Gary Bradski
and Adrian Kaehler

Robot-Vision Signal Processing Primitives

Recent advances in vision algorithms and increases in computer performance have made new capabilities available in autonomous robotics for real-time applications. In general, computer-vision-based solutions to robotics problems employ a high-level system architecture that makes use of low-level processing blocks. While the high-level system architecture is still an active area of research, many of the underlying low-level processing blocks and their associated methods have begun to stabilize, yielding a set of operators that has been found useful in a wide variety of tasks.

In this article, we will briefly review these operators, which we call robot-vision signal-processing (SP) primitives, and their associated classes of methods. Although our taxonomy is quite general and related to those used in image processing and pattern recognition, we focus on the specific use of these primitives and associated classes in robot vision for SP purposes before presenting a robot-vision example—the Stanley robot racing car.

SP PRIMITIVE CLASSES

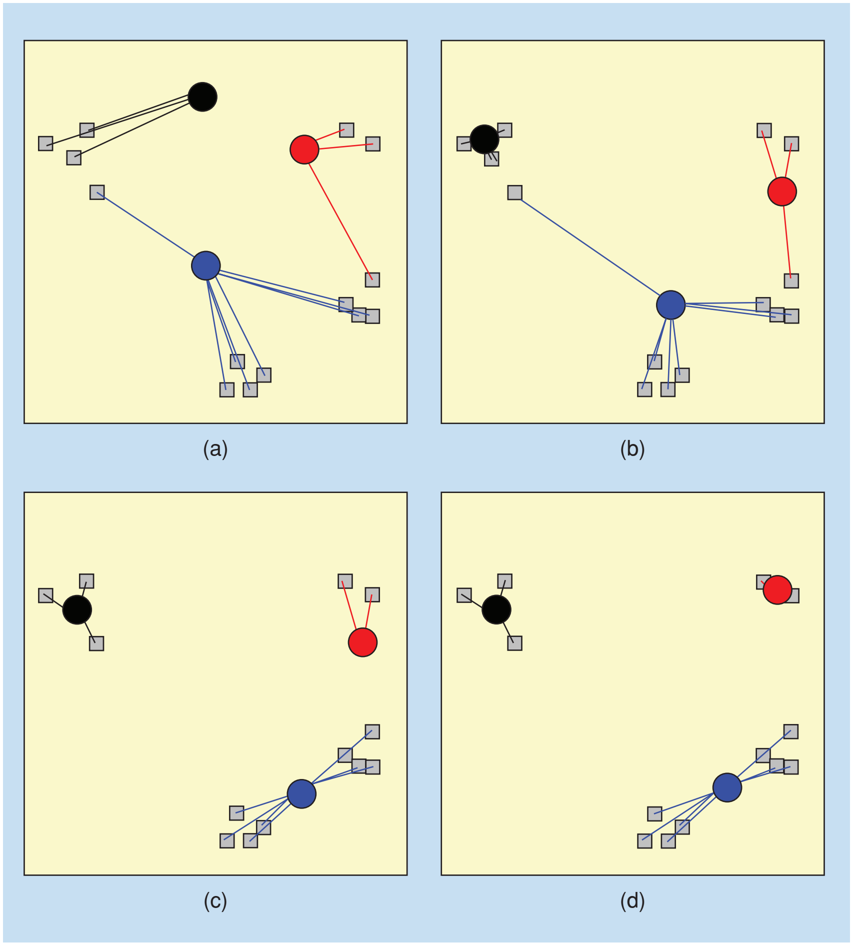
FILTERING

Filtering methods and primitives have been discussed extensively in numerous textbooks [1]. Similar to other application domains, filtering methods in robot vision aim at helping subsequent processing stages, such as edge detection and tracking, to focus on objects of interest. Data that is deemed less important is filtered from the image using convolution in the spatial or frequency domains, morphology, or other ideas. From the variety of filtering methods available, examples of popular choices for robot vision are smoothing using Gaussian pyramids, patch matching via cross correlation, and despeckling methods.

SHAPE ANALYSIS

Shape analysis methods, which are also well-known and part of numerous applications, aim at identifying image entities for further processing. Shape analysis methods can be boundary-based (in which case they identify the edges or contours of the object) or region-based (in which case they identify the area covered by the object) [2], [3]. From the numerous shape analysis methods available, most of the popular shape analysis methods in image processing and pattern recognition are also used in robot vision.

Digital Object Identifier 10.1109/MSP.2007.910427



[FIG1] K-means algorithm (a) starts with assigned cluster centers and finds which data points are locally closest; (b) moves the cluster centers to the locally closest centroids; (c) finds the new closest data points; and (d) moves to the now stable centroid of each cluster center.

1053-5888/08/$25.00©2008IEEE

DENSITY MODELING

Density modeling methods model a distribution of image- or subimage-related features such as color, pixel location, etc. There are numerous such methods [4], simple histogram binning often being used in robot vision applications.

Histogram binning is performed by simply placing a grid over the data space and counting how many data points fall into each grid bin. In a color image in the RGB space, this may mean collecting the normalized red and green color values, given by $R/(R+G+B)$ and $G/(R+G+B)$. In low dimensions this can be a very effective technique and is sometimes sufficient to allow object localization.

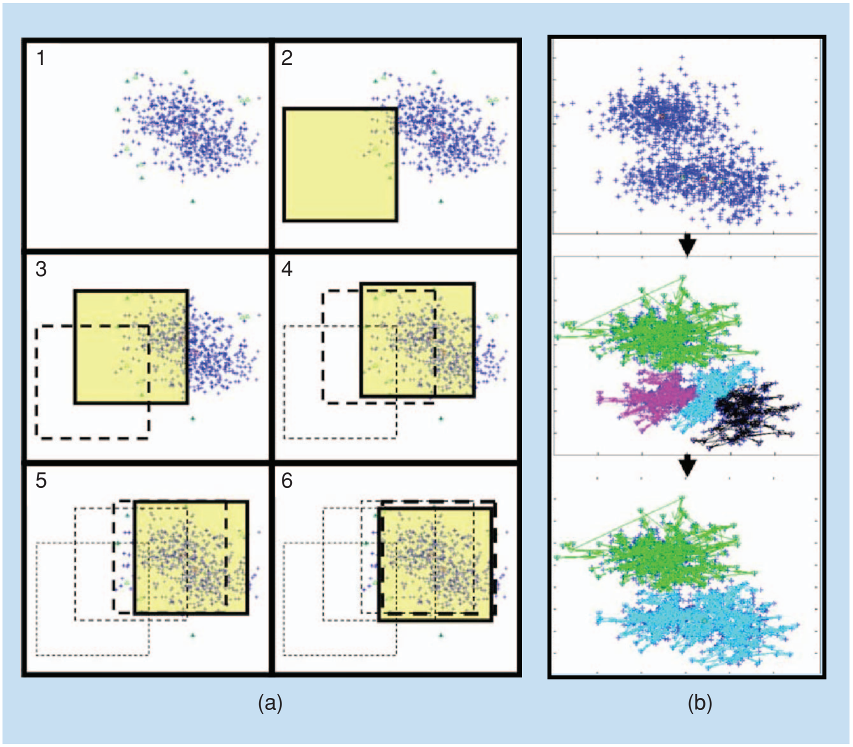
However, in multidimensional spaces the memory and processing time requirements of histogram binning grow exponentially with the number of feature types being modeled. To address these limitations, the histogram is sometimes coded as a binary tree. The existing bins in a histogram can then be stored in a balanced binary tree (using, for instance, the red-black algorithm [5]) that branches on bin number (so that the look-up time is only $\log_2$ bins) and only bins with data are actually stored.

Once the histogram has been obtained, the distribution of objects given the measures employed can be computed [6]. Let O be the object to recognize and M the measure used for recognition. Assume that the number of objects to recognize is N and the number of measures used is K. With many examples of an object O_n, we use a measure k and construct a histogram of the measures given the object $H(M_k|O_n)$ using histogram binning. For recognition purposes, we would like to compute the probability of the object given the measures $P(O_n|M_k)$, which is given by a Bayesian decision

$$P(O_n|M_k) = \frac{P(M_k|O_n)P(O_n)}{\sum_n P(M_k|O_n)P(O_n)},$$

where $P(M_k|O_n)$ is the probability of the measures given the object. Using local feature detectors, as long as these don't overlap, we can assume statistical independence to yield

$$P(O_n|M_1, M_2 \dots M_K) = \frac{\prod_k p(M_k|O_n)P(O_n)}{\sum_n \prod_k p(M_k|O_n)P(O_n)}.$$



[FIG2] (a) Mean-shift algorithm example: 1) 2-D data distribution, 2) window, and 3)–6) the mode of the distribution; (b) Mean-shift agglomerative clustering, modeling from zero to four to two clusters.

CLUSTERING

Two of the most often used clustering algorithms in robot vision are K-means [7] and mean-shift [8].

The K-means algorithm starts with k clusters. As shown in Figure 1, the algorithm performs iteratively the following steps [9]:

```
DO
    1) Compute the centroid of each cluster.
    2) Associate each data point with the closest centroid to form a new cluster.
    3) Go to Step 1.
UNTIL the centroid moves less than a threshold.
```

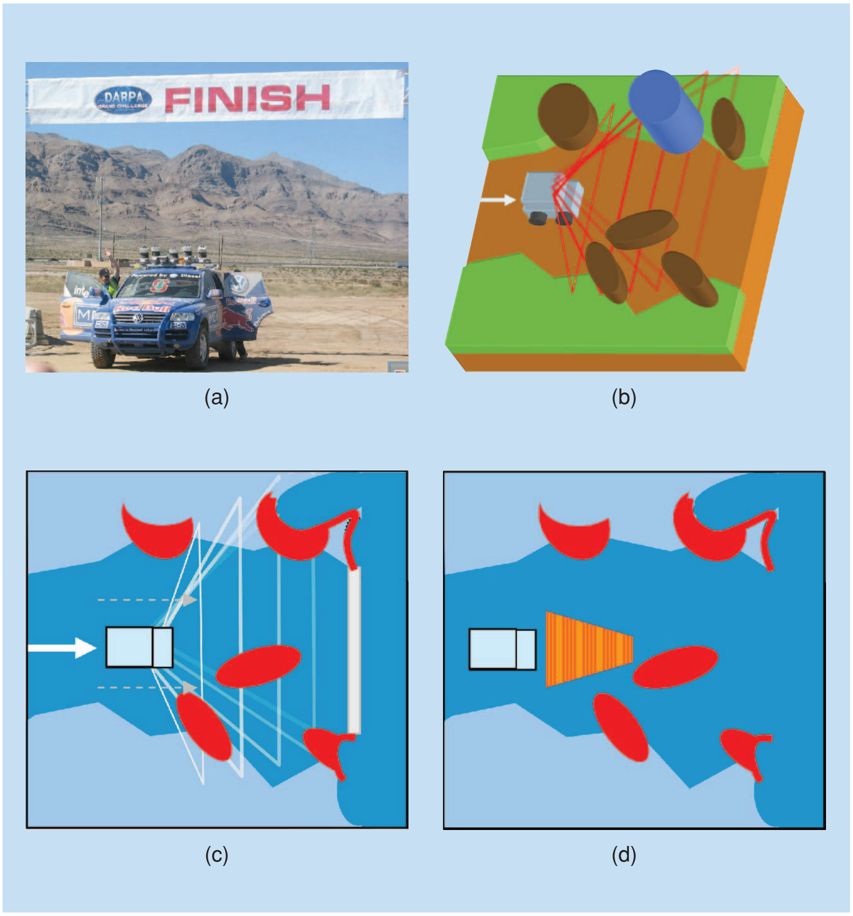
The mean-shift algorithm finds the mode (peak) of a distribution by performing the following steps [8], [10]:

```
DO
    1) Select a search window size.
    2) Select the initial location of the search window.
    3) Compute the mean location in the search window.
    4) Center the search window at the mean location found in Step 3.
    5) Repeat Steps 3 and 4.
UNTIL the mean location moves less than a preset threshold.
```

An example of finding the mode of a given set of two-dimensional (2-D) data using this method is illustrated in Figure 2(a).

The mean-shift algorithm can also be used to separate out different parts (modes) of a data distribution as follows [10]:

```
DO
    1) Place a zero-radius mean-shift window over each data point.
    2) Enlarge the mean-shift window radius until the points begin to
```



[FIG3] **(a) Stanley winning the DARPA Grand Challenge robot race across the desert. (b) Laser range finders integrating motion from the car create a 3-D model of the terrain surrounding the vehicle. (c) This is converted to a top-down view obstacle map in the center where red areas need to be avoided, dark areas are drivable, and lighter areas are unknown. (d) A representative patch of drivable terrain is selected and passed on to the vision system to extend out into the scene.**

substantially move. Record this as a node point.

3) Merge windows that overlap and record where they came from. All the data points that were "owned" by the merging windows now belong to the merged window.

4) Go to Step 2.

UNTIL only one window is left.

The branching nodes now form a cluster tree, which also tracks what points belonged to what node along the way. This forms an agglomerative clustering/model of the data.

An example of the agglomerative clustering of data using the mean-shift algorithm is shown in Figure 2(b).

TRACKING

Tracking methods are widely used in computer vision and robot vision applications. A method popular in the latter applications is that based on the Harris corner detector. Let I_x, I_y denote partial derivatives of the image intensity I with respect to x and y, respectively. Assume that the following expression holds:

$$\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = AA^T,$$

where $A = [I_x\, I_y]^T$ is the image gradient.

A Harris corner can be thought of as a point that has high local curvature in all directions. Therefore, if the eigenvalues of AA^T are both large in magnitude, then we define a corner point. Further, if the brightness of a small patch (dx, dy) remains constant over time, then

$$\frac{dI(x(t), y(t), t)}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0.$$

The popular Lucas and Kanade tracking algorithm [11] takes advantage of this to define the tracking equation given by $(AA^T)v = -AI_t$, where v is the velocity and I_t is the temporal derivative of the image intensity. This equation is valid for small motions. Tracking can be further refined by embedding the tracker in a Kalman or particle filter.

APPLICATION: STANLEY

Stanley, shown in Figure 3(a), was Stanford University's robot racing car that won the 2005 DARPA Grand Challenge race [12] across the desert, running the 132-mi-long race completely autonomously in 6 h, 53 min, averaging 19.1 mi/h and winning the US$2M first-place prize as well as a place in robot history.

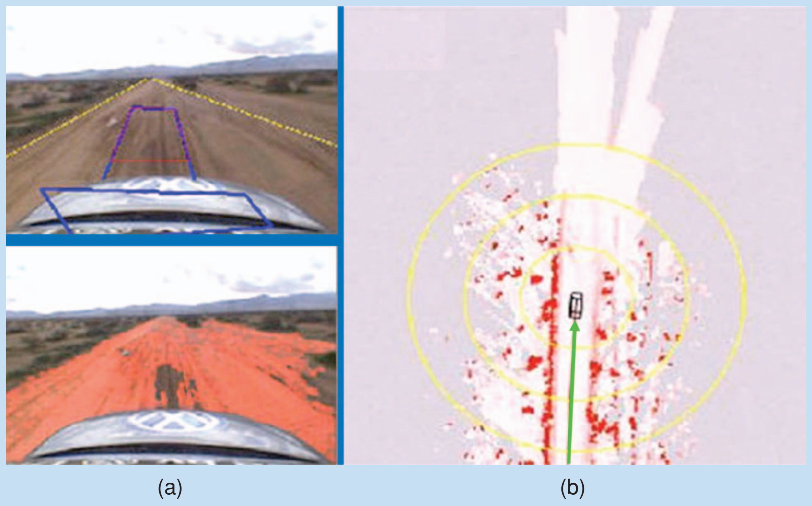
The simple distribution modeling mentioned earlier, followed by clustering, played an essential part in Stanley's victory. It enabled the car to use cameras to see out beyond the laser range finder range, allowing the robot to safely drive faster than laser range finders alone allowed.

More specifically, as shown in Figure 3(b)–(d), the laser range finder, integrating motion from the car, makes a three-dimensional (3-D) model of the terrain immediately surrounding the car. A patch of drivable terrain on the obstacle map is then passed to the vision system. In effect, the laser range finding system indicates to the vision system that there is a patch of good terrain and asks it to find other such patches.

The patch of good terrain from the laser range finding system is converted into the coordinate system of the cameras (the two systems are calibrated together prior to the race with the assumption that the robot is rolling on a flat plane). This is shown at upper-left portion of Figure 4(a). The distribution of colors in the drivable patch is then modeled using the *k*-means algorithm. A Gaussian color model is fit to each of the *k* color centers found. Then these Gaussians are normalized to obtain the probability of a pixel being part of a road object given the measures in a Bayesian decision model as described earlier.

Once the vision system assigns pixels in the scene as drivable or not, the camera coordinate system is again transformed into a top-down bird's eye view map. The vision and the laser maps are fused together into a near and far drivability map, as illustrated in Figure 4(b). This drivability map is then passed to the planner to make path and speed decisions.

The laser range finder map was sufficient for the robot car to drive at 25 mi/h and still stop within range (20 m) of what the laser could reliably see. With the vision system, the robot car could often drive much faster. Seeing out to 40 m allowed for a 35 mi/h speed; seeing further allowed much faster speeds, but course and race strategy limited Stanley to a maximum of 38 mi/h in the competition. When the vision system detected obstacles ahead, it slowed down to laser range finder speed of 25 mi/h. When the vision system found long-range drivable terrain, the car sped up to 35 mi/h and beyond. By so doing, Stanley was able to finish the race in the time mentioned earlier, showing that the extra speed from the robot vision system was crucial to Stanley's win.



[FIG4] **Laser range finder in a robot vision system. (a) At the upper left, the blue rectangle in front of the car delineates drivable terrain as judged from the laser range finders. The yellow lines show the actual road boundaries from postprocessing. At the lower left, the vision system has segmented other drivable areas. (b) These views are merged into a drivability map which is passed to the planning system. The yellow circles show the laser range.**

CONCLUSIONS

We briefly discussed a set of SP primitives that are used in robot vision systems, paving the way for our presentation of Stanley the robot racing car. While high-level architectures of such systems are still being developed, the low-level SP primitives have evolved into a stable set that is increasingly used. This set, along with processing modules that are increasingly available in software libraries such as the free OpenCV vision library [13] (described in [10]), make possible the inclusion of robot-like vision primitives in numerous other camera-based applications.

AUTHORS

Gary Bradski (garybradski@gmail.com) is a consulting professor with Stanford University. His research interests are in scalable machine learning and computer vision.

Adrian Kaehler (adrian@cs.stanford.edu) is a senior scientist at Applied Minds corporation and a visiting researcher at Stanford University. His research interests include computer vision, machine learning, and robotics.

REFERENCES

[1] D.A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 2002.

[2] F.J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679–698, 1986.

[3] J. Koplowitz and J. DeLeone "Computer vision and image understanding," *Hiearchical Representation of Chain-Encoded Binary Image Contours,* vol. 63, no. 2, pp. 344– 352, Mar. 1996.

[4] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning. New York: Springer, ch. 6, 2001.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press and McGraw-Hill, 2001, ch. 13, pp. 273–301.

[6] B. Schiele and J. Crowley, "Probabilistic object recognition using multidimensional receptive field histograms," in *Proc. Int. Conf. Pattern Recognition*, Aug. 1996, pp. 50-54.

[7] J.B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, Berkeley, pp. 281–297, 1967.

[8] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE PAMI*, vol. 24, no. 5, pp. 603–619, May 2002.

[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Boston: Academic Press, 1990.

[10] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA: O'Reilly Press, 2008.

[11] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Imaging Understanding Workshop*, 1981, pp. 121–130.

[12] DARPA Grand Challenge, 2005 [Online] Available: http://en.wikipedia.org/wiki/2005_DARPA_Grand_Challenge

[13] Open Source Computer Vision Library (OpenCV) [Online] Available: http://www.intel.com/technology/computing/opencv/

SP