K

Kalman Filter

Gregory F. Welch
Institute for Simulation & Training, The University of Central Florida, Orlando, FL, USA

Synonyms

Kalman-Bucy filter; KF

Related Concepts

▸Sensor Fusion

Definition

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error. The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

Background

In 1960, Rudolf E. Kalman published his famous paper describing a recursive solution to the discrete-data linear filtering problem [1]. Since that time, due in large part to advances in digital computing, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. The goal of the filter is to produce evolving optimal estimates of a modeled process from noisy measurements of the process.

Theory

The Kalman filter addresses the general problem of trying to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

at time step k, with a measurement $z \in \mathbb{R}^m$ that is

$$z_k = Hx_k + v_k. \tag{2}$$

The random variables w_k and v_k represent the *process noise* and *measurement noise*, respectively. They are assumed to be independent of each other, white, and with normal probability distributions

$$p(w) \sim N(0, Q), \text{and} \tag{3}$$
$$p(v) \sim N(0, R). \tag{4}$$

The $n \times n$ matrix A in the difference Eq. (1) relates the state x at the previous time step $k-1$ to the state x at the current step k , in the absence of either a driving function or process noise. The $n \times l$ matrix B relates an optional control input $u \in \mathbb{R}^l$ to the state x. The $m \times n$ matrix H in the measurement Eq. (2) relates the state to the measurement z_k.

K. Ikeuchi (ed.), *Computer Vision*, DOI 10.1007/978-0-387-31439-6,

One usually does not know the true form of the process (1) and associated noise parameter (3) nor the true measurement model (2) and associated noise parameter (4), but in practice one can often arrive at useful models via analytical formulations and laboratory-based measurements.

Using the process and measurement models (1)–(4), and real (noisy) measurements $\hat{z}_k$ at each time step k, the *Kalman filter* is used to recursively estimate the first two statistical moments of the process: the mean $\hat{x}_k$ and the error covariance P_k.

The filter is typically implemented in two steps, a *time update* step and a *measurement update* step, as follows:

Time update:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$
$$P_k^- = AP_{k-1}A^\intercal + Q$$

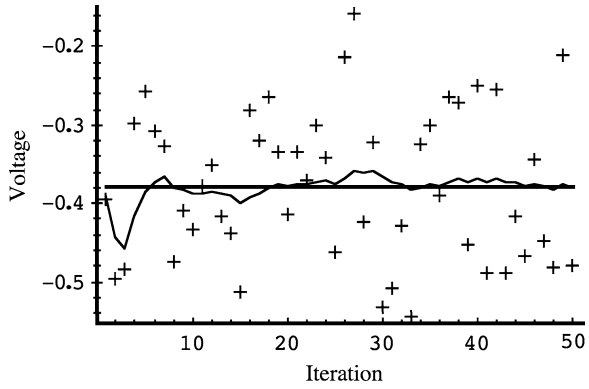
Measurement update:

$$K = P_k^- H^\intercal (HP_k^- H^\intercal + R)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K(\hat{z}_k - H\hat{x}_k^-)$$
$$P_k = (I - KH)P_k^-$$

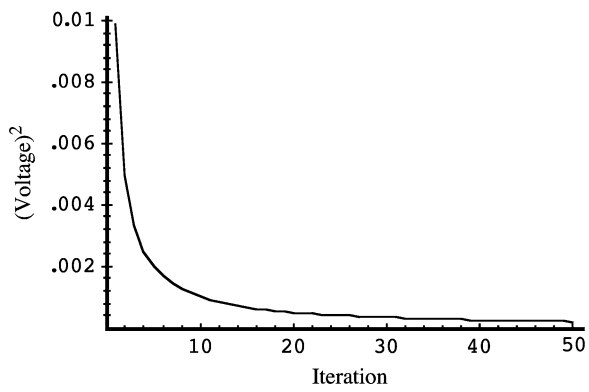
Repeatedly applying these steps recursively estimates the process mean $\hat{x}_k$ and the error covariance P_k. Because the measurements can vary in form and timing, the filter is often characterized as a tool for *sensor fusion*.

The *Kalman filter* is optimal in that the $n \times m$ *Kalman gain* matrix K minimizes the trace of a posteriori error covariance P_k.

An accessible high-level introduction to the general idea of the Kalman filter can be found in Chap. 1 of [2]. A more complete introduction can be found in [3] and in [4] which also contains some interesting historical narrative. More extensive references include [2, 5–9].



Kalman Filter, Fig. 1 The true random constant x_k (*solid line*), the noisy measurements $\hat{z}_k$ (*cross marks*), and the filter estimate $\hat{x}_k$



Kalman Filter, Fig. 2 The error covariance P_k. After 50 iterations, the covariance has settled to a relatively small 0.0002 volts^2

Application

Despite the fact that employed process models rarely match the corresponding true systems, and the noise models rarely exhibit the characteristics required for optimality (zero mean, normally distributed, and independence over space and time), the Kalman filter remains popular – perhaps due to its relative simplicity and robustness. It continues to be used widely in diverse application areas such as electronics, robotics, localization, navigation, and even economics. In computer vision, variations of the Kalman filter are typically used to estimate structure, motion, and camera parameters. Early examples include [10–13]. Both the OpenCV software project [14] and the Matlab numerical computing environment [15] include Kalman filter functions.

Experimental Results

A relatively simple example of using the Kalman filter to estimate a scalar random constant is given in [3], with complete details for the structure of the filter, the parameters, the initial conditions, and various results. The example presumes access to noisy measurements of a voltage that is corrupted by a 0.1 volt RMS white measurement noise. Referring back to Eqs. (1) and (2), the value to be estimated is presumed constant so $A = 1$, there is no control input so $u = 0$

(and B is irrelevant), and the noisy measurements are of the state (the voltage) directly so $H = 1$. For a true voltage of $x = -0.37727$, $Q = 1 \times 10^{-5}$, and $R = (0.1)^2 = 0.01$, plots for the true voltage x_k, noisy measurements, and estimated voltage $\hat{x}_k$ are shown in Fig. 1; and the error covariance P_k is shown in Fig. 2.

References

1. Kalman RE (1960) A new approach to linear filtering and prediction problems. Trans ASME J Basic Eng 82 (Series D):35–45
2. Maybeck PS (1979) Stochastic models, estimation and control, vol 1. Volume 141 of mathematics in science and engineering. Academic, New York
3. Welch G, Bishop G (1995) An introduction to the Kalman filter. Technical report TR95-041, Department of Computer Science, University of North Carolina at Chapel Hill
4. Sorenson HW (1970) Least-squares estimation: from gauss to kalman. IEEE Spectr 7:63–68
5. Brown RG, Hwang PYC (1996) Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions, 3rd edn. Wiley, New York
6. Gelb A (1974) Applied optimal estimation. MIT, Cambridge
7. Grewal MS, Andrews AP (2001) Kalman filtering theory and practice using MATLAB, 2nd edn. Information and system sciences series. Wiley, New York
8. Jacobs O (1993) Introduction to control theory, 2nd edn. Oxford University Press, Oxford/New York
9. Lewis FL (1986) Optimal estimation: with an introduction to stochastic control theory. Wiley, New York
10. Stuller J, Krishnamurthy G (1983) Kalman filter formulation of low-level television image motion estimation. Comput Vis Graph Image Process 21(2):169–204
11. Matthies L, Kanade T, Szeliski R (1989) Kalman filter-based algorithms for estimating depth from image sequences. Int J Comput Vis 3(3):209–238
12. van Pabst JV, Krekel PFC (1993) Multisensor data fusion of points, line segments, and surface segments in 3D space. Proc. SPIE, Sensor Fusion VI, 190 (August 20, 1993), pp 190–201
13. Azarbayejani A, Pentland A (1995) Recursive estimation of motion, structure, and focal length. IEEE Trans Pattern Anal Mach Intell 17(6):562–575
14. Bradski G (2000) The OpenCV Library. Dr. Dobb's Journal of Software Tools. http://opencv.willowgarage.com/wiki/CiteOpenCV
15. MATLAB (2012) Version 8.0.0.273 (R2012b). The MathWorks Inc. Natick, Massachusetts. http://www.mathworks.com/products/matlab/

Kalman-Bucy Filter

▸Kalman Filter

Karhunen–Loève Transform (KLT)

▸Principal Component Analysis (PCA)

Kernel Estimation

▸Blind Deconvolution

KF

▸Kalman Filter

Kinematic Chain Motion Models

▸Kinematic Motion Models

Kinematic Motion Models

Christoph Bregler
Courant Institute, New York University, New York, NY, USA

Synonyms

Kinematic chain motion models

Definition

Kinematic motion models are mathematical models that describe the motion of objects without consideration of forces.

Background

Although kinematics is in general more broadly defined, in computer vision, the term kinematic motion model is usually used synonymously with kinematic chain motion models, a term that comes from the field of Robotics. Such a model defines a set of rigid objects (called links) that are connected with joints. The motion of the links is constraint by the degrees of freedom of the joints. For instance, a link can only

rotate relative to another link around a joint axis. These models are most commonly used to describe human and animal skeletal models or robotic manipulators. The motion constraints can be used for robust visual tracking of skeletal configurations in single-view or multi-view video. Other kinematic models include special cases like one single rigid object, or more general motion models like deformable models, often called nonrigid models. A kinematic motion model does not consider mass distributions and forces that influence the motion. This is described by so-called Dynamical Models.

Theory

A kinematic chain is a sequence of rigid links l_i that are connected by joints at location j_i (Fig. 1). There are different types of joints (Fig. 2). The simplest is a "revolute joint," which has one axis of rotation α_i. Other possible joint types are "prismatic joints" (sliding along an axis) and joints with 2 or 3 axes of rotations (sometimes called "ball joints" or "spherical joints"). The configuration of the kinematic chain is defined by the relative joint angles between links. For instance, Fig. 1a shows a chain model with all angles set to 0 and another configuration (Fig. 1b, c) with different angle values. The first link in the chain is called the base link l_0. To represent a human skeleton, the base link is usually the hip and several chains originating from l_0 define spine, head, arms, and legs. Connecting several chains this way leads to a kinematic tree, but for simplicity only chains are discussed. The base joint l_0 is either fixed or can move with any arbitrary translation T_0 and rotation R_0. The configuration $\theta = [R_0, T_0, \alpha_1, \alpha_k]$ of all $k + 6$ degrees of freedom (local joint angles and l_0 translation and orientation) is often called the pose.

Forward Kinematics

Mathematically the kinematic model (M) can be defined in terms of how points P_i on a specific link in the rest-pose are moved to points Q_i by a new kinematic configuration θ (Fig. 1).

$$Q_i = M(\theta, P_i) \tag{1}$$

A very important aspect of kinematic chain models is, that a rotation of a link l_a affects all link motions further down the chain l_b with $a > b$. For example, a rotation of a shoulder joint affects the global elbow position and rotation of the wrist link. But a local rotation of the elbow will not affect links further up. Starting with the scenario of changing the joint angle α_k of the last link l_k only (Fig. 1b), the motion of a point P_i on l_k to Q_i can be calculated by the following translation and rotation using homogeneous coordinates for P_i and $Q_i = [x, y, z, 1]^T$:

$$\begin{aligned} Q_i &= \begin{bmatrix} & \mathbf{R_k} & & (j_k - R_k \cdot j_k) \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \\ P_i &= G_k \cdot P_i \end{aligned} \tag{2}$$

The rotation matrix R_n can be parameterized using Euler Angles or the exponential map of a twist [1]. The exponential map of a twist leads to simpler computation of derivatives in tracking equations (as described below):

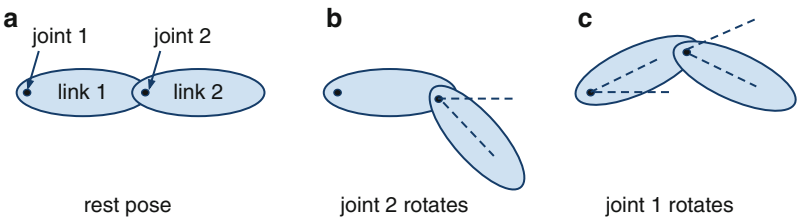
$$G_k(\alpha) = \exp\left(\begin{bmatrix} 0 & -\omega_z & \omega_y & v_1 \\ \omega_z & 0 & -\omega_x & v_2 \\ -\omega_y & \omega_x & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \alpha \right)$$

$$\text{with twist } \xi_k = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{3}$$

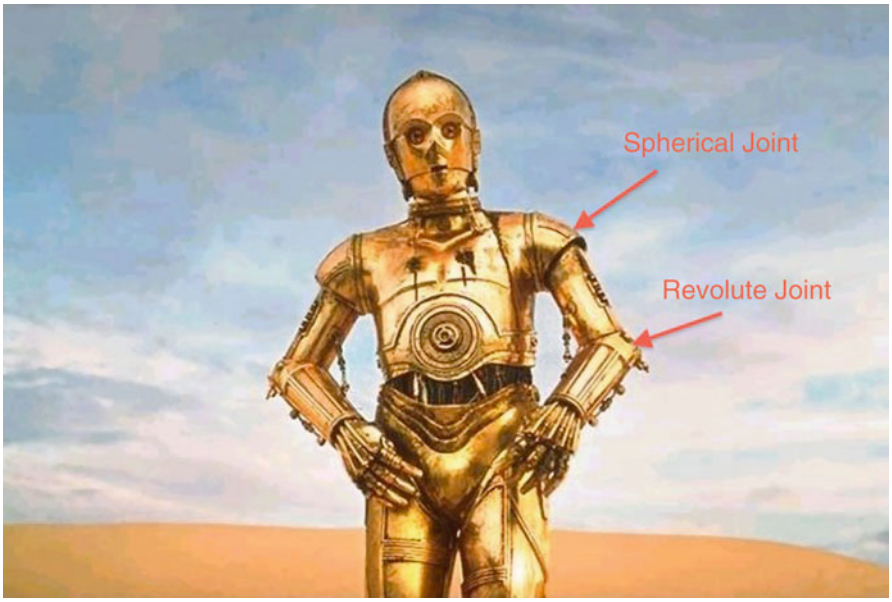
For a single axis joint (1 DOF), the twist ξ_k defines the axis at joint j_k (with $\omega_x^2 + \omega_y^2 + \omega_z^2 = 1$). α is the angle around the axis. The twist is constant, and α is the only varying parameter. For ball joints, it is modeled as a sequence on single axis joints around the same location j_k. Alternatively it can be modeled as letting all directional components for the twist vary.

Rotations of links further up the chain can be incorporated link by link. For instance, all points that are affected by a rotation around link location l_{k-1} are defined by G_{k-1}, including the points Q_i that have been moved by G_k already (Fig. 1c)

$$Q_i' = G_{k-1}(\alpha_{k-1}) \cdot Q_i = G_{k-1}(\alpha_{k-1}) \cdot G_k(\alpha_k) \cdot P_i \tag{4}$$



Kinematic Motion Models, Fig. 1 TODO example of a kinematic chain representation. (**a**) Rest pose. (**b**) Joint 2 rotates. (**c**) Joint 1 rotates



Kinematic Motion Models, Fig. 2 TODO different kinematic joint types

Working all the way backwards to the base link l_0 results in a product of G_0 to G_k

$$Q_i'' = G_0 \cdot G_1 \cdot \ldots \cdot G_k \cdot P_i = M_k(R_0, T_0, \alpha_1, \ldots, \alpha_k) \cdot P_i = M(\theta) \cdot P_i \quad (5)$$

The position and motion M_i of links further up the chain ($i < k$) can be modeled in the same way, as the product of G_0 to G_i. Using for G_0 to G_k the exponential map of twist representation, this equation is referred to as the "product of exponential map" representation [1]. An alternative representation is the more traditional Denavit-Hartenberg representation [2], which does not allow such an easy construction of the motion model or simple computation of derivatives and linearizations (more advantages of the product of exponential map representation are discussed in [1]).

Application

Kinematic Tracking

As mentioned earlier, kinematic chain models are commonly used in computer vision to track the articulated poses of humans. If visual markers are identified with high accuracy, either using a motion capture system or patterns that are easy to track, estimating the pose $\theta = [R_0, T_0, \alpha_1, \ldots \alpha_k]$ can be done with an error minimization technique. For instance, a marker-based motion capture system estimates 3D reconstructions of body markers $\hat{Q}_i$. The function $M(\theta)$ in Eq. (5) can be easily linearized [1] and used in estimating the 3D pose θ with the Newton-Raphson method or any other optimization technique. If 2D points are located in an image, the same can be applied to video tracking in single or multi-view footage using a camera model that relates 3D points to 2D projections. The $M(\theta)$ model can even be incorporated in optical flow models, and

a direct estimation of θ is possible from spatiotemporal image gradients [3]. More recent techniques do not assume 2D point tracks and use probabilistic techniques. For example, features like silhouettes, edges, color features, or patches can be estimated. The M function is then used to evaluate a pose hypothesis θ with an additional mapping of M to the feature space [4–11].

Most recently alternatives to kinematic motion models that use different data structures have been also applied to the domain of human pose estimation [12–16].

Model Estimation

Estimating the geometry of the kinematic chain (the point locations P_i in the rest pose, or other geometric shape representations) and all joint axes ξ_k in the rest pose can also be done with optimization techniques. This requires collecting a sequence of different poses that ideally go through the entire "range of motion" of a person or articulated object, a common procedure for motion capture systems, called "subject calibration." The same can be also done for computer vision systems. Once a sufficient number of frames with different pose configurations are collected, jointly the poses θ_t for each frame t and the common model parameters ξ_k and P_i (that are constant over time) are estimated. Some example techniques are described in [17, 18].

Animation

Kinematic chain models are also used extensively in computer graphics and computer animation. Animating a character efficiently requires locally controlling the joint angles in the same way as previously outlined. All major 3D graphics packages support skeletal models that are based on kinematic chain motion models.

References

1. Murray R, Li Z, Sastry S (1994) A mathematical introduction to robotic manipulation. CRC, Boca Raton
2. Denavit J, Hartenberg R (1955) A kinematic notation for lower-pair mechanisms based on matrices. Trans ASME J Appl Mech 22(1):215–221
3. Bregler C, Malik J, Pullen K (2004) Twist based acquisition and tracking of animal and human kinematics. Int J Comput Vis 56(3):179–194
4. Hogg D (1983) Model-based vision: a program to see a walking person. Image Vis Comput 1(1):5–20
5. Rohr K (1994) Towards model-based recognition of human movements in image sequences. CVGIP-Image Underst 59(1):94–115
6. Gavrila D, Davis L (1996) 3-D model-based tracking of humans in action: a multi-view approach. In: 1996 IEEE computer society conference on computer vision and pattern recognition. San Francisco, pp 73–80
7. Rehg J, Kanade T (2005) Model-based tracking of self-occluding articulated objects. In: Proceedings of the fifth international conference on computer vision. IEEE, Cambridge, MA, pp 612–617
8. Duetscher J, Blake A, Reid I (2000) Articulated body motion capture by annealed particle filtering. In: IEEE conference on computer vision pattern recognition (CVPR). IEEE computer society, Miami, FL, p 2126
9. Sidenbladh H, Black M, Fleet D (2000) Stochastic tracking of 3D human figures using 2D image motion. In: European conference on Computer Vision 2000 (ECCV), Dublin, pp 702–718
10. Kakadiaris L, Metaxas D (2002) Model-based estimation of 3D human motion. IEEE Trans Pattern Anal Mach Intell 22(12):1453–1459
11. Sigal L, Bhatia S, Roth S, Black M, Isard M (2004) Tracking loose-limbed people. In Proceedings. IEEE conference Computer Vision and Pattern Recognition (CVPR), Washington, DC, Vol.1, pp I-421-I-428
12. Ramanan D, Forsyth D, Zisserman A (2005) Strike a pose: tracking people by finding stylized poses. In: Proc. IEEE conference Computer Vision and Pattern Recognition (CVPR), San Diego, CA, vol. 1, pp 271–278
13. Felzenszwalb P, Huttenlocher D (2005) Pictorial structures for object recognition. Int J Comput Vis 61(1):55–79
14. Mori G, Malik J (2002) Estimating human body configurations using shape context matching. European conference on Computer Vision (ECCV), Copenhagen, pp 150–180
15. Shakhnarovich G, Viola P, Darrell T (2003) Fast pose estimation with parameter-sensitive hashing 9th International conference on computer vision (ICCV), Nice France, Vol. 2 750–757
16. Agarwal A, Triggs B (2005) Recovering 3D human pose from monocular images. IEEE Trans Pattern Anal Mach Intell 28(1):44–58
17. Lu T, O'connor J (1999) Bone position estimation from skin marker co-ordinates using global optimisation with joint constraints. J Biomech 32(2):129–134
18. Reinbolt J, Schutte J, Fregly B, Koh B, Haftka R, George A, Mitchell K (2005) Determination of patient-specific multi-joint kinematic models through two-level optimization. J Biomech 38(3):621–626