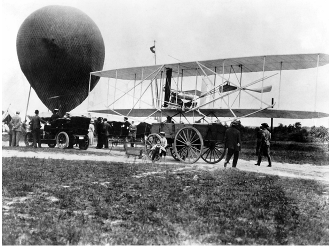
# Algebra, Geometry and Robots

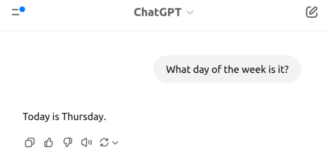
## The two-way relationship between mathematics and robotics, with a focus on modern algebra and geometry.

Sepehr Saryazdi

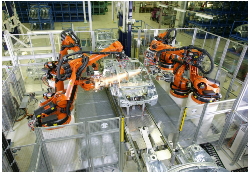
University of Sydney

# The Story of Humanity

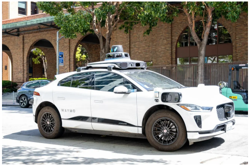




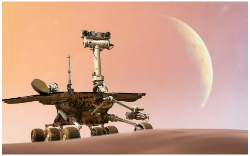
# Humanity's Next Tool: Robots



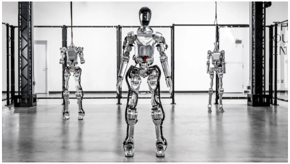
KUKA - Robotic Arms




Waymo -
Self-Driving Cars

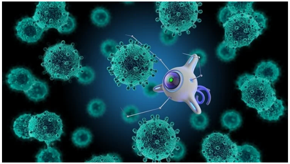


NASA - Martian Rovers



BMW -
Humanoid Robots

Boston Dynamics -
Dog Robots



Nanorobots

# What is Robotics?

## Common Definition of Robotics

$$R(t) = \text{Robotics at time } t$$

$$A(t) := \text{What's Theoretically Possible at time } t$$

$$P(t) := \text{What's Already Possible at time } t$$

$$R(t) := A(t) - P(t)$$

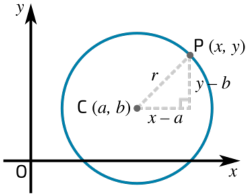
## Goal of Robotics

$$\lim_{t \to \infty} R(t) = 0$$

# The Story of Mathematics

$$1 + 2 = 3$$

$$ax^2 + bx + c = 0$$




$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$GL_n(\mathbb{R})$$

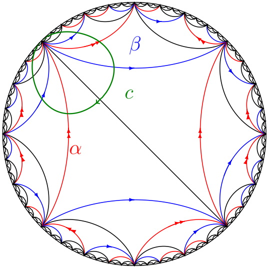
$$\mathrm{Grp} \to \mathrm{Set}$$

# The Story of Mathematics: Today's Mathematics

Algebra Analysis

Topology Geometry

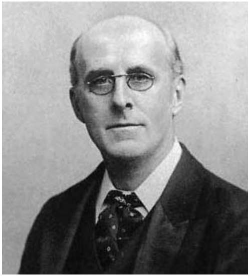
# Who am I?




Recent Master's in Mathematics (Geometric Topology)


Starting PhD with ACFR and CSIRO's Robotic Perception and Autonomy Group

# Kempe - Steam Engines



Alfred Kempe
1849-1922

Steam Engine

# Kempe - Tracing Curves

https://www.youtube.com/watch?v=9NbwE4PMeyQ

# Kempe - Curves

## Curves in the Plane (Algebraic Subset)

$$P(x, y) = \sum_{ij} c_{ij} x^i y^j$$

A curve $\mathcal{C} \subseteq \mathbb{R}^2$ is defined as the set of solutions to the equation $P(x, y) = 0$.

# Kempe - Available Linkages

Parallelogram Linkage

Peaucellier–Lipkin Linkage

https://demonstrations.wolfram.com/PeaucelliersAndHartsInversor/

# Kempe - Free Rhombus Linkage

## Rhombus Linkage

$$\text{Want: } P(x,y) = \sum_{ij} c_{ij} x^i y^j = 0$$

$$x = r\cos(\alpha) + r\cos(\beta)$$

$$y = r\sin(\alpha) + r\sin(\beta)$$

# Kempe - Substitute and Trig!

## Substitute

$$x = r\cos(\alpha) + r\cos(\beta)$$

$$y = r\sin(\alpha) + r\sin(\beta)$$

$$P(x,y) = \sum_{ij} c_{ij}(r\cos(\alpha) + r\cos(\beta))^i (r\sin(\alpha) + r\sin(\beta))^j = 0$$

## Refactor Everything Into cos's

$$\sin(\theta) = \cos\left(\frac{\pi}{2} - \theta\right)$$

$$\cos(A)\cos(B) = \frac{1}{2}\left(\cos(A+B) + \cos(A-B)\right)$$

# Kempe - Substitute and Trig!

## Refactor Everything Into cos's

$$\sin(\theta) = \cos\left(\frac{\pi}{2} - \theta\right)$$

$$\cos(A)\cos(B) = \frac{1}{2}\left(\cos(A+B) + \cos(A-B)\right)$$

## Collect Like Terms and Simplify

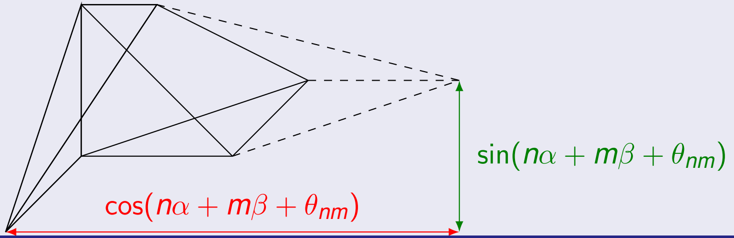
$$P(x, y) = \sum_{n,m\in\mathbb{Z}} c'_{nm} \cos(n\alpha + m\beta + \theta_{nm}), \theta_{nm} \in \{0, \pm\pi/2\}$$

# Kempe - Linkage Arithmetic

## Simplified Form

$$P(x, y) = \sum_{n,m \in \mathbb{Z}} c'_{nm} \cos(n\alpha + m\beta + \theta_{nm}), \theta_{nm} \in \{0, \pm\pi/2\}$$

## Treat Each cos Term Like The Projection of a 'New' Linkage!

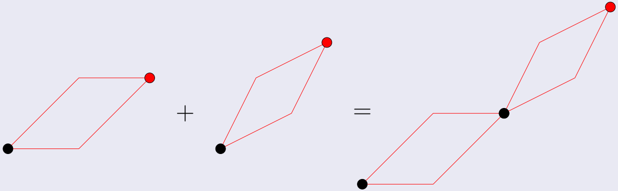


# Kempe - Linkage Arithmetic

## Simplified Form

$$P(x,y) = \sum_{n,m\in\mathbb{Z}} c'_{nm}\cos(n\alpha + m\beta + \theta_{nm}), \theta_{nm} \in \{0, \pm\pi/2\}$$

## Adding Linkages



# Kempe - Linkage Arithmetic

https://demonstrations.wolfram.com/KempesTranslator/

## Translating Linkages

$$a + \cos(\theta), a, \theta \in \mathbb{R}$$

# Kempe - Linkage Arithmetic

https://demonstrations.wolfram.com/KempesMultiplicator/

## Multiplying Angles

$$\cos(n\alpha), n \in \mathbb{Z}$$

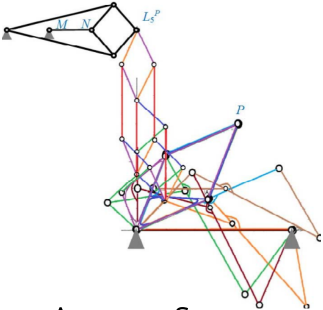
# Kempe - Linkage Arithmetic

https://demonstrations.wolfram.com/KempesAngleAdder/

## Adding Angles

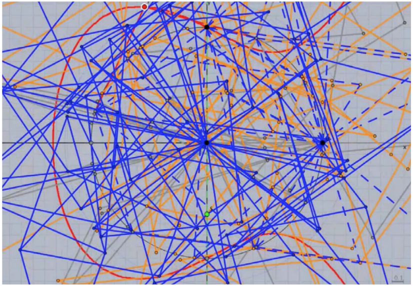
$$\cos(\theta_1 + \theta_2), \theta_1, \theta_2 \in \mathbb{R}$$

# Kempe - Combining Linkages




Anupam Saxena

$$(x-y)(x+y+1/\sqrt{2})=0$$



Alexander Kobel

$$x^3-y^2-x+1=0$$

## Simplified Form

$$P(x,y)=\sum_{n,m\in\mathbb{Z}} c'_{nm}\cos(n\alpha+m\beta+\theta_{nm})=0$$

# Kempe's Universality Theorem

**Kempe's Universality Theorem (Statement made rigorous by Thurston & Kapovich-Millson)**

Let $\mathcal{C} \subseteq \mathbb{R}^2$ be a curve. Let $f : [a, b] \to \mathcal{C}$ be a map such that $f(t) = (p(t), q(t))$ where $p, q \in \mathbb{R}[x]$ are polynomials. Then there is a linkage $\mathcal{L}$ with some pinned vertices, and a vertex $v$ of $\mathcal{L}$ so that $v$ traces out $f([a, b])$ over all the linkage's configurations.

# Kempe's Universality Theorem

**Corollary - (Thurston) There is a linkage which signs your name.**

Break up your name into $k$ segments, each of which is the image of a smooth function $f_k : [a_k, b_k] \to \mathbb{R}^2$. Apply the Stone-Weierstrass Approximation Theorem to approximate your curves by polynomials. Apply the previous result and carefully combine linkages, or apply the result to Lagrange polynomial interpolation.

# Kempe's Universality Theorem - Generalisations

- The proofs were rigorously completed in 2002 by Kapovich and Millson.
- (Kapovich-Millson) The result generalises to higher-dimensional algebraic sets, i.e. simultaneous solutions to $P_1(x_1, \ldots, x_n) = 0, P_2(x_1, \ldots, x_n) = 0, \ldots, P_m(x_1, \ldots, x_n) = 0.$
- (Kapovich-Millson) Let $M$ be any smooth compact manifold. Then there is a linkage $\mathcal{L}$ whose configuration space is diffeomorphic to a disjoint union of a number of copies of $M$.

# Gröbner Basis - Polynomials

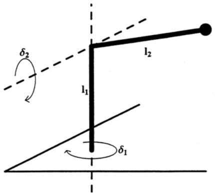
Bruno Buchberger

1942 - Now

■ **Example: Systems of Polynomial Equations**

(This example is taken from (Buchberger, Kutzler 1986)).

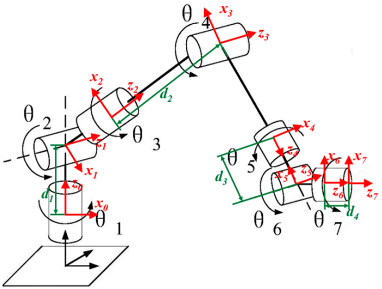
Systems of multivariate polynomial equations are pervasive in all areas of engineering. For example, consider the following simple robot:




Robotics Example

Bruno Buchberger

# Gröbner Basis - Polynomials in Kinematics




General Kinematics Problem
Xuanming Zhang et. al.

Simple Kinematics Example

# Gröbner Basis - Polynomials in Kinematics

## Kinematics As A Map

$$F : [0, 2\pi)^3 \to \mathbb{R}^2$$
$$(\theta_1, \theta_2, \theta_3) \mapsto (x, y)$$

# Gröbner Basis - Polynomials in Kinematics

## Kinematics Map

$$(x, y) = F(\theta_1, \theta_2, \theta_3)$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \end{pmatrix}$$

# Gröbner Basis - Polynomials in Kinematics

## Kinematics Map

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1\cos(\theta_1) + L_2\cos(\theta_1+\theta_2) + L_3\cos(\theta_1+\theta_2+\theta_3) \\ L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) + L_3\sin(\theta_1+\theta_2+\theta_3) \end{pmatrix}$$

$$= \begin{pmatrix} L_1c_1 + L_2c_{12} + L_3c_{123} \\ L_1s_1 + L_2s_{12} + L_3s_{123} \end{pmatrix}$$

## Kinematics Polynomials

$$\begin{aligned} L_1c_1 + L_2c_{12} + L_3c_{123} - x &= 0 \\ L_1s_1 + L_2s_{12} + L_3s_{123} - y &= 0 \\ c_1^2 + s_1^2 - 1 &= 0 \\ c_{12}^2 + s_{12}^2 - 1 &= 0 \\ c_{123}^2 + s_{123}^2 - 1 &= 0 \end{aligned}$$

# Gröbner Basis - Polynomials in Kinematics

## Kinematics Polynomials

$$\begin{aligned}
L_1 c_1 + L_2 c_{12} + L_3 c_{123} - x &= 0 \\
L_1 s_1 + L_2 s_{12} + L_3 s_{123} - y &= 0 \\
c_1^2 + s_1^2 - 1 &= 0 \\
c_{12}^2 + s_{12}^2 - 1 &= 0 \\
c_{123}^2 + s_{123}^2 - 1 &= 0
\end{aligned}$$

## Abstracted Polynomials

$$\begin{aligned}
P_1(x_i, y_i) = \quad & L_1 x_1 + L_2 x_2 + L_3 x_3 - x &= 0 \\
P_2(x_i, y_i) = \quad & L_1 y_1 + L_2 y_2 + L_3 y_3 - y &= 0 \\
P_3(x_i, y_i) = \quad & x_1^2 + y_1^2 - 1 &= 0 \\
P_4(x_i, y_i) = \quad & x_2^2 + y_2^2 - 1 &= 0 \\
P_5(x_i, y_i) = \quad & x_3^2 + y_3^2 - 1 &= 0
\end{aligned}$$

# Gröbner Basis - Extending Gauss-Jordan

## Linear Equations

$$\begin{aligned} ax_1 + x_2 &= 1 \\ x_2 + x_3 &= 2 \\ bx_1 + x_2 + x_3 &= 1 \end{aligned} \quad \Leftrightarrow \begin{pmatrix} a & 1 & 0 \\ 0 & 1 & 1 \\ b & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

## Gauss-Jordan Elimination

$$R_3 \mapsto aR_3 - bR_1$$

# Gröbner Basis - Extending Gauss-Jordan

## Linear Equations

$$\begin{aligned} ax_1 + x_2 &= 1 \\ x_2 + x_3 &= 2 \\ bx_1 + x_2 + x_3 &= 1 \end{aligned} \quad \Leftrightarrow \quad \begin{pmatrix} a & 1 & 0 \\ 0 & 1 & 1 \\ b & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

## Gauss-Jordan Elimination

$$R_3 \mapsto aR_3 - bR_1$$

$$\Leftrightarrow$$

$$R_3 \mapsto \frac{ab}{b} R_3 - \frac{ab}{a} R_1$$

# Gröbner Basis - Triangularisation!

## Abstracted Polynomials

$$\begin{aligned}
P_1(x_i, y_i) &= & L_1x_1 + L_2x_2 + L_3x_3 - x &= 0 \\
P_2(x_i, y_i) &= & L_1y_1 + L_2y_2 + L_3y_3 - y &= 0 \\
P_3(x_i, y_i) &= & x_1^2 + y_1^2 - 1 &= 0 \\
P_4(x_i, y_i) &= & x_2^2 + y_2^2 - 1 &= 0 \\
P_5(x_i, y_i) &= & x_3^2 + y_3^2 - 1 &= 0
\end{aligned}$$

## Compute S-Polynomial

$$S(P_i, P_j) = \frac{\mathrm{lcm}(\mathrm{LT}(P_i), \mathrm{LT}(P_j))}{\mathrm{LT}(P_i)} P_i - \frac{\mathrm{lcm}(\mathrm{LT}(P_i), \mathrm{LT}(P_j))}{\mathrm{LT}(P_j)} P_j$$

# Gröbner Basis - Triangularisation!

## Abstracted Polynomials

$$\begin{aligned}
P_1(x_i, y_i) &= \quad L_1x_1 + L_2x_2 + L_3x_3 - x &= 0\\
P_2(x_i, y_i) &= \quad L_1y_1 + L_2y_2 + L_3y_3 - y &= 0\\
P_3(x_i, y_i) &= \quad x_1^2 + y_1^2 - 1 &= 0\\
P_4(x_i, y_i) &= \quad x_2^2 + y_2^2 - 1 &= 0\\
P_5(x_i, y_i) &= \quad x_3^2 + y_3^2 - 1 &= 0
\end{aligned}$$

## S-Polynomial Example

$$S(P_1, P_2) = \frac{\mathrm{lcm}(\mathrm{LT}(P_1), \mathrm{LT}(P_2))}{\mathrm{LT}(P_1)} P_i - \frac{\mathrm{lcm}(\mathrm{LT}(P_1), \mathrm{LT}(P_2))}{\mathrm{LT}(P_2)} P_2$$

$$= \frac{\cancel{L_3}\cancel{x_3}y_3}{\cancel{L_3}\cancel{x_3}} P_1 - \frac{\cancel{L_3}x_3\cancel{y_3}}{\cancel{L_3}\cancel{y_3}} P_= \begin{array}{l} y_3(L_1x_1 + L_2x_2 + L_3\cancel{x_3} - x) \\ -x_3(L_1y_1 + L_2y_2 + L_3\cancel{y_3} - y) \end{array}$$

# Gröbner Basis - Triangularisation!

```python
import sympy as sp
x1,x2,x3,y1,y2,y3 = sp.Symbol("x1"), sp.Symbol("x2"), sp.Symbol("x3"), sp.Symbol("y1"), sp.Symbol("y2"), sp.Symbol("y3")
L1,L2,L3 = 1,1,1
x,y = sp.Symbol("x"), sp.Symbol("y")
F = [L1*x1+L2*x2+L3*x3 - x,
     L1*y1+L2*y2+L3*y3 - y,
     x1**2 + y1**2 - 1,
     x2**2 + y2**2 - 1,
     x3**2 + y3**2 - 1]

basis = sp.GroebnerBasis(F, x1, x2, x3, y1, y2, y3, order='lex')
print(basis)
```

```
GroebnerBasis([-x + x1 + x2 + x3, x2**2 + y2**2 - 1, x**2 - 2*x*x2 - 2*x*x3 + 2*x2*x3 + y**2 - 2*y*y2 - 2*y*y3 + 2*y2*y3 + 1, x**4*y + 2*x**2*y**3 + 2*x**2*y + x2*y2*(4*x**3 + 4*x*y**2 - 4*x) + x2*y3
*(2*x**3 + 2*x*y**2 - 2*x) + x2*(-2*x**3*y - 2*x*y**3 + 2*x*y) + x3*y2*(2*x**3 + 2*x*y**2 - 2*x) + x3*y3*(4*x**3 + 4*x*y**2 - 4*x) + x3*(-2*x**3*y - 2*x*y**3 + 2*x*y) + y**5 + 2*y**3 - 3*y + y2**2*y3
*(-8*x**2 - 8*y**2) + y2**2*(4*x**2*y + 4*y**3 + 4*y) + y2*y3**2*(-8*x**2 - 8*y**2) + y2*y3*(12*x**2*y + 12*y**3 + 4*y) + y2*(-2*x**4 - 6*x**2*y**2 + 2*x**2 - 4*y**4 - 4*y**2) + y3**2*(4*x**2*y + 4*y
**3 + 4*y) + y3*(-2*x**4 - 6*x**2*y**2 + 2*x**2 - 4*y**4 - 4*y**2), -x**3 - x*y**2 + 2*x*y*y2 + 2*x*y*y3 - 2*x*y2*y3 - 2*x*y3**2 + x + 2*x2*y3**2 + x2*(2*x**2 - 2) + 2*x3*y*y2 + 2*x3*y*y3 - 2*x3*y2*y
3 + x3*(x**2 - y**2 - 1), x3**2 + y3**2 - 1, -x**4 - 2*x**2*y**2 - 2*x**2 - 8*x*x3*y*y2 - 8*x*x3*y*y3 + 8*x*x3*y2**2 + 8*x*x3*y2*y3 + x3*(4*x**3 + 4*x*y**2 - 4*x) - y**4 - 2*y**2 + 8*y*y2**2*y3 + 8*y
*y2*y3**2 + y2**2*(-4*x**2 - 4*y**2 - 4) + y2*y3*(-4*x**2 - 12*y**2 - 4) + y2*(4*x**2*y + 4*y**3 + 4*y) + y3**2*(4*x**2 - 4*y**2 - 4) + y3*(4*x**2*y + 4*y**3 + 4*y) + 3, -x**6 - 3*x**4*y**2 + 5*x**4
 - 3*x**2*y**4 + 2*x**2*y**2 - 7*x**2 + x3*y3**2*(8*x**3 + 8*x*y**2 - 8*x) + x3*y3*(-8*x**3*y - 8*x*y**3 + 8*x*y) + x3*(2*x**5 + 4*x**3*y**2 - 4*x**3 + 2*x*y**4 - 4*x*y**2 + 2*x) - y**6 - 3*y**4 + y**
2 + y2**2*y3**2*(-16*x**2 - 16*y**2) + y2**2*y3*(16*x**2*y + 16*y**3 + 16*y) + y2**2*(-4*x**4 - 8*x**2*y**2 + 8*x**2 - 4*y**4 - 8*y**2 - 4) + y2*y3**3*(-16*x**2 - 16*y**2) + y2*y3**2*(32*x**2*y + 32*
y**3 + 16*y) + y2*y3*(-4*x**4 - 24*x**2*y**2 + 8*x**2 - 20*y**4 - 24*y**2 - 4) + y2*(4*x**4*y + 8*x**2*y**3 - 8*x**2*y + 4*y**5 + 8*y**3 + 4*y) + y3**3*(8*x**2*y + 8*y**3 + 8*y) + y3**2*(-4*x**4 - 16
*x**2*y**2 + 8*x**2 - 12*y**4 - 16*y**2 - 4) + y3*(6*x**4*y + 12*x**2*y**3 - 4*x**2*y + 6*y**5 + 12*y**3 - 2*y) + 3, -y + y1 + y2 + y3, x**8 + 4*x**6*y**2 - 12*x**6 + 6*x**4*y**4 - 20*x**4*y**2 + 30*
x**4 + 4*x**2*y**6 - 4*x**2*y**4 + 28*x**2*y**2 - 28*x**2 + y**8 + 4*y**6 - 2*y**4 - 12*y**2 + y2**4*y3**2*(64*x**2 + 64*y**2) + y2**4*y3*(-64*x**2*y - 64*y**3 - 64*y) + y2**4*(16*x**4 + 32*x**2*y**2
 - 32*x**2 + 16*y**4 + 32*y**2 + 16) + y2**3*y3**3*(128*x**2 + 128*y**2) + y2**3*y3**2*(-256*x**2*y - 256*y**3 - 128*y) + y2**3*y3*(32*x**4 + 192*x**2*y**2 - 64*x**2 + 160*y**4 + 192*y**2 + 32) + y2*
*3*(-32*x**4*y - 64*x**2*y**3 + 64*x**2*y - 32*y**5 - 64*y**3 - 32*y) + y2**2*y3**4*(64*x**2 + 64*y**2) + y2**2*y3**3*(-256*x**2*y - 256*y**3 - 128*y) + y2**2*y3**2*(48*x**4 + 352*x**2*y**2 - 96*x**2
 + 304*y**4 + 288*y**2 + 48) + y2**2*y3*(-80*x**4*y - 224*x**2*y**3 + 96*x**2*y - 144*y**5 - 224*y**3 - 16*y) + y2**2*(8*x**6 + 40*x**4*y**2 - 40*x**4 + 56*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 24*y**
6 + 56*y**4 + 8*y**2 - 24) + y2*y3**4*(-64*x**2*y - 64*y**3 - 64*y) + y2*y3**3*(32*x**4 + 192*x**2*y**2 - 64*x**2 + 160*y**4 + 192*y**2 + 32) + y2*y3**2*(-80*x**4*y - 224*x**2*y**3 + 96*x**2*y - 144*
y**5 - 224*y**3 - 16*y) + y2*y3*(8*x**6 + 72*x**4*y**2 - 40*x**4 + 120*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 56*y**6 + 120*y**4 - 24*y**2 - 24) + y2*(-8*x**6*y - 24*x**4*y**3 + 40*x**4*y - 24*x**2*y**
5 + 16*x**2*y**3 - 56*x**2*y - 8*y**7 - 24*y**5 + 8*y**3 + 24*y) + y3**4*(16*x**4 + 32*x**2*y**2 - 32*x**2 + 16*y**4 + 32*y**2 + 16) + y3**3*(-32*x**4*y - 64*x**2*y**3 + 64*x**2*y - 32*y**5 - 64*y**3
 - 32*y) + y3**2*(8*x**6 + 40*x**4*y**2 - 40*x**4 + 56*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 24*y**6 + 56*y**4 + 8*y**2 - 24) + y3*(-8*x**6*y - 24*x**4*y**3 + 40*x**4*y - 24*x**2*y**5 + 16*x**2*y**3 -
 56*x**2*y - 8*y**7 - 24*y**5 + 8*y**3 + 24*y) + 9], x1, x2, x3, y1, y2, y3, domain='ZZ[x,y]', order='lex')
```

# Gröbner Basis - Triangularisation!

```
GroebnerBasis([-x + x1 + x2 + x3, x2**2 + y2**2 - 1, x**2 - 2*x*x2 - 2*x*x3 + 2*x2*x3 + y**2 - 2*y*y2 - 2*y*y3 + 2*y2*y3 + 1, x**4*y + 2*x**2*y**3 + 2*x**2*y + x2*y2*(4*x**3 + 4*x*y**2 - 4*x) + x2*y3
*(2*x**3 + 2*x*y**2 - 2*x) + x2*(-2*x**3*y - 2*x*y**3 + 2*x*y) + x3*y2*(2*x**3 + 2*x*y**2 - 2*x) + x3*y3*(4*x**3 + 4*x*y**2 - 4*x) + x3*(-2*x**3*y - 2*x*y**3 + 2*x*y) + y**5 + 2*y**3 - 3*y + y2**2*y3
*(-8*x**2 - 8*y**2) + y2**2*(4*x**2*y + 4*y**3 + 4*y) + y2*y3**2*(-8*x**2 - 8*y**2) + y2*y3*(12*x**2*y + 12*y**3 + 4*y) + y2*(-2*x**4 - 6*x**2*y**2 + 2*x**2 - 4*y**4 - 4*y**2) + y3**2*(4*x**2*y + 4*y
**3 + 4*y) + y3*(-2*x**4 - 6*x**2*y**2 + 2*x**2 - 4*y**4 - 4*y**2), -x**3 - x*y**2 + 2*x*y*y2 + 2*x*y*y3 - 2*x*y2*y3 - 2*x*y3**2 + x + 2*x2*y3**2 + x2*(2*x**2 - 2) + 2*x3*y*y2 + 2*x3*y*y3 - 2*x3*y2*y
3 + x3*(x**2 - y**2 - 1), x3**2 + y3**2 - 1, -x**4 - 2*x**2*y**2 - 2*x**2 - 8*x*x3*y*y2 - 8*x*x3*y*y3 + 8*x*x3*y2**2 + 8*x*x3*y2*y3 + x3*(4*x**3 + 4*x*y**2 - 4*x) - y**4 - 2*y**2 + 8*y*y2**2*y3 + 8*y
*y2*y3**2 + y2**2*(-4*x**2 - 4*y**2 - 4) + y2*y3*(-4*x**2 - 12*y**2 - 4) + y2*(4*x**2*y + 4*y**3 + 4*y) + y3**2*(4*x**2 - 4*y**2 - 4) + y3*(4*x**2*y + 4*y**3 + 4*y) + 3, -x**6 - 3*x**4*y**2 + 5*x**4
 - 3*x**2*y**4 + 2*x**2*y**2 - 7*x**2 + x3*y3**2*(8*x**3 + 8*x*y**2 - 8*x) + x3*y3*(-8*x**3*y - 8*x*y**3 + 8*x*y) + x3*(2*x**5 + 4*x**3*y**2 - 4*x**3 + 2*x*y**4 - 4*x*y**2 + 2*x) - y**6 - 3*y**4 + y**
2 + y2**2*y3**2*(-16*x**2 - 16*y**2) + y2**2*y3*(16*x**2*y + 16*y**3 + 16*y) + y2**2*(-4*x**4 - 8*x**2*y**2 + 8*x**2 - 4*y**4 - 8*y**2 - 4) + y2*y3**3*(-16*x**2 - 16*y**2) + y2*y3**2*(32*x**2*y + 32*
y**3 + 16*y) + y2*y3*(-4*x**4 - 24*x**2*y**2 + 8*x**2 - 20*y**4 - 24*y**2 - 4) + y2*(4*x**4*y + 8*x**2*y**3 - 8*x**2*y + 4*y**5 + 8*y**3 + 4*y) + y3**3*(8*x**2*y + 8*y**3 + 8*y) + y3**2*(-4*x**4 - 16
*x**2*y**2 + 8*x**2 - 12*y**4 - 16*y**2 - 4) + y3*(6*x**4*y + 12*x**2*y**3 - 4*x**2*y + 6*y**5 + 12*y**3 - 2*y) + 3, -y + y1 + y2 + y3, x**8 + 4*x**6*y**2 - 12*x**6 + 6*x**4*y**4 - 20*x**4*y**2 + 30*
x**4 + 4*x**2*y**6 - 4*x**2*y**4 + 28*x**2*y**2 - 28*x**2 + y**8 + 4*y**6 - 2*y**4 - 12*y**2 + y2**4*y3**2*(64*x**2 + 64*y**2) + y2**4*y3*(-64*x**2*y - 64*y**3 - 64*y) + y2**4*(16*x**4 + 32*x**2*y**2
 - 32*x**2 + 16*y**4 + 32*y**2 + 16) + y2**3*y3**3*(128*x**2 + 128*y**2) + y2**3*y3**2*(-256*x**2*y - 256*y**3 - 128*y) + y2**3*y3*(32*x**4 + 192*x**2*y**2 - 64*x**2 + 160*y**4 + 192*y**2 + 32) + y2*
*3*(-32*x**4*y - 64*x**2*y**3 + 64*x**2*y - 32*y**5 - 64*y**3 - 32*y) + y2**2*y3**4*(64*x**2 + 64*y**2) + y2**2*y3**3*(-256*x**2*y - 256*y**3 - 128*y) + y2**2*y3**2*(48*x**4 + 352*x**2*y**2 - 96*x**2
 + 304*y**4 + 288*y**2 + 48) + y2**2*y3*(-80*x**4*y - 224*x**2*y**3 + 96*x**2*y - 144*y**5 - 224*y**3 - 16*y) + y2**2*(8*x**6 + 40*x**4*y**2 - 40*x**4 + 56*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 24*y**
6 + 56*y**4 + 8*y**2 - 24) + y2*y3**4*(-64*x**2*y - 64*y**3 - 64*y) + y2*y3**3*(32*x**4 + 192*x**2*y**2 - 64*x**2 + 160*y**4 + 192*y**2 + 32) + y2*y3**2*(-80*x**4*y - 224*x**2*y**3 + 96*x**2*y - 144*
y**5 - 224*y**3 - 16*y) + y2*y3*(8*x**6 + 72*x**4*y**2 - 40*x**4 + 120*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 56*y**6 + 120*y**4 - 24*y**2 - 24) + y2*(-8*x**6*y - 24*x**4*y**3 + 40*x**4*y - 24*x**2*y**
5 + 16*x**2*y**3 - 56*x**2*y - 8*y**7 - 24*y**5 + 8*y**3 + 24*y) + y3**4*(16*x**4 + 32*x**2*y**2 - 32*x**2 + 16*y**4 + 32*y**2 + 16) + y3**3*(-32*x**4*y - 64*x**2*y**3 + 64*x**2*y - 32*y**5 - 64*y**3
 - 32*y) + y3**2*(8*x**6 + 40*x**4*y**2 - 40*x**4 + 56*x**2*y**4 - 48*x**2*y**2 + 56*x**2 + 24*y**6 + 56*y**4 + 8*y**2 - 24) + y3*(-8*x**6*y - 24*x**4*y**3 + 40*x**4*y - 24*x**2*y**5 + 16*x**2*y**3 -
 56*x**2*y - 8*y**7 - 24*y**5 + 8*y**3 + 24*y) + 9], x1, x2, x3, y1, y2, y3, domain='ZZ[x,y]', order='lex')
```

## Triangular Form

$$-x + x_1 + x_2 + x_3 = 0$$

$$x_2^2 + y_2^2 - 1 = 0$$

$$x^2 - 2xx_2 - 2xx_3 + 2x_2x_3 + y^2 - 2yy_2 - 2yy_3 + 2y_2y_3 + 1 = 0$$

$$\vdots$$

# Gröbner Basis - Triangularisation!

## Result

$$\cos(\theta_1) = x_1(x, y)$$

$$\cos(\theta_1 + \theta_2) = x_2(x, y)$$

$$\cos(\theta_1 + \theta_2 + \theta_3) = x_3(x, y)$$

$$(\theta_1, \theta_2, \theta_3) \in F^{-1}[(x, y)]$$

# Lie Theory - Algebra to Geometry

## General Form of a Line

$$\ell : ax + by + c = 0$$

## Point Inclusion

$$P = (x_0, y_0)^T \in \ell \Leftrightarrow ax_0 + by_0 + c = 0$$

# Lie Theory - Algebra to Geometry

## General Form of a Line

$$\ell : ax + by + c = 0$$

## Point Inclusion

$$P = (x_0, y_0)^T \in \ell \Leftrightarrow ax_0 + by_0 + c = 0$$

## Modified Point Inclusion

$$\tilde{P} = (x_0, y_0, 1)^T \in \ell \Leftrightarrow (a, b, c) \cdot \tilde{P} = 0$$

# Lie Theory - Algebra to Geometry

## General Form of a Line

$$\ell : ax + by + c = 0$$

## Modified Point Inclusion

$$\tilde{P} = (x_0, y_0, 1)^T \in \ell \Leftrightarrow (a, b, c) \cdot \tilde{P} = 0$$

## Projective Points $P \in \mathbb{P}^2 = (\mathbb{R}^3 \setminus \{\mathbf{0}\}) / \sim$

$$[x : y : z]^T = [\lambda x : \lambda y : \lambda z]^T, \lambda \in \mathbb{R} \setminus \{0\}$$

## Projective Point Inclusion

$$\tilde{P} = [x_0 : y_0 : 1]^T \in \ell \Leftrightarrow [a : b : c] \cdot \tilde{P} = 0$$

# Lie Theory - Algebra to Geometry

## Duality of Lines and Points

$$\ell = [a : b : c]$$

$$P = \ell^* = [a : b : c]^T = \left[\frac{a}{c} : \frac{b}{c} : 1\right]^T \text{ if } c \neq 0$$

# Lie Theory - Algebra to Geometry

## Intersection of Two Lines

$$\ell_1 = [a : b : c]$$

$$\ell_2 = [a' : b' : c']$$

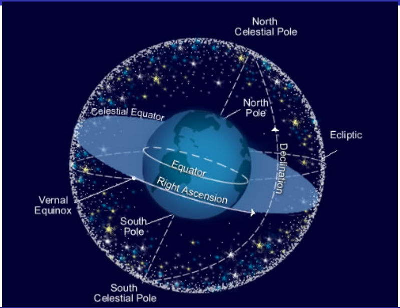
$$P = (\ell_1^* \times \ell_2^*)^*$$

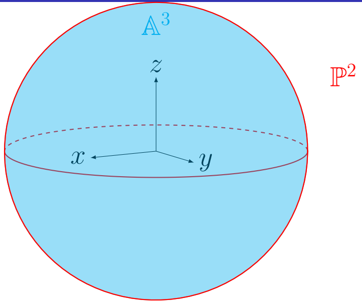
# Lie Theory - Algebra to Geometry

## Points at Infinity

$$\mathbb{P}^2 = \underbrace{\mathbb{A}^2}_{[x:y:1]} \sqcup \underbrace{\mathbb{P}^1}_{[x:y:0]}$$

# Lie Theory - Algebra to Geometry




## Points at Infinity

$$\mathbb{P}^3 = \underbrace{\mathbb{A}^3}_{[x:y:z:1]} \sqcup \underbrace{\mathbb{P}^2}_{[x:y:z:0]}$$

$$\lim_{x\to\infty} [x : y : z : 1] = \lim_{x\to\infty} \left[1 : \frac{y}{x} : \frac{z}{x} : \frac{1}{x}\right] = [1 : 0 : 0 : 0]$$

# Lie Theory - Special Euclidean Group

## Special Orthogonal Group

$$\mathrm{SO}(3) = \left\{ R \in \mathrm{Mat}_3(\mathbb{R}) \,\middle|\, R^T R = \mathbb{I}, \det R = 1 \right\}$$

$$\mathrm{SO}(3) \curvearrowright \mathbb{E}^3$$

## Special Euclidean Group

$$\mathrm{SE}(3) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathrm{Mat}_4(\mathbb{R}) \,\middle|\, R \in \mathrm{SO}(3), t \in \mathbb{R}^3 \right\}$$

$$\mathrm{SE}(3) \curvearrowright \mathbb{P}^3 = \mathbb{E}^3 \sqcup \mathbb{P}^2$$

# Lie Theory - Special Euclidean Group

## Special Euclidean Group

$$\mathrm{SE}(3) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathrm{Mat}_4(\mathbb{R}) \middle| R \in \mathrm{SO}(3), t \in \mathbb{R}^3 \right\}$$

$$\mathrm{SE}(3) \curvearrowright \mathbb{P}^3 = \mathbb{E}^3 \sqcup \mathbb{P}^2$$

# Lie Theory - Special Euclidean Group Lie Algebra

## Special Euclidean Group

$$\mathrm{SE}(3) = \left\{ \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \in \mathrm{Mat}_4(\mathbb{R}) \middle| R \in \mathrm{SO}(3), t \in \mathbb{R}^3 \right\}$$

$$\mathrm{SE}(3) \curvearrowright \mathbb{P}^3 = \mathbb{E}^3 \sqcup \mathbb{P}^2$$

$$T_g\mathrm{SE}(3) = \mathrm{span}\left\{ \frac{\partial}{\partial \alpha}(g), \frac{\partial}{\partial \beta}(g), \frac{\partial}{\partial \gamma}(g), \frac{\partial}{\partial x}(g), \frac{\partial}{\partial y}(g), \frac{\partial}{\partial z}(g) \right\}$$

## Lie Algebra of Special Euclidean Group

$$\mathfrak{se}(3) := T_{\mathbb{I}}\mathrm{SE}(3)$$

# Lie Theory - Special Euclidean Group Lie Algebra

## Lie Algebra of Special Euclidean Group

$$\mathfrak{se}(3) := T_{\mathbb{I}}SE(3)$$

## Velocity Vectors

$$V \in \mathfrak{se}(3)$$

$$V = c_1 \frac{\partial}{\partial \alpha}(\mathbb{I}) + c_2 \frac{\partial}{\partial \beta}(\mathbb{I}) + c_3 \frac{\partial}{\partial \gamma}(\mathbb{I}) + c_4 \frac{\partial}{\partial x}(\mathbb{I}) + c_5 \frac{\partial}{\partial y}(\mathbb{I}) + c_6 \frac{\partial}{\partial z}(\mathbb{I})$$

# Lie Theory - Special Euclidean Group Lie Algebra

## Velocity Vectors

$$V \in \mathfrak{se}(3)$$

$$V = c_1 \frac{\partial}{\partial \alpha} + c_2 \frac{\partial}{\partial \beta} + c_3 \frac{\partial}{\partial \gamma} + c_4 \frac{\partial}{\partial x} + c_5 \frac{\partial}{\partial y} + c_6 \frac{\partial}{\partial z}$$

## Exponential Map

$$\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$$

$$\exp(V) = \gamma_V(1)$$

# Lie Theory - Lie Algebras for Motion Planning

## Exponential Map

$$\exp : \mathfrak{se}(3) \to \mathrm{SE}(3)$$

$$\exp(V) = \gamma_V(1), \gamma \text{ a geodesic with initial velocity } V \text{ at } \mathbb{I}$$

# Lie Theory - Lie Algebras for Motion Planning

https://www.youtube.com/watch?v=6Wmw4lUHlX8

# Others

- Category Theory for Robots (http://ames.caltech.edu/A%20categorical%20theory.pdf)
- Path Planning with Homotopy (https://ieeexplore.ieee.org/document/10598326)
- Optimisation

# Summary

- Kempe was inspired by robotics to pursue questions about curve-producing linkages. This led to more mathematics as his work was carried forward by Thurston and completed by Kapovich and Millson.
- Buchburger was solving problems in computational algebra with potential applications for robotics in mind. His work later became an important modern tool in robotics and mathematics.
- Projective geometry was invented to simplify proofs in mathematics. With improved numerical stability, it became an essential part of robotics. Lie Theory then generalised derivatives to manifolds, commonly present within a robot's state space.