Representations of geometric objects:

- There are a variety of options; each supports some queries well, others poorly; not aware of anything that does everything well.

- Considerations
  - Should the model be a solid?
  - Do you care about boundaries (eg. curvature, normals, contact, cutting, etc)
    or volumes? (eg. being a solid, path planning, some kinds of collision)
  - Will you intersect, union, etc objects?
  - At what resolution do you wish to rep.? will that change?
  - Will there be <u>interactive</u> modelling?
  - Are you going to manufacture?
  - Do you care to model physical properties? (stiffness etc)

Some <u>Volume</u> options

- <u>Voxels</u>
  - insert a grid in space, label 1 if occupied else zero

  + Powerful; flexible; volume easy to estimate; ignores detail below some scale

  - ignores detail below some scale; can be wasteful of space; can give poor repn's of some objects.

  + fast intersection, union; models are solids; relatively easy to render; ray carving algorithms allow some easy modelling of free space

- <u>Octrees</u>

  + more space-efficient than voxels

  - savings can be hard to get.

<u>CSG</u> (constructive solid geometry).

- Select primitives, usually simple (often:

  always
  - sphere
  - infinite cylinder
  - half-space.

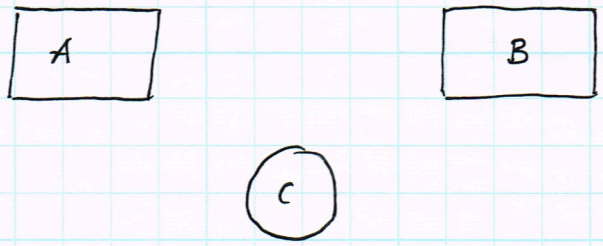
  sometimes
  - ∞ cylinder w/ given x-section (extruded curve)
  - volume created by spinning curve around axis.

- Now object is logical combination of primitives

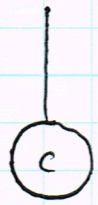
  — <u>operators</u>

  - rotate, translate, scale
  - $\cap$, $\cup$, $-$ cartesian subtraction

- <u>Notice</u> · these operators need to be stabilized.



$$\left(\text{translate}(A) \cap B\right) \cup \text{translate}(C)$$



← in the worst case, this could be a sphere w/ a whisker!

<u>so</u> define

∂ ← takes boundary of set

i ← takes interior

c ← takes closure.

<u>then</u>: $A \cap^* B = c\left(i(B) \cap i(A)\right)$

gets rid of this problem

$$A \cup^* B = c\left(i(A) \cup i(B)\right)$$

etc.

CSG has some advantages

- w/ regularized operators objects are solids
- model contains a coherent, slightly semantic record of how it was made.
- Good choices of primitives lead to
  - easy alg's for
    - ray tracing
    - intersection
    - union
    - volume computation

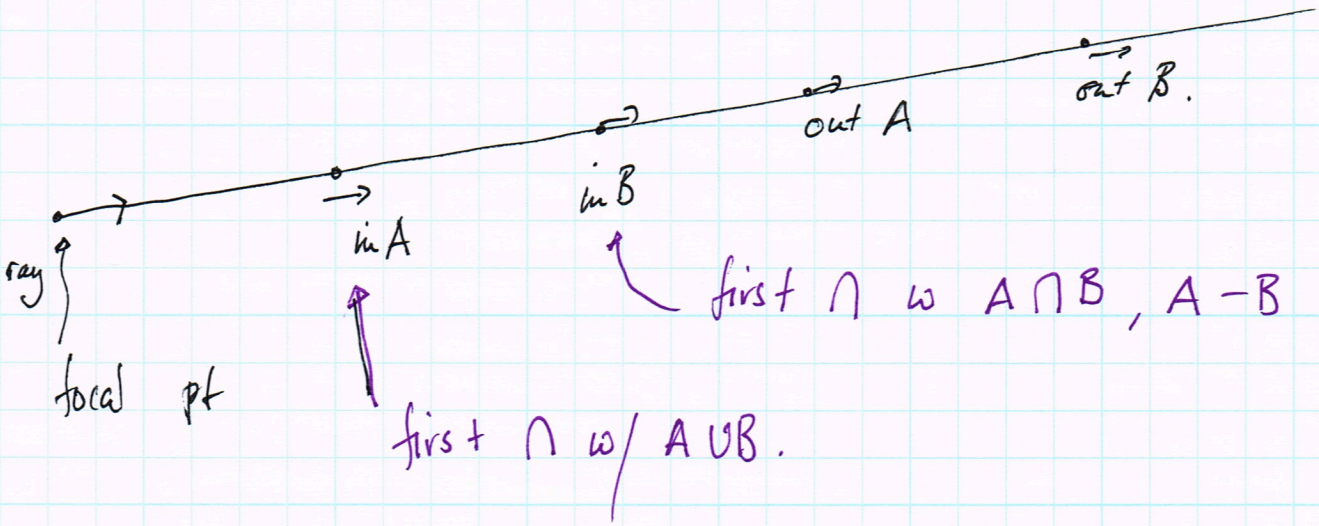
But

- v hard to tell if two models are the same.
- v. hard to get to CSG from any other repn

Raytracing CSG

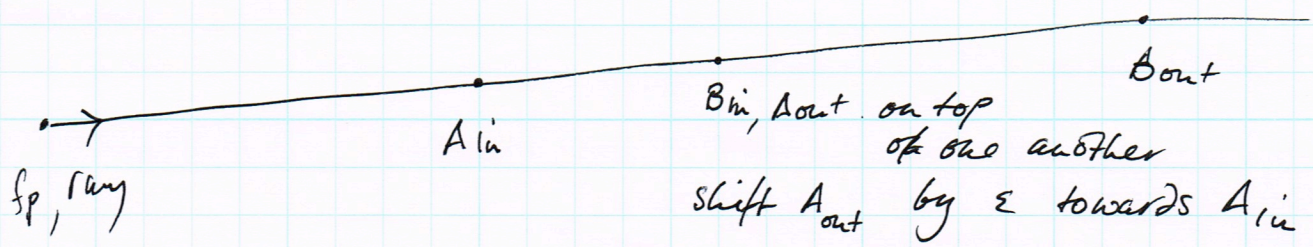
- assume primitives admit exact intersections w/ rays (rootfinder).
- Now 1 ray w prims, reason about first intersection pt.

e.g. op (A , B)



<u>notice</u> regularizing operator can be achieved by allowing pts to "cancel"

e.g. $A \cap^{*} B$



shift A_{out} by ε towards A_{in}

B_{in} by ε towards B_{out}

→ No ∩

<u>OR</u> B_{in}, A_{out} enters + leaves $A \cap B$ at same location

— cancel, ray doesn't enter

Scan to voxels/octree

- easy, if primitives are convex
- keep track of fact there are 3 types of box
  - → all verts in = inside
  - → otherwise — boundary.
  - → all verts out = outside
- I know no alg to go from boxes to CSG better than

  $\cup$ (all boxes)

  (which gives bad CSG!)

BIG ISSUE FOR VISION

— · Many ways to represent the same thing w/ CSG

+ (maybe) · Nice story about increasing detail.

Other Solid reps:

- for $\phi(\underline{x})$ a continuous fn,
  $\phi(x) \geq 0$ gives a solid
- generally hard to work with in this generality
- Algebraic
  - ϕ polynomial.
  - can use this as CSG primitive

  BUT
  - raytracing is tricky
    - exact roots of polynomial
    - use Sturm sequence + deflation,

    BUT
  - telling whether a voxel lies in or out is tricky in all but simplest cases, EVEN 2D

Sketch of Sturm Sequences.

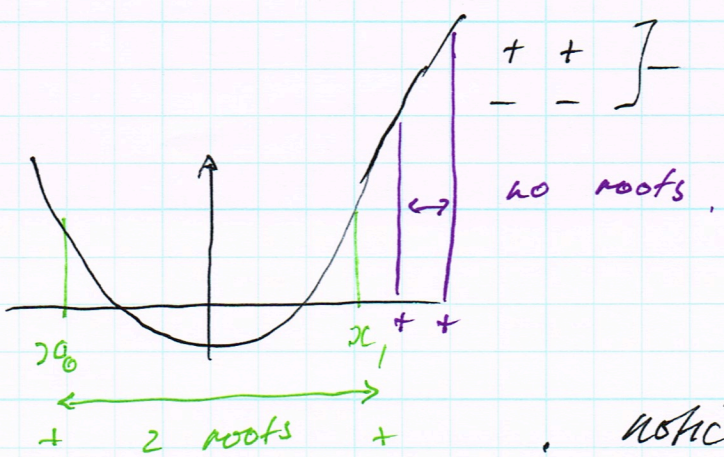
- I wish to know if a polynomial $p(x)$ of ONE var, x, has a root in the interval $[x_0 \cdot x_1]$

- if $p(x)$ is linear, easy — test signs at either end of interval.

- <u>quadratic case</u>

  $+ \quad -$ → guarantees 1 root

  $+ \quad +$
  $- \quad -$] no guarantees!



- notice, if you see $+, +$, to have 2 roots, the derivative <u>must</u> have a zero

- So $\quad + \quad + \leftarrow$ poly
  $\quad + \quad + \leftarrow$ deriv

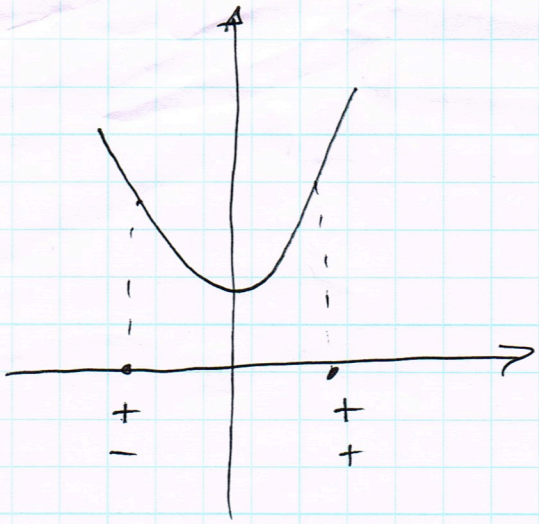
  means <u>no</u> roots.

BUT

+	+	– poly
–	+	– deriv

doesn't guarantee roots

eg



· need another term

Now consider $p = ax^2 + bx + c$

$$-\operatorname{rem}(p, p_x) = b^2 - 4ac = \cancel{\ }$$

+	+	← p
–	+	← p_x
+	+	← rem

$\longrightarrow$ 2 roots

= # of sign changes at one end

– # at other

+	+
–	+
–	–

$\longrightarrow$ 0 roots

"

+	+
–	+
0	0

$\longrightarrow$ 1 root.

"

general construction

$$p_1 = p$$

$$p_2 = p_x$$

$$p_n = -\mathrm{rem}(p_{n-1}, p_{n-2})$$

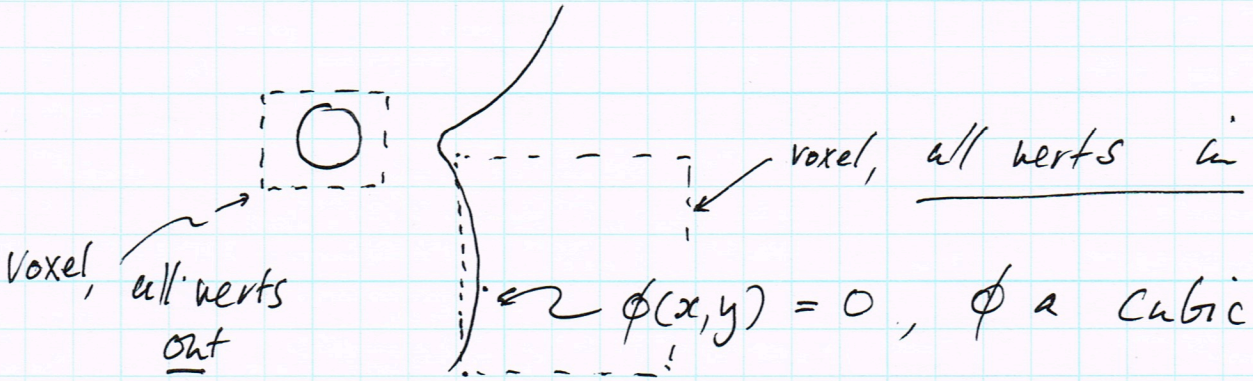
$$\vdots$$

all the way to a constant

- # of sign changes at one end − # at other = # roots.

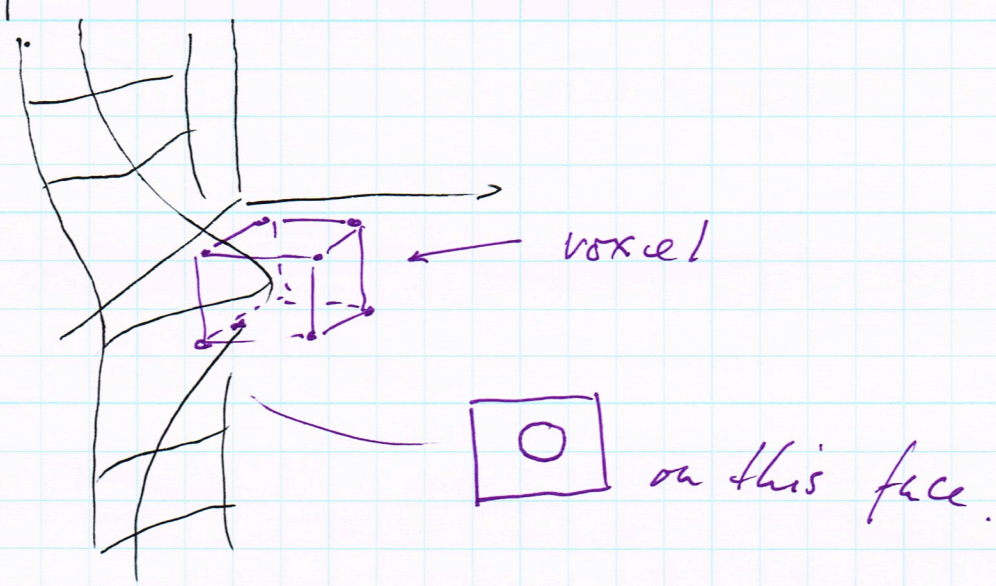
NOTICE THE PROBLEM:

coefficients of p_n are high degree in coeffs of p_1



example, in 2D

- fairly obvious algorithm may suggest itself
  - check
    - (a) all verts in/out
    - (b) no $\wedge$ ⁿ faces.
- this is OK, not great in 2D.
- serious probs in 3D



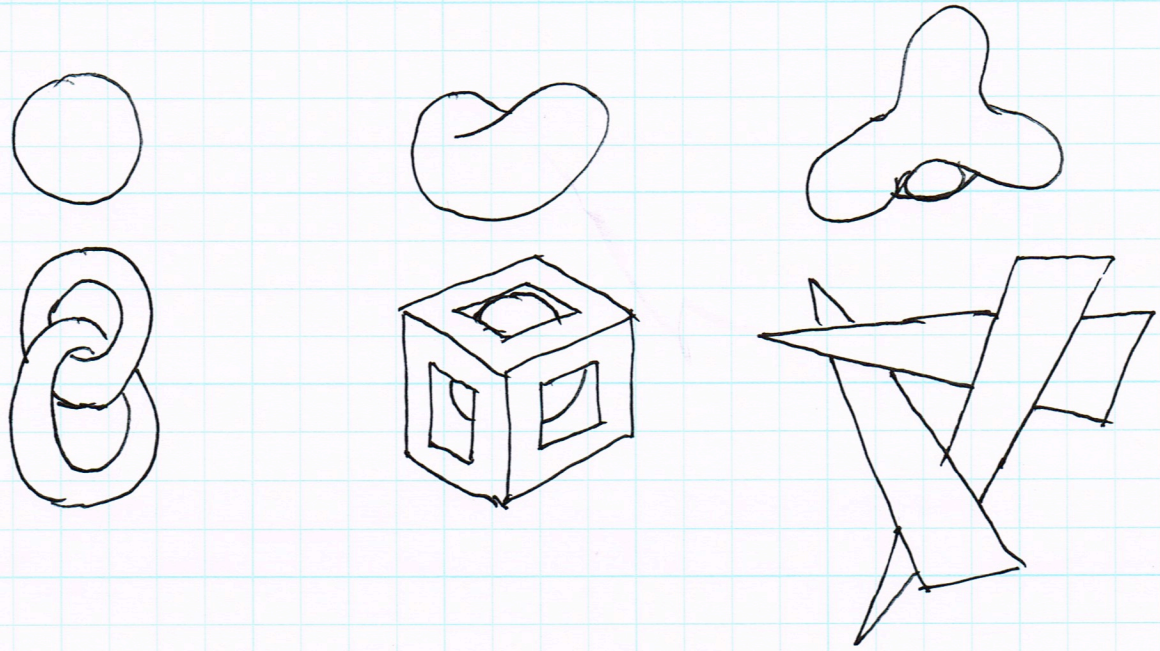
REQUIRED

test that, in some block, a polynomial has no zeros

This test, a <u>multivariate Sturm sequence</u> exists, but is of no practical use in any case of interest

Shape from contour

- Intriguing, classic, vision qu.
- I supply an object contour
  - in various forms:
    - ~~outline~~ silhouette only
    - sil + internal boundaries
    - + creases ?
- Produce a reasonable estimate of shape.
- Hard, pretty obviously
- People do this quite well, though not clear how one measures this precisely



Some strategies from literature

- Assume view is orthograpic (universal)
- Assume surface is SOR, and axis { is parallel to / is at known angle to } image plane.
- Assume SHGC, x-section observed, axis as above

→ doesn't happen often – v. special cases.

- Build a ~~long sur~~ height map
  - ⓐ issue; reconstruction is not height map at contour
  - ⓑ issue; what is height of contour

(a)

$(x, y, z(x,y))$

(10c)

now, notice at outline points, <u>normal</u> is ~~(0,0,1)~~ $(\alpha, \beta, 0)$

<u>BUT</u> for $(x, y, z(x,y))$, normal is

$$\frac{(-z_x, -z_y, 1)}{\sqrt{1 + z_x^2 + z_y^2}}$$

(i.e. at contour, $z_x, z_y \infty$.)

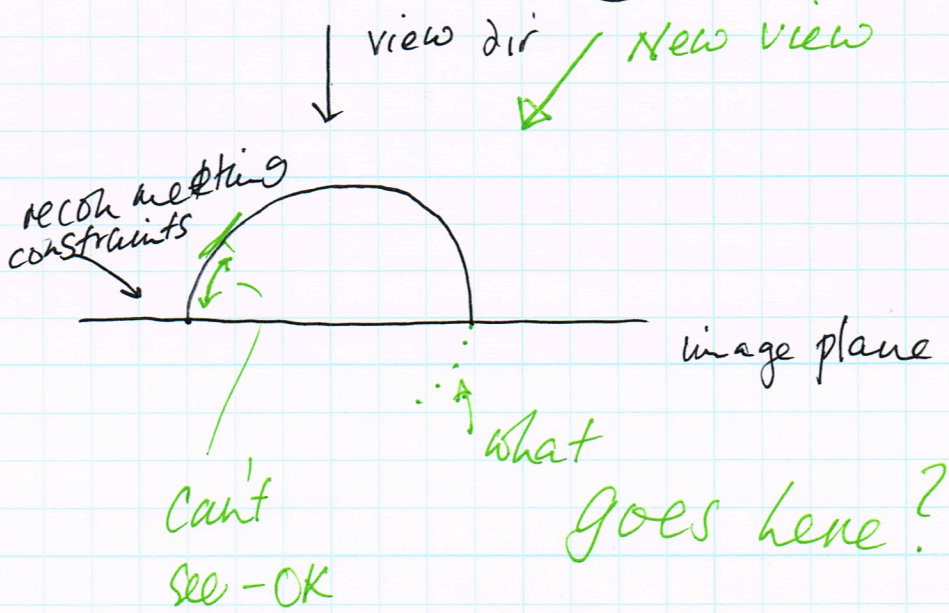
So height map doesn't work.

Strategy 1:

"ignore" this issue,

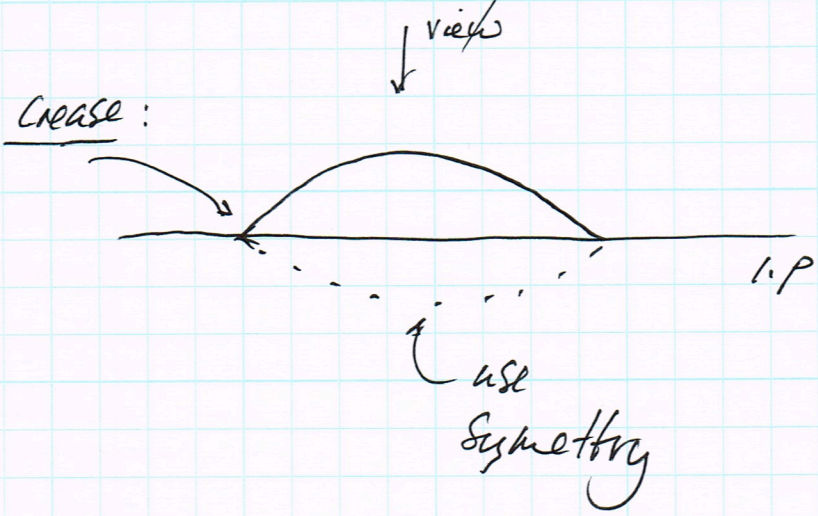
argue that we do not <u>WANT</u> this constraint to be met.

- e.g. for reconstruction, may view recon at <u>new</u> angle – what texture, geometry should we use?



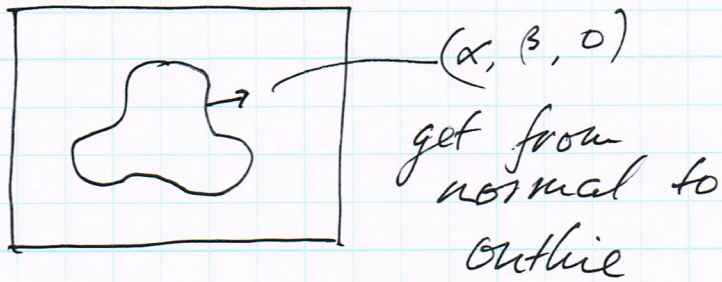
Instead:

recon w/ crease at boundary



This can be achieved by.

- assume C.G. is plane (wrong, but ...)
- now est normal along outline $(\alpha, \beta, 0)$



- Now use POISSON smoothing to get α, β in interior

(i.e solve $\nabla^2 \alpha = 0$ in R

s.t. α = meas on ∂R)

etc.

- now assume $\dfrac{1}{\sqrt{1+p^2+q^2}} \geq \omega$

- get unit normal field by.

  - α, β from above

  - $U.N. = \left[\alpha, \beta, \sqrt{1-\alpha^2-\beta^2}\right]$

    OK, by POISSON smoothing

    $= (n_0, n_1, n_2)$

  - <u>MOD</u> $U.N = \begin{cases} (n_0, n_1, n_2) & \text{if } n_2 \geq \omega \\ \cancel{(n_0(1-\omega^2), \omega)} & \\ (n_0(1-\omega^2-n_2^2), n_1(1-\omega^2-n_2^2), \omega) & \text{otherwise} \end{cases}$

    $= (n_0^*, n_1^*, n_2^*)$

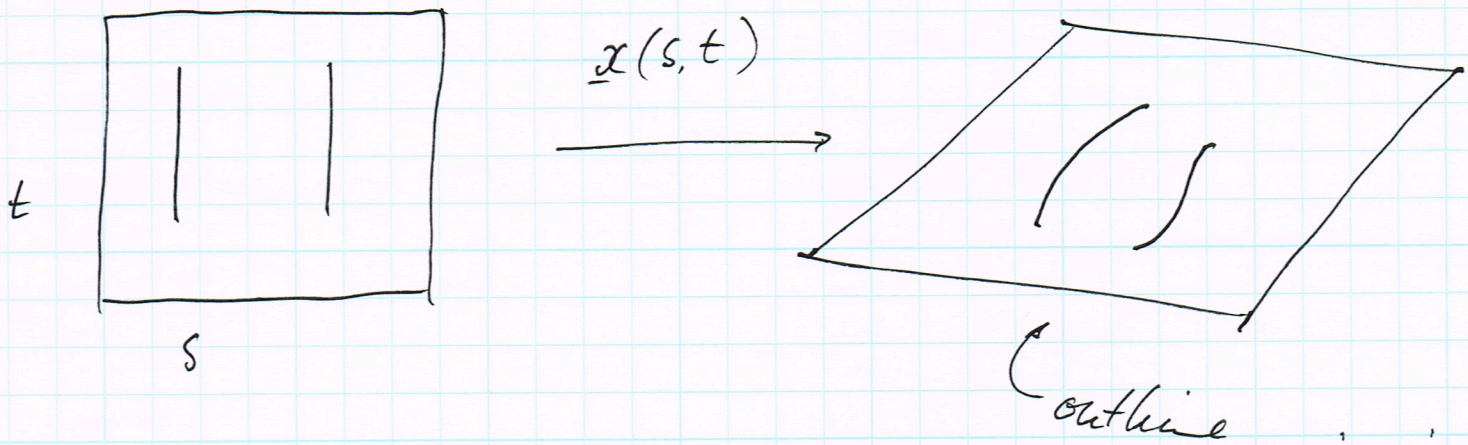
- Then solve by integration.

$$\min_{h} \left(\left(h_x + \frac{n_0^*}{n_2^*} \right)^2 + \left(h_y + \frac{n_1^*}{n_2^*} \right)^2 \right)$$

s.t. $h = 0$ on boundary

- Variants are likely possible, but this is quite good.

. Alternative (Prasad + Zisserman)



<u>Choose</u> curves in s,t to csp to outlines (eg, $t=\text{const}$); csp between curves

<u>choose</u> cost fn. eg.

minimize <u>linearized</u> bending energy

$$\int \left(x_{ss}\right)^2 + 2\left(x_{st}\right)^2 + \left(x_{tt}\right)^2 \, ds\, dt$$

<u>then</u> we get <u>linear</u> constraints from outline

$$\underline{x}(s,t) = (x_0(s,t),\ x_1(s,t),\ x_2(s,t))$$

then

$$x_0(s, \text{const}) = \text{outline } x(s)$$
$$x_1(s, \text{const}) = \text{outline } y(s)$$

on outline points

$$n(s) \cdot x_s(s, \text{const}) = 0$$

known

$$n(s) \cdot x_t(s, \text{const}) = 0$$

$\therefore$ ~~Grad~~ Grid up (s,t); consider

$x_0(s,t)$, $x_1(s,t)$, $x_2(s,t)$ on grid

Becomes quadratic problem in node values w/ linear constraints

$\longrightarrow$ <u>easy</u>

Super Quadrics

$$\left(\frac{x}{A}\right)^r + \left(\frac{y}{B}\right)^s + \left(\frac{z}{C}\right)^t = 1$$

where r, s, t are not necessarily integer

ADV: blobby objects, can make sharp corners

DISADV: most ops harder than for algebraic (that fractional power).

Metaballs:

- define some reference objects (pts, lines, etc). o_i
- define a distance cost fn. $f(r)$

  eg $f(r) = (1 - r^2)^2$

- Volume $= \sum_i f(\|x \to o_i\|)$ $\lessgtr$ threshold.

ADV:

- rich families of quite complex shapes
- seems like an obvious application to shape from contour (cf Adelson's puffballs, below)
-

DISADV: mechanics, rendering, etc, fairly complex, comparable to low degree algebraic case

PUFFBALLS

- See paper.