Linear Algebra for Computer Vision

Hunter Wills

April 17, 2014

1 Abstract

This paper is designed to educate undergraduate students with a strong background in upper level mathematics on the ways that linear algebra applies to modern computer vision. No background in computer vision is assumed, but readers should have a familiarity with topics such as linear transformations, least-squares and common matrix decompositions.

The paper begins with a discussion of the basics of computer vision to provide readers with a slightly better understanding of what the field is, before more in-depth procedures are covered. These topics are only supplemental to the readers understanding of linear algebra in computer vision and thus are not covered in great detail.

Next, the reader's understanding of linear algebra principles is extended through the discussion of some of the mathematics that commonly goes into computer vision. This includes expansions on least squares, the usefulness of singular value decompositions, and a comparison of when, and how, various matrix decompositions are used. In addition, a section discussing the benefits of efficiency that linear algebra provides to computer vision is included.

Finally, direct examples of the applications of linear algebra to computer vision are discussed. This includes structure from motion, eigenfaces, and convolution. Formulas for calculation and analysis of each of these areas are included when necessary. However, due to the problem specific variability of these applications, the theory of application is covered much more in depth. A section detailing the ways that linear algebra can be used to help the field of computer vision to continue to grow is also provided.

Should the reader find themselves in need of a visualization of what is occurring in any applications, an appendix of visualizations is provided at the end of this end of this document.

2 Basics of Computer Vision

Computer vision is a relatively new and vastly growing field born out of the study of artificial intelligence. Through the study of computer vision, professionals attempt to replicate in computers the ability to process and identify visuals in 2D and 3D pictures. While this ability comes quite easily to humans and animals alike, it is much more difficult to reproduce in computers. When the field of computer vision was first beginning, it was thought of as a relatively simple problem to solve in the field of artificial intelligence. In 1966, one MIT undergraduate student was given the task to "spend the summer linking a camera to a computer and getting the computer to describe what it saw." (Szeliski 2011). Much to the student and his professor's surprise, this proved a much more difficult undertaking than they had expected. One of the primary reasons that computer vision proves so complex is because "vision is an inverse problem, in which we seek to recover some unknowns given insufficient information to fully specify a solution." (Szeliski 2011) To solve such a problem, mathematical models must be used in order to gain the best understanding of the problem available.

The most common approaches to solving problems in computer vision are statistical and linear models. In statistical models, probability is used to determine what is most likely to be occurring in an image based on the known parameters of the problem and the proximity of

parameters to those of predetermined predictable outcomes. In models we solve with linear algebra, often multiple images at different but close times or different angles are taken. Properties identified in these images such as points, lines, and planes are stored as vectors in the camera matrix. Each of these vectors can be transformed via projective geometry so that the relative coordinate system of each image is aligned and the two images can be compared.

Through the use of linear algebra and these other mathematical models, the field of computer vision has expanded rapidly. Currently computer vision is being used in solving vital problems in a vast array of fields including medical imaging, surveillance, and face and object detection and identification. The techniques that linear algebra provides for solving complicated mathematical models is an essential tool in solving problems in each of these fields.

Because we examine projections of linear systems in computer vision, it is often easier to describe points in homogenous coordinates. Homogenous coordinates are essentially taking the idea of our standard Euclidean coordinate system and adding an extra dimension. One of the benefits of the use of homogenous coordinates is that they account for the concept of infinity, while infinity can only be approximated with limits in the Euclidean system. When using planes and lines to interpret curved surfaces or the intersection of two parallel lines, the concept of infinity becomes important. The benefit of homogenous coordinates can be imagined through picturing how as one looks at two parallel railway lines extending into the far off distance, the two lines appear to be getting closer and closer together and intersecting at some infinitely far away point. The point $(1, 2)$ in Cartesian coordinates becomes $(1, 2, 1)$ in homogenous coordinates and as this point moves out towards infinity we can continue to describe it in homogenous coordinates as $(1, 2, 0)$.

Oftentimes we will manipulate homogenous vectors through transformations. These transformations are linear transformations which usually act on the vector through multiplication with the transformation's matrix representation. Common transformations and the way they act on an image vector are displayed in the figure below:

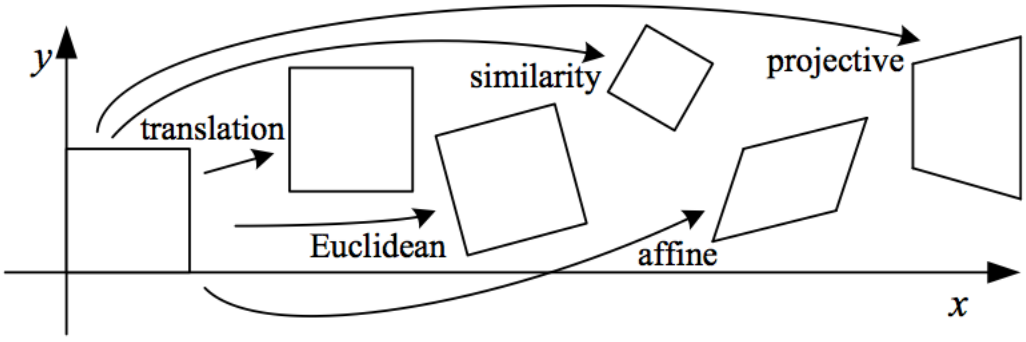

Figure 1: Examples of 2D transformations from Richard Szeliski

Aside from the actual image matrix, the most important matrix in computer vision is the camera matrix. The camera matrix or camera projection matrix is the representation of a

transformation of real world vectors into projected camera interpreted vectors. This matrix holds the information of all of the transformations that a world vector has undergone in order to become an image vector multiplied with a perspective-transforming matrix to account for each camera's focal length or other unique parameters. For example if a matrix with focal length f were to rotate a vector in homogenous coordinates, (x, y, z, w), by an angle θ about the z axis before projecting it onto the camera's image plane, the image vector, $\mathbf{c}$, would be given as:

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} cos\theta & sin\theta & 0 & 0 \\ -sin\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

To analyze collected data taken from two cameras looking at a system from two separate positions we utilize a different matrix known as the fundamental matrix. This matrix is useful in comparison and identification of points that are the same in the two pictures since the fundamental matrix satisfies the equipolar constraint that,

$$\mathbf{x}' F \mathbf{x} = \mathbf{0},$$

where $\mathbf{x}'$ and $\mathbf{x}$ are the two homogenous image vectors obtained from either camera and F is the fundamental matrix. There are numerous ways of finding the fundamental matrix, but they are all very complex. Usually you determine the fundamental matrix during calibration and measure rotation and translation that would align the two cameras. However, this matrix can also be found by physically selecting a point on the image of either camera and estimating the matrix that will satisfy the equipolar constraint.

3 Least Squares

When utilizing linear algebra to solve problems in computer vision, least squares is a commonly important tool. Computer vision often deals with attempting to interpret real world data such as the intensity and wavelength of light. Like most real world data, these values are error-prone causing one camera's interpretation of a scene to appear slightly different from another camera's interpretation. Such is the case for image matching.

In image matching, we begin with two images of a similar scene and attempt to match specific features in each scene to prove that the figures in each scene are the same. For example, take the case of face recognition. By looking at key features such as the nose, eyes, or jawline, we gain a set of vectors which can sufficiently describe each face. By comparing the features of the unknown face with the features of a face in our known database, we can find the most likely individual who the face belongs to.

In class, we learned the most general method of computing least squares, but in practice least squares is subdivided between computations such as total least squares, robust least squares, or non-linear least squares. Recall that the traditional least squares problem is attempting to minimize $\|A\mathbf{x} - \mathbf{b}\|$ for an $n \times m$ matrix, A and the vector $\mathbf{b}$.

The primary difference between the least squares method that we have used and total least squares is that instead of dealing with one particular variable of uncertainty, total

least squares assumes "uncertainty in all directions"(Szeliski 2011). What this means is that instead of treating the least squares problem as not only having error in measurements of $\mathbf{b}$, but also having error in the values of the data matrix, A. The problem in computer vision usually ignores the vector $\mathbf{b}$ and focuses simply on minimizing $\|A\mathbf{x}\|$ with the constraint that $\|\mathbf{x}\|= 1$. This minimum value can be solved for using a singular value decomposition since "the value of $\mathbf{x}$ that minimizes the constrained problem is the eigenvector associated with the smallest eigenvalue of A^*A."(Szeliski 2011)

Robust least squares, also known as least trimmed squares, is a way of solving the least squares problem when there are outliers in the data. The least trimmed squares method is a method which attempts to minimize only as subset of the residual so instead of solving $\|A\mathbf{x} - \mathbf{b}\|$ for values of $\mathbf{x} = [x_1, x_2, ...x_n]$ the problem is solved for only a subset $k < n$ of values contained in the $\mathbf{x}$ vector. Which values of $\mathbf{x}$ that are solved are found by computing the "ordered absolute residuals"(Doornick 2011) which means computing the residuals for every possible subset of k values from $\mathbf{x}$ and then selecting the solution which corresponds to the minimal residual.

Non-linear least squares is useful in solving some problems where the functions we are attempting to fit are not linear in the unknown parameters. This form of problem is "usually solved by iteratively relinearizing"(Szeliski 2011) about the estimate of the unknown parameter. In this case the solution to the overall non-linear least squares problem becomes solving $\|J\mathbf{\Delta p} - \mathbf{r}\|$ for each of the residuals, $\mathbf{r}$, to the solution of the linear least squares problem for the current estimate of the unknown parameters, $\mathbf{p}$, where J is the Jacobian matrix of partial derivatives of the functions we are fitting to the current estimate of the unknown parameters.

4 Singular Value Decomposition

In addition to its use in solving total least square algorithms, the singular value decomposition is an extremely useful tool across computer vision. Part of the reason for this is the singular value decomposition can be used to to show the strength of the relationship between data sets. A common tool for calculating these relationships is principal component analysis.

Principal component analysis takes a data matrix, A and forms a new matrix M of vectors ordered according to their variance. It is found through the formula:

$$M = AW$$

where W is a matrix composed of the eigenvectors of A^*A. A quick look at the singular value decomposition of A^*A shows that the matrix of "right singular vectors"(Szeliski 2011) of A, V, is the matrix of eigenvectors for A^*A. Thus a singular value decomposition is a very simple way of finding the principal component analysis.

$$\begin{aligned} A^*A &= (USV^*)^*USV^* \\ &= VSU^*USV^* \\ &= VS^2V^* \end{aligned}$$

The singular value decomposition is also a very handy tool for estimation of inverses of singular matrices. A matrix is nonsingular matrix if it has all nonzero singular values. In this case the inverse is very easy to calculate and can be found by simply performing the following calculation on the singular value decomposition

$$A = USV^*,\ A^{-1} = VS^{-1}U^*$$

where S^{-1} can be calculated easily by taking the inverse of each singular value. However, if the singular value of zero appears in the singular value decomposition of the matrix, then the matrix is singular and the inverse is approximated by

$$A^{-1} = VS_0^{-1}U^*$$

where S_0^{-1} has entries of the inverse of the singular value when the singular value is greater than some small threshold value and 0 otherwise.

5 Benefits of Various Decompositions

The singular value decomposition is the most common and useful decomposition in computer vision. The goal of computer vision is to explain the three dimensional world through two dimensional pictures. In the real world, most of these pictures will produce both square and non-square singular matrices and transformations. Inverting transformations from two dimensions to three dimensions will therefore not be completely accurate, but can be estimated quite well through singular value decomposition. Singular value decomposition will also allow us to establish a sense of order in objects and is therefore useful whenever attempting to compare.

While singular value decompositions are widespread in computer vision, at times it is more useful to use other decompositions such as Cholesky and QR decompositions. While a singular value decomposition breaks a single matrix down into three separate matrices with special properties, this decomposition can be rather space intensive. The Cholesky decomposition on the other hand breaks a matrix down into an upper triangular matrix and its adjoint and therefore only needs to store a single matrix in which half the entries are zero. This is quite beneficial when attempting to manipulate many large images with multiple parameters. Cholesky decompositions also remove the need for pivoting from traditional LU decompositions. The removal of the need for pivoting makes Cholesky decompositions less "sensitive to roundoff errors or reordering"(Szeliski 2011). In addition, Cholesky decompositions provide increased efficiency as will be discussed in a later section. Due to its ease of computation, speed of calculating solutions, low strain on memory, and the small benefit to efficiency it provides Cholesky decompositions are useful whenever speed of calculation is important. This is often the case when solving a least squares algorithm.

Cholesky decompositions require the matrix to be positive definite. When solving least squares problems, a positive definite matrix can be formed by converting the problem from solving $A\mathbf{x} = \mathbf{b}$ to solving $(A^*A)\mathbf{x} = A^*\mathbf{b}$. Covariance matrices, used in comparing relationships between data sets, are "positive definite unless there exists a linear combination of its variables which has zero variance."(O'Hagan, Buck, Daneshkhah, Eiser, Garthwaite, Jenkinson, Oakley, and Rakow 2006) When analyzing data sets in computer vision, zero

variance results are restricted from the calculation of the covariance matrix since such data would add no new beneficial information. Therefore, Cholesky decompositions can be used quite often in computer vision. However, when they cannot be used it is often beneficial to use a QR decomposition. The QR decomposition will allow us to compute projections more easily while preserving accuracy from the image data. This decomposition is often used when performing calculations going from a higher dimensional space to a lower one.

6 Efficiency

A great benefit that linear algebra provides to computer vision is that linear algebra algorithms usually provide improved efficiency over other algorithms. The Cholesky decomposition in particular is an extremely efficient decomposition to perform. Not only is the Cholesky decomposition very good for memory efficiency by storing only a single matrix composed half of 0's, this decomposition is extremely time efficient as it is twice as fast to compute as an LU decomposition. Since the Cholesky decomposition is comprised of a lower triangular matrix and an upper triangular matrix, solving for solutions to linear systems with it are extremely easy to compute through back solving.

Statistical methods of computer vision require the storage of a large amount of data and computations performed on each of the stored entries. While these methods may produce a more accurate and complete solution to computer vision problems by giving the "best" representation of the real world, it is usually sufficient to gain a "good" approximation of the real world situation. Speed and reduced memory consumption are usually much more important. Approximating the non-linear real world through models in linear algebra provides these more efficient predictions and often still produces a model which is quite close to the real world depiction.

7 Structure From Motion

Imagine a picture of the Eiffel Tower. The photograph displays a tall metal structure with a relative width and height which are easily identifiable, but in the process of being made into a two dimensional photo we have lost important information about the tower. We can no longer obtain depth information about the tower; it is no longer clear how far one girder is in front of another. A key goal in computer vision is attempting to regain this lost information by inverting the transformation from real three dimensional space to real two dimensional space.

The process of redetermining this lost information is known as structure from motion. In structure from motion, either multiple images of a stationary structure are taken from differing positions or a stationary camera is used to take multiple images of a structure undergoing motion. Here, the first case will be examined. A singular value decomposition is used to estimate the movement necessary to get from the location where one picture was taken to the location where the second picture was taken. Each camera utilizes its own coordinate system for which it is the origin. To compare the relative locations of the two, the single stationary coordinate system of the Eiffel Tower must be used. As discussed

earlier, the process of capturing the image places the image vectors from three dimensional space under multiple translations and rotations and finally a projection onto the image plane. However this transformation is not invertible so returning to three dimensional space proves difficult.

The reason singular value decomposition helps in structure from motion is that it can give us an approximate inverse of the transformation. From the image, a singular value decomposition is used on the known projection matrix of the camera to form an approximate inverse for estimating the three dimensional interpretation of the image. Recall that the process for forming a pseudo-inverse from a singular value decomposition of $A = USV^*$ is calculating $A^{-1} = VS_0^{-1}U^*$ where S_0^{-1} is comprised of the inverse of the corresponding singular value when the singular value is greater than some arbitrary threshold value and 0 otherwise. Once the approximate 3D image is found, the 3D image is transformed according to the other camera's projection matrix and this image is compared with the image taken by the other camera. A multitude of clearly identifiable points in each image is marked and the quality of the inverse transformation is measured by how well they minimize the distance between locations of marked points. If the locations are not within reasonable bounds, additional transformations are estimated into the calculation of the projection matrix and the process is repeated with a new singular value decomposition.

Calculating the generalized inverse and matching feature points are sources for major error in forming structure from motion. The error in calculation of the generalized inverse is reduced when the number of photographs used in forming the structure is increased, but this also increases the error in matching feature points. When possible, computer vision professions will often mark feature points manually to avoid error in the computer's attempt to match them.

8 Eigenfaces

Another intensive implementation of linear algebra in computer vision is through face detection and recognition. One of the common ways of approaching face recognition is through the eigenfaces method. Given an n by m image of a face, one can treat the image as an n by m dimensional matrix or, as is more useful for eigenface recognition, the image can be treated as an nm dimensional vector.

$$\begin{bmatrix} i_{1,1} & \dots & i_{n,1} \\ \dots & \dots & \dots \\ i_{1,m} & \dots & i_{n,m} \end{bmatrix} \rightarrow \begin{bmatrix} i_{1,1} \\ \dots \\ i_{n,1} \\ i_{1,2} \\ \dots \\ i_{n,m} \end{bmatrix} = \mathbf{i}$$

Each of these vectors can be packed together into a new matrix called the image matrix. A singular value decomposition can be useful here as a principal component analysis is performed on the image matrix in order to produce "our original data in terms of the eigenvectors"(Smith 2002) of the covariance matrix. Using a set of known images of a particular face the covariance matrix is found. Using the initial set of face image vectors as

$\mathbf{i_1}, \mathbf{i_2}, ...$ and the average of these vectors to be $\mathbf{a} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{i_j}$, we can simplify our calculations significantly by finding a smaller set of the most significant eigenvectors through the following singular value decomposition:

$$I = [\mathbf{i_1} - \mathbf{a}|\mathbf{i_2} - \mathbf{a}|...|\mathbf{i_m} - \mathbf{a}]$$
$$I^*I = VS^2V^* = [\mathbf{e_1}|\mathbf{e_2}|...|\mathbf{e_m}] \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & ... & 0 \\ 0 & 0 & \lambda_m \end{bmatrix} \mathbf{V}^*$$

Thus our calculations are simplified from the enormous calculation of nm by nm to simply m by m matricies. Each eigenvector is "a linear combination of the original face images,"(Turk, Pentland 1991) and is referred to as an eigenface. The eigenfaces with the largest corresponding eigenvalues are used to describe the set of face images. Here, the eigenfaces are denoted as $\mathbf{e_i}$ and the corresponding eigenvalues are λ_i. When we select the most sufficient eigenfaces through their eigenvalues we reduce the number of vectors from m to a smaller number on the order of 20.

Next, a new face image is introduced and compared to see which person from the original image set the face belongs to. For this comparison, each face image is broken down into its "eigenface components"(Turk, Pentland 1991) through the calculation:

$$w_i = \mathbf{e_i^*}(\mathbf{f} - \mathbf{a})$$

where w_i is the i^{th} eigenface component, $\mathbf{e_i^*}$ is the adjoint of the i^{th} eigenface, $\mathbf{f}$ is the current image vector and $\mathbf{a}$ is the average image vector. These components make up the $\mathbf{w}$ vector that approximates how much each eigenface contributes to making the formation of the face image. The square of the inner product is taken between between the new face's component vector and each of the known faces component vectors in what is known as computing the Euclidian distance. The face to which the new face belongs (if it indeed belongs to any of the faces) is the one which "minimizes the Euclidian distance"(Turk, Pentland 1991).

9 Convolution

Convolution, also known as neighborhood filtering, is an application of linear algebra in computer vision that is relatively simple to perform and is popularly used. Convolution is the process in which, given a large image matrix, a much smaller convolution matrix is used to manipulate only portions of the image matrix at a time. This is a linear filtering that can be used to perform such filtering objectives as "add soft blur, sharpen details, accentuate edges, or removing noise."(Szeliski 2011)

When performing a convolution, each pixel in the image matrix is replaced with a linear combination of its surrounding pixels. Which linear combination of the neighborhood of pixels is determined by the convolution kernel. The convolution kernel is a small square matrix, usually 3 by 3 or 5 by 5, which is used to form the linear combination of neighborhood pixels according to the following formula (for an n by n kernel):

$$b = \sum_{i=0}^{n}\sum_{j=0}^{n} A_{i,j}K_{i,j}$$

where A is the n by n matrix representing the neighborhood of pixels around the pixel we are convoluting, K is the convolution kernel, and b is the new value of the pixel.

Discovering which convolution kernel is very complicated. It involves discovery of a linear transformation which will move calculations from Euclidean space to a new space such as the Laplace or Fourier transform and using linear algebra to find an element in the center of the kernel of the desired transform. The proper transformation must be found to produce the desired result. That is to say a different transform is used for finding a sharpening convolution kernel than a edge defining convolution kernel. Usually computer scientists circumvent this challenge by looking up pre-calculated convolution kernels and using them to perform their convolution.

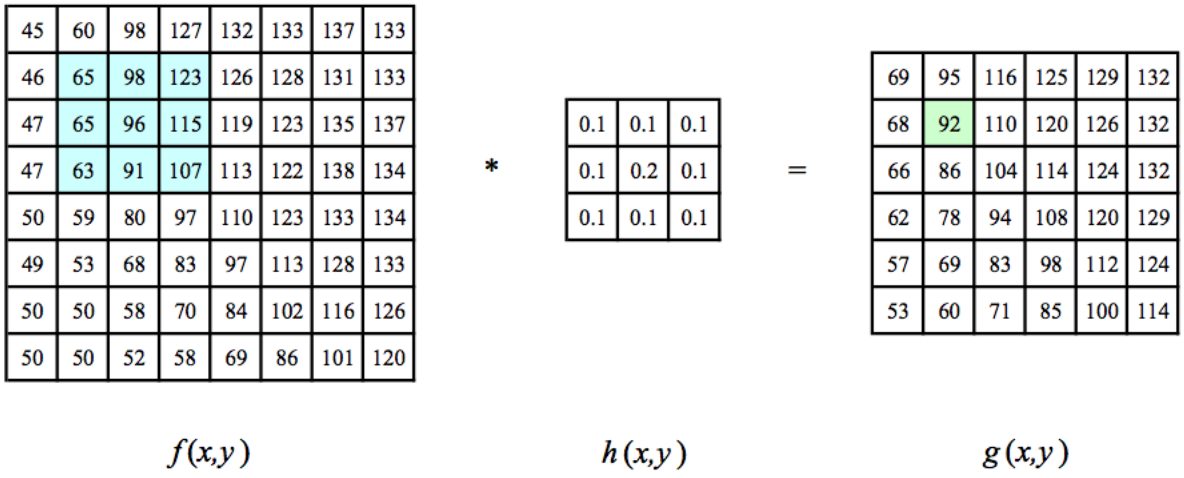

Figure 2: Example of a convolution from Richard Szeliski. The highlighted pixels on the left are manipulated via the convolution kernel $h(x, y)$ to produce the highlighted pixel on the right. Note that the edges of the matrix are eliminated since there is insufficient data to perform a convolution on these pixels.

10 Future Linear Algebra of Computer Vision

In its earlier years, computer vision relied more heavily on statistical approaches than linear algebra approaches. However, the mindset of computer scientists has shifted more and more away from the path of attempting to find the "right" answer to a vision problem, and towards the path of finding the best answer to the problem given limited time. Computer vision has shifted to an approach of approximating the world through linearized systems.

Through the continued study of linear algebra, these linearized approximations have become increasingly more accurate and efficient. Today, a rapidly growing field where a depth of study of linear algebra has dramatically improved output quality is through gesture technology. In this field, applications of linear algebra have extended to calculations such as

higher order singular value decomposition which is an extension of traditional singular value decomposition to factor multilinear tensors.

In addition, as computer vision gets faster and more accurate in visualizing the world around us, automation of everyday systems is improved. For example, companies like Google use computer vision to rapidly predict the surroundings of their driverless car with the best accuracy possible to know where lanes, vehicles, and other objects are.

According to one computer vision expert from Georgia Tech University, even some of the most fundamental problems in computer vision such as "structure from motion is not perfect." (Interviews 2014) He believes that through the "deeper study of linear algebra" (Interviews 2014) and more sophisticated computer systems, computer vision will continue to grow and move closer to this more perfect solution.

11 Sources

1. Szeliski, Richard. 2011. Computer vision algorithms and applications. London: Springer.

2. Trefethen, Lloyd N., and David Bau. 1997. Numerical linear algebra. Philadelphia, PA: Society for Industrial and Applied Mathematics.

3. Doornik, Jurgen A. 2011. Robust Estimation Using Least Trimmed Squares. Oxford, UK: Oxford Martin School.

4. Smith, Lindsay I. 2002. A Tutorial on Principal Components Analysis.

5. Turk, Matthew, and Alex Pentland. 1991. Eigenfaces for Recognition. Massachusetts Institute of Technology. Journal of Cognitive Neuroscience.

6. Massachusetts Institute of Technology. Chemical Engineering Notes: Cholesky Decomposition.

7. Shah, Mubarak. 1997. Fundamentals of Computer Vision. Orlando, FL: University of Central Florida.

8. Golub, Gene H., and Charles F. Van Loan. 1983. Matrix computations. Baltimore: Johns Hopkins University Press.

9. Lui, Yui M. 2012. Human Gesture Recognition on Product Manifolds. Journal of Machine Learning Research.

10. Ahn, Song H. 2005. Homogenous Coordinates.

11. Castro, Luis P., Saburou Saitoh, and Nguyen Minh Tuan. 2012. Convolutions, Integral Transforms, and Integral Equations by means of the Theory of Reproducing Kernels. Opuscula Mathematica Vol. 32.

12. O'Hagan, Anthony, Caitlin E. Buck, Alireza Daneshkhah, J. Richard Eiser, Paul H. Garthwaite, David J. Jenkinson, Jeremy E. Oakley, and Tim Rakow. 2006. Uncertain Judgements: Eliciting Experts' Probabilities. West Sussex: John Wiley and Sons Ltd.

13. Interviews of industry computer scientists coming from a Ph.D. program in Computer Vision from Georgia Tech University. 2014.

12 Appendix: Examples and Visualizations

Figure 3: A picture of railway lines showing how two parallel lines appear get closer together and intersect at infinity from Song Ho Ahn's Homogenous Coordinates.

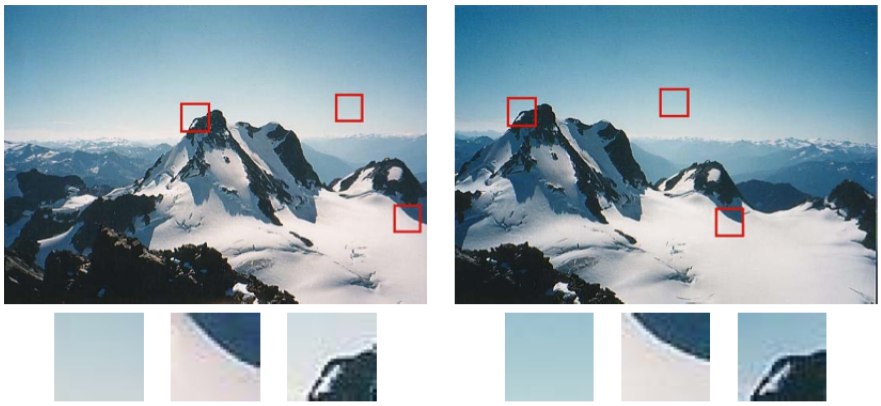

Figure 4: Two pictures being matched based on the identification of three highlighted feature points from Richard Szeliski.

Figure 5: On the left are a couple of the pictures taken to create a structure from motion representation shown on the right.

Figure 6: A set of face images for calculations of eigenfaces and the seven most significant eigenfaces calculated. Images taken from Eigenfaces for Recognition by Matthew Turk and Alex Pentland.

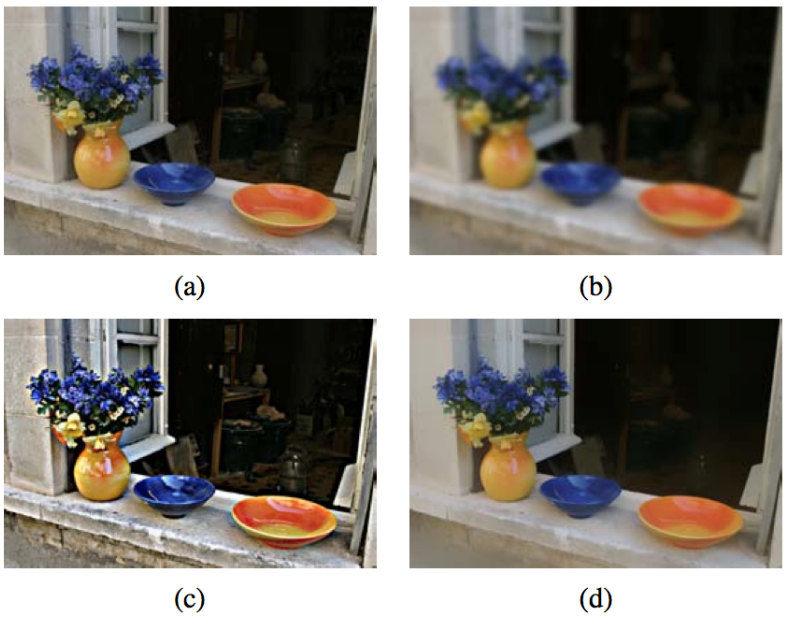

Figure 7: A visual that has been manipulated via multiple convolution transforms from Richard Szeliski. (a) is the original, (b) is blurred, (c) is sharpened, and (d) is smoothed with an edge-preserving filter.