Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота
бакалавр

на тему **«Машине навчання та комп'ютерна алгебра»**

Виконав:
студент 4 курсу, групи МФ-41
спеціальність 122 «Комп'ютерні науки»
освітньо-професійна програма «Теоретична і прикладна інформатика»

Гаврюткін В. В.

Керівник: Жолткевич Г.М.

Керівник: Лейкін А.

Рецензент:

Харків — 2025 року

Ministry of Education and Science of Ukraine

V. N. Karazin Kharkiv National University
School of Mathematics and Computer Sciences
Department of Theoretical and Applied Informatics

Qualification work
bachelor

the topic **«Machine Learning and Computer Algebra»**

Performer: IV year student, group MF-41
Specialty 122 - Computer science
Educational and professional program
«Theoretical and Applied Computer Science»

Vladyslav Havriutkin

Supervisor: Hryhorii Zholtkevych

Supervisor: Anton Leykin

Reviewer:

Kharkiv — 2025

Contents

1 Introduction

Over the past decade, the fields of computer vision and robotics have seen dramatic advances driven by data-driven methods, yet many core geometric estimation tasks still rely on classical symbolic and algebraic algorithms. While deep learning excels at high-level segmentation and recognition, low-level pose estimation, state filtering, and kinematic solving continue to depend on recurring calls to polynomial solvers embedded within RANSAC-style frameworks. These algebraic routines are exact and robust, but can be computationally expensive in large-scale or real-time systems. Our thesis explores a hybrid approach: we retain the rigor of symbolic minimal-problem solvers while using lightweight neural classifiers to prune unnecessary solver calls before they execute, thereby combining the computer algebra and machine learning.

In the introductory section, we first review the development of algebraic notation and the rise of computer algebra systems, tracing a lineage from Viète and Descartes through Galois and Noether to the interactive symbolic engines of today. We then motivate the need for speed in geometric pipelines: in Structure-from-Motion, stereo reconstruction, Lidar-camera calibration, and manipulator kinematics. While techniques such as homotopy continuation and efficient matrix factorization have reduced per-call cost, the number of calls remains high. By learning a data-driven ś́olvability pruning (section 3.2), we aim to discard overdetermined samples that are algebraically unsolvable, reducing both computation and energy consumption without sacrificing accuracy. The subsequent sections detail our approach and test it in the COLMAP's geometric verification step (section 1.2.3).

1.1 History of Algebra and Computer Algebra

Algebra, in its broadest sense, has always been the study of the rules for manipulating mathematical symbols. Its roots extend as far back as the clay tablets of ancient Babylon (circa 1800–2000 B.C.), which record problems equivalent to modern linear and quadratic equations. One such tablet [tab24] gives a remarkably accurate approximation of $\sqrt{2}$ to six decimal places, demonstrating not only numerical ingenuity but an implicit understanding of polynomial relationships. Babylonian scribes solved problems of the form "find two numbers whose sum and product are given," foreshadowing the factorization and root-finding methods that would later be formalized in classical mathematics.

In the Hellenistic period, Greek mathematicians such as Euclid and Diophantus introduced geometric and rhetorical approaches to algebraic problems. Euclid's *Elements* treated ratios and proportions, while Diophantus's *Arithmetica* presented dozens of problems solved by ad hoc symbolic manipulation—making him the "father of algebra" in a rhetorical tradition. However, symbolic notation remained cumbersome, and general methods for solving systems of equations were largely absent.

The 9th–12th centuries saw the flowering of algebraic thought in Islamic mathematics. Al-Khwarizmi's *Kitâb al-Jabr wal-Muqâbala* (c.820 A.D.) systematically solved linear and quadratic equations using rhetorical procedures—*al-jabr* (completion) and *al-muqâbala* (balancing). Later scholars, including Ibn al-Banna and al-Qalasadi, began to introduce more compact notations for unknowns and operations, setting the stage for symbolic algebra. Their work conveyed equations in shorter phrases and, in some cases, used abbreviated letters for variables, thus improving clarity and paving the way for the symbolic breakthroughs of the Renaissance.

In the 16th and 17th centuries, Europe witnessed the formalization of algebraic notation. François Viète replaced rhetorical methods with literal symbols for unknowns and parameters, and René Descartes introduced the (x, y) coordinate system and the use of superscripts for powers—thereby unifying algebra and geometry under analytic methods. Gottfried Leibniz later proposed the concept of a matrix to organize systems of linear equations, anticipating the matrix algebra that is central to modern computational

methods.

The 19th century marked a shift from computational techniques to abstract structures. Évariste Galois developed group theory to understand solvability of polynomial equations, while Dedekind and Noether axiomatized rings and ideals. In this period, algebra became the language for a wide array of mathematical theories: from commutative algebra to algebraic geometry, each area treating polynomial systems as objects of structural investigation. These abstract foundations, though highly theoretical, enabled systematic methods for solving and classifying polynomial equations of many variables and high degree.

The advent of electronic computers in the mid-20th century introduced new opportunities for algebra. Early programs in the 1960s and 1970s performed symbolic differentiation and series expansion; by the 1980s, dedicated computer algebra systems (CAS) such as Macsyma, Maple, and Mathematica provided interactive environments for exact algebraic manipulation. CAS emphasize *symbolic computation*—where expressions are manipulated in closed form rather than evaluated numerically—thus guaranteeing mathematical exactness and enabling proofs, simplifications, and factorization beyond the reach of floating-point methods.

Today, a broad spectrum of CAS tools serve both general and specialized needs. General-purpose systems like Maple and Mathematica support a wide variety of algebraic and analytic operations, while open-source platforms such as SageMath and Axiom integrate multiple back-ends for polynomials, matrices, and combinatorics. In the realm of nonlinear algebra or algebraic geometry, academically developed systems—CoCoA, Macaulay2, Singular, and Risa/Asir—implement Gröbner bases, resultants, and homotopy continuation to solve systems of polynomial equations exactly or to characterize their solution sets.

Among the flagship algorithms in CAS is Buchberger's algorithm for computing Gröbner bases, which transforms an ideal in a multivariate polynomial ring into a canonical generating set. Resultant methods eliminate variables from polynomial systems by computing determinants of structured matrices, while numerical-symbolic techniques such as homotopy continuation track solution paths under parameter deformations. These algorithms have become indispensable in robotics, control theory, computer vision, and cryptography, wherever systems of algebraic equations arise.

In computer vision, many estimation problems [HDLP23], [MLC23], [FB81] reduce to *minimal problems*: finding the simplest algebraic system with finitely many solutions that fits the data. The RANSAC [FB81] paradigm—random sample consensus—solves these minimal problems robustly by randomly sampling small subsets of correspondences, fitting an algebraic model, and retaining the solution with the largest inlier set. Despite its effectiveness, RANSAC's repeated polynomial solves can be computationally expensive, motivating new approaches that integrate CAS algorithms with statistical and machine learning methods.

In this paper, we leverage advances in both algebraic foundations and modern computing to propose a neural-assisted pruning procedure for RANSAC. By combining group action in section 3.4 (via straightening transforms and Cayley parametrization) with learned classifiers section 3.5, we aim to reduce the number of expensive solver calls while preserving accuracy. The historical trajectory—from clay-tablet equations to abstract ideals and interactive CAS—underscores how algebraic techniques have continually evolved to meet computational demands. Our work injects data-driven pruning into a classical estimation pipeline, demonstrating a new approach for practical speed-ups in computer vision.

1.2 Overview of Results in the Investigated Application Areas

Our study provides a high-level survey across four canonical application areas where polynomial minimal solvers, and in particular the RANSAC paradigm, serve as fundamental tools for robust estimation. In the

first domain—state estimation from sensor input—we encounter polynomial systems derived from Doppler shift measurements or time-of-flight data, where a minimal set of transmitter–receiver readings yields algebraic equations whose finitely many solutions correspond to potential emitter positions and velocities under noise. The second domain, relative camera pose estimation in computer vision, formulates the five-point or essential-matrix solver as a minimal problem on projective point correspondences between two calibrated views, requiring repeated RANSAC-based sampling to identify inlier matches that satisfy the coplanarity constraint. Third, 3D reconstruction pipelines build upon these minimal-solver calls within an incremental Structure-from-Motion framework—alternating between pose estimation, triangulation, and bundle adjustment—before invoking multi-view stereo to densify the scene geometry. Finally, manipulator kinematics translates the Denavit–Hartenberg convention into a system of polynomial equations that characterize the feasible joint angles of a general 6R serial robot arm, yielding up to sixteen discrete solutions for a given end-effector pose. Together, these examples illustrate the ubiquity of minimal problems and the centrality of RANSAC's sample-consensus approach in rejecting spurious subsets of data under real-world noise and outliers. Detailed formulations and application-specific discussions will be presented in the following sections.

1.2.1 State Estimation from Sensor Input

State (position and velocity) estimation problem can be solved by utilizing Doppler shift [DS14] and modeled as a polynomial system. Applications of such a model is not hard to find. For example, Doppler shift based systems are used to determine the position and velocity of the underwater animals [GPM+20]. Besides, there are numerous applications in natural disaster prevention, monitoring, tracking vehicles, etc [FSQ+15]. Doopler's original formula can be restated as a system of polynomial equations [MLC23] that might provide up to finitely many states of the transmitter.

Consider an emitter with unknown velocity $v \in \mathbb{R}^3$ and position $r \in \mathbb{R}^3$ that emits a signal with an unknown frequency $f \in \mathbb{R}_{>0}$. Set of receivers, located at $r_i \in \mathbb{R}^3$ with velocities $v_i \in \mathbb{R}^3$ receive the signal at frequencies $f_i \in \mathbb{R}_{>0}$. The goal is to find v, r and f (fig. 1). According to Doppler effect, the observed by the ith receiver frequency f_i can be expressed in terms of the frequency at the emitter as

$$f_i = (1 - \frac{\dot{\rho}_i}{c})f \tag{1}$$

where $\dot{\rho}_i$ is the derivative of the range ρ_i between the emitter and ith receiver. The range can be expressed using Euclidean distance as:

$$\rho_i = \sqrt{(r_i - r)^\top (r_i - r)}$$

$$c^2(f - f_i)^2(r_i - r)^\top (r_i - r) - f^2[(r_i - r)^\top (v_i - v)]^2 = 0$$

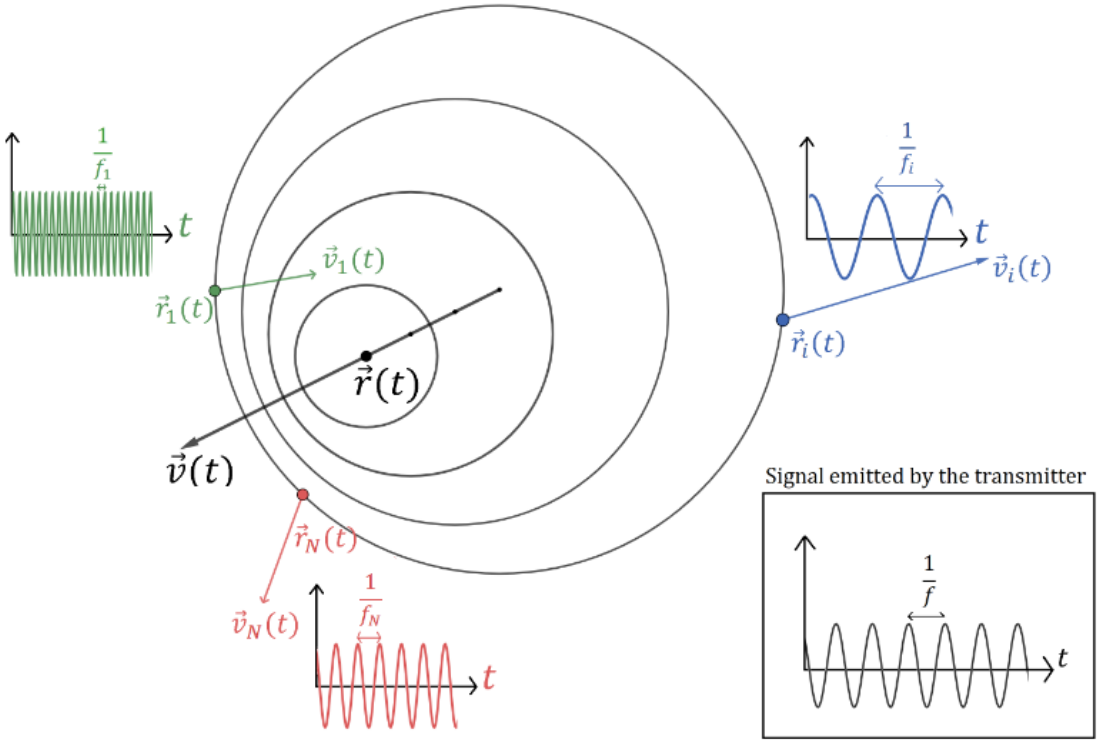

Figure 1: Illustration of the Doppler shift effect. (image from [MLC23])

As shown in [MLC23] the described system have solutions when 7 receivers are given. When the source frequency is known, 6 receivers is sufficient to have up to finitely many solutions of the system. Homotopy continuation [BSHW13, Mor09] can be used to solve the problem. Furthermore, certain restrictions (such as stationary receivers) can be imposed in order to simplify the system.

Note, that in practice, when one gets data from 7 receivers and attempts to solve it - spurious solutions are likely to arise. The corresponding system will have finitely many solutions algebraically, but those solutions might not represent the physical state of the system correctly. This happens, because the data provided by sensors often contains noise. The way to overcome this is to collect more data, so that one can compare and pick the best solution. RANSAC (see section 3.1) provides an efficient way to implement that idea.

1.2.2 Relative Pose Problem

Two cameras giving two images of the same space from two different view points are given. The goal is to reconstruct camera projection matrices, essentially finding a relative position of one camera with respect to the other camera. The described problem is called relative pose problem and is a step in the scene reconstruction pipeline, such as COLMAP

Consider the world coordinate system with a center at a point O and basis $\vec{d_1}, \vec{d_2}, \vec{d_3}$. Camera's coordinate system is located at a point C with bases vectors $\vec{c_1}, \vec{c_2}, \vec{c_3}$ and has an image plane π associated with it. Image plane π has an image coordinate system attached to it with a center o and bases vectors $\vec{b_1}, \vec{b_2}$. We are interested in projecting a point X from the world coordinate system on the image plane π, i.e. a transformation from $(O, \vec{d_1}, \vec{d_2}, \vec{d_3})$ to $(o, \vec{b_1}, \vec{b_2})$ (see section 1.2.2).

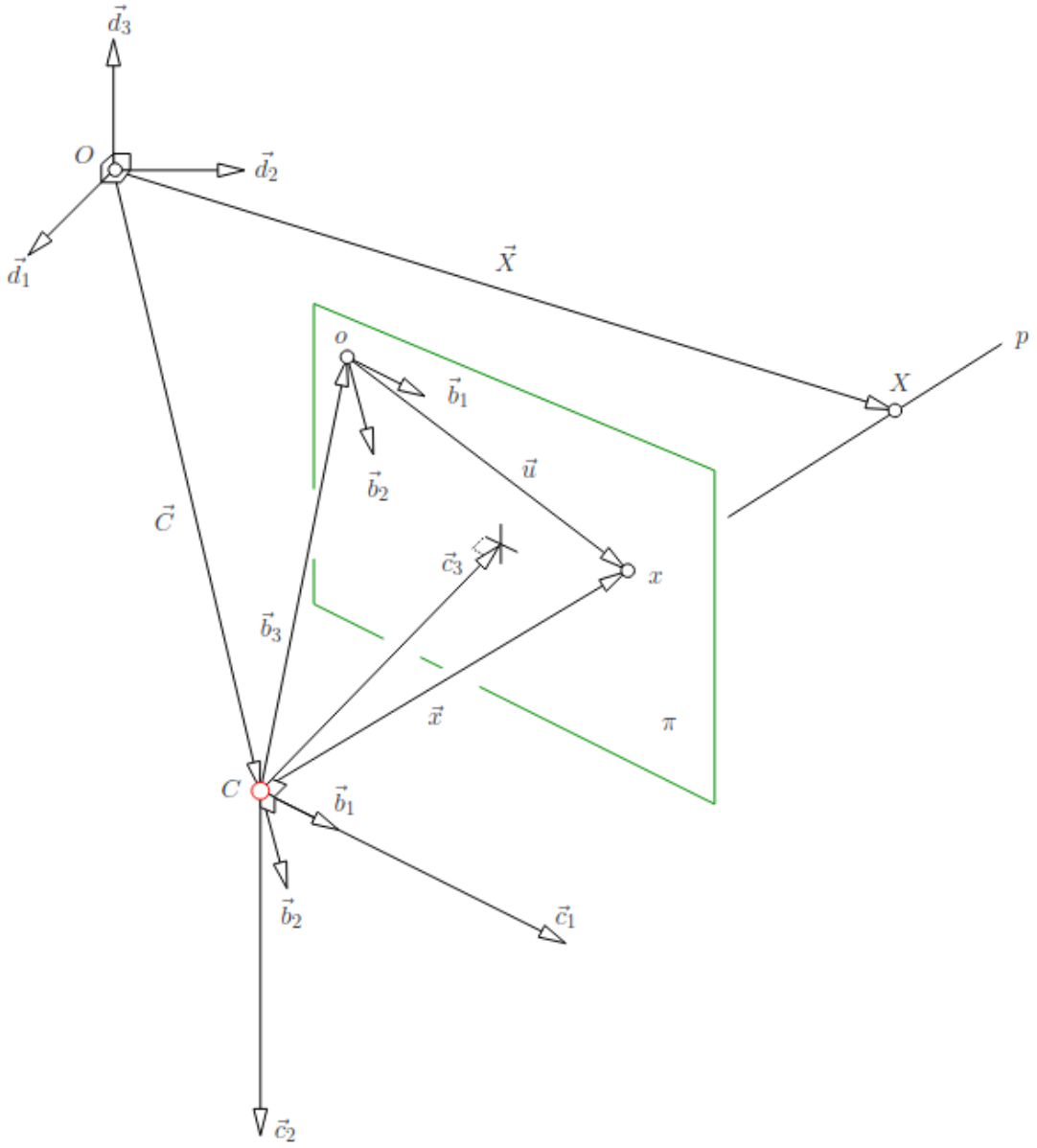

Figure 2: Perspective Camera Model. (image from [Tom])

This projection can be modeled as a linear transformation in projective space, from $\mathbb{P}^3$ to $\mathbb{P}^2$, using homogeneous coordinates. Any such transformation is defined only up to a nonzero scale factor, and the convention is to work within an affine chart where the last coordinate is normalized to 1.

Using the described setup, a point in the world $X = [x_1, x_2, x_3]^\top \in \mathbb{R}^3$ is represented by homogeneous coordinates $[x_1, x_2, x_3, 1]^\top \in \mathbb{P}^3$. Thus, camera projection is described by a 3×4 real matrix P that maps

$$\vec{p} = [p_1, p_2, p_3]^\top = P[x_1, x_2, x_3, 1]^\top$$

where $\vec{p} \in \mathbb{P}^2$ is a scaled representative of the sought point on the image plane. To obtain inhomogeneous image coordinates, we dehomogenize by dividing by the last coordinate (i.e., normalize the representative so that its last component is 1):

$$[\vec{u}, 1]^\top = \left[\frac{p_1}{p_3}, \frac{p_2}{p_3}, 1\right]^\top$$

As show in [Tom] P can be factorized as $P = K[R \mid t] \in \mathbb{R}^{3\times4}$, where $R \in \mathbb{R}^{3\times3}, t \in \mathbb{R}^3$ represent rotation and translation of the camera relative to the center of the world O; and $K \in \mathbb{R}^{3\times3}$ represents intrinsic parameters of a camera, such as focal length and center of the image plane. Intrinsic parameters are given in practice, and the details are irrelevant to the goals of this paper. The **epipolar geometry** (see section 3.3) is then utilized to set up a polynomial system for the two view scenario. RANSAC is applied to find the **essential matrix** (see section 3.3) that fits the most of the available data (point-to-point correspondences) from the given two views. Recently [HDLP23] showed a new approach, where they incorporated neural networks and Homotopy Continuation to accelerate classical RANSAC paradigm. Later, we use relative pose problem as a testing ground for our suggested approach in section 3.5.

1.2.3 3D Reconstruction in Computer Vision

The goal in 3D reconstruction is to recover a rich, often dense, representation of a physical scene from multiple images captured at different viewpoints. In modern pipelines, such as COLMAP ([SF16a], [SZPF16]), the process begins with feature extraction and matching across all image pairs. Keypoints are detected—commonly via SIFT or its variants—and described by high-dimensional descriptors; these descriptors are then matched exhaustively or via vocabulary-tree indexing to form correspondences. Matching quality is crucial, since later geometric steps rely on sufficient inlier ratios among these correspondences to produce accurate reconstructions.

Once tentative matches are established, COLMAP implements an incremental Structure-from-Motion (SfM) pipeline ([SF16b]) that alternates between pose estimation, triangulation, and bundle adjustment. The pipeline first selects an initial image pair with high overlap and robust baseline, then solves the two-view geometry problem: estimating the essential or fundamental matrix via RANSAC with a minimal solver (five-point for essential, seven-point for fundamental), followed by nonlinear refinement. Inlier correspondences are triangulated to obtain an initial sparse 3D point cloud, and the relative poses of the two cameras section 1.2.2.

New images are added one at a time by re-projecting existing 3D points into each candidate image and matching predicted projections to detected keypoints, thereby establishing 2D–3D correspondences. A Perspective-n-Point (PnP) solver within a RANSAC loop estimates each new camera's position and orientation relative to the growing model. After each successful registration, global or local bundle adjustment is performed to jointly optimize all camera poses and 3D point positions under a reprojection-error cost.

The **geometric verification** step—highlighted in section 1.2.3—occurs at each minimal-solver invocation, where RANSAC is used to robustly estimate essential or fundamental matrices and PnP solutions in the presence of outliers. This step dominates runtime in many SfM systems because each RANSAC iteration requires solving a polynomial system and evaluating reprojection errors for potentially thousands of correspondences. In our work, we inject a solvability-pruning procedure into precisely this stage, rejecting over-determined subsets of correspondences that are unlikely to admit a valid algebraic solution before the solver is called, thereby reducing the number of expensive model-fits.

After the incremental SfM completes, COLMAP produces a sparse reconstruction comprising camera poses and a set of 3D points with visibility data. To obtain a dense surface, Multi-View Stereo (MVS) techniques ([SZFP16]) are applied. Patch-based MVS methods iterate over each pixel in a reference view, initializing a 3D patch from nearby sparse points or via depth-map seeding, and then refine depth and normal estimates by aggregating photometric consistency scores across multiple views.

COLMAP's integration of SfM and MVS in a single end-to-end software suite allows flexible control over each stage. Users can adjust feature thresholds, RANSAC inlier tolerances, bundle adjustment schedules, and MVS depth-map parameters to balance speed and fidelity. In large-scale outdoor scenes with thousands of high-resolution images, the computational cost can be further mitigated by distributed or GPU-accelerated components. In particular, descriptor matching and dense depth-map computation are naturally data-parallel; COLMAP already supports CUDA for Patch-Match stereo, and our solvability pruning could similarly be offloaded to GPU kernels for batched minimal-solver screening.

In practice, 3D reconstruction pipelines must handle a wide variety of challenges: repetitive structures, weak textures, illumination changes, and dynamic objects can all introduce mismatches and spurious correspondences. Robust geometric verification, augmented by learned pruning, helps maintain reconstruction quality by focusing solver effort on subsets most likely to yield valid models. Empirically, we show in section 3.5.2 that injecting pruning into COLMAP's geometric verification may reduce solver calls by over 60% on benchmarks without measurable degradation of the final dense mesh's completeness or accuracy. This demonstrates that learned pre-filtering of minimal-problem samples can substantially accelerate large-scale 3D reconstruction workflows while preserving the high-quality results for which COLMAP is known.

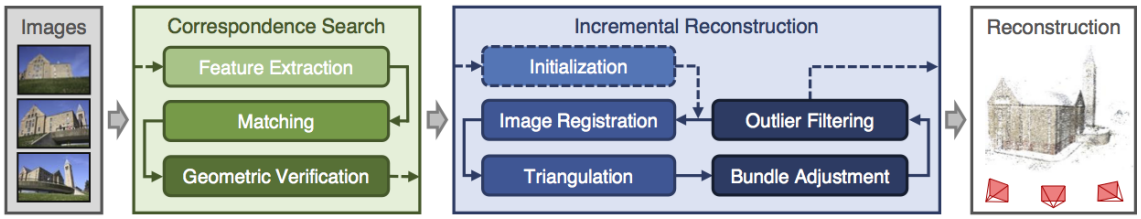

Figure 3: Structure-from-Motion pipeline in COLMAP. (image from [SF16a, SZPF16, COL])

1.2.4 Manipulator Kinematics

Methods of modern algebra are applied in the kinematic problems of serial robot manipulators [DJ]. The popular manipulator choice is a six-revolute (6R) arm with arbitrary geometry (arbitrary link lengths, no constraints), which allows a variety of possible configurations. It is known [DJ] that a 6R arm can be oriented in (at most) 16 different ways for the given end-effector pose. The configurations of such a manipulator can be described as an algebraic variety (see section 2), since each possible set of joint angles satisfying the task corresponds to one solution in a zero-dimensional variety.

A standard and highly systematic way to describe a manipulator as a system of polynomial equations is the Denavit–Hartenberg (DH) formalism. Under the DH convention, one assigns to each of the six moving joints four parameters: namely, the link length a_i, the link twist α_i, the link offset d_i, and the joint angle θ_i—and numbers the links (and associated frames) from 1 to 7, where link 1 is the fixed base and link 7 is the end-effector or "hand." For each link i, a right-handed coordinate frame i is attached so that the z_i-axis coincides with the axis of revolute joint i, and the x_i-axis is chosen along the common normal between z_{i-1} and z_i (or along the link common normal if the joints are offset) (section 1.2.4). This construction ensures that each homogeneous transformation describes only one joint's motion and one link's geometry, decoupling rotational and translational effects.

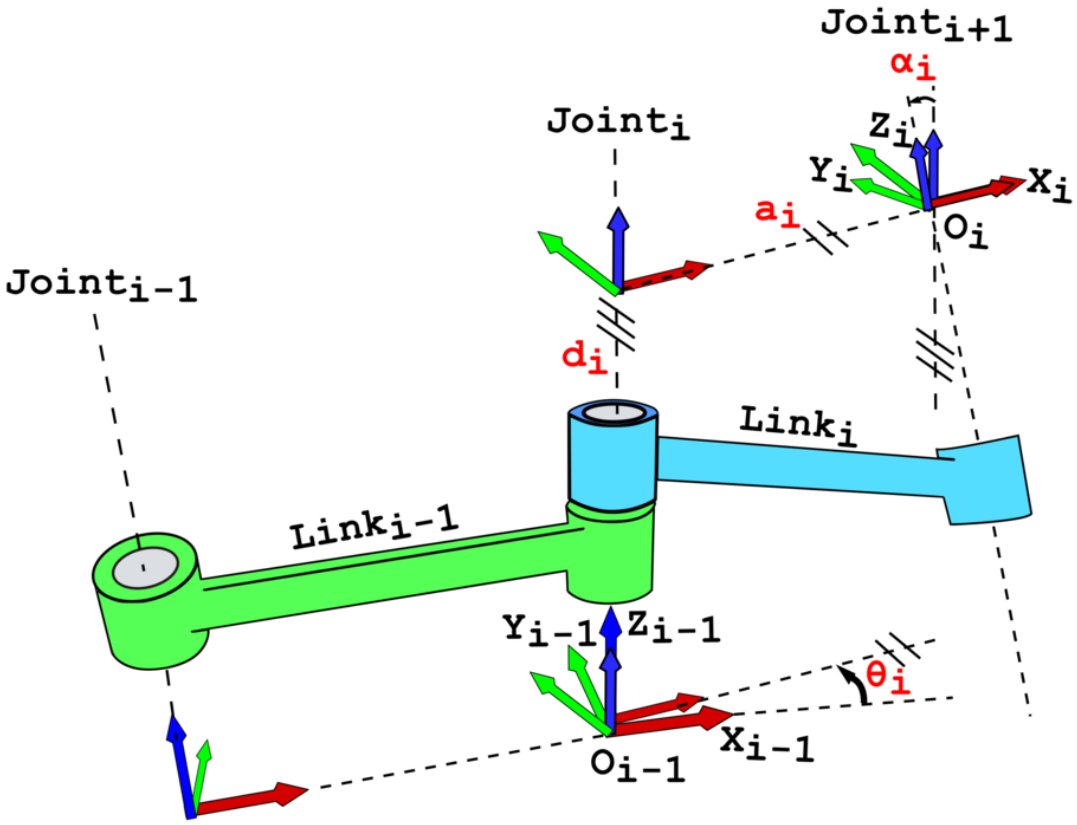

Figure 4: Denavit-Hartenberg Formalism. (image from [Wik])

As a result, the 4×4 matrix transformation A_i relates the $(i+1)$–th coordinate frame to the i–th frame:

$$A_i \;=\; \begin{pmatrix} c_i & -s_i\lambda_i & s_i\mu_i & a_i c_i \\ s_i & c_i\lambda_i & -c_i\mu_i & a_i s_i \\ 0 & \mu_i & \lambda_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

where $s_i = \sin\theta_i$, $c_i = \cos\theta_i$; $\mu_i = \sin\alpha_i$, $\lambda_i = \cos\alpha_i$; a_i is the length of link $i+1$; and d_i is the offset along the z_i–axis. By chaining these six transforms, one obtains the forward-kinematics map from the base frame to the end-effector:

$$T(\theta_1, \ldots, \theta_6) \;=\; A_1\, A_2\, A_3\, A_4\, A_5\, A_6.$$

In practice, the link parameters a_i, d_i, μ_i, and λ_i are known from the robot's design, and the goal is to solve the inverse problem: for a desired end-effector pose $A_{\text{end}} \in \mathrm{SE}(3)$, find all joint angles $\theta_1, \ldots, \theta_6$ satisfying

$$A_{\text{end}} \;=\; A_1\, A_2\, A_3\, A_4\, A_5\, A_6. \tag{3}$$

This inverse kinematics problem yields six polynomial equations in the six unknowns $\{\theta_i\}$. Due to the trigonometric terms, one typically rewrites c_i and s_i as rational functions via tangent–half-angle substitutions, converting (3) into a system of multivariate polynomials.

The classical algebraic approach [DJ] reduces this system to a single univariate polynomial—often of degree up to 16—by eliminating variables through resultants or matrix determinant methods. Solving that polynomial (e.g., via eigenvalue decomposition) recovers candidate θ_1 values, which are then back-substituted to find the remaining angles. More modern methods reformulate inverse kinematics

as a mixed-integer or nonlinear optimization problem [DIT19], trading algebraic completeness for robustness under modeling errors. Research has also considered manipulators with redundant degrees of freedom—such as 7R arms [TMHP20].

In summary, the 6R inverse kinematics problem lies at the intersection of algebraic geometry and robotics: its finitely many solutions correspond to the zero-dimensional variety defined by (3), and the task of efficiently computing those solutions continues to inspire both new elimination techniques and hybrid algebraic-numerical algorithms.

1.3 Overview of Our Results

In this thesis, we develop and evaluate a novel *solvability pruning* sampler that is injected into the classical RANSAC pipeline for minimal-problem solvers in computer vision. Our key innovation is a lightweight neural classifier ϕ that, given an over-determined sample of 6 point to point correspondences in the relative pose problem, predicts whether the underlying algebraic system yields at least one valid solution. By rejecting non-solvable samples before invoking an expensive polynomial solver, we achieve substantial reductions in total solver calls.

Concretely, our contributions are:

1. **Pruning network architecture.** We design and test three different classifying procedure ϕ as a compact feed-forward networks that operate on an 18-dimensional feature vector derived from six point correspondences in the relative pose problem. The networks use ReLU activations and a single logit output, trained with binary cross-entropy to distinguish solvable vs. unsolvable samples.

2. **Dimensionality reduction via straightening.** To compress the full 24 parameters of six pairs, we introduce a *straightening transformation* in projective space that canonically maps two point-to-point correspondences to the fixed geometric position. Applying this transform to three disjoint pairs and representing the resulting rotations via Cayley parameters allows us to reduce the input to only 18 real numbers, preserving all solvability information while minimizing computational overhead.

3. **Synthetic validation.** We generate 10,000 fabricated camera-pair samples with controlled translations and random rotations, corrupting 10% of six-tuples to simulate unsolvable cases. On this dataset, pruning procedure ϕ achieves over 90% recall and 70% precision, which translate into the speed-up factors between $\times 7.8$ and $\times 778$ depending on the true solvability ratio (10% down to 0.1%)—all while discarding fewer than 1.3% of truly solvable samples.

4. **Integration and real-world experiments.** We inject out pruning procedure into COLMAP's geometric verification stage, extracting 6-tuples from real feature matches on a 128-image dataset of UNC's "South" building. We successfully remove up to 65% of redundant solver runs. At the same time, we encounter the speed-up problem and discuss possible ways to resolve it.

5. **Reproducible code and guidelines.** All source code, training scripts, and detailed instructions are released in an open-source repository to facilitate adoption and extension to other minimal problems (e.g., five-point essential, P3P, seven-point fundamental).

The remainder of the thesis is organized as follows. In section 2 we review some essential mathematical preliminaries. In section 3.1 we discuss the classical RANSAC approach in details. In section 3.2 we introduce a concept of *solvability pruning* for RANSAC improvement. In section 3.3 we discuss the geometry of relative pose problem and introduce the essential matrix. In section 3.4 we introduce the concept of and show the construction algorithm for *straightening transformation*. In section 3.4.2 we

demonstrate the way to utilize *straightening transformation* to reduce the dimension of the relative pose problem. In section 3.5 we train three different multi-layer perceptrons as a pruning procedure; we inject it into the geometrical verification step of COLMAP and provide performance metrics. Finally, in section 4 we summarize our work and layout directions for the future research.

2 Mathematical Preliminaries

Vector Products in $\mathbb{R}^3$. In three dimensions, the cross product $\times : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$ is defined by

$$u \times v = (u_2 v_3 - u_3 v_2,\ u_3 v_1 - u_1 v_3,\ u_1 v_2 - u_2 v_1)^\top,$$

which yields a vector orthogonal to both u and v with magnitude $\|u\|\,\|v\| \sin\theta$ (where θ is the angle between u and v). The cross product is bilinear, anti-commutative ($u \times v = -\,v \times u$), and satisfies the Jacobi identity

$$u \times (v + w) = u \times v + u \times w, \quad (\alpha u) \times v = u \times (\alpha v) = \alpha(u \times v).$$

A key identity is the *vector triple product*: for any $u, v, w \in \mathbb{R}^3$,

$$u \times (v \times w) = v\,(u \cdot w) - w\,(u \cdot v),$$

which exhibits how two consecutive cross products reduce to a linear combination of v and w weighted by dot products. We utilize this property when constructing *straightening transformation*.

Polynomial Rings in n Variables. Let K be a field. The polynomial ring

$$K[x_1, \ldots, x_n]$$

is the set of all finite K-linear combinations of monomials $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$. A general element $f \in K[x_1, \ldots, x_n]$ can be written

$$f = \sum_{\alpha \in \mathbb{N}^n} c_\alpha\, x^\alpha, \quad c_\alpha \in K, \quad x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n},$$

and its *total degree* is $\max\{\sum_i \alpha_i : c_\alpha \neq 0\}$.

Ideals and Affine Varieties. Let K be a field and let $K[x_1, \ldots, x_n]$ denote the polynomial ring in n variables over K. An *ideal* $I \subseteq K[x_1, \ldots, x_n]$ is a subset satisfying

$$0 \in I, \quad f, g \in I \implies f + g \in I, \quad f \in I,\ h \in K[x_1, \ldots, x_n] \implies hf \in I.$$

The *affine variety* defined by I is

$$\mathcal{V}(I) = \big\{\, x \in K^n : f(x) = 0\ \forall f \in I \big\}.$$

Gröbner Bases. Fix a monomial order $\prec$ on $K[x_1, \ldots, x_n]$. A finite subset $G = \{g_1, \ldots, g_t\} \subset I$ is a *Gröbner basis* of I if the leading term of every $f \in I$ is divisible by the leading term of some $g_i \in G$. Buchberger's algorithm computes a Gröbner basis by iteratively processing *S-polynomials* and performing multivariate division until all remainders vanish.

Projective Space and Homogeneous Coordinates. The projective n-space over K is $\mathbb{P}^n = (K^{n+1} \setminus \{0\})/\sim$, where $(X_0, \ldots, X_n) \sim \lambda(X_0, \ldots, X_n)$ for all $\lambda \in K^\times$. A point in $\mathbb{P}^n$ is written $[X_0 : X_1 : \cdots : X_n]$. A polynomial $F \in K[X_0, \ldots, X_n]$ is *homogeneous of degree* d if every monomial in F has total degree d. Dehomogenization by setting $X_0 = 1$ yields the affine chart $\{[1 : x_1 : \cdots : x_n]\} \cong K^n$.

The Special Orthogonal Group $\mathrm{SO}(3)$**.** The special rotation groups is defined as:

$$\mathrm{SO}(3) = \{\, R \in \mathbb{R}^{3\times 3} : R^\top R = I_3,\ \det R = 1\}.$$

Cayley Parametrization of $\mathfrak{so}(3)$**.** For $\omega = (\omega_1, \omega_2, \omega_3)^\top \in \mathbb{R}^3$, let $[\omega]_\times \in \mathfrak{so}(3)$ be the skew-symmetric matrix

$$[\omega]_\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}.$$

The *Cayley map*

$\mathcal{C} : \mathfrak{so}(3) \setminus \{-I\} \to \mathrm{SO}(3)$ is defined by

$$\mathcal{C}(A) \;=\; (I - A)\,(I + A)^{-1},$$

providing a rational parametrization of the $\mathrm{SO}(3)$ group.

3 Solvability Pruning for Relative Pose Problem

3.1 Classical RANSAC Paradigm

As we have already seen in the examples described in section 1.2, obtained dataset often contains noise. Therefore any polynomial system based on it, while having solutions, might yield spurious results. To overcome this challenge, one can collect more data and find the solution that agrees with most of the observed data points. In this scenario, a set of data points P of size n is given. Notice, that the system created using all n data points will be over constrained and - since data is noisy - likely not have any solutions at all. Therefore, it makes sense to solve only **minimal problems**. Essentially, a minimal problem is a system of polynomial equations that have finitely many solutions. Let's say a system requires k parameters. The challenge is to choose such a subset $p \in P$ of size k, that the corresponding system results in the most meaningful solution in terms of P. While being robust, RANSAC has inefficiencies. First, it wastes computational effort on models that produce small consensus sets. Second, most of minimal problems in geometry are optimization problems, which require costly check and removal on spurious solutions on step 2. The second problem was recently addressed in [HDLP23]. Authors suggested a way to avoid solving for spurious solutions by using Homotopy Continuation (HC) method and learning a procedure σ (essentially, a lightweight neural network) that picks $a = \sigma(S^*)$ as a starting point for the HC. With this 'pick and solve' approach, authors achieved ten times speedup in time for certain minimal problems. The state of the art approach was suggested by Fishler and Bolles in 1981 [FB81] and is called **random sample consensus** (RANSAC). As before, we are given a model that requires k data points to instantiate and a set P of size $n > k$. RANSAC pipeline operates as follows:

Algorithm 1 RANSAC

Require: P — set of points, $k \in \mathbb{Z}_{>0}$ — sample size, t — tolerance

Randomly select $S \subset P$ such that $|S| = k$.
Instantiate model $M \leftarrow \text{FitModel}(S)$.
Determine consensus set $S^* \leftarrow \{\, p \in P \mid \text{Error}(p, M) < t\}$.
if $|S^*| \geq t$ **then**
    Refine model $M^* \leftarrow \text{FitModel}(S^*)$.
    return M^*
else
    Continue sampling (or, if max iterations reached, return best-so-far consensus)
end if

3.2 Modifying RANSAC with Solvability Pruning

Several studies have successfully attempted to improve RANSAC's performance. For example, in [CM05] authors define and utilize linear ordering on the set of correspondences to achieve computational savings. Another modification, which also can be found in COLMAP, is locally optimized RANSAC [CMK03] that uses iterated least squares refitting applied to the inliers.

Unlike already suggested approaches, our method proposes to modify RANSAC with the solvability pruning classifier in order to speed up the state-of-the-art algorithms by reducing the number of solver calls. Taking inspiration from [HDLP23], we suggest a way to avoid solving minimal problems that are likely to produce a small or empty consensus set. Instead of picking a subset of size k, that model requires, we attempt to pick an over constrained subset S of size $k+1$. Algebraically, the derived from S system is over determined, therefore doesn't have solutions in generic case. But since the underlying data describes an actual physical model (like position of a transmitter, or position of a camera), even an over determined system should have a solution, if data points are good enough. Furthermore, if picked $k+1$ points end up producing a model M, then the size of consensus set for M is at least $k+1$. Therefore, M is more likely to be not spurious.

The challenge is to decide whether $k+1$ points result in a solvable system. Just attempting to relax (i.e. get rid of one equation) and solve it is no different from the original RANSAC pipeline. We suggest training a **solvability pruning** procedure ϕ that classifies system as solvable or not.

Given a model that requires k data points to instantiate, and a set P of size $n > k$. The pipeline with solvability pruning is the following:

Algorithm 2 RANSAC with Pruning

Require: P — set of points, $k \in \mathbb{Z}_{>0}$ — sample size, t — tolerance, ϕ - pruning procedure

Randomly select $S \subset P$ such that $|S| = k + 1$.
if $\phi(S) = 0$ **then**
    Discard S and continue sampling
else
    Pick S_k - first k elements of S
    Instantiate model $M \leftarrow \text{FitModel}(S_k)$.
    Determine consensus set $S^* \leftarrow \{\, p \in P \mid \text{Error}(p, M) < t\}$.
    if $|S^*| \geq t$ **then**
        Refine model $M^* \leftarrow \text{FitModel}(S_k^*)$.
        return M^*
    else
        Continue sampling (or, if max iterations reached, return best-so-far consensus)
    end if
end if

3.3 Epipolar Geometry and Essential Matrix

The geometric verification part of the COLMAP pipeline discussed in section 1.2.3 that we address involves the classical application of RANSAC with a solver for the relative camera pose problem.

When a 3D point $X \in \mathbb{R}^3$ is observed by two calibrated cameras from distinct viewpoints, it is projected onto two image points $\mathbf{u}_1 \in \mathbb{P}^2$ and $\mathbf{u}_2 \in \mathbb{P}^2$, corresponding to the first and second camera respectively. The line joining C_1 and C_2 is called the *baseline*. The intersection points e_1, e_2 of the baseline with each of the image planes are called *epipoles*. The plane defined by C_1, C_2 and the 3D point X is called the *epipolar plane*. This plane intersects each image plane in lines l_1, l_2 — known as *epipolar line*—on which the corresponding image points lie (see section 3.3). We assume that the intrinsic calibration matrices K_1 and K_2 of both cameras are known, and the image points are normalized.

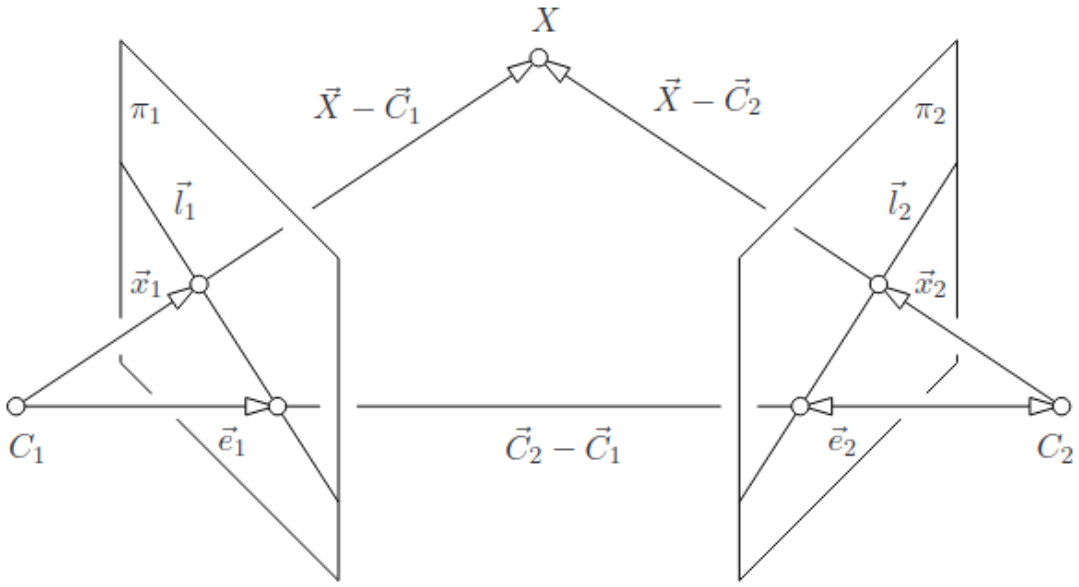

Figure 5: Epipolar Geometry of Two Cameras. (image from [Tom])

As shown in details in [Tom], the geometric constraint between the two views can be written in terms of the *essential matrix* $E \in \mathbb{R}^{3\times3}$, as

$$\mathbf{u}_2^\top E\, \mathbf{u}_1 = 0$$

This equation states that $\mathbf{u}_1, \mathbf{u}_2$, and the baseline direction all lie in the same epipolar plane, and thus are coplanar. The matrix E encodes the relative pose (rotation and translation) between the two cameras:

$$E = [\mathbf{t}]_\times R$$

Here, $R \in \mathrm{SO}(3)$ is the relative rotation between the two camera coordinate frames, and $\mathbf{t} \in \mathbb{R}^3$ is the relative translation (up to scale). The notation $[\mathbf{t}]_\times$ denotes the skew-symmetric matrix corresponding to the cross product with vector $\mathbf{t}$:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

Thus, for any pair of corresponding normalized image points $(\mathbf{x}_1, \mathbf{x}_2)$, the constraint

$$\mathbf{x}_2^\top E\, \mathbf{x}_1 = 0$$

must hold. The essential matrix itself is subject to additional constraints. First, considering the way it is constructed, E must have rank 2, i.e.,

$$\det(E) = 0$$

Moreover, it must satisfy the so-called *trace constraint*:

$$2EE^\top E - \operatorname{trace}(EE^\top)E = 0$$

These constraints reduce the degrees of freedom of the essential matrix to five: three from the rotation $R \in \mathrm{SO}(3)$ and two from the translation vector $\mathbf{t} \in \mathbb{P}^2$.

Therefore, a minimal problem in relative pose recovery requires five point-to-point correspondences.

3.4 Utilizing the group action

In practice, we are given a set P of size n that contains pairs p of coordinates in the image planes of the corresponding points:

$$p = (p_{i1}, p_{i2}) = ([x_{i1}, y_{i1}]^\top, [x_{i2}, y_{i2}]^\top) \in P$$

In this case a 'model' is an essential matrix E. As discussed in section 3.3, five point-to-point correspondences are required to instantiate (solve for) E. Following solvabililty pruning approach discussed in **??**, we will pick a subset $S \subset P$ of size 6. Our goal is to train a pruning procedure ϕ that takes S as an input and classifies it as solavable or not.

Each pair $p \in S$ has 4 parameters - two coordinates for each point. We pick six pairs, therefore ϕ must take $6 * 4 = 24$ input parameters. We aim to keep ϕ as lightweight as possible, in order to actually gain speed advantage. For this purpose, we introduce a special group action called **straightening transformation** that will allow us to reduce dimension of the problem to 18.

3.4.1 Straightening transformation

Proposition 1. *Consider the projective plane $\mathbb{P}^2$. Given two points*

$$p_1 = [x_1, y_1, 1], \quad p_2 = [x_2, y_2, 1]$$

such that $p_1 \times p_2 \neq 0$ interpreted as points in the projective plane $\mathbb{P}^2$.

There exist two transformations $M \in \mathrm{PSO}(3)$ such that

$$Mp_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad Mp_2 = \begin{bmatrix} 0 \\ n \\ 1 \end{bmatrix} \in \mathbb{P}^2,$$

for some scalar $n \in \mathbb{R}$.

Proof. A transformation M can be interpreted as a composition:

$$M = \sigma \circ T,$$

where $T \in \mathrm{SO}(3)$ and σ is the dehomogenization operator defined by

$$\sigma\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix}, \quad z \neq 0.$$

Thus, our goal reduces to finding a rotation matrix T such that

$$Tp_1 = \begin{bmatrix} 0 \\ 0 \\ a_1 \end{bmatrix}, \quad Tp_2 = \begin{bmatrix} 0 \\ a_2 \\ a_3 \end{bmatrix},$$

for some scalars $a_1, a_2, a_3 \in \mathbb{R}$.

Existence by construction. We now show that such a transformation exists, and can be explicitly constructed. Define the inverse of $T \in \mathrm{SO}(3)$ as

$$T^{-1} = \begin{bmatrix} \frac{\vec{p_1} \times \vec{p_2}}{\|\vec{p_1} \times \vec{p_2}\|} & \frac{\vec{p_1} \times (\vec{p_1} \times \vec{p_2})}{\|\vec{p_1}\| \cdot \|\vec{p_1} \times \vec{p_2}\|} & \frac{\vec{p_1}}{\|\vec{p_1}\|} \end{bmatrix},$$

where $\vec{p_1}, \vec{p_2} \in \mathbb{R}^3$ denote the homogeneous coordinate vectors.

We verify that this choice of T^{-1} satisfies the desired conditions:

Let $a_1 = \|\vec{p_1}\|$. Then,

$$T^{-1} \begin{bmatrix} 0 \\ 0 \\ a_1 \end{bmatrix} = a_1 \cdot \frac{\vec{p_1}}{\|\vec{p_1}\|} = \vec{p_1},$$

so

$$T\vec{p_1} = \begin{bmatrix} 0 \\ 0 \\ a_1 \end{bmatrix}.$$

Define constants:

$$a_2 = -\frac{\|\vec{p_1}\| \cdot \|\vec{p_1} \times \vec{p_2}\|}{\vec{p_1} \cdot \vec{p_1}}, \quad a_3 = \frac{\vec{p_1} \cdot \vec{p_2}}{\vec{p_1} \cdot \vec{p_1}}.$$

Then,

$$T^{-1} \begin{bmatrix} 0 \\ a_2 \\ a_3 \end{bmatrix} = a_2 T^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + a_3 T^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= a_2 \left(\frac{\vec{p_1} \times (\vec{p_1} \times \vec{p_2})}{\|\vec{p_1}\| \cdot \|\vec{p_1} \times \vec{p_2}\|} \right) + a_3 \left(\frac{\vec{p_1}}{\|\vec{p_1}\|} \right)$$

$$= -\frac{\vec{p_1} \times (\vec{p_1} \times \vec{p_2})}{\vec{p_1} \cdot \vec{p_1}} + \frac{\vec{p_1} \cdot \vec{p_2}}{\vec{p_1} \cdot \vec{p_1}} \vec{p_1}$$

$$= -\frac{1}{\vec{p_1}\cdot\vec{p_1}}\left(\vec{p_1}(\vec{p_1}\cdot\vec{p_2}) - \vec{p_2}(\vec{p_1}\cdot\vec{p_1})\right) + \frac{\vec{p_1}\cdot\vec{p_2}}{\vec{p_1}\cdot\vec{p_1}}\vec{p_1} = \vec{p_2}.$$

Hence,

$$T\vec{p_2} = \begin{bmatrix} 0 \\ a_2 \\ a_3 \end{bmatrix}.$$

This completes the construction of the transformation $M = \sigma \circ T \in \mathrm{PSO}(3)$ that maps p_1 to the origin and p_2 to a point on the vertical axis. We call M a *straightening transformation*.

Number of solutions. Any $T \in \mathrm{SO}(3)$ satisfying

$$T\,\vec{p}_1 = \begin{pmatrix} 0 \\ 0 \\ b_1 \end{pmatrix}, \quad T\,\vec{p}_2 = \begin{pmatrix} 0 \\ b_2 \\ b_3 \end{pmatrix}$$

for some b_1, b_2, b_3 must preserve lengths and inner products:

$$\|T\,\vec{p}_1\| = \|\vec{p}_1\| \implies b_1 = \pm\|\vec{p}_1\|, \quad \|T\,\vec{p}_2\|^2 = b_2^2 + b_3^2 = \|\vec{p}_2\|^2,$$

$$(T\,\vec{p}_1)\cdot(T\,\vec{p}_2) = \vec{p}_1\cdot\vec{p}_2 \implies b_3 = \frac{\vec{p}_1\cdot\vec{p}_2}{b_1}.$$

Hence the only possible triples (b_1, b_2, b_3) are

$$\left(\pm\|\vec{p}_1\|,\ \pm\frac{\|\vec{p}_1 \times \vec{p}_2\|}{\|\vec{p}_1\|},\ \frac{\vec{p}_1\vec{p}_2}{\|\vec{p}_1\|}\right),$$

and exactly two of these choices lie in $\mathrm{SO}(3)$. No other assignment of (b_1, b_2, b_3) can be realized by a proper rotation.

□

Thus, one has just two options for the choice of a straightening transformation when two points in the projective plane are given. We adopt the convention where we pick representative with a positive sign. Below, we summarize the algorithm we use to find a straightening transformation and use it to encode the given points.

Algorithm 3 Straightening Algorithm

Require: $p_1 = [x_1, y_1, 1]$ - point in the first camera, $p_2 = [x_2, y_2, 1]$ - point in the second camera, ϵ - tolerance

1: **if** $|p_1 \times p_2| < \epsilon$ **then**
2: **return** error, points are collinear
3: **end if**
4: $r_3 = \frac{p_1}{|p_1|}$
5: $r_1 = \frac{p_1 \times p_2}{|p_1 \times p_2|}$
6: $r_2 = r_3 \times r_1$
7: $T = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix}$ - straightening transformation
8: $e = \frac{\text{second coordinate of } Tp_2}{\text{third coordinate of } Tp_2}$ - encoding number for p_1 and p_2
9: **return** T, e

3.4.2 Manipulating straightening transformations

Now, we show how to use the straightening transformation above to reduce dimension of the system to 18. First, we pick a subset $S \subset P$ that consists of triple of pair correspondences $(u_1, v_1), (u_2, v_2), (u_3, v_3)$ where:

$$u_i = (\vec{p}_{i1}, \vec{p}_{(i+1)1}) = \left(\begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix}, \begin{bmatrix} x_{(i+1)1} \\ y_{(i+1)1} \\ 1 \end{bmatrix} \right), i = \overline{1,3}$$

$$v_i = [\vec{p}_{i2}, \vec{p}_{(i+1)2}] = [\begin{bmatrix} x_{i2} \\ y_{i2} \\ 1 \end{bmatrix}, \begin{bmatrix} x_{(i+1)2} \\ y_{(i+1)2} \\ 1 \end{bmatrix}], i = \overline{1,3}$$

and $\vec{p}_{i1} \in \mathbb{P}^2$ - points from the first camera, $\vec{p}_{i2} \in \mathbb{P}^2$ - points from the second camera.

For each pair (u_i, v_i) we call $(R_i, S_i) \in SO(3) \times SO(3)$ an **encoding transformation**, where R_i is a straightening transformation for u_i and S_i is a straightening transformation for v_i

Now, let us consider the original essential matrix equations in three batches:

$$u_i^\top E v_i = 0, i = \overline{1,3}$$

We apply a corresponding encoding transformations to each batch:

$$u_i^\top R_i E S_i^\top v_i = 0$$

$$\begin{bmatrix} 0 & 0 \\ 0 & a_i \\ 1 & 1 \end{bmatrix}^\top R_i^\top E S_i \begin{bmatrix} 0 & 0 \\ 0 & b_i \\ 1 & 1 \end{bmatrix} = 0$$

Let us arbitrary pick $\widetilde{E}$ to be equal to $R_1^\top E S_1$ from the first batch. Now we rewrite the initial system in terms of $\widetilde{E}$:

$$\begin{bmatrix} 0 & 0 \\ 0 & a_1 \\ 1 & 1 \end{bmatrix}^\top \widetilde{E} \begin{bmatrix} 0 & 0 \\ 0 & b_1 \\ 1 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 \\ 0 & a_2 \\ 1 & 1 \end{bmatrix}^\top R_2^\top R_1 \widetilde{E} S_1^\top S_2 \begin{bmatrix} 0 & 0 \\ 0 & b_2 \\ 1 & 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 \\ 0 & a_3 \\ 1 & 1 \end{bmatrix}^\top R_3^\top R_1 \widetilde{E} S_1^\top S_3 \begin{bmatrix} 0 & 0 \\ 0 & b_3 \\ 1 & 1 \end{bmatrix} = 0$$

The first batch has just 2 parameters - a_1, b_1. The second batch has $3 + 3 = 6$ parameters for $R_2^\top R_1 \in SO(3)$ and $S_1^\top S_2 \in SO(3)$, and a_2, b_2 - total of 8 parameters. Third batch has 8 parameters by analogy. Therefore, the dimension of the problem is reduced to 18. We won't attempt to actually solve the transformed system, instead we will train ϕ to now take only 18 parameters and decide whether the initial pick of 6 points is spurious or not.

3.5 Training and Experimenting with Pruning Procedure

In this section, we describe the process of data extraction, model training and testing results.

As discussed in section 3.4.2, we begin by selecting six-tuple samples of matched image features across two views; each sample consists of three independent pairs of correspondences. Rather than operating on the raw pixel coordinates directly, we apply the *straightening transformation* (see Section 3.4.1). After applying the encoding procedures we end up with 18-dimensional feature vector that is sufficient for pruning decisions. Once encoded, these vectors form the inputs to our lightweight neural classifier, which we train to predict solvability labels. In the remainder of this section, we present the exact input format and then summarize the resulting classification accuracy and solver-call reduction metrics achieved on both synthetic and real-world datasets.

$$
\begin{aligned}
&a_1, b_1, \\
&\mathcal{C}()(R_2^\top R_1),\ \mathcal{C}()(S_1^\top S_2),\ a_2,\ b_2, \\
&\mathcal{C}()(R_3^\top R_1),\ \mathcal{C}()(S_1^\top S_3),\ a_3,\ b_3.
\end{aligned}
$$

Here, R_i and S_i denote the straightening rotations for the ith correspondence pair in the first and second views, respectively; $a_i, b_i \in \mathbb{R}$ are the scalar offsets arising after straightening; and $\mathcal{C}(()\cdot) \in \mathbb{R}^3$ is the Cayley parametrization of a rotation in SO(3). This normalized input format serves as an input for ϕ.

3.5.1 Experiments on Fabricated Data

Training a classifier. Our first experimental classifier is trained on the fabricated data. First camera is created with the identify matrix as rotation and $[0, 0, -d]^\top, 0 < d \leq 3$ as a translation vector. The choice of the translation is motivated by the geometrical intuition of this camera not being too far away from the observed space. Second camera's rotation is set to be random $SO(3)$ rotation, but exactly the same translation vector. Geometrically, those two cameras are located next to each other, but observe space in slightly different orientation. We put the described constraints on the fabricated data, so that relatively small neural network can capture underlying data distribution and provide us with proof of concept. In the fabricated case, our dataset is a collection of $10,000$ independent samples

$$\mathcal{D} = (x_i, y_i)_{i=1}^{10,000}, x_i \in \mathbb{R}^{18}, y_i \in \{0, 1\}$$

During simulation, 10% of 6-tuples are intentionally corrupted and labeled as not solvable. Our classifier is a feed-forward neural network $\phi : \mathbb{R}^{18} \to \mathbb{R}$ with a single logit unit as an output. Specifically,

$$\phi(x) = W_3\sigma(W_2\sigma(W_1x + b_1) + b_2) + b_3$$

,

where: $W_1 \in \mathbb{R}^{64\times18}, b_1 \in \mathbb{R}^{64}$, $W_2 \in \mathbb{R}^{32\times64}, b_2 \in \mathbb{R}^{32}$, $W_3 \in \mathbb{R}^{1\times32}, b_3 \in \mathbb{R}$, σ is the ReLU activation function: $\sigma(u) = \max(0, u)$

Logit units are converted into a probability estimate

$$\hat{p} = \frac{1}{1 + e^{-\phi(x)}}$$

If $\hat{p}$ is greater than 0.5 we predict 1 (i.e. solvable), else 0 (i.e. not solvable).

We use the binary cross-entropy with logits as an loss function:

$$\mathcal{L}(\theta) = -\frac{1}{\mathcal{B}} \sum_{(x,y)\in\mathcal{B}} [y \log \hat{p}(\phi(x)) + (1-y)\log(1-\hat{p}(\phi(x))]$$

,

where $\mathcal{B}$ denotes a training batch. We train for 200 epochs with a batch size of 32. We use PyTorch [PyT] and Adam optimizer [KB15].

As a result, we achieve 75% accuracy on the validation set with nearly 70% precision and 90% recall (see section 3.5.1).

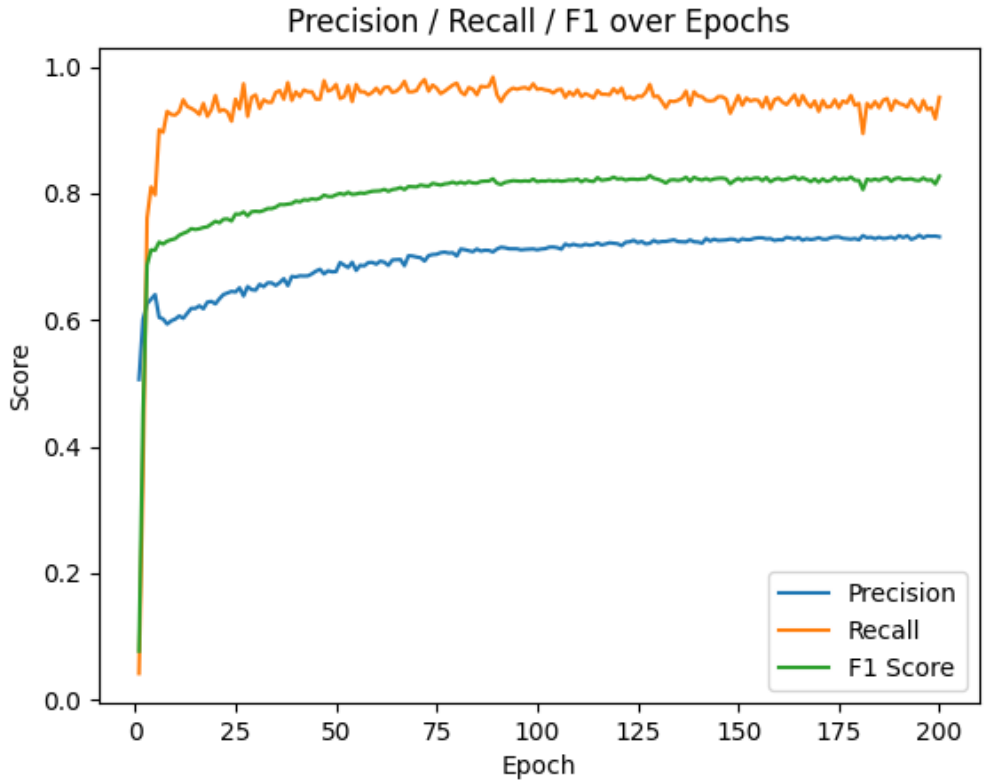

Figure 6: Metrics on Fabricated Data

Speed Up Estimate. Consider the RANSAC pipeline in relative pose problem in which one samples N random 6-tuples from a large set of feature matches and runs a solver on each. Let

$$N \quad = \quad \text{total number of 6-tuples sampled,}$$

and denote by

$$P \quad = \quad \frac{\text{truly solvable 6-tuples}}{N} \quad \in (0,1)$$

the *ratio* of solvable tuples in the random pool.

Let $\text{TP} = N \times P \times \text{Recall}$ be the number of true positives (solvable tuples correctly identified by ϕ), FP be the number of false positives. Since

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

the total number of tuples forwarded to the solver is

$$\text{Kept} = \text{TP} + \text{FP} = \frac{\text{TP}}{\text{Precision}} = \frac{N \times P \times \text{Recall}}{\text{Precision}}. \tag{4}$$

Therefore, the number of solver calls *skipped* is

$$\text{Removed} = N \; - \; \frac{N \times P \times \text{Recall}}{\text{Precision}} = N \left(1 - \tfrac{P \times \text{Recall}}{\text{Precision}}\right). \tag{5}$$

If each solver invocation dominates total runtime and the classification cost is negligible, the *speed-up factor* is

$$S \; = \; \frac{\text{Original solves}}{\text{New solves}} \; = \; \frac{N}{\frac{N \times P \times \text{Recall}}{\text{Precision}}} \; = \; \frac{\text{Precision}}{P\,\text{Recall}}. \tag{6}$$

Table 1 shows estimated speed up for different percentage P of truly solvable 6-tuples in a dataset:

Table 1: Estimated Speed Up

P	Kept (%)	Removed (%)	Speed-up (S)
10%	$\frac{0.10 \cdot 0.90}{0.70} = 12.9\%$	87.1%	$\frac{0.70}{0.10 \cdot 0.90} \approx 7.8$
1%	$\frac{0.01 \cdot 0.90}{0.70} = 1.29\%$	98.71%	$\frac{0.70}{0.01 \cdot 0.90} \approx 77$
0.1%	$\frac{0.001 \cdot 0.90}{0.70} = 0.129\%$	99.871%	$\frac{0.70}{0.001 \cdot 0.90} \approx 778$

These calculations show that even moderate pruning performance can significantly reduce the number of solver calls in practical RANSAC pipelines.

3.5.2 Experiments on the Real Data from COLMAP

Now we attempt to train and incorporate a pruning proceduRe into existing 3D reconstruction pipeline of COLMAP discussed in section 1.2.3. Dataset of 128 images of the "South" building at UNC Chapel Hill is used [Zac]. We implement the following algorithm in Python to extract feature matches from the COLMAP database:

Algorithm 4 Matches Extraction

Require: Set of 128 images and their feature matches identified by COLMAP

Randomly select 100 pairs of images out of the 128 images.
for each pair in selected image pairs **do**
  Randomly select 10,000 6-tuples out of $\binom{N}{6}$ point correspondences
  for all 5-tuples drawn from each 6-tuple **do**
    Solve for essential matrix E using the 5-point algorithm
    if the held-out point does satisfy E **then**
      Label the 6-tuple as "solvable"
    else if all 5-point subsets have been tested **then**
      Label the 6-tuple as "not solvable"
    else
      continue testing next 5-point subset
    end if
  end for
end for

After extraction phase, we utilize the algorithm discussed in section 3.4.2 to reduce dimension to 18 using the straightening transformation. Then we proceed to training three models (A, B, C) of different sizes. Each model is trained on 10^6 data points for 30 epochs with 128 as a batch size. As before, ReLU is used as an activation function and BCE as a loss function. The pruning procedure is injected into COLMAP'S RANSAC method as a custom sampler. Tests are run on CPU (no CUDA support). We use early stopping to get the best model over all epochs. Further we discuss a performance of each model.

Model A. Model A is a fully-connected multi-layer perceptron with seven layers. Two hidden layers have 256 neurons each, the next two have 128 neurons each, followed by two layers with 64 and 32 neurons. The output layer is only 1 neuron. The table below demonstrates this architecture.

Table 2: Architecture of the model A.

Layer	Input Dim	Output Dim	Activation
1	d_{in}	256	ReLU
2	256	256	ReLU
3	256	128	ReLU
4	128	128	ReLU
5	128	64	ReLU
6	64	32	ReLU
7	32	1	(none)

Figure 7: Metrics of the Model A

This is the biggest model out of three we trained. The best accuracy the model reaches is 65%, with 30% precision and 67% recall. Obviously, this performance is worse when compared to the one we had in the fabricated scenario section 3.5.1, since constraints that were used during the simulations are now gone. Despite that, the size of the model allows it to reach decent performance in the classification task. In the experiment on real data, model A accepted only 35% of all tuples of point-to-point correspondences, while yielding a solution with essentially same size of a consensus set. Precisely,

$$\frac{|\text{Consensus set of } E_{def}|}{|\text{Consensus set of } E_{prune}|} = 1.00131$$

, where E_{def} - the best solution found by default RANSAC, E_{prune} - the best solution found by RANSAC with pruning procedure. That indicates that we successfully can get rid off 65% of spurious correspondences. Unfortunately, this advantage does not translates into the desired speed up. Since the model is too big and we used CPU only, it takes a lot of time to run the pruning procedure for each separate pick of 6 correspondences. As a result, the performance is 15 times slower on average. We discuss possible ways to overcome this challenge in section 4.

Model B. In the attempt to make model faster, we reduce a number of layers and neurons in them. Model B is a fully-connected, feed-forward neural network with three layers. The input of dimension 18 is projected to 64 units, followed by a ReLU activation. A second hidden layer then maps these 64 units to 32 units, again with ReLU. Finally, a single-unit output layer (no activation) produces the scalar prediction. Table 4 summarizes this architecture.

Table 3: Architecture of Model B.

Layer	Input Dim	Output Dim	Activation
1	18	64	ReLU
2	64	32	ReLU
3	32	1	(none)

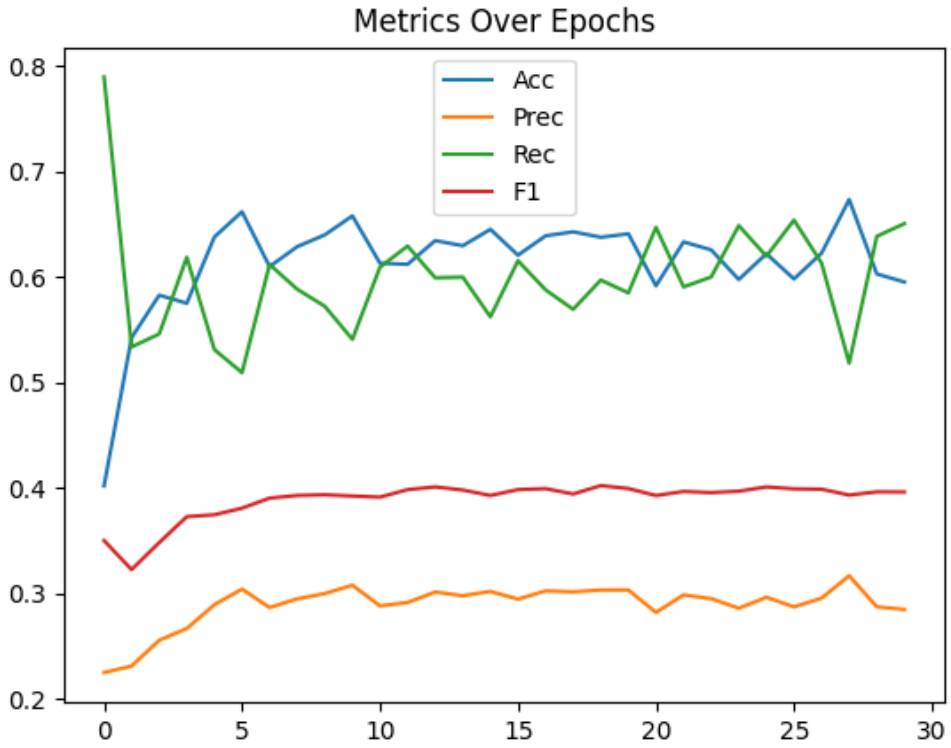

Figure 8: Metrics of the Model B

This is the medium model out of the three we trained. Total accuracy reaches 60%, with 29% precision and 60% recall. This demonstrates, that the size of model A did not provide a big advantage. Compared to the fabricated-scenario network in section 3.5.1, performance is naturally lower since those simulation constraints are no longer enforced. Nevertheless, the compact size still yields decent classification results.

On real data, Model B accepts 41% of all tuples of point-to-point correspondences. Ratio of consensus sets are still close to 1:

$$\frac{|\text{Consensus set of } E_{def}|}{|\text{Consensus set of } E_{prune}|} = 1.003,$$

Such a consensus set performance is to be expected, since the model B throws away less tuples than model A. When compared to default RANSAC, model B runs 10 times slower, which is an improvement over model A, but is still poor.

Model C. Model C is attempted to be even smaller than model B. It is fully-connected, feed-forward neural network with three layers. The input of dimension 18 is projected to 32 units, followed by a ReLU activation. A second hidden layer then maps these 32 units to 32 units, again with ReLU. Finally, a single-unit output layer (no activation) produces the scalar prediction. Table 4 summarizes this architecture.

Table 4: Architecture of Model C.

Layer	Input Dim	Output Dim	Activation
1	18	32	ReLU
2	32	32	ReLU
3	32	1	(none)

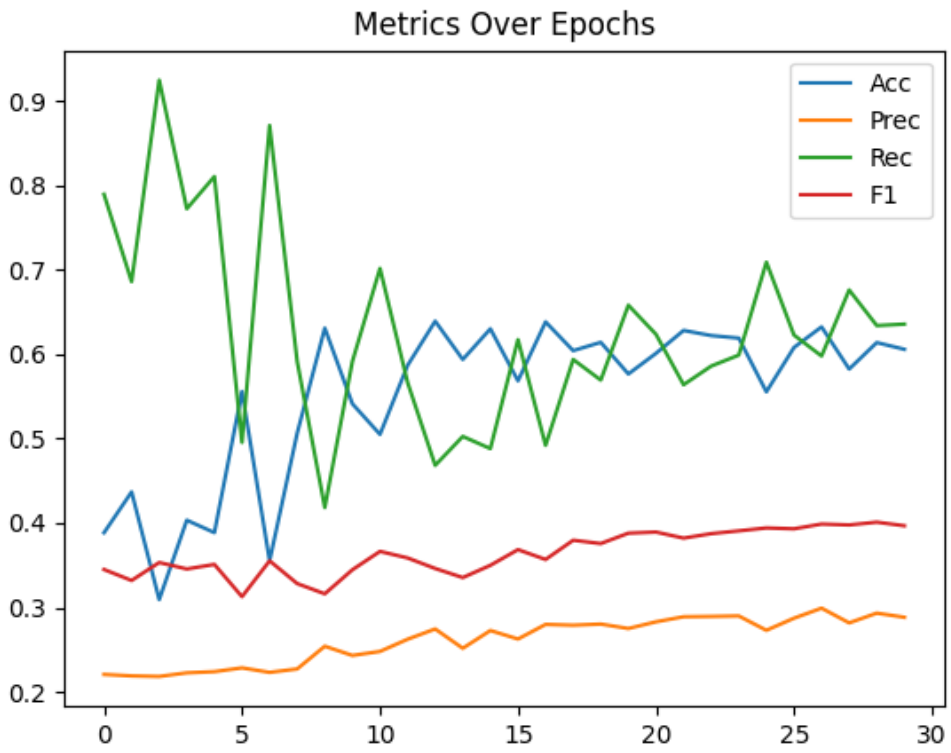

Figure 9: Metrics of the Model C

This is the medium model out of the three we trained. Total accuracy reaches 58%, with 24% precision and 49% recall. On real data, Model C accepts 54% of all tuples of point-to-point correspondences. Ratio of consensus sets are still close to 1. In terms of speed, model C does not give an improvement over model B, since the reduction in the number of neurons was not significant enough.

Summary In this set of experiments on real COLMAP data, Model A achieves the highest classification accuracy (65%) and recall (67%) but at the cost of low precision (30%) and a substantial slowdown (15 times) due to its depth and CPU-only evaluation. Model B offers a balanced trade-off, with 60% accuracy, 29% precision, and 60% recall, while reducing the slowdown to 10 times and increasing the acceptance rate to 41%. Model C further shrinks the network, yielding 58% accuracy, 24% precision, and 49% recall, and an acceptance rate of 54%, but its computational speed remains similar to Model B. Overall, Model B provides the best compromise between performance and efficiency, whereas Model A maximizes accuracy at a heavy computational cost, and Model C offers minimal gains in speed at the expense of predictive quality.

Table 5: Comparison of pruning models on real COLMAP data

Model	Accuracy	Precision	Recall	Acceptance Rate	Speed Factor
A	65%	30%	67%	35%	15×
B	60%	29%	60%	41%	10×
C	58%	24%	49%	54%	10×

4 Conclusions

In this work, we have introduced a novel *solvability pruning* framework for minimal-problem solvers in computer vision, specifically targeting the relative pose estimation task within RANSAC-based pipelines. By injecting a lightweight neural pruning network ϕ that discriminates between solvable and unsolvable over-determined samples, we significantly reduce the number of expensive algebraic solver calls. Our approach leverages a carefully designed *straightening transformation* in projective space to canonicalize pairs of point correspondences and employs a Cayley parametrization of rotations in SO(3) to compress each 6-tuple of correspondences into an 18-dimensional feature vector. This dimensionality reduction preserves all information relevant to solvability while minimizing computational overhead during classification.

We presented a detailed construction of the straightening procedure, proved its existence and uniqueness up to sign choices, and demonstrated how three successive pairs of correspondences yield a reduced representation of the six-point minimal problem. Our synthetic experiments on fabricated camera pairs—where the cameras share a translation vector and exhibit random relative rotations—provided a clear proof of concept: the classifier attained over 90% recall and 70% precision on held-out data, translating to theoretical speed-up factors. These results validate the potential of solvability pruning in scenarios where the cost of algebraic solver invocations dominates overall runtime.

In real-data experiments integrated within the COLMAP Structure-from-Motion pipeline, we trained three variants of the pruning network (Models A, B, and C) on one million extracted six-tuples drawn from feature matches on a 128-image dataset of UNC's "South" building. While the performance on real data was lower than in the synthetic setting—owing to the increased complexity and noise—the networks still achieved meaningful reductions in solver calls, discarding up to 65% of unsolvable samples while preserving consensus-set quality (with consensus-size ratios near 1.00). These findings demonstrate that solvability pruning can be incorporated into production pipelines with minimal degradation of reconstruction accuracy.

Despite these promising outcomes, we encountered practical limitations in our CPU-only implementation. The overhead of running the classifier on each sample eroded the theoretical speed gains, resulting in a net slowdown when using larger networks (Model A) and only modest improvements with smaller architectures (Models B and C). This highlights the critical importance of optimizing inference performance—through model compression, quantization, or hardware acceleration—before real-time or large-scale deployment. Furthermore, our experiments focused on a single minimal problem (the six-point essential-matrix solver); the generality of the approach to other minimal solvers, noise regimes, and feature-matching pipelines remains to be fully explored.

In summary, solvability pruning presents a compelling paradigm for reducing redundant algebraic computations in minimal-problem solvers. By learning to reject unsolvable over-determined subsets, our framework achieves dramatic reductions in solver calls under favorable conditions and shows robust behavior on real-world data. The combination of group-action-based dimensionality reduction and lightweight neural classification offers a flexible template for accelerating a wide range of RANSAC-style estimation problems in computer vision, robotics and state estimation.

4.1 Future Work

In the coming stages of this research, we plan to develop solvability pruning approach further. A primary goal involves rethinking the classifier architecture itself. While fully connected perceptrons provided a straightforward proof of concept, they treat all input dimensions homogeneously and can be suboptimal in capturing the intrinsic structure of geometrical point-correspondence data. To address this, one can investigate convolutional encodings, graph neural networks, and transformer-style attention modules that can dynamically weight the influence of each correspondence. These architectures might yield higher precision–recall trade-offs while reducing the total parameter count and inference time.

Hardware acceleration represents another critical frontier. Our experiments thus far have been confined to CPU inference, but modern GPUs excel at bulk tensor operations and parallel sampling workflows. We will therefore redesign the sampling pipeline to generate large batches of candidate subsets—on the order of hundreds or thousands of six-tuples—then process them in a single forward pass on the GPU. This batched approach not only reduces classification overhead but also enables asynchronous overlap between pruning and solver execution. By grouping tuples into contiguous GPU memory buffers, we anticipate achieving significant time improvements over standard RANSAC method in COLMAP.

Although the six-point essential matrix problem served as our initial test field, the underlying principles of solvability pruning extend naturally to other minimal solvers. One should next consider applying our framework to the seven-point fundamental matrix algorithm, which introduces additional algebraic constraints and solution multiplicities. By training the pruning network on seven-point samples—each encoded with the straightening and Cayley transform pipeline- scenarios with uncalibrated cameras can be accelerated.

Finally, to ground empirical findings in formal guarantees, one of the next steps is a development of a strict probabilistic framework that links classifier performance metrics—precision, recall, and rejection rate—to RANSAC's convergence probability and expected number of iterations. By modeling the distribution of solvable samples as a Bernoulli process and incorporating classifier decision errors, we can attempt to derive bounds on the probability that RANSAC finds a correct model within a given computational budget. These theoretical analyses will provide practitioners with concrete guidelines for choosing classifier thresholds, sample sizes, and iteration counts, striking the right balance between speed and reliability.

Through these concerted efforts—ranging from neural-architecture search to hardware-accelerated batching, from expansion to seven-point solvers and higher-order over-sampling, to rigorous probabilistic analysis—solvability pruning can be transformed from a promising concept into a robust, high-performance component of modern geometric estimation pipelines.

References

[BSHW13] Daniel J. Bates, Andrew J. Sommese, Jonathan D. Hauenstein, and Charles W. Wampler. *Numerically Solving Polynomial Systems with Bertini*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013. _eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611972702.

[CM05] Ondrej Chum and Jiri Matas. Matching with prosac " progressive sample consensus. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, page 220–226, USA, 2005. IEEE Computer Society.

[CMK03] Ondrej Chum, Jiri Matas, and Josef Kittler. Locally optimized ransac. volume 2781, pages 236–243, 09 2003.

[COL] COLMAP developers. Tutorial: COLMAP 3.12.0.dev0 documentation. Accessed: 29 May 2025.

[DIT19] Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. *The International Journal of Robotics Research*, 38(12-13):1420–1441, October 2019. Publisher: SAGE Publications Ltd STM.

[DJ] Dinesh Manocha and John Canny. Efficient Inverse Kinematics for General 6R Manipulators.

[DS14] Christian Doppler and Frantisek Josef Studnicka. *Ueber Das Farbige Licht Der Doppelsterne Und Einiger Anderer Gestirne Des Himmels. Versuch Einer Das Bradley'sche Aberrations-theorem ALS Integrirenden*. Nabu Press, January 2014.

[FB81] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FSQ+15] Emad Felemban, Faisal Karim Shaikh, Umair Mujtaba Qureshi, Adil A. Sheikh, and Saad Bin Qaisar. Underwater Sensor Network Applications: A Comprehensive Survey. *International Journal of Distributed Sensor Networks*, 11(11):896832, November 2015. Publisher: SAGE Publications.

[GPM+20] Douglas Gillespie, Laura Palmer, Jamie Macaulay, Carol Sparling, and Gordon Hastie. Passive acoustic methods for tracking the 3D movements of small cetaceans around marine structures. *PLOS ONE*, 15(5):e0229058, May 2020. Publisher: Public Library of Science.

[HDLP23] Petr Hruby, Timothy Duff, Anton Leykin, and Tomas Pajdla. Learning to solve hard minimal problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–16, 2023.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015. Preprint on arXiv:1412.6980.

[MLC23] Michela Mancini, Anton Leykin, and John A. Christian. State estimation of a moving frequency source from observations at multiple receivers, August 2023. arXiv:2308.05223 [math].

[Mor09] Alexander Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, January 2009.

[PyT] PyTorch contributors. PyTorch: An imperative style, high-performance deep learning library. Accessed: 29 May 2025.

[SF16a] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[SF16b] Johannes Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. June 2016.

[SZFP16] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise View Selection for Unstructured Multi-View Stereo. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 501–518, Cham, 2016. Springer International Publishing.

[SZPF16] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[tab24] Mct 042 YBC 07289 artifact entry. https://cdli.earth/P255048, nov 6 2024. [Online; accessed 2025-05-13].

[TMHP20] Pavel Trutman, Safey El Din Mohab, Didier Henrion, and Tomas Pajdla. Globally Optimal Solution to Inverse Kinematics of 7DOF Serial Manipulator, July 2020. arXiv:2007.12550 [cs].

[Tom] Tomas Pajdla. *Elements of Geometry for CV*.

[Wik] Wikipedia contributors. Denavit–Hartenberg parameters. Accessed: 29 May 2025.

[Zac] Christopher Zach. South building dataset: 128 images of the “south” building at unc chapel hill. Accessed: 29 May 2025.