Towards Choosing Consistent Geometric Constraints

F. C. Langbein A. D. Marshall
R. R. Martin

27th March 2002

Department of Computer Science
Cardiff University

Reverse Engineering

- Engineering converts a concept into an artifact
- Reverse engineering converts an artifact into a concept
- Desired result is a representation of the design intent, not a simple copy
- Reverse engineered models suffer from inaccuracies caused by
  - ⋆ sensing errors during data acquisition
  - ⋆ approximation and numerical errors from reconstruction algorithms
  - ⋆ possible wear of the artifact
  - ⋆ manufacturing method used to make the artifact

Beautification

- **Goal:** Reconstruct an *ideal* model of a physical object with intended geometric regularities
- Only for engineering objects with planar, spherical, cylindrical, conical and toroidal surfaces
- Previous approaches:
  - ⋆ Augment the surface fitting step by constraint solving methods [Fisher,Benkő]
  - ⋆ Manually identify features like slots and pockets and use them to drive the segmentation and surface fitting [Thompson]
- **Our approach:** Improve the model in a post-processing step called **beautification**

Beautification Strategy

Analyser

Detect potential regularities which are approximately present in the initial model

→

Hypothesizer

Solve a constraint system derived from the regularities which describes a complete, improved model with likely regularities

←

Reconstruction

Reconstruct an improved model, fix topological problems, align model with coordinate axes, etc.

Beautification Strategy

Analyser

Detect potential regularities which are approximately present in the initial model

→

Hypothesizer

Constraint Selection

Based on priorities and inconsistencies select a set of likely constraints

↓ ↑

Constraint Solver

Try to solve constraint system and indicate inconsistencies (solvability test)

←

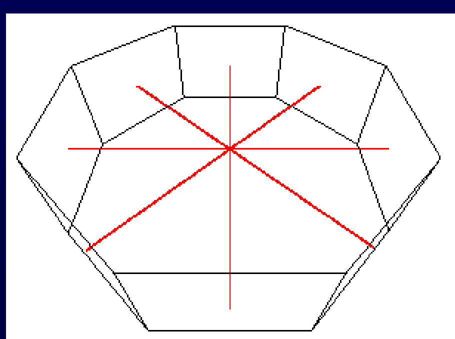
Reconstruction

Reconstruct an improved model, fix topological problems, align model with coordinate axes, etc.

Geometric Regularities

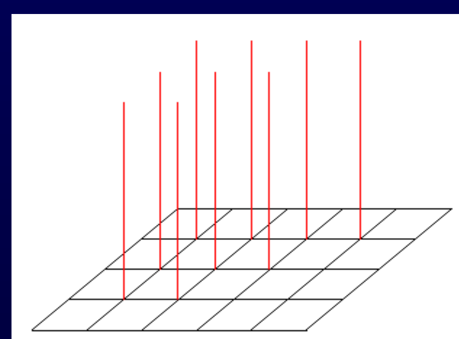
Directions

- Parallel directions
- Directions with same angle relative to a special direction
- Symmetrical arrangements of directions



Axes

- Axis intersections
- Aligned axes
- Parallel axes arranged equally spaced along lines, grids or on cylinders



Positions

- Equal positions
- Positions equal under projection
- Positions arranged regularly on lines and grids

Parameter

- Equal lengths
- Equal angles
- Integer relations
- Special values:
  - ⋆ integers
  - ⋆ simple fractions

Constraint Selection & Solving

- Beautification as constraint selection and solving problem
  - ⋆ Regularities become sets of geometric constraints
  - ⋆ Model topology is described by geometric constraints
- Algorithm has to select desirable constraints such that the constraint system has a solution
- Finding a solution is a secondary task

Selection & Solving Strategy I

I. Prioritize the constraints based on
  – how well they are satisfied in the initial model
  – how common and desirable the related regularity is

II. Select initial set S of constraints using selection rules considering the priorities:
  a. Resolve simple inconsistencies between constraints
     (constraints between the same objects with different constants)
  b. Only add constraints if constraints they depend on are also present
     (incidence hierarchies, low- to higher-level regularities, etc.)

Selection & Solving Strategy II

III. In order of highest to lowest priority remove a constraint c from S until S is empty also considering dependencies:

a. Try to add c to the selected constraint system C

b. If c could not be added, then ignore c and adjust S according to the selection rules

Central issues of the algorithm:

- Selection rules and priorities determine which regularities are favoured
- Step III.a determines the solvability and eventually a solution of the selected constraint system

Selection & Priorities

- Priorities determine locally which regularity to choose in case of an inconsistency, no global meaning
- Selection rules add additional structure to selection process in order to avoid arbitrary order
- Algorithmic details are explained elsewhere. . .
- Future improvements for selection:
  - ⋆ Make decisions in the context of the whole model, not just locally with respect to the regularities
  - ⋆ Reduce/replace user-defined constants for priorities by simpler methods based on multiple-choice questions and learning
  - ⋆ AI techniques, e.g. belief networks, neural networks

Constraint Solver & Solvability Test

- Given: consistent constraint system C,
  additional constraint c
- Problem: try to expand C to C' by adding c such that
  - ⋆ C' is consistent (has at least one solution)
  - ⋆ C' has less solutions than C (c is not redundant)
- Try to simplify C' by identifying and solving a sub-system
- Approach: degree-of-freedom analysis in a topological context
  - ⋆ Geometric objects are elements of abstract manifolds
  - ⋆ Constraints limit the allowed values for the objects to sub-manifolds

Geometric Constraint Problem

Elements of a Geometric Constraint Solving Problem:

M	Finite set of abstract manifolds representing types of geometric objects (set of all planes, set of all vertices, . . .)
O	Finite set of geometric objects (variables in the constraint system)
$t : O \to M$	Function assigning a type to each element of O, special type constraints
C	Finite set of geometric constraints
$f : O \to \bigcup_{m \in M} m$	Functions assigning a value to each object in O

Geometric Types

- M is a set of manifolds whose elements are geometric objects of a single type:
  - ⋆ Set of all vertices: $M_V \simeq \mathbb{R}^3$ (position)
  - ⋆ Set of all planes: $M_P \simeq \mathbb{R}^1 \times \mathbb{S}^2$ (distance & direction)
  - ⋆ Set of all spheres: $M_S \simeq \mathbb{R}^3 \times \mathbb{R}^+$ (position & radius)

    $\ldots$
- t assigns a type to each object $o \in O$:
  - ⋆ Type constraint requiring that $f(o) \in t(o)$

Vertex Distance Constraint

- Geometric objects: $o_1, o_2 \in O$
- Geometric types: $t(o_1) = t(o_2) = \mathbb{R}^3$

$$(f(o_1), f(o_2)) \in \mathbb{R}^3 \times \mathbb{R}^3$$

- Constraint: constant distance λ between vertices o_1, o_2

$$(f(o_1), f(o_2)) \in \{(x_1, x_2) : \|x_1 - x_2\| = \lambda\} =: c$$

$\star$ c is a sub-manifold of $\mathbb{R}^3 \times \mathbb{R}^3$

$\star$ c is homeomorphic to

(1) $\mathbb{R}^3 \times \mathbb{S}^2$: Choose first vertex freely, then the 2nd vertex is determined by a direction

(2) $\mathbb{S}^2 \times \mathbb{R}^3$: Analogously, $o_1 \leftrightarrow o_2$

$\longrightarrow$ Two options to interpret c as sub-manifold of $\mathbb{R}^3 \times \mathbb{R}^3$

Vertex on Plane Constraint

- Geometric objects: vertex $v \in O$, plane $p \in O$

$$(f(v), f(p)) \in \mathbb{R}^3 \times (\mathbb{R}^1 \times \mathbb{S}^2)$$

- Constraint: vertex v in plane p

$$(f(v), f(p)) \in \{(x_1, (x_2, x_3)) : x_3^t x_1 = x_2\} =: c$$

⋆ c as sub-manifold of $\mathbb{R}^3 \times (\mathbb{R}^1 \times \mathbb{S}^2)$ is homeomorphic to

(1) $\mathbb{R}^2 \times (\mathbb{R}^1 \times \mathbb{S}^2)$

- ◦ Choose an arbitrary plane
- ◦ Choose a point on the plane

(2) $\mathbb{R}^3 \times (\mathbb{R}^0 \times \mathbb{S}^2)$

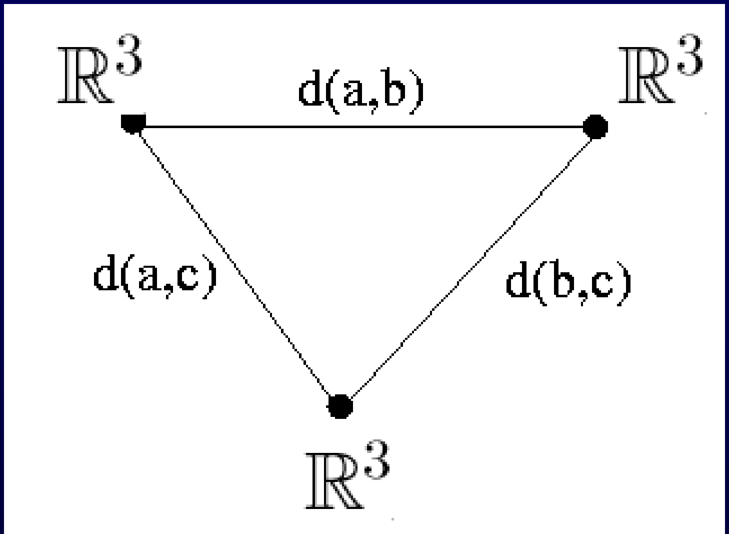
- ◦ Choose an arbitrary point
- ◦ Choose a normal for the plane through the point

General Geometric Constraints

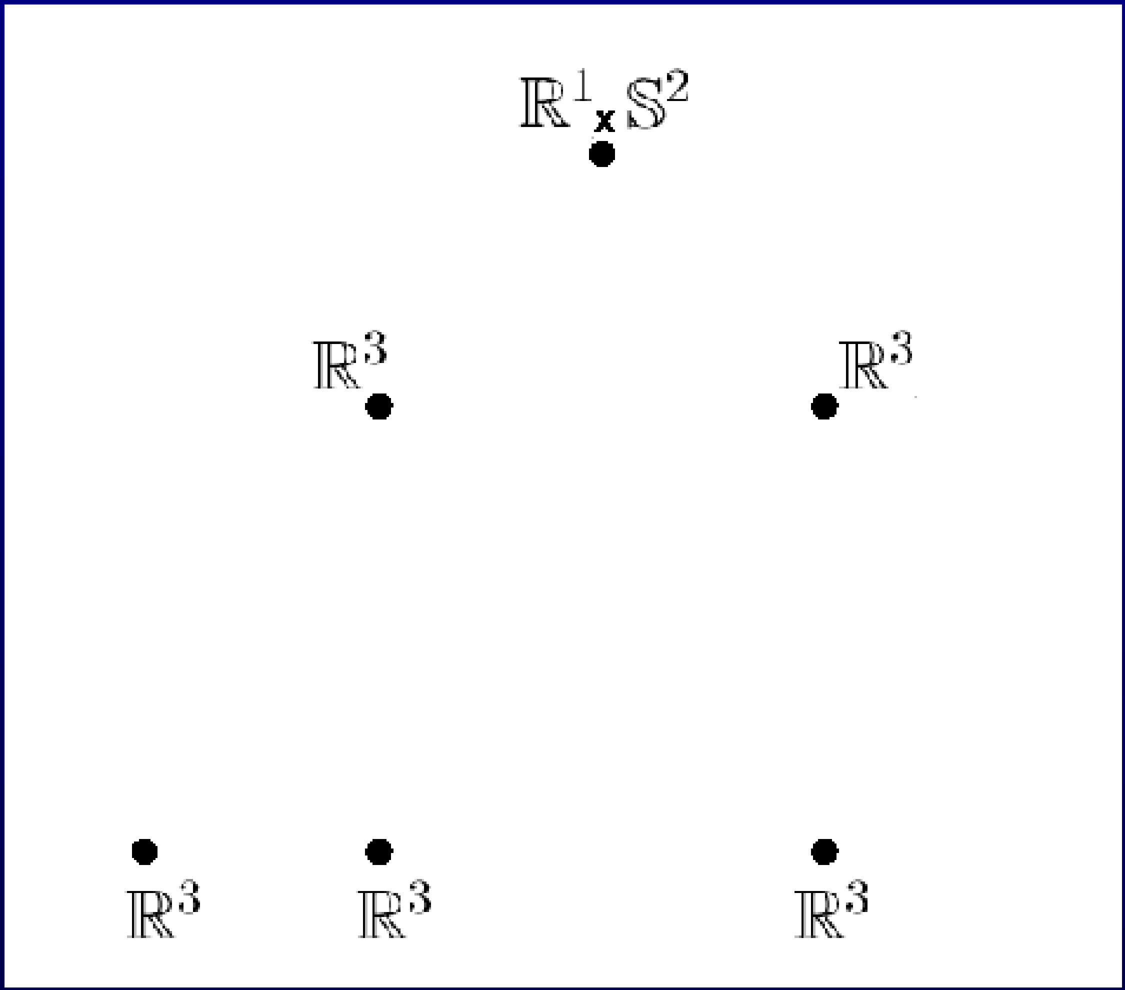
- General form of a constraint as pair $(o, c) \in C$:
  - $\star$ $o = (o_1, \ldots, o_n)$ is a tuple of geometric objects $o_l \in O$ which are involved in the constraint
  - $\star$ c is a set of allowed values, i.e. $(f(o_1), \ldots, f(o_n)) \in c$
  - $\star$ From the type constraints: $c \subset t(o_1) \times \cdots \times t(o_n) =: T$
  - $\star$ c is a sub-manifold of T
  - $\star$ For each c there is a set of product-manifolds s:
    - $*$ s has the same form than T, relates to the same objects
    - $*$ s is homeomorphic to c
- However, exact properties still unclear

Constraint Graph

- The constraints $(o, c) \in C$ define a (hyper-)graph:
  - ⋆ The geometric objects O are the vertices
  - ⋆ The types $t(o)$ label the vertices
  - ⋆ The o from the constraints are the edges
  - ⋆ The c are labels for the edges
- Graph for three distances between three vertices:

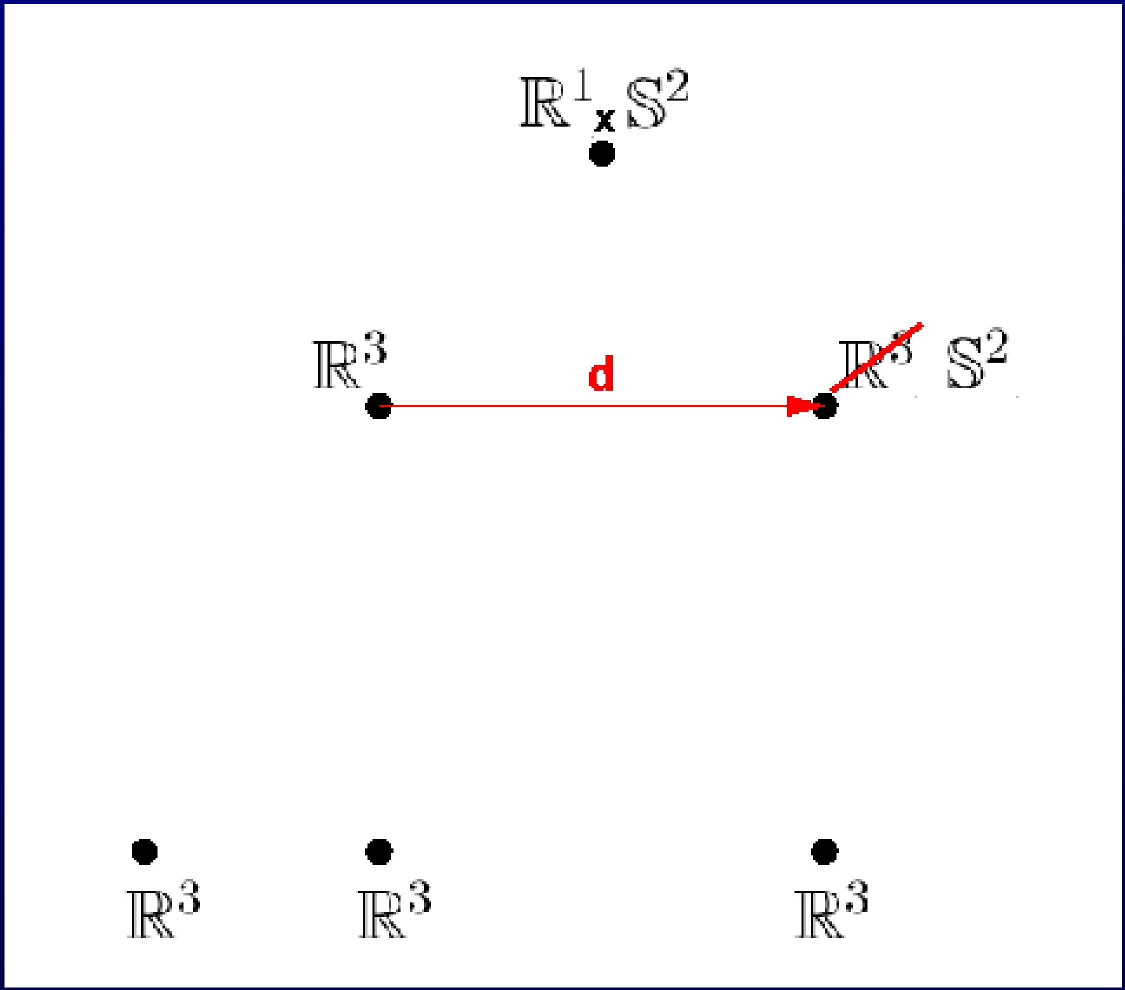


Constraint System Example



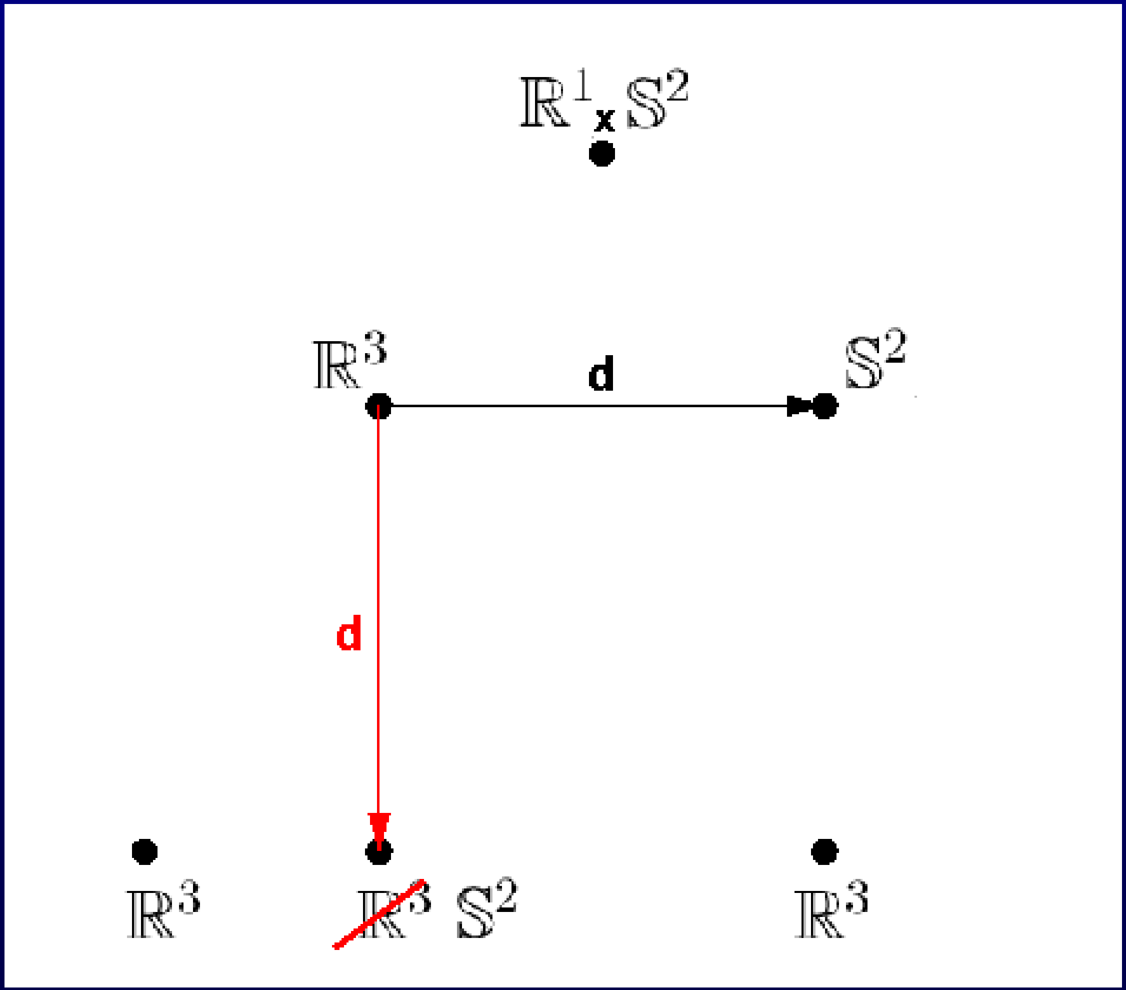
Simple example with 5 vertices, 1 plane

Constraint System Example



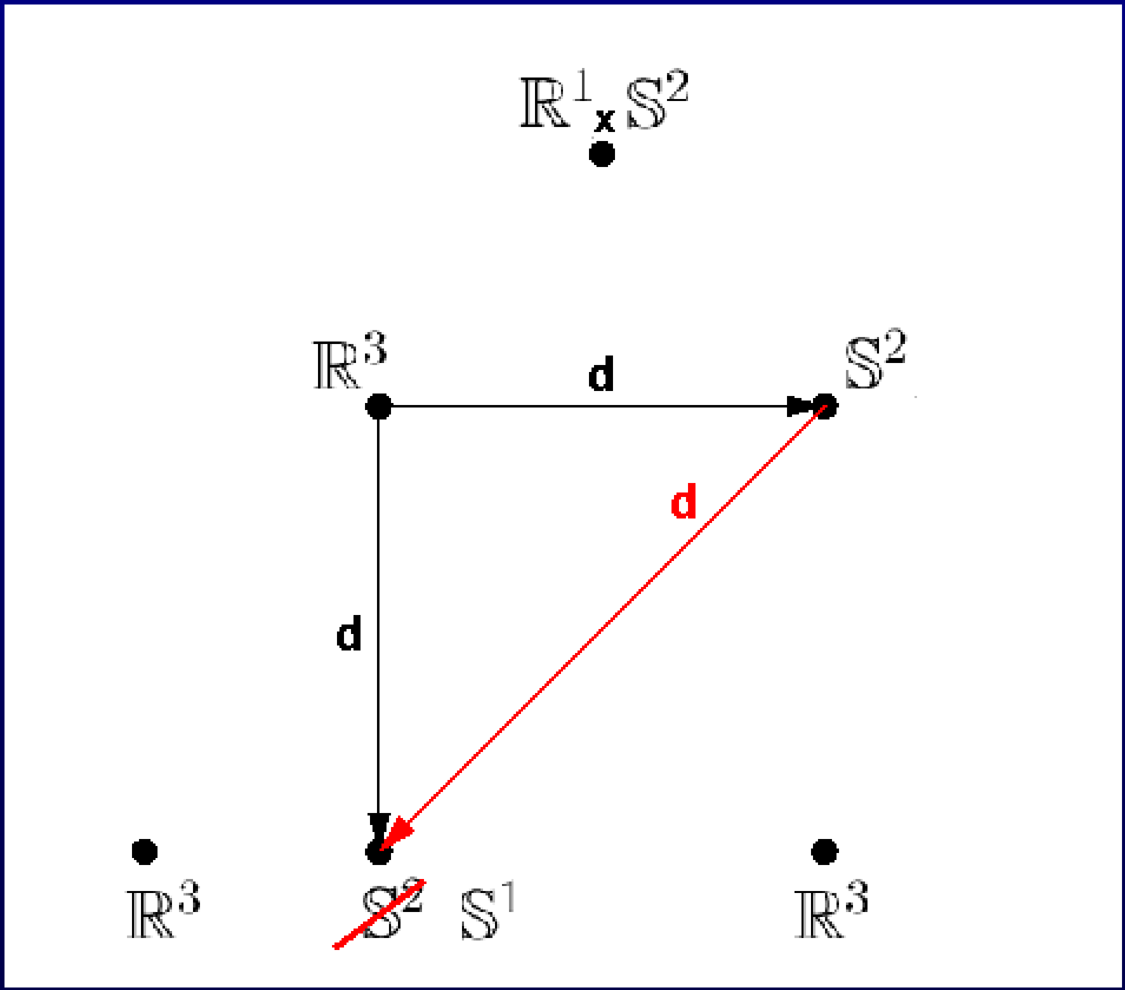
Adding distance constraint 1

Constraint System Example



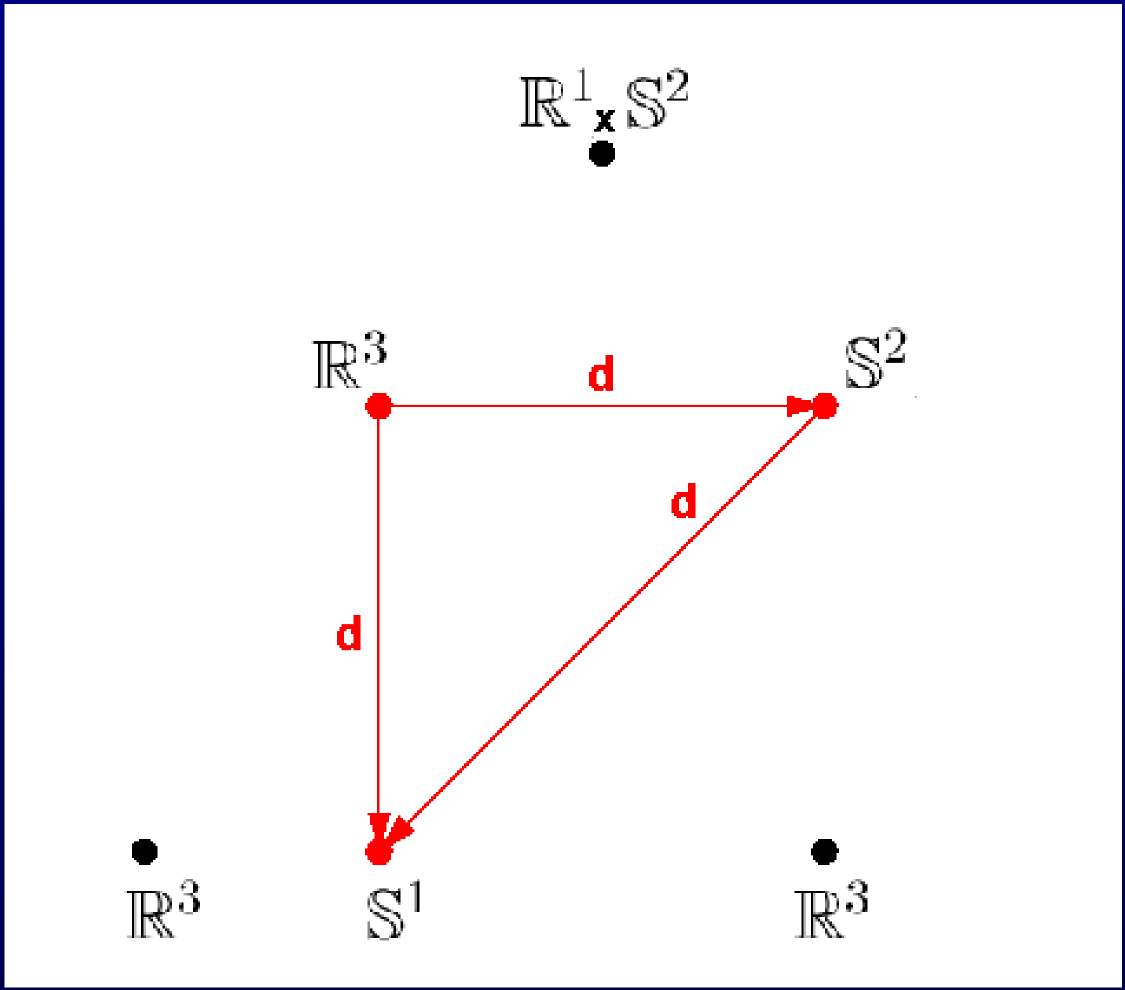
Adding distance constraint 2

Constraint System Example



Adding distance constraint 3

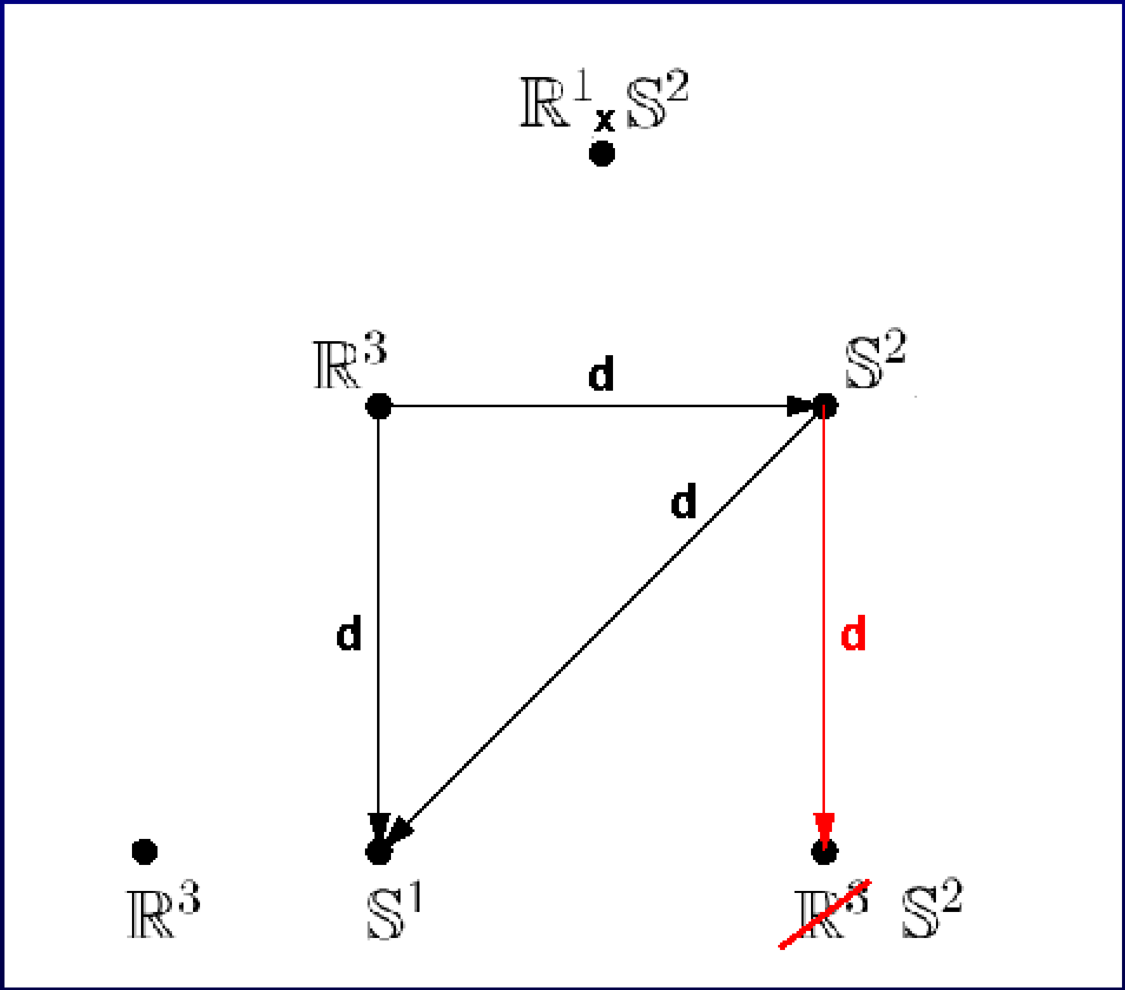
Constraint System Example



Solvable sub-system 1

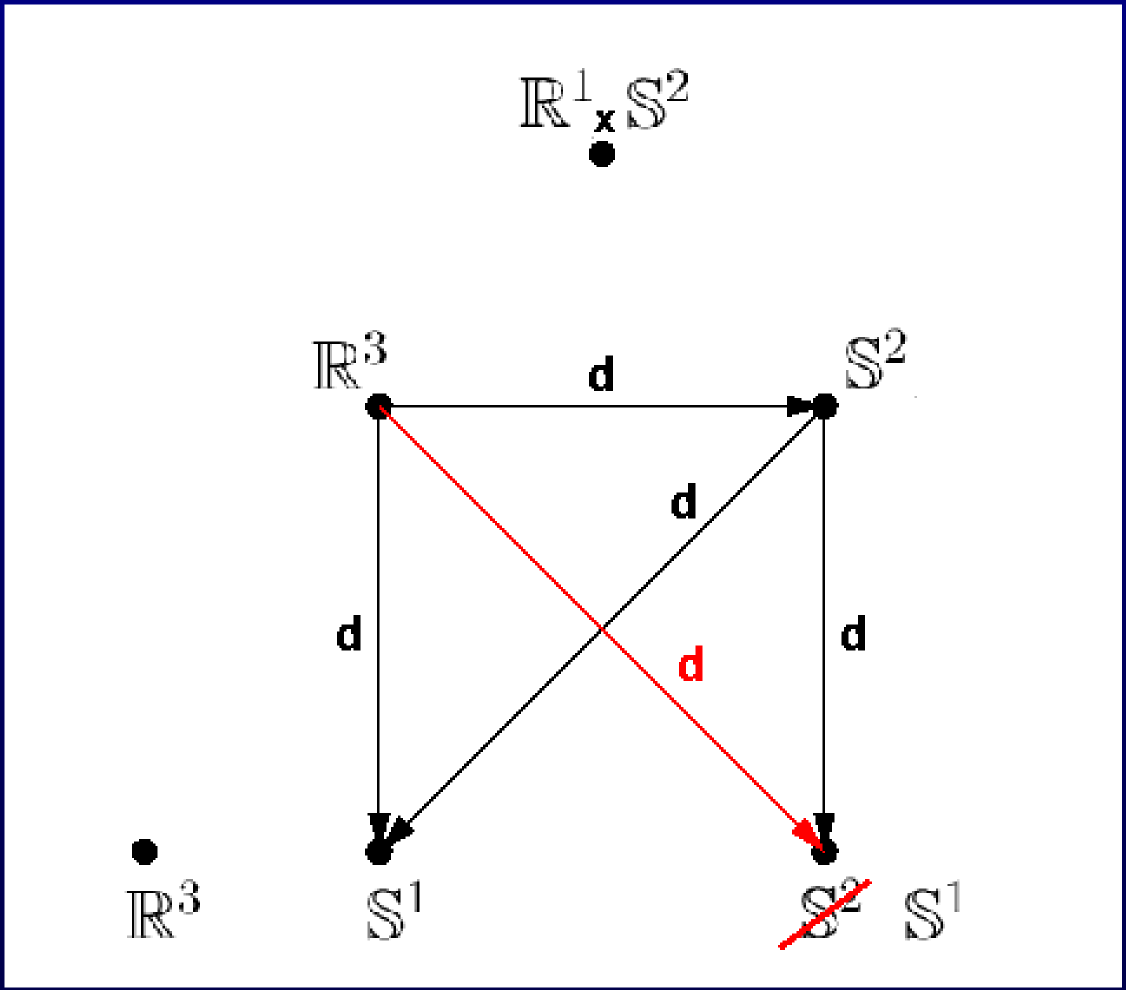
$\longrightarrow$ unique modulo rotations and translations

Constraint System Example



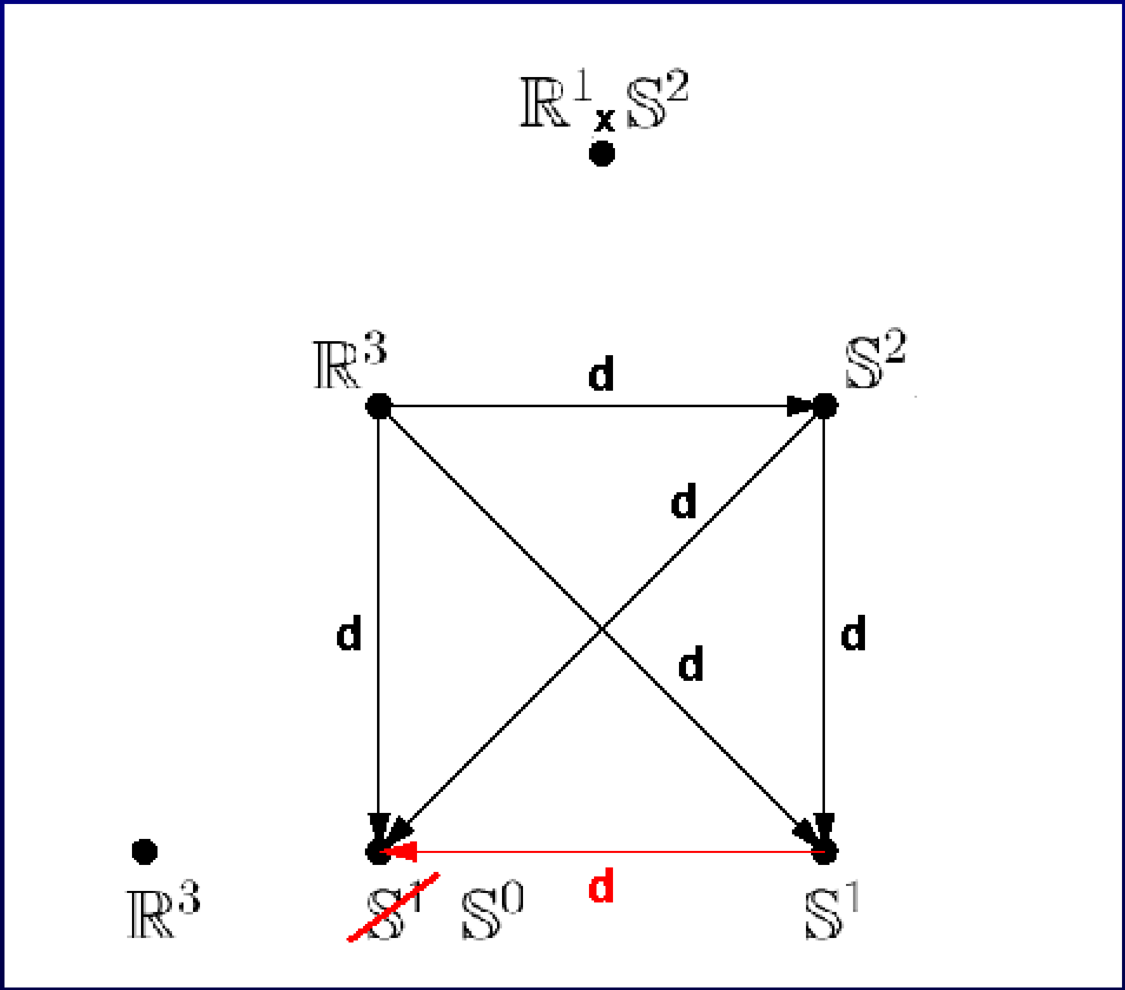
Adding distance constraint 4

Constraint System Example



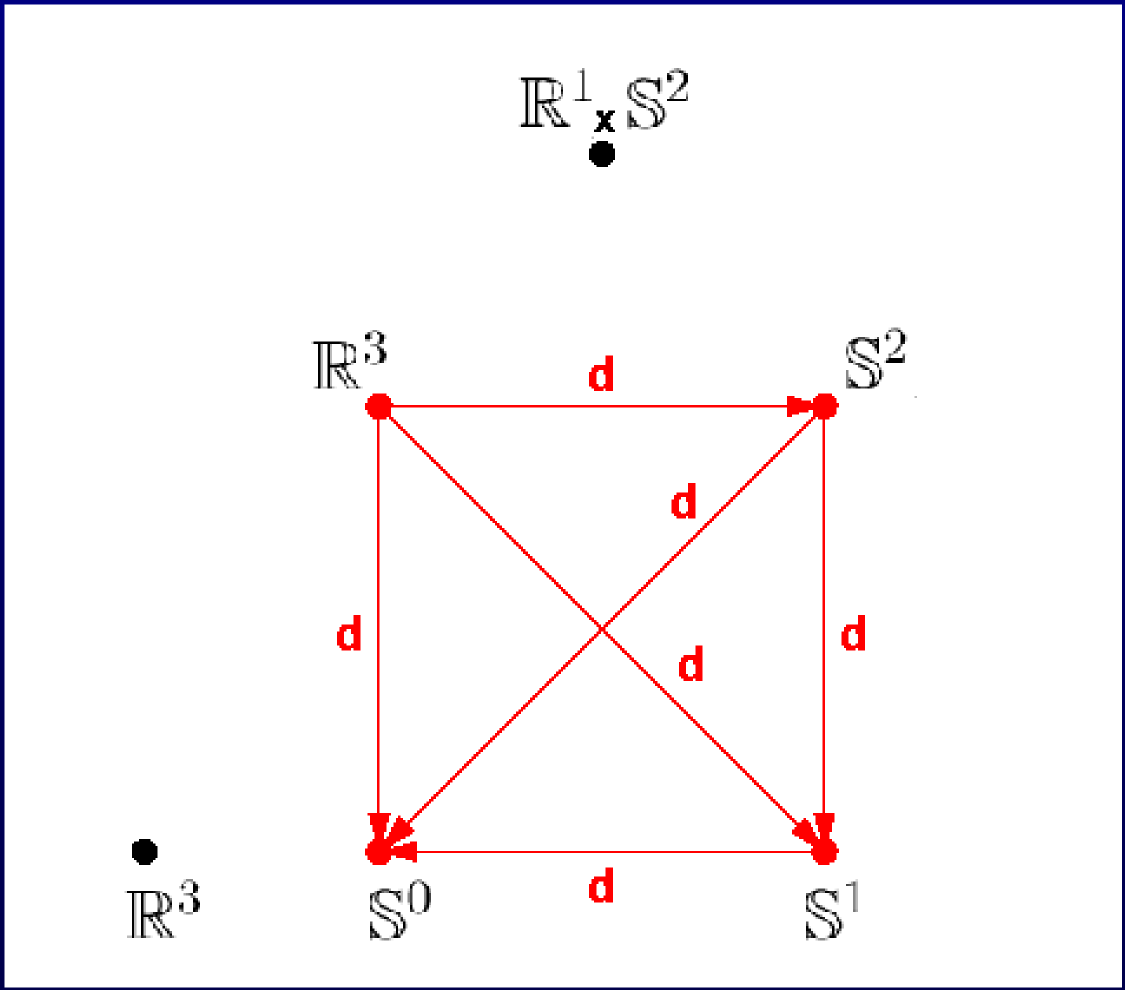
Adding distance constraint 5

Constraint System Example



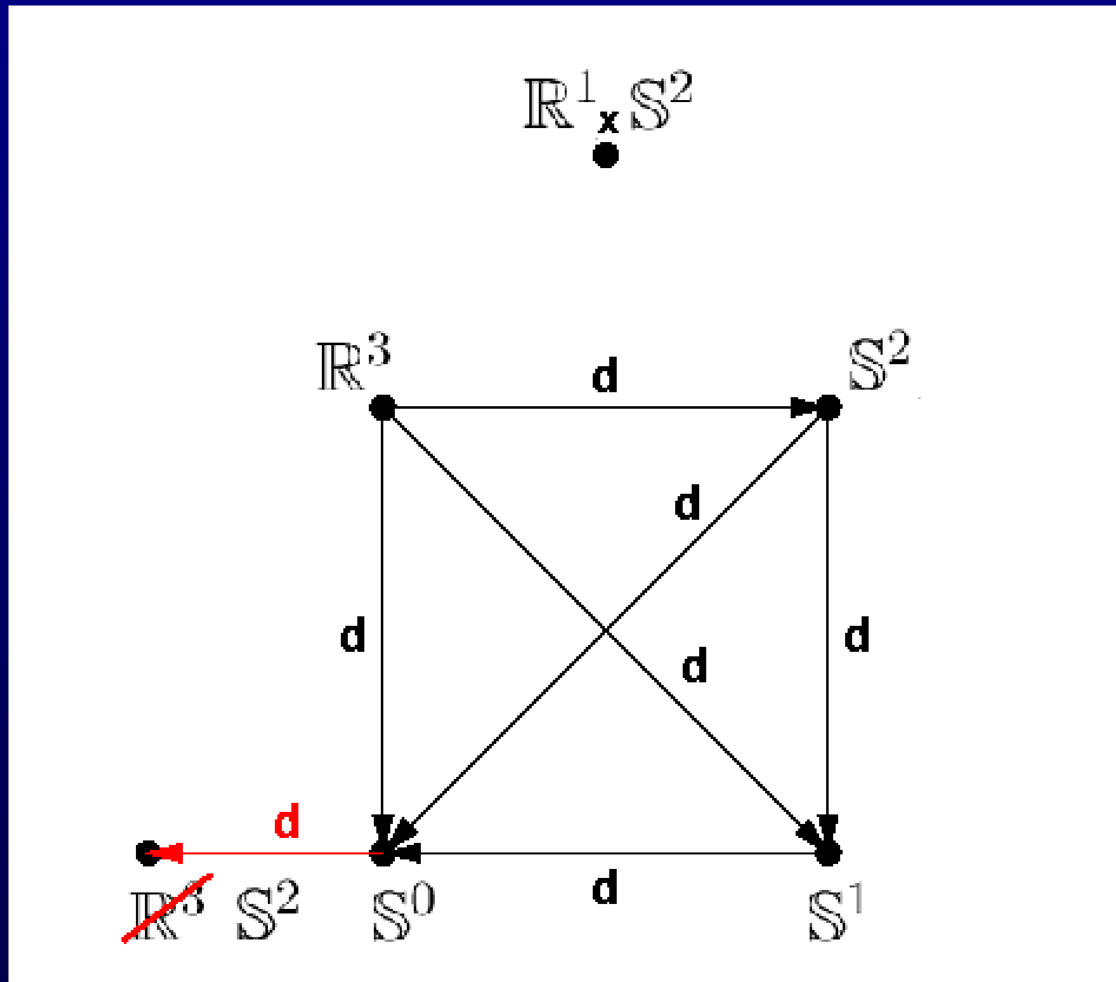
Adding distance constraint 6

Constraint System Example



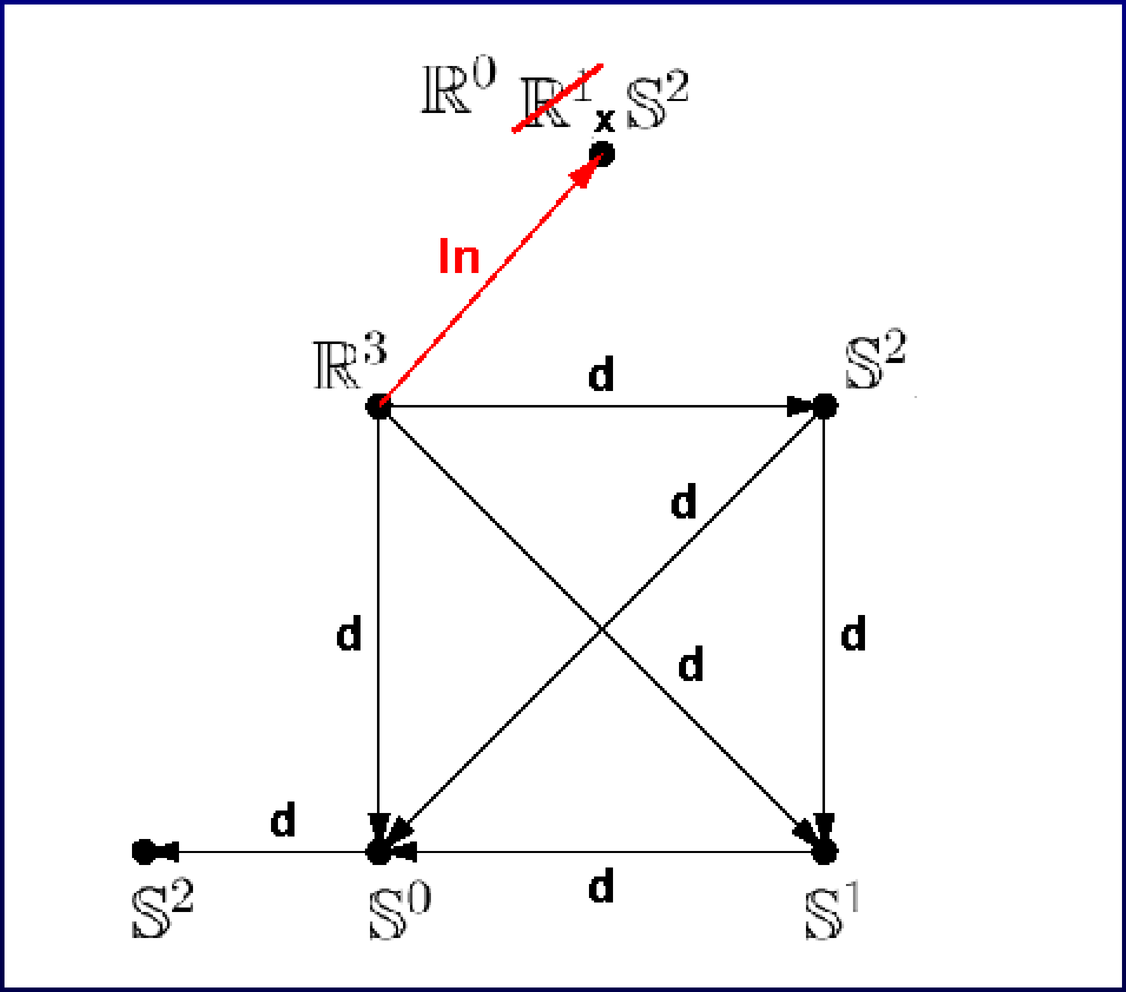
Solvable sub-system 2

Constraint System Example



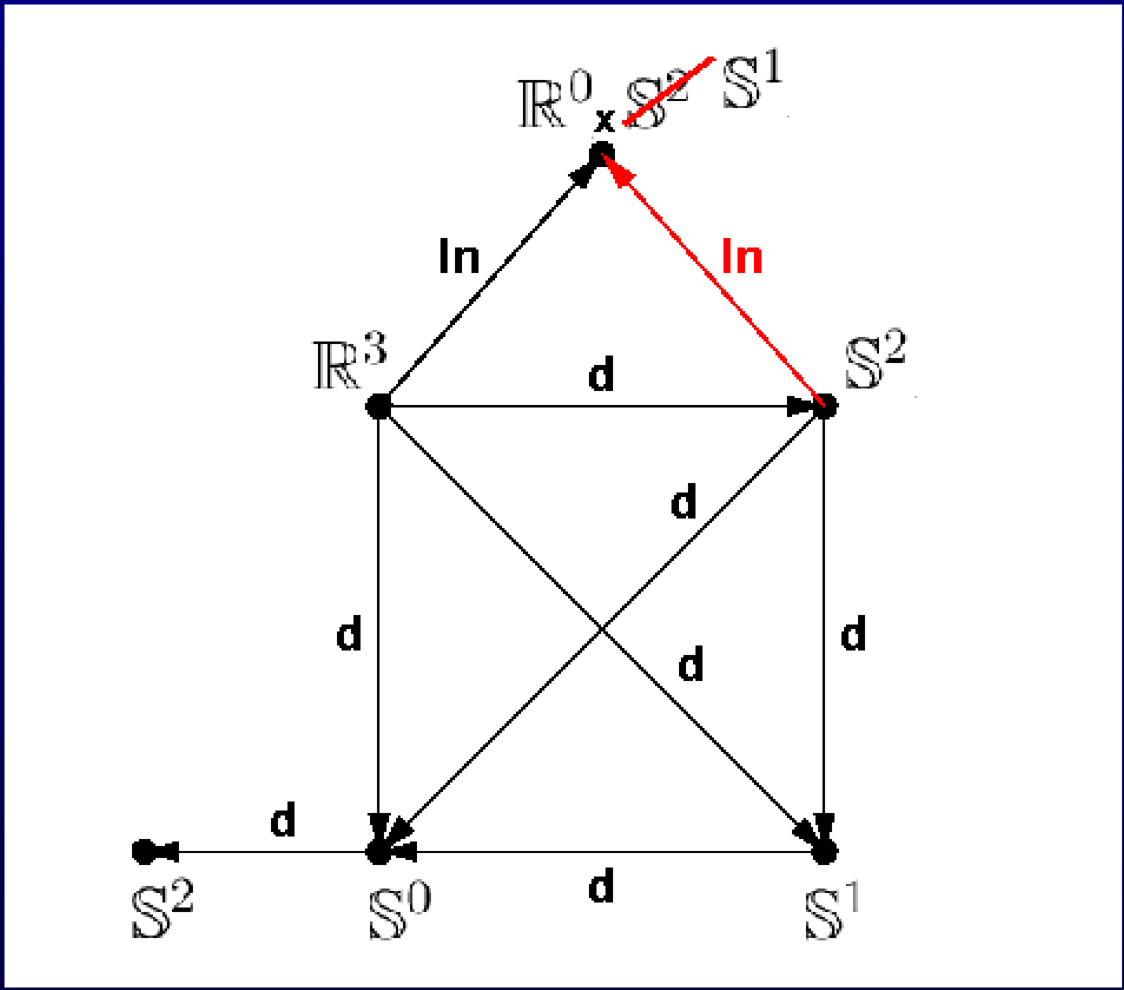
Adding distance constraint 7

Constraint System Example



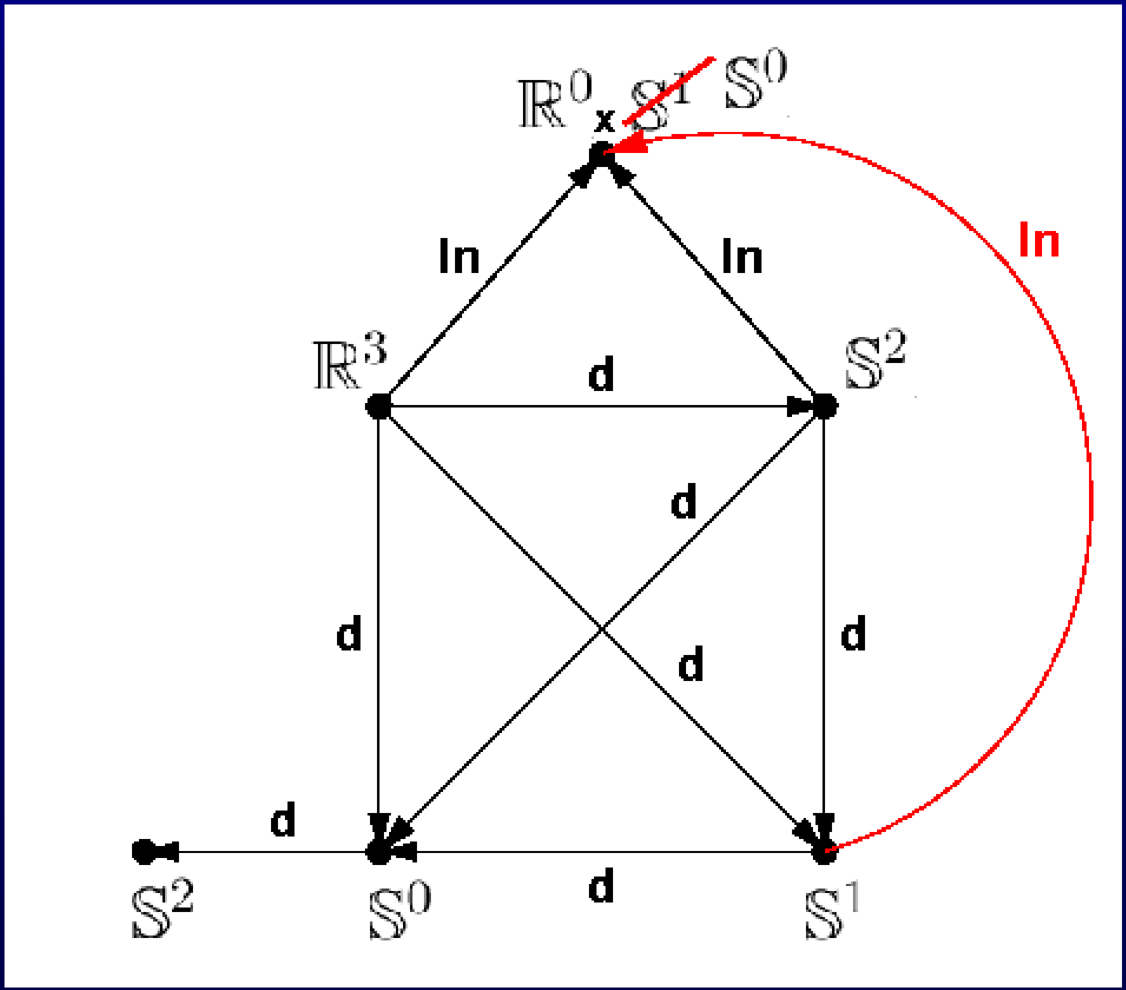
Adding vertex on plane constraint 1

Constraint System Example



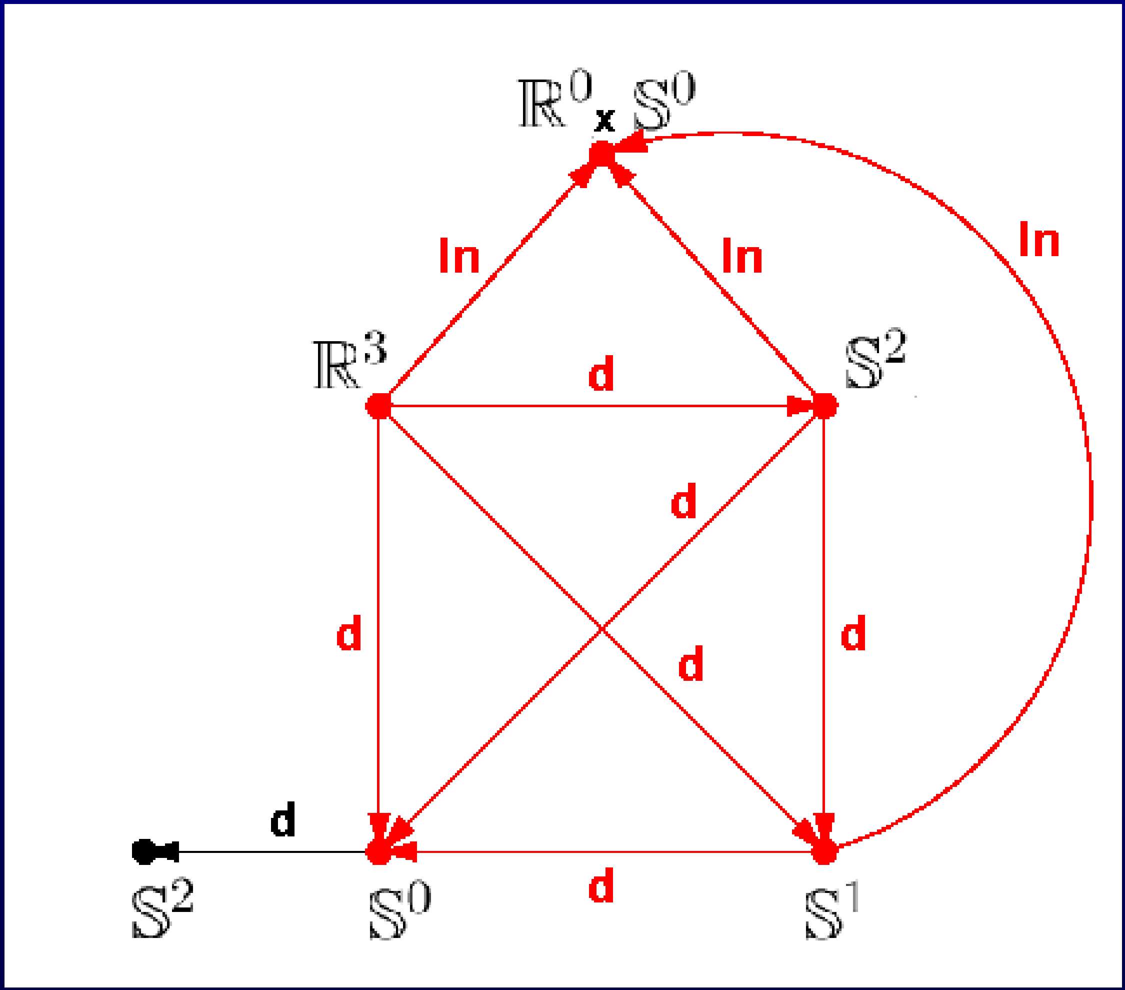
Adding vertex on plane constraint 2

Constraint System Example



Adding vertex on plane constraint 3

Constraint System Example



Solvable sub-system 3

Solvability

- An assignment of values f is *consistent* if it fulfills all constraints:

$$\forall o \in O \quad f(o) \in t(o) \quad \wedge$$
$$\forall ((o_1, \ldots, o_n), c) \in C \quad (f(o_1), \ldots, f(o_n)) \in c$$

- A constraint system is
  - ⋆ *consistent*, if there exists at least one consistent f
  - ⋆ *solvable*, if the set of consistent f modulo rotations and translations is discrete

    (locally unique solutions)
  - ⋆ *has a solution*, if it is solvable and consistent

Adding Constraints

- Adding a constraint (o, c) reduces allowed values by intersecting them with c
- Generic case:
  - ⋆ Original set of allowed values represented as:
  $$M_1 \times \cdots \times M_n$$
  - ⋆ Constraint reduces the dimension of one or more M_l as indicated by c and its different interpretations s
- Non-generic case:
  - ⋆ Hidden constraints create a special relation
  - ⋆ Dimension reduction is higher or lower than indicated by the s.

Adding Constraints Example

- Example: point $\mathbb{R}^3$ in intersection of two spheres
  - ⋆ Generic case:
    - ∗ Point on circle $\mathbb{S}^1$
  - ⋆ Special cases:
    - ∗ Point on sphere $\mathbb{S}^2$
      Hidden constraint: centres are equal
    - ∗ Equal to point $\mathbb{S}^0$
      Hidden constraint: centres and point on a line
  - ⋆ Inconsistent/over-constrained case:
    - ∗ Intersection empty $\emptyset$
      Contradiction to other constraint(s) present

Solvable Sub-Systems

- Our constraints do not fix a position or direction
- Objects are at most determined uniquely modulo rotations and translations
- Generically a system is solvable if $\mathbb{R}^3$ (translations) and $\mathbb{S}^2 \times \mathbb{S}^1$ (rotations) are the only non-discrete manifolds (special cases of lower-dimensional objects embedded in $\mathbb{E}^3$ exist)
- Any sub-system for which this is true is a solvable sub-system
- Any other sub-system has to be higher-dimensional
- Detecting and preserving solvability of sub-systems is essential and has to be studied in more detail

Summary

- Presented approach to choosing consistent constraints from automatically generated, inconsistent regularities
- Constraint selection based on priorities
- Constraint solver and solvability test based on dimensions of manifolds / sub-manifolds:
  - ⋆ Relates equations to constraint graph
  - ⋆ Algorithm similar to successful dense algorithm [Hoffmann, Lomonosov, Sitharam]
  - ⋆ Can handle generically inconsistent constraints
  - ⋆ May be possible to identify non-generic cases
- However, current approach not yet complete and clean:
  - ⋆ Constraint properties and solvability criteria