Reconstruction Problems in Computer Vision

Applications and Algebra

Felix Rydell

KTH Stockholm

Agenda

The aim of this presentation is to introduce computer vision with applications, motivate and define basic concepts of algebraic geometry, and finally I will talk about a current project.

This talk is divided into three parts:

1. Computer vision, applications and reconstruction problems,
2. Introduction to algebraic geometry,
3. Algebraic vision, my current project.

Agenda

The aim of this presentation is to introduce computer vision with applications, motivate and define basic concepts of algebraic geometry, and finally I will talk about a current project.

This talk is divided into three parts:

1. Computer vision, applications and reconstruction problems,
2. Introduction to algebraic geometry,
3. Algebraic vision, my current project.

Agenda

The aim of this presentation is to introduce computer vision with applications, motivate and define basic concepts of algebraic geometry, and finally I will talk about a current project.

This talk is divided into three parts:

1. Computer vision, applications and reconstruction problems,
2. Introduction to algebraic geometry,
3. Algebraic vision, my current project.

Agenda

The aim of this presentation is to introduce computer vision with applications, motivate and define basic concepts of algebraic geometry, and finally I will talk about a current project.

This talk is divided into three parts:

1. Computer vision, applications and reconstruction problems,
2. Introduction to algebraic geometry,
3. Algebraic vision, my current project.

Agenda

The aim of this presentation is to introduce computer vision with applications, motivate and define basic concepts of algebraic geometry, and finally I will talk about a current project.

This talk is divided into three parts:

1. Computer vision, applications and reconstruction problems,
2. Introduction to algebraic geometry,
3. Algebraic vision, my current project.

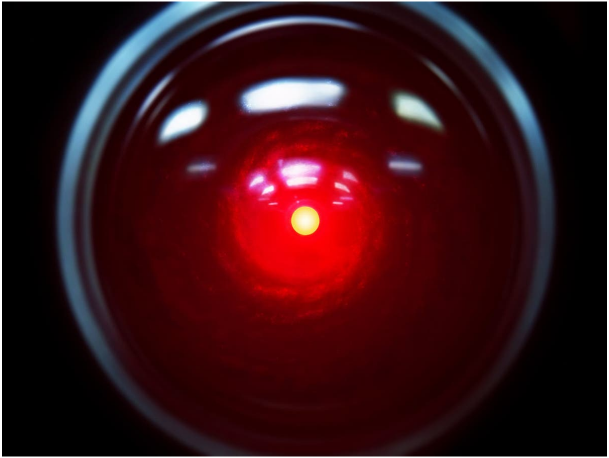


Figure: Hal 9000, the artificial intelligence from *2001: A Space Oddysey (1968)*.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

What Is Computer Vision?

- *Computer vision* is an umbrella term. It is essentially the study of how computers can learn information from images (and videos.) This is the *perceptual* component of intelligence. It seeks to understand and automate tasks that the human visual system can do.

- Examples of tasks in computer vision:
  - Reading hand-written text, number plates (convolutional neural networks,)
  - Self-driving cars,
  - Deep-fakes, making faces older or younger in films,
  - Surveillance, monitoring pools for drowing people,
  - Object recognition for automated check out lanes,
  - Increasing resolution of pictures and denoising (auto-encoders,)
  - **Reconstruction of 3D objects from images.**

- **Mathematics:** The mathematical models and theory. Description and classification of problems. **Statistics:** Estimations of accuracy. **Engineering:** Techniques and algorithms.

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.
- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.
- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  – Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  – Better spacial understanding for self-driving vehicles,
  – Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  – Input a number of images,
  – Find points and lines that two cameras have incommon,
  – Solve the reconstruction problem with algebra,
  – Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  – Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  – Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  – Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  – Better spacial understanding for self-driving vehicles,
  – Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  – Input a number of images,
  – Find points and lines that two cameras have incommon,
  – Solve the reconstruction problem with algebra,
  – Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  – Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  – Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  - Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  - Better spacial understanding for self-driving vehicles,
  - Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  - Input a number of images,
  - Find points and lines that two cameras have incommon,
  - Solve the reconstruction problem with algebra,
  - Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  - Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  - Or we don't have any information about the cameras (pictures from the internet.)

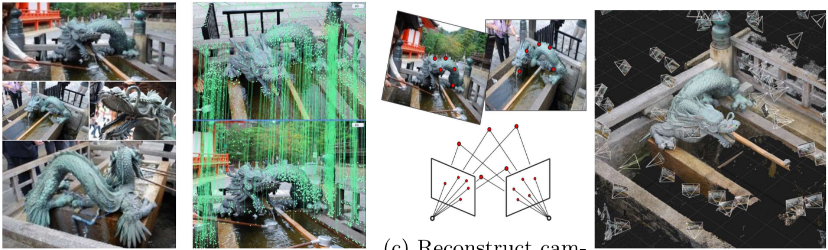
Reconstructing 3D models

- *Reconstruction of 3D objects from images* is the main focus of this talk and it has several applications:
  – Automated construction of 3D models of cities from aerial photograps. This is used by cityplanners and moviemakers,
  – Better spacial understanding for self-driving vehicles,
  – Modelling humans and objects, used for films and videogames.

- The *reconstruction pipeline* works as follows:
  – Input a number of images,
  – Find points and lines that two cameras have incommon,
  – Solve the reconstruction problem with algebra,
  – Output a 3D model.

- Note that there are two flavours of reconstruction problems.
  – Either we know the positions of the cameras and their orientation (self-driving vehicles,)
  – Or we don't have any information about the cameras (pictures from the internet.)

Reconstructing 3D models



(a) Input images (b) Image matching (c) Reconstruct cameras and 3D points (d) Output

Figure: The reconstruction pipeline.

Reconstructing 3D models

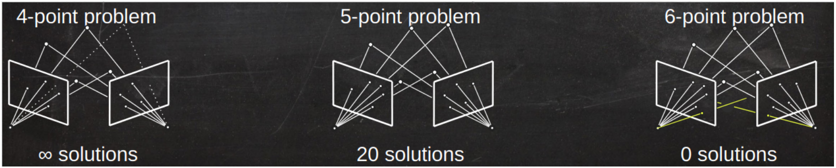


Figure: Different algebraic reconstruction problems. The number of complex solutions are listed for generic positions (up to rotation and translation.)

Reconstructing 3D models

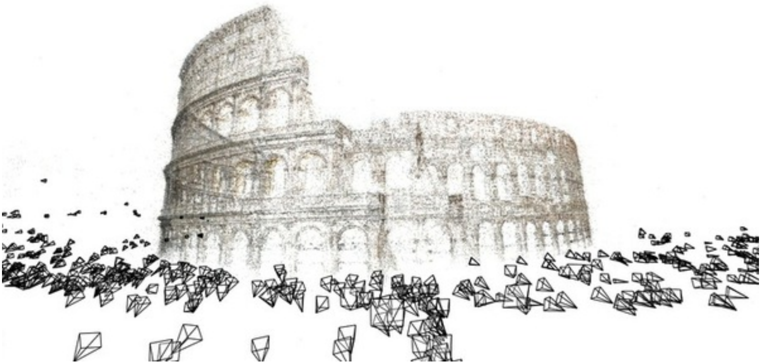


Figure: Point reconstruction of the colloseum using pictures found on the internet taken by tourists' smartphones. Also the camera position (and their directions) have been recovered. Taken from *Reconstructing Rome,* by Agarwal, Furukawa et. al.

Reconstructing 3D models

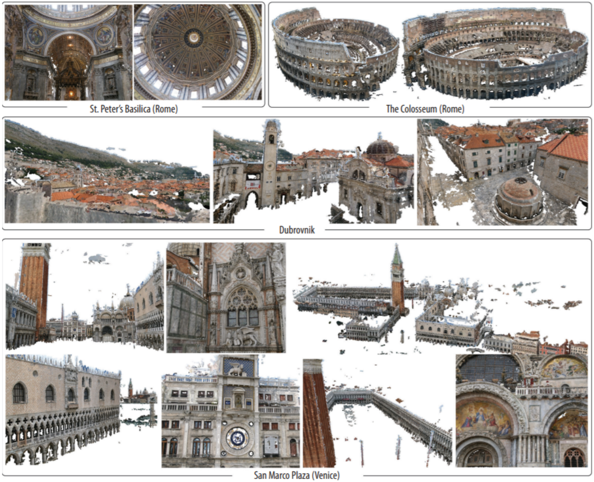


Figure: Finished 3D models. The pictures used were taken by tourists. Taken from *Reconstructing Rome*, by Agarwal, Furukawa et. al.

Types of Cameras

- There are many types of cameras, and in some cases the type really matters in applications:
  - *Pinhole camera:* The "easiest" kind, takes the whole picture at once. Well-understood from a mathematical perspective,
  - *Rolling shutter camera:* Common in smartphones, scans each line of the image one by one,
  - *Pushbroom camera:* Common in satellites, scans a specific line over and over (so that if stands still it scans the same line repeatedly.)

- The difference between the pinhole camera and rolling shutter camera becomes relevant when the camera is moving. For a moving rolling shutter camera, lines can become conics. Visualization.

Types of Cameras

- There are many types of cameras, and in some cases the type really matters in applications:
  - *Pinhole camera:* The “easiest” kind, takes the whole picture at once. Well-understood from a mathematical perspective,
  - *Rolling shutter camera:* Common in smartphones, scans each line of the image one by one,
  - *Pushbroom camera:* Common in satellites, scans a specific line over and over (so that if stands still it scans the same line repeatedly.)

- The difference between the pinhole camera and rolling shutter camera becomes relevant when the camera is moving. For a moving rolling shutter camera, lines can become conics. Visualization.

Types of Cameras

- There are many types of cameras, and in some cases the type really matters in applications:
  - *Pinhole camera:* The "easiest" kind, takes the whole picture at once. Well-understood from a mathematical perspective,
  - *Rolling shutter camera:* Common in smartphones, scans each line of the image one by one,
  - *Pushbroom camera:* Common in satellites, scans a specific line over and over (so that if stands still it scans the same line repeatedly.)

- The difference between the pinhole camera and rolling shutter camera becomes relevant when the camera is moving. For a moving rolling shutter camera, lines can become conics. Visualization.

Types of Cameras

- There are many types of cameras, and in some cases the type really matters in applications:
  - *Pinhole camera:* The "easiest" kind, takes the whole picture at once. Well-understood from a mathematical perspective,
  - *Rolling shutter camera:* Common in smartphones, scans each line of the image one by one,
  - *Pushbroom camera:* Common in satellites, scans a specific line over and over (so that if stands still it scans the same line repeatedly.)

- The difference between the pinhole camera and rolling shutter camera becomes relevant when the camera is moving. For a moving rolling shutter camera, lines can become conics. Visualization.

Types of Cameras

- There are many types of cameras, and in some cases the type really matters in applications:
  - *Pinhole camera:* The "easiest" kind, takes the whole picture at once. Well-understood from a mathematical perspective,
  - *Rolling shutter camera:* Common in smartphones, scans each line of the image one by one,
  - *Pushbroom camera:* Common in satellites, scans a specific line over and over (so that if stands still it scans the same line repeatedly.)

- The difference between the pinhole camera and rolling shutter camera becomes relevant when the camera is moving. For a moving rolling shutter camera, lines can become conics. Visualization.

Types of Cameras

Figure: Picture taken with a rolling shutter camera shows bent rotor blades. In our 3D reconstruction, we want the rotor blades to be straight.

Part 2 – Algebraic Geometry

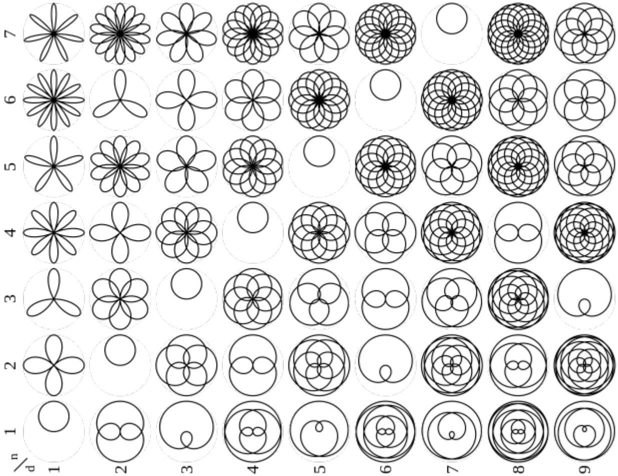


Figure: The family of *rose curves*.

Motivation

- *Algebraic geometry* is the study of solutions to polynomial equations. For example,

$$V = \{(x, y) \in K^2 : x^2 + y^2 = 1\},$$

where K is a field (usually $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$.)

- Observe that $\mathbb{C}$ is an algebraically closed field (due to the fundamental theorem of algebra,) which means that its geometry is "nice" and "simple." The real numbers $\mathbb{R}$ model the real world and computers can symbolically work with rational numbers $\mathbb{Q}$.

- This field can be thought of as a generalization of linear algebra, which is the study of the solution set of linear equations.

- The world can be understood through polynomials! In particular, real valued functions can in a broad sense be approximated by polynomials, via Taylor approximation or Stone-Weierstrass theorem.

Motivation

- *Algebraic geometry* is the study of solutions to polynomial equations. For example,
$$V = \{(x,y) \in K^2 : x^2 + y^2 = 1\},$$
where K is a field (usually $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$.)

- Observe that $\mathbb{C}$ is an algebraically closed field (due to the fundamental theorem of algebra,) which means that its geometry is "nice" and "simple." The real numbers $\mathbb{R}$ model the real world and computers can symbolically work with rational numbers $\mathbb{Q}$.

- This field can be thought of as a generalization of linear algebra, which is the study of the solution set of linear equations.

- The world can be understood through polynomials! In particular, real valued functions can in a broad sense be approximated by polynomials, via Taylor approximation or Stone-Weierstrass theorem.

Motivation

- *Algebraic geometry* is the study of solutions to polynomial equations. For example,
$$V = \{(x, y) \in K^2 : x^2 + y^2 = 1\},$$
where K is a field (usually $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$.)

- Observe that $\mathbb{C}$ is an algebraically closed field (due to the fundamental theorem of algebra,) which means that its geometry is "nice" and "simple." The real numbers $\mathbb{R}$ model the real world and computers can symbolically work with rational numbers $\mathbb{Q}$.

- This field can be thought of as a generalization of linear algebra, which is the study of the solution set of linear equations.

- The world can be understood through polynomials! In particular, real valued functions can in a broad sense be approximated by polynomials, via Taylor approximation or Stone-Weierstrass theorem.

Motivation

- *Algebraic geometry* is the study of solutions to polynomial equations. For example,
$$V = \{(x, y) \in K^2 : x^2 + y^2 = 1\},$$
where K is a field (usually $\mathbb{Q}$, $\mathbb{R}$ or $\mathbb{C}$.)

- Observe that $\mathbb{C}$ is an algebraically closed field (due to the fundamental theorem of algebra,) which means that its geometry is "nice" and "simple." The real numbers $\mathbb{R}$ model the real world and computers can symbolically work with rational numbers $\mathbb{Q}$.

- This field can be thought of as a generalization of linear algebra, which is the study of the solution set of linear equations.

- The world can be understood through polynomials! In particular, real valued functions can in a broad sense be approximated by polynomials, via Taylor approximation or Stone-Weierstrass theorem.

Geometry vs. Algebra

- Consider the ring of polynomials $K[x_1, \ldots, x_n]$ ($K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$.) An ideal $\mathcal{I}$ of this ring is a subring such that $f \cdot \mathcal{I} \subseteq \mathcal{I}$ for any polynomial $f \in K[x_1, \ldots, x_n]$. The ideal generated by polynomials $f_1, \ldots, f_k$ is

$$\langle f_1, \ldots, f_k \rangle := \left\{ \sum_{i=1}^{k} a_i f_i : a_i \in K[x_1, \ldots, x_n] \right\}.$$

- An *algebraic set* (variety) $\mathcal{V}(\mathcal{I})$ is the set of points in K^n for which all polynomials in $\mathcal{I}$ vanish. **Key idea:** If $f_1, \ldots, f_k$ generate $\mathcal{I}$, then

$$\mathcal{V}(\mathcal{I}) = \{x \in K^n : f_1(x) = \cdots = f_n(x) = 0\}.$$

A computer understands algebraic geometry (an ideal) as a set of finitely many generators:

Theorem (Hilbert's Basis Theorem)

If $K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$ then any ideal of $K[x_1, \ldots, x_n]$ is finitely generatored.

Geometry vs. Algebra

- Consider the ring of polynomials $K[x_1, \ldots, x_n]$ ($K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$.) An ideal $\mathcal{I}$ of this ring is a subring such that $f \cdot \mathcal{I} \subseteq \mathcal{I}$ for any polynomial $f \in K[x_1, \ldots, x_n]$. The ideal generated by polynomials $f_1, \ldots, f_k$ is

$$\langle f_1, \ldots, f_k \rangle := \left\{ \sum_{i=1}^{k} a_i f_i : a_i \in K[x_1, \ldots, x_n] \right\}.$$

- An *algebraic set* (variety) $\mathcal{V}(\mathcal{I})$ is the set of points in K^n for which all polynomials in $\mathcal{I}$ vanish. **Key idea:** If $f_1, \ldots, f_k$ generate $\mathcal{I}$, then

$$\mathcal{V}(\mathcal{I}) = \{x \in K^n : f_1(x) = \cdots = f_n(x) = 0\}.$$

A computer understands algebraic geometry (an ideal) as a set of finitely many generators:

Theorem (Hilbert's Basis Theorem)

If $K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$ then any ideal of $K[x_1, \ldots, x_n]$ is finitely generatored.

Geometry vs. Algebra

- Consider the ring of polynomials $K[x_1, \ldots, x_n]$ ($K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$.) An ideal $\mathcal{I}$ of this ring is a subring such that $f \cdot \mathcal{I} \subseteq \mathcal{I}$ for any polynomial $f \in K[x_1, \ldots, x_n]$. The ideal generated by polynomials $f_1, \ldots, f_k$ is

$$\langle f_1, \ldots, f_k \rangle := \left\{ \sum_{i=1}^{k} a_i f_i : a_i \in K[x_1, \ldots, x_n] \right\}.$$

- An *algebraic set* (variety) $\mathcal{V}(\mathcal{I})$ is the set of points in K^n for which all polynomials in $\mathcal{I}$ vanish. **Key idea:** If $f_1, \ldots, f_k$ generate $\mathcal{I}$, then

$$\mathcal{V}(\mathcal{I}) = \{x \in K^n : f_1(x) = \cdots = f_n(x) = 0\}.$$

A computer understands algebraic geometry (an ideal) as a set of finitely many generators:

Theorem (Hilbert's Basis Theorem)

If $K = \mathbb{Q}, \mathbb{R}, \mathbb{C}$ then any ideal of $K[x_1, \ldots, x_n]$ is finitely generatored.

Projective Geometry

- Projective geometry was "first studied" by painters long ago who wanted to draw proportions of structures and buildings in a realistic way.
- **Key idea:** Compact sets have nice properties that are nice to work with. Sets as $\mathbb{R}^n$ and $\mathbb{C}^n$ are non-compact, but this can be fixed via *projectivization.*
- For a field K, we define the projective space

$$\mathbb{P}_K^{n-1} := \{L \subseteq K^n : L \text{ is } 1-\dim \text{ linear subspace}\} = (K^n \setminus \{0\})/\sim,$$

where $x, y \in K^n$ are related by $\sim$ if they differ by a non-zero constant. Its elements are written $(x_1 : \cdots : x_n)$, for some $(x_1, \ldots, x_n) \in K^n$ that spans the line.

- **Key idea:** Just as algebraic geometry becomes easier over $\mathbb{C}$ instead of $\mathbb{R}$, it also becomes easier in projective space compared to the usual *affine space.*
- Human vision works by identifying lines going through the center of the eye with points.

Projective Geometry

- Projective geometry was "first studied" by painters long ago who wanted to draw proportions of structures and buildings in a realistic way.
- **Key idea:** Compact sets have nice properties that are nice to work with. Sets as $\mathbb{R}^n$ and $\mathbb{C}^n$ are non-compact, but this can be fixed via *projectivization.*

- For a field K, we define the projective space

$$\mathbb{P}_K^{n-1} := \{L \subseteq K^n : L \text{ is } 1-\dim \text{ linear subspace}\} = (K^n \setminus \{0\})/\sim,$$

where $x, y \in K^n$ are related by $\sim$ if they differ by a non-zero constant. Its elements are written $(x_1 : \cdots : x_n)$, for some $(x_1, \ldots, x_n) \in K^n$ that spans the line.

- **Key idea:** Just as algebraic geometry becomes easier over $\mathbb{C}$ instead of $\mathbb{R}$, it also becomes easier in projective space compared to the usual *affine space.*

- Human vision works by identifying lines going through the center of the eye with points.

Projective Geometry

- Projective geometry was "first studied" by painters long ago who wanted to draw proportions of structures and buildings in a realistic way.
- **Key idea:** Compact sets have nice properties that are nice to work with. Sets as $\mathbb{R}^n$ and $\mathbb{C}^n$ are non-compact, but this can be fixed via *projectivization.*

- For a field K, we define the projective space

$$\mathbb{P}_K^{n-1} := \{L \subseteq K^n : L \text{ is } 1-\dim \text{ linear subspace}\} = (K^n \setminus \{0\})/\sim,$$

where $x, y \in K^n$ are related by $\sim$ if they differ by a non-zero constant. Its elements are written $(x_1 : \cdots : x_n)$, for some $(x_1, \ldots, x_n) \in K^n$ that spans the line.

- **Key idea:** Just as algebraic geometry becomes easier over $\mathbb{C}$ instead of $\mathbb{R}$, it also becomes easier in projective space compared to the usual *affine space.*

- Human vision works by identifying lines going through the center of the eye with points.

Projective Geometry

- Projective geometry was "first studied" by painters long ago who wanted to draw proportions of structures and buildings in a realistic way.
- **Key idea:** Compact sets have nice properties that are nice to work with. Sets as $\mathbb{R}^n$ and $\mathbb{C}^n$ are non-compact, but this can be fixed via *projectivization.*

- For a field K, we define the projective space

$$\mathbb{P}_K^{n-1} := \{L \subseteq K^n : L \text{ is } 1-\dim \text{ linear subspace}\} = (K^n \setminus \{0\})/\sim,$$

where $x, y \in K^n$ are related by $\sim$ if they differ by a non-zero constant. Its elements are written $(x_1 : \cdots : x_n)$, for some $(x_1, \ldots, x_n) \in K^n$ that spans the line.

- **Key idea:** Just as algebraic geometry becomes easier over $\mathbb{C}$ instead of $\mathbb{R}$, it also becomes easier in projective space compared to the usual *affine space.*

- Human vision works by identifying lines going through the center of the eye with points.

Projective Geometry

- Projective geometry was "first studied" by painters long ago who wanted to draw proportions of structures and buildings in a realistic way.
- **Key idea:** Compact sets have nice properties that are nice to work with. Sets as $\mathbb{R}^n$ and $\mathbb{C}^n$ are non-compact, but this can be fixed via *projectivization.*

- For a field K, we define the projective space

$$\mathbb{P}_K^{n-1} := \{L \subseteq K^n : L \text{ is } 1-\dim \text{ linear subspace}\} = (K^n \setminus \{0\})/\sim,$$

where $x, y \in K^n$ are related by $\sim$ if they differ by a non-zero constant. Its elements are written $(x_1 : \cdots : x_n)$, for some $(x_1, \ldots, x_n) \in K^n$ that spans the line.

- **Key idea:** Just as algebraic geometry becomes easier over $\mathbb{C}$ instead of $\mathbb{R}$, it also becomes easier in projective space compared to the usual *affine space.*

- Human vision works by identifying lines going through the center of the eye with points.

Grassmannians

- One example of a common algebraic construction is the *Grassmanian*, defined as

$$\mathrm{Gr}(k, \mathbb{R}^n) := \{L \subseteq \mathbb{R}^n : L \text{ linear space of dim } k\}.$$

In terms of projective space, we have a natural correspondence

$$\mathrm{Gr}(k-1, \mathbb{P}^{n-1}_{\mathbb{R}}) \cong \mathrm{Gr}(k, \mathbb{R}^n).$$

- **Key idea:** These can be viewed as algebraic sets. This allows us to work efficiently with linear spaces, viewing them as points.

- For example, $\mathrm{Gr}(1, \mathbb{P}^3) \cong \mathrm{Gr}(2, \mathbb{R}^4)$ lie in $\mathbb{P}^5$ by the following *Plücker embedding* sending the space spanned by a, b, represented as the matrix

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix}$$

to the vector of the six 2×2 minors of this matrix in $\mathbb{P}^5$. This is a *natural* bijection.

Grassmannians

- One example of a common algebraic construction is the *Grassmanian*, defined as

$$\mathrm{Gr}(k, \mathbb{R}^n) := \{L \subseteq \mathbb{R}^n : L \text{ linear space of dim } k\}.$$

In terms of projective space, we have a natural correspondence

$$\mathrm{Gr}(k-1, \mathbb{P}^{n-1}_{\mathbb{R}}) \cong \mathrm{Gr}(k, \mathbb{R}^n).$$

- **Key idea:** These can be viewed as algebraic sets. This allows us to work efficiently with linear spaces, viewing them as points.

- For example, $\mathrm{Gr}(1, \mathbb{P}^3) \cong \mathrm{Gr}(2, \mathbb{R}^4)$ lie in $\mathbb{P}^5$ by the following *Plücker embedding* sending the space spanned by a, b, represented as the matrix

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix}$$

to the vector of the six 2×2 minors of this matrix in $\mathbb{P}^5$. This is a *natural* bijection.

Grassmannians

- One example of a common algebraic construction is the *Grassmanian*, defined as

$$\mathrm{Gr}(k, \mathbb{R}^n) := \{L \subseteq \mathbb{R}^n : L \text{ linear space of dim } k\}.$$

In terms of projective space, we have a natural correspondence

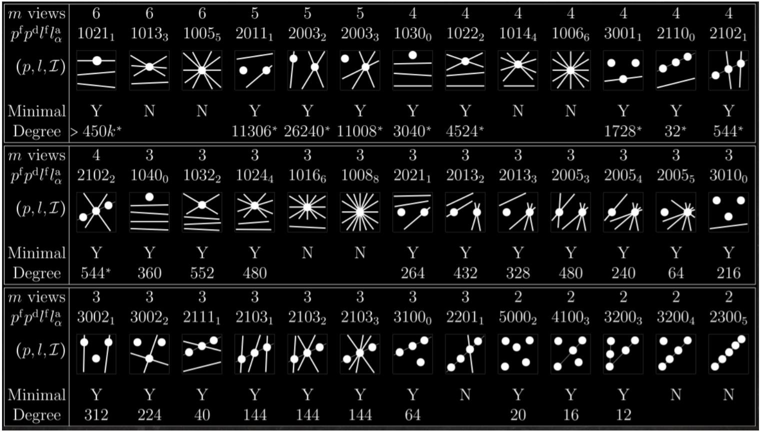
$$\mathrm{Gr}(k-1, \mathbb{P}^{n-1}_{\mathbb{R}}) \cong \mathrm{Gr}(k, \mathbb{R}^n).$$

- **Key idea:** These can be viewed as algebraic sets. This allows us to work efficiently with linear spaces, viewing them as points.

- For example, $\mathrm{Gr}(1, \mathbb{P}^3) \cong \mathrm{Gr}(2, \mathbb{R}^4)$ lie in $\mathbb{P}^5$ by the following *Plücker embedding* sending the space spanned by a, b, represented as the matrix

$$\begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \end{bmatrix}$$

to the vector of the six 2×2 minors of this matrix in $\mathbb{P}^5$. This is a *natural* bijection.



m views	6	6	6	5	5	5	4	4	4	4	4	4	4
$p^{\mathrm{f}}p^{\mathrm{d}}l^{\mathrm{f}}l^{\mathrm{a}}_{\alpha}$	1021_1	1013_3	1005_5	2011_1	2003_2	2003_3	1030_0	1022_2	1014_4	1006_6	3001_1	2110_0	2102_1
$(p, l, \mathcal{I})$													
Minimal	Y	N	N	Y	Y	Y	Y	Y	N	N	Y	Y	Y
Degree	$> 450k^*$			11306*	26240*	11008*	3040*	4524*			1728*	32*	544*

m views	4	3	3	3	3	3	3	3	3	3	3	3	3
$p^{\mathrm{f}}p^{\mathrm{d}}l^{\mathrm{f}}l^{\mathrm{a}}_{\alpha}$	2102_2	1040_0	1032_2	1024_4	1016_6	1008_8	2021_1	2013_2	2013_3	2005_3	2005_4	2005_5	3010_0
$(p, l, \mathcal{I})$													
Minimal	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y
Degree	544*	360	552	480			264	432	328	480	240	64	216

m views	3	3	3	3	3	3	3	3	2	2	2	2	2
$p^{\mathrm{f}}p^{\mathrm{d}}l^{\mathrm{f}}l^{\mathrm{a}}_{\alpha}$	3002_1	3002_2	2111_1	2103_1	2103_2	2103_3	3100_0	2201_1	5000_2	4100_3	3200_3	3200_4	2300_5
$(p, l, \mathcal{I})$													
Minimal	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	N	N
Degree	312	224	40	144	144	144	64		20	16	12		

Figure: Different algebraic reconstruction problems with finitely many solutions. Taken from *PLMP - Point-Line Minimal Problems in Complete Multi-View Visibility*, by Kohn et. al.

Modelling Cameras Mathematically

- A camera is modelled mathematically as a 3×4 matrix C of rank 3, projecting points (or lines) in $\mathbb{P}^3$ to the camera image plane $\mathbb{P}^2$. The *focus* (center) of the camera is equal to the kernel of C.

- Let $C^{(1)}, \dots, C^{(m)}$ be m camera matrices with different foci. We define the *joint camera map*

$$\Phi_C : \mathbb{P}^3 \dashrightarrow (\mathbb{P}^2)^m, X \mapsto (C^{(1)}X, \dots, C^{(m)}X),$$

sending a point in 3-space to its point images in the m cameras.

- Similarly, we have the alternative setting

$$\Upsilon_C : \mathrm{Gr}(1, \mathbb{P}^3) \dashrightarrow \mathrm{Gr}(1, \mathbb{P}^2)^m, L \mapsto (C^{(1)}L, \dots, C^{(m)}L),$$

sending a line in 3-space to its line images in the m cameras.

Modelling Cameras Mathematically

- A camera is modelled mathematically as a 3×4 matrix C of rank 3, projecting points (or lines) in $\mathbb{P}^3$ to the camera image plane $\mathbb{P}^2$. The *focus* (center) of the camera is equal to the kernel of C.

- Let $C^{(1)}, \ldots, C^{(m)}$ be m camera matrices with different foci. We define the *joint camera map*

$$\Phi_C : \mathbb{P}^3 \dashrightarrow (\mathbb{P}^2)^m, X \mapsto (C^{(1)}X, \ldots, C^{(m)}X),$$

sending a point in 3-space to its point images in the m cameras.

- Similarly, we have the alternative setting

$$\Upsilon_C : \mathrm{Gr}(1, \mathbb{P}^3) \dashrightarrow \mathrm{Gr}(1, \mathbb{P}^2)^m, L \mapsto (C^{(1)}L, \ldots, C^{(m)}L),$$

sending a line in 3-space to its line images in the m cameras.

Modelling Cameras Mathematically

- A camera is modelled mathematically as a 3×4 matrix C of rank 3, projecting points (or lines) in $\mathbb{P}^3$ to the camera image plane $\mathbb{P}^2$. The *focus* (center) of the camera is equal to the kernel of C.

- Let $C^{(1)}, \dots, C^{(m)}$ be m camera matrices with different foci. We define the *joint camera map*

$$\Phi_C : \mathbb{P}^3 \dashrightarrow (\mathbb{P}^2)^m, X \mapsto (C^{(1)}X, \dots, C^{(m)}X),$$

sending a point in 3-space to its point images in the m cameras.

- Similarly, we have the alternative setting

$$\Upsilon_C : \mathrm{Gr}(1, \mathbb{P}^3) \dashrightarrow \mathrm{Gr}(1, \mathbb{P}^2)^m, L \mapsto (C^{(1)}L, \dots, C^{(m)}L),$$

sending a line in 3-space to its line images in the m cameras.

Multi-View Varieties

- The reconstruction problem becomes easier when we know the camera specifications, i.e. we know the matrices $C^{(1)}, \ldots, C^{(m)}$. Reconstructing in this case becomes an exercise in linear algebra, after the image points (or lines) have been fitted into the multi-view variety.

- **Key idea:** It's difficult for a computer to work with the *joint images* $\Phi_C(\mathbb{P}^3)$ and $\Upsilon_C(\mathrm{Gr}(1, \mathbb{P}^3))$. Instead, we consider the *Zariski closure* of these images, written $\mathcal{M}_C$ in the point case and $\mathcal{L}_C$ in the line case. These are called *multi-view varieties.* The Zariski closure of a set A is the smallest algebraic set containing A.

- Any data comes with noise and we therefore fit them into the multi-view varieties by finding the closest point to it in Euclidean distance.

Multi-View Varieties

- The reconstruction problem becomes easier when we know the camera specifications, i.e. we know the matrices $C^{(1)}, \ldots, C^{(m)}$. Reconstructing in this case becomes an exercise in linear algebra, after the image points (or lines) have been fitted into the multi-view variety.

- **Key idea:** It's difficult for a computer to work with the *joint images* $\Phi_C(\mathbb{P}^3)$ and $\Upsilon_C(\mathrm{Gr}(1, \mathbb{P}^3))$. Instead, we consider the *Zariski closure* of these images, written $\mathcal{M}_C$ in the point case and $\mathcal{L}_C$ in the line case. These are called *multi-view varieties.* The Zariski closure of a set A is the smallest algebraic set containing A.

- Any data comes with noise and we therefore fit them into the multi-view varieties by finding the closest point to it in Euclidean distance.

Multi-View Varieties

- The reconstruction problem becomes easier when we know the camera specifications, i.e. we know the matrices $C^{(1)}, \ldots, C^{(m)}$. Reconstructing in this case becomes an exercise in linear algebra, after the image points (or lines) have been fitted into the multi-view variety.

- **Key idea:** It's difficult for a computer to work with the *joint images* $\Phi_C(\mathbb{P}^3)$ and $\Upsilon_C(\mathrm{Gr}(1, \mathbb{P}^3))$. Instead, we consider the *Zariski closure* of these images, written $\mathcal{M}_C$ in the point case and $\mathcal{L}_C$ in the line case. These are called *multi-view varieties.* The Zariski closure of a set A is the smallest algebraic set containing A.

- Any data comes with noise and we therefore fit them into the multi-view varieties by finding the closest point to it in Euclidean distance.

Multi-View Varieties

- In the *generic case* (the cameras are chosen randomly for instance,) we can write the ideals defining the multi-view varieties. Geometrically, they are explained as follows:

Theorem

In the point case, the variety consists of the image points whose back-projected lines in $\mathbb{P}^3$ meet in a point. In the line case, the variety conists of the image lines whose back-projected planes in $\mathbb{P}^3$ meet in a line.

- Practicioners are interested in the robustness of the two approaches. There is numerical evidence that the line approach is more stable. We want algebraic evidence for this.

- In this direction, the *Euclidean distance degree* of a variety $\mathcal{V} \subseteq \mathbb{R}^n$ is the number of complex critical points of

$$\|v - x\|_2^2,$$

over $v \in \mathcal{V}$ and generic $x \in \mathbb{R}^n$ (randomly chosen.)

Multi-View Varieties

- In the *generic case* (the cameras are chosen randomly for instance,) we can write the ideals defining the multi-view varieties. Geometrically, they are explained as follows:

Theorem

In the point case, the variety consists of the image points whose back-projected lines in $\mathbb{P}^3$ meet in a point. In the line case, the variety conists of the image lines whose back-projected planes in $\mathbb{P}^3$ meet in a line.

- Practicioners are interested in the robustness of the two approaches. There is numerical evidence that the line approach is more stable. We want algebraic evidence for this.

- In this direction, the *Euclidean distance degree* of a variety $\mathcal{V} \subseteq \mathbb{R}^n$ is the number of complex critical points of

$$\|v - x\|_2^2,$$

over $v \in \mathcal{V}$ and generic $x \in \mathbb{R}^n$ (randomly chosen.)

Multi-View Varieties

- In the *generic case* (the cameras are chosen randomly for instance,) we can write the ideals defining the multi-view varieties. Geometrically, they are explained as follows:

Theorem

In the point case, the variety consists of the image points whose back-projected lines in $\mathbb{P}^3$ meet in a point. In the line case, the variety conists of the image lines whose back-projected planes in $\mathbb{P}^3$ meet in a line.

- Practicioners are interested in the robustness of the two approaches. There is numerical evidence that the line approach is more stable. We want algebraic evidence for this.

- In this direction, the *Euclidean distance degree* of a variety $\mathcal{V} \subseteq \mathbb{R}^n$ is the number of complex critical points of

$$\|v - x\|_2^2,$$

over $v \in \mathcal{V}$ and generic $x \in \mathbb{R}^n$ (randomly chosen.)

Multi-View Varieties

• In the *generic case* (the cameras are chosen randomly for instance,) we can write the ideals defining the multi-view varieties. Geometrically, they are explained as follows:

Theorem

In the point case, the variety consists of the image points whose back-projected lines in $\mathbb{P}^3$ meet in a point. In the line case, the variety conists of the image lines whose back-projected planes in $\mathbb{P}^3$ meet in a line.

• Practicioners are interested in the robustness of the two approaches. There is numerical evidence that the line approach is more stable. We want algebraic evidence for this.

• In this direction, the *Euclidean distance degree* of a variety $\mathcal{V} \subseteq \mathbb{R}^n$ is the number of complex critical points of

$$\|v - x\|_2^2,$$

over $v \in \mathcal{V}$ and generic $x \in \mathbb{R}^n$ (randomly chosen.)

Thank you for listening!

- A few relevant sources:
  - S. Agarwal, Y. Furukawa et.al. *Reconstructing Rome.*
  - R. Szeliski. *Computer Vision, Algorithms and Applications.*
  - M. Trager, M. Hebert, J. Ponce. *The joint image handbook.*
  - J. Kileel. *Algebraic Geometry for Computer Vision.*
  - K. Kohn et. al. *PLMP - Point-Line Minimal Problems in Complete Multi-View Visibility.*