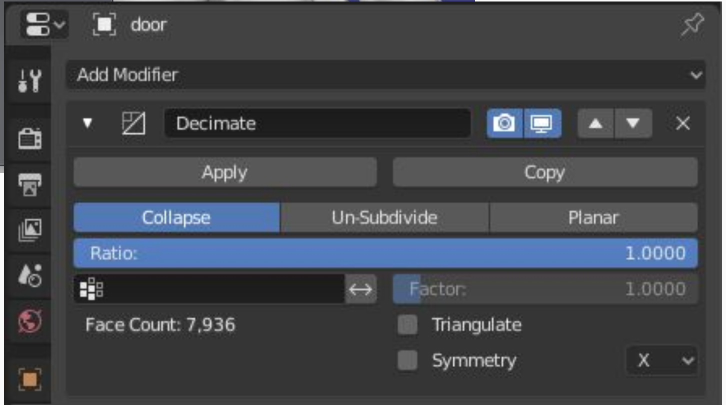
Alpha Complex/Shape
Build a Polyline with NonFaux Edges
CSG Operation
Close Holes
Convex Hull
Crease Marking with NonFaux Edges
Curvature flipping optimization
Cut mesh along crease edges
Delaunay Triangulation
Generate Scalar Harmonic Field
Iso Parametrization Build Atlased Mesh
Iso Parametrization Remeshing
Iso Parametrization transfer between meshes
Iso Parametrization: Main
Marching Cubes (APSS)
Marching Cubes (RIMLS)
Mesh aging and chipping simulation
Planar flipping optimization
Points Cloud Movement
Refine User-Defined
Simplfication: MC Edge Collapse
Simplification: Clustering Decimation
Simplification: Quadric Edge Collapse Decimation
Simplification: Quadric Edge Collapse Decimation (with texture)
Subdivision Surfaces: Butterfly Subdivision
Subdivision Surfaces: Catmull-Clark
Subdivision Surfaces: LS3 Loop
Subdivision Surfaces: Loop
Subdivision Surfaces: Midpoint
Surface Reconstruction: Ball Pivoting
Surface Reconstruction: VCG
Tri to Quad by 4-8 Subdivision
Tri to Quad by smart triangle pairing
Turn into Quad-Dominant mesh
Turn into a Pure-Triangular mesh
Uniform Mesh Resampling
Vertex Attribute Seam
Voronoi Filtering
Screened Poisson Surface Reconstruction

Simplification: Quadric Edge Collapse Decimation

Simplify a mesh using a Quadric based Edge Collapse Strategy; better than clustering but slower

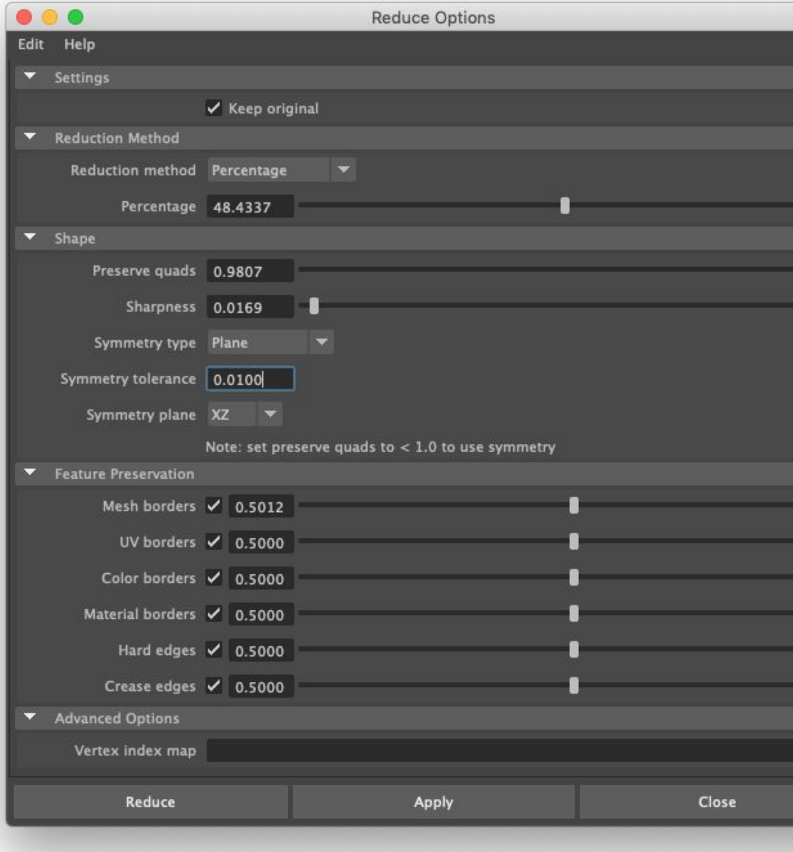
Target number of faces: 435653
Percentage reduction (0..1): 0
Quality threshold: 0.3
□ Preserve Boundary of the mesh
Boundary Preserving Weight: 1
□ Preserve Normal
□ Preserve Topology
☑ Optimal position of simplified vertices
□ Planar Simplification
□ Weighted Simplification
☑ Post-simplification cleaning
□ Simplify only selected faces

MeshLab



Ratio: 1.0000
Factor: 1.0000
Face Count: 7,936
□ Triangulate
□ Symmetry X

Blender



☑ Keep original
Reduction method: Percentage
Percentage: 48.4337
Preserve quads: 0.9807
Sharpness: 0.0169
Symmetry type: Plane
Symmetry tolerance: 0.0100
Symmetry plane: XZ
Note: set preserve quads to < 1.0 to use symmetry
Mesh borders ☑ 0.5012
UV borders ☑ 0.5000
Color borders ☑ 0.5000
Material borders ☑ 0.5000
Hard edges ☑ 0.5000
Crease edges ☑ 0.5000
Vertex index map

Autodesk Maya

Naïve MS

2-Manifold locally resembles 2D Euclidean space.

```
// Polyreduce reduces number of polygons
// while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {

    for !m.Eval(c) {
        local := m.Pick()
        local.Simplify()
    }

}
```

Issues:

1. Non-scalable (serialized process)
2. NP-hard computation
3. ...

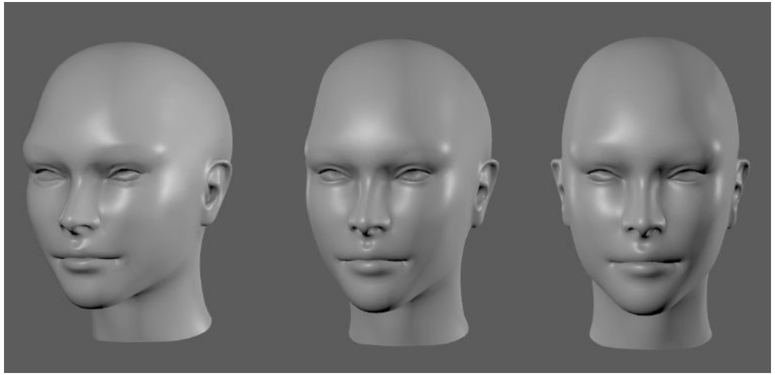
What really matters? Practitioner's Perspective

1. Reduction speed → **Computation complexity**

What really matters? Practitioner's Perspective

1. Reduction speed → **Computation complexity**
2. Expert preference → **Reduction quality**
3. Manual intervention → **Automation level**

Can you tell the difference?

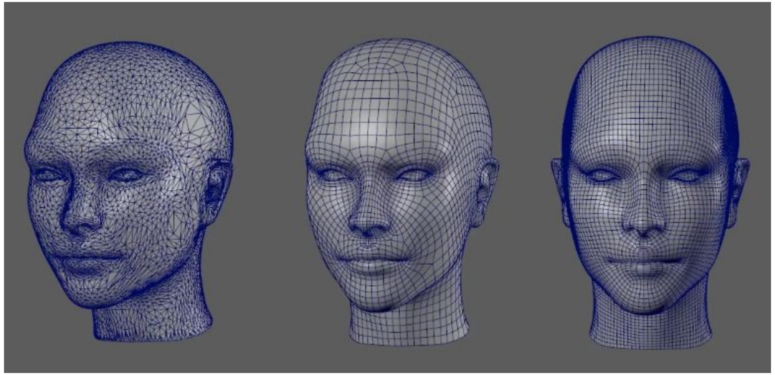


🙂 🙂 🙂

What really matters? Practitioner's Perspective

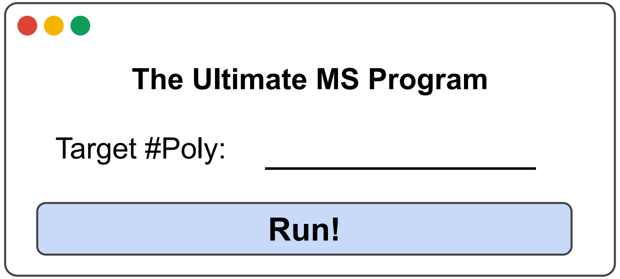
1. Reduction speed → **Computation complexity**
2. Expert preference → **Reduction quality**
3. Manual intervention → **Automation level**

Can you tell the difference?

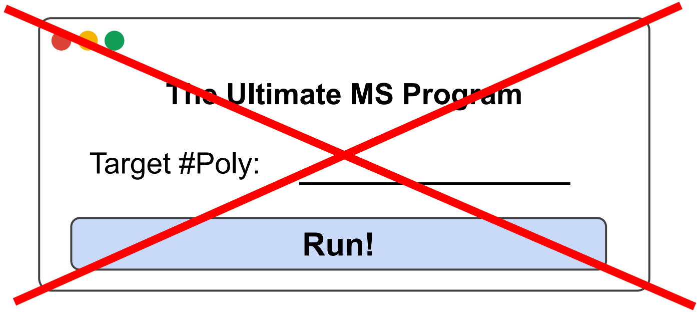


😡 🙂 😡

The Ultimate MS Program



The Ultimate MS Program



Impractical

1. Reduction speed → Computation complexity

   Solution: Thread multiplexing 🔥🔥🔥

2. Expert preference → Reduction quality
3. Manual intervention → Automation level

   Solution: Hyperparameter reduction & Imitation learning 😎😎😎

Computation Complexity:

From Parallelism to Concurrency

Concurrent MS

```
// Polyreduce reduces number of polygons
// while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {

      for !m.Eval(c) {
            local := m.Pick()
            local.Simplify()
      }

}
```

Inspiration

```
// SGD implements mini-batch
// stochastic gradient descent.
func SGD(m *Model, d *Dataset) {

      for !m.Converge() {
            miniB:= d.Batch()
            m.GradientDescent(miniB)
      }

}
```

Concurrent MS

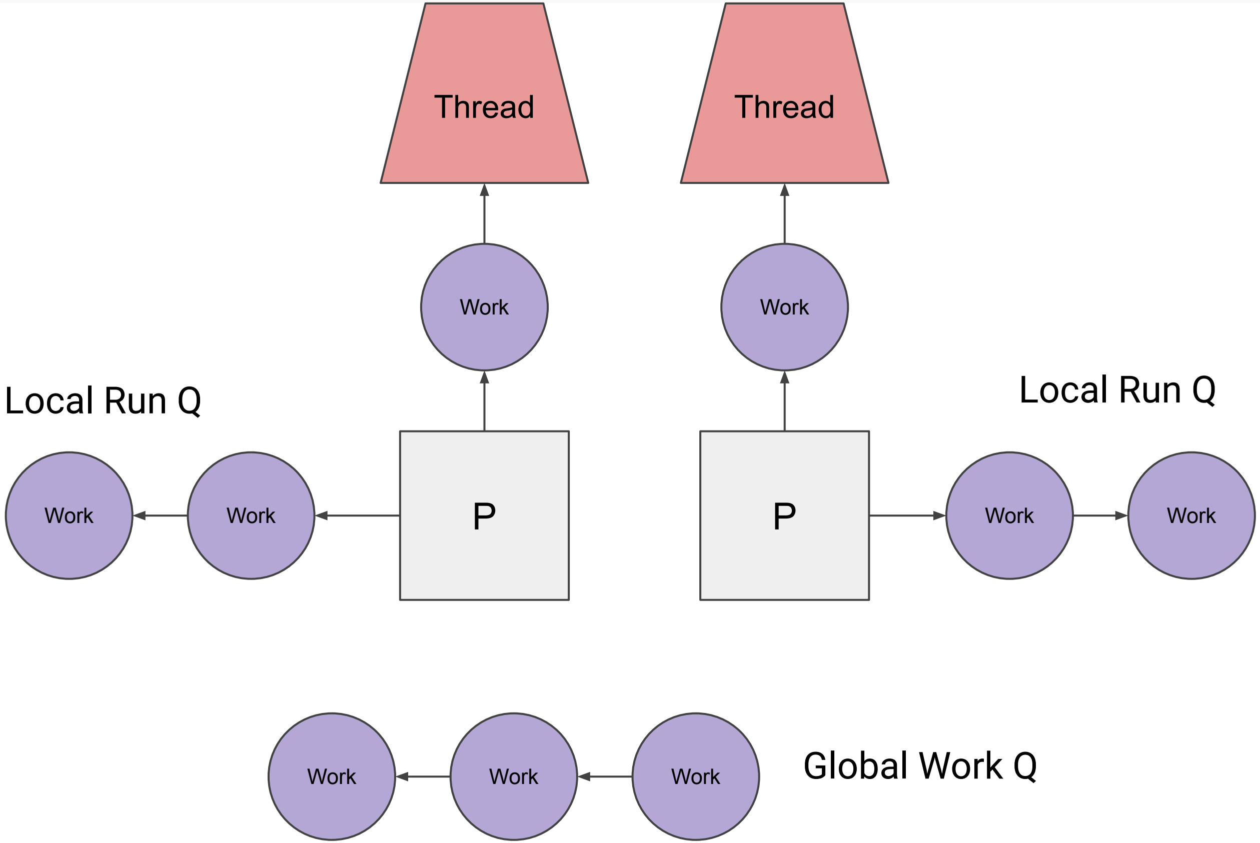
```go
// Polyreduce reduces number of polygons
// while preserving local topologies.
func Polyreduce(m *Mesh, c *Criteria) {

      for !m.Eval(c) {
            local := m.Pick()
            local.Simplify()
      }

}
```

Inspiration

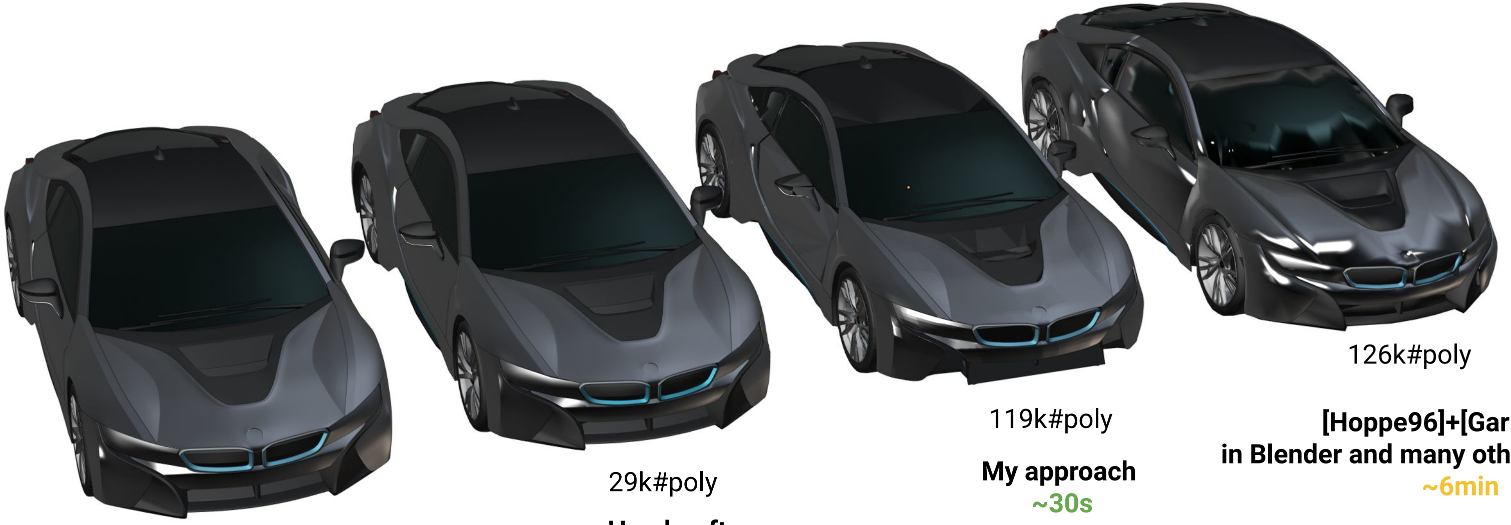
```go
// SGD implements mini-batch
// stochastic gradient descent.
func SGD(m *Model, d *Dataset) {

      for !m.Converge() {
            miniB:= d.Batch()
            m.GradientDescent(miniB)
      }

}
```

```go
func Polyreduce(m *Mesh, c *Criteria) {

      // SPEEDUP1: build workQ concurrently
      for local := m.Pick(); local != nil; {
            sched.Submit(func() {
                  quality, ok := local.Eval(c)
                  if ok {
                        workQ.Push(quality, local)
                  }
            })
            local = m.Pick()
      }
      sched.Wait()                 // sync barrier

      // SPEEDUP2: run workQ concurrently
      for w := workQ.Pop(); w != nil; {
            sched.Submit(w.Simplify)
            w = workQ.Pop()
      }
      sched.Wait()                 // sync barrier
}

var sched Sched // M:N work-steal scheduling
func (s *Sched) Submit(f func()) { … }
```

Thread Multiplexing: Work-steal scheduling



We are not even close



640k#poly

Ground truth

29k#poly

Handcraft
40 hours+

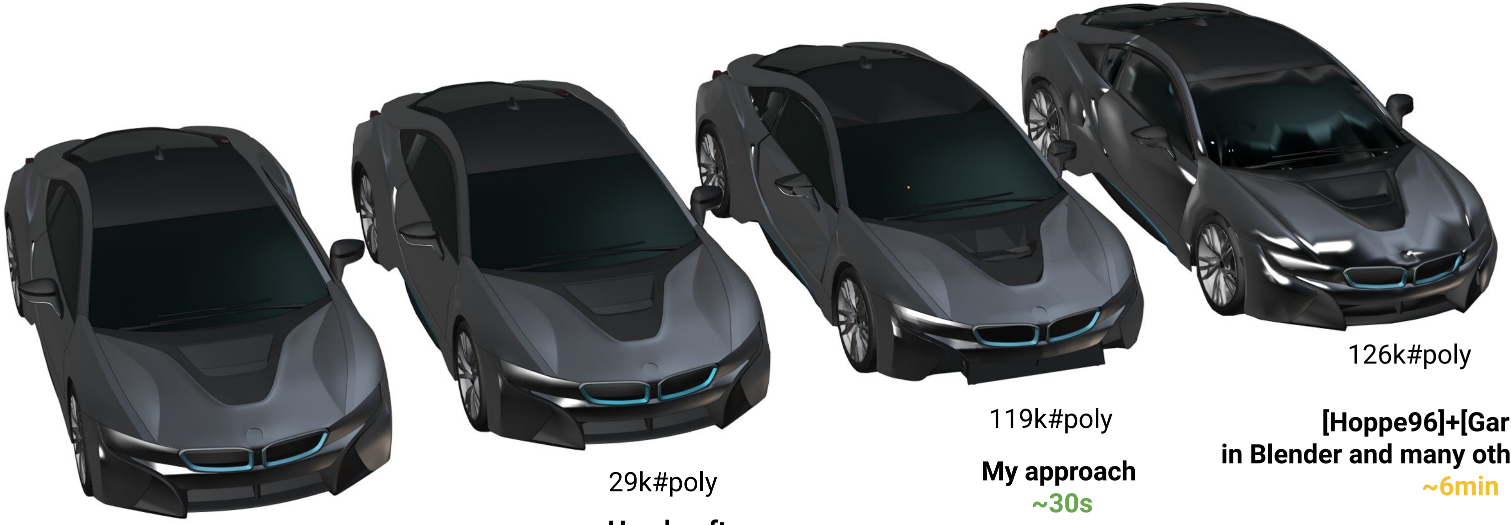
119k#poly

My approach
~30s

126k#poly

[Hoppe96]+[Garland97]
in Blender and many other 3D softwares
~6min

We are not even close



640k#poly

Ground truth

29k#poly

Handcraft
40 hours+

119k#poly

My approach
~30s

126k#poly

[Hoppe96]+[Garland97]
in Blender and many other 3D softwares
~6min

"MS is a solved problem!"

Quality & Automation:

From Fully Automatic to Semi Automatic

No Free Lunch [Wolpert'96, Wolpert and Macready'97]

$$\sum_{\text{model}} \text{cost}(\text{model}, \text{method}) = \text{const.}$$

Modeling Human Cost in MS

$$T = [\ MS + DT(p)\ ] * A(p)$$

If p1 > p2 :

$$T1 - T2 = MS * [\ A(p1) - A(p2)\ ] + [\ DT(p1) * A(p1) - DT(p2) * A(p2)\ ]$$

Modeling Human Cost in MS

$$T = [\ MS + DT(p)\ ] * A(p)$$

If p1 > p2 :

$$\begin{aligned} T1 - T2 &= MS * [\ A(p1) - A(p2)\ ] + [\ DT(p1) * A(p1) - DT(p2) * A(p2)\ ] \\ &> MS * [\ A(p1) - A(p2)\ ] + [\ DT(p2) * A(p1) - DT(p2) * A(p2)\ ] \end{aligned}$$

Hick's Law

Modeling Human Cost in MS

$$T = [\ MS + DT(p)\ ] * A(p)$$

If $p1 > p2$:

$$\begin{aligned} \mathbf{T1 - T2} &= MS * [\ A(p1) - A(p2)\ ] + [\ DT(p1) * A(p1) - DT(p2) * A(p2)\ ] \\ &> MS * [\ A(p1) - A(p2)\ ] + [\ DT(p2) * A(p1) - DT(p2) * A(p2)\ ] \\ &= [\ MS + DT(p2)\ ] * [\ A(p1) - A(p2)\ ] \end{aligned}$$

Hick's Law

⇒ **Reduce attempts**

Two Groups of Web APIs (*talk to me for beta access*)

ConcurrentMS

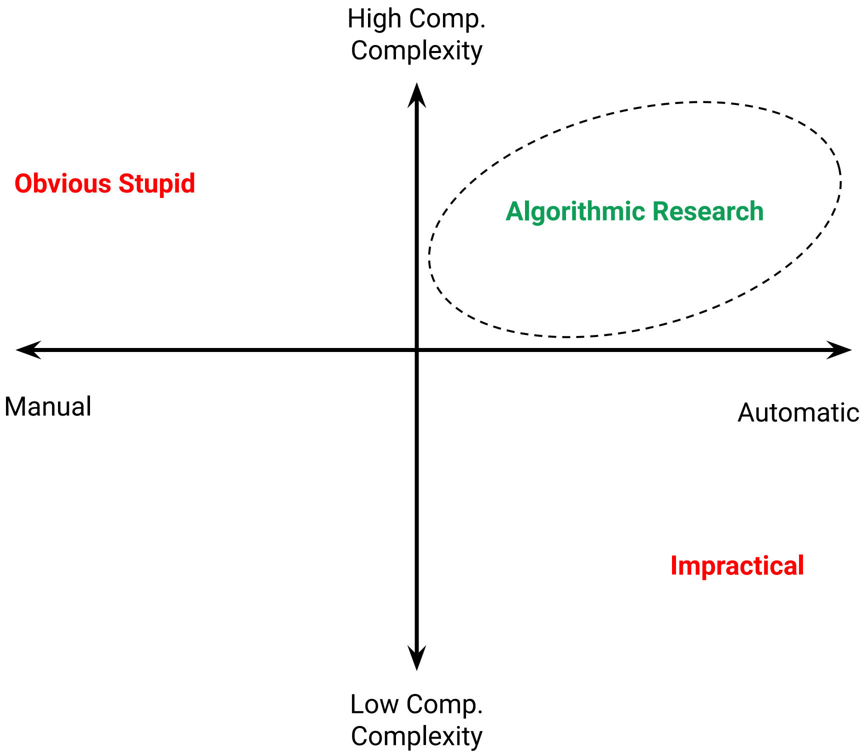
```
type PolyReduce interface {
    Upload(m *Model) (OpID string)
    Upload(OpID string, c *Config)
    Run(OpID string)                // 1m #poly ≈ 1 min → 1 model
    Download(OpID string) (m *Model)
}
```

↓

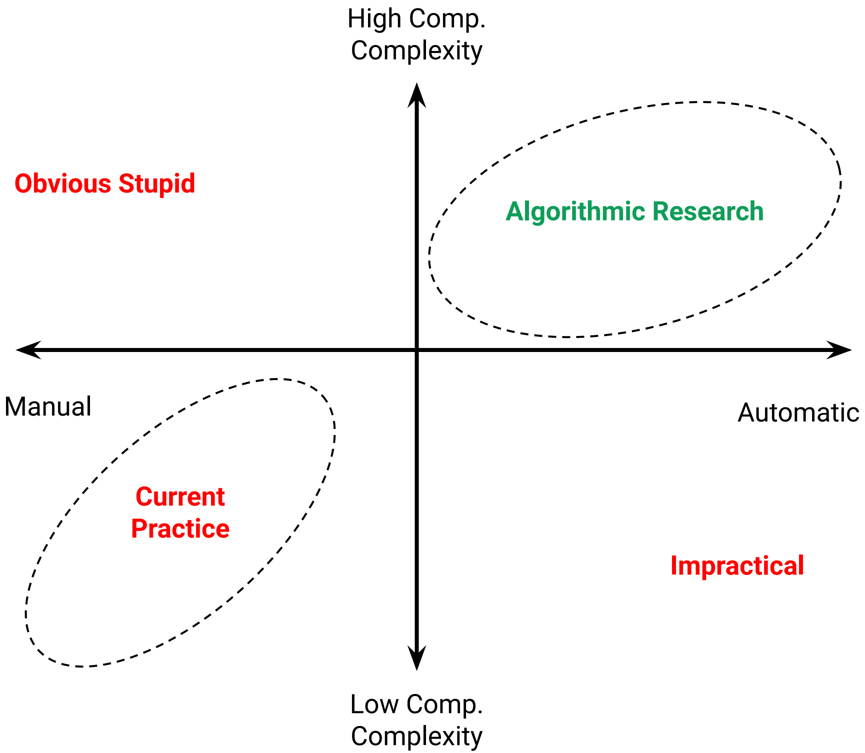
Available, but untested

```
type ProPolyReduce interface {
    Upload(m *Model) (SessID string)
    Run(SessID string) (OpIDs []OpID) // 1m #poly ≈ 2 min → 4 models
    Eval(OpIDs []OpID, Scores []int)
    Download(OpID string) (m *Model)
}
```

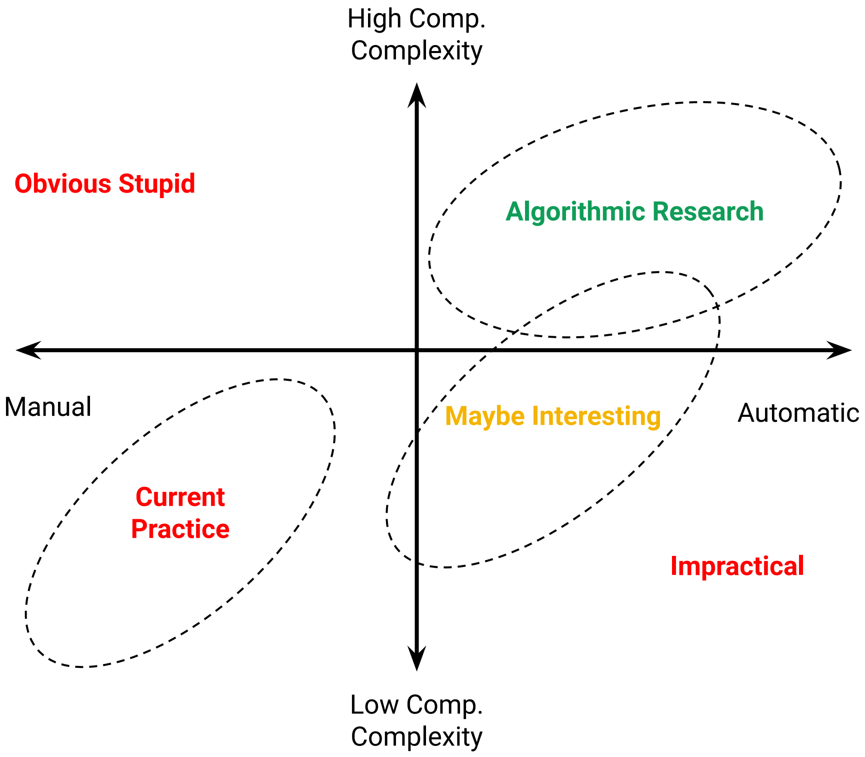
Automation, Speed, Quality Tradeoff



Automation, Speed, Quality Tradeoff



Automation, Speed, Quality Tradeoff



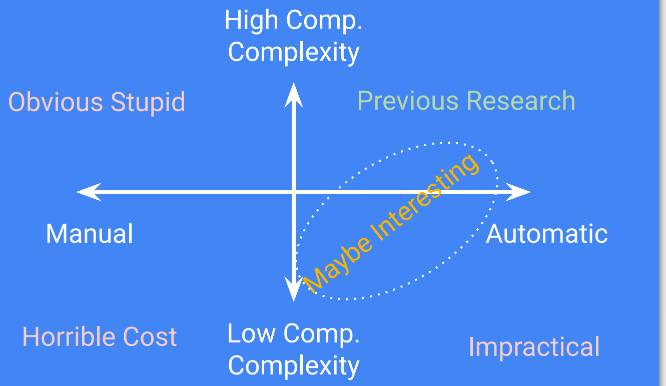
Bernhard Riemann (1826-1866)

Simplicity is Complicated:
On the Balance of Performance and Knobs

Changkun Ou
LMU Munich

IDC 2019 Autumn
Vienna, Austria
Oct. 9, 2019

"The performance improvement does not materialize from the air, it comes with code complexity increase."



Random

Why MS is a well-studied problem to you?
Or, why it isn't?

When did you start thinking about using MS?
What are your expectations from MS? Why?
How did you evaluate MS outcomes in your 3D projects?

What are principles qualifying 3D Artists v.s. Non-3D Artists?
Is it quantitative measurable? Why?

When did industrial MS fail to your case? Why?

What is your maximum tolerance of X to MS? Why?
where X = speed, features preservation, and …

How could human solve retopology so incredible?
What did we miss in MS?

Any thing in mind?