Chapter 1

Introduction

Figure 1.1 The human visual system has no problem interpreting the subtle variations in translucency and shading in this photograph and correctly segmenting the object from its background.

R. Szeliski, *Computer Vision*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-34372-9_1

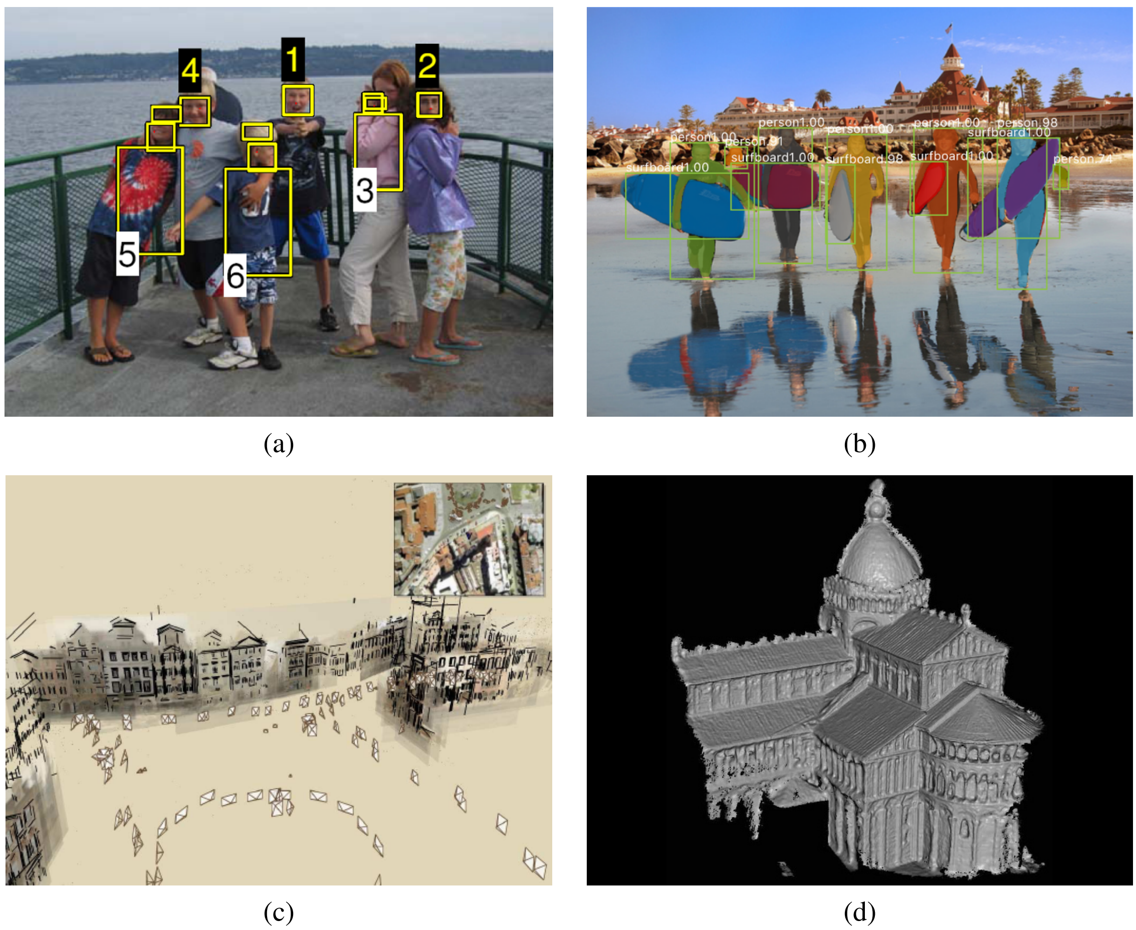

Figure 1.2 Some examples of computer vision algorithms and applications. (a) *Face detection* algorithms, coupled with color-based clothing and hair detection algorithms, can locate and recognize the individuals in this image (Sivic, Zitnick, and Szeliski 2006) © 2006 Springer. (b) *Object instance segmentation* can delineate each person and object in a complex scene (He, Gkioxari *et al.* 2017) © 2017 IEEE. (c) *Structure from motion* algorithms can reconstruct a sparse 3D point model of a large complex scene from hundreds of partially overlapping photographs (Snavely, Seitz, and Szeliski 2006) © 2006 ACM. (d) *Stereo matching* algorithms can build a detailed 3D model of a building façade from hundreds of differently exposed photographs taken from the internet (Goesele, Snavely *et al.* 2007) © 2007 IEEE.

1.1 What is computer vision?

As humans, we perceive the three-dimensional structure of the world around us with apparent ease. Think of how vivid the three-dimensional percept is when you look at a vase of flowers sitting on the table next to you. You can tell the shape and translucency of each petal through the subtle patterns of light and shading that play across its surface and effortlessly segment each flower from the background of the scene (Figure 1.1). Looking at a framed group portrait, you can easily count and name all of the people in the picture and even guess at their emotions from their facial expressions (Figure 1.2a). Perceptual psychologists have spent decades trying to understand how the visual system works and, even though they can devise optical illusions[1] to tease apart some of its principles (Figure 1.3), a complete solution to this puzzle remains elusive (Marr 1982; Wandell 1995; Palmer 1999; Livingstone 2008; Frisby and Stone 2010).

Researchers in computer vision have been developing, in parallel, mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery. Here, the progress in the last two decades has been rapid. We now have reliable techniques for accurately computing a 3D model of an environment from thousands of partially overlapping photographs (Figure 1.2c). Given a large enough set of views of a particular object or façade, we can create accurate dense 3D surface models using stereo matching (Figure 1.2d). We can even, with moderate success, delineate most of the people and objects in a photograph (Figure 1.2a). However, despite all of these advances, the dream of having a computer explain an image at the same level of detail and causality as a two-year old remains elusive.

Why is vision so difficult? In part, it is because it is an *inverse problem*, in which we seek to recover some unknowns given insufficient information to fully specify the solution. We must therefore resort to physics-based and probabilistic *models*, or machine learning from large sets of examples, to disambiguate between potential solutions. However, modeling the visual world in all of its rich complexity is far more difficult than, say, modeling the vocal tract that produces spoken sounds.

The *forward* models that we use in computer vision are usually developed in physics (radiometry, optics, and sensor design) and in computer graphics. Both of these fields model how objects move and animate, how light reflects off their surfaces, is scattered by the atmosphere, refracted through camera lenses (or human eyes), and finally projected onto a flat (or curved) image plane. While computer graphics are not yet perfect, in many domains, such as rendering a still scene composed of everyday objects or animating extinct creatures such as dinosaurs, the illusion of reality is essentially there.

In computer vision, we are trying to do the inverse, i.e., to describe the world that we see in one or more images and to reconstruct its properties, such as shape, illumination, and color distributions. It is amazing that humans and animals do this so effortlessly, while computer vision algorithms are so error prone. People who have not worked in the field often underestimate the difficulty of the problem. This misperception that vision should be easy dates back to the early days of artificial intelligence (see Section 1.2), when it was initially believed that the *cognitive* (logic proving and planning) parts of intelligence were intrinsically more difficult than the *perceptual* components (Boden 2006).

The good news is that computer vision *is* being used today in a wide variety of real-world applications, which include:

- **Optical character recognition (OCR):** reading handwritten postal codes on letters (Fig-

[1] Some fun pages with striking illusions include https://michaelbach.de/ot, https://www.illusionsindex.org, and http://www.ritsumei.ac.jp/~akitaoka/index-e.html.

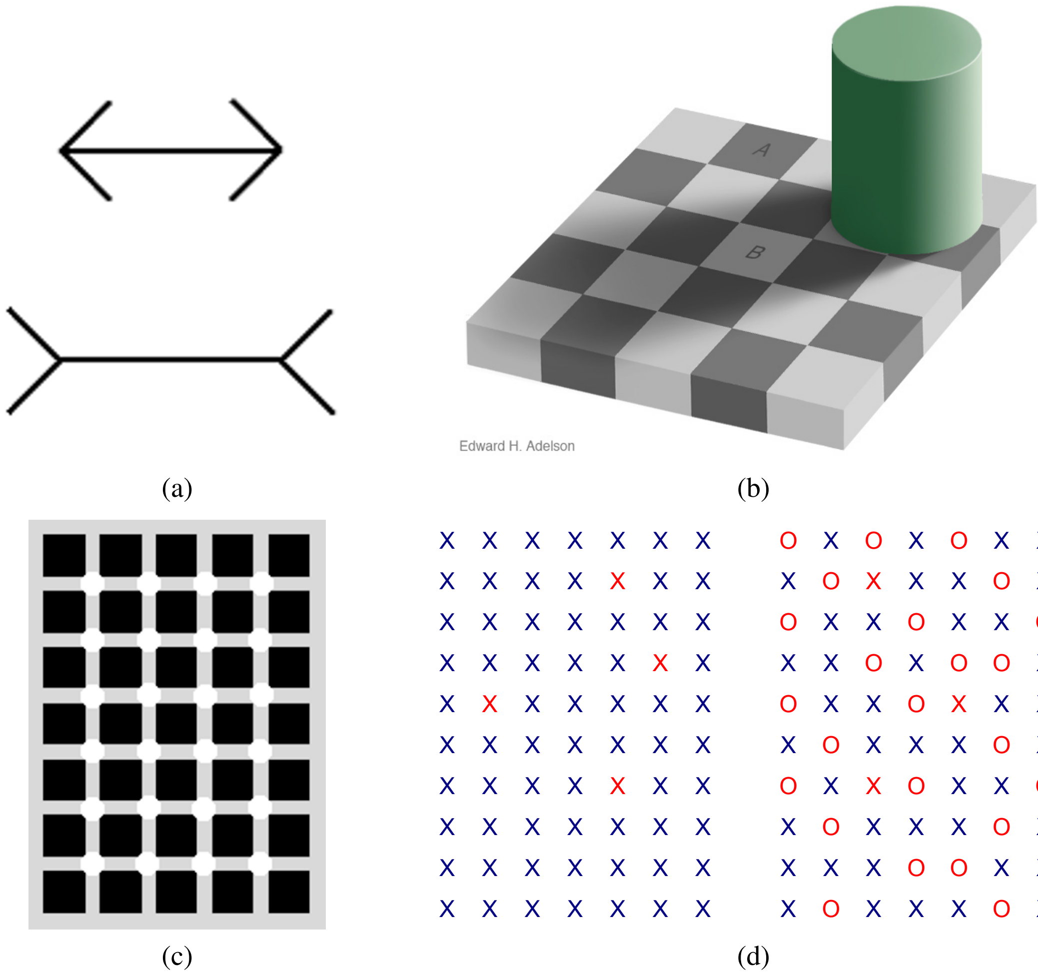

Figure 1.3 Some common optical illusions and what they might tell us about the visual system: (a) The classic Müller-Lyer illusion, where the lengths of the two horizontal lines appear different, probably due to the imagined perspective effects. (b) The "white" square B in the shadow and the "black" square A in the light actually have the same absolute intensity value. The percept is due to *brightness constancy*, the visual system's attempt to discount illumination when interpreting colors. Image courtesy of Ted Adelson, http://persci.mit.edu/gallery/checkershadow. (c) A variation of the Hermann grid illusion, courtesy of Hany Farid. As you move your eyes over the figure, gray spots appear at the intersections. (d) Count the red *X*s in the left half of the figure. Now count them in the right half. Is it significantly harder? The explanation has to do with a *pop-out* effect (Treisman 1985), which tells us about the operations of parallel perception and integration pathways in the brain.

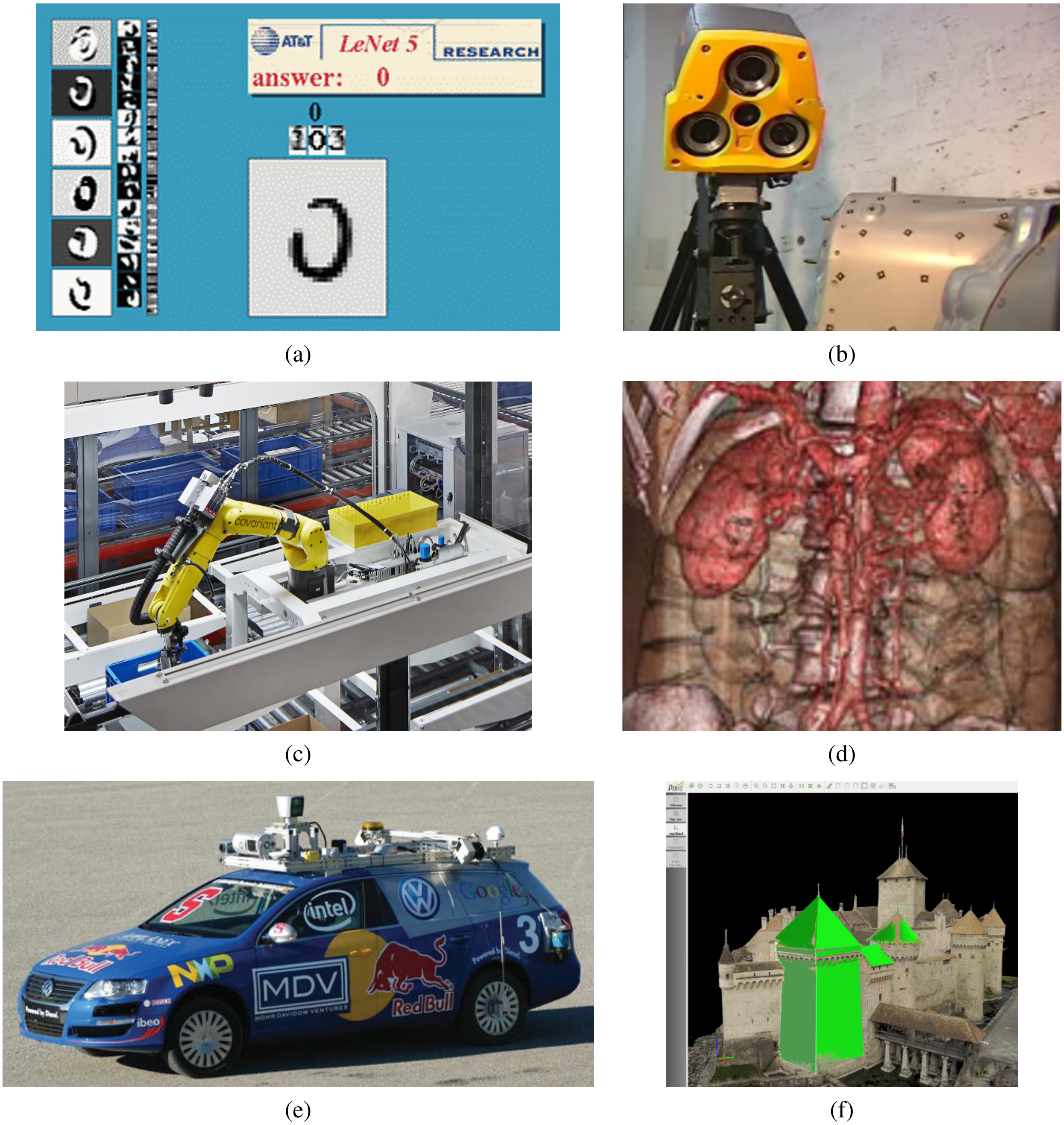

Figure 1.4 Some industrial applications of computer vision: (a) optical character recognition (OCR), http://yann.lecun.com/exdb/lenet; (b) mechanical inspection, http://www.cognitens.com; (c) warehouse picking, https://covariant.ai; (d) medical imaging, http://www.clarontech.com; (e) self-driving cars, (Montemerlo, Becker *et al.* 2008) © 2008 Wiley; (f) drone-based photogrammetry, https://www.pix4d.com/blog/mapping-chillon-castle-with-drone.

ure 1.4a) and automatic number plate recognition (ANPR);

- **Machine inspection:** rapid parts inspection for quality assurance using stereo vision with specialized illumination to measure tolerances on aircraft wings or auto body parts (Figure 1.4b) or looking for defects in steel castings using X-ray vision;
- **Retail:** object recognition for automated checkout lanes and fully automated stores (Wingfield 2019);
- **Warehouse logistics:** autonomous package delivery and pallet-carrying "drives" (Guizzo 2008; O'Brian 2019) and parts picking by robotic manipulators (Figure 1.4c; Ackerman 2020);
- **Medical imaging:** registering pre-operative and intra-operative imagery (Figure 1.4d) or performing long-term studies of people's brain morphology as they age;
- **Self-driving vehicles:** capable of driving point-to-point between cities (Figure 1.4e; Montemerlo, Becker *et al.* 2008; Urmson, Anhalt *et al.* 2008; Janai, Güney *et al.* 2020) as well as autonomous flight (Kaufmann, Gehrig *et al.* 2019);
- **3D model building (photogrammetry):** fully automated construction of 3D models from aerial and drone photographs (Figure 1.4f);
- **Match move:** merging computer-generated imagery (CGI) with live action footage by tracking feature points in the source video to estimate the 3D camera motion and shape of the environment. Such techniques are widely used in Hollywood, e.g., in movies such as Jurassic Park (Roble 1999; Roble and Zafar 2009); they also require the use of precise *matting* to insert new elements between foreground and background elements (Chuang, Agarwala *et al.* 2002).
- **Motion capture (mocap):** using retro-reflective markers viewed from multiple cameras or other vision-based techniques to capture actors for computer animation;
- **Surveillance:** monitoring for intruders, analyzing highway traffic and monitoring pools for drowning victims (e.g., https://swimeye.com);
- **Fingerprint recognition and biometrics:** for automatic access authentication as well as forensic applications.

David Lowe's website of industrial vision applications (http://www.cs.ubc.ca/spider/lowe/vision.html) lists many other interesting industrial applications of computer vision. While the above applications are all extremely important, they mostly pertain to fairly specialized kinds of imagery and narrow domains.

In addition to all of these industrial applications, there exist myriad *consumer-level* applications, such as things you can do with your own personal photographs and video. These include:

- **Stitching:** turning overlapping photos into a single seamlessly stitched panorama (Figure 1.5a), as described in Section 8.2;
- **Exposure bracketing:** merging multiple exposures taken under challenging lighting conditions (strong sunlight and shadows) into a single perfectly exposed image (Figure 1.5b), as described in Section 10.2;
- **Morphing:** turning a picture of one of your friends into another, using a seamless *morph* transition (Figure 1.5c);

- **3D modeling:** converting one or more snapshots into a 3D model of the object or person you are photographing (Figure 1.5d), as described in Section 13.6;
- **Video match move and stabilization:** inserting 2D pictures or 3D models into your videos by automatically tracking nearby reference points (see Section 11.4.4)[2] or using motion estimates to remove shake from your videos (see Section 9.2.1);
- **Photo-based walkthroughs:** navigating a large collection of photographs, such as the interior of your house, by flying between different photos in 3D (see Sections 14.1.2 and 14.5.5);
- **Face detection:** for improved camera focusing as well as more relevant image searching (see Section 6.3.1);
- **Visual authentication:** automatically logging family members onto your home computer as they sit down in front of the webcam (see Section 6.2.4).

The great thing about these applications is that they are already familiar to most students; they are, at least, technologies that students can immediately appreciate and use with their own personal media. Since computer vision is a challenging topic, given the wide range of mathematics being covered[3] and the intrinsically difficult nature of the problems being solved, having fun and relevant problems to work on can be highly motivating and inspiring.

The other major reason why this book has a strong focus on applications is that they can be used to *formulate* and *constrain* the potentially open-ended problems endemic in vision. Thus, it is better to think back from the problem at hand to suitable techniques, rather than to grab the first technique that you may have heard of. This kind of working back from problems to solutions is typical of an **engineering** approach to the study of vision and reflects my own background in the field.

First, I come up with a detailed problem definition and decide on the constraints and specifications for the problem. Then, I try to find out which techniques are known to work, implement a few of these, evaluate their performance, and finally make a selection. In order for this process to work, it is important to have realistic **test data**, both synthetic, which can be used to verify correctness and analyze noise sensitivity, and real-world data typical of the way the system will finally be used. If machine learning is being used, it is even more important to have representative unbiased **training data** in sufficient quantity to obtain good results on real-world inputs.

However, this book is not just an engineering text (a source of recipes). It also takes a **scientific** approach to basic vision problems. Here, I try to come up with the best possible models of the physics of the system at hand: how the scene is created, how light interacts with the scene and atmospheric effects, and how the sensors work, including sources of noise and uncertainty. The task is then to try to invert the acquisition process to come up with the best possible description of the scene.

The book often uses a **statistical** approach to formulating and solving computer vision problems. Where appropriate, probability distributions are used to model the scene and the noisy image acquisition process. The association of prior distributions with unknowns is often called *Bayesian modeling* (Appendix B). It is possible to associate a risk or loss function with misestimating the answer (Section B.2) and to set up your inference algorithm to minimize the expected risk. (Consider a robot trying to estimate the distance to an obstacle: it is usually safer to underestimate than to overestimate.) With statistical techniques, it often helps to gather lots of training data from which to

[2] For a fun student project on this topic, see the "PhotoBook" project at http://www.cc.gatech.edu/dvfx/videos/dvfx2005.html.

[3] These techniques include physics, Euclidean and projective geometry, statistics, and optimization. They make computer vision a fascinating field to study and a great way to learn techniques widely applicable in other fields.

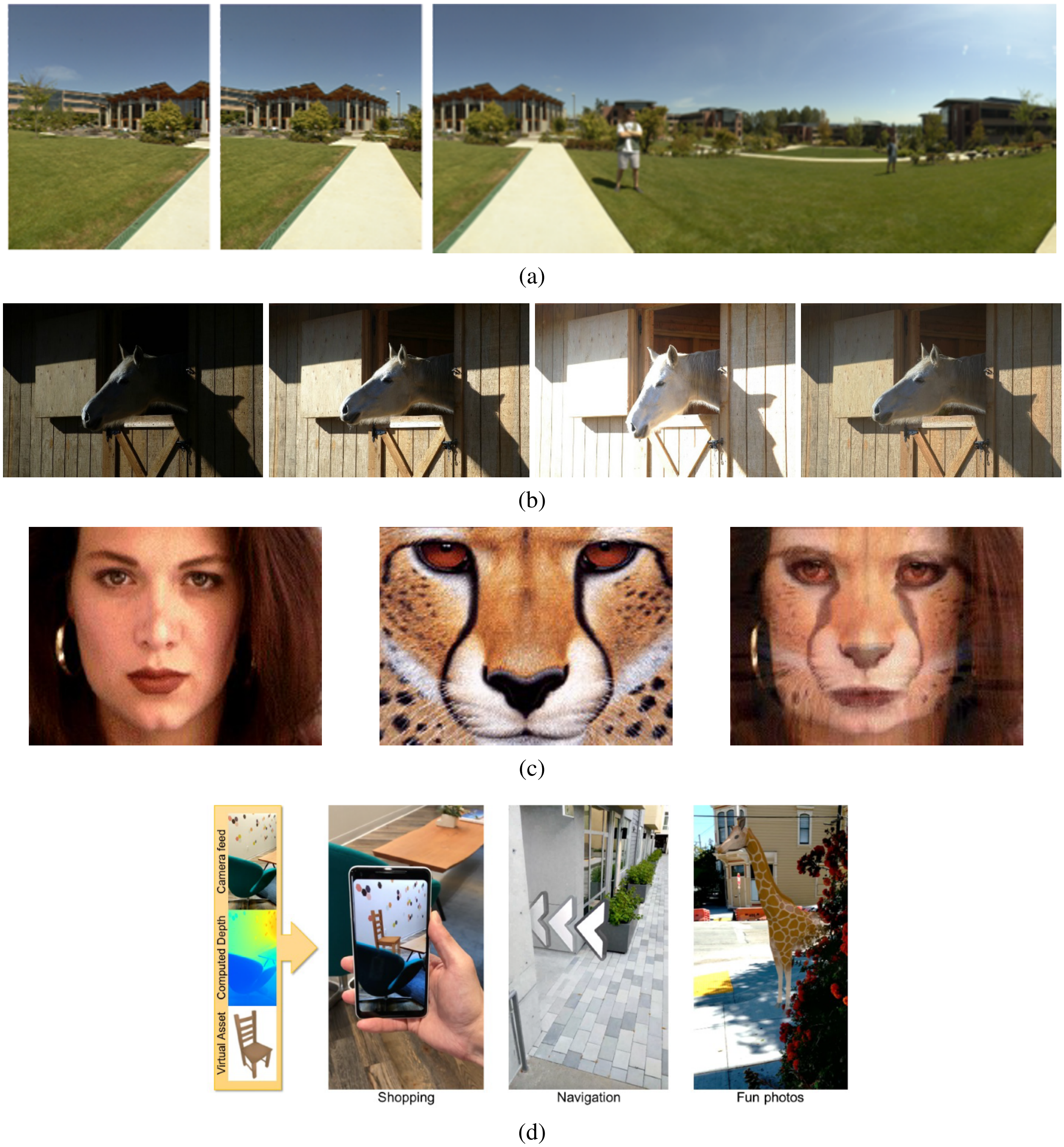

Figure 1.5 Some consumer applications of computer vision: (a) image stitching: merging different views (Szeliski and Shum 1997) © 1997 ACM; (b) exposure bracketing: merging different exposures; (c) morphing: blending between two photographs (Gomes, Darsa *et al.* 1999) © 1999 Morgan Kaufmann; (d) smartphone augmented reality showing real-time depth occlusion effects (Valentin, Kowdle *et al.* 2018) © 2018 ACM.

learn probabilistic models. Finally, statistical approaches enable you to use proven inference techniques to estimate the best answer (or distribution of answers) and to quantify the uncertainty in the resulting estimates.

Because so much of computer vision involves the solution of inverse problems or the estimation of unknown quantities, my book also has a heavy emphasis on **algorithms**, especially those that are known to work well in practice. For many vision problems, it is all too easy to come up with a mathematical description of the problem that either does not match realistic real-world conditions or does not lend itself to the stable estimation of the unknowns. What we need are algorithms that are both **robust** to noise and deviation from our models and reasonably **efficient** in terms of run-time resources and space. In this book, I go into these issues in detail, using Bayesian techniques, where applicable, to ensure robustness, and efficient search, minimization, and linear system solving algorithms to ensure efficiency.[4] Most of the algorithms described in this book are at a high level, being mostly a list of steps that have to be filled in by students or by reading more detailed descriptions elsewhere. In fact, many of the algorithms are sketched out in the exercises.

Now that I've described the goals of this book and the frameworks that I use, I devote the rest of this chapter to two additional topics. Section 1.2 is a brief synopsis of the history of computer vision. It can easily be skipped by those who want to get to "the meat" of the new material in this book and do not care as much about who invented what when.

The second is an overview of the book's contents, Section 1.3, which is useful reading for everyone who intends to make a study of this topic (or to jump in partway, since it describes chapter interdependencies). This outline is also useful for instructors looking to structure one or more courses around this topic, as it provides sample curricula based on the book's contents.

1.2 A brief history

In this section, I provide a brief personal synopsis of the main developments in computer vision over the last fifty years (Figure 1.6) with a focus on advances I find personally interesting and that have stood the test of time. Readers not interested in the provenance of various ideas and the evolution of this field should skip ahead to the book overview in Section 1.3.

1970s. When computer vision first started out in the early 1970s, it was viewed as the visual perception component of an ambitious agenda to mimic human intelligence and to endow robots with intelligent behavior. At the time, it was believed by some of the early pioneers of artificial intelligence and robotics (at places such as MIT, Stanford, and CMU) that solving the "visual input" problem would be an easy step along the path to solving more difficult problems such as higher-level reasoning and planning. According to one well-known story, in 1966, Marvin Minsky at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" (Boden 2006, p. 781).[5] We now know that the problem is slightly more difficult than that.[6]

What distinguished computer vision from the already existing field of digital image processing (Rosenfeld and Pfaltz 1966; Rosenfeld and Kak 1976) was a desire to recover the three-dimensional

[4] In some cases, deep neural networks have also been shown to be an effective way to speed up algorithms that previously relied on iteration (Chen, Xu, and Koltun 2017).

[5] Boden (2006) cites (Crevier 1993) as the original source. The actual Vision Memo was authored by Seymour Papert (1966) and involved a whole cohort of students.

[6] To see how far robotic vision has come in the last six decades, have a look at some of the videos on the Boston Dynamics https://www.bostondynamics.com, Skydio https://www.skydio.com, and Covariant https://covariant.ai websites.

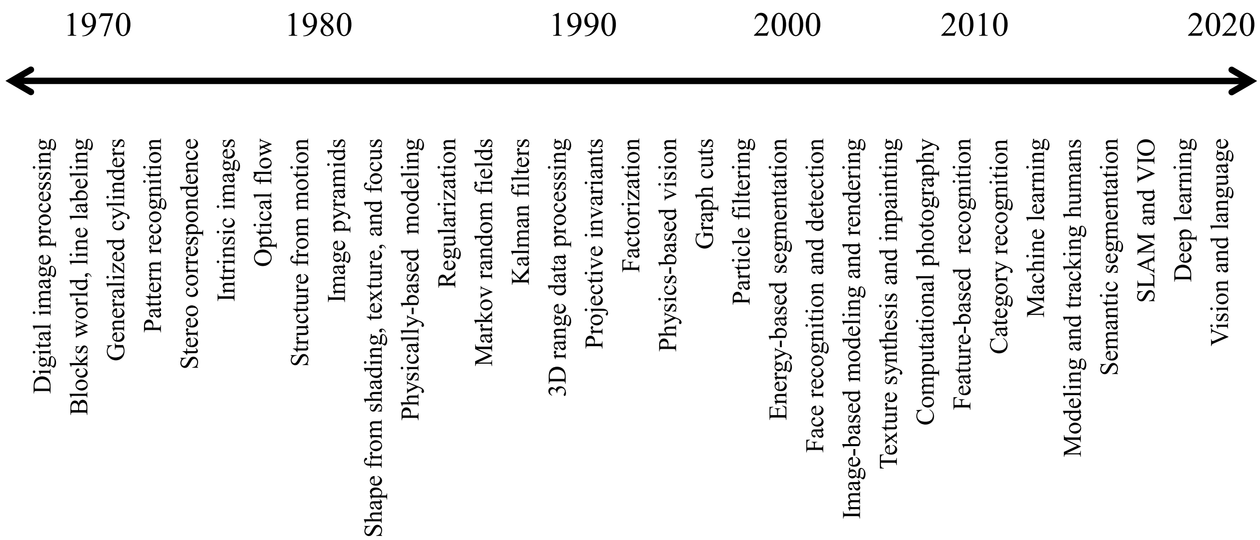

Figure 1.6 A rough timeline of some of the most active topics of research in computer vision.

structure of the world from images and to use this as a stepping stone towards full scene understanding. Winston (1975) and Hanson and Riseman (1978) provide two nice collections of classic papers from this early period.

Early attempts at scene understanding involved extracting edges and then inferring the 3D structure of an object or a "blocks world" from the topological structure of the 2D lines (Roberts 1965). Several *line labeling* algorithms (Figure 1.7a) were developed at that time (Huffman 1971; Clowes 1971; Waltz 1975; Rosenfeld, Hummel, and Zucker 1976; Kanade 1980). Nalwa (1993) gives a nice review of this area. The topic of edge detection was also an active area of research; a nice survey of contemporaneous work can be found in (Davis 1975).

Three-dimensional modeling of non-polyhedral objects was also being studied (Baumgart 1974; Baker 1977). One popular approach used *generalized cylinders*, i.e., solids of revolution and swept closed curves (Agin and Binford 1976; Nevatia and Binford 1977), often arranged into parts relationships[7] (Hinton 1977; Marr 1982) (Figure 1.7c). Fischler and Elschlager (1973) called such *elastic* arrangements of parts *pictorial structures* (Figure 1.7b).

A qualitative approach to understanding intensities and shading variations and explaining them by the effects of image formation phenomena, such as surface orientation and shadows, was championed by Barrow and Tenenbaum (1981) in their paper on *intrinsic images* (Figure 1.7d), along with the related *2½-D sketch* ideas of Marr (1982). This approach has seen periodic revivals, e.g., in the work of Tappen, Freeman, and Adelson (2005) and Barron and Malik (2012).

More quantitative approaches to computer vision were also developed at the time, including the first of many feature-based stereo correspondence algorithms (Figure 1.7e) (Dev 1974; Marr and Poggio 1976, 1979; Barnard and Fischler 1982; Ohta and Kanade 1985; Grimson 1985; Pollard, Mayhew, and Frisby 1985) and intensity-based optical flow algorithms (Figure 1.7f) (Horn and Schunck 1981; Huang 1981; Lucas and Kanade 1981; Nagel 1986). The early work in simultaneously recovering 3D structure and camera motion (see Chapter 11) also began around this time (Ullman 1979; Longuet-Higgins 1981).

[7] In robotics and computer animation, these linked-part graphs are often called *kinematic chains*.

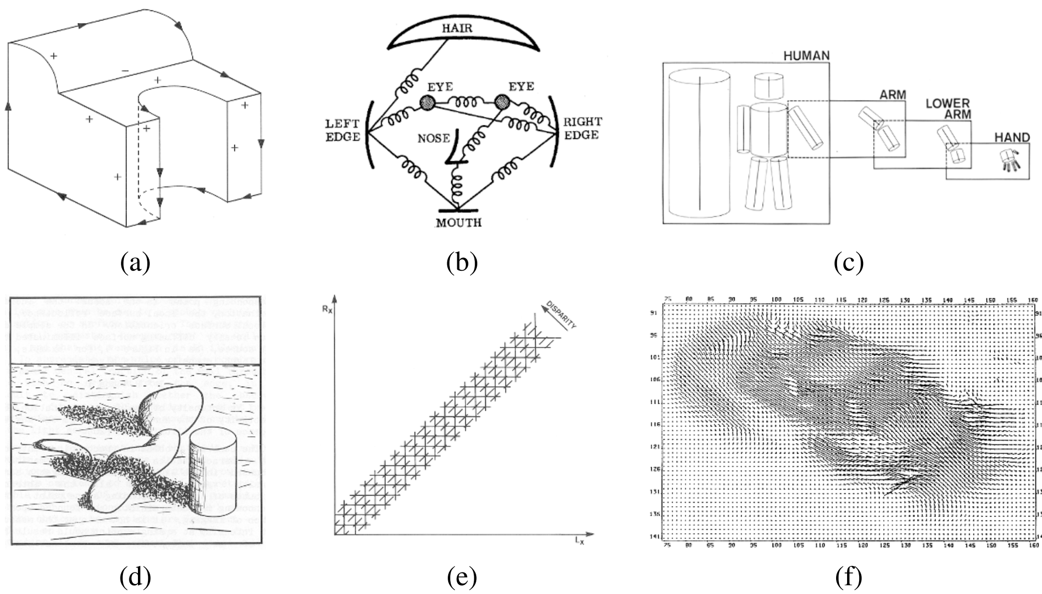

Figure 1.7 Some early (1970s) examples of computer vision algorithms: (a) line labeling (Nalwa 1993) © 1993 Addison-Wesley, (b) pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE, (c) articulated body model (Marr 1982) © 1982 David Marr, (d) intrinsic images (Barrow and Tenenbaum 1981) © 1973 IEEE, (e) stereo correspondence (Marr 1982) © 1982 David Marr, (f) optical flow (Nagel and Enkelmann 1986) © 1986 IEEE.

A lot of the philosophy of how vision was believed to work at the time is summarized in David Marr's (1982) book.[8] In particular, Marr introduced his notion of the three levels of description of a (visual) information processing system. These three levels, very loosely paraphrased according to my own interpretation, are:

- **Computational theory:** What is the goal of the computation (task) and what are the constraints that are known or can be brought to bear on the problem?

- **Representations and algorithms:** How are the input, output, and intermediate information represented and which algorithms are used to calculate the desired result?

- **Hardware implementation:** How are the representations and algorithms mapped onto actual hardware, e.g., a biological vision system or a specialized piece of silicon? Conversely, how can hardware constraints be used to guide the choice of representation and algorithm? With the prevalent use of graphics chips (GPUs) and many-core architectures for computer vision, this question is again quite relevant.

As I mentioned earlier in this introduction, it is my conviction that a careful analysis of the problem specification and known constraints from image formation and priors (the scientific and statistical approaches) must be married with efficient and robust algorithms (the engineering approach) to design successful vision algorithms. Thus, it seems that Marr's philosophy is as good a guide to framing and solving problems in our field today as it was 25 years ago.

[8] More recent developments in visual perception theory are covered in (Wandell 1995; Palmer 1999; Livingstone 2008; Frisby and Stone 2010).

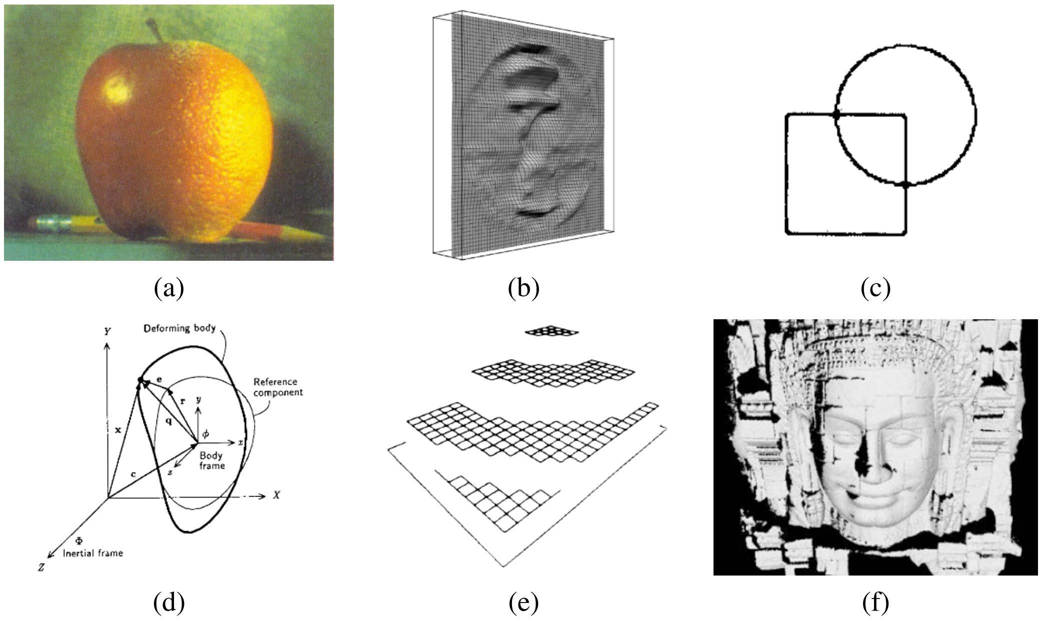

Figure 1.8 Examples of computer vision algorithms from the 1980s: (a) pyramid blending (Burt and Adelson 1983b) © 1983 ACM, (b) shape from shading (Freeman and Adelson 1991) © 1991 IEEE, (c) edge detection (Freeman and Adelson 1991) © 1991 IEEE, (d) physically based models (Terzopoulos and Witkin 1988) © 1988 IEEE, (e) regularization-based surface reconstruction (Terzopoulos 1988) © 1988 IEEE, (f) range data acquisition and merging (Banno, Masuda *et al.* 2008) © 2008 Springer.

1980s. In the 1980s, a lot of attention was focused on more sophisticated mathematical techniques for performing quantitative image and scene analysis.

Image pyramids (see Section 3.5) started being widely used to perform tasks such as image blending (Figure 1.8a) and coarse-to-fine correspondence search (Rosenfeld 1980; Burt and Adelson 1983b; Rosenfeld 1984; Quam 1984; Anandan 1989). Continuous versions of pyramids using the concept of *scale-space* processing were also developed (Witkin 1983; Witkin, Terzopoulos, and Kass 1986; Lindeberg 1990). In the late 1980s, wavelets (see Section 3.5.4) started displacing or augmenting regular image pyramids in some applications (Mallat 1989; Simoncelli and Adelson 1990a; Simoncelli, Freeman *et al.* 1992).

The use of stereo as a quantitative shape cue was extended by a wide variety of *shape-from-X* techniques, including shape from shading (Figure 1.8b) (see Section 13.1.1 and Horn 1975; Pentland 1984; Blake, Zisserman, and Knowles 1985; Horn and Brooks 1986, 1989), photometric stereo (see Section 13.1.1 and Woodham 1981), shape from texture (see Section 13.1.2 and Witkin 1981; Pentland 1984; Malik and Rosenholtz 1997), and shape from focus (see Section 13.1.3 and Nayar, Watanabe, and Noguchi 1995). Horn (1986) has a nice discussion of most of these techniques.

Research into better edge and contour detection (Figure 1.8c) (see Section 7.2) was also active during this period (Canny 1986; Nalwa and Binford 1986), including the introduction of dynamically evolving contour trackers (Section 7.3.1) such as *snakes* (Kass, Witkin, and Terzopoulos 1988), as well as three-dimensional *physically based models* (Figure 1.8d) (Terzopoulos, Witkin, and Kass 1987; Kass, Witkin, and Terzopoulos 1988; Terzopoulos and Fleischer 1988).

Researchers noticed that a lot of the stereo, flow, shape-from-X, and edge detection algorithms could be unified, or at least described, using the same mathematical framework if they were posed as variational optimization problems and made more robust (well-posed) using regularization (Fig-

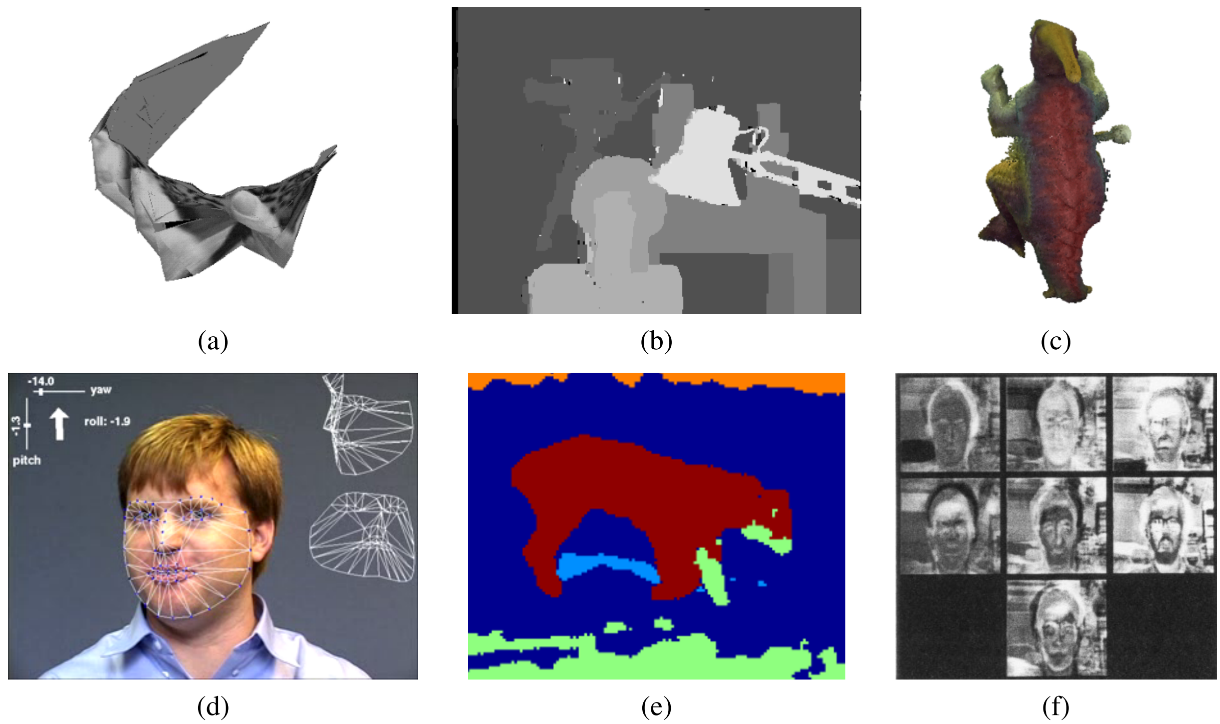

Figure 1.9 Examples of computer vision algorithms from the 1990s: (a) factorization-based structure from motion (Tomasi and Kanade 1992) © 1992 Springer, (b) dense stereo matching (Boykov, Veksler, and Zabih 2001), (c) multi-view reconstruction (Seitz and Dyer 1999) © 1999 Springer, (d) face tracking (Matthews, Xiao, and Baker 2007), (e) image segmentation (Belongie, Fowlkes *et al.* 2002) © 2002 Springer, (f) face recognition (Turk and Pentland 1991).

ure 1.8e) (see Section 4.2 and Terzopoulos 1983; Poggio, Torre, and Koch 1985; Terzopoulos 1986b; Blake and Zisserman 1987; Bertero, Poggio, and Torre 1988; Terzopoulos 1988). Around the same time, Geman and Geman (1984) pointed out that such problems could equally well be formulated using discrete *Markov random field* (MRF) models (see Section 4.3), which enabled the use of better (global) search and optimization algorithms, such as simulated annealing.

Online variants of MRF algorithms that modeled and updated uncertainties using the Kalman filter were introduced a little later (Dickmanns and Graefe 1988; Matthies, Kanade, and Szeliski 1989; Szeliski 1989). Attempts were also made to map both regularized and MRF algorithms onto parallel hardware (Poggio and Koch 1985; Poggio, Little *et al.* 1988; Fischler, Firschein *et al.* 1989). The book by Fischler and Firschein (1987) contains a nice collection of articles focusing on all of these topics (stereo, flow, regularization, MRFs, and even higher-level vision).

Three-dimensional range data processing (acquisition, merging, modeling, and recognition; see Figure 1.8f) continued being actively explored during this decade (Agin and Binford 1976; Besl and Jain 1985; Faugeras and Hebert 1987; Curless and Levoy 1996). The compilation by Kanade (1987) contains a lot of the interesting papers in this area.

1990s. While a lot of the previously mentioned topics continued to be explored, a few of them became significantly more active.

A burst of activity in using projective invariants for recognition (Mundy and Zisserman 1992) evolved into a concerted effort to solve the structure from motion problem (see Chapter 11). A lot of the initial activity was directed at *projective reconstructions*, which did not require knowledge of camera calibration (Faugeras 1992; Hartley, Gupta, and Chang 1992; Hartley 1994a; Faugeras and Luong 2001; Hartley and Zisserman 2004). Simultaneously, *factorization* techniques (Section 11.4.1) were developed to solve efficiently problems for which orthographic camera approximations were applicable (Figure 1.9a) (Tomasi and Kanade 1992; Poelman and Kanade 1997; Anandan and Irani 2002) and then later extended to the perspective case (Christy and Horaud 1996; Triggs 1996). Eventually, the field started using full global optimization (see Section 11.4.2 and Taylor, Kriegman, and Anandan 1991; Szeliski and Kang 1994; Azarbayejani and Pentland 1995), which was later recognized as being the same as the *bundle adjustment* techniques traditionally used in photogrammetry (Triggs, McLauchlan *et al.* 1999). Fully automated 3D modeling systems were built using such techniques (Beardsley, Torr, and Zisserman 1996; Schaffalitzky and Zisserman 2002; Snavely, Seitz, and Szeliski 2006; Agarwal, Furukawa *et al.* 2011; Frahm, Fite-Georgel *et al.* 2010).

Work begun in the 1980s on using detailed measurements of color and intensity combined with accurate physical models of radiance transport and color image formation created its own subfield known as *physics-based vision*. A good survey of the field can be found in the three-volume collection on this topic (Wolff, Shafer, and Healey 1992a; Healey and Shafer 1992; Shafer, Healey, and Wolff 1992).

Optical flow methods (see Chapter 9) continued to be improved (Nagel and Enkelmann 1986; Bolles, Baker, and Marimont 1987; Horn and Weldon Jr. 1988; Anandan 1989; Bergen, Anandan *et al.* 1992; Black and Anandan 1996; Bruhn, Weickert, and Schnörr 2005; Papenberg, Bruhn *et al.* 2006), with (Nagel 1986; Barron, Fleet, and Beauchemin 1994; Baker, Scharstein *et al.* 2011) being good surveys. Similarly, a lot of progress was made on dense stereo correspondence algorithms (see Chapter 12, Okutomi and Kanade (1993, 1994); Boykov, Veksler, and Zabih (1998); Birchfield and Tomasi (1999); Boykov, Veksler, and Zabih (2001), and the survey and comparison in Scharstein and Szeliski (2002)), with the biggest breakthrough being perhaps global optimization using *graph cut* techniques (Figure 1.9b) (Boykov, Veksler, and Zabih 2001).

Multi-view stereo algorithms (Figure 1.9c) that produce complete 3D surfaces (see Section 12.7) were also an active topic of research (Seitz and Dyer 1999; Kutulakos and Seitz 2000) that continues to be active today (Seitz, Curless *et al.* 2006; Schöps, Schönberger *et al.* 2017; Knapitsch, Park *et al.* 2017). Techniques for producing 3D volumetric descriptions from binary silhouettes (see Section 12.7.3) continued to be developed (Potmesil 1987; Srivasan, Liang, and Hackwood 1990; Szeliski 1993; Laurentini 1994), along with techniques based on tracking and reconstructing smooth occluding contours (see Section 12.2.1 and Cipolla and Blake 1992; Vaillant and Faugeras 1992; Zheng 1994; Boyer and Berger 1997; Szeliski and Weiss 1998; Cipolla and Giblin 2000).

Tracking algorithms also improved a lot, including contour tracking using *active contours* (see Section 7.3), such as *snakes* (Kass, Witkin, and Terzopoulos 1988), *particle filters* (Blake and Isard 1998), and *level sets* (Malladi, Sethian, and Vemuri 1995), as well as intensity-based (*direct*) techniques (Lucas and Kanade 1981; Shi and Tomasi 1994; Rehg and Kanade 1994), often applied to tracking faces (Figure 1.9d) (Lanitis, Taylor, and Cootes 1997; Matthews and Baker 2004; Matthews, Xiao, and Baker 2007) and whole bodies (Sidenbladh, Black, and Fleet 2000; Hilton, Fua, and Ronfard 2006; Moeslund, Hilton, and Krüger 2006).

Image segmentation (see Section 7.5) (Figure 1.9e), a topic which has been active since the earliest days of computer vision (Brice and Fennema 1970; Horowitz and Pavlidis 1976; Riseman and Arbib 1977; Rosenfeld and Davis 1979; Haralick and Shapiro 1985; Pavlidis and Liow 1990), was also an active topic of research, producing techniques based on minimum energy (Mumford and Shah 1989) and minimum description length (Leclerc 1989), *normalized cuts* (Shi and Malik 2000), and *mean shift* (Comaniciu and Meer 2002).

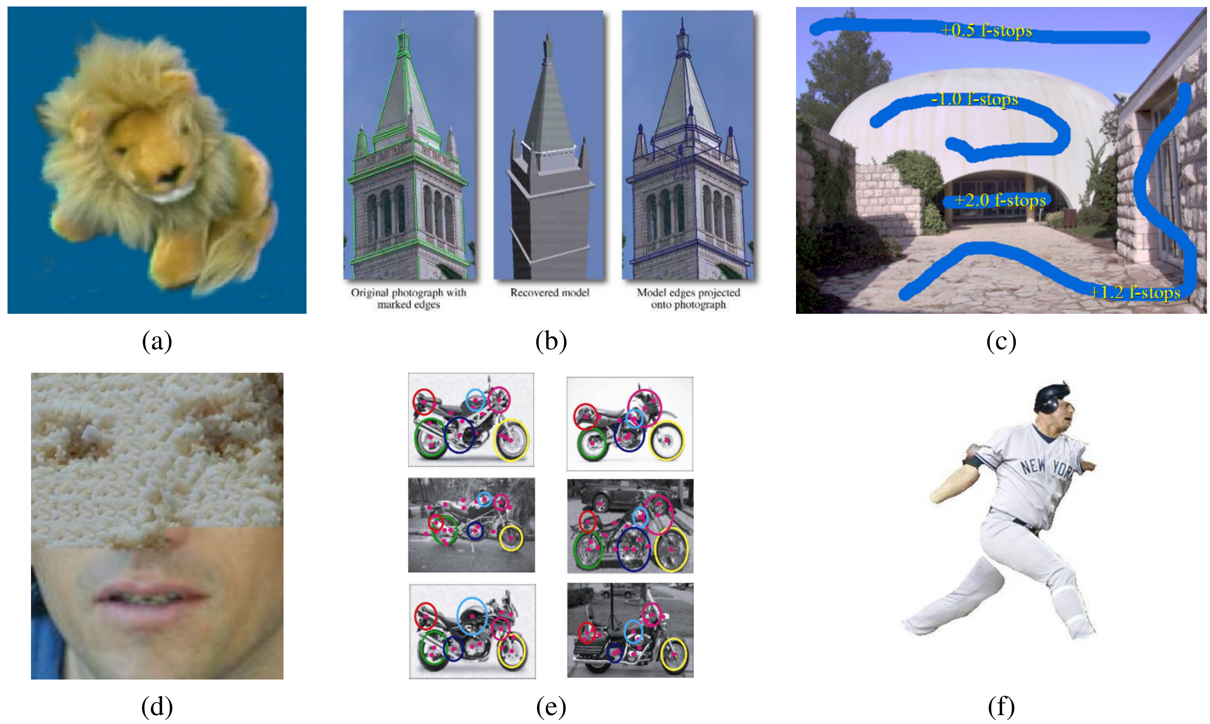

Figure 1.10 Examples of computer vision algorithms from the 2000s: (a) image-based rendering (Gortler, Grzeszczuk *et al.* 1996), (b) image-based modeling (Debevec, Taylor, and Malik 1996) © 1996 ACM, (c) interactive tone mapping (Lischinski, Farbman *et al.* 2006) (d) texture synthesis (Efros and Freeman 2001), (e) feature-based recognition (Fergus, Perona, and Zisserman 2007), (f) region-based recognition (Mori, Ren *et al.* 2004) © 2004 IEEE.

Statistical learning techniques started appearing, first in the application of principal component *eigenface* analysis to face recognition (Figure 1.9f) (see Section 5.2.3 and Turk and Pentland 1991) and linear dynamical systems for curve tracking (see Section 7.3.1 and Blake and Isard 1998).

Perhaps the most notable development in computer vision during this decade was the increased interaction with computer graphics (Seitz and Szeliski 1999), especially in the cross-disciplinary area of *image-based modeling and rendering* (see Chapter 14). The idea of manipulating real-world imagery directly to create new animations first came to prominence with *image morphing* techniques (Figure1.5c) (see Section 3.6.3 and Beier and Neely 1992) and was later applied to *view interpolation* (Chen and Williams 1993; Seitz and Dyer 1996), panoramic image stitching (Figure1.5a) (see Section 8.2 and Mann and Picard 1994; Chen 1995; Szeliski 1996; Szeliski and Shum 1997; Szeliski 2006a), and full light-field rendering (Figure 1.10a) (see Section 14.3 and Gortler, Grzeszczuk *et al.* 1996; Levoy and Hanrahan 1996; Shade, Gortler *et al.* 1998). At the same time, image-based modeling techniques (Figure 1.10b) for automatically creating realistic 3D models from collections of images were also being introduced (Beardsley, Torr, and Zisserman 1996; Debevec, Taylor, and Malik 1996; Taylor, Debevec, and Malik 1996).

2000s. This decade continued to deepen the interplay between the vision and graphics fields, but more importantly embraced data-driven and learning approaches as core components of vision. Many of the topics introduced under the rubric of image-based rendering, such as image stitching (see Section 8.2), light-field capture and rendering (see Section 14.3), and *high dynamic range*

(HDR) image capture through exposure bracketing (Figure1.5b) (see Section 10.2 and Mann and Picard 1995; Debevec and Malik 1997), were re-christened as *computational photography* (see Chapter 10) to acknowledge the increased use of such techniques in everyday digital photography. For example, the rapid adoption of exposure bracketing to create high dynamic range images necessitated the development of *tone mapping* algorithms (Figure 1.10c) (see Section 10.2.1) to convert such images back to displayable results (Fattal, Lischinski, and Werman 2002; Durand and Dorsey 2002; Reinhard, Stark *et al.* 2002; Lischinski, Farbman *et al.* 2006). In addition to merging multiple exposures, techniques were developed to merge flash images with non-flash counterparts (Eisemann and Durand 2004; Petschnigg, Agrawala *et al.* 2004) and to interactively or automatically select different regions from overlapping images (Agarwala, Dontcheva *et al.* 2004).

Texture synthesis (Figure 1.10d) (see Section 10.5), quilting (Efros and Leung 1999; Efros and Freeman 2001; Kwatra, Schödl *et al.* 2003), and inpainting (Bertalmio, Sapiro *et al.* 2000; Bertalmio, Vese *et al.* 2003; Criminisi, Pérez, and Toyama 2004) are additional topics that can be classified as computational photography techniques, since they re-combine input image samples to produce new photographs.

A second notable trend during this decade was the emergence of feature-based techniques (combined with learning) for object recognition (see Section 6.1 and Ponce, Hebert *et al.* 2006). Some of the notable papers in this area include the *constellation model* of Fergus, Perona, and Zisserman (2007) (Figure 1.10e) and the *pictorial structures* of Felzenszwalb and Huttenlocher (2005). Feature-based techniques also dominate other recognition tasks, such as scene recognition (Zhang, Marszalek *et al.* 2007) and panorama and location recognition (Brown and Lowe 2007; Schindler, Brown, and Szeliski 2007). And while *interest point* (patch-based) features tend to dominate current research, some groups are pursuing recognition based on contours (Belongie, Malik, and Puzicha 2002) and region segmentation (Figure 1.10f) (Mori, Ren *et al.* 2004).

Another significant trend from this decade was the development of more efficient algorithms for complex global optimization problems (see Chapter 4 and Appendix B.5 and Szeliski, Zabih *et al.* 2008; Blake, Kohli, and Rother 2011). While this trend began with work on graph cuts (Boykov, Veksler, and Zabih 2001; Kohli and Torr 2007), a lot of progress has also been made in message passing algorithms, such as *loopy belief propagation* (LBP) (Yedidia, Freeman, and Weiss 2001; Kumar and Torr 2006).

The most notable trend from this decade, which has by now completely taken over visual recognition and most other aspects of computer vision, was the application of sophisticated machine learning techniques to computer vision problems (see Chapters 5 and 6). This trend coincided with the increased availability of immense quantities of partially labeled data on the internet, as well as significant increases in computational power, which makes it more feasible to learn object categories without the use of careful human supervision.

2010s. The trend towards using large labeled (and also self-supervised) datasets to develop machine learning algorithms became a tidal wave that totally revolutionized the development of image recognition algorithms as well as other applications, such as denoising and optical flow, which previously used Bayesian and global optimization techniques.

This trend was enabled by the development of high-quality large-scale annotated datasets such as ImageNet (Deng, Dong *et al.* 2009; Russakovsky, Deng *et al.* 2015), Microsoft COCO (Common Objects in Context) (Lin, Maire *et al.* 2014), and LVIS (Gupta, Dollár, and Girshick 2019). These datasets provided not only reliable metrics for tracking the progress of recognition and semantic segmentation algorithms, but more importantly, sufficient labeled data to develop complete solutions based on machine learning.

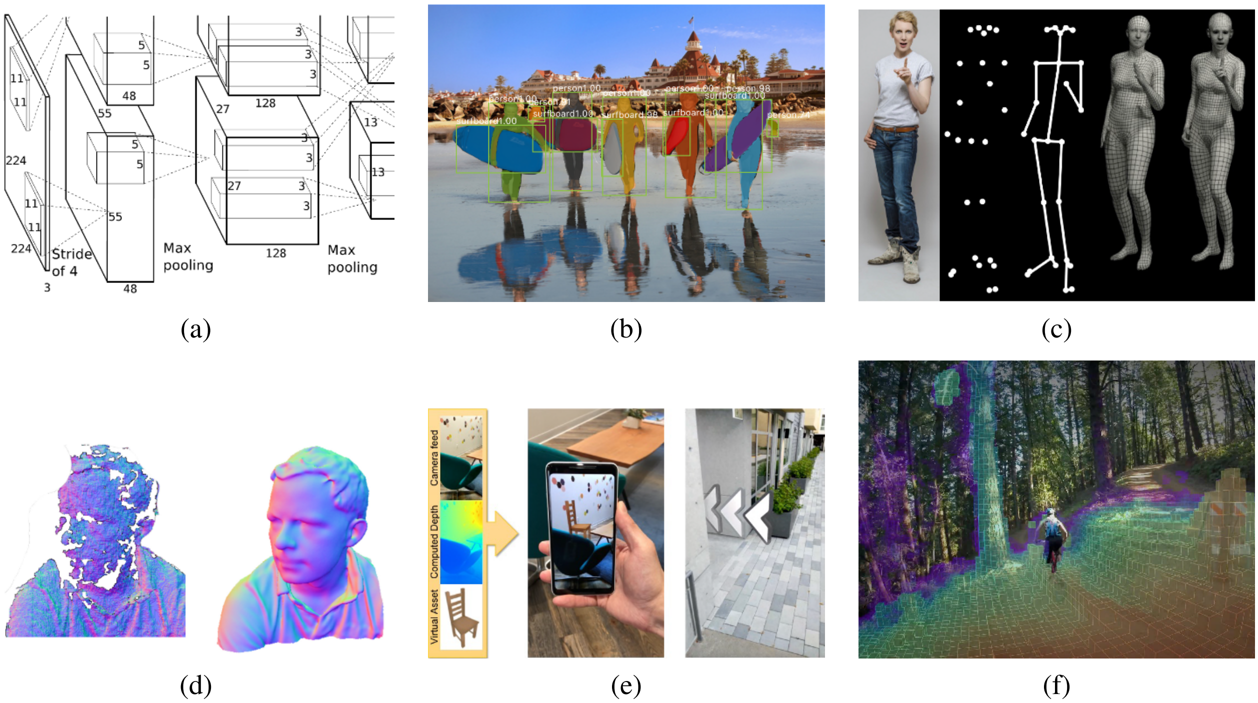

Figure 1.11 Examples of computer vision algorithms from the 2010s: (a) the SuperVision deep neural network © Krizhevsky, Sutskever, and Hinton (2012); (b) object instance segmentation (He, Gkioxari *et al.* 2017) © 2017 IEEE; (c) whole body, expression, and gesture fitting from a single image (Pavlakos, Choutas *et al.* 2019) © 2019 IEEE; (d) fusing multiple color depth images using the KinectFusion real-time system (Newcombe, Izadi *et al.* 2011) © 2011 IEEE; (e) smartphone augmented reality with real-time depth occlusion effects (Valentin, Kowdle *et al.* 2018) © 2018 ACM; (f) 3D map computed in real-time on a fully autonomous Skydio R1 drone (Cross 2019).

Another major trend was the dramatic increase in computational power available from the development of general purpose (data-parallel) algorithms on graphical processing units (GPGPU). The breakthrough SuperVision ("AlexNet") deep neural network (Figure 1.11a; Krizhevsky, Sutskever, and Hinton 2012), which was the first neural network to win the yearly ImageNet large-scale visual recognition challenge, relied on GPU training, as well as a number of technical advances, for its dramatic performance. After the publication of this paper, progress in using deep convolutional architectures accelerated dramatically, to the point where they are now the only architecture considered for recognition and semantic segmentation tasks (Figure 1.11b), as well as the preferred architecture for many other vision tasks (Chapter 5; LeCun, Bengio, and Hinton 2015), including optical flow (Sun, Yang *et al.* 2018)), denoising, and monocular depth inference (Li, Dekel *et al.* 2019).

Large datasets and GPU architectures, coupled with the rapid dissemination of ideas through timely publications on arXiv as well as the development of languages for deep learning and the open sourcing of neural network models, all contributed to an explosive growth in this area, both in rapid advances and capabilities, and also in the sheer number of publications and researchers now working on these topics. They also enabled the extension of image recognition approaches to video understanding tasks such as action recognition (Feichtenhofer, Fan *et al.* 2019), as well as structured regression tasks such as real-time multi-person body pose estimation (Cao, Simon *et al.* 2017).

Specialized sensors and hardware for computer vision tasks also continued to advance. The

Microsoft Kinect depth camera, released in 2010, quickly became an essential component of many 3D modeling (Figure 1.11d) and person tracking (Shotton, Fitzgibbon *et al.* 2011) systems. Over the decade, 3D body shape modeling and tracking systems continued to evolve, to the point where it is now possible to infer a person's 3D model with gestures and expression from a single image (Figure 1.11c).

And while depth sensors have not yet become ubiquitous (except for security applications on high-end phones), computational photography algorithms run on all of today's smartphones. Innovations introduced in the computer vision community, such as panoramic image stitching and bracketed high dynamic range image merging, are now standard features, and multi-image low-light denoising algorithms are also becoming commonplace (Liba, Murthy *et al.* 2019). Lightfield imaging algorithms, which allow the creation of soft depth-of-field effects, are now also becoming more available (Garg, Wadhwa *et al.* 2019). Finally, mobile augmented reality applications that perform real-time pose estimation and environment augmentation using combinations of feature tracking and inertial measurements are commonplace, and are currently being extended to include pixel-accurate depth occlusion effects (Figure 1.11e).

On higher-end platforms such as autonomous vehicles and drones, powerful real-time SLAM (simultaneous localization and mapping) and VIO (visual inertial odometry) algorithms (Engel, Schöps, and Cremers 2014; Forster, Zhang *et al.* 2017; Engel, Koltun, and Cremers 2018) can build accurate 3D maps that enable, e.g., autonomous flight through challenging scenes such as forests (Figure 1.11f).

In summary, this past decade has seen incredible advances in the performance and reliability of computer vision algorithms, brought in part by the shift to machine learning and training on very large sets of real-world data. It has also seen the application of vision algorithms in myriad commercial and consumer scenarios as well as new challenges engendered by their widespread use (Su and Crandall 2021).

1.3 Book overview

In the final part of this introduction, I give a brief tour of the material in this book, as well as a few notes on notation and some additional general references. Since computer vision is such a broad field, it is possible to study certain aspects of it, e.g., geometric image formation and 3D structure recovery, without requiring other parts, e.g., the modeling of reflectance and shading. Some of the chapters in this book are only loosely coupled with others, and it is not strictly necessary to read all of the material in sequence.

Figure 1.12 shows a rough layout of the contents of this book. Since computer vision involves going from images to both a semantic understanding as well as a 3D structural description of the scene, I have positioned the chapters horizontally in terms of where in this spectrum they land, in addition to vertically according to their dependence.[9]

Interspersed throughout the book are sample **applications**, which relate the algorithms and mathematical material being presented in various chapters to useful, real-world applications. Many of these applications are also presented in the exercises sections, so that students can write their own.

At the end of each section, I provide a set of **exercises** that the students can use to implement, test, and refine the algorithms and techniques presented in each section. Some of the exercises are suitable as written homework assignments, others as shorter one-week projects, and still others as

[9] For an interesting comparison with what is known about the human visual system, e.g., the largely parallel *what* and *where* pathways (Goodale and Milner 1992), see some textbooks on human perception (Palmer 1999; Livingstone 2008; Frisby and Stone 2010).

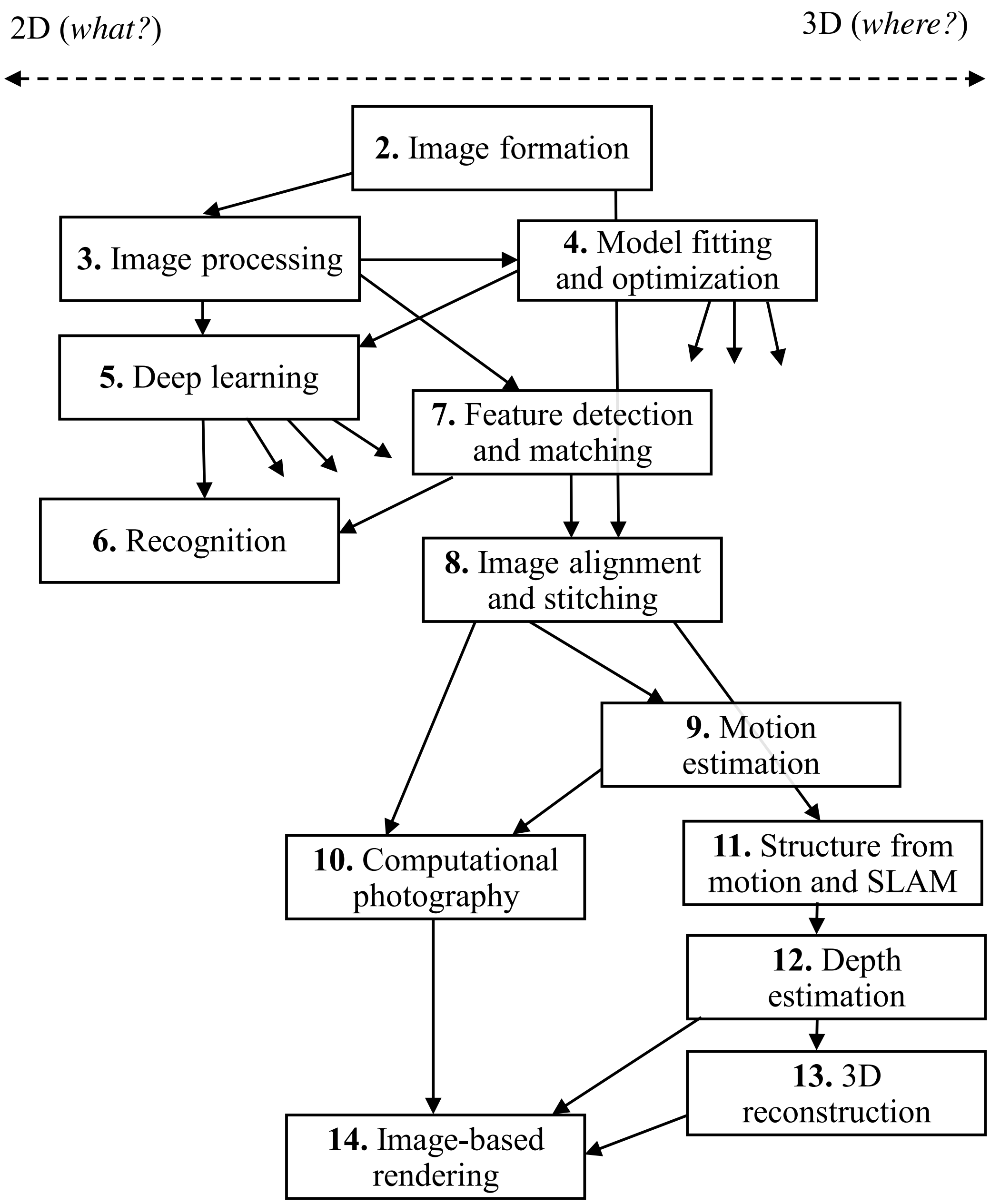

Figure 1.12 A taxonomy of the topics covered in this book, showing the (rough) dependencies between different chapters, which are roughly positioned along the left–right axis depending on whether they are more closely related to images (left) or 3D geometry (right) representations. The "what-where" along the top axis is a reference to separate visual pathways in the visual system (Goodale and Milner 1992), but should not be taken too seriously. Foundational techniques such as optimization and deep learning are widely used in subsequent chapters.

open-ended research problems that make for challenging final projects. Motivated students who implement a reasonable subset of these exercises will, by the end of the book, have a computer vision software library that can be used for a variety of interesting tasks and projects.

If the students or curriculum do not have a strong preference for programming languages, Python, with the NumPy scientific and array arithmetic library plus the OpenCV vision library, are a good environment to develop algorithms and learn about vision. Not only will the students learn how to program using array/tensor notation and linear/matrix algebra (which is a good foundation for later use of PyTorch for deep learning), you can also prepare classroom assignments using Jupyter notebooks, giving you the option to combine descriptive tutorials, sample code, and code to be extended/modified in one convenient location.[10]

As this is a reference book, I try wherever possible to discuss which techniques and algorithms work well in practice, as well as provide up-to-date pointers to the latest research results in the areas that I cover. The exercises can be used to build up your own personal library of self-tested and validated vision algorithms, which is more worthwhile in the long term (assuming you have the time) than simply pulling algorithms out of a library whose performance you do not really understand.

The book begins in Chapter 2 with a review of the image formation processes that create the images that we see and capture. Understanding this process is fundamental if you want to take a scientific (model-based) approach to computer vision. Students who are eager to just start implementing algorithms (or courses that have limited time) can skip ahead to the next chapter and dip into this material later. In Chapter 2, we break down image formation into three major components. Geometric image formation (Section 2.1) deals with points, lines, and planes, and how these are mapped onto images using *projective geometry* and other models (including radial lens distortion). Photometric image formation (Section 2.2) covers *radiometry*, which describes how light interacts with surfaces in the world, and *optics*, which projects light onto the sensor plane. Finally, Section 2.3 covers how sensors work, including topics such as sampling and aliasing, color sensing, and in-camera compression.

Chapter 3 covers image processing, which is needed in almost all computer vision applications. This includes topics such as linear and non-linear filtering (Section 3.3), the Fourier transform (Section 3.4), image pyramids and wavelets (Section 3.5), and geometric transformations such as image warping (Section 3.6). Chapter 3 also presents applications such as seamless image blending and image morphing.

Chapter 4 begins with a new section on data fitting and interpolation, which provides a conceptual framework for global optimization techniques such as *regularization* and *Markov random fields* (MRFs), as well as *machine learning*, which we cover in the next chapter. Section 4.2 covers classic regularization techniques, i.e., piecewise-continuous smoothing splines (aka *variational techniques*) implemented using fast iterated linear system solvers, which are still often the method of choice in time-critical applications such as mobile augmented reality. The next section (4.3) presents the related topic of *MRFs*, which also serve as an introduction to Bayesian inference techniques, covered at a more abstract level in Appendix B. The chapter also discusses applications to interactive colorization and segmentation.

Chapter 5 is a completely new chapter covering machine learning, deep learning, and deep neural networks. It begins in Section 5.1 with a review of classic *supervised machine learning* approaches, which are designed to classify images (or regress values) based on intermediate-level features. Section 5.2 looks at *unsupervised learning*, which is useful for both understanding unlabeled training data and providing models of real-world distributions. Section 5.3 presents the basic elements of

[10] You may also be able to run your notebooks and train your models using the Google Colab service at https://colab.research.google.com.

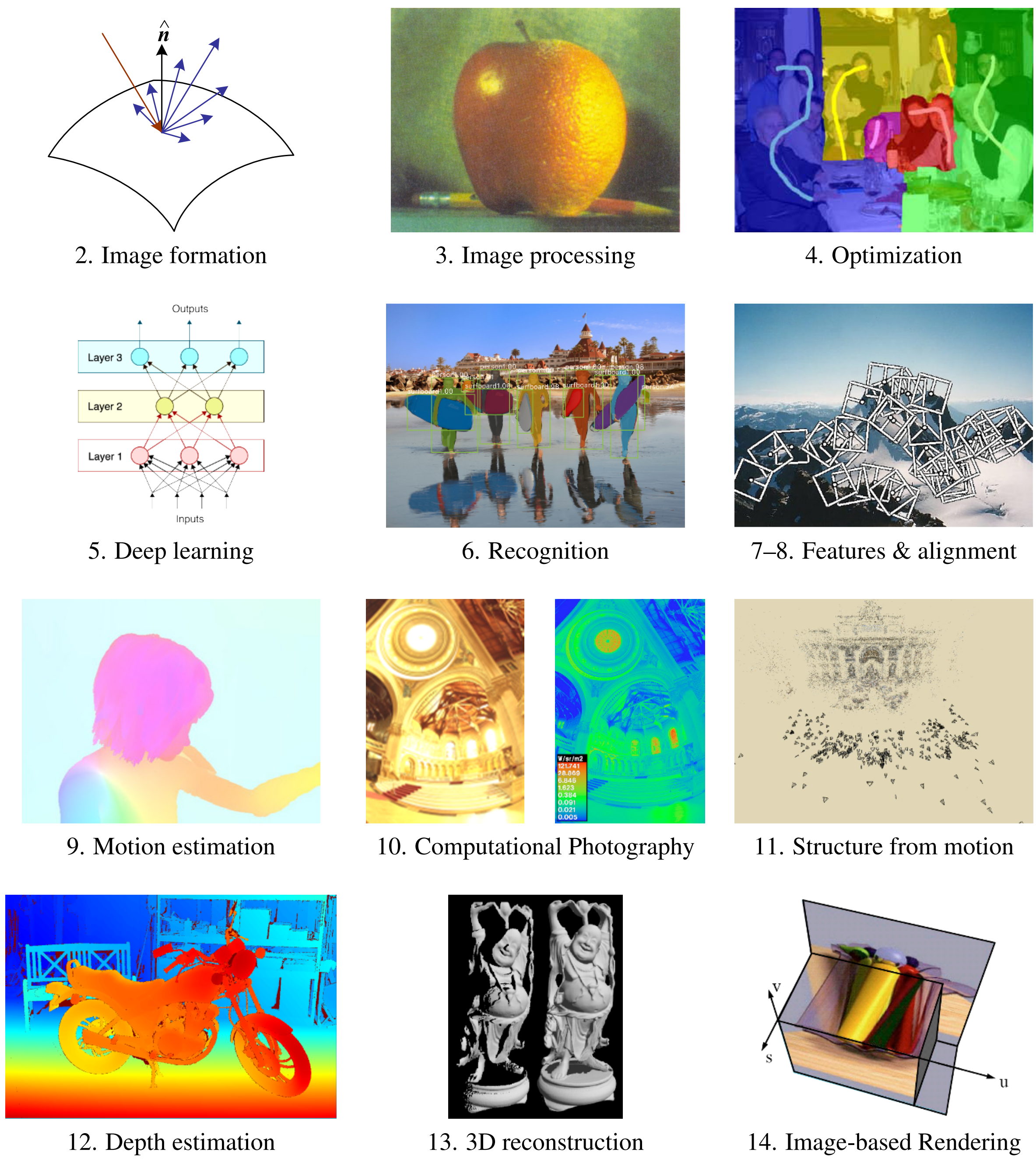

Figure 1.13 A pictorial summary of the chapter contents. Sources: Burt and Adelson (1983b); Agarwala, Dontcheva *et al.* (2004); Glassner (2018); He, Gkioxari *et al.* (2017); Brown, Szeliski, and Winder (2005); Butler, Wulff *et al.* (2012); Debevec and Malik (1997); Snavely, Seitz, and Szeliski (2006); Scharstein, Hirschmüller *et al.* (2014); Curless and Levoy (1996); Gortler, Grzeszczuk *et al.* (1996)—see the figures in the respective chapters for copyright information.

feedforward neural networks, including weights, layers, and activation functions, as well as methods for network training. Section 5.4 goes into more detail on convolutional networks and their applications to both recognition and image processing. The last section in the chapter discusses more complex networks, including 3D, spatio-temporal, recurrent, and generative networks.

Chapter 6 covers the topic of *recognition*. In the first edition of this book this chapter came last, since it built upon earlier methods such as segmentation and feature matching. With the advent of deep networks, many of these intermediate representations are no longer necessary, since the network can learn them as part of the training process. As so much of computer vision research is now devoted to various recognition topics, I decided to move this chapter up so that students can learn about it earlier in the course.

The chapter begins with the classic problem of *instance recognition*, i.e., finding instances of known 3D objects in cluttered scenes. Section 6.2 covers both traditional and deep network approaches to whole *image classification*, i.e., what used to be called *category recognition*. It also discusses the special case of facial recognition. Section 6.3 presents algorithms for *object detection* (drawing bounding boxes around recognized objects), with a brief review of older approaches to face and pedestrian detection. Section 6.4 covers various flavors of *semantic segmentation* (generating per-pixel labels), including *instance segmentation* (delineating separate objects), *pose estimation* (labeling pixels with body parts), and *panoptic segmentation* (labeling both things and stuff). In Section 6.5, we briefly look at some recent papers in *video understanding* and *action recognition*, while in Section 6.6 we mention some recent work in image captioning and visual question answering.

In Chapter 7, we cover feature detection and matching. A lot of current 3D reconstruction and recognition techniques are built on extracting and matching *feature points* (Section 7.1), so this is a fundamental technique required by many subsequent chapters (Chapters 8 and 11) and even in instance recognition (Section 6.1). We also cover edge and straight line detection in Sections 7.2 and 7.4, contour tracking in Section 7.3, and low-level segmentation techniques in Section 7.5.

Feature detection and matching are used in Chapter 8 to perform *image alignment* (or *registration*) and *image stitching*. We introduce the basic techniques of feature-based alignment and show how this problem can be solved using either linear or non-linear least squares, depending on the motion involved. We also introduce additional concepts, such as uncertainty weighting and robust regression, which are essential to making real-world systems work. Feature-based alignment is then used as a building block for both 2D applications such as image stitching (Section 8.2) and computational photography (Chapter 10), as well as 3D geometric alignment tasks such as pose estimation and structure from motion (Chapter 11).

The second part of Chapter 8 is devoted to *image stitching*, i.e., the construction of large panoramas and composites. While stitching is just one example of *computational photography* (see Chapter 10), there is enough depth here to warrant a separate section. We start by discussing various possible motion models (Section 8.2.1), including planar motion and pure camera rotation. We then discuss global alignment (Section 8.3), which is a special (simplified) case of general bundle adjustment, and then present *panorama recognition*, i.e., techniques for automatically discovering which images actually form overlapping panoramas. Finally, we cover the topics of *image compositing* and *blending* (Section 8.4), which involve both selecting which pixels from which images to use and blending them together so as to disguise exposure differences.

Image stitching is a wonderful application that ties together most of the material covered in earlier parts of this book. It also makes for a good mid-term course project that can build on previously developed techniques such as image warping and feature detection and matching. Sections 8.2–8.4 also present more specialized variants of stitching such as whiteboard and document scanning, video summarization, *panography*, full 360° spherical panoramas, and interactive photomontage for blending repeated action shots together.

In Chapter 9, we generalize the concept of feature-based image alignment to cover dense intensity-based motion estimation, i.e., *optical flow*. We start with the simplest possible motion models, translational motion (Section 9.1), and cover topics such as hierarchical (coarse-to-fine) motion estimation, Fourier-based techniques, and iterative refinement. We then present parametric motion models, which can be used to compensate for camera rotation and zooming, as well as affine or planar perspective motion (Section 9.2). This is then generalized to spline-based motion models (Section 9.2.2) and finally to general per-pixel optical flow (Section 9.3). We close the chapter in Section 9.4 with a discussion of layered and learned motion models as well as video object segmentation and tracking. Applications of motion estimation techniques include automated morphing, video denoising, and frame interpolation (slow motion).

Chapter 10 presents additional examples of *computational photography*, which is the process of creating new images from one or more input photographs, often based on the careful modeling and calibration of the image formation process (Section 10.1). Computational photography techniques include merging multiple exposures to create *high dynamic range* images (Section 10.2), increasing image resolution through blur removal and *super-resolution* (Section 10.3), and image editing and compositing operations (Section 10.4). We also cover the topics of texture analysis, synthesis, and *inpainting* (hole filling) in Section 10.5, as well as non-photorealistic rendering and style transfer.

Starting in Chapter 11, we delve more deeply into techniques for reconstructing 3D models from images. We begin by introducing methods for *intrinsic* camera calibration in Section 11.1 and *3D pose estimation*, i.e., *extrinsic* calibration, in Section 11.2. These sections also describe the applications of single-view reconstruction of building models and 3D *location recognition*. We then cover the topic of *triangulation* (Section 11.2.4), which is the 3D reconstruction of points from matched features when the camera positions are known.

Chapter 11 then moves on to the topic of *structure from motion*, which involves the simultaneous recovery of 3D camera motion and 3D scene structure from a collection of tracked 2D features. We begin with two-frame structure from motion (Section 11.3), for which algebraic techniques exist, as well as robust sampling techniques such as RANSAC that can discount erroneous feature matches. We then cover techniques for multi-frame structure from motion, including factorization (Section 11.4.1), bundle adjustment (Section 11.4.2), and constrained motion and structure models (Section 11.4.8). We present applications in visual effects (*match move*) and sparse 3D model construction for large (e.g., internet) photo collections. The final part of this chapter (Section 11.5) has a new section on *simultaneous localization and mapping* (SLAM) as well as its applications to autonomous navigation and mobile augmented reality (AR).

In Chapter 12, we turn to the topic of stereo correspondence, which can be thought of as a special case of motion estimation where the camera positions are already known (Section 12.1). This additional knowledge enables stereo algorithms to search over a much smaller space of correspondences to produce dense depth estimates using various combinations of matching criteria, optimization algorithm, and/or deep networks (Sections 12.3–12.6). We also cover *multi-view* stereo algorithms that build a true 3D surface representation instead of just a single depth map (Section 12.7), as well as *monocular depth inference* algorithms that hallucinate depth maps from just a single image (Section 12.8). Applications of stereo matching include head and gaze tracking, as well as depth-based background replacement (*Z-keying*).

Chapter 13 covers additional 3D shape and appearance modeling techniques. These include classic *shape-from-X* techniques such as shape from shading, shape from texture, and shape from focus (Section 13.1). An alternative to all of these *passive* computer vision techniques is to use *active rangefinding* (Section 13.2), i.e., to project patterned light onto scenes and recover the 3D geometry through triangulation. Processing all of these 3D representations often involves interpolating or simplifying the geometry (Section 13.3), or using alternative representations such as surface point sets (Section 13.4) or implicit functions (Section 13.5).

The collection of techniques for going from one or more images to partial or full 3D models is often called *image-based modeling* or *3D photography*. Section 13.6 examines three more specialized application areas (architecture, faces, and human bodies), which can use *model-based reconstruction* to fit parameterized models to the sensed data. Section 13.7 examines the topic of *appearance modeling*, i.e., techniques for estimating the texture maps, albedos, or even sometimes complete *bi-directional reflectance distribution functions* (BRDFs) that describe the appearance of 3D surfaces.

In Chapter 14, we discuss the large number of image-based rendering techniques that have been developed in the last three decades, including simpler techniques such as view interpolation (Section 14.1), layered depth images (Section 14.2), and sprites and layers (Section 14.2.1), as well as the more general framework of light fields and Lumigraphs (Section 14.3) and higher-order fields such as environment mattes (Section 14.4). Applications of these techniques include navigating 3D collections of photographs using *photo tourism*.

Next, we discuss video-based rendering, which is the temporal extension of image-based rendering. The topics we cover include video-based animation (Section 14.5.1), periodic video turned into *video textures* (Section 14.5.2), and 3D video constructed from multiple video streams (Section 14.5.4). Applications of these techniques include animating still images and creating home tours based on 360° video. We finish the chapter with an overview of the new emerging field of *neural rendering*.

To support the book's use as a textbook, the appendices and associated website contain more detailed mathematical topics and additional material. Appendix A covers linear algebra and numerical techniques, including matrix algebra, least squares, and iterative techniques. Appendix B covers Bayesian estimation theory, including maximum likelihood estimation, robust statistics, Markov random fields, and uncertainty modeling. Appendix C describes the supplementary material that can be used to complement this book, including images and datasets, pointers to software, and course slides.

1.4 Sample syllabus

Teaching all of the material covered in this book in a single quarter or semester course is a Herculean task and likely one not worth attempting.[11] It is better to simply pick and choose topics related to the lecturer's preferred emphasis and tailored to the set of mini-projects envisioned for the students.

Steve Seitz and I have successfully used a 10-week syllabus similar to the one shown in Table 1.1 as both an undergraduate and a graduate-level course in computer vision. The undergraduate course[12] tends to go lighter on the mathematics and takes more time reviewing basics, while the graduate-level course[13] dives more deeply into techniques and assumes the students already have a decent grounding in either vision or related mathematical techniques. Related courses have also been taught on the topics of 3D photography and computational photography. Appendix C.3 and the book's website list other courses that use this book to teach a similar curriculum.

[11] Some universities, such as Stanford (CS231A & 231N), Berkeley (CS194-26/294-26 & 280), and the University of Michigan (EECS 498/598 & 442), now split the material over two courses.

[12] http://www.cs.washington.edu/education/courses/455

[13] http://www.cs.washington.edu/education/courses/576

Week	Chapter	Topics
1.	Chapters 1–2	Introduction and image formation
2.	Chapter 3	Image processing
3.	Chapters 4–5	Optimization and learning
4.	Chapter 5	Deep learning
5.	Chapter 6	Recognition
6.	Chapter 7	Feature detection and matching
7.	Chapter 8	Image alignment and stitching
8.	Chapter 9	Motion estimation
9.	Chapter 10	Computational photography
10.	Chapter 11	Structure from motion
11.	Chapter 12	Depth estimation
12.	Chapter 13	3D reconstruction
13.	Chapter 14	Image-based rendering

Table 1.1 Sample syllabus for a one semester 13-week course. A 10-week quarter could go into lesser depth or omit some topics.

When Steve and I teach the course, we prefer to give the students several small programming assignments early in the course rather than focusing on written homework or quizzes. With a suitable choice of topics, it is possible for these projects to build on each other. For example, introducing feature matching early on can be used in a second assignment to do image alignment and stitching. Alternatively, direct (optical flow) techniques can be used to do the alignment and more focus can be put on either graph cut seam selection or multi-resolution blending techniques.

In the past, we have also asked the students to propose a final project (we provide a set of suggested topics for those who need ideas) by the middle of the course and reserved the last week of the class for student presentations. Sometimes, a few of these projects have actually turned into conference submissions!

No matter how you decide to structure the course or how you choose to use this book, I encourage you to try at least a few small programming tasks to get a feel for how vision techniques work and how they fail. Better yet, pick topics that are fun and can be used on your own photographs, and try to push your creative boundaries to come up with surprising results.

1.5 A note on notation

For better or worse, the notation found in computer vision and multi-view geometry textbooks tends to vary all over the map (Faugeras 1993; Hartley and Zisserman 2004; Girod, Greiner, and Niemann 2000; Faugeras and Luong 2001; Forsyth and Ponce 2003). In this book, I use the convention I first learned in my high school physics class (and later multi-variate calculus and computer graphics courses), which is that vectors $\mathbf{v}$ are lower case bold, matrices $\mathbf{M}$ are upper case bold, and scalars (T, s) are mixed case italic. Unless otherwise noted, vectors operate as column vectors, i.e., they post-multiply matrices, $\mathbf{Mv}$, although they are sometimes written as comma-separated parenthesized lists $\mathbf{x} = (x, y)$ instead of bracketed column vectors $\mathbf{x} = [x\ y]^T$. Some commonly used matrices are $\mathbf{R}$ for rotations, $\mathbf{K}$ for calibration matrices, and $\mathbf{I}$ for the identity matrix. Homogeneous coordinates (Section 2.1) are denoted with a tilde over the vector, e.g., $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\mathbf{x}}$ in $\mathcal{P}^2$. The cross product operator in matrix form is denoted by $[\,]_\times$.

1.6 Additional reading

This book attempts to be self-contained, so that students can implement the basic assignments and algorithms described here without the need for outside references. However, it does presuppose a general familiarity with basic concepts in linear algebra and numerical techniques, which are reviewed in Appendix A, and image processing, which is reviewed in Chapter 3.

Students who want to delve more deeply into these topics can look in Golub and Van Loan (1996) for matrix algebra and Strang (1988) for linear algebra. In image processing, there are a number of popular textbooks, including Crane (1997), Gomes and Velho (1997), Jähne (1997), Pratt (2007), Russ (2007), Burger and Burge (2008), and Gonzalez and Woods (2017). For computer graphics, popular texts include Hughes, van Dam *et al.* (2013) and Marschner and Shirley (2015), with Glassner (1995) providing a more in-depth look at image formation and rendering. For statistics and machine learning, Chris Bishop's (2006) book is a wonderful and comprehensive introduction with a wealth of exercises, while Murphy (2012) provides a more recent take on the field and Hastie, Tibshirani, and Friedman (2009) a more classic treatment. A great introductory text to deep learning is Glassner (2018), while Goodfellow, Bengio, and Courville (2016) and Zhang, Lipton *et al.* (2021) provide more comprehensive treatments. Students may also want to look in other textbooks on computer vision for material that we do not cover here, as well as for additional project ideas (Nalwa 1993; Trucco and Verri 1998; Hartley and Zisserman 2004; Forsyth and Ponce 2011; Prince 2012; Davies 2017).

There is, however, no substitute for reading the latest research literature, both for the latest ideas and techniques and for the most up-to-date references to related literature.[14] In this book, I have attempted to cite the most recent work in each field so that students can read them directly and use them as inspiration for their own work. Browsing the last few years' conference proceedings from the major vision, graphics, and machine learning conferences, such as CVPR, ECCV, ICCV, SIGGRAPH, and NeurIPS, as well as keeping an eye out for the latest publications on arXiv, will provide a wealth of new ideas. The tutorials offered at these conferences, for which slides or notes are often available online, are also an invaluable resource.

[14]For a comprehensive bibliography and taxonomy of computer vision research, Keith Price's Annotated Computer Vision Bibliography https://www.visionbib.com/bibliography/contents.html is an invaluable resource.