3D Shape Representations I:

Implicit Functions, Parametric Reps, and Fitting

Richard (Hao) Zhang

CMPT 464/764: Geometric Modeling in Computer Graphics

Lecture 3

Outline

- Implicit reps
- Parametric reps

} **Smooth** curves and surfaces

Outline

- Implicit reps
- Parametric reps
- Meshes (subdivision)
- Point clouds
- Voxels

Implicit reps, Parametric reps → Smooth curves and surfaces

Meshes (subdivision), Point clouds, Voxels → **Discrete** representations

Outline

- Implicit reps
- Parametric reps

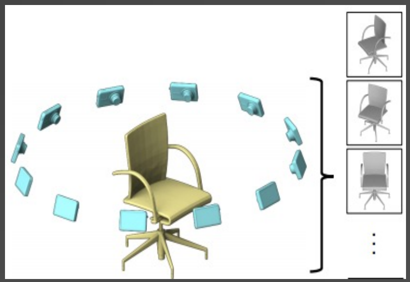
Smooth curves and surfaces

- Meshes (subdivision)
- Point clouds
- Voxels

Discrete representations

- Projective reps (multi-view)

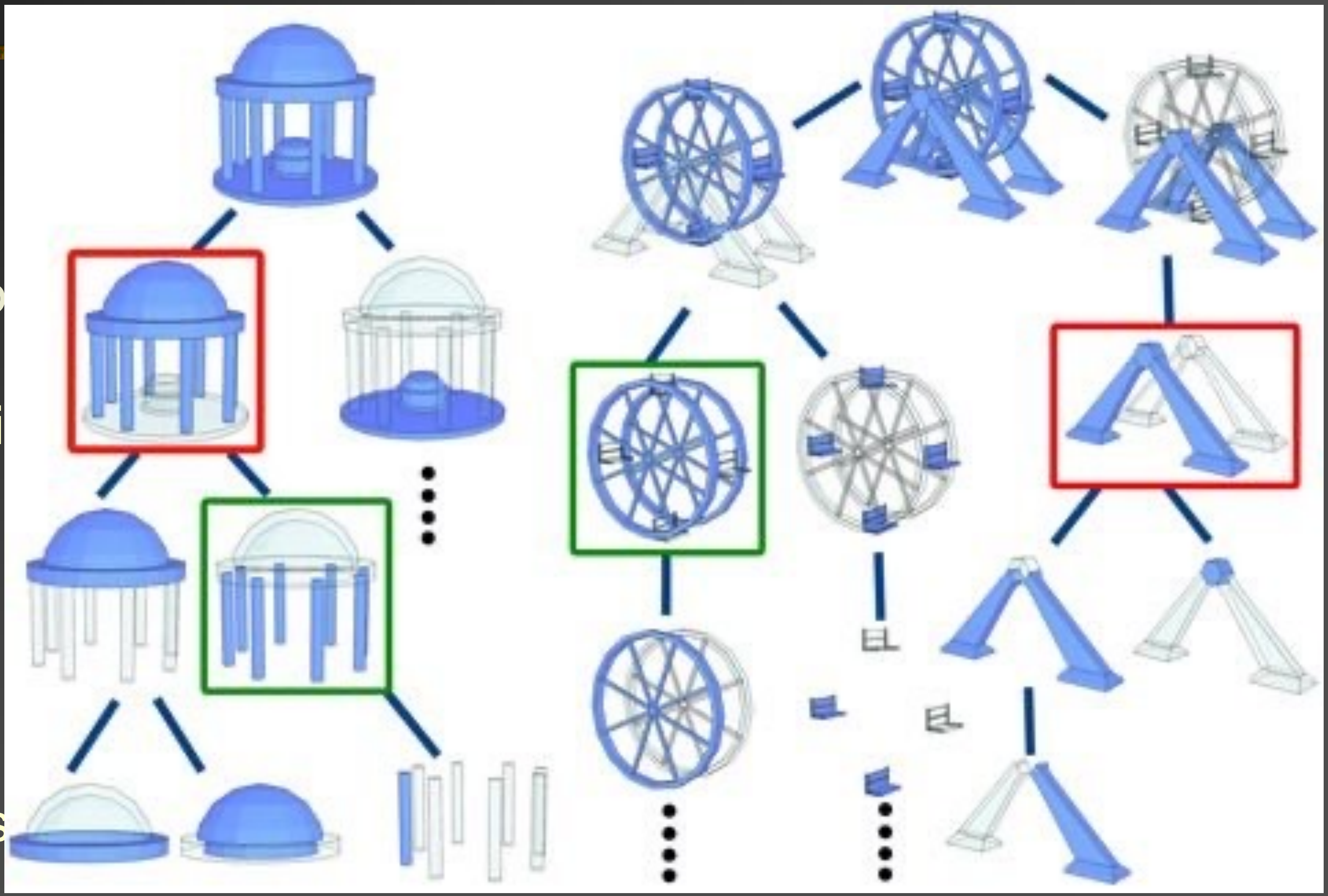
3D → 2D



Outline

Symmetry hierarchies

- Implicit reps
- Parametric rep
- Meshes (subdi
- Point clouds
- Volumes
- Projective reps
- Structured reps



Parts + relations = structures
Encompasses all low-level reps

Today

- Implicit reps
- Parametric reps

Smooth curves and surfaces

- Meshes (subdivision)
- Point clouds
- Volumes

Discrete representations

- Projective reps

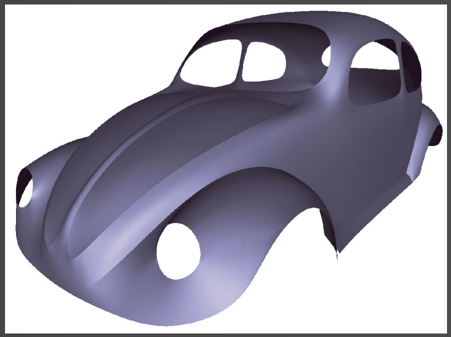
3D → 2D

- Structured reps

Parts + relations = structures
Encompasses all low-level reps

Why smooth curves & surfaces?

- Naturally, to model smooth shapes, e.g.,
  - Body of an automobile
  - Shape of cartoon characters (Shrek)
  - Motion curves in animation



[Zorin 01]

- Compact, analytical representation
- Smoothness can often be guaranteed analytically
- Theory of smooth curves and surfaces is well-developed

Why smooth curves & surfaces?

- Study of smooth curves and surfaces
  - e.g., the notion of arc length, area, curvature, surface normal, tangents, parameterization, etc.

  forms the basis behind geometric modeling and processing using other primitives,
  - e.g., polygonal meshes, subdivision surfaces, point clouds
- A lot of work in discrete shape processing involves

  discretization of the theory for the continuous and the smooth

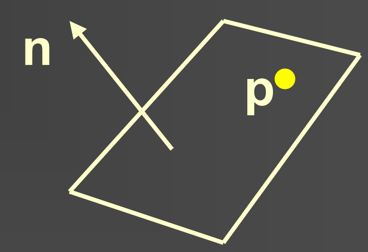
1. Implicit function representations

$$\text{Shape} = \{\mathbf{x} \in \mathbf{R}^k \mid f(\mathbf{x}) = 0\},$$

e.g., for a plane, $f(\mathbf{x}) = \mathbf{n} \bullet (\mathbf{x} - \mathbf{p})$, and

for a sphere, $f(\mathbf{x}) = (\mathbf{x} - \mathbf{c})^2 - r^2$,

- f: **inside-outside function**
  - **x** is inside the shape, if $f(\mathbf{x}) < 0$
  - **x** is outside the shape, if $f(\mathbf{x}) > 0$

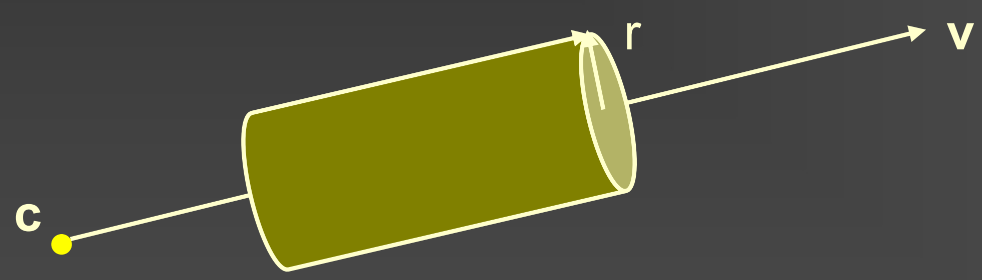


For this to work efficiently, f should be easy to evaluate

Exercise: cylinder primitive

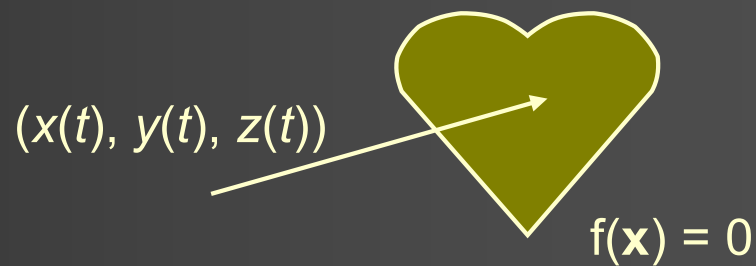
Shape = $\{\mathbf{x} \in \mathbf{R}^k \mid f(\mathbf{x}) = 0\}$,

What is $f(\mathbf{x})$ for a bi-infinite (unbounded) cylinder with center $\mathbf{c}$, orientation vector $\mathbf{v}$, and base radius r?

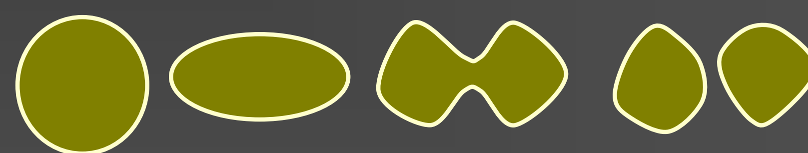


Where are implicit reps used?

- Bresenham line drawing algorithm
- Intersections tests in ray tracing or **collision detection**
- Intermediate representation in surface reconstruction
- *** Evolve a surface by evolving its 3D scalar field, governed by a **level-set** — topology changes automatic



Conversion between implicit and parametric is not always easy



Evolution of a 2D curve

Where are implicit reps used?

- Bresenham line drawing algorithm
- Intersections tests in ray tracing or **collision detection**
- surface reconstruction
- *** Evolve a surface by evolving its 3D scalar field, governed by a **level-set** — topology changes automatic
- In the DL era, implicit functions are desirable **neural representations** for 3D shapes in terms visual quality

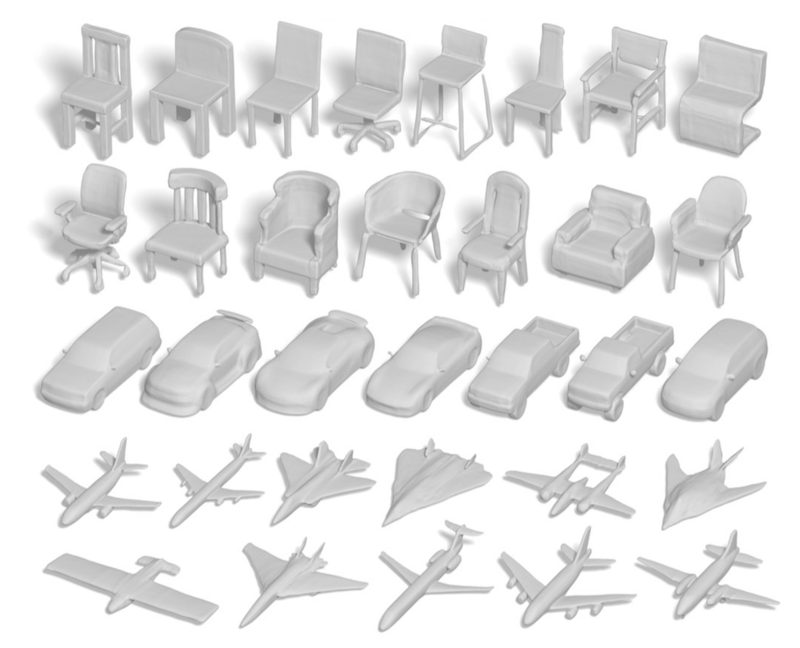


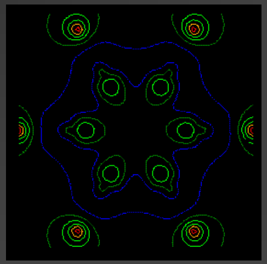
Figure 1: Shapes generated by our implicit field generative adversarial network (IM-GAN), which was trained on 64^3 or 128^3 voxelized shapes. The output shapes are sampled at 512^3 resolution and rendered via marching cubes.

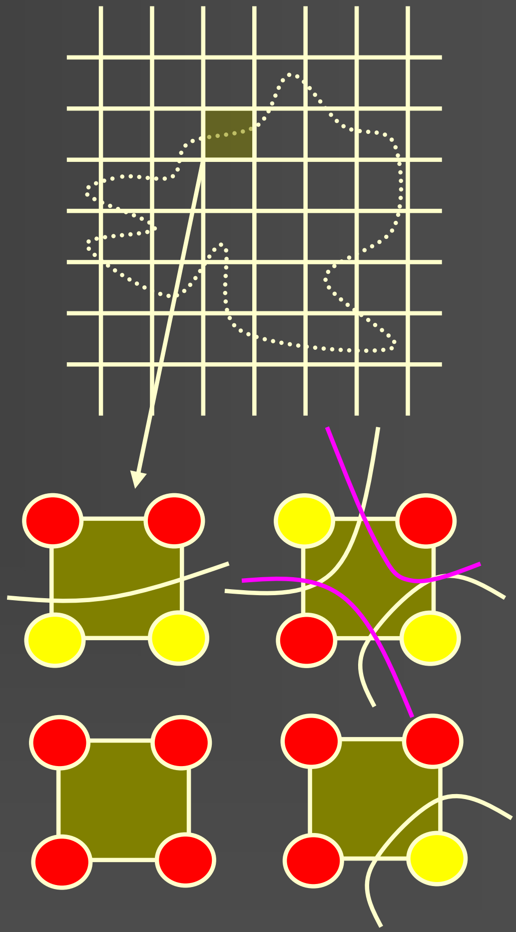
[Chen & Zhang CVPR 2019]

Rendering of an implicit form f(**x**) = 0

Convert to discrete forms, e.g., a mesh

- In 2D case, overlay a **regular grid**
- Assign signs to grid points depending on f
  - f(**x**) < 0: **x** ← **–**
  - f(**x**) > 0: **x** ← **+**
- Visit one cell at a time
  1. Linearly interpolate along edge to determine point of intersection
  2. Connect points depending on sign at corners





Generalization to 3D: **Marching cubes** (later)

2. Parametric curves & surfaces

- 2D **planar curve segment**:

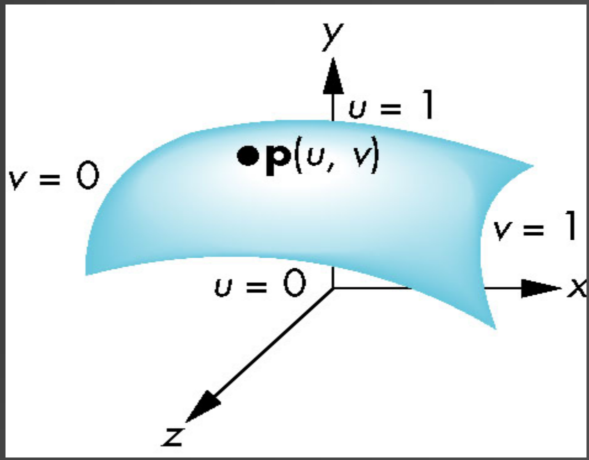
  $(x(t), y(t)), t \in [0, 1]$

- 3D **space curve segment**:

  $(x(t), y(t), z(t)), t \in [0, 1]$

- 3D **surface patch**:

  $(x(u, v), y(u, v), z(u, v)), u, v \in [0, 1]$



Use of polynomials

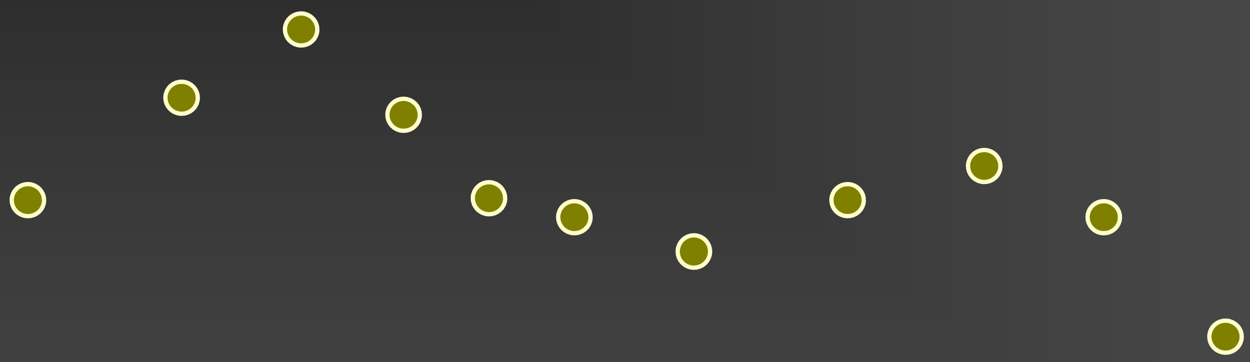
- In computer graphics, we prefer parametric curves and surfaces defined by polynomials
  - **Approximation power:** Can approximate any continuous function to any accuracy (Weierstrass's Theorem)
  - All **derivatives and integrals** are available (infinitely smooth) and easy to compute
  - **Compact representation**
  - Can offer **local control** for shape design with the use of **piecewise polynomials**

Degree of polynomials

- Degree 0 – 2: simple but not enough flexibility
- High-degree: unnecessarily complex and easy to introduce undesirable wiggles — most objects have a **fair** shape
- Most common in graphics as well as computer-aided geometric design (CAGD): **parametric cubic curves and surfaces**

Scattered point interpolation

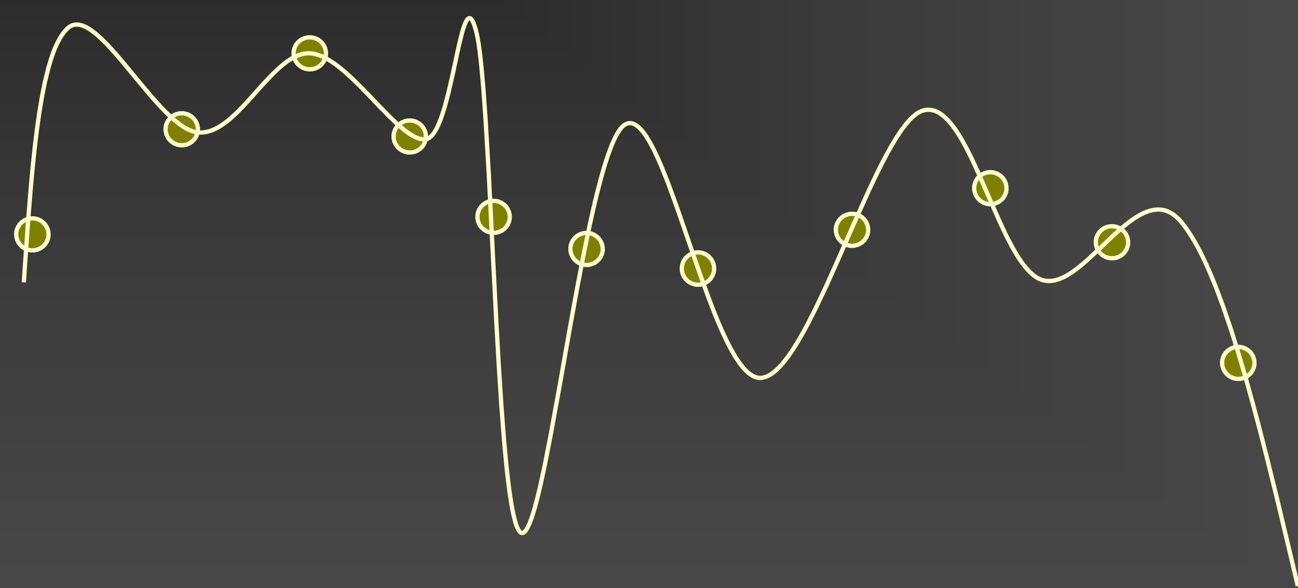
- Consider an interpolation problem:



What is the polynomial function here?

High-degree polynomials

- Consider an interpolation problem:



High-degree polynomial interpolant: **smooth but not fair**

Fairness vs. smoothness

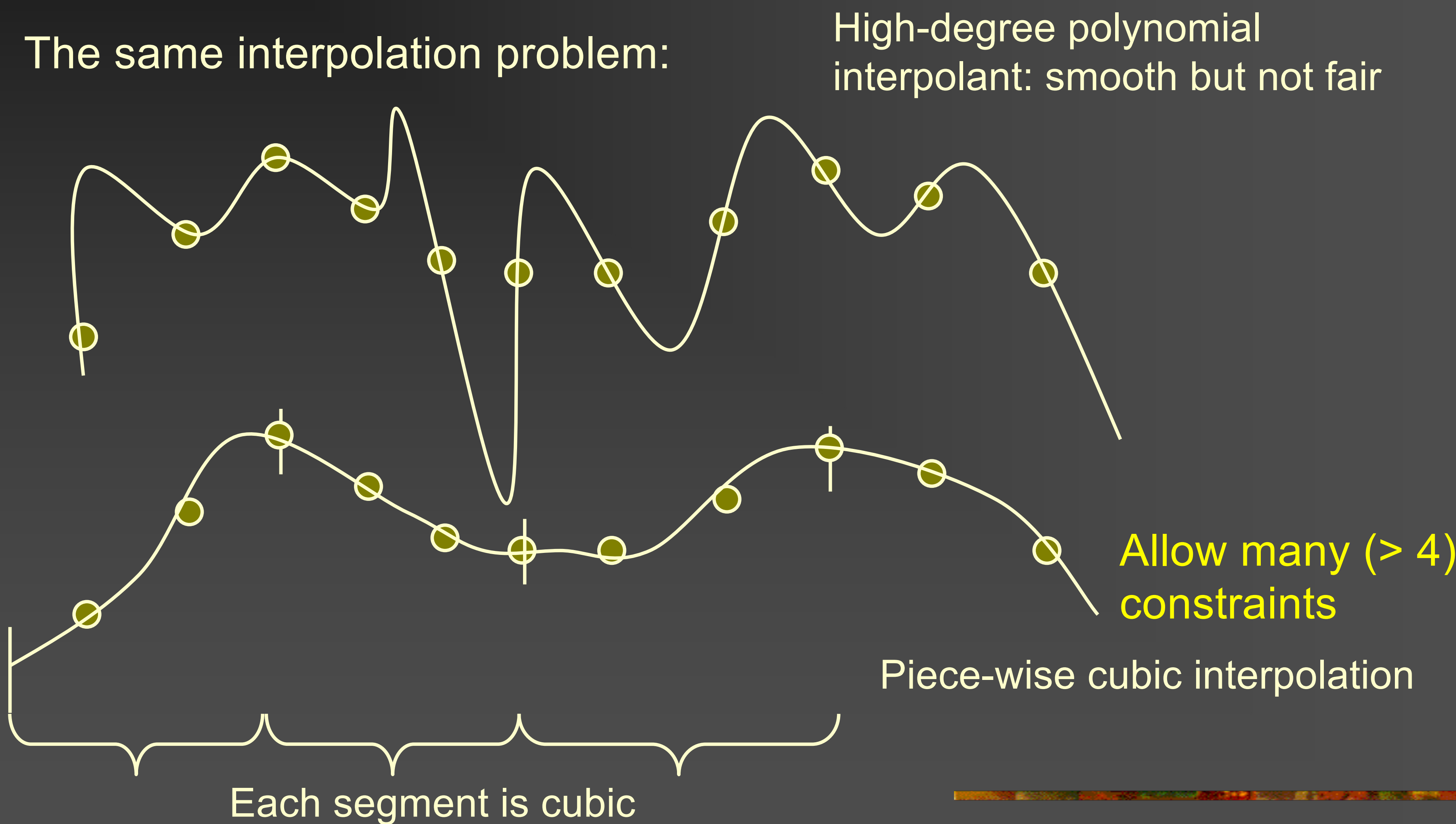
- Smoothness of curves and surfaces:
  - **Local property**: often achieved by design
  - Related to existence and continuity of various derivatives,

    e.g., $3x^{100} - 9x^2 + \ldots$ is infinitely smooth, is it "visually pleasing"?
- Fairness (often appears in CAGD literature)
  - **Global property**: achieved by some form of **energy minimization**
  - Related to the "energy" of a curve or surface

    e.g., $3x^{100} - 9x^2 + \ldots$ has high bending energy — not visually pleasing

Remedy: piece-wise polynomials

- The same interpolation problem:



Parametric cubic segment

Consider a single piece: $x(t) = a_3t^3 + a_2t^2 + a_1t + a_0$

$y(t) = b_3t^3 + b_2t^2 + b_1t + b_0$

$z(t) = c_3t^3 + c_2t^2 + c_1t + c_0$

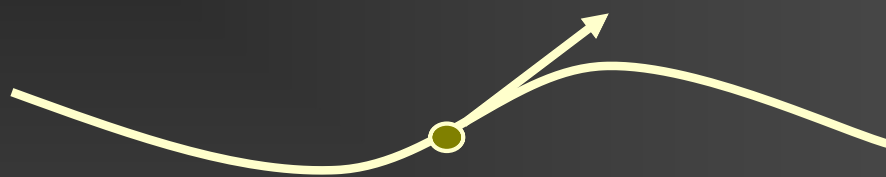
In matrix form:

$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \quad \text{or} \quad x(t) = TA$$

T is said to be the **monomial basis**

Derivatives and continuity

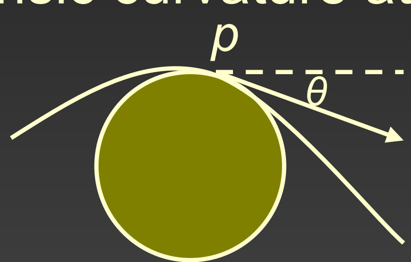
- 1^{st}-order derivative of $(x(t), y(t))$: $(x'(t), y'(t))$ – **tangent**



- 2^{nd}-order derivative: $(x''(t), y''(t))$ – related to **curvature**
- **Parametric continuity** of a curve (**smoothness of motion**):
  - $\mathbf{C}^0$ continuous: curve is joined or connected
  - $\mathbf{C}^1$: requires $\mathbf{C}^0$ & 1^{st}-order derivative is continuous
  - $\mathbf{C}^2$: requires $\mathbf{C}^0$ & $\mathbf{C}^1$ & 2^{nd}-order derivative is continuous
  - $\mathbf{C}^n$: requires $\mathbf{C}^0$ & … & $\mathbf{C}^{n-1}$ & n-th derivative continuous

Curvature of plane curve

- Extrinsic vs. intrinsic definitions
- Intrinsic curvature at a point *p* on a plane curve:



1/*R*, where *R* is the radius of the **osculating circle**

  - Osculating circle: limit circle passing through *p* and its neighbors
  - **Unit of curvature: inverse distance**
- Extrinsic curvature at *p* of plane curve (*x*(*t*), *y*(*t*))

$$\kappa = \frac{d\theta}{ds} = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} = 1/R$$

where θ is the **turning angle** and *s* is **arc length**

Continuity of piecewise curves

- A single polynomial segment is always $\mathbf{C}^{\infty}$
- But we mostly deal with **piecewise** polynomial curves
- Key: what happens at the joints between segments
  - $\mathbf{C}^0$: curve segments are connected
  - $\mathbf{C}^1$: $\mathbf{C}^0$ & **1st-order derivatives agree at joints**
  - $\mathbf{C}^2$: $\mathbf{C}^0$ & $\mathbf{C}^1$ & 2nd-order derivatives agree at joints, etc.
- If parametric continuity not possible to enforce, can relax to
  - **"Visual" smoothness**: direction of tangents stays the same, but magnitude (speed) may change

Geometric continuity

- **geometric continuity**
  - $\mathbf{G}^0$ continuous: curve segments are connected (same as $\mathbf{C}^0$)
  - $\mathbf{G}^1$: $\mathbf{G}^0$ & **1^{st}-order derivatives are proportional at joints**.
  - Note:
    - Proportional = same direction but may have different magnitudes
    - Weaker than $\mathbf{C}^1$
  - $\mathbf{G}^2$: $\mathbf{G}^1$ & 2^{nd}-order derivative proportional at joints
- Example: $p(t) = (3t, t^3)$ and $q(t) = (4t+3, 2t^2+4t+1)$ with $t \in [0, 1]$ for each. Is this $\mathbf{C}^0$, $\mathbf{G}^1$, and/or $\mathbf{C}^1$?

  $p(1)=q(0)=(3,1)$, so $\mathbf{G}^0$; $p'(1)=(3,3)$ and $q'(0)=(4,4)$, so $\mathbf{G}^1$ not $\mathbf{C}^1$

Now on to curve design

- Do you say to yourself,

"I want to design a cubic curve $a_3t^3 + a_2t^2 + a_1t + a_0$ with $a_3 = 1$, $a_2 = -9$, $a_1 = 4$, and $a_0 = 21$"?

Curves with the right design constraints

- Want to design piecewise cubic polynomial curves that satisfy certain **design constraints**, e.g.,
  - Curve should pass certain points
  - Curve should have some given derivatives at specific points
  - Curve should be smooth: $\mathbf{G}^1$, $\mathbf{C}^1$, $\mathbf{C}^2$, or …
  - Curve must be contained in certain area, or has at most this length, etc.
- Need to use **proper basis** functions to facilitate the design process
- Often, the basis used identifies the curve representation

Basis functions and control points

- Recall basis expansion: $x(t) = P_1 b_1(t) + P_2 b_2(t) + P_3 b_3(t) + P_4 b_4(t)$
- Monomial basis, $\{1, t, t^2, t^3\}$: only one of many possible bases for cubic polynomials
- From a design point of view, want P_1, P_2, P_3, and P_4 to represent **observable** quantities (not so for monomial basis), e.g.,
  - Position: for interpolation
  - Derivatives: to control direction and smoothness, etc.
- P_1, P_2, P_3, and P_4 serve as **control points**
- Control points are blended by the basis functions b_1, b_2, b_3, and b_4

Example 1: Cubic Hermite curves

- Defined by two points (P_1 and P_4) and two tangents (R_1 and R_4)
- Aim: Achieve $\mathbf{C}^1$ or $\mathbf{G}^1$ continuity
- Want cubic curve $x(t)$, $t \in [0, 1]$, such that

  $x(0) = P_1$

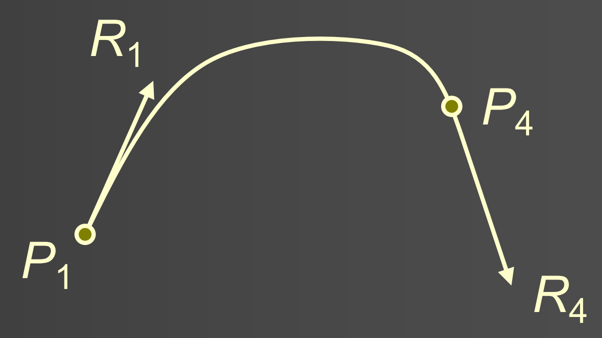
  $x(1) = P_4$

  $x'(0) = R_1$

  $x'(1) = R_4$

  (y and z are similar)

- Usage example: determining the trajectory of a ball in animation



Let us note that the control "points" P_1, P_4, R_1, and R_4 are all observable quantities and they control the shape of the curve

Cubic Hermite curves

- $x(t) = TA = a_3t^3 + a_2t^2 + a_1t + a_0$, where $T = [t^3\ t^2\ t\ 1]$ and $A = [a_3\ a_2\ a_1\ a_0]^T$. We want

$$
\begin{aligned}
x(0) &= P_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A \\
x(1) &= P_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A \\
x'(0) &= R_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} A \\
x'(1) &= R_4 = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} A
\end{aligned}
$$

$$
\text{or } G = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} A = BA
$$

- Hence, $G = BA$ and thus $A = B^{-1}G$
- It follows that $x(t) = TA = TB^{-1}G = HG$

Cubic Hermite curves

- How to interpret this: $x(t) = TA = TB^{-1}G = HG$
  - G: vector of observables or control points
  - H: vector of cubic **Hermite basis (blending)** functions

$$H = [2t^3 - 3t^2 + 1,\ -2t^3 + 3t^2,\ t^3 - 2t^2 + t,\ t^3 - t^2]$$

  - For any G, use H to blend four control points to get curve $x(t)$
  - The matrix $M_{\text{hermite}} = B^{-1}$ is really a **change-of-basis matrix:** changes the monomial basis T into the Hermite basis H
  - Hermite curves are completely determined by M_{hermite}

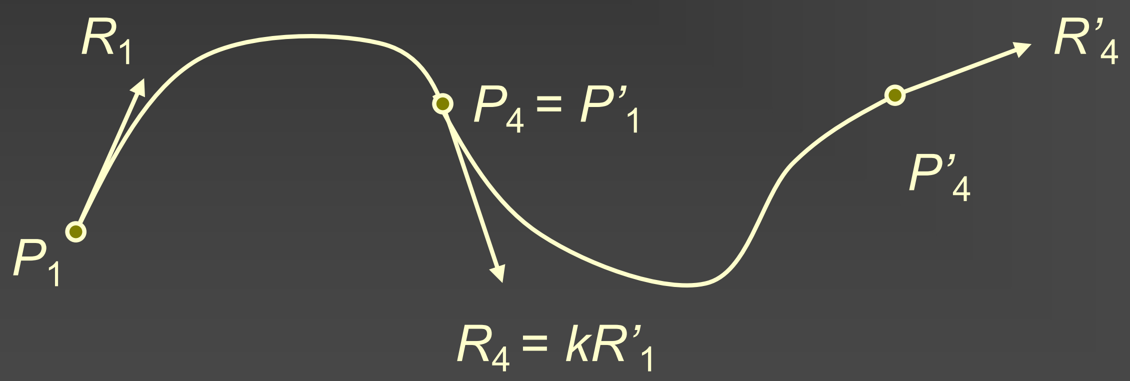
The cubic Hermite matrix

$$M_{Hermite} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- The Hermite change of basis matrix or its basis identifies the **Hermite representation** of cubic parametric curves
- Any cubic parametric curve can be specified in Hermite form

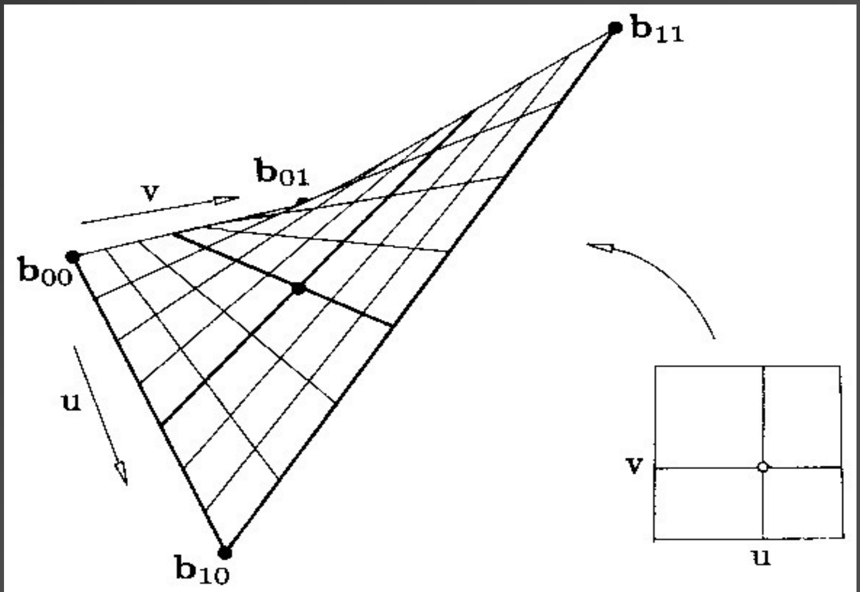
Piecewise Hermite curves

- Can obviously enforce **C**^{1} or **G**^{1} continuity at the joints
- Each segment parameterized over [0, 1] as usual



From curves to surfaces

- One easy way: sweep a curve whose control points also trace out some curves, e.g., bilinear interpolation
- Fit the simplest surface between four points
- Sweep a straight line and each point on the line traces a straight line
- An example of a **ruled surface**
- An example of **tensor-product surfaces**



bilinear interpolation

Tensor-product (TP) surfaces

- The curve to sweep:

$$p(u)=\sum_{i=0}^{m} a_i A_i(u)$$

- Control point a_i goes through a curve

$$a_i = q_i(v) = \sum_{j=0}^{n} P_{ij} B_j(v)$$

- The resulting surface is a tensor-product surface

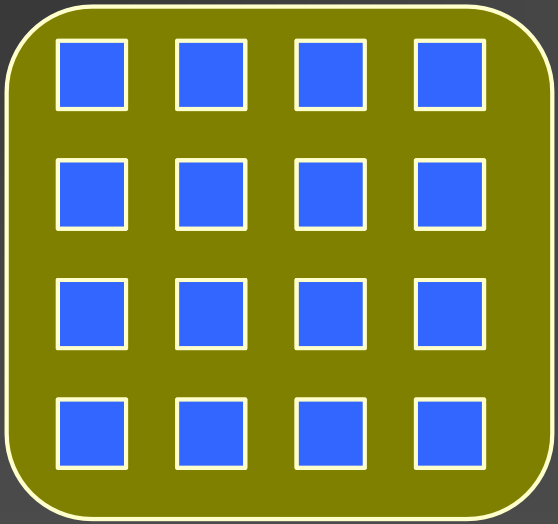
$$p(u,v)=\sum_{i=0}^{m}\sum_{j=0}^{n} P_{ij} A_i(u) B_j(v) = \mathbf{A(u)}^{\mathrm{T}}\mathbf{PB(v)}$$

- Surface is controlled by the grid of control points P_{ij}

Tensor-product (TP) surfaces

$$p(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} A_i(u) B_j(v) = \mathbf{A}(\mathbf{u})^{\mathrm{T}} \mathbf{P} \mathbf{B}(\mathbf{v})$$

$[\ A_0(u)\ \ A_1(u)\ \ A_2(u)\ \ A_3(u)\ ]$

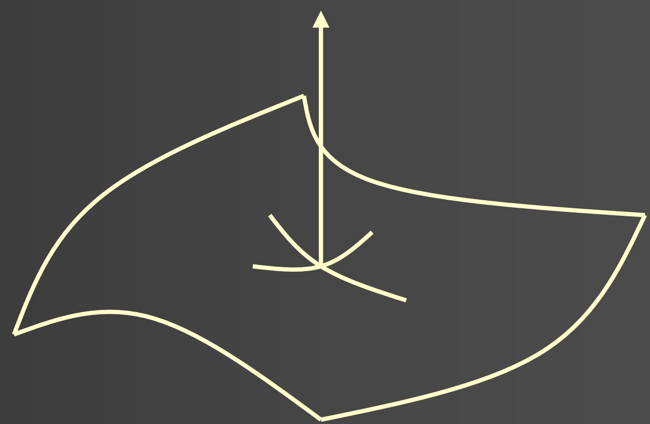


$\begin{bmatrix} B_0(v) \\ B_1(v) \\ B_2(v) \\ B_3(v) \end{bmatrix}$

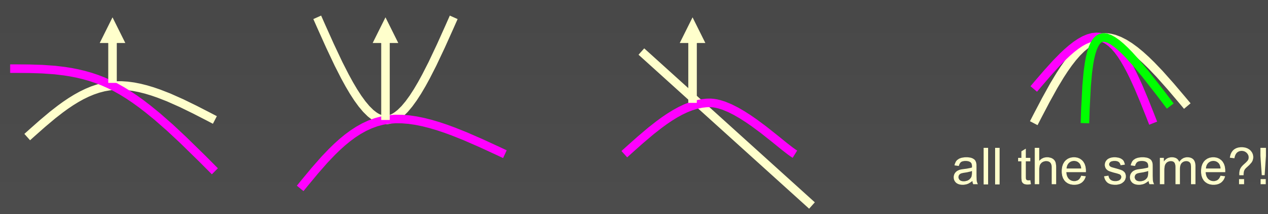
Curvature of surfaces

- **Regular point** on a surface
  - Consider all curves lying in the surface through the point
  - Point is regular if tangent vectors of all these curves lie in the same plane — the **tangent plane**
- Surface **normal** at regular point: normal to tangent plane
- Intersection between surface and a plane through the normal is called a **normal section**
- **Principal curvatures**: maximum (κ_1) and minimum (κ_2) curvatures of the normal sections

Curvature of surfaces



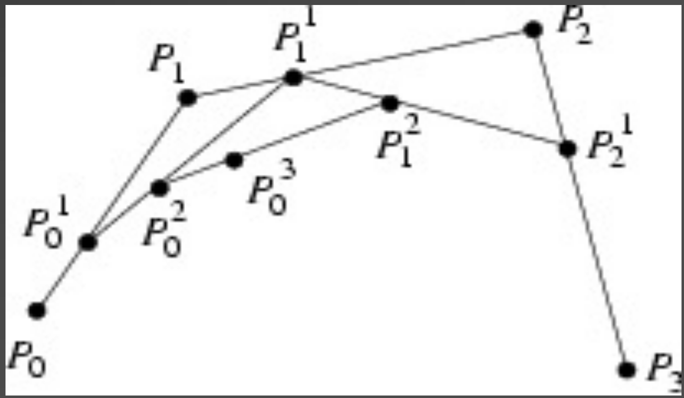
- **Gaussian curvature:** $\kappa_1 \kappa_2$
- **Mean curvature:** $\frac{\kappa_1 + \kappa_2}{2}$
- For a regular point, the two **principal (curvature) directions are perpendicular**
- **Elliptic, hyperbolic, parabolic, umbilical points**



Exercises

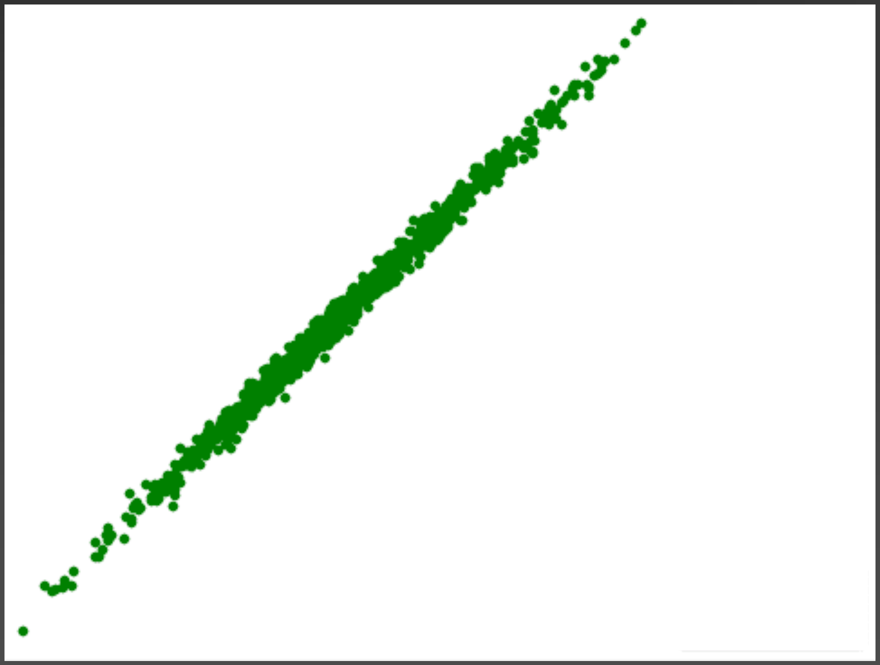
- Design a curve of your own, e.g., interpolate two end points and interpolate position and tangent at midpoint – compute C.O.B. matrix

- Identify the curve …
  - **de Casdeljau** algorithm



Primitive fitting

- Given a set of points, find the parameters of a primitive (e.g., a line or plane, a sphere, or a cylinder) to provide the **best fitting**



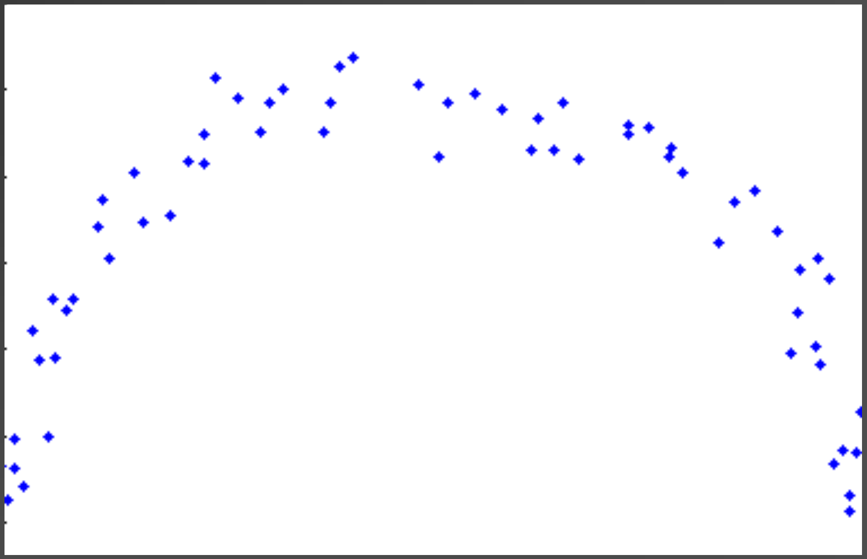
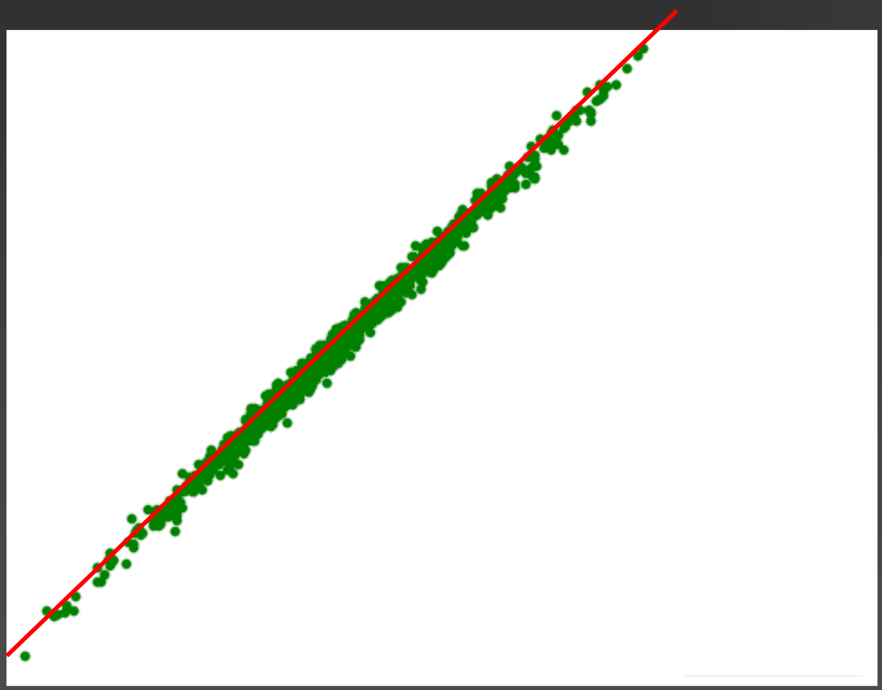


Image taken from Ragon Ebker
https://www.baeldung.com/cs/ransac

What is the best fitting?

- **Least square** (LSQ) fitting: find the primitive which minimizes the **sum of squared distances** from the set of points



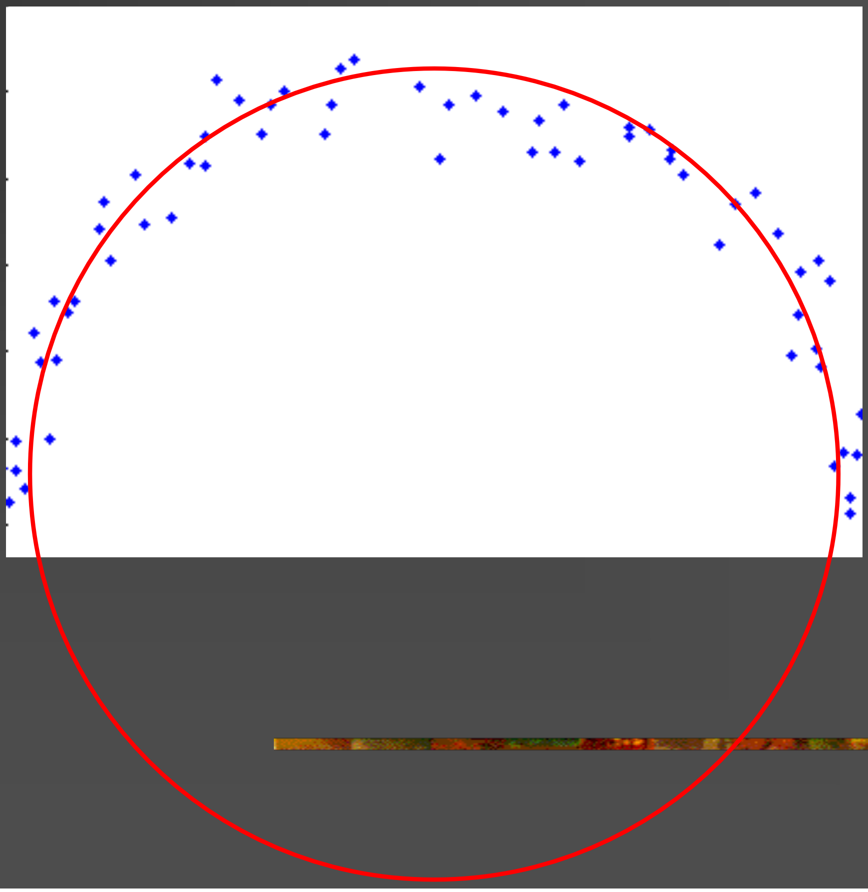


Image taken from Ragon Ebker
https://www.baeldung.com/cs/ransac

Problem with LSQ

- **Outliers!**

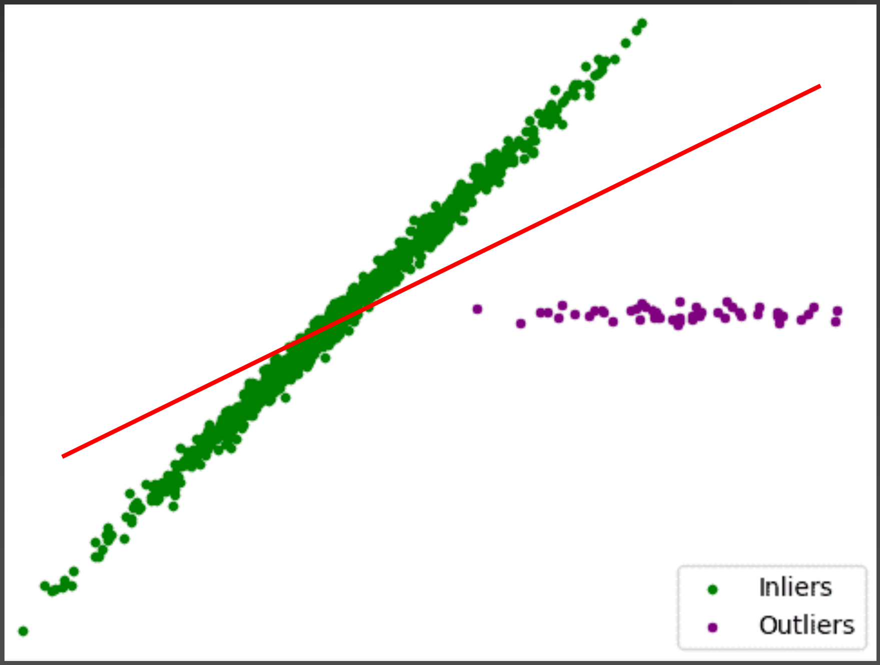


Image taken from Ragon Ebker
https://www.baeldung.com/cs/ransac

Problem with LSQ

- Even very few outliers can cause problems

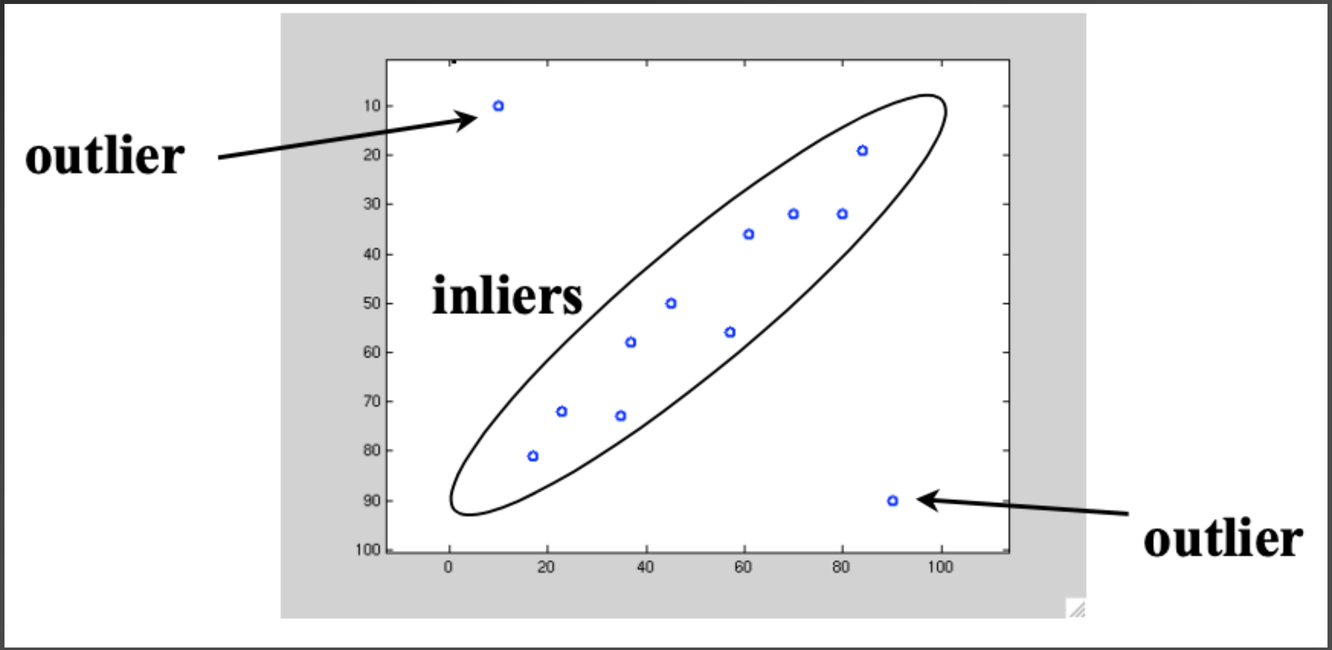


Image taken from Robert Collins
https://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf

Problem with LSQ

- Even very few outliers can cause problems

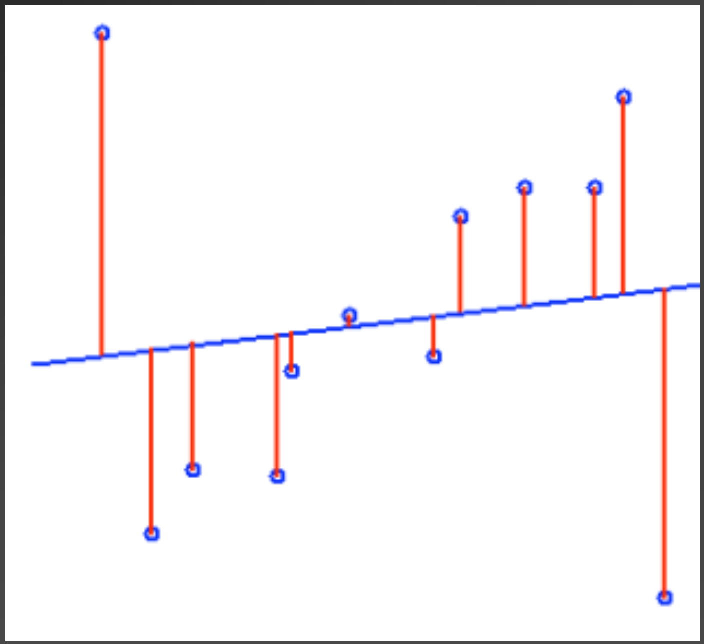


Image taken from Robert Collins
https://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf

A good solution: RANSAC

- **RANSAC** = **RAN**dom **SA**mple **C**onsensus
- Key idea: classify points into inliers, outliers, and **eliminate** the latter
- The model/primitive is only fit to the inliers

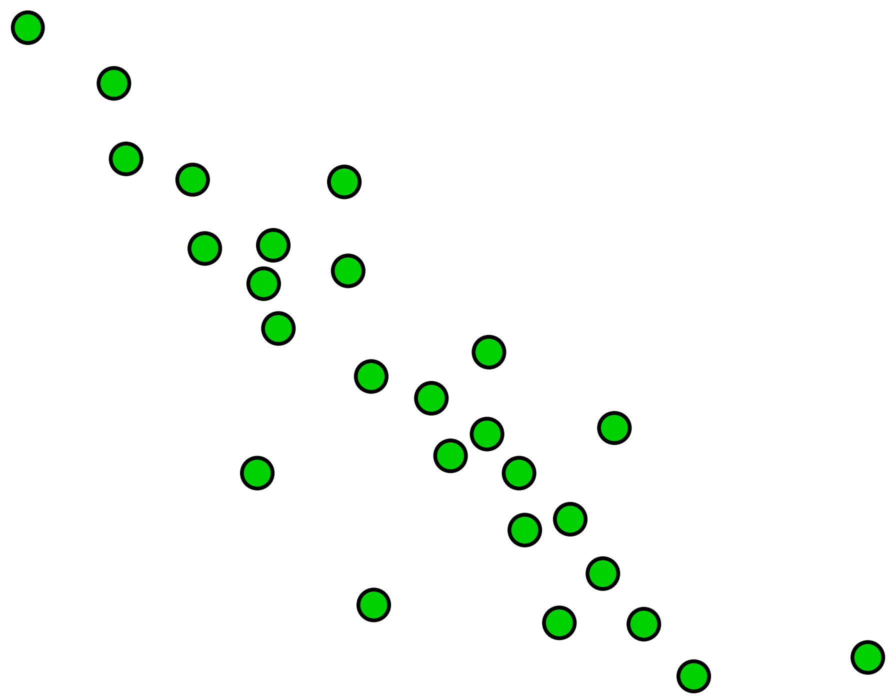
M. A. Fischler and R. C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". *Comm. of the ACM* **24**: 381--395.

Image taken from Robert Collins
https://www.cse.psu.edu/~rtc12/CSE486/lecture15.pdf

RANSAC

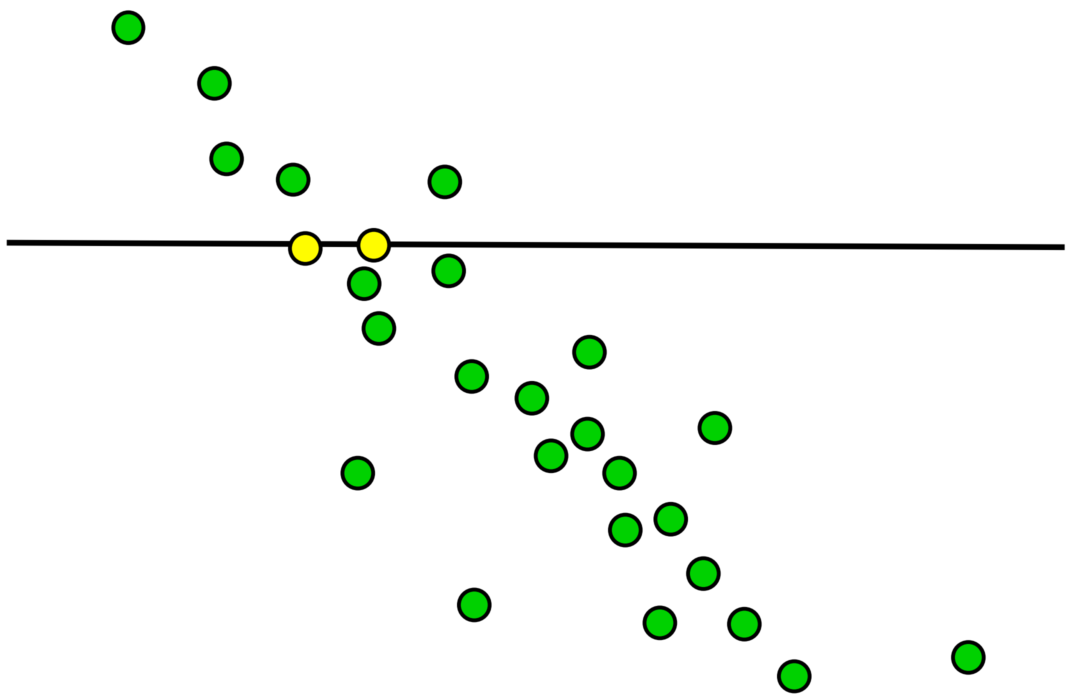
- Key idea: classify points into inliers, outliers, and **eliminate** the latter
- The model/primitive is only fit to the inliers\
- **RANSAC** = **RAN**dom **SA**mple **C**onsensus

Ransac Procedure

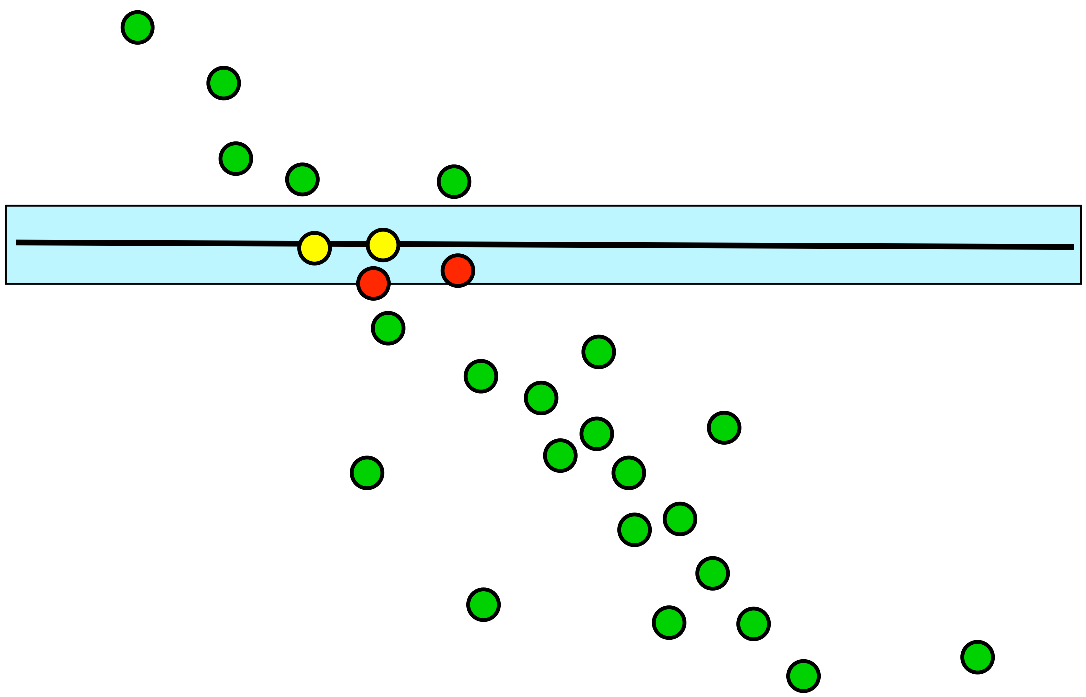


Robert Collins
CSE486, Penn State

Ransac Procedure



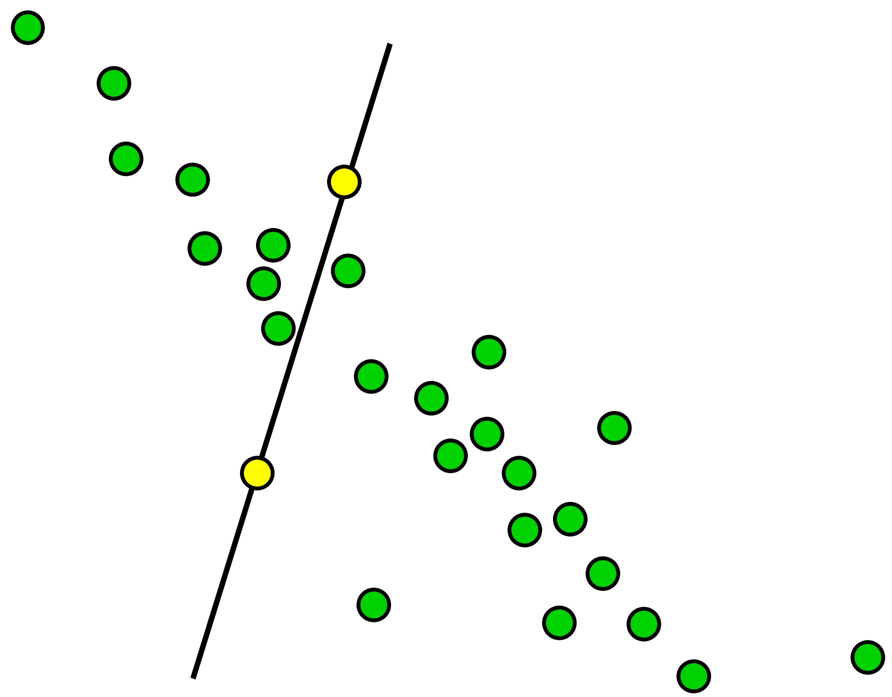
Ransac Procedure



Count = 4

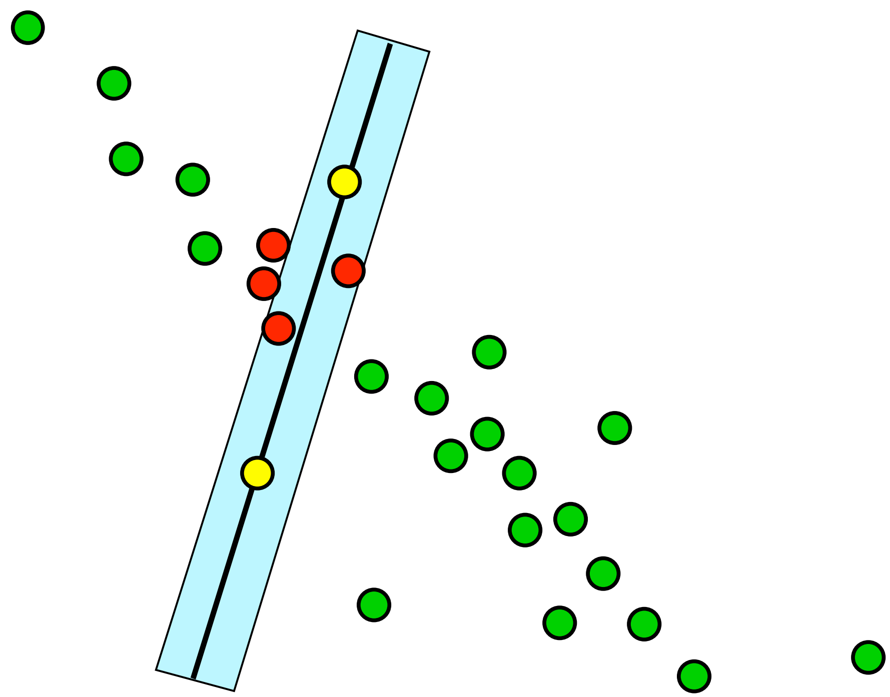
Robert Collins
CSE486, Penn State

Ransac Procedure



Robert Collins
CSE486, Penn State

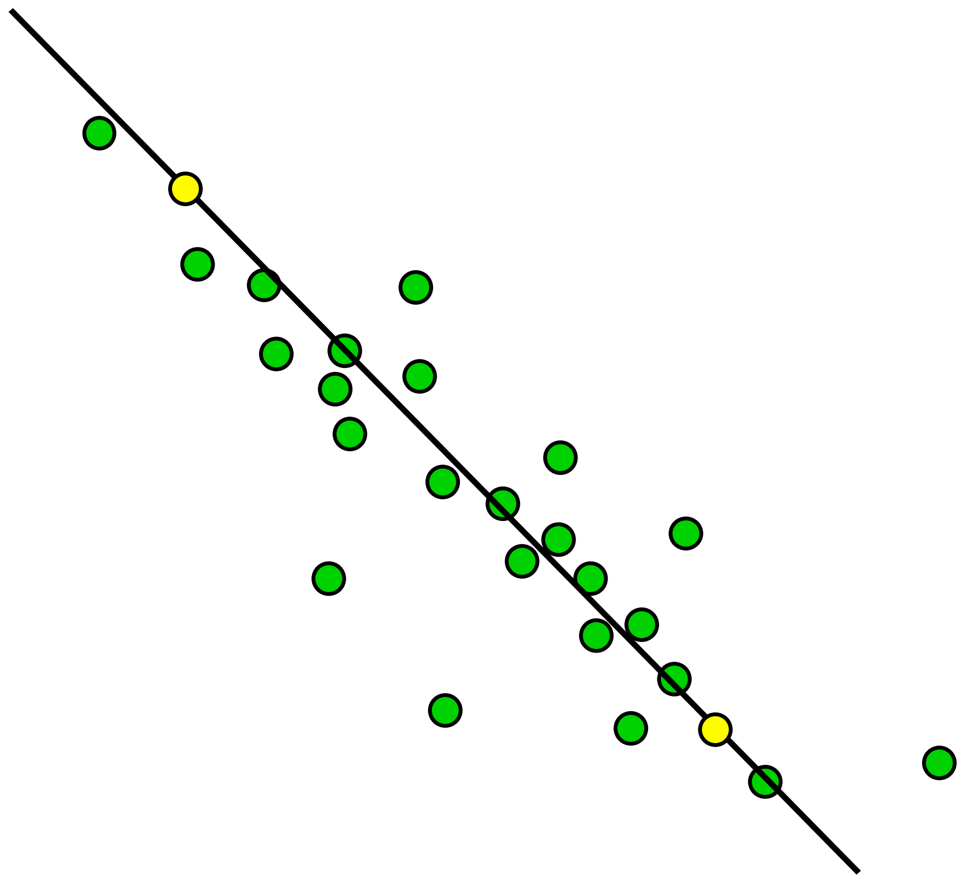
Ransac Procedure



Count = 6

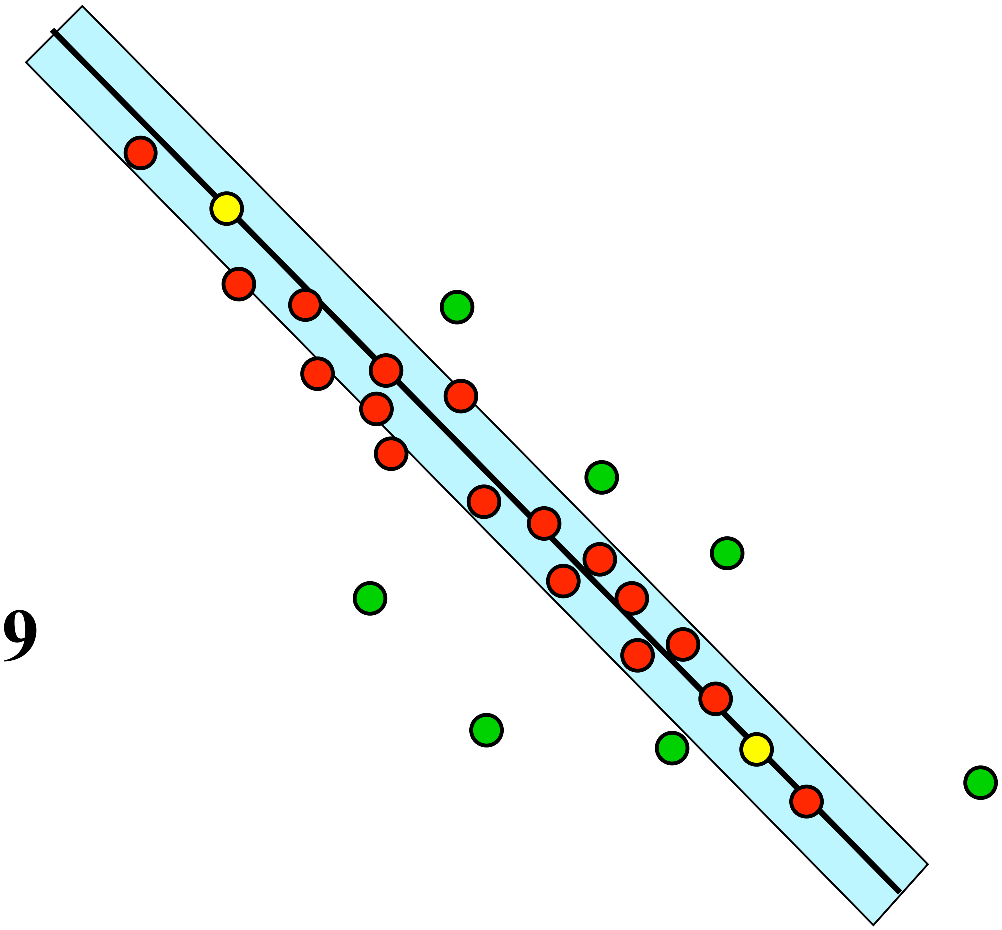
Robert Collins
CSE486, Penn State

Ransac Procedure



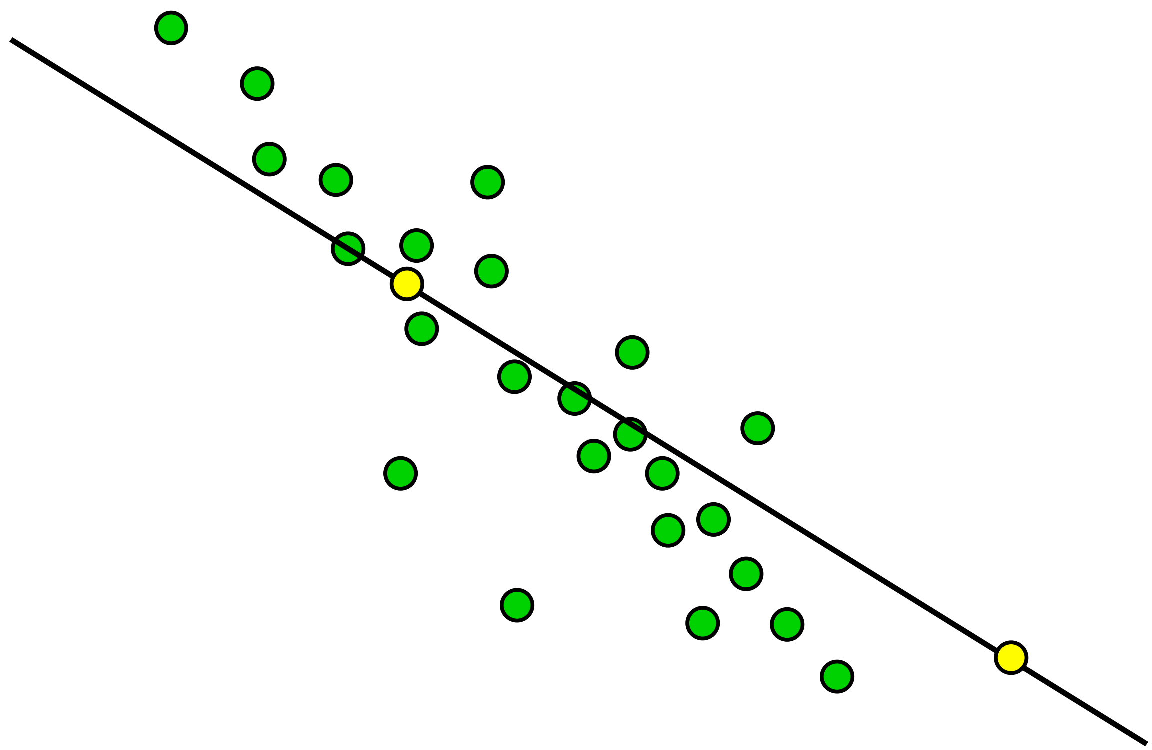
Robert Collins
CSE486, Penn State

Ransac Procedure

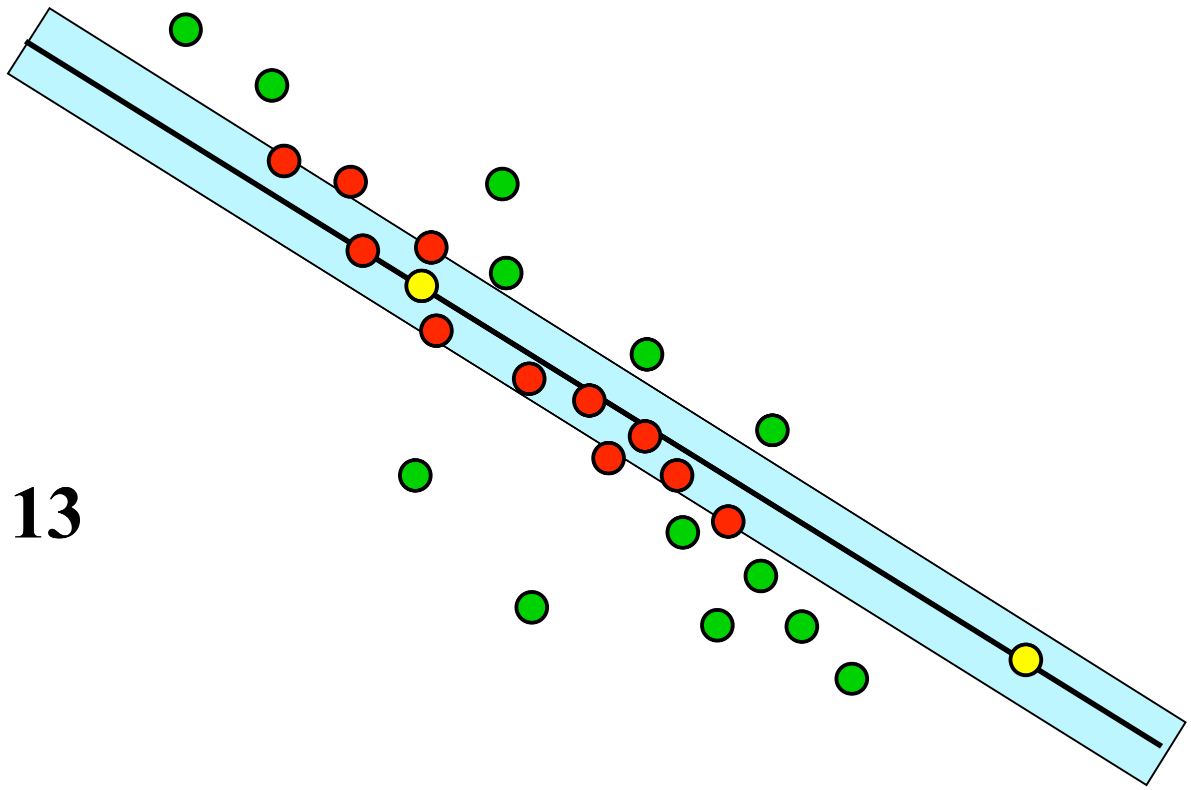


Count = 19

Ransac Procedure



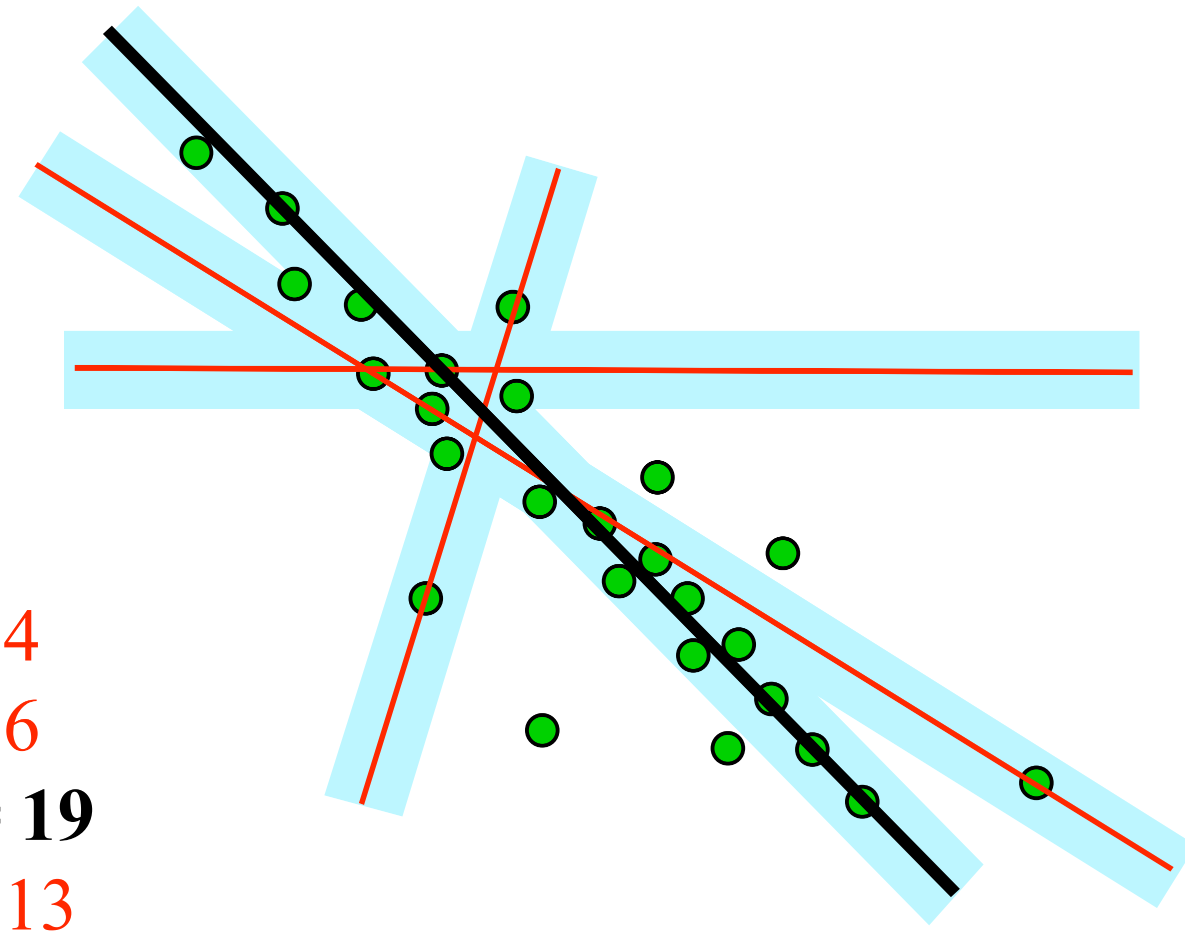
Ransac Procedure



Count = 13

Robert Collins
CSE486, Penn State

Ransac Procedure



Count = 4

Count = 6

Count = 19

Count = 13

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
  s — the smallest number of points required
  N — the number of iterations required
  d — the threshold used to identify a point that fits well
  T — the number of nearby points required
    to assert a model fits well
Until **N** iterations have occurred
  Draw a sample of **s** points from the data
    uniformly and at random
  Fit to that set of **s** points
  For each data point outside the sample
    Test the distance from the point to the line
      against **d** if the distance from the point to the line
      is less than **d** the point is close
  end
  If there are **T** or more points close to the line
    then there is a good fit. Refit the line using all
    these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

(Forsyth & Ponce)