Document and Content Analysis

Summer 2009

Lecture 10
Camera-Captured Document Analysis

Thomas Breuel
Faisal Shafait

So far processed scanned documents only!

So far processed scanned documents only!

- Scanners produce:
  - Flat pages
  - Uniform illumination
  - Desktop scanners are cheap and produce nice images
  - High-End scanners can process large amounts of individual pages pretty quickly

Why would we need cameras to capture documents?

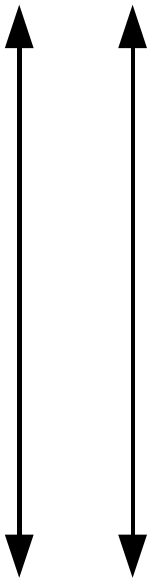
Why would we need cameras to capture documents?

- Cameras offer:
  - Fast and easy capture for daily use
  - Non-contact imaging
  - Non-destructive document capture

- Besides, most people already have cameras (digital cameras, cell-phone cameras)

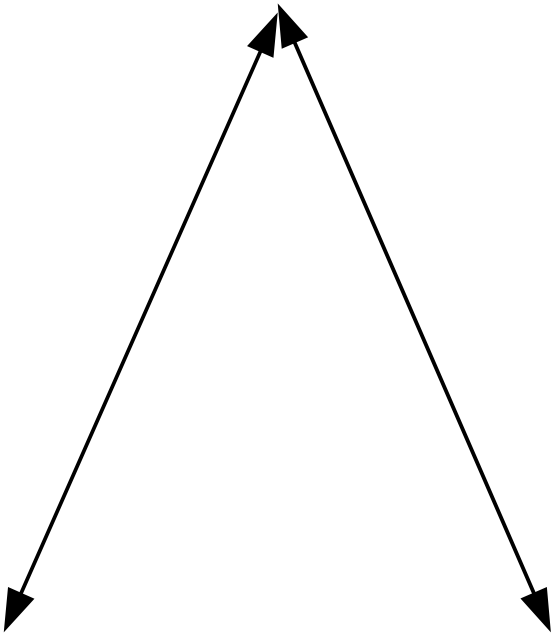
To analyze documents captured with a camera, we need to understand the world as a camera sees it!

Parallel Lines



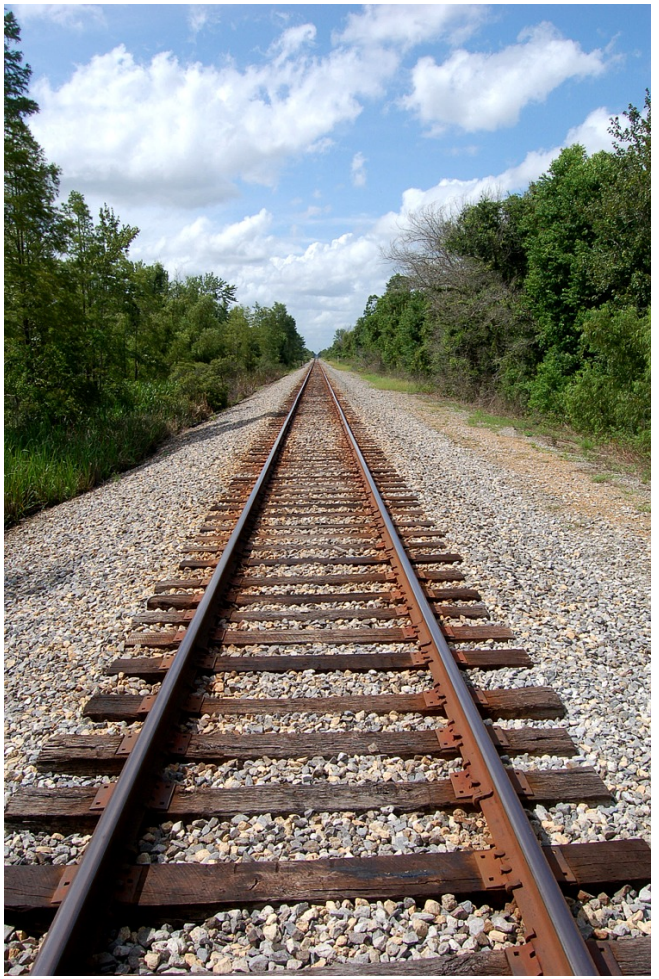
Where do these lines meet?

Intersecting Lines



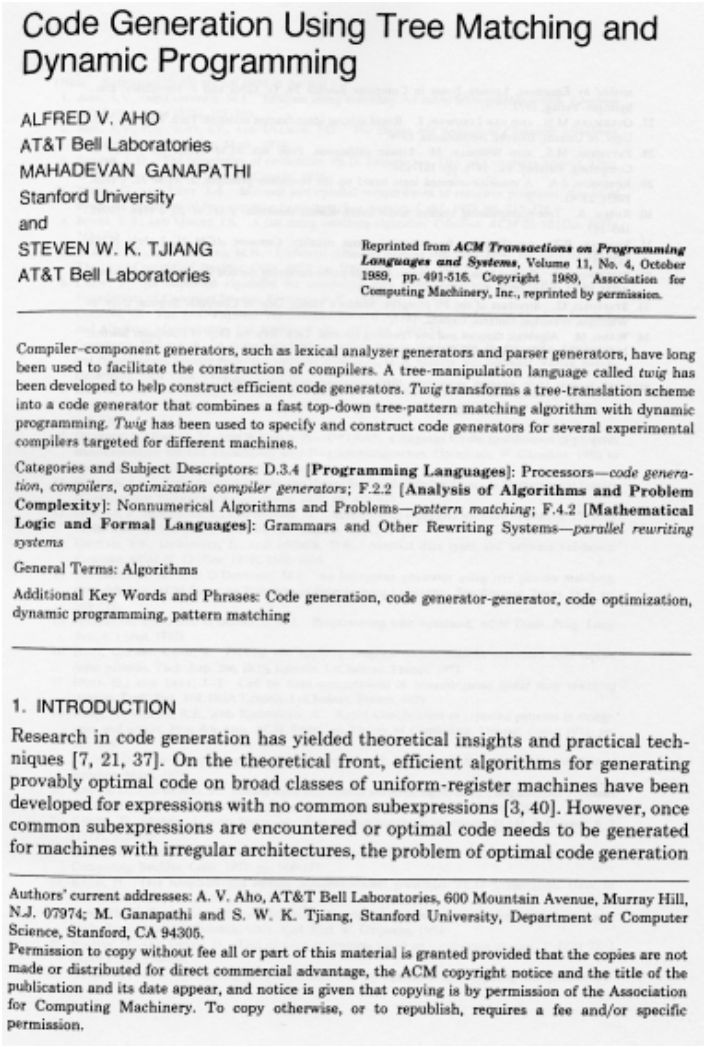
Where do these lines meet?

Perspective Distortion



Where do these lines meet?

- Cameras introduce several distortions in the image
- Camera-captured document analysis needs to handle these distortions
- Key distortions relavant to document captures are:
  - Perspective distortion
  - Uneven page surface



Code Generation Using Tree Matching and Dynamic Programming

ALFRED V. AHO
AT&T Bell Laboratories
MAHADEVAN GANAPATHI
Stanford University
and
STEVEN W. K. TJIANG
AT&T Bell Laboratories

Reprinted from *ACM Transactions on Programming Languages and Systems*, Volume 11, No. 4, October 1989, pp. 491-516. Copyright 1989, Association for Computing Machinery, Inc., reprinted by permission.

Compiler–component generators, such as lexical analyzer generators and parser generators, have long been used to facilitate the construction of compilers. A tree-manipulation language called *twig* has been developed to help construct efficient code generators. *Twig* transforms a tree-translation scheme into a code generator that combines a fast top-down tree-pattern matching algorithm with dynamic programming. *Twig* has been used to specify and construct code generators for several experimental compilers targeted for different machines.

Categories and Subject Descriptors: D.3.4 [**Programming Languages**]: Processors—*code generation, compilers, optimization compiler generators*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*pattern matching*; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems—*parallel rewriting systems*

General Terms: Algorithms

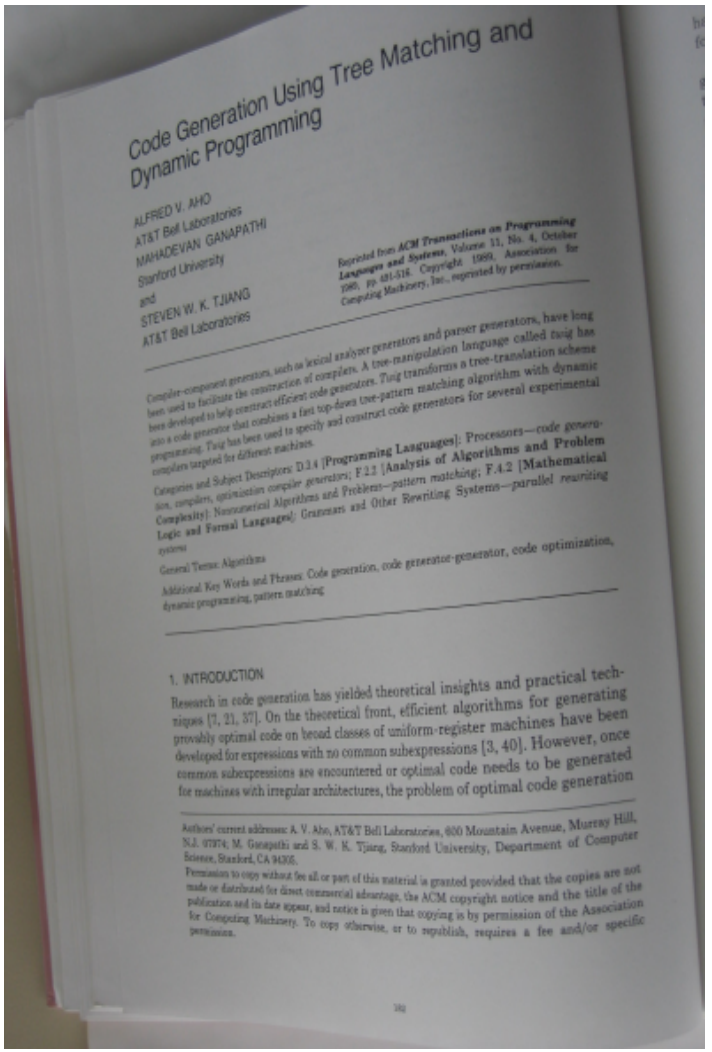
Additional Key Words and Phrases: Code generation, code generator-generator, code optimization, dynamic programming, pattern matching

1. INTRODUCTION

Research in code generation has yielded theoretical insights and practical techniques [7, 21, 37]. On the theoretical front, efficient algorithms for generating provably optimal code on broad classes of uniform-register machines have been developed for expressions with no common subexpressions [3, 40]. However, once common subexpressions are encountered or optimal code needs to be generated for machines with irregular architectures, the problem of optimal code generation

Authors' current addresses: A. V. Aho, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, N.J. 07974; M. Ganapathi and S. W. K. Tjiang, Stanford University, Department of Computer Science, Stanford, CA 94305.

(a) Scanned Book Page



(b) Camera-Captured Book Page

Traditional techniques for document analysis, including those we have studied so far assume a flat page.

What should we do now?

Traditional techniques for document analysis, including those we have studied so far assume a flat page.

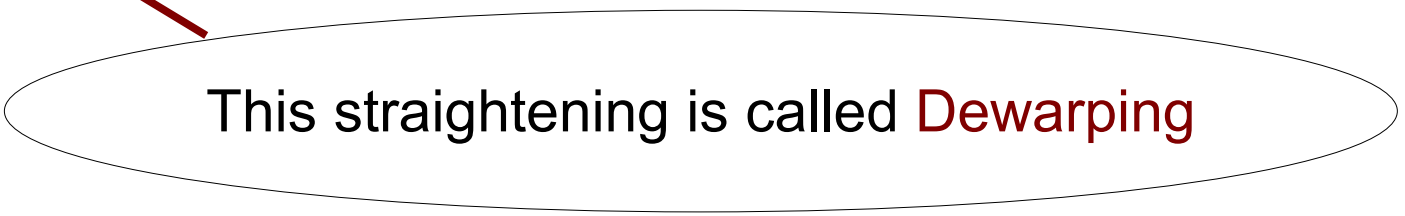
What should we do now?

- Develop new techniques for camera-captured document analysis
- "Straighten" camera-captured documents such that they look the same as scanned documents

Traditional techniques for document analysis, including those we have studied so far assume a flat page.

What should we do now?

- Develop new techniques for camera-captured document analysis
- "Straighten" camera-captured documents such that they look the same as scanned documents



Dewarping techniques can be divided into two broad categories:

- 3D-model based approaches
- Single camera based approaches

Dewarping

Dewarping techniques can be divided into two broad categories:

- 3D-model based approaches
  - Require specialized hardware
  - Careful camera calibration needed
  - Stereo-Cameras
  - Can do exact reconstruction (dewarping)
- Single camera based approaches
  - No special setup needed
  - Exact reconstruction not possible
  - Approximate solutions also quite acceptable

3D Model Based Approaches

- Techniques based on 3D computer vision:
  - Projective Geometry
  - Camera Models
  - Stereo Vision

Projective Geometry of 2D

- A vector of the projective space P^n has n+1 elements
- Fundamental property of vectors of a projective space: Scaling equivalence:

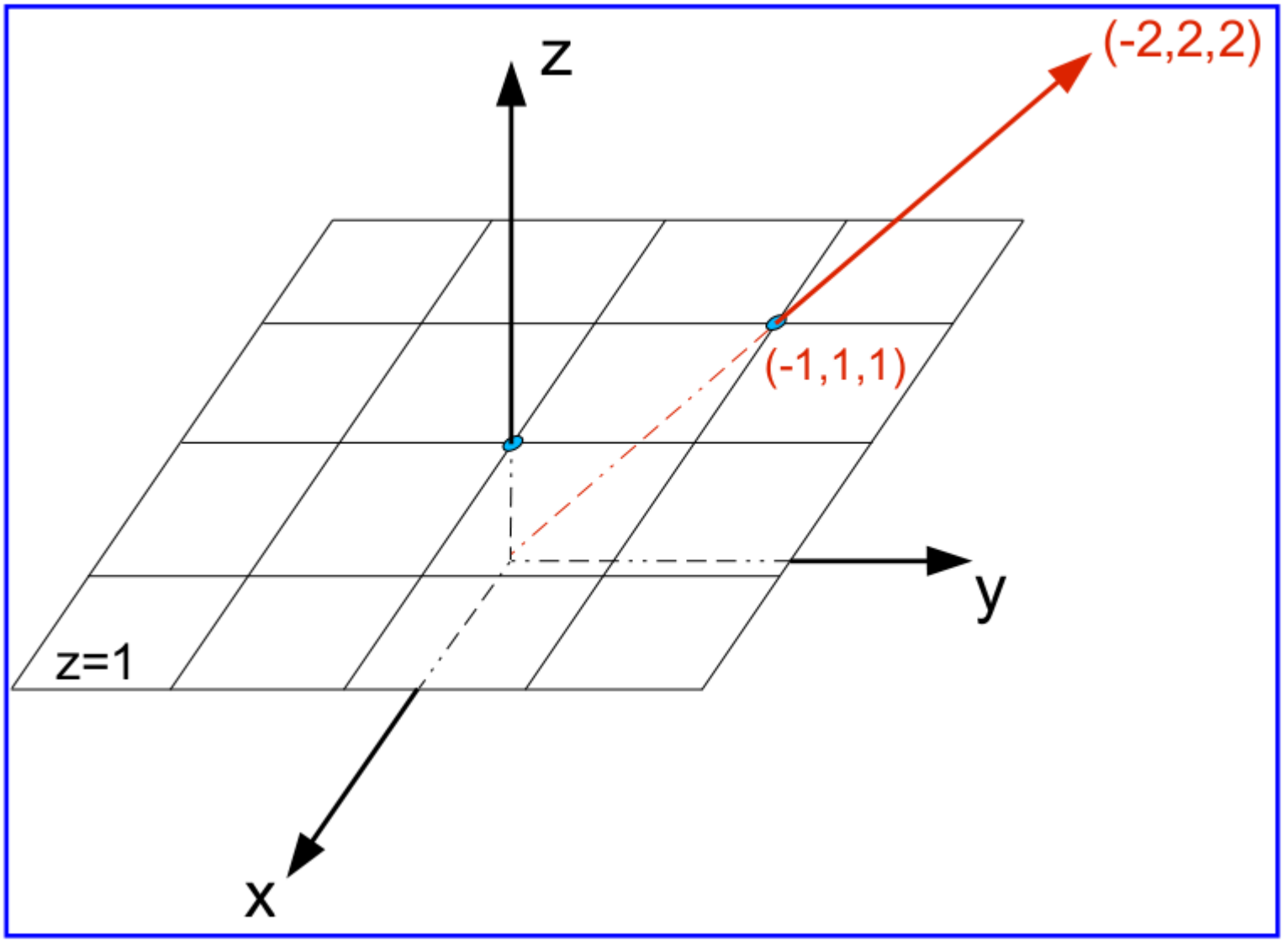
$$\left(x_1, x_2, \ldots, x_n, x_{n+1}\right)^\top \sim \lambda \left(x_1, x_2, \ldots, x_n, x_{n+1}\right)^\top ;\ \lambda \neq 0$$

- Two main reason for choosing the projective space for 3D computations:
  1. All linear transformations between points (rotation, translation, projection) can be written as matrix-vector multiplications
  2. Points at infinity can be represented in projective space

Homogeneous Coordinates

- *Homogeneous coordinates* make calculations possible in *projective space* just as *Cartesian coordinates* do in *Euclidean space*.
- Cartesian coordinates of a point in 2D Euclidean space are $(x,y)^T$
- Homogeneous coordinates of a point in 2D projective space are $(x,y,1)^T$

Illustration of Projective Space P^2



Projective Transformations

Translation

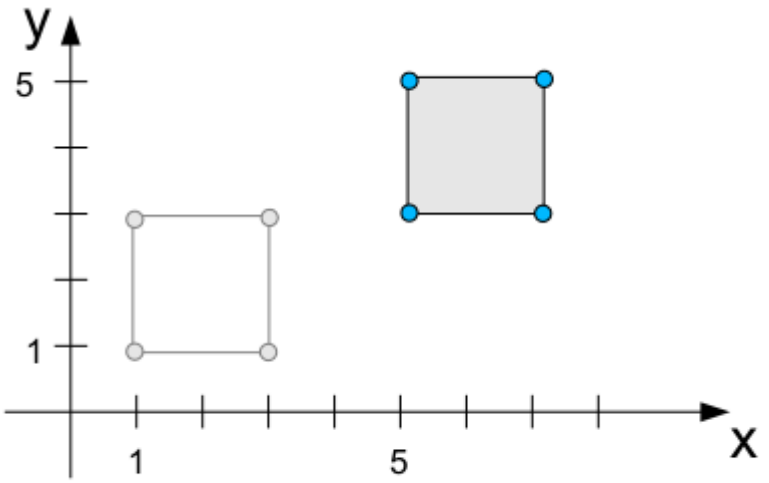
- Transformation matrix:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{I} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \left[= \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix} \right]$$

$\mathbf{t} = (t_x, t_y)^\top$: Translation vector

- Degrees of Freedom = 2

Example of Translation



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x+4 \\ y+2 \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Invariants: Length, Area, Angles

Projective Transformations

Euclidean Transformation

- Translation, Rotation
- Transformation matrix:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & t_x \\ \sin\varphi & \cos\varphi & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

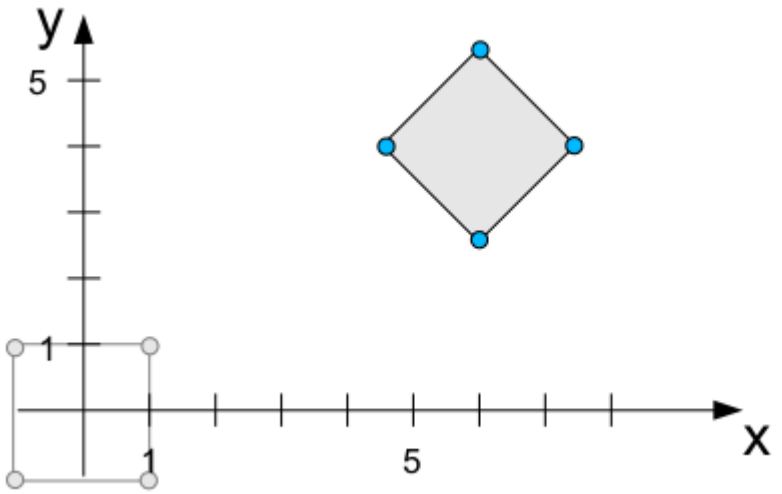
Upper left matrix $\mathtt{R}$:

- φ: Rotation angle
- $\mathtt{R}$: Rotation matrix, $\det(\mathtt{R}) = 1$

$\mathbf{t} = (t_x, t_y)^\top$: Translation vector

- Degrees of Freedom = 3

Example of Euclidean Transformation



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos(45°) & -\sin(45°) & 6 \\ \sin(45°) & \cos(45°) & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- First translation, then rotation

$$\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathtt{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Invariants: Length, Area, Angles

Projective Transformations

Similarity Transformation

- Translation, Rotation, Scaling
- Transformation matrix:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\varphi & -s\sin\varphi & t_x \\ s\sin\varphi & s\cos\varphi & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} s\mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

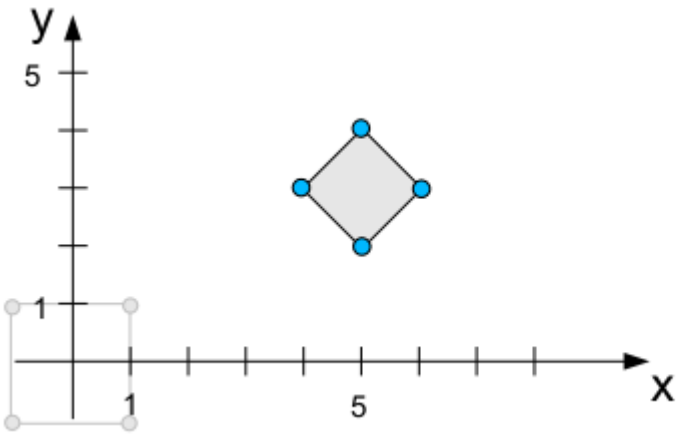
Upper left matrix $s\mathrm{R}$:

- s: Scaling factor
- φ: Rotation angle, i. e. R: Rotation matrix, $\det(\mathrm{R}) = 1$

$\mathbf{t} = (t_x, t_y)^\top$: Translation vector

- Degrees of Freedom = 4

Example of Similarity Transformation



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}\cos(45°) & -\frac{1}{\sqrt{2}}\sin(45°) & 5 \\ \frac{1}{\sqrt{2}}\sin(45°) & \frac{1}{\sqrt{2}}\cos(45°) & 3 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Rotation and scaling are commutative as

$$\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s\mathrm{R} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

- Invariants: Relative length, relative area, angles

Projective Transformations

Affine Transformation

- Translation, Rotation, non-isotropic Scaling
- Transformation matrix:

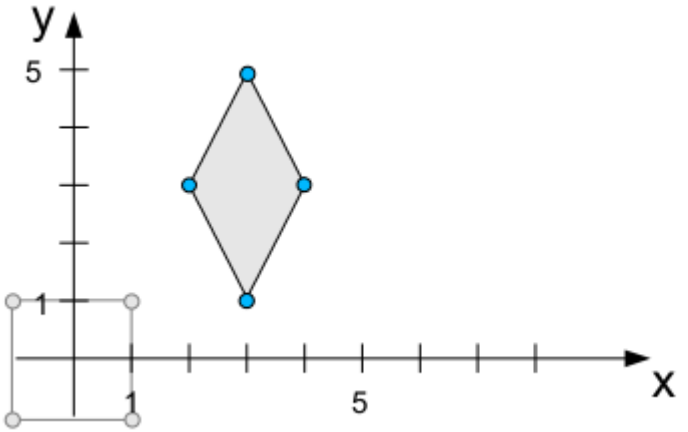
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Upper left matrix $\mathbf{A}$:

- Rotation and shearing with respect to two arbitrary orthogonal axes
- Must be non-singular

- Degrees of Freedom = 6

Example of Affine Transformation



$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}\cos(45°) & -\frac{1}{\sqrt{2}}\sin(45°) & 3 \\ \sqrt{2}\sin(45°) & \sqrt{2}\cos(45°) & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- *This* example of an affine transformation can be decomposed to

$$\begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(45°) & -\sin(45°) & 0 \\ \sin(45°) & \cos(45°) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Invariants: Parallelism of lines

Projective Transformations

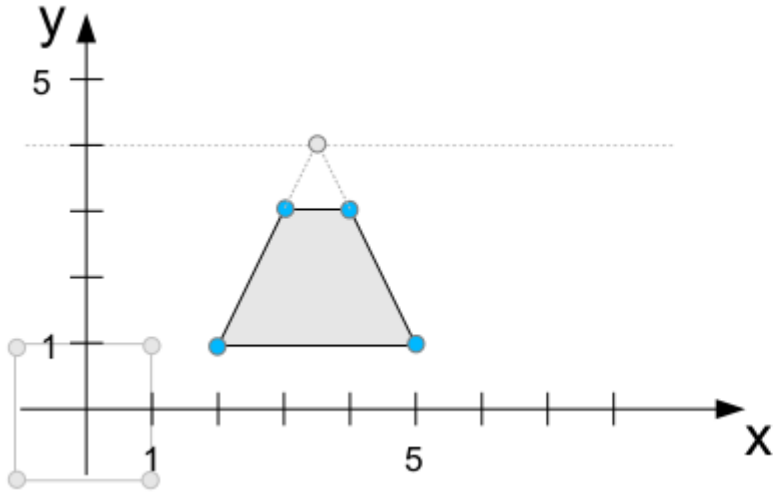
Projective Transformation

- Includes Affine Transformation
- Transformation matrix:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & v \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Degrees of Freedom = 8

Example of Projective Transformation



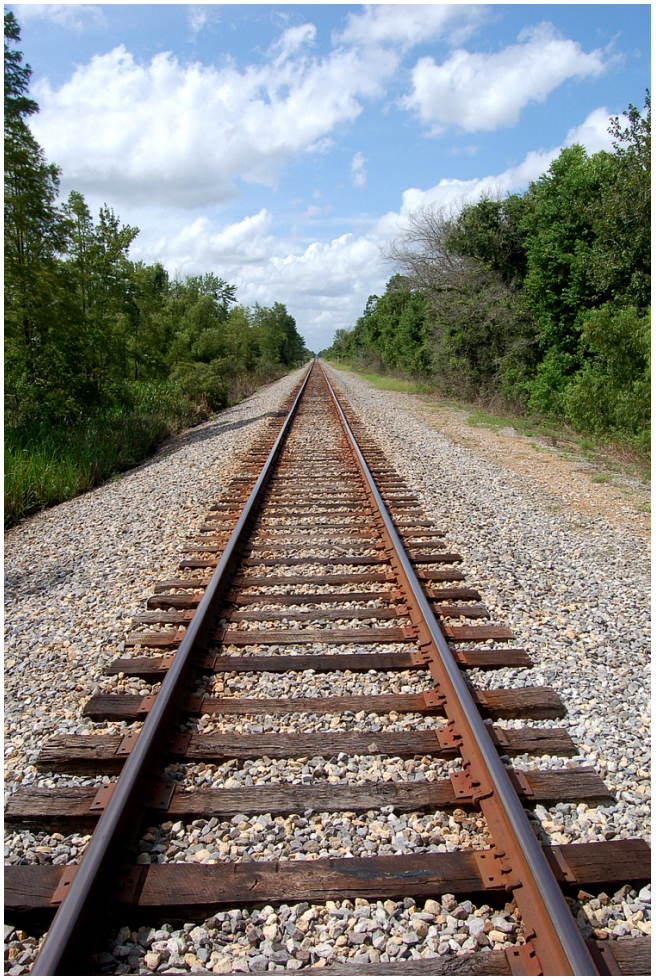
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1.5 & 3.5 & 7 \\ 0 & 4 & 5 \\ 0 & 1 & 2 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Point at infinity in y-direction gets projected to

$$\mathbf{p} = \begin{bmatrix} 1.5 & 3.5 & 7 \\ 0 & 4 & 5 \\ 0 & 1 & 2 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 3.5 \\ 4 \\ 1 \end{pmatrix}$$

- **p** is the vanishing point of the lines parallel to the y-axis

Perspective Distortion



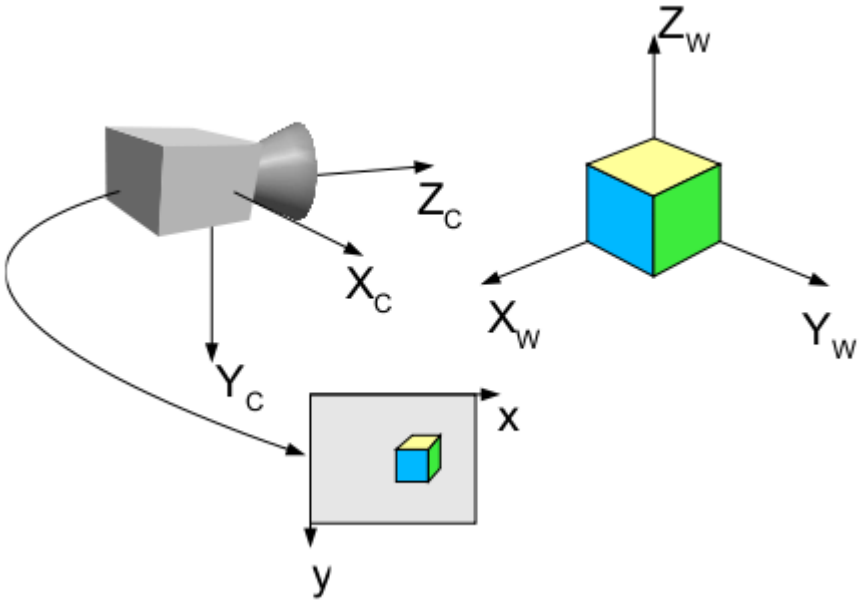
Where do these lines meet?

Projective Geometry can give us the answer!

3D Model Based Approaches

- Techniques based on 3D computer vision:
  - Projective Geometry
  - Camera Models
  - Stereo Vision

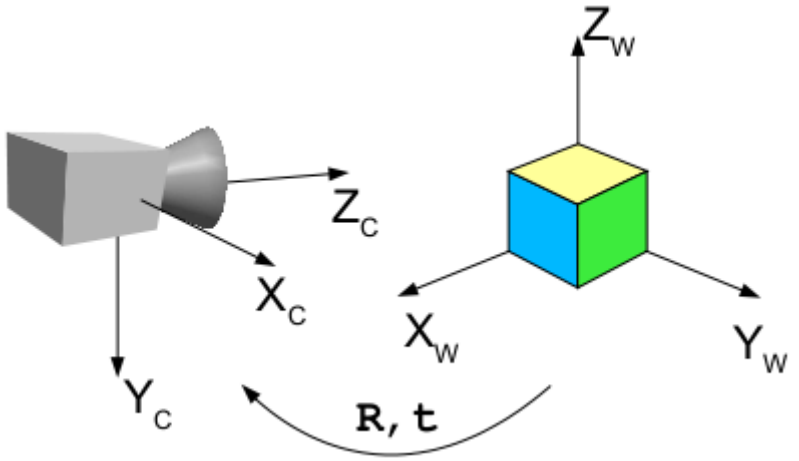
Camera Models



- Modeling the projection process consists of transforming point coordinates from

1. world coordinates X_{Ew} to camera coordinates X_{Ec}
2. camera coordinates X_{Ec} to image coordinates (x, y)

World to Camera Coordinates

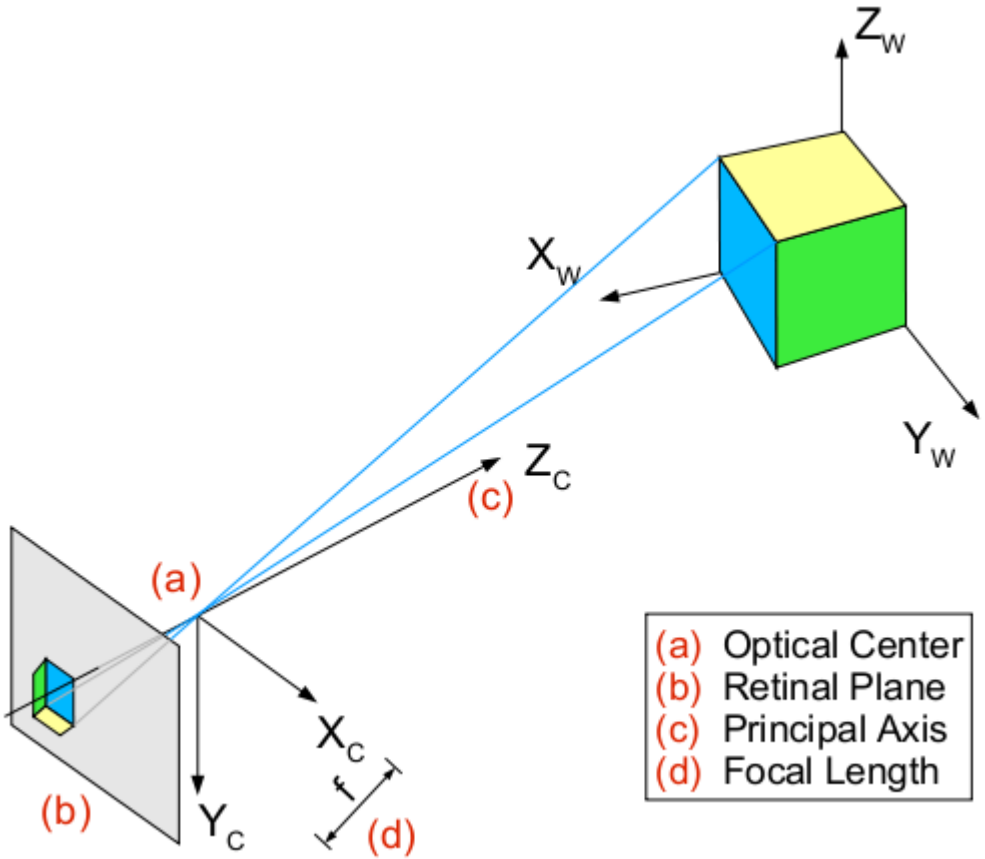


- The world and camera coordinate systems are related by rotation and translation
- Transformation of points is Euclidean in 3D

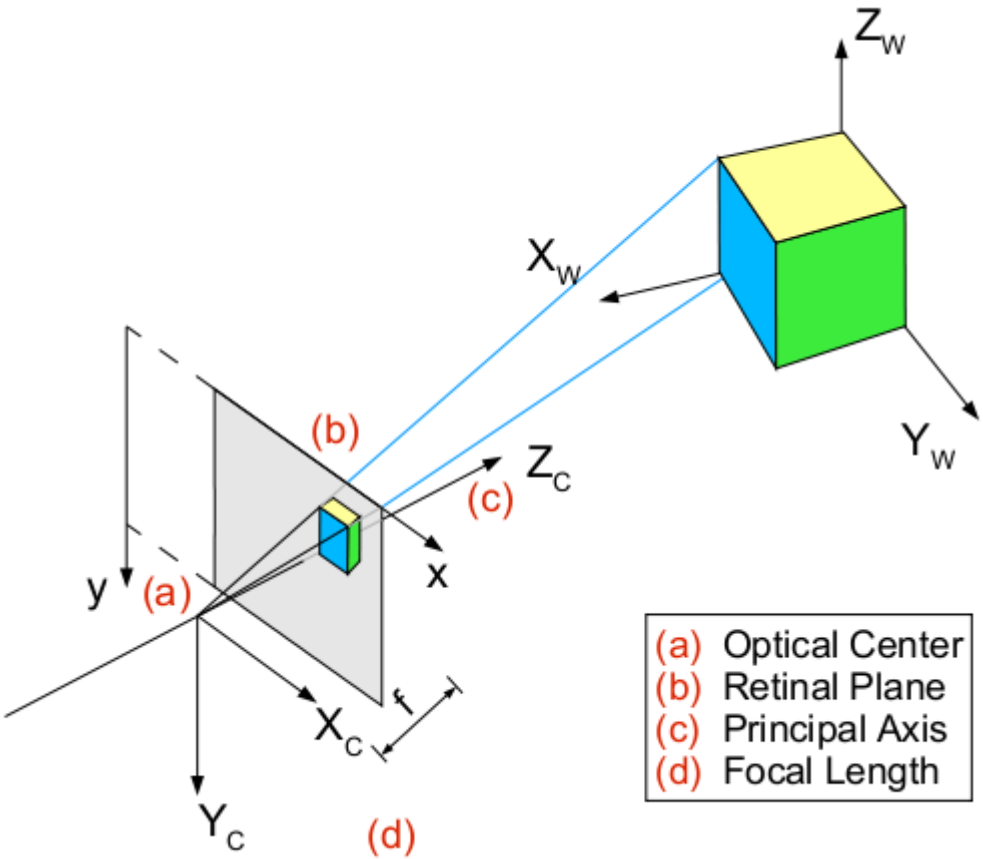
$$\mathbf{X}_c = \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X}_w$$

Camera to Image coordinates

- Camera models describe projection from camera coordinates to image coordinates mathematically
- Easiest model which is suitable for wide range of cameras is the **pinhole camera**



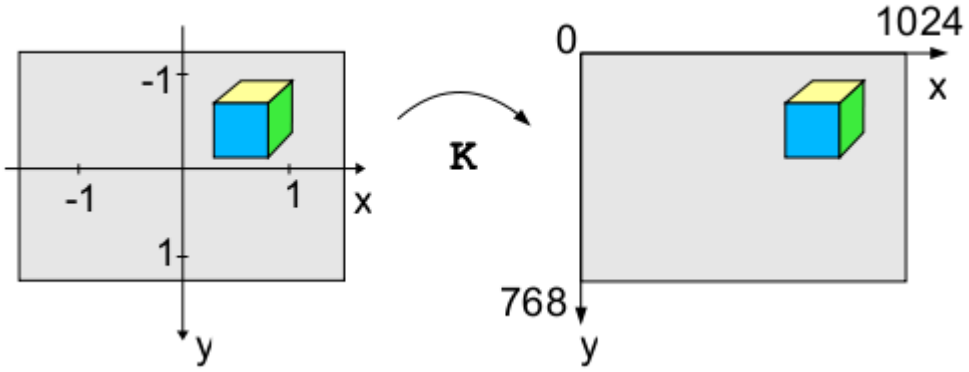
•Identical Model with image upright in front of the optical center



Pinhole Camera

- Pinhole camera with retinal plane in front of the optical center is easier to describe
- Transformation of camera coordinates to image coordinates:
  1. Canonic projection (independent of camera data)
  2. Transformation of projected coordinates to image coordinates
- By convention:
  - Principal axis points in Z_c-direction
  - Right-handed coordinate systems

Projected to pixel coordinates



- Intrinsic projection to pixel coordinates
- Pixel coordinates have origin in upper left corner
- Pixel dimensions scale x and y coordinates
- Used affine homography is called **intrinsic matrix** K

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \underbrace{\begin{bmatrix} \alpha_x & s & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}} \begin{pmatrix} x_p \\ y_p \\ w_p \end{pmatrix}$$

- Overall Projection:

$$\mathtt{P} = \mathtt{K} \cdot \mathtt{P}_0 \cdot \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

$$= \begin{bmatrix} k_x & s & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} k_x & s & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Matrix}} \cdot \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{\text{Extrinsic Matrix}}$$

- These matrices represent internal and external camera parameters

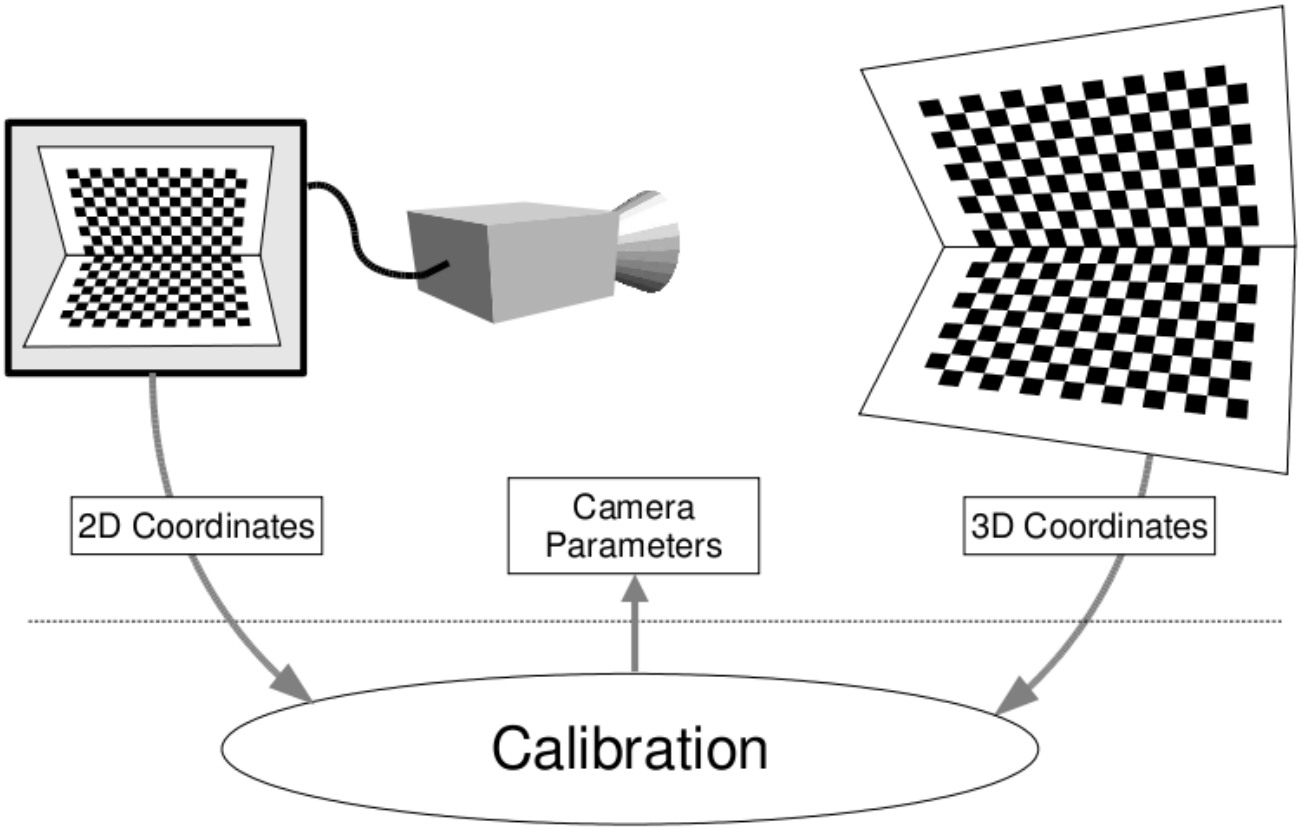
What is all this good for?

What is all this good for?

If we know the intrinsic and extrinsic parameters of a camera, we can use it to do **real-world** measurements (widely used in machine vision industry)

Camera Calibration

•Camera calibration: Estimating camera parameters given a set of image/world point correspondences
•A Calibration object i.e. a pattern with known world coordinates is used for this purpose



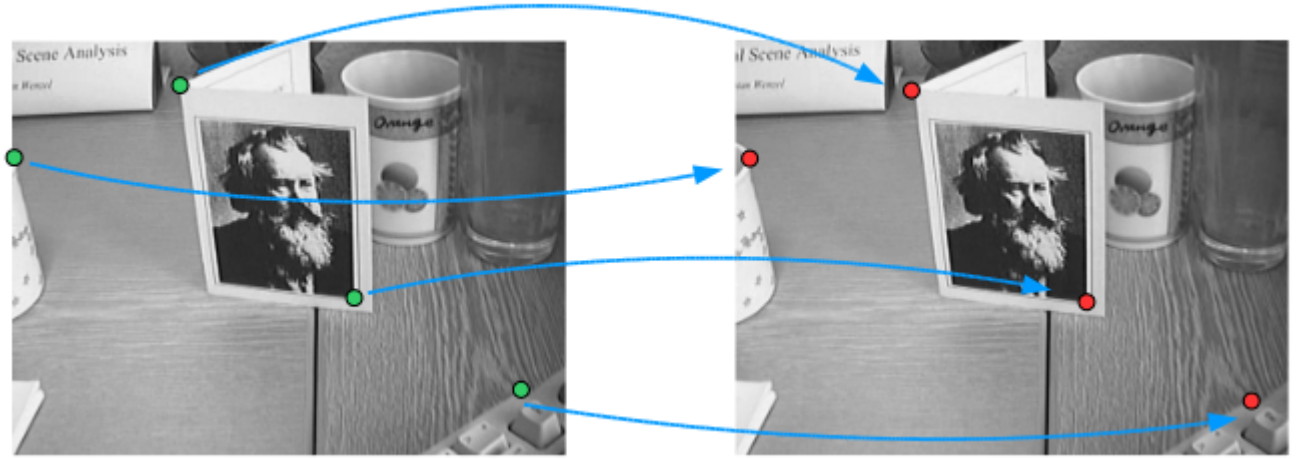
3D Model Based Approaches

- Techniques based on 3D computer vision:
  - Projective Geometry
  - Camera Models
  - Stereo Vision

Stereo Vision

- So far camera calibration is based on correspondences between the image (2D) and coordinates of a calibration object (2D or 3D)
- World coordinates of a calibration object, i.e. image content, have to be known
- Typically camera is looking at unknown image content
- Correspondences between image points can be established if two or more snapshots of the scene are available
- 3D reconstruction can be done if relative pose of the two cameras is known (**stereo cameras**)

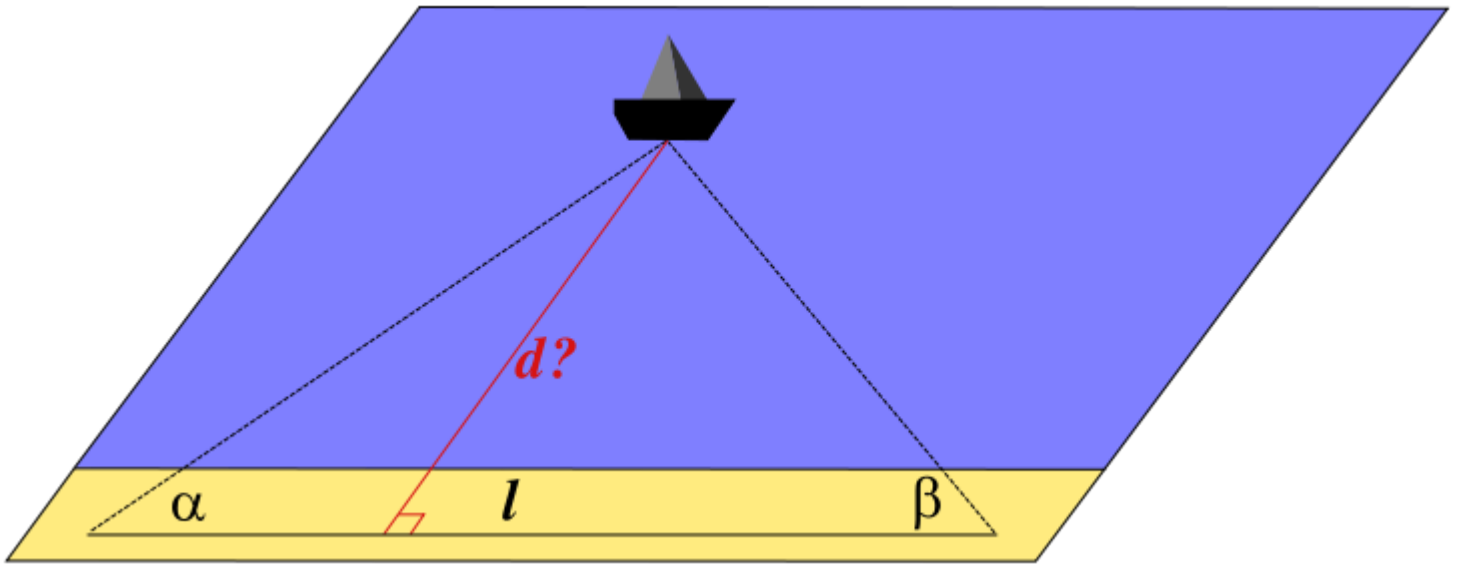
Triangulation



•After establishing correspondences between points, the distance of each point from the camera plane can be calculated using *Triangulation.* Hence the depth information that was lost during the imaging process can be recovered.

Triangulation

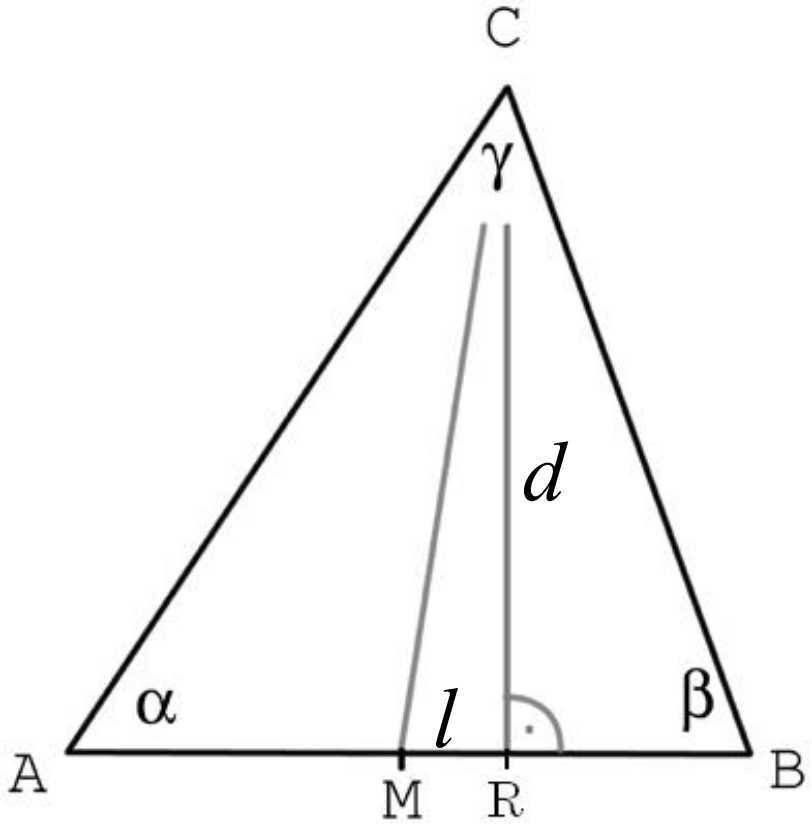
- *Triangulation* is the process of determining the location of a point by measuring angles to it from known points at either end of a fixed baseline, rather than measuring distances to the point directly



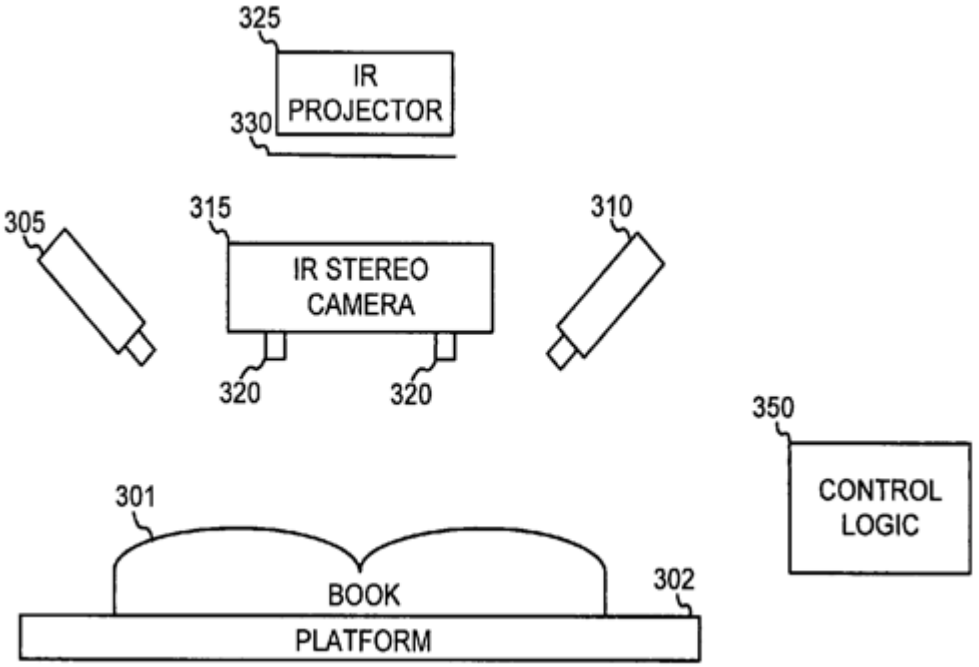
Triangulation

$$l = \frac{d}{\tan\alpha} + \frac{d}{\tan\beta}$$

$$d = l \,/\, \left(\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}\right)$$

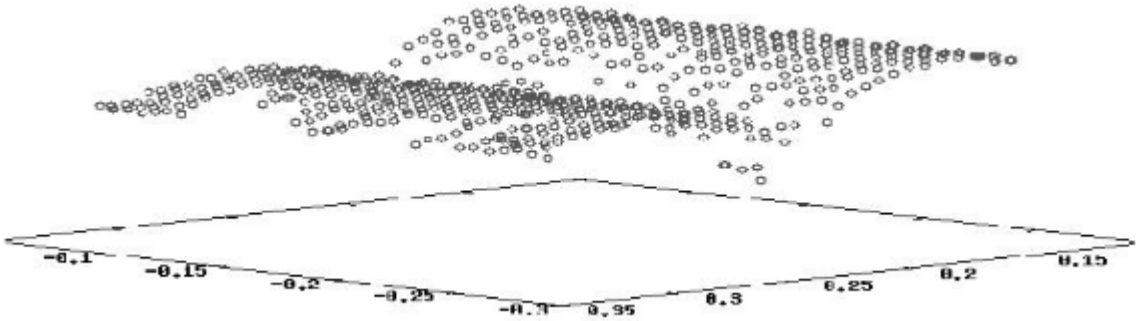


Google's Book Scanner



- Infrared light displays a calibration pattern on the book surface
- Infrared cameras reconstruct the 3D book surface
- Geometric transformations are applied to “flatten” the obtained surface.

3D Shape Reconstruction



since the Autobiography was published from the original manuscript, by Franklin's grandson. In the present volume it is printed from the genuine copy. Notes have been added to illustrate some parts, and the whole is divided into chapters, of suitable length, for the convenience of readers.

In writing the Continuation, it has been the author's aim to follow out the plan of the Autobiography, by confining himself strictly to a narrative of the principal events and incidents in Franklin's life, as far as these could be ascertained from his writings, his public acts, and the testimony of his contemporaries. In executing this task, he has had access to a large mass of papers left by Franklin, including his correspondence with many persons in various parts of the world, and also to copious materials, of much value, procured in England, France, and the United States, all of which were for several years in his possession, while he was preparing for the press a new and complete edition of Franklin's Works. As he has spared no pains in his researches, or in his endeavours to make their results useful to the public, he trusts that his efforts have not been wholly without success, and that they will be regarded as having added something to the tribute justly due to the memory of the philosopher, statesman, and philanthropist, whose fame is an honor not more to the land of his birth, than to the age in which he lived.

November, 1843.

Summary so far

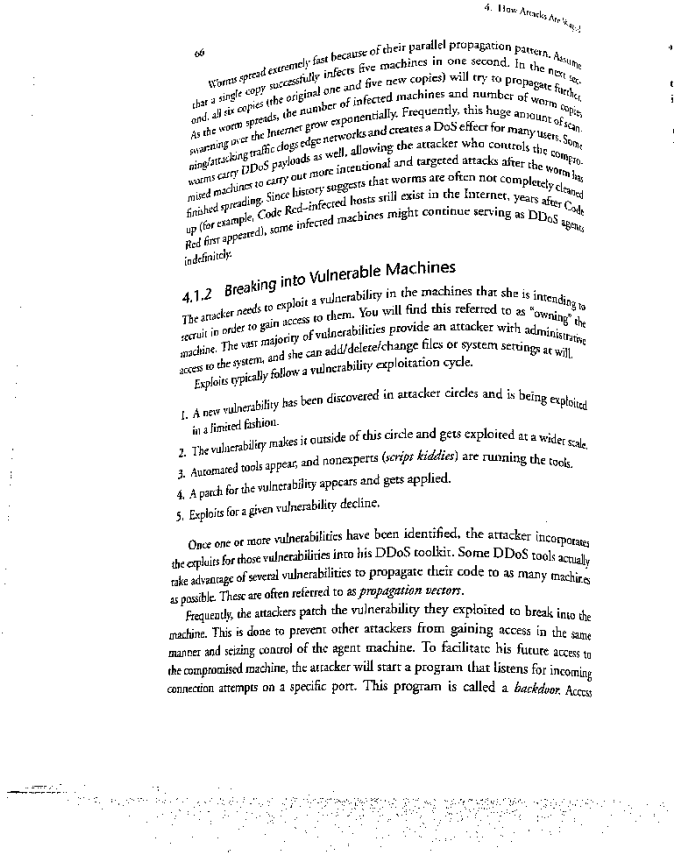
- Stereo 3D shape reconstruction and dewarping of books works but needs a lot of geometry (computer vision) and specialized hardware
- Not suitable for *home* users
- Capturing documents with a single hand-held camera is much easier but we can not get the 3D shape and hence perfect flattening is not possible

Monocular Dewarping

- Developing techniques for dewarping using a single camera is a hot topic of research.
- It works pretty well because document images are nicely structured with parallel text lines.
- Most of dewarping techniques simple focus on finding text lines in the camera-captured document and then straighten the text lines.

Monocular Dewarping

- Input image



Worms spread extremely fast because of their parallel propagation pattern. Assume that a single copy successfully infects five machines in one second. In the next second, all six copies (the original one and five new copies) will try to propagate further. As the worm spreads, the number of infected machines and number of worm copies swarming over the Internet grow exponentially. Frequently, this huge amount of scanning/attacking traffic clogs edge networks and creates a DoS effect for many users. Some worms carry DDoS payloads as well, allowing the attacker who controls the compromised machines to carry out more intentional and targeted attacks after the worm has finished spreading. Since history suggests that worms are often not completely cleaned up (for example, Code Red–infected hosts still exist in the Internet, years after Code Red first appeared), some infected machines might continue serving as DDoS agents indefinitely.

4.1.2 Breaking into Vulnerable Machines

The attacker needs to exploit a vulnerability in the machines that she is intending to recruit in order to gain access to them. You will find this referred to as "owning" the machine. The vast majority of vulnerabilities provide an attacker with administrative access to the system, and she can add/delete/change files or system settings at will.

Exploits typically follow a vulnerability exploitation cycle.

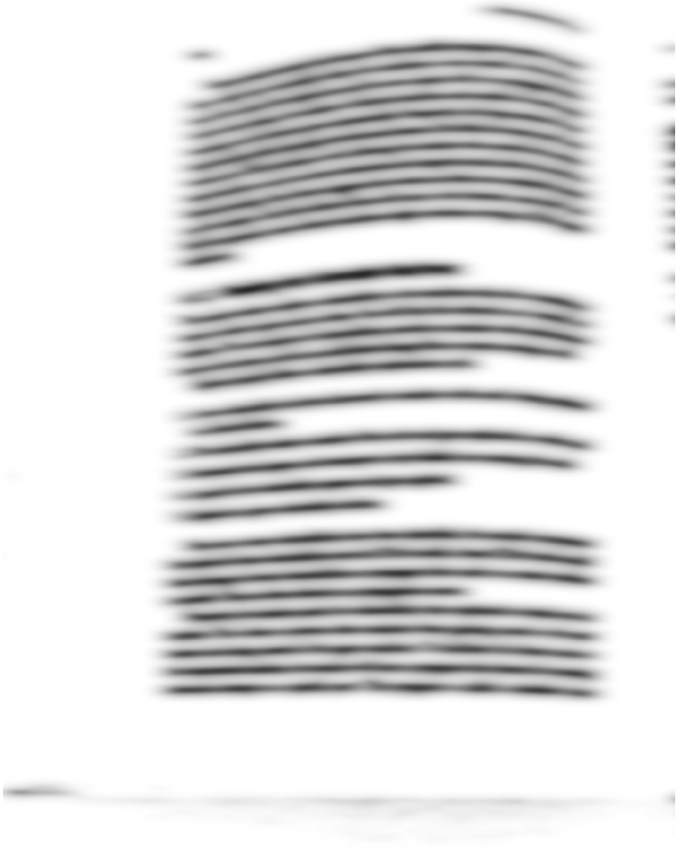
1. A new vulnerability has been discovered in attacker circles and is being exploited in a limited fashion.
2. The vulnerability makes it outside of this circle and gets exploited at a wider scale.
3. Automated tools appear, and nonexperts (*script kiddies*) are running the tools.
4. A patch for the vulnerability appears and gets applied.
5. Exploits for a given vulnerability decline.

Once one or more vulnerabilities have been identified, the attacker incorporates the exploits for those vulnerabilities into his DDoS toolkit. Some DDoS tools actually take advantage of several vulnerabilities to propagate their code to as many machines as possible. These are often referred to as *propagation vectors*.

Frequently, the attackers patch the vulnerability they exploited to break into the machine. This is done to prevent other attackers from gaining access in the same manner and seizing control of the agent machine. To facilitate his future access to the compromised machine, the attacker will start a program that listens for incoming connection attempts on a specific port. This program is called a *backdoor*. Access

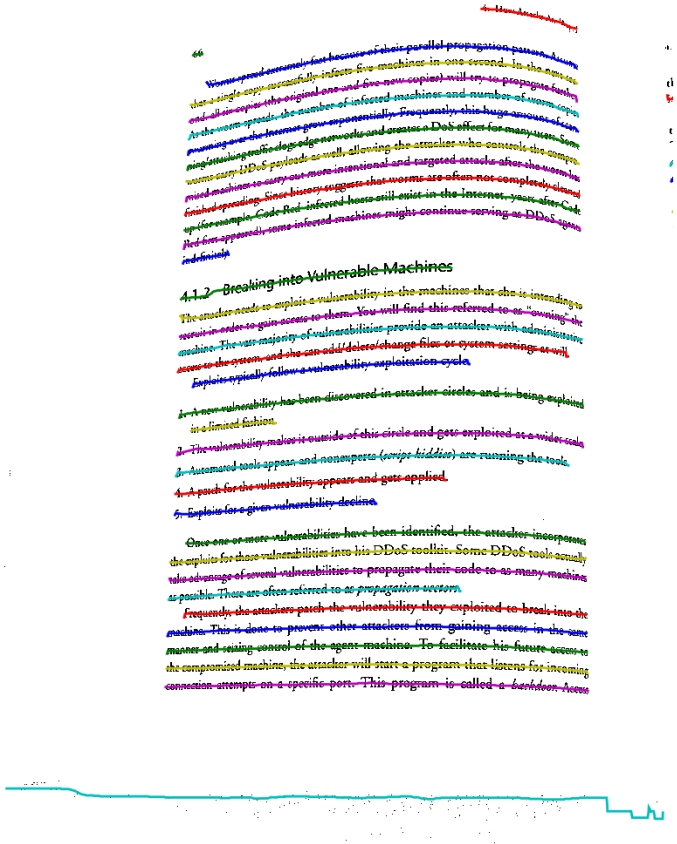
Monocular Dewarping

- Smoothed image (anisotropic Gaussian Smoothening)



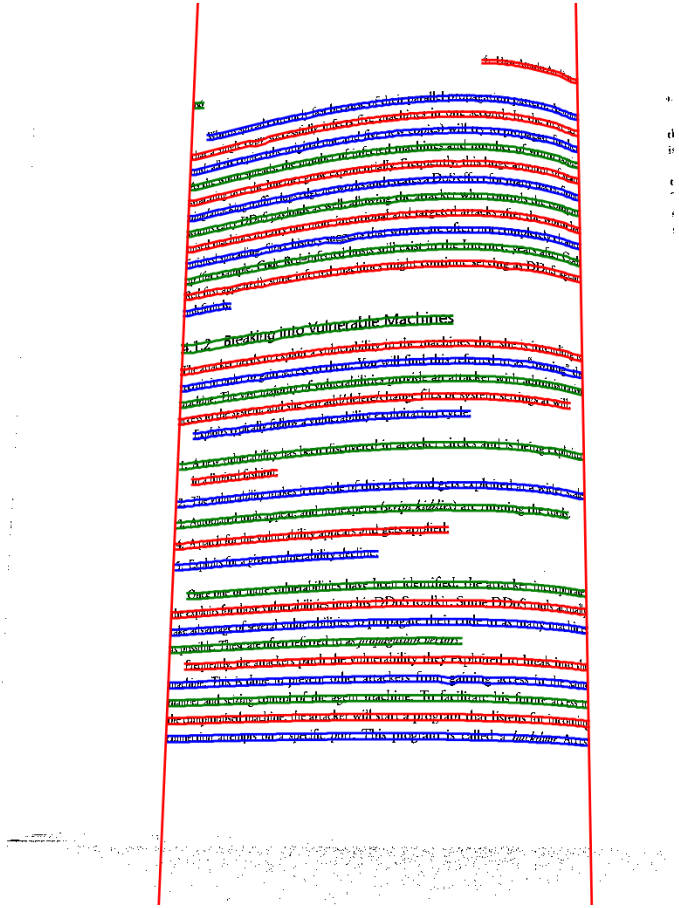
Monocular Dewarping

- Text-line detection



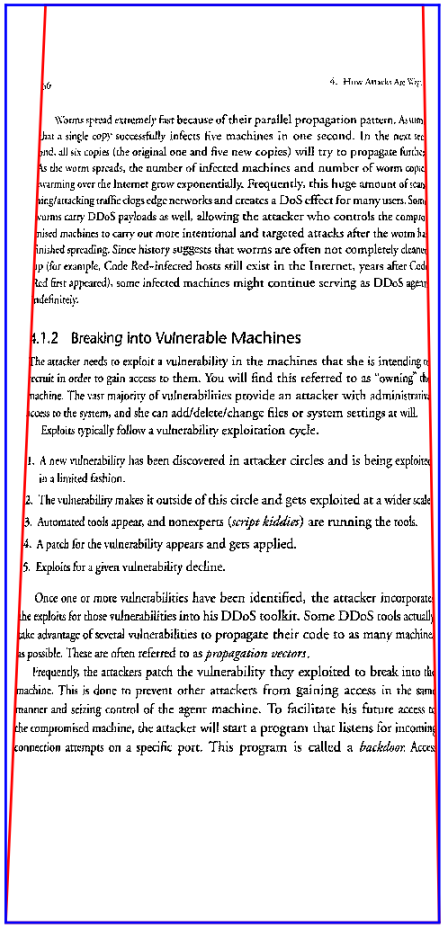
Monocular Dewarping

- Estimating baseline and x-line of detected text-lines



Monocular Dewarping

- Correcting geometric distortions: straightening individual curled textlines



96 6. How Attacks Are Wag

Worms spread extremely fast because of their parallel propagation pattern. Assum that a single copy successfully infects five machines in one second. In the next sec ond, all six copies (the original one and five new copies) will try to propagate further As the worm spreads, the number of infected machines and number of worm copie swarming over the Internet grow exponentially. Frequently, this huge amount of scan ning/attacking traffic clogs edge networks and creates a DoS effect for many users. Some worms carry DDoS payloads as well, allowing the attacker who controls the compro mised machines to carry out more intentional and targeted attacks after the worm ha finished spreading. Since history suggests that worms are often not completely cleaned up (for example, Code Red–infected hosts still exist in the Internet, years after Code Red first appeared), some infected machines might continue serving as DDoS agent indefinitely.

6.1.2 Breaking into Vulnerable Machines

The attacker needs to exploit a vulnerability in the machines that she is intending to recruit in order to gain access to them. You will find this referred to as "owning" the machine. The vast majority of vulnerabilities provide an attacker with administrative access to the system, and she can add/delete/change files or system settings at will.

Exploits typically follow a vulnerability exploitation cycle.

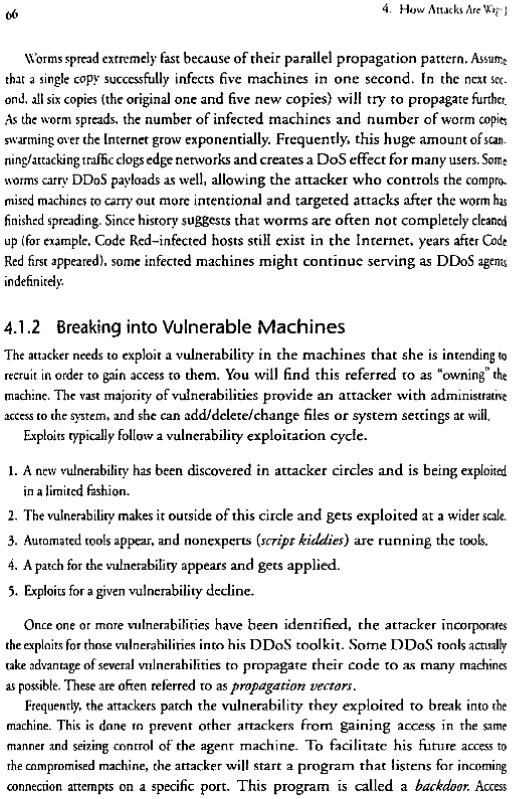
1. A new vulnerability has been discovered in attacker circles and is being exploited in a limited fashion.
2. The vulnerability makes it outside of this circle and gets exploited at a wider scale.
3. Automated tools appear, and nonexperts (*script kiddies*) are running the tools.
4. A patch for the vulnerability appears and gets applied.
5. Exploits for a given vulnerability decline.

Once one or more vulnerabilities have been identified, the attacker incorporates the exploits for those vulnerabilities into his DDoS toolkit. Some DDoS tools actually take advantage of several vulnerabilities to propagate their code to as many machines as possible. These are often referred to as *propagation vectors*.

Frequently, the attackers patch the vulnerability they exploited to break into the machine. This is done to prevent other attackers from gaining access in the same manner and seizing control of the agent machine. To facilitate his future access to the compromised machine, the attacker will start a program that listens for incoming connection attempts on a specific port. This program is called a *backdoor*. Access

Monocular Dewarping

- Correcting perspective distortion: four point homography algorithm



66 4. How Attacks Are Wag[illegible]

Worms spread extremely fast because of their parallel propagation pattern. Assume that a single copy successfully infects five machines in one second. In the next second, all six copies (the original one and five new copies) will try to propagate further. As the worm spreads, the number of infected machines and number of worm copies swarming over the Internet grow exponentially. Frequently, this huge amount of scanning/attacking traffic clogs edge networks and creates a DoS effect for many users. Some worms carry DDoS payloads as well, allowing the attacker who controls the compromised machines to carry out more intentional and targeted attacks after the worm has finished spreading. Since history suggests that worms are often not completely cleaned up (for example, Code Red–infected hosts still exist in the Internet, years after Code Red first appeared), some infected machines might continue serving as DDoS agents indefinitely.

4.1.2 Breaking into Vulnerable Machines

The attacker needs to exploit a vulnerability in the machines that she is intending to recruit in order to gain access to them. You will find this referred to as "owning" the machine. The vast majority of vulnerabilities provide an attacker with administrative access to the system, and she can add/delete/change files or system settings at will.

Exploits typically follow a vulnerability exploitation cycle.

1. A new vulnerability has been discovered in attacker circles and is being exploited in a limited fashion.
2. The vulnerability makes it outside of this circle and gets exploited at a wider scale.
3. Automated tools appear, and nonexperts (*script kiddies*) are running the tools.
4. A patch for the vulnerability appears and gets applied.
5. Exploits for a given vulnerability decline.

Once one or more vulnerabilities have been identified, the attacker incorporates the exploits for those vulnerabilities into his DDoS toolkit. Some DDoS tools actually take advantage of several vulnerabilities to propagate their code to as many machines as possible. These are often referred to as *propagation vectors*.

Frequently, the attackers patch the vulnerability they exploited to break into the machine. This is done to prevent other attackers from gaining access in the same manner and seizing control of the agent machine. To facilitate his future access to the compromised machine, the attacker will start a program that listens for incoming connection attempts on a specific port. This program is called a *backdoor*. Access

Stereo Dewarping for Home Users

- Decapod project at IUPR
- Goal: Create high-quality searchable PDFs from personal collections
- Scientific Challenges: Making Stereo Dewarping just work
  - Use consumer cameras
  - Make stereo calibration seamless for the users
- HiWi Jobs, Student Projects, Masters Theses available

Summary

- Why moving to cameras from scanners
- Basics of 3D dewarping
  - Projective Geometry
  - Camera Models
  - Stereo Vision
- Dewarping using a single camera