รายงานวิจัยฉบับสมบูรณ์

โครงการ
การใช้หลักการทางสถิติเพื่อพัฒนาการมองเห็นของหุ่นยนต์แบบเคลื่อนที่ได้และมีกล้องสามตัว

Statistical Approaches to Trinocular Stereo Mobile Robot Vision

Project #MRG4780209

โดย Asst. Prof. Dr. Matthew N. Dailey และคณะ

February 2008

1. Acknowledgments

I am grateful to the Thailand Research Fund for giving me the opportunity to develop the research described in this report. I thank the TRF staff, especially my program officer, K. Saengpetch, for stepping me through the process. Dr. Manukid Parnichkun of Mechatronics at AIT has been an excellent mentor.

Some of the work described in this report was carried out in cooperation with students. Mr. Nico Hochgeschwender of the Ravensburg-Weingarten University of Applied Sciences, Germany, and Mr. Yoichi Nakaguro of Sirindhorn International Institute of Technology, Thammasat University, worked on the Kanade-Lucas-Tomasi 3D point sensor model. Mr. Nyan Bo Bo of Sirindhorn International Institute of Technology, Thammasat University, worked on the hand detection system. Mr. Ramesh Marikhu of the Asian Institute of Technology worked on the image segmentation system.

Finally, I am grateful to my colleagues Dr. Stanislav Makhanov and Dr. Bunyarit Uyyanonvara at the Sirindhorn International Institute of Technology, Thammasat University, for their research collaboration and valuable discussions.

2. บทคัดย่อ

วิทยาการหุ่นยนต์มีศักยภาพมหาศาลที่จะเปลี่ยนวิถีชีวิตของเราในปัจจุบันไปในทางที่ดีขึ้น แต่อุปสรรคสำคัญประการหนึ่งในการพัฒนาวิทยาการด้านนี้ให้เกิดผลสัมฤทธิ์ คือความยากในการคิดค้นอัลกอริทึมสำหรับการรับรู้ (perception) และการทำความเข้าใจสภาพแวดล้อมของหุ่นยนต์ โครงการนี้มีจุดประสงค์ที่จะพัฒนาอัลกอริทึมสำหรับการมองเห็นของหุ่นยนต์ (machine vision) เพื่อการรับรู้และการทำความเข้าใจสภาพแวดล้อมของหุ่นยนต์ โดยแบ่งงานเป็น 2 ส่วน กล่าวคือ ส่วนแรกเป็นส่วนพื้นฐาน ประกอบด้วยการคิดค้นอัลกอริทึมใหม่ ๆ และการพัฒนาซอฟท์แวร์ไลบรารีเพื่อสังเคราะห์ข้อมูลจากฉาก (scene) 3 มิติที่ได้จากกล้องจับภาพ 1 ตัว (monocular camera) หรือกล้องจับภาพ 3 ตัวที่จับภาพพร้อมกัน (trinocular stereo camera) ส่วนที่สองเป็นการใช้ผลลัพธ์จากส่วนแรกเพื่อคิดค้นและพัฒนาวิธีการใหม่ ๆ ในการทำความเข้าใจเนื้อหาของฉาก ตัวอย่างเช่น คิดค้นวิธีการตรวจหามือคนที่ปรากฏในลำดับภาพวิดีโอ คิดค้นวิธีการวิเคราะห์ภาพ 2 มิติเพื่อหาโครงสร้างเครือข่ายบางชนิด เช่น แผนที่ที่ประกอบด้วยถนนสายต่าง ๆ เป็นต้น คณะผู้วิจัยมีเป้าหมายให้ผลลัพธ์ที่ได้จากโครงการนี้ เป็นก้าวหนึ่งในการพัฒนาความสามารถในการรับรู้ของหุ่นยนต์ เพื่อการใช้ประโยชน์ของเทคโนโลยีแขนงนี้ในรูปแบบต่าง ๆ ในอนาคต

Robotics technology has the potential to revolutionize our lives, but one of the largest obstacles to widespread adoption of robots that live and work with us is a lack of algorithms for perception and understanding of the environment. In this project, we develop machine vision algorithms for a variety of tasks related to perception and understanding of the environment. In the foundational part of the project, we have developed new algorithms and software libraries for extracting information about a three-dimensional scene from trinocular stereo as well as monocular video streams. Building on the base software libraries, we have also developed new approaches to understanding the contents of a scene, including a method for detecting human hands in video sequences and extracting network-like structures from 2D images. The work is a step along the path to improved perceptual abilities for interactive robotic applications.

Keywords: Structure from motion, stereo vision, robot vision, hand tracking, network extraction, statistical modeling.

2.1 ชื่อโครงการ

การใช้หลักการทางสถิติเพื่อพัฒนาการมองเห็นของหุ่นยนต์แบบเคลื่อนที่ได้และมีกล้องสามตัว
Statistical Approaches to Trinocular Stereo Mobile Robot Vision

2.2 ชื่อหัวหน้าโครงการ

Asst. Prof. Dr. Matthew N. Dailey
Computer Science and Information Management
School of Engineering and Technology
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120
Phone: +66 2 524 5712
FAX: +66 2 524 5721
E-mail: mdailey@ait.ac.th

2.3 สาขาวิชาที่ทำงานวิจัย

Machine Vision, Robotics, Statistical Modeling, Machine Learning

2.4 งบประมาณทั้งโครงการ

240,000 บาท/ปี

2.5 ระยะเวลาดำเนินงาน

2 ปี

2.6 ปัญหาที่ทำการวิจัย และ ความสำคัญของปัญหา

Over the past 30 years, robotics technology has developed to the point that robotics products are moving out of the laboratory and into retail markets. Robotics technology is expanding to keep us safe, keep us healthy, and eliminate dangerous and tedious tasks from our lives.

However, although massive amounts of cheap compute power will be available to robotics applications in the near future, the algorithms we have for *exploiting* this massive compute power are sorely lacking. When compared to humans (or even rats), modern technology is most glaringly deficient in *perception and understanding of the environment*. The gap in visual perception abilities between animals and machines is perhaps the greatest obstacle robotics practitioners face today.

Machine vision research aims to close this crucial gap. In this project, we aim to develop machine vision algorithms for structure learning and scene understanding, towards ultimately endowing robotic applications with the ability to interact with the environment and humans in that environment.

2.7 วัตถุประสงค์

Our project has the following goals, all involving the use of low-cost trinocular and monocular video cameras to extract 3D information from 2D image data, isolate objects of interest in an image, and track objects of interest over time.

- Develop methods for extracting information about a three-dimensional scene from the motion of a trinocular camera rig through that environment.
- Develop methods for tracking human hands in a monocular video sequence.
- Develop methods for extracting network-like structures from 2D images.

Our early work focused on the use of trinocular camera rigs to extract 3D information from the world, and our more recent work has focused on developing a set of modules for extracting useful information from simpler, lower-cost monocular camera images.

2.8 ระเบียบวิธีวิจัย

In the first year of the project, we built a trinocular stereo camera rig and developed a collection of software libraries for capturing images trinocular and monocular images, manipulating images, extracting features from those images, and obtaining correspondences between features over time. In the second year of the

project, we applied the software libraries to the machine vision applications described in the objectives: extracting 3D information from 2D images, detecting human hands, detecting human faces, and extracting network structures from images.

2.9 ผลการวิจัย

We have made significant progress on all of the goals of the project, leading to several publications in refereed international conferences and journals. The results may be summarized as follows:

(a) We have developed a new sensor model for extracting 3D line segments from images obtained by a trinocular stereo camera rig and tested the model in a series of simulation experiments.

(b) We have developed a new sensor model for extracting 3D points from images obtained by a trinocular stereo camera rig and tested the model in a series of real-world experiments using the hardware constructed under this grant.

(c) We have developed a new approach to detecting human hands in low-resolution images based on the Viola and Jones cascade technique and a Mahalanobis classifier. We have tested this system on monocular video sequences of crowded scenes.

(d) We have developed a new approach to extracting network-like structures from 2D images. The technique has many applications in computer and robot vision; thus far we have tested the system on extraction of road networks from satellite images.

3. เนื้อหางานวิจัย

3.1 Sensor model for 3D line segments

Our algorithm, VL-SLAM, is based on the "FastSLAM" family of algorithms proposed by Montemerlo, Thrun and colleagues [14]. At each point $t \in 1 \ldots T$, the robot performs an action u_t taking it from position s_{t-1} to s_t and uses its sensors to obtain an observation z_t. We seek a recursive estimate of

$$p(s_{0:t}, \Theta \mid u_{1:t}, z_{1:t}) \tag{1}$$

where Θ is a map containing the positions of each of a set of point landmarks. Rather than estimate the distribution (1) analytically, we approximate the posterior with a discrete set of M_t samples (sometimes called particles)

$$\left\{ < s_{0:t}^{[m]}, \Theta_{0:t}^{[m]} >, \text{ where each index } m \in 1 \ldots M_t \right\}. \tag{2}$$

Here $s_{0:t}^{[m]}$ is the specific robot trajectory from time 0 to time t associated with particle m, and $\Theta_{0:t}^{[m]}$ is the stochastic landmark map associated with particle m (the map is derived from $s_{0:t}^{[m]}$, $z_{1:t}$, and $u_{1:t}$). FastSLAM (and VL-SLAM) use the sequential Monte Carlo techniques of sequential importance sampling and importance resampling. First, for each particle, we sample from some *proposal distribution*

$$\pi(s_{0:t}, \Theta_{0:t} \mid z_{1:t}, u_{1:t}) \tag{3}$$

to obtain a temporary set of particles for time t, then evaluate the *importance weight* $w^{[m]}$ for each temporary particle, where

$$w(s_{0:t}, \Theta_{0:t}) = \frac{p(s_{0:t}, \Theta_{0:t} \mid u_{1:t}, z_{1:t})}{\pi(s_{0:t}, \Theta_{0:t} \mid u_{1:t}, z_{1,t})}. \tag{4}$$

The importance weights are normalized to sum to 1, then we sample M_t particles, with replacement, from the temporary particle set according to the normalized weights.

VL-SLAM extends FastSLAM with a new sensor model for 3D line segments and a new proposal distribution $\pi(\cdot)$ appropriate for environments with highly ambiguous observation-model correspondences. We first describe the 3D line segment sensor model then VL-SLAM proposal distribution.

3.1.1 VL-SLAM 3D Line Segment Sensor Model

After each robot motion u_t, a set of trinocular stereo images is captured, and a set z_t of landmark measurements (line segments) is extracted from those images. These line segment measurements, along with the measured motion u_t, are used to update each particle's map and position estimate.

Our system assumes a calibrated stereo camera rig with three pinhole cameras. It can handle general fundamental matrices (the images need not be perfectly rectified), but we do assume that one camera is roughly horizontally displaced and a second camera is roughly vertically displaced from a third (reference) camera.

The basic 2D feature in our system is the line segment. We extract line segments using Canny's method [3] following the implementation in VISTA [10]. The edge detector first performs nonmaxima suppression, links the edge pixels into chains, and retains the strong edges with hysteresis. Once edge chains are extracted from the image, we approximate each chain by a sequence of line segments. Short line segments, indicating edges with high curvature, are simply discarded in the current system.

We use a straightforward stereo matching algorithm similar to the approach of [19]. For each line segment in the reference image, we compute the segment's midpoint, then consider each segment intersecting that midpoint's

epipolar line in the horizontally displaced image. Segments not meeting line orientation and disparity constraints are discarded. Each of these potential matches determines the location and orientation of a segment in the third image. If such a consistent segment is indeed found in the third image, the potential match is retained; otherwise it is discarded. If at the end of this process, we have one and only one consistent match, we assume it correct; otherwise, the reference image line segment is simply ignored.

Now our goal is to estimate a three-dimensional line from the three observed two-dimensional lines. Infinite lines have four instrinsic parameters, so it would make sense to use a four-dimensional representation of a lines. However, since VL-SLAM uses a Kalman filter to combine landmark observations, we require a linear parameterization of landmarks, and no linear four-dimensional representation of lines exists [1]. Instead we represent lines with six components: a 3D point representing the midpoint of the observed line segment and a 3D vector whose direction represents the direction of the line and whose length represents the distance from the line segment's midpoint to one of its endpoints. This 6D representation behaves well under linear combination, so long as the direction vectors are flipped to have a positive dot product.

First we obtain a maximum likelihood estimate of the infinite 3D line's parameters assuming Gaussian measurement error in the image using Levenberg-Marquardt minimization [11]. As an initial estimate of the line's parameters, we use the 3D line (uniquely) determined by two of the 2D line segment measurements. Once the infinite line has been estimated, we find the segment's extrema and midpoint using the observed data.

Through each step of the 3D line estimation process, we maintain explicit Gaussian error estimates. We begin by assuming spherical Gaussian measurement error in the image with a standard deviation of one pixel. Arranging the n (x, y) coordinates of the pixels in a line as a column vector $\mathbf{x}$, the covariance of $\mathbf{x}$ is simply $\Sigma_{\mathbf{x}} = I_{2n \times 2n}$. Since the vector of parameters $\mathbf{l}$ describing the 2D line best fitting $\mathbf{x}$ is a nonlinear function $\mathbf{l} = \mathbf{f}(\mathbf{x})$, the covariance of $\mathbf{l}$ is $\Sigma_{\mathbf{l}} = J \Sigma_{\mathbf{x}} J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ evaluated at $\mathbf{x}$.

The maximum likelihood estimate of the 3D line obtained from the three 2D line segments $\mathbf{l} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3)$ is clearly not a simple function, since it is computed by an iterative optimization procedure. However, if $\mathbf{l} = \mathbf{f}(\mathbf{L})$ is the function mapping from the parameter space to the measurement space, it turns out that, to first order, $\hat{\mathbf{L}}$ is a random variable with covariance matrix $(J^T \Sigma_{\mathbf{l}} J)^{-1}$, where J is the Jacobian matrix $\frac{\partial \mathbf{l}}{\partial \mathbf{L}}$ [6]. The rank of the resulting covariance matrix is only four, however, so to constrain the remaining two degrees of freedom, we add to the rank-deficient covariance matrix a covariance matrix describing the expected error in our estimate of the segment's midpoint and another covariance matrix describing the expected error in our estimate of the segment's length. This gives us a full-rank covariance matrix that restricts matching line segments to not only be similar in terms of their supporting infinite line, but also to overlap and have similar length.

Once the six-dimensional representation of an observed 3D line is estimated from a trinocular line correspondence, it is necessary to transform that line from camera coordinates into robot coordinates, since the reference camera is in general translated and rotated relative to the robot itself. It is also necessary to transform landmarks from robot coordinates into world coordinates, when the robot's position is determined, for instance, and from world coordinates back to robot coordinates, when a landmark in the map is considered as a possible match for an observed (robot coordinate) landmark. In each of these cases, the transformed line $\mathbf{L}' = \mathbf{t}(\mathbf{L})$ is computed as a nonlinear function of the original line, and the transformed line's covariance is propagated by $\Sigma_{\mathbf{L}'} = J \Sigma_{\mathbf{L}} J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{t}}{\partial \mathbf{L}}$ evaluated at $\mathbf{L}$.

3.1.2 VL-SLAM Proposal Distribution

The proposal distribution $\pi(\cdot)$ (3) can be any distribution that is straightforward to sample from. However, it is best if $\pi(\cdot)$ closely approximates the full joint posterior (1), in which case the importance weights will be nearly uniform, and most particles will "survive" the resampling step. In FastSLAM 1.0 [14], the proposal distribution

is simply $p(s_t \mid s_{t-1}, u_t)$, i.e. the motion model predicting s_t given a previous position s_{t-1} and action u_t. The authors observe that this proposal distribution, while simple to sample from, does not take into account the current observation z_t. This leads to FastSLAM 2.0, in which the proposal distribution is $p(s_t \mid s_{0:t-1}^{[m]}, \Theta_{0:t-1}^{[m]}, u_{1:t}, z_{1:t})$. This distribution takes not only the previous robot pose s_{t-1} and current action u_t into account, but also considers the current map $\Theta_{0:t-1}$ and new observation z_t. In the general case, this distribution could be quite difficult to sample from, but the authors find that by linearizing the sensor model and applying the Markov assumption, the proposal distribution can be approximated to first order by a Gaussian distribution whose mean and covariance can be calculated from known quantities, *if the correspondence between the observation z_t and the current map $\Theta_{0:t-1}$ is known*. When the correspondences are unknown (the usual case in SLAM), FastSLAM 2.0 assumes the maximum likelihood correspondence or draws a sample from a probability distribution over all possible correspondences. When the observations and landmarks are sparse, as is the case in the FastSLAM environment, this is straightforward, and FastSLAM 2.0 is much more successful than FastSLAM 1.0, since it uses the available set of particles wisely [14].

In VL-SLAM, however, each observation consists of on the order of 100 individual 3D line segments, and typically the landmark database contains several potential matches for each observed line. This means that it is impossible to consider even a small fraction of the possible correspondences for each particle. In practice, to limit the computational complexity, we must draw a single correspondence from the set of all possible correspondences without considering too many alternatives. But how can we choose a likely correspondence for a given observation?

In VL-SLAM, when propagating a particle forward from time $t-1$ to time t, we first fraw a sample s_t' from the robot's motion model to establish a correspondence between the observed line segments and the current map (resembling FastSLAM 1.0), then from that intermediate sample point, assuming the established correspondence, sample again, from the FastSLAM 2.0 proposal distribution. As in FastSLAM 2.0, the proposal distribution is closer to the full joint posterior distribution, concentrating more of the temporary particles in regions of high probability according to the full joint posterior.

To calculate the importance weights for the the VL-SLAM proposal distribution, we first introduce random variables n_t indicating the correspondence between the line segments observed at time t and the map. In VL-SLAM, the mth particle's map $\Theta_{0:t}^{[m]}$ is a deterministic function of the sampled trajectory $s_{0:t}^{[m]}$, the sampled correspondences $n_{1:t}^{[m]}$, and the observations $z_{1:t}$, so we rewrite the desired full joint posterior as

$$p(s_{0:t}, n_{1:t} \mid u_{1:t}, z_{1:t}). \tag{5}$$

Now, assuming we have a good estimate of the full joint posterior at time $t-1$, the VL-SLAM proposal distribution can be written as the product

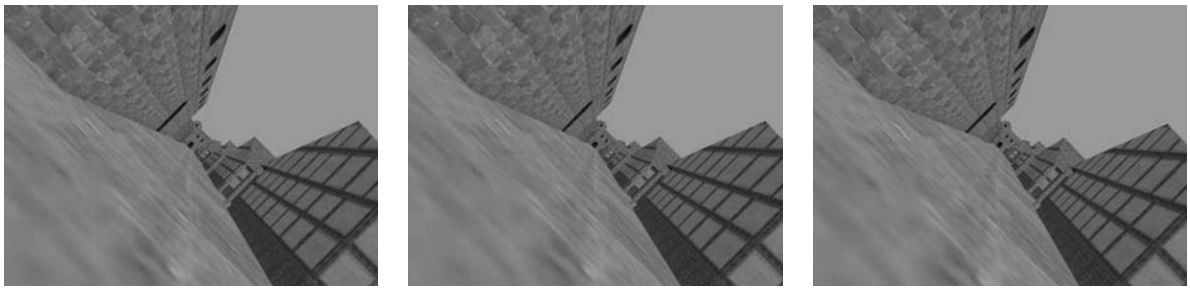
$$\begin{aligned} p(s_t^{[m]} \mid n_t^{[m]}, s_t'^{[m]}, s_{0:t-1}^{[m]}, n_{0:t-1}^{[m]}, z_{1:t}, u_{1:t}) \times p(n_t^{[m]} \mid s_t'^{[m]} s_{0:t-1}^{[m]}, n_{1:t-1}^{[m]}, z_{1:t-1}, u_{1:t}) \times \\ p(s_t'^{[m]} \mid s_{0:t-1}^{[m]}, n_{1:t-1}^{[m]}, z_{1:t-1}, u_{1:t}) \times p(s_{0:t-1}^{[m]}, n_{0:t-1}^{[m]} \mid u_{1:t-1}, z_{1:t}), \end{aligned} \tag{6}$$

where s_t' represents the intermediate sample drawn from the motion model. For the mth particle, the importance weight is the ratio of the expressions in (5) and (6), which, with several applications of Bayes' rule and the Markov assumption, can be closely approximated as (details ommitted):

$$w_t^{[m]} = \frac{p(s_t^{[m]} \mid s_{t-1}^{[m]}, u_t) p(z_t \mid s_{0:t}^{[m]}, n_{1:t}^{[m]}, z_{1:t-1})}{p(s_t^{[m]} \mid z_t, s_t'^{[m]}, n_{1:t}^{[m]}, s_{0:t-1}^{[m]}, z_{1:t-1}, u_{1:t}) p(s_t'^{[m]} \mid s_{t-1}^{[m]}, u_t)} \tag{7}$$

Following [14], we linearize the sensor model and motion model, which leads to straightforward Gaussian approximations for each of the terms in (7).

Except for the sensor model and proposal distribution just described, VL-SLAM is similar to FastSLAM (see [14] for details). Once correspondences and the sampled pose are determined for an individual particle, each



(a) (b) (c)

รูปที่ 1: Sample trinocular image set captured in simulation. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

observed landmark is combined with its corresponding map landmark using an extended Kalman filter, or initialized as a new landmark in the map. To achieve fast search for landmarks corresponding to a given observation, each particle's map is stored in a binary k-D tree whose leaves are the 3D line segments with associated Gaussian uncertainties. However, to minimize total memory requirements and to enable constant-time copying of maps during the resampling stage, the particles are allowed to share subtrees.

As we shall see in the next section, the diversity of possible correspondences introduced by the first sampling step (as in FastSLAM 1.0), combined with the use of the current observation z_t in the proposal distribution (as in FastSLAM 2.0), allows VL-SLAM to outperform both FastSLAM 1.0 and FastSLAM 2.0 on a challenging synthetic testbed.

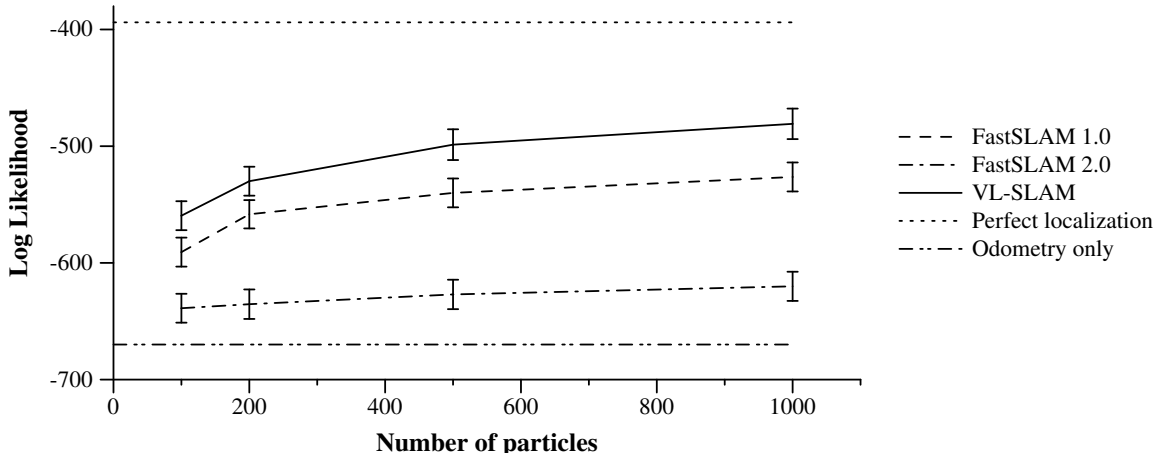
3.1.3 Experimental Results

To enable rigorous testing of VL-SLAM in an environment with a precisely known ground truth, we implemented a virtual reality simulation allowing a virtual robot to move through a virtual world rendered with OpenGL from a VRML model. We chose as an environment a publically-available 3D model of Housestead's fort, a Roman garrison from the 3rd century A.D. on Hadrian's Wall in Britain [2]. A sample view from our virtual trinocular stereo rig is shown in รูปที่1.

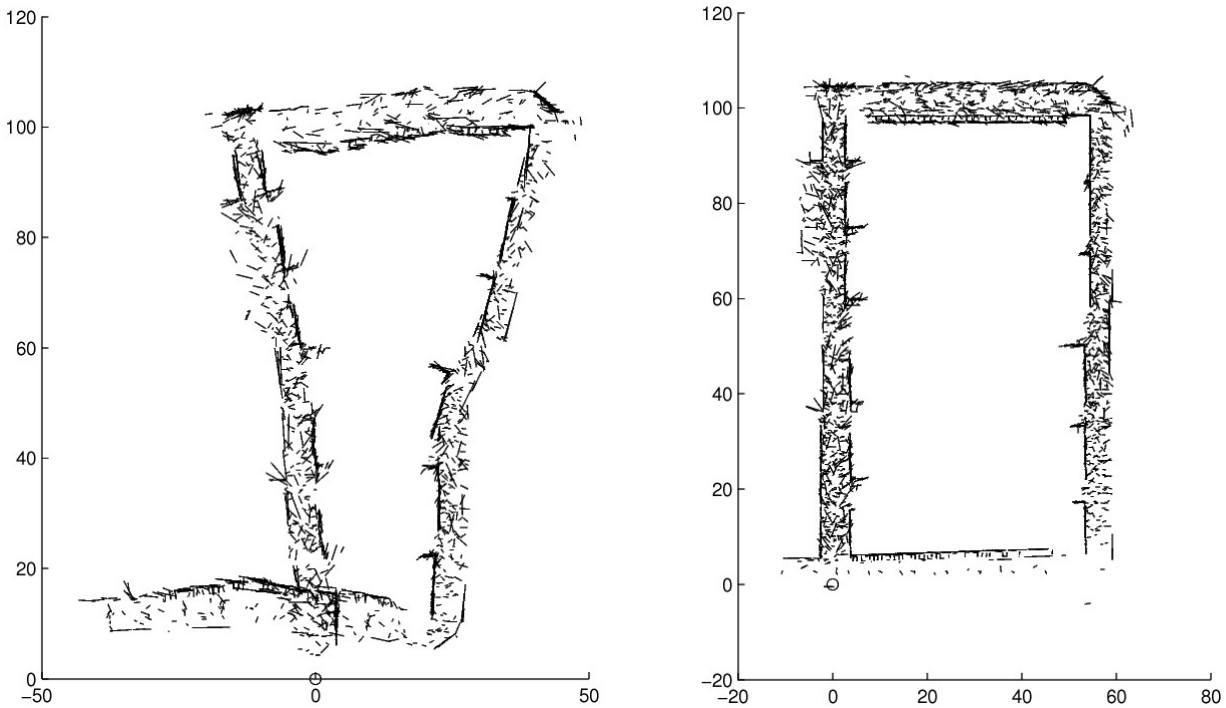
We teleoperated our virtual robot through this virtual world in a long loop of about 300m. At approximately 1m intervals, the virtual camera rig was instructed to capture a set of stills from its three cameras. The virtual camera models a real 10cm baseline, 70° field of view trinocular rig we recently built in our lab. To make the dataset somewhat challenging, we simulated the effects of a traveling on an imperfect outdoor surface, so that the robot's vertical (Z) position varied approximately ± 0.04m from 0, its pitch and roll varied ± 2.5 degrees from 0, and its yaw varied ± 3 degrees from its expected course.

This environment is an interesting testbed for VL-SLAM because, on the one hand, it generates many long, strong, straight edges that should be useful for localization. On the other hand, it is highly textured, creating a large number of edges, and the textures are highly repetitive in many places, leading to many ambiguities for correspondence algorithms. It is also large enough to preclude fine-grained grid-based techniques and noisy enough to preclude the use of flat-earth or three-degree-of-freedom assumptions.

We compared VL-SLAM with our own implementations of FastSLAM 1.0 and 2.0. As previously discussed, FastSLAM 2.0 was not designed to handle large observations with highly uncertain correspondences. In our implementation, we simply obtain the maximum likelihood correspondence assuming the robot is at the position obtained by propagating $s_{t-1}^{[m]}$ forward in time according to odometry to obtain $\hat{s}_t^{[m]}$. With this caveat about



รูปที่ 2: Log likelihood of line observations according to the best particle's sampled robot position and map, averaged over 320 sets of observations.



รูปที่ 3: Map constructed by VL-SLAM (a), compared to the the map assuming perfect knowledge of the robot's trajectory (b).

the FastSLAM 2.0 results, รูปที่2 shows one measure of each algorithm's performance: the log-likelihood of the observation data given the best particle's robot trajectory sample and map; Figure 3 shows the final map according to the best VL-SLAM particle. All of the localization algorithms do much better than the baseline (odometry-only) algorithm. Due to its commitment to robot position $\hat{s}_t^{[m]}$ when determining correspondences in our implementation, FastSLAM 2.0 fares rather poorly. Since FastSLAM 1.0 samples from the motion model before obtaining a correspondence, it performs much better, but VL-SLAM, which combines the best features of both algorithms, outperforms them both.

3.2 Sensor model for 3D points

We have also developed a sensor model for vision-based SLAM that, rather than using 3D line segments, uses 3D points as landmarks for localization and mapping.

3.2.1 Trinocular KLT as a sensor model for FastSLAM

In FastSLAM, the sensor model is fully described by the conditional probability $p(z_t \mid s_t, \Theta_{t-1}, n_t)$, explicitly conditioning on n_t, the set of correspondences between observations z_t and landmarks stored in Θ_{t-1}. The distribution is assumed to be a deterministic measurement function $f(\Theta_{t-1}, s_t, n_t)$ corrupted by Gaussian noise.

In our case, the observations are sets of 3D points in robot-relative coordinates, estimated by triangulation with a trinocular stereo vision rig. Our 3D point extraction procedure begins by obtaining 2D KLT (Kanade-Lucas-Tomasi) corner features [13] from each of three calibrated images simultaneously captured by the trinocular camera rig. We then find sets of corresponding features across the three images and triangulate to obtain an estimate of the putative feature's 3D position relative to the robot.

The basic idea of using KLT as a 2D feature detector is to find points with a complex local gradient field. Complexity of the gradient field is measured by the smaller eigenvalue of the matrix

$$Z = \begin{pmatrix} g_x^2 & g_{xy} \\ g_{xy} & g_y^2 \end{pmatrix} \tag{8}$$

in which the quantities are integrals of the squared gradient (in the case of g_x^2 and g_y^2) or the integral of the product of x and y gradients (g_{xy}) in a neighborhood around the point of interest. A point is selected as a KLT feature if the smaller eigenvalue λ_2 of Z is a local maximum and above some threshold λ. The motivation is that image points meeting the criterion have a local gradient structure that cannot be described by a single eigenvector (as would be the case for a simple edge), but have a more complex corner-like structure that should be easy to detect under various imaging conditions.

After extracting a set of KLT feature points from each of the three images acquired at time t, we attempt to find triples of corresponding points as a necessary step prior to triangulation. For each KLT point $p_{1,i}$ detected in image 1, we search image 2 for potentially corresponding points. For each point $p_{2,j}$ in image 2 close enough to the epipolar line corresponding to $p_{1,i}$, we triangulate using the calibrated intrinsic and extrinsic parameters of the camera rig to predict the putative object feature's appearance in image 3. If a suitable KLT point $p_{3,k}$ exists in image 3, we consider the triple $(p_{1,i}, p_{2,j}, p_{3,k})$ a candidate match and continue searching for other possible matches for $p_{1,i}$. If no consistent triples or more than one consistent triple is found for $p_{1,i}$, we throw it out. On the completion of this simple correspondence algorithm, we have a set of corresponding triples of 2D points that can then used for 3D estimation. Typically we begin with about 200 KLT points in each image and end up with about 20 corresponding triples.

(a) (b) (c)

รูปที่ 4: Trinocular image set captured in the lab. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

The last step of obtaining a sensor measurement is to estimate a 3D landmark in robot-relative coordinates given each triple of corresponding 2D KLT points. For each correspondence (p_1, p_2, p_3), we obtain an initial estimate of the 3D position P by triangulating from p_1 and p_2, then we use the Levenburg-Marquardt nonlinear least squares optimization algorithm [11] to find the 3D position P maximizing the likelihood of the 2D observations (p_1, p_2, p_3) assuming spherical Gaussian error in the measured image coordinates. We also obtain an estimate of confidence in the 3D point landmark position P by propagating the assumed measurement error through the maximum likelihood estimation procedure using the standard first-order approximation [6].

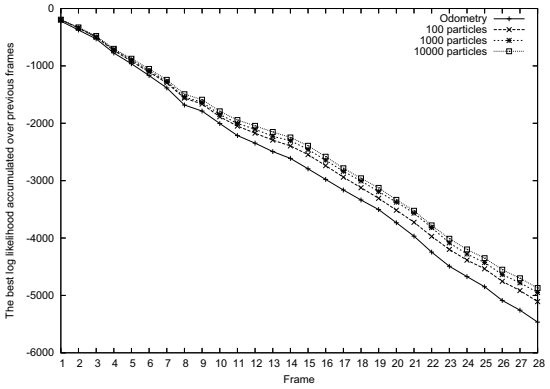
After 2D feature detection, correspondence estimation, and triangulation, we obtain a set of 3D point landmark observations with associated error covariance matrices. The set of landmarks with covariances makes up z_t, the robot's observation at time t, which is input to FastSLAM. From this point on, our system is identical to Thrun et al.'s FastSLAM 1.0 algorithm [14].

3.2.2 Experimental methods

To test KLT-based FastSLAM, we performed an experiment in the Image and Vision Computing Laboratory at SIIT. The room is a typical laboratory with desks, bookshelves and computers. Figure 4 shows an image set captured in the lab with the 10cm-baseline trinocular camera rig that was used in the experiment.

In this experiment, rather than mounting the rig on a robot, we simulated robot motions by manually moving a camera tripod. The simulated robot's position s_t in world coordinates at time t is defined as a vector with six degrees of freedom $s_t = (x, y, z, \phi, \theta, \psi)^T$. Here the x and y axes span a plane parallel to the floor of the lab, and z is the vertical distance of the reference camera's origin from the ground plane. The remaining three variables represent the robot's orientation. ϕ, θ and ψ stand for pitch, roll and yaw of the camera rig, respectively. During the experiment, due to the flat floor, z, pitch, and roll was always equal to zero throughout the experiment.

The camera rig cannot move itself, so in the experiment we roughly pushed or rotated the rig by hand from its original position to the next destination position in order to emulate a real robot move. Since each move of the rig is not perfect, the rig normally reaches a position slightly away (in terms of x, y and yaw, we do not measure z, pitch and roll since they are assumed to be zero in the experiment) from its destination position. So we treated the



รูปที่ 5: Log likelihood of observation sequence given the model.

difference of the original position and the desired destination position as robot odometry, and the difference of the original position and the actually reached position as a true move. To make the experiment simpler, we composed camera rig odometry so that each odometric move involves only translation or only rotation. More specifically, odometry is of the form $(x, y, 0, 0, 0, 0)^T$ for translation, and $(0, 0, 0, 0, 0, \psi)^T$ for rotation.

The actual path of the camera rig consisted of 29 positions $D_0, D_1, \ldots, D_{28}$ marked on the floor of the lab. At first the rig was positioned at D_0, which we defined to be the origin of the world coordinate system. The rig was then moved to each destination. Along the way, at each position, we measured the true position $T_1, T_2, \ldots, T_{28}$ of the rig and captured a trinocular image set. The simulated odometry measurements $O_1, O_2, \ldots, O_{28}$ were computed as $O_i = D_i - T_{i-1}$.

In this indoor experiment the robot's path was approximately composed of a 4 meter forward translation from O_1 to O_{10} (roughly 0.4 meters per move), a 180 degree rotation from O_{11} to O_{18} (roughly 22.5 degrees per move), and finally a 4 meter forward translation from O_{19} to O_{28} (roughly 0.4 meters per move).

Image sets (29 frames including the initial state) and odometry (28 six dimensional vectors) were collected in the lab. They were used as the input for KLT-Based FastSLAM to estimate the path of the camera rig and generate a 3D metric map of the lab. We ran the algorithm with 100, 1000 and 10000 particles. In order to compare the algorithm's performance against a baseline, we also ran the same mapping algorithm purely using odometry as the estimate of the camera position.

3.2.3 Experimental results

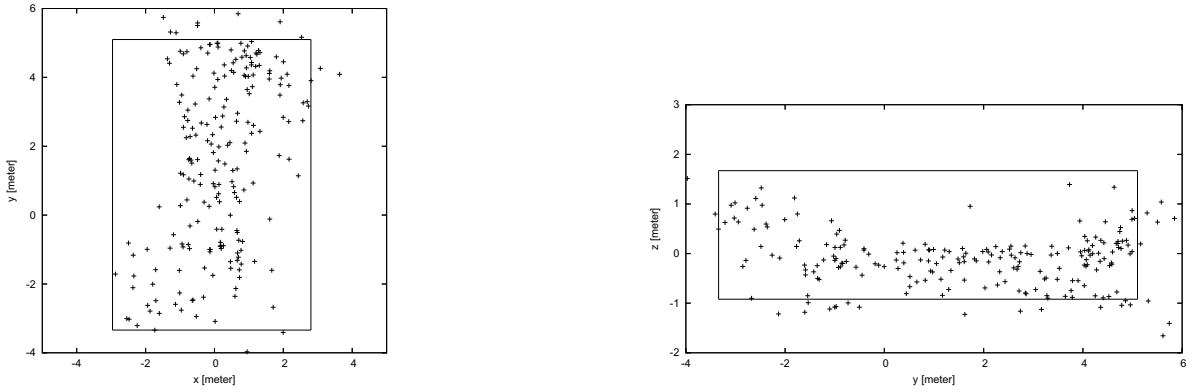
Log likelihood is a measure of accuracy of the current landmark observation given the previous observation. It is given by

$$\log\left(p(z_t \mid s_t^{[m]}, \Theta_{t-1}^{[m]})\right) \sim -\frac{1}{2} In\left|2\pi Q_t^{[m]}\right| - \frac{1}{2}(z_t - \hat{z}_t^{[m]})^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]}), \tag{9}$$

with covariance

$$Q_t^{[m]} = G_t^{[m]T} \Sigma_{t-1}^{[m]} G_t^{[m]} + R_t, \tag{10}$$

where $\hat{z}_t$ is an estimation of the new observation z_t, Σ_{t-1} is the covariance of the landmark before the new observation is made, G_t is the Jacobian of the sensor model with respect to the landmark, and R_t is the covariance of the Gaussian noise of the new observation [14].



(a) (b)

รูปที่ 6: Projection of the 3D metric map into 2D planes. The boundary of the lab is shown as a rectangle in the figures. (a) KLT point landmarks projected into $x - y$ plane, the top view of landmarks. (b) KLT point landmarks projected into $y - z$ plane, the side view of landmarks.

For each particle of each sequence of observation, we calculated the accumulated log likelihood, which is an addition of log likelihood over all the past sequences. It tells the degree of consistency of the map recorded in a particle. For each sequence of observation, we chose the particle that has the best (largest) value of accumulated log likelihood. The result is shown in รูปที่5. As the number of the particle used in the FastSLAM algorithm increases, the accumulated log likelihood becomes better. The result tells that the particle filter is working properly in the experiment, i.e. with more particles, the better localization of the camera rig and estimate of landmark positions for each observation sequence is achieved.

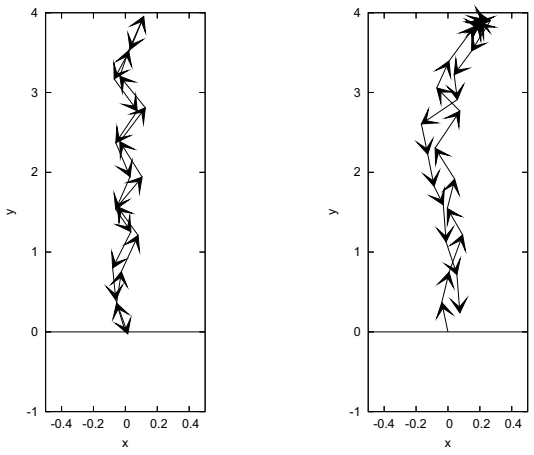
รูปที่6 shows 2D projections of the generated 3D map of the lab using 1000 particles. Only KLT point landmarks that were observed more than twice over all the observation sequences are plotted since landmarks observed only once tend to be noisy observations. Point landmarks in the map captured the actual distribution of edges and corners of objects seen in the lab.

In รูปที่7, the estimated path of the camera rig is shown.

3.3 Hand detection in monocular video

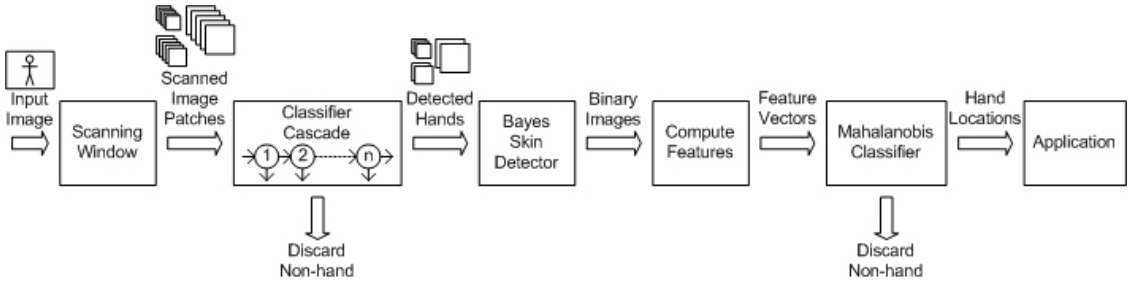
The next main machine vision module we have developed is a hand detection system.

รูปที่8 depicts our hand detection system's architecture schematically. A scan window sweeps over the input image at multiple scales. Each resulting image patch is classified as either hand or non-hand by a boosted classifier cascade [16, 9]. To further reduce false positive detections using a priori knowledge of hand color and geometry, each positive detection from the classifier cascade is further processed by a skin detection module, a feature extractor, and a simple classifier based on Mahalanobis distance to the "average" hand. We describe each of the modules in turn.



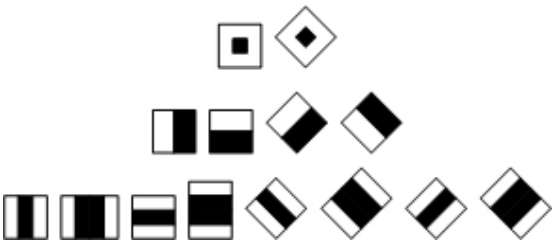
(a) (b)

รูปที่ 7: Path of the camera rig projected onto the $x - y$ plane. Each move is represented as a vector. The rig was put at $(x, y) = (0, 0)$ initially and was moved 28 times while taking an image set after each move. (a) True path of the camera rig. (b) Estimated path of the camera rig using 1000 particles.



รูปที่ 8: Hand detection system architecture.

(a)



(b)

รูปที่ 9: Haar-like features used to construct weak classifiers in the boosted classifier cascade. (a) Viola and Jones features. (b) Lienhart and Maydt features.

Algorithm Train-Cascade

Given: $\mathcal{P}_0$, a set of k positive examples
$\mathcal{N}_0$, an initial set of k negative examples
α_p, the desired true positive rate per stage
α_f, the desired false positive rate per stage

Returns: C, a cascade of ensemble classifiers

$C \leftarrow H_0 \leftarrow$ AdaBoost($\mathcal{P}_0, \mathcal{N}_0, \alpha_p, \alpha_f$)

$i \leftarrow 0$

Repeat:

Test C on new, known-to-be-negative examples

$\mathcal{N}_i \leftarrow$ The top k false positives from test

$\mathcal{P}_i \leftarrow (\mathcal{P}_{i-1} -$ positives dropped by $H_{i-1})$

$H_i \leftarrow$ AdaBoost($\mathcal{P}_i, \mathcal{N}_i, \alpha_p, \alpha_f$)

$C \leftarrow C$ with H_i appended to the cascade

$i \leftarrow i + 1$

Until performance is "good enough"

Return C

รูปที่ 10: Cascade training algorithm. The algorithm utilizes the AdaBoost routine, which, given a training set $\langle \mathcal{P}, \mathcal{N} \rangle$, finds a combination of weak threshold classifiers obtaining a true positive rate of at least α_p and a false positive rate at most α_f on that training set.

3.3.1 Boosted classifier cascade

The core of our object detection system is the cascade of boosted classifiers originally proposed by Viola and Jones [15, 16] and later modified by Lienhart and Maydt [9, 8]. The cascade reduces processing time while preserving classifier accuracy through the use of a sequence of classifiers tuned for reasonably low false positive rates but extremely high detection rates. The cascade quickly rejects most detection windows unlikely to contain the object of interest and spends more compute time on the detection windows most likely to contain the object of interest.

Each stage in the cascade uses the "boosting" ensemble learning method [5] to induce a strong nonlinear classification rule that is a linear combination of the "votes" of multiple weak threshold classifiers, each considering the output of a single Haar wavelet-like filter at a fixed location in the detection window. Viola and Jones' original method [15] uses Freund and Shapire's "discrete" Adaboost algorithm [5] with a set of five types of Haar-like features for the weak threshold classifiers (รูปที่9(a)). Lienhart and colleagues' method [9, 8] uses the "gentle" Adaboost algorithm and additional Haar-like features (รูปที่9(b)). Here we refer to both types of classifier as a *V&J cascade*.

The first step in constructing a V&J cascade for object detection is to obtain a large training set of images containing or not containing the object of interest. We then extract an initial set $\mathcal{P}_0$ of positive detection windows and an initial set $\mathcal{N}_0$ of negative detection windows and execute the procedure Train-Cascade, detailed in รูปที่10.

3.3.2 Bayesian skin detector

Our skin detector is a Bayesian maximum likelihood classifier based on color histograms [18, 7]. The classifier estimates the class $S \in \{skin, nonskin\}$ of a single pixel based only on its observed color x measured in some

color space. This simple method is extremely efficient and surprisingly effective. We let

$$\hat{s} = \arg\max_{s} P(X = x \mid S = s),$$

where the likelihood $P(X \mid S)$ is modeled by a color histogram estimated from training data we obtained in a pilot study. Our color histograms have two dimensions, namely the hue and saturation axes of the HSV color space, which we quantize into $16^2 = 256$ bins.

3.3.3 Feature extraction

The output of the Bayesian skin detector is a binary image in which one value represents skin pixels and the other value represents non-skin pixels. We have found that a few simple features extracted from this binary image allow surprisingly accurate classification. The particular features we extract are:

3.1 The *area* (in pixels) of the largest connected component of skin pixels.

3.2 The length of the *perimeter* of the largest connected component of skin pixels.

3.3 The *eccentricity* of the largest connected component of skin pixels.

3.4 The number of pixels in the largest skin component intersecting the detection window *boundary*.

Clearly, when the area feature is especially large or especially small, the given image patch is unlikely to contain a hand. The perimeter length and eccentricity features provide additional information about the shape of the detected skin "blob." Finally, since a properly detected hand will only intersect the boundary of the detection window at the wrist, the boundary feature provides information about how wrist-like the boundary is.

When combined with a reasonable classifier, these four features are sufficient to correct most of the V&J cascade's mistakes. We next describe our classifier.

3.3.4 Mahalanobis classifier

Given a feature vector $\boldsymbol{x}$ consisting of the area, perimeter, eccentricity, and boundary features, we must determine whether $\boldsymbol{x}$ represents a true hand or a false positive. We tackle this problem by applying a threshold θ to the dissimilarity of the given feature vector $\boldsymbol{x}$ from the mean feature vector $\boldsymbol{\mu}$. Our dissimilarity measure is the Mahalanobis distance

$$d(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}).$$

Here the mean hand feature vector $\boldsymbol{\mu}$, covariance matrix Σ, and distance threshold θ are estimated from a training set.

Once we obtain a final classification for each possible detection window, the positively detected hands could then be forwarded to another component in an integrated application, for example a gesture recognition module. In the current paper we simply evaluate the efficacy of the proposed algorithm on a series of video segments.

3.3.5 Experimental methods

Here we describe an experiment in which we captured video sequences of humans walking in an indoor environment then evaluated the hand detection system on those video sequences.

รูปที่ 11: Example training images scaled to 24×24.

3.3.5.1 Data acquisition

For purposes of training and testing the hand detection system, we captured four video sequences of four different people walking in and out of a moderately cluttered laboratory environment. The video sequences were captured at 15 frames per second with an IEEE 1394 Web camera, and each sequence lasted approximately three minutes.

After video acquisition, we manually located all visible hands in every image of all four sequences, for a total of 2246 hands. We designated the first 2000 as training examples and reserved the remaining 246 for testing.

Our criteria for positioning the detection window on the hand was that that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of selection window (see รูปที่11 for examples).

As already described, the cascade learning algorithm, at step i, requires a set $\mathcal{N}_i$ of negative examples that do not contain hands for training. For this purpose, we randomly selected eight frames from the training data that did not contain and hands. The OpenCV implementation of the V&J cascade (see below) scans these eight images to produce new negative examples for training at each stage.

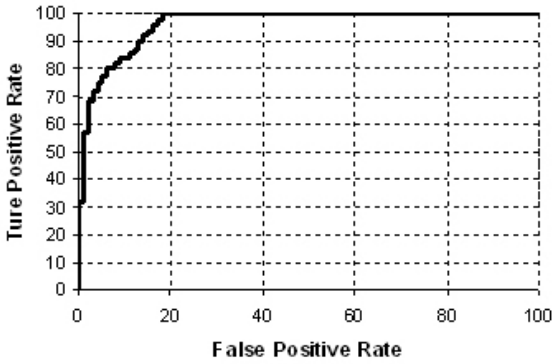
3.3.5.2 Boosted classifier training

To train the classifier cascade, we used Linehart and Maydt's approach [9] as implemented in OpenCV [4]. With 2000 positive hand examples and 8 background images for negative examples, we trained 30 stages using GentleBoost, $\alpha_p = 0.995$, and $\alpha_f = 0.5$. This took two weeks on a 3 GHz Pentium 4 PC with 1 GB of RAM.

We found that the classifier's training set performance peaked at 26 stages, so we used only the first 26 stages, comprising 763 weak classifiers, in subsequent analysis.

3.3.5.3 Skin detector training

To train the skin detector, we selected 10 images containing one or more humans from a set of independent video sequences captured under various lighting conditions at several different locations. We manually marked the skin pixels in each image, extracted the hue (H) and saturation (S) of the resulting 70,475 skin and 1,203,094 non-skin



รูปที่ 12: ROC curve between true positive rate and false positive rate

pixels, quantized the H-S values into 16 bins, and constructed two 2D histograms: one for skin pixels, and one for non-skin pixels.

3.3.5.4 Gathering data to train the post-processor

We ran OpenCV's performance testing utility, which had been modified to produce true positive and false positive image patches, on all labeled images from the training set with a detection window scale factor of 1.1. The resulting image patches were scaled to the standard size of 24×24. Then, we randomly selected 1000 true positive and 1000 false positive image patches for the training process of post-processor system.

3.3.5.5 Parameter estimation for the post-processor

To estimate the parameters of the Mahalanobis distance-based post processor, we applied skin detection to the 2000 training patches, eliminated all connected skin components smaller than 36 pixels, and filled in holes with a morphological opening operator. We then extracted features (area, eccentricity, perimeter, and boundary pixel count) for the largest connected skin blob in each of the 2000 patches. We randomly split the true positives into two groups, using the first 500 for mean and covariance calculation and using the remaining 500 to determine the best Mahalanobis distance threshold. Using the ROC curve in รูปที่12 to explore the tradeoff between true positives and false positives, we selected the point where the false positive rate was as low as possible (18%) while maintaining a 100% true positive rate.

3.3.6 Results

To analyse the performance of our system, we selected 4 frames from a video sequence which had never been used in the training process, and fed it to our system. We manually classified the detections at each stage of the system as a false positive or true positive. We found that the classifier cascade, by itself, performed relatively poorly, but that the post processing system was extremely effective in eliminating false positives produced by the classifier cascade.

ตารางที่1 shows that without the post-processing system, the false positive rate is too high for practical application. ตารางที่2, on the other hand, shows that when we add post processing to our system, the false positive rate decreases rapidly.

Image 1 and 2 in รูปที่13 illustrate example detection results from our complete system. In both images, we observe that regions on the person's head, especially in the chin and neck areas, are detected as hands by the

Image	Number of True Positives	Number of False Positives
1	1	22
2	2	35
3	1	19
4	1	18

ตารางที่ 1: Test results without post-processing

Image	Number of True Positives	Number of False Positives
1	1	2
2	2	5
3	1	6
4	1	2

ตารางที่ 2: Test results with post-processing

system. The most probable reason for this kind of false positive is that we did not include such image patches as negative examples during training of the boosted classifier cascade. We only used background images to generate negative examples. Unfortunately, not even the post-processing system can reject this kind of false positive because the shape of the skin blobs in those regions are very similar to the shape of skin blobs in patches actually containing hands. We believe that including human body parts other than hands as negative training examples will eliminate these types of false positives.

3.4 Extracting network-like structures from 2D images

The next machine vision module we have constructed is capable of extracting network-like structures from 2D images. The technique is based on quadratic snakes. We apply it to the task of extracting roads from satellite images, but the technique could be used in a variety of map-reading tasks.

3.4.1 Quadratic snake model

This section provides a brief overview of the quadratic snake proposed by Rochery et al. [12]. An *active contour* or *snake* is parametrically defined as

$$\gamma(p) = \begin{bmatrix} x(p) & y(p) \end{bmatrix}^T, \tag{11}$$

where p is the curvilinear abscissa of the contour and the vector $\begin{bmatrix} x(p) & y(p) \end{bmatrix}^T$ defines the Cartesian coordinates of the point $\gamma(p)$.

The energy functional is given by

$$E_s(\gamma) = E_g(\gamma) + \lambda E_i(\gamma), \tag{12}$$

where $E_g(\gamma)$ is the *geometric energy* and $E_i(\gamma)$ is the *image energy* of the contour γ. λ is a free parameter determining the relative importance of the two terms.

To apply the method to road extraction, we define the geometric energy functional to be

$$E_g(\gamma) = L(\gamma) + \alpha A(\gamma) - \frac{\beta}{2} \iint \boldsymbol{t}(p) \cdot \boldsymbol{t}(p') \; \Psi\left(\|\gamma(p) - \gamma(p')\|\right) \; dp \; dp', \tag{13}$$

Image 1 Image 2

รูปที่ 13: Preliminary results on test data from the hand detector.

where $L(\gamma)$ is the Euclidean length of γ over Ω, $A(\gamma)$ is the area enclosed by γ, $\boldsymbol{t}(p)$ is the unit-length tangent to γ at point p, and $\Psi(z)$, given the distance z between two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of each term. Clearly, for positive β, $E_g(\gamma)$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The interaction function $\Psi(z)$ is a smooth function expressing the radius of the region in which parallel tangents should be encouraged and anti-parallel tangents should be discouraged. $\Psi(\cdot)$ needs to be tuned for the road width and road width variability expected in an image. During snake evolution, weighting by $\Psi(z)$ in Equation 13 discourages two points with anti-parallel tangents (the opposite sides of a putative road) from coming too close to each other.

The image energy functional $E_i(\gamma)$ is defined as

$$\begin{aligned} E_i(\gamma) = & \int \boldsymbol{n}(p) \cdot \nabla I(\gamma(p))\ dp \\ & - \iint \boldsymbol{t}(p) \cdot \boldsymbol{t}(p')\ \nabla I(\gamma(p)) \cdot \nabla I(\gamma(p'))\ \Psi(\|\gamma(p) - \gamma(p')\|)\ dp\ dp', \end{aligned} \tag{14}$$

where $I : \Omega \rightarrow [0, 255]$ is an image and $\nabla I(\gamma(p))$ is the gradient of I evaluated at $\gamma(p)$.

The first linear term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand to enclose dark regions surrounded by light roads.[1] The quadratic term favors nearby point pairs with two different configurations, one with parallel tangents and parallel gradients and the other with anti-parallel tangents and anti-parallel gradients.

3.4.2 GVF external force

To encourage rapid convergence to a minimum of the energy functional, we adopt Xu and Prince's gradient vector field (GVF) [17] technique. The GVF is a vector field

$$V^{\text{GVF}}(\boldsymbol{x}) = \begin{bmatrix} u(\boldsymbol{x}) & v(\boldsymbol{x}) \end{bmatrix}^T$$

[1]For dark roads on a light background, we simply negate the linear term. In the rest of the paper, we assume light roads on dark background.

minimizing the energy functional

$$E(V^{\text{GVF}}) = \int_{\Omega} \mu(u_x^2(\boldsymbol{x}) + u_y^2(\boldsymbol{x}) + v_x^2(\boldsymbol{x}) + v_y^2(\boldsymbol{x})) + \|\nabla \tilde{I}(\boldsymbol{x})\|^2 \ \|V(\boldsymbol{x}) - \nabla \tilde{I}(\boldsymbol{x})\|^2 \ d\boldsymbol{x}, \tag{15}$$

where $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term inside the integral encourages a smooth vector field whereas the second term encourages fidelity to $\nabla \tilde{I}$. μ is a free parameter controlling the relative importance of the two terms.

We obtain $\tilde{I}$ using oriented filtering and Canny edge detection. We use elongated Laplacian of Gaussian filters that emphasize road-like structures, deemphasize non-road-like structures, and, to a certain extent, fill in short gaps where a road has low contrast with the background. The resulting binary Canny image is ideal because it only includes information about road-like edges that have survived sharpening by the oriented filters. The GVF field on top of the sharpened edge image is ideal because it points toward the road-like edges from a long distance, and, during snake evolution, it pushes the snake in an appropriate direction. This speeds evolution and makes it easier to find suitable parameters to obtain fast convergence.

3.4.3 Cooperating quadratic snakes

A single quadratic snake is unable to extract enclosed regions and multiple disconnected networks in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, merge, and disappear as necessary. Due to the area term $\alpha A(\gamma)$ in Equation 14, specifying the points on γ in a counterclockwise direction creates a *shrinking snake* and specifying the points on γ in a clockwise direction creates a *growing snake*. An enclosed region (loop or a grid cell) can be extracted effectively by initializing two snakes, one shrinking snake covering the whole road network and another growing snake inside the enclosed region.

Splitting a snake We split a snake into two snakes whenever two of its arms are squeezed too close together, i.e. when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other in both directions of traversal around the contour. d^{split} should be less than 2η, where η is the maximum step size.

Merging two snakes The merging algorithm selects points having high curvature and merges two snakes when 1) two selected points are closer than a prescribed minimal merging distance d^{merge}, 2) the traversal direction (clockwise or counterclockwise) of the two snakes is the same, and 3) the tangents at the two high curvature points are nearly anti-parallel. High curvature points are those with $\kappa_\gamma(p) > 0.6\kappa_\gamma^{\max}$, where $\kappa_\gamma^{\max}$ is the maximum curvature for any point on γ. When these conditions are satisfied, the two snakes are combined into a single snake by deleting the high curvature points and merging at the holes.

Limiting the merge decision to high curvature points ensures that merging only occurs if two snakes have semi-circular tips of their arms facing each other. It might seem that merging at low curvature points should also be permitted. However, as already explained, snakes normally repel each other due to the quadratic term in the internal energy (Equation 13). Consequently, low curvature segments can approach each other when high-gradient features allow the external energy to overcome the geometric energy. When this occurs for low curvature segments, the two snakes are most likely positioned on different sides of a road and merging should not be allowed. There are several other (rare) cases when snakes face each other at low curvature parts. However they should not be merged in those cases either.

(a) (b) (c) (d)

รูปที่ 14: Experimental results. (a) Original test image, from Google Earth. (b) Ground truth. (c) Segmentation after convergence with a single snake. (d) Segmentation after convergence with a snake family allowing split, delete, and merge.

Considering only the high curvature points also saves computational costs. In particular, the merging procedure requires computation of the angle between tangents only for the selected points. The number of those points usually does not exceed 10% of the total number of points.

The conditions that the traversal direction of two snakes should be the same and that the tangents at the two high curvature points should be anti-parallel reflect the fact that in our system, nested snakes form a tree structure. We initialize all the snakes at the first level with the same direction of traversal. The second level has the opposite direction of traversal and so on. When two snakes from the same level merge, we assign the resulting snake the same direction. Snakes from two consecutive levels do not merge. Growing and shrinking behavior is controlled by the area constant (α) and the weight on the geometric energy (β).

Deleting a snake A snake γ is deleted if it has perimeter less than L^{delete}.

3.4.4 Experimental results

To evaluate the proposed method for extraction of roads from satellite imagery, we used the image shown in รูปที่14(a). We manually determined the ground truth segmentation for the image (รูปที่14(b)), then compared the performance of a single quadratic snake with the performance of the multiple snake approach described in Section 2. We first performed several experiments to tune the system's parameters. Then, for each experimental condition, we initialized the contour to a rounded rectangle surrounding the entire image then ran the snake update equation until convergence.

The single snake took 579 iterations to converge, and the snake family took 500 iterations to converge. The resulting segmentations of the image into road and non-road pixels are shown in รูปที่14(c) and (d), respectively. Compared to the ground truth, the family of multiple cooperating snakes is considerably better than the single snake. The single snake obtained precision of 0.4641 and recall of 0.9324, compared to the snake family, which obtained precision 0.6233 and recall 0.9240.

The excessive detection of the closed region in รูปที่14(f) as road is due to the fact that it is surrounded by a loop, preventing the snake from entering the interior region. This problem can be easily solved with manual initialization of a clockwise snake inside the enclosed region.

หนังสืออ้างอิง

[1] N. Ayache and O.D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[2] British Broadcasting Corporation. Housestead's Fort (3D model), 2004. `http://www.bbc.co.uk/history/3d/houstead.shtml`.

[3] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[4] Intel Corporation. OpenCV Computer Vision Library (software). Open source software available at `http://sourceforge.net/projects/opencv/`.

[5] Y. Freund and R. E. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 5(1):119–139, 1997.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. University Press, Cambridge, UK, 2000.

[7] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.

[8] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs, 2002.

[9] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 900–903, 2002.

[10] A. Pope and D. Lowe. Vista: A software environment for computer vision research. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[11] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1988.

[12] M. Rochery, I. H. Jermyn, and J. Zerubia. Higher order active contours. *International Journal of Computer Vision*, 69(1):27–42, 2006.

[13] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Patter Recognition (CVPR '94)*, pages 593–600, 1994.

[14] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 2004. To appear.

[15] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[16] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[17] C. Xu and J. L. Prince. Gradient Vector Flow: A new external force for snakes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 66–71, 1997.

[18] B. D. Zarit, B. J. Super, and F. K. H. Quek. Comparison of five color models in skin pixel classification. In *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63, 1999.

[19] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

4. ผลที่ได้จากโครงการ

Our results have been published as follows:

- Dailey, M.N. and Parnichkun, M. Landmark-based simultaneous localization and mapping with stereo vision. *International Journal for Manufacturing Science & Technology*, 8(2), 17–22, 2006.
- Dailey, M.N. and Parnichkun, M. Simultaneous localization and mapping with stereo vision. In *Proceedings of the International Conference on Automation, Robotics, and Computer Vision*, 2006.
- Nakaguro, Y., Dailey, M.N., and Makhanov, S. SLAM with KLT point features. In *International Workshop on Advanced Imaging Technology (IWAIT)*, 2007.
- Dailey, M.N. and Bo Bo, N. Towards real-time hand tracking in crowded scenes. In *Proc. 2005 Asian Conference on Industrial Automation and Robotics*, pages F-70, 2005.
- Bo Bo, N., Dailey, M.N., and Uyyanonvara, B. Robust hand tracking in low-resolution video sequences. In *3rd International Conference on Advances in Computer Science and Technology (ACST 2007)*, 2007.
- Marikhu, R., Dailey, M.N., Makhanov, S., and Honda, K. A family of quadratic snakes for road extraction. In *8th Asian Conference on Computer Vision*, volume 4843 of Lecture Notes in Computer Science, pages 85-94, 2007.

Further results are reported in the following submitted papers:

- Bo Bo, N., Dailey, M.N., and Uyyanonvara, B. Natural-posed hand detection in low-resolution images. Submitted to *Songlanakarin Journal of Science and Technology*.
- Dailey, M.N., Makhanov, S., and Marikhu, R. Multiple quadratic snakes for road extraction. Submitted to *Image and Vision Computing*.

5. ภาคผนวก

Attached, please find the following reprints and preprints:

- Dailey, M.N. and Parnichkun, M. Landmark-based simultaneous localization and mapping with stereo vision. *International Journal for Manufacturing Science & Technology*, 8(2), 17–22, 2006.

- Dailey, M.N. and Parnichkun, M. Simultaneous localization and mapping with stereo vision. In *Proceedings of the International Conference on Automation, Robotics, and Computer Vision*, 2006.

- Nakaguro, Y., Dailey, M.N., and Makhanov, S. SLAM with KLT point features. In *International Workshop on Advanced Imaging Technology (IWAIT)*, 2007.

- Dailey, M.N. and Bo Bo, N. Towards real-time hand tracking in crowded scenes. In *Proc. 2005 Asian Conference on Industrial Automation and Robotics*, pages F-70, 2005.

- Bo Bo, N., Dailey, M.N., and Uyyanonvara, B. Robust hand tracking in low-resolution video sequences. In *3rd International Conference on Advances in Computer Science and Technology (ACST 2007)*, 2007.

- Bo Bo, N., Dailey, M.N., and Uyyanonvara, B. Natural-posed hand detection in low-resolution images. Submitted to *Songlanakarin Journal of Science and Technology*.

- Marikhu, R., Dailey, M.N., Makhanov, S., and Honda, K. A family of quadratic snakes for road extraction. In *8th Asian Conference on Computer Vision*, volume 4843 of Lecture Notes in Computer Science, pages 85-94, 2007.

- Dailey, M.N., Makhanov, S., and Marikhu, R. Multiple quadratic snakes for road extraction. Submitted to *Image and Vision Computing*.

Landmark-based Simultaneous Localization and Mapping with Stereo Vision

Matthew N. Dailey
Sirindhorn International Institute of Technology
Thammasat University
Pathumthani, Thailand 12121
mdailey@siit.tu.ac.th

Manukid Parnichkun
Mechatronics
Asian Institute of Technology
Pathumthani, Thailand 12120
manukid@ait.ac.th

Abstract— **SLAM, the problem of a mobile robot building a map of its environment while simultaneously having to determine its location within that map, is one of the current fundamental challenges of robotics. Although a great deal of success has been achieved with laser range finders as sensors and a planar world assumption, low-cost commercial robots will benefit from robust SLAM using cameras only. Towards the goal of a robust, six degree-of-freedom, vision-based SLAM algorithm, we describe the application of "FastSLAM" [1] to the problem of estimating a map from observations of 3D line segments using a trinocular stereo camera rig. By maintaining not only multiple hypotheses about the robot's position in space, but also maintaining multiple maps corresponding to those possible position hypothesis, the algorithm has the potential to overcome the systematic map errors induced by incorrect correspondence estimation.**

I. Introduction

Simultaneous localization and mapping (SLAM) is one of the most active areas in mobile robotics research. The task is to construct a metric map of the robot's workspace from noisy sensor readings, while simultaneously estimating the robot's location from the partial map and noisy odometry measurements.

The difficulty of the SLAM problem depends on the characteristics of the robot's environment, the characteristics of its sensors, and the level of map detail required by the application. The environment could be relatively benign, say, if it is indoors with flat floors and good traction for robot wheels. But it could also be quite subversive, for instance, in the case of aircraft and submarines.

The most common sensors in use today are laser range finders and video cameras. Lasers have some key benefits: they are accurate and provide range information directly. However, they are heavy, expensive, and primarily two-dimensional (although, of course, more than one plane could in principle be scanned). Cameras, on the other hand, are lightweight, small, and cheap, but are fairly difficult to work with. In particular, extracting range information from cameras normally requires triangulation from multiple perspectives or a great deal of prior knowledge.

The last characteristic, the level of detail required by the application, could be fairly coarse. In the case of a robotic wheelchair whose job is to get its occupant from place to place in a home, the detailed shape and size of pieces of furniture would probably not be relevant. But other applications would require an enormous amount of accurate detail in their maps. Imagine an autonomous land mine detector operating in a dense jungle. This robot would not only have to make certain that not a single square foot of jungle is missed, but would also require extremely detailed knowledge of the many obstructions preventing it from navigating between points.

To date, the majority of SLAM research has focused on two-dimensional maps acquired from laser range scanning. Several successful systems have been demonstrated; most use either a 2D occupancy grid [2], [3] or a database of landmarks in the plane [1], [4].

We are primarily interested in building real-time, vision-based SLAM systems for complex environments requiring motion estimates with 6 degrees of freedom. As for laser-based systems, the common choices for vision-based mapping are again either occupancy grids or landmark databases. But now we are dealing with 3D occupancy grids and databases of 3D landmarks. Although 3D volumetric grids have been shown to make realistic, detailed maps [5], [6], they require a prohibitive amount of memory and processing for large-scale environment mapping. So here we focus on SLAM algorithms for learning of 3D landmark databases from video data.

Zhang and Faugeras [7] were among the early pioneers in this area. Using mobile robots equipped with trinocular stereo camera rigs, they were able to successfully map small indoor environments. They modeled the world as a collection of straight line segments and tracked the lines' positional uncertainty with Kalman filters.

Se, Lowe, and Little [8] demonstrate the use of SIFT (scale invariant feature transform) point features as landmarks for the SLAM problem. They also used a trinocular stereo camera rig, and modeled the positional uncertainty of the landmarks with Kalman filters.

Davison, Cid, and Kita [9] demonstrate a single-camera SLAM algorithm capable of learning a set of 3D point features. Their algorithm uses an ingenious mix of particle distributions with the extended Kalman filter to overcome the initial range uncertainty arising from the use of only one camera.

All of the vision-based approaches estimate camera motion (either with or without the help of odometry) then update a map. Motion estimation and map updating both depend critically on solving a correspondence problem, either between

features in successive camera frames or between observed features and map features. This is clearly a problem when ambiguities lead to correspondence errors. If erroneous correspondences are used to estimate motion, motion estimates can be thrown off, causing the map to be updated from an incorrect position, in turn leading to systematic drift in the map.

In a series of recent papers (summarized in [1]), Montemerlo, Thrun, and colleagues have proposed an interesting way around this same problem for laser range finder-based SLAM systems. The idea is to maintain multiple possible maps corresponding to multiple possible data association hypotheses. Their algorithm, "FastSLAM," uses a particle filter for sampling from the space of possible robot paths, but unlike previous systems based on particle filters (e.g. [3]), *each particle* maintains its own map. The new approach yields impressive results in mapping a large outdoor environment, and with appropriate optimization, is suitable for real-time implementation.

In this paper, we explore the possibility of using a similar approach to maintain multiple correspondence hypotheses in the realm of 3D visual landmark-based SLAM. The landmarks in our case are 3D line segments obtained from a trinocular stereo vision system. We find that the method is indeed capable of maintaining the robot's location while constructing a consistent landmark map.

II. Landmark-based SLAM with stereo vision

Following [1], our algorithm maintains at time t a set of particles, each representing a possible robot path from time 0 to time t. Associated with each particle is a candidate position s_t for the robot at time t and a map, represented as a k-D tree of landmarks with associated Gaussian uncertainties.

In our system, after each robot motion u_t, a set of trinocular stereo images is captured, and a set z_t of landmark measurements (line segments) is extracted from those images. These line segment measurements, along with the measured motion u_t, are used to update each particle's map and position estimate.

A. Landmark extraction

Our algorithm assumes a calibrated stereo camera rig with three pinhole cameras. To simplify matching, we further assume that the three cameras share the same image plane, that the first and second cameras are rectified so that the epipolar lines correspond to the same rows in each image, and that the first and third cameras are rectified so that the epipolar lines correspond to the same columns in each image.

a) 2D line segment extraction: The basic 2D feature in our system is the line segment. We extract line segments using Canny's method [10] following the implementation in VISTA [11]. The edge detector first performs nonmaxima suppression, links the edge pixels into chains, and retains the strong edges with hysteresis. Once edge chains are extracted from the image, we approximate each chain by a sequence of line segments. Short line segments, indicating edges with high curvature, are simply discarded in the current system.

b) Stereo matching: We use a straightforward stereo matching algorithm similar to the approach of [7]. For each line segment in the reference image, we compute the segment's midpoint, then consider each segment intersecting that midpoint's epipolar line in the second image. Segments not meeting line orientation and disparity constraints are discarded. Each of these potential matches determines the location and orientation of a segment in the third image. If such a consistent segment is indeed found in the third image, the potential match is retained; otherwise it is discarded. If at the end of this process, we have one and only one consistent match, we assume it correct; otherwise, the reference image line segment is simply ignored.

c) 3D line representation: Now our goal is to estimate a three-dimensional line from the three observed two-dimensional lines. As we shall see in the next section, each landmark in our map is represented by a point with an associated Gaussian uncertainty. Furthermore, landmark positions and uncertainties are updated using the Kalman filter update rule. Since the Kalman filter update requires inversion of the measurement covariance matrix, the most straightforward approach is to use a *minimal* 3D line representation. In particular, we represent a line as a 4-vector $\mathbf{L} = (\sigma, \sigma, r, \sigma)^T$, where, assuming P is the projection of the origin onto $\mathbf{L}$, the parameters σ, σ, and r describe P in spherical coordinates, and σ is the rotation of $\mathbf{L}$ around the line through the origin and P.

d) 3D line estimation: Since 2D lines have two intrinsic parameters and 3D lines have four intrinsic parameters, we have 6 independent nonlinear equations in four unknowns. Assuming Gaussian measurement error in the image, maximum likelihood estimation of the 3D line's parameters becomes a nonlinear least squares problem, which we solve with Levenberg-Marquardt minimization [12]. As an initial estimate of the line's parameters, we use the 3D line (uniquely) determined by two of the 2D measurements.

e) Error propagation: Through each step of the 3D line estimation process, we maintain explicit Gaussian error estimates. We begin by assuming spherical Gaussian measurement error in the image with a standard deviation of one pixel. Arranging the n (x, y) coordinates of the pixels in a line as a column vector $\mathbf{x}$, the covariance of $\mathbf{x}$ is simply $\Sigma_{\mathbf{x}} = I_{2n\times 2n}$. Since the vector of parameters $\mathbf{l}$ describing the 2D line best fitting $\mathbf{x}$ is a nonlinear function $\mathbf{l} = \mathbf{f}(\mathbf{x})$, the covariance of $\mathbf{l}$ is $\Sigma_{\mathbf{l}} = J\Sigma_{\mathbf{x}}J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ evaluated at $\mathbf{x}$.

The maximum likelihood estimate of the 3D line $\hat{\mathbf{L}} = (\sigma, \sigma, r, \sigma)^T$ obtained from the three 2D line segments $\mathbf{l} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3)$ is clearly not a simple function, since it is computed by an iterative optimization procedure. However, if $\mathbf{l} = \mathbf{f}(\mathbf{L})$ is the function mapping from the parameter space to the measurement space, it turns out that, to first order, $\hat{\mathbf{L}}$ is a random variable with covariance matrix $(J^T\Sigma_{\mathbf{l}}J)^{\times 1}$, where J is the Jacobian matrix $\frac{\partial \mathbf{l}}{\partial \mathbf{L}}$ [13].

f) Numerical stability: Unfortunately, the minimal 4-parameter representation of 3D lines leads to covariance ma-

trices that are very nearly singular, creating numerical stability problems. To understand the issue, consider a horizontal line on the ground one meter to the right of the robot, running parallel to the robot's line of sight. If the line is observed at a distance of 10 meters, a small error in our estimate of the direction of that line translates to a large error in our estimate of r, the length of the projection of the origin (the robot's position) onto the line. At the same time, that small error in estimating the orientation of the line translates to a very small error in σ, the rotation in the plane of the projection of the origin onto the line. This situation means our error ellipses tend to be quite elongated with high condition numbers. To prevent numerical problems when the covariance matrices are inverted, we enforce constraints on the condition number and minimum singular value of each covariance matrix. In particular, we let USU^T be the singular value decomposition of the positive definite covariance matrix Σ. If σ_1 is the largest singular value of Σ, and σ_n is the smallest singular value of Σ, we construct an adjusted singular value matrix S' by adding to each of the singular values σ_i the smallest nonnegative quantity σ guaranteeing that the smallest adjusted singular value $\sigma'_n = \sigma_n + \sigma$ is above a small threshold and that the new condition number σ'_1/σ'_n is below another threshold. We then replace Σ with the better-conditioned matrix $US'U^T$.

g) Line transforms: Once the four-dimensional representation of an observed 3D line is estimated from a trinocular line correspondence, it is necessary to transform that line from camera coordinates into robot coordinates, since the reference camera is in general translated and rotated relative to the robot itself. It is also necessary to transform landmarks from robot coordinates into world coordinates, when the robot's position is determined, for instance, and from world coordinates back to robot coordinates, when a landmark in the map is considered as a possible match for an observed (robot coordinate) landmark. In each of these cases, the transformed line $\mathbf{L}' = \mathbf{t}(\mathbf{L})$ is computed as a nonlinear function of the original line, and the transformed line's covariance is propagated by $\Sigma_{\mathbf{L}'} = J\Sigma_{\mathbf{L}}J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{t}}{\partial \mathbf{L}}$ evaluated at $\mathbf{L}$.

B. Particle update

Here we follow Montemerlo and colleagues' "FastSLAM 1.0" algorithm [1], adapted to the case of line segments as landmarks. FastSLAM's goal is to estimate the posterior

$$p(s^t, \Sigma \mid z^t, u^t, n^t)$$

where s^t is the robot's path from time 0 to time t, Σ is the map (a list of landmark positions and covariances), z^t is the set of observed landmarks from time 0 to t, u^t is the robot's assumed motion from time 0 to t, and n^t is the set of correspondences between the observed features z^t and stored features σ^t in the map.

FastSLAM is efficient, despite estimating a posterior over the robot's path from time 0 to t, because the posterior can be factored

$$p(s^t, \Sigma \mid n^t, z^t, u^t) = p(s^t \mid n^t, z^t, u^t) \prod_{n=1}^{N} p(\sigma_n \mid s^t, n^t, z^t)$$

since the individual feature estimates in the map are conditionally independent given the robot's path and a correspondence between map features and observations.

In FastSLAM, the distribution $p(s^t \mid n^t, z^t, u^t)$ is approximated by a particle filter. Each particle in the filter represents one possible robot path s^t from time 0 to t. Since the map feature estimates $p(\sigma_n \mid s^t, n^t, z^t)$ themselves depend on the robot path, each particle also carries with it an extended Kalman filter (EKF) for each of the landmarks in the map.

Here we provide a capsule summary of the FastSLAM algorithm as adapted to our system. For details of FastSLAM, the reader should refer to [1]. At each time step t, the robot performs an action u_t, captures a set of images, and extracts a set of 3D line segments z_t from the image set as described in section II-A. Each particle m in the particle filter is then updated with the new robot action u_t and z_t as follows:

1) Using the particle's previous state estimate $s_{t\times 1}$, we draw a sample s_t from the motion model $p(s_t \mid s_{t\times 1}, u_t)$, which is approximated by a Gaussian distribution around the position $\hat{s}_t$ we would obtain if the robot exactly performed action u_t from position $s_{t\times 1}$.
2) Assuming the sampled position s_t, for each observed landmark z_{ti}, we find the most likely correspondence with a stored landmark, i.e.
$$\hat{n}_{ti} = \arg\max_{n_{ti}} p(z_{ti} \mid n_{ti}, \hat{n}^{t\times 1}, s^t, z^{t\times 1}, u^t)$$
3) For each observed landmark, if the probability of the observation given the most likely correspondence $p(z_{ti} \mid \hat{n}_{ti}, \hat{n}^{t\times 1}, s^t, z^{t\times 1}, u^t)$ is below threshold, we add it to the map as a new landmark. If the probability is above threshold, we combine the new observation with the old landmark position estimate using the extended Kalman filter update equation.
4) Over all the observed landmarks, we calculate the importance weight $w_t^{[m]}$, assuming conditional independence of the individual observations z_{ti}:
$$w_t^{[m]} = \prod_i p(z_{ti} \mid \hat{n}_{ti}, \hat{n}^{t\times 1}, s^t, z^{t\times 1}, u^t)$$

After the particles are updated, we then resample the particles, with replacement, proportional to the importance weights calculated in step 4. above. The updated particles with large weights, indicating a high degree of consistency between observed and stored landmarks, are more likely to survive the resampling step than those particles with small weights, indicating less consistency between observed and stored landmarks.

III. Experimental methods

We tested the vision-based SLAM algorithm in simulation with a virtual robot moving through a virtual world, rendered with OpenGL from a VRML model. Our simulator allows

us to test localization and mapping algorithms in arbitrarily complex environments, with a known ground truth (the VRML model itself). We plan to release this simulator as open source software in the near future.

For our SLAM experiments, we chose as an environment a publically-available 3D model of Housestead's fort, a Roman garrison from the 3rd century A.D. on Hadrian's Wall in Britain [14]. A sample view from our virtual trinocular stereo rig is shown in Figure 1.

This environment is an interesting test of vision-based SLAM algorithms because, on the one hand, it generates many long, strong, straight edges that should be useful for localization. On the other hand, it is highly textured, creating a large number of edges, and the textures are highly repetitive in many places, leading to many ambiguities for correspondence algorithms.

We teleoperated our virtual robot through this virtual world in a long loop of about 300m. At approximately 1m intervals, the virtual camera rig was instructed to capture a set of stills from its three cameras. To make the problem more challenging, we simulated the effects of a traveling on a rough outdoor surface, so that the robot's vertical (Z) position varied approximately ± 0.04m from 0, its pitch and roll varied ± 2.5 degrees from 0, and its yaw varied ± 3 degrees from its expected course.

To enable the robot to view a large portion of the nearby environment, while also accurately estimating the depth of objects at reasonable distance, we used a fairly long stereo baseline (1m) and a fairly wide angle lens (70 degrees horizontal field of view). A more realistic setup achieving similar accuracy would require using more than one camera rig with narrow fields of view and more narrow baselines.

We plan to release the data set with ground truth to interested researchers via the WWW.

IV. Results

We report the results of three experiments here. First, to determine whether the 3D line segment sensor is accurate enough to support map building assuming known positions, we ran the mapping algorithm as described earlier, but with the known "true" simulated robot positions rather than the positions estimated by the SLAM algorithm. This resulted in a 3D line map whose projection into the plane is shown in Figure 2(a).

In a second experiment, we sought to obtain a lower bound on performance. We ran the mapping algorithm as described, but rather than estimating the robot's position from sensor data, we simply assumed the robot motions as described by the simulated odometry measurements were correct. This resulted in an extremely bad 3D map, whose projection into the plane is shown in Figure 2(b). Clearly, this map would be nearly useless for navigation.

In the final, third experiment, we ran our adaptation of FastSLAM on the same sensor data. We used 100 particles, each estimating the robot's position with 6 degrees of freedom. The result of one run is shown in Figure 2(c). Although the map is far from the "perfect" map of Figure 2(a), it is clear that the SLAM algorithm enables recovery from some of the distortions caused by bad odometry data in Figure 2(b).

V. Conclusion

In this paper, we have shown that Thrun, Montemerlo, and colleagues' FastSLAM algorithm, originally designed for 2D laser-based SLAM, is quite feasible for vision-based SLAM with 3D lines as landmarks. With the help of a straightforward line segment correspondence algorithm, the trinocular stereo camera sensor produces 3D landmarks suitable for use by FastSLAM's particle filter for weighting the importance of sampled robot positions. By maintaining multiple robot path hypotheses and a separate maximum likelihood map for each hypothesized path, the algorithm can potentially overcome the thorny problem of systematic error induced by erroneous correspondences.

However, there are serious issues to address. With a good 3D line representation and a sufficiently large number of particles in the filter, it should be possible to achieve, very nearly, the "perfect" map shown in Figure 2(a). The main problem is that in its current implementation, our algorithm is too slow. The run time is dominated by particle update step 2), namely the determination of the most likely correspondence for an observed landmark. This is partially due to the fact that we do not currently attempt to keep the k-D trees storing our landmark databases balanced, but even more so due to the fact that the measurement error for the 4-dimensional representation of 3D lines tends to be large, for the same reasons causing the numerical instability in Section II-A. This means that a large fraction of landmarks in the k-D tree must be considered as potentially corresponding features, and the full comparison is fairly expensive. The large number of observation-map feature comparisons in turn limits the number of particles we can use while maintaining a run time within range of real time performance. Thrun et al. [1] cite good performance for their FastSLAM 1.0 algorithm with 100 particles, but their problem is only a 3 degree of freedom problem, whereas ours is a 6 degree of freedom problem, most likely requiring far more particles to achieve similar performance.

With these issues in mind, in our current research, we are currently exploring better 3D line representations, as well as the use of Montemerlo, Thrun, and colleagues' "FastSLAM 2.0" algorithm, which promises a dramatic decrease in the number of particles necessary by focusing the particle filter's proposal distribution in regions more likely to be consistent with the observed sensor measurements.

Acknowledgments

This research was supported by Thailand Research Fund (TRF) grant MRG4780209 to MND. Some of the simulation software used in this project was donated by Vision Robotics Corporation of San Diego, CA USA. We are grateful to Thavida Maneewarn for suggestions on covariance matrix regularization.

(a) (b) (c)

Fig. 1. Sample trinocular image set captured in simulation. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

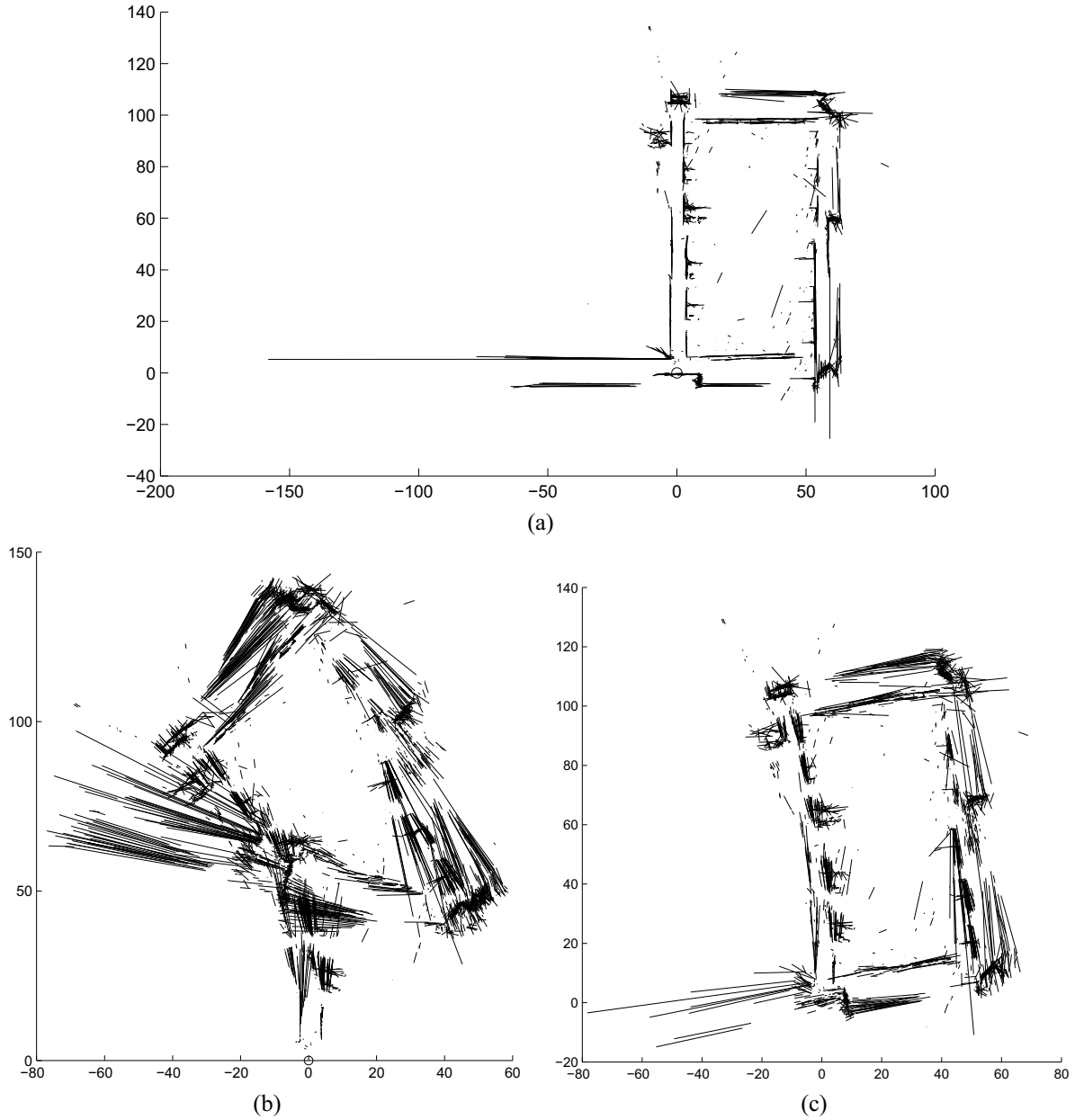


Fig. 2. Mapping results for Housestead's fort. (a) 2D projection of the 3D map obtained with perfect localization. (b) Raw map obtained assuming the location predicted by odometry. (c) Map estimated by the vision-based SLAM algorithm.

REFERENCES

[1] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An effi cient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, 2004 (in press).

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1–2, pp. 99–141, 2001.

[3] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and and 3D mapping," in *IEEE Conference on Robotics and Automation*, vol. 1, 2000, pp. 321–328.

[4] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[5] H. Moravec, "DARPA MARS program research progress: Robust navigation by probabilistic volumetric sensing," Carnegie Mellon University, Tech. Rep., 2002, `http://www.ri.cmu.edu/~hpm/project.archive/robot.papers/2002/ARPA.MARS/Report.0202.html`.

[6] M. Garcia and A. Solanas, "3D simultaneous localization and modeling from stereo vision," in *IEEE Conference on Robotics and Automation*, 2004, pp. 847–853.

[7] Z. Zhang and O. Faugeras, *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

[8] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *The International Journal of Robotics Research*, vol. 21, no. 8, pp. 735–758, 2002.

[9] A. Davison, Y. Cid, and N. Kita, "Real-time 3D SLAM with wide-angle vision," in *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[10] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, 1986.

[11] A. Pope and D. Lowe, "Vista: A software environment for computer vision research," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[12] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C*. Cambridge: Cambridge University Press, 1988.

[13] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: University Press, 2000.

[14] British Broadcasting Corporation, "Housestead's fort (3d model)," 2004, `http://www.bbc.co.uk/history/3d/houstead.shtml`.

Simultaneous Localization and Mapping with Stereo Vision

Matthew N. Dailey
Computer Science and Information Management
Asian Institute of Technology
Pathumthani, Thailand
Email: mdailey@ait.ac.th

Manukid Parnichkun
Mechatronics
Asian Institute of Technology
Pathumthani, Thailand
Email: manukid@ait.ac.th

***Abstract*— In the simultaneous localization and mapping (SLAM) problem, a mobile robot must build a map of its environment while simultaneously determining its location within that map. We propose a new algorithm, for visual SLAM (VSLAM), in which the robot's only sensory information is video imagery. Our approach combines stereo vision with a popular sequential Monte Carlo (SMC) algorithm, the Rao-Blackwellised particle filter, to simultaneously explore multiple hypotheses about the robot's six degree-of-freedom trajectory through space and maintain a distinct stochastic map for each of those candidate trajectories. We demonstrate the algorithm's effectiveness in mapping a large outdoor virtual reality environment in the presence of odometry error.**

***Keywords*—Localization, mapping, stereo vision, Rao-Blackwellised particle filter, visual landmarks**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is the problem of a mobile robot constructing a metric map from noisy sensor readings while simultaneously estimating its location from the partial map and noisy odometry measurements. SLAM is one of the fundamental challenges for mobile robotics research. Altough recent years have seen great advances in 2D mapping with laser range finders, exclusively vision-based SLAM (VSLAM) is still limited to relatively small scale, highly structured indoor environments.

We are interested in taking VSLAM beyond the typical office building environment into larger, but still structured, environments such as college campuses, office parks, and shopping malls. Potential application areas include security, inspection, landscape maintenance, agriculture, and personal service.

Achieving this goal without giving the robot an a-priori map requires new technology. The majority of vision-based SLAM research to date has focused on automatic construction of occupancy grids or topological maps (see [8] for a survey), both of which are inappropriate for large-scale metric mapping. The ideal approach would construct a *sparse* 3D representation of the environment.

Early VSLAM systems did use sparse features, but they typically compressed the map to 2D. For example, Kriegman, Triendl, and Binford's system [11] uses a stereo sensor to extract vertical lines from the environment. Observed lines are used to reduce odometric uncertainty using an extended Kalman filter (EKF), then the observations are in turn used to update an environment map containing 2D point features representing the observed vertical lines. Yagi, Nishizawa, and Yachida's system [21] took a similar approach but used a single omnidirectiona vision sensor and accumulation of measurements over time, rather than stereo, to determine the positions of vertical line landmarks. These systems and others have amply demonstrated the efficacy of VSLAM based on line landmarks in constrained indoor environments with smooth floors.

Faugeras and colleagues [1], [22] were the first to develop a VSLAM system storing a sparse 3D map. Their system first constructs a "local" 3D line segment map of the current scene using trinocular stereo. It explicitly represents the uncertainty about each feature's robot-relative pose in the form of a covariance matrix. The new local map is registered against the current global map and used to update an estimate of the robot's position using an EKF. Finally, assuming the robot's position, the global map is be updated with the freshly observed features, again using EKFs.

Se, Lowe, and Little [16] demonstrate the use of SIFT (scale invariant feature transform) point features as landmarks for the VSLAM problem. Their system also uses a trinocular stereo camera rig and models the positional uncertainty of the landmarks with Kalman filters.

Sim and Dudek [17] take a different approach; rather than prespecifying the features (lines, points, corners, and so on) that should be used for map building and localization, their system learns generative models for the appearance of salient features during exploration.

Until quite recently, most VSLAM systems limited themselves by separating the motion estimation and map estimation problems. Typically, at each step, the robot's location would be estimated via Bayesian inference or some other estimation technique, then that position would be assumed for the map update. While this approach leads to fast algorithms, not considering alternative robot poses when estimating landmark positions is suboptimal. Other researchers in the robotics community took a formal probabilistic approach and explored the possibility of representing, at each point in time, the full joint posterior distribution over robot trajectories and landmark

1–4244–0342–1/06/$20.00

positions. Smith, Self, and Cheeseman [19] introduced the "stochastic map," which represents not only the positions of landmarks in the world with their associated uncertainties, but also the uncertainty of the robot's position, the covariance between each pair of landmarks, and the covariance between the robot's position and each landmark. This seminal theoretical work inspired many successful SLAM systems, e.g. [2], [7], [12]. In a particularly impressive demonstration of the power of the stochastic map approach, Davison and colleagues [5], [6] have solved the VSLAM problem with point landmarks extracted from a single camera without odometry. Their system runs in real time at 30 Hz.

While the stochastic map very accurately represents all of the available information about landmark and robot positions (within the limits of the Gaussian approximation), the method unfortunately cannot scale to the thousands of landmarks needed for large-scale environments, due to the size of the full covariance matrix.

Murphy [13], however, recognized that in SLAM, map elements are conditionally independent given the robot's trajectory through time. He used this insight in the design of the Rao-Blackwellised particle filter (RBPF), in which the joint posterior over robot trajectories and maps is represented by a set of samples or particles, each particle containing one possible robot trajectory and the corresponding stochastic map. The fact that the robot's trajectory is fixed for a given particle has an important consequence: all of the covariances between different map elements in the stochastic map become 0. For a landmark map, this means the covariance matrix for each individual landmark is sufficient to represent all of the available knowledge of the environment.

Murphy only demonstrated the RBPF on a toy problem, but more recent work has applied the technique to the real world with immense success. Montemerlo, Thrun, and colleagues [20] use the RBPF and 2D point landmarks measured by a laser scanner to construct large-scale 2D maps. In their system, each particle represents a possible robot trajectory, set of data associations, and landmark map. The maps are stored in a tree structure that allows sharing subtrees between particles, allowing a real-time implementation that scales to thousands of landmarks. Eliazar and Parr [9] also use the RBPF and a laser scanner for SLAM, but build a 2D occupancy grid rather than a landmark database. Their algorithm also requires a sophisticated data structure that allows sharing maps between particles.

Even more recently, researchers have begun to apply the RBPF to the VSLAM problem. Sim et al. [18] extract SIFT point features from stereo data and combine the observations with visual odometry to build 3D landmark maps. According to the authors, this state-of-the art system has constructed the largest and most detailed VSLAM map ever, in a large indoor laboratory environment.

In our work, we take a similar approach, combining the RBPF with vision sensors, except that we use 3D line segments for localization and map building, rather than the more commonly used point features [18], [20]. Line parameters can be estimated more accurately than points, since the estimate incorporates more observed data. This means it may be possible to obtain more accurate robot localization from line landmarks than point landmarks, depending on the characteristics of the robot's workspace. Lines also provide more information about the environment's geometry than do points, allowing more sophisticated inference about the structure of the world. However, lines also have an important disadvantage with respect to points: they are less distinctive, making it more difficult to find correct correspondences between a set of observed lines and the lines in a stored model. We overcome this difficulty by sampling many possible poses from the robot's motion model, obtaining a different possible observation-model correspondence given each robot pose, and allowing the "fittest" correspondences to survive in the particle filter.

Our algorithm is called VL-SLAM (Visual Line-based SLAM). Here we describe VL-SLAM and demonstrate its effectiveness in a series of experiments. The main contributions of this paper are 1) an effective sensor model for line landmarks obtained from a stereo camera rig, 2) a new proposal distribution for the RBPF that overcomes the limitations imposed by highly uncertain correspondences, and 3) experimental evidence of the feasibility of VL-SLAM using realistic, albeit synthetic, data.

II. VL-SLAM

VL-SLAM is based on the "FastSLAM" family of algorithms proposed by Montemerlo, Thrun and colleagues [20]. At each point $t \in 1 \ldots T$, the robot performs an action u_t taking it from position s_{t-1} to s_t and uses its sensors to obtain an observation z_t. We seek a recursive estimate of

$$p(s_{0:t}, \Theta \mid u_{1:t}, z_{1:t}) \tag{1}$$

where Θ is a map containing the positions of each of a set of point landmarks. Rather than estimate the distribution (1) analytically, we approximate the posterior with a discrete set of M_t samples (sometimes called particles)

$$\left\{ < s_{0:t}^{[m]}, \Theta_{0:t}^{[m]} >, \text{ where each index } m \in 1 \ldots M_t \right\}. \tag{2}$$

Here $s_{0:t}^{[m]}$ is the specific robot trajectory from time 0 to time t associated with particle m, and $\Theta_{0:t}^{[m]}$ is the stochastic landmark map associated with particle m (the map is derived from $s_{0:t}^{[m]}$, $z_{1:t}$, and $u_{1:t}$). FastSLAM (and VL-SLAM) use the sequential Monte Carlo techniques of sequential importance sampling and importance resampling. First, for each particle, we sample from some *proposal distribution*

$$\pi(s_{0:t}, \Theta_{0:t} \mid z_{1:t}, u_{1:t}) \tag{3}$$

to obtain a temporary set of particles for time t, then evaluate the *importance weight* $w^{[m]}$ for each temporary particle, where

$$w(s_{0:t}, \Theta_{0:t}) = \frac{p(s_{0:t}, \Theta_{0:t} \mid u_{1:t}, z_{1:t})}{\pi(s_{0:t}, \Theta_{0:t} \mid u_{1:t}, z_{1,t})}. \tag{4}$$

The importance weights are normalized to sum to 1, then we sample M_t particles, with replacement, from the temporary particle set according to the normalized weights.

VL-SLAM extends FastSLAM with a new sensor model for 3D line segments and a new proposal distribution $\pi(\cdot)$ appropriate for environments with highly ambiguous observation-model correspondences. We first describe the 3D line segment sensor model then VL-SLAM proposal distribution.

A. VL-SLAM 3D Line Segment Sensor Model

After each robot motion u_t, a set of trinocular stereo images is captured, and a set z_t of landmark measurements (line segments) is extracted from those images. These line segment measurements, along with the measured motion u_t, are used to update each particle's map and position estimate.

Our system assumes a calibrated stereo camera rig with three pinhole cameras. It can handle general fundamental matrices (the images need not be perfectly rectified), but we do assume that one camera is roughly horizontally displaced and a second camera is roughly vertically displaced from a third (reference) camera.

The basic 2D feature in our system is the line segment. We extract line segments using Canny's method [4] following the implementation in VISTA [14]. The edge detector first performs nonmaxima suppression, links the edge pixels into chains, and retains the strong edges with hysteresis. Once edge chains are extracted from the image, we approximate each chain by a sequence of line segments. Short line segments, indicating edges with high curvature, are simply discarded in the current system.

We use a straightforward stereo matching algorithm similar to the approach of [22]. For each line segment in the reference image, we compute the segment's midpoint, then consider each segment intersecting that midpoint's epipolar line in the horizontally displaced image. Segments not meeting line orientation and disparity constraints are discarded. Each of these potential matches determines the location and orientation of a segment in the third image. If such a consistent segment is indeed found in the third image, the potential match is retained; otherwise it is discarded. If at the end of this process, we have one and only one consistent match, we assume it correct; otherwise, the reference image line segment is simply ignored.

Now our goal is to estimate a three-dimensional line from the three observed two-dimensional lines. Infinite lines have four instrinsic parameters, so it would make sense to use a four-dimensional representation of a lines. However, since VL-SLAM uses a Kalman filter to combine landmark observations, we require a linear parameterization of landmarks, and no linear four-dimensional representation of lines exists [1]. Instead we represent lines with six components: a 3D point representing the midpoint of the observed line segment and a 3D vector whose direction represents the direction of the line and whose length represents the distance from the line segment's midpoint to one of its endpoints. This 6D representation behaves well under linear combination, so long as the direction vectors are flipped to have a positive dot product.

First we obtain a maximum likelihood estimate of the infinite 3D line's parameters assuming Gaussian measurement error in the image using Levenberg-Marquardt minimization [15]. As an initial estimate of the line's parameters, we use the 3D line (uniquely) determined by two of the 2D line segment measurements. Once the infinite line has been estimated, we find the segment's extrema and midpoint using the observed data.

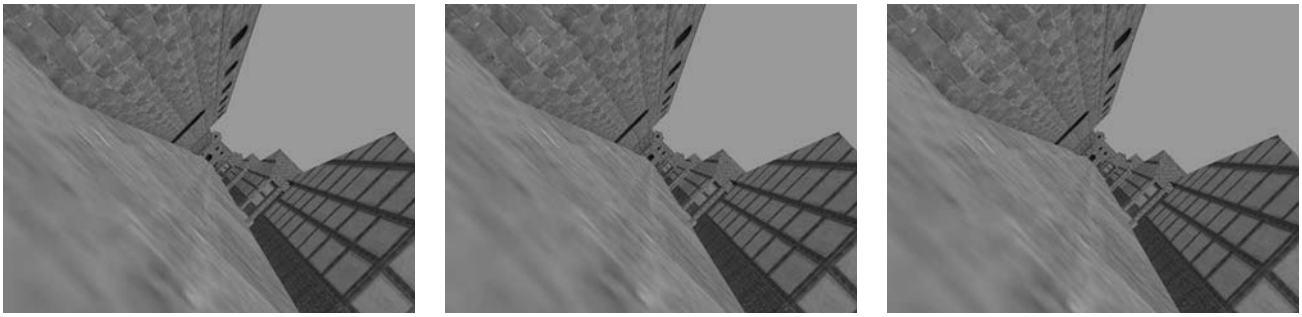
Through each step of the 3D line estimation process, we maintain explicit Gaussian error estimates. We begin by assuming spherical Gaussian measurement error in the image with a standard deviation of one pixel. Arranging the n (x, y) coordinates of the pixels in a line as a column vector $\mathbf{x}$, the covariance of $\mathbf{x}$ is simply $\Sigma_{\mathbf{x}} = I_{2n\times 2n}$. Since the vector of parameters $\mathbf{l}$ describing the 2D line best fitting $\mathbf{x}$ is a nonlinear function $\mathbf{l} = \mathbf{f}(\mathbf{x})$, the covariance of $\mathbf{l}$ is $\Sigma_{\mathbf{l}} = J\Sigma_{\mathbf{x}}J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ evaluated at $\mathbf{x}$.

The maximum likelihood estimate of the 3D line obtained from the three 2D line segments $\mathbf{l} = (\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3)$ is clearly not a simple function, since it is computed by an iterative optimization procedure. However, if $\mathbf{l} = \mathbf{f}(\mathbf{L})$ is the function mapping from the parameter space to the measurement space, it turns out that, to first order, $\hat{\mathbf{L}}$ is a random variable with covariance matrix $(J^T\Sigma_{\mathbf{l}}J)^{-1}$, where J is the Jacobian matrix $\frac{\partial \mathbf{l}}{\partial \mathbf{L}}$ [10]. The rank of the resulting covariance matrix is only four, however, so to constrain the remaining two degrees of freedom, we add to the rank-deficient covariance matrix a covariance matrix describing the expected error in our estimate of the segment's midpoint and another covariance matrix describing the expected error in our estimate of the segment's length. This gives us a full-rank covariance matrix that restricts matching line segments to not only be similar in terms of their supporting infinite line, but also to overlap and have similar length.

Once the six-dimensional representation of an observed 3D line is estimated from a trinocular line correspondence, it is necessary to transform that line from camera coordinates into robot coordinates, since the reference camera is in general translated and rotated relative to the robot itself. It is also necessary to transform landmarks from robot coordinates into world coordinates, when the robot's position is determined, for instance, and from world coordinates back to robot coordinates, when a landmark in the map is considered as a possible match for an observed (robot coordinate) landmark. In each of these cases, the transformed line $\mathbf{L}' = \mathbf{t}(\mathbf{L})$ is computed as a nonlinear function of the original line, and the transformed line's covariance is propagated by $\Sigma_{\mathbf{L}'} = J\Sigma_{\mathbf{L}}J^T$, where J is the Jacobian matrix $\frac{\partial \mathbf{t}}{\partial \mathbf{L}}$ evaluated at $\mathbf{L}$.

B. VL-SLAM Proposal Distribution

The proposal distribution $\pi(\cdot)$ (3) can be any distribution that is straightforward to sample from. However, it is best if $\pi(\cdot)$ closely approximates the full joint posterior (1), in which case the importance weights will be nearly uniform, and most particles will "survive" the resampling step. In FastSLAM 1.0 [20], the proposal distribution is simply $p(s_t \mid s_{t-1}, u_t)$, i.e. the motion model predicting s_t given a previous position s_{t-1} and action u_t. The authors observe that this proposal distribution,



(a) (b) (c)

Fig. 1. Sample trinocular image set captured in simulation. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

while simple to sample from, does not take into account the current observation z_t. This leads to FastSLAM 2.0, in which the proposal distribution is $p(s_t \mid s^{[m]}_{0:t-1}, \Theta^{[m]}_{0:t-1}, u_{1:t}, z_{1:t})$. This distribution takes not only the previous robot pose s_{t-1} and current action u_t into account, but also considers the current map $\Theta_{0:t-1}$ and new observation z_t. In the general case, this distribution could be quite difficult to sample from, but the authors find that by linearizing the sensor model and applying the Markov assumption, the proposal distribution can be approximated to first order by a Gaussian distribution whose mean and covariance can be calculated from known quantities, *if the correspondence between the observation z_t and the current map $\Theta_{0:t-1}$ is known*. When the correspondences are unknown (the usual case in SLAM), FastSLAM 2.0 assumes the maximum likelihood correspondence or draws a sample from a probability distribution over all possible correspondences. When the observations and landmarks are sparse, as is the case in the FastSLAM environment, this is straightforward, and FastSLAM 2.0 is much more successful than FastSLAM 1.0, since it uses the available set of particles wisely [20].

In VL-SLAM, however, each observation consists of on the order of 100 individual 3D line segments, and typically the landmark database contains several potential matches for each observed line. This means that it is impossible to consider even a small fraction of the possible correspondences for each particle. In practice, to limit the computational complexity, we must draw a single correspondence from the set of all possible correspondences without considering too many alternatives. But how can we choose a likely correspondence for a given observation?

In VL-SLAM, when propagating a particle forward from time $t-1$ to time t, we first fraw a sample s'_t from the robot's motion model to establish a correspondence between the observed line segments and the current map (resembling FastSLAM 1.0), then from that intermediate sample point, assuming the established correspondence, sample again, from the FastSLAM 2.0 proposal distribution. As in FastSLAM 2.0, the proposal distribution is closer to the full joint posterior distribution, concentrating more of the temporary particles in regions of high probability according to the full joint posterior.

To calculate the importance weights for the the VL-SLAM proposal distribution, we first introduce random variables n_t indicating the correspondence between the line segments observed at time t and the map. In VL-SLAM, the mth particle's map $\Theta^{[m]}_{0:t}$ is a deterministic function of the sampled trajectory $s^{[m]}_{0:t}$, the sampled correspondences $n^{[m]}_{1:t}$, and the observations $z_{1:t}$, so we rewrite the desired full joint posterior as

$$p(s_{0:t}, n_{1:t} \mid u_{1:t}, z_{1:t}). \tag{5}$$

Now, assuming we have a good estimate of the full joint posterior at time $t-1$, the VL-SLAM proposal distribution can be written as the product

$$\begin{aligned} p(s^{[m]}_t \mid n^{[m]}_t, s'^{[m]}_t, s^{[m]}_{0:t-1}, n^{[m]}_{0:t-1}, z_{1:t}, u_{1:t}) \times \\ p(n^{[m]}_t \mid s'^{[m]}_t s^{[m]}_{0:t-1}, n^{[m]}_{1:t-1}, z_{1:t-1}, u_{1:t}) \times \\ p(s'^{[m]}_t \mid s^{[m]}_{0:t-1}, n^{[m]}_{1:t-1}, z_{1:t-1}, u_{1:t}) \times \\ p(s^{[m]}_{0:t-1}, n^{[m]}_{0:t-1} \mid u_{1:t-1}, z_{1:t}), \end{aligned} \tag{6}$$

where s'_t represents the intermediate sample drawn from the motion model. For the mth particle, the importance weight is the ratio of the expressions in (5) and (6), which, with several applications of Bayes' rule and the Markov assumption, can be closely approximated as (details ommitted):

$$w^{[m]}_t = \frac{p(s^{[m]}_t \mid s^{[m]}_{t-1}, u_t) p(z_t \mid s^{[m]}_{0:t}, n^{[m]}_{1:t}, z_{1:t-1})}{p(s^{[m]}_t \mid z_t, s'^{[m]}_t, n^{[m]}_{1:t}, s^{[m]}_{0:t-1}, z_{1:t-1}, u_{1:t}) p(s'^{[m]}_t \mid s^{[m]}_{t-1}, u_t)} \tag{7}$$

Following [20], we linearize the sensor model and motion model, which leads to straightforward Gaussian approximations for each of the terms in (7).

Except for the sensor model and proposal distribution just described, VL-SLAM is similar to FastSLAM (see [20] for details). Once correspondences and the sampled pose are determined for an individual particle, each observed landmark is combined with its corresponding map landmark using an extended Kalman filter, or initialized as a new landmark in the map. To achieve fast search for landmarks corresponding to a given observation, each particle's map is stored in a binary k-D tree whose leaves are the 3D line segments with associated Gaussian uncertainties. However, to minimize total memory requirements and to enable constant-time copying of maps

during the resampling stage, the particles are allowed to share subtrees.

As we shall see in the next section, the diversity of possible correspondences introduced by the first sampling step (as in FastSLAM 1.0), combined with the use of the current observation z_t in the proposal distribution (as in FastSLAM 2.0), allows VL-SLAM to outperform both FastSLAM 1.0 and FastSLAM 2.0 on a challenging synthetic testbed.

III. Experimental Results

To enable rigorous testing of VL-SLAM in an environment with a precisely known ground truth, we implemented a virtual reality simulation allowing a virtual robot to move through a virtual world rendered with OpenGL from a VRML model. We chose as an environment a publically-available 3D model of Housestead's fort, a Roman garrison from the 3rd century A.D. on Hadrian's Wall in Britain [3]. A sample view from our virtual trinocular stereo rig is shown in Figure 1.

We teleoperated our virtual robot through this virtual world in a long loop of about 300m. At approximately 1m intervals, the virtual camera rig was instructed to capture a set of stills from its three cameras. The virtual camera models a real 10cm baseline, 70° field of view trinocular rig we recently built in our lab. To make the dataset somewhat challenging, we simulated the effects of a traveling on an imperfect outdoor surface, so that the robot's vertical (Z) position varied approximately ± 0.04m from 0, its pitch and roll varied ± 2.5 degrees from 0, and its yaw varied ± 3 degrees from its expected course.

This environment is an interesting testbed for VL-SLAM because, on the one hand, it generates many long, strong, straight edges that should be useful for localization. On the other hand, it is highly textured, creating a large number of edges, and the textures are highly repetitive in many places, leading to many ambiguities for correspondence algorithms. It is also large enough to preclude fine-grained grid-based techniques and noisy enough to preclude the use of flat-earth or three-degree-of-freedom assumptions.

We compared VL-SLAM with our own implementations of FastSLAM 1.0 and 2.0. As previously discussed, FastSLAM 2.0 was not designed to handle large observations with highly uncertain correspondences. In our implementation, we simply obtain the maximum likelihood correspondence assuming the robot is at the position obtained by propagating $s_{t-1}^{[m]}$ forward in time according to odometry to obtain $\hat{s}_t^{[m]}$. With this caveat about the FastSLAM 2.0 results, Figure 2 shows one measure of each algorithm's performance: the log-likelihood of the observation data given the best particle's robot trajectory sample and map; Figure 3 shows the final map according to the best VL-SLAM particle. All of the localization algorithms do much better than the baseline (odometry-only) algorithm. Due to its commitment to robot position $\hat{s}_t^{[m]}$ when determining correspondences in our implementation, FastSLAM 2.0 fares rather poorly. Since FastSLAM 1.0 samples from the motion model before obtaining a correspondence, it performs much better, but VL-SLAM, which combines the best features of both algorithms, outperforms them both.

IV. Conclusion

In this paper, we have demonstrated the feasibility of VL-SLAM on a challenging synthetic data set. The VL-SLAM proposal distribution improves on FastSLAM in environments with large numbers of ambiguous observations. However, Figure 2 shows that there is still improvement to be made: the log-likelihood of the observations given perfect localization is still much better than the log-likelihood of the observations under VL-SLAM's model. This means there is still information about the robot's position to be exploited in the observed data. This is also evidenced in Figure 3, which compares the VL-SLAM map to the map constructed with perfect knowledge of the robot's location. Although the map is locally fairly accurate, global drift occurs throughout the run, preventing the algorithm from closing the loop when the robot returns to its starting position. This is most likely due to an impoverished set of particles.

In future work, we plan to improve VL-SLAM's loop closing behavior and evaluate the algorithm on a variety of indoor and outdoor real-world data sets.

Acknowledgments

This research was supported by Thailand Research Fund grant MRG4780209 to MND.

References

[1] N. Ayache and D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[2] S. Borthwick and H. Durrant-Whyte. Simultaneous localisation and map building for autonomous guided vehicles. In *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, pages 1442–1447, 1994.

[3] British Broadcasting Corporation. Housestead's Fort (3D model), 2004. `http://www.bbc.co.uk/ history/3d/houstead.shtml`.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[5] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1403–1410, 2003.

[6] A. Davison, Y. Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[7] A. Davison and N. Kita. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 384–391, 2001.

[8] G. DeSouza and A. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[9] A. Eliazar and R. Parr. DP-SLAM 2.0. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. University Press, Cambridge, UK, 2000.

[11] D. Kriegman, F. Triendl, and T. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):792–803, 1989.

[12] P. Moutarlier and R. Chatila. Stochastic multisensory data fusion for mobile robot location and environment modeling. In *5th International Symposium on Robotics Research*, 1989.

[13] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[14] A. Pope and D. Lowe. Vista: A software environment for computer vision research. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[15] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1988.

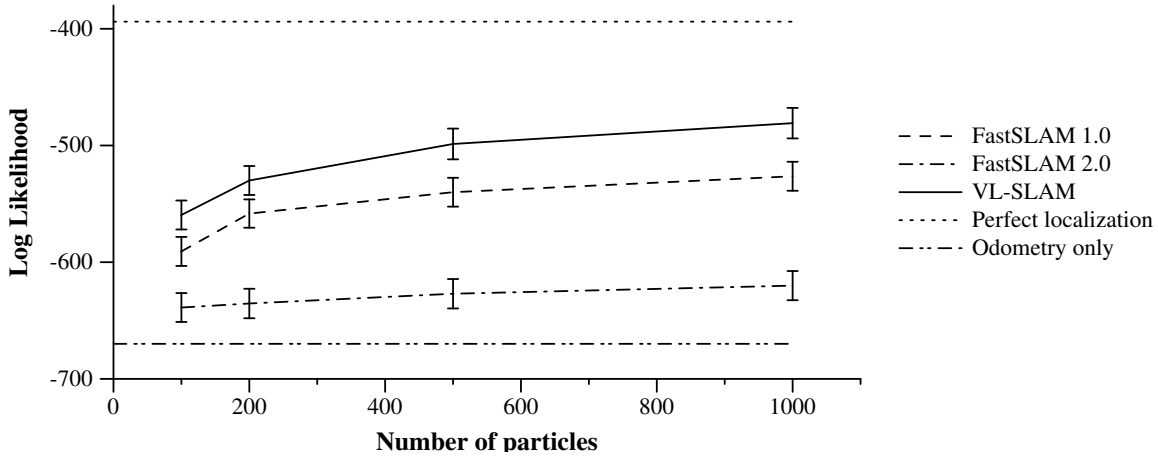


Fig. 2. Log likelihood of line observations according to the best particle's sampled robot position and map, averaged over 320 sets of observations.

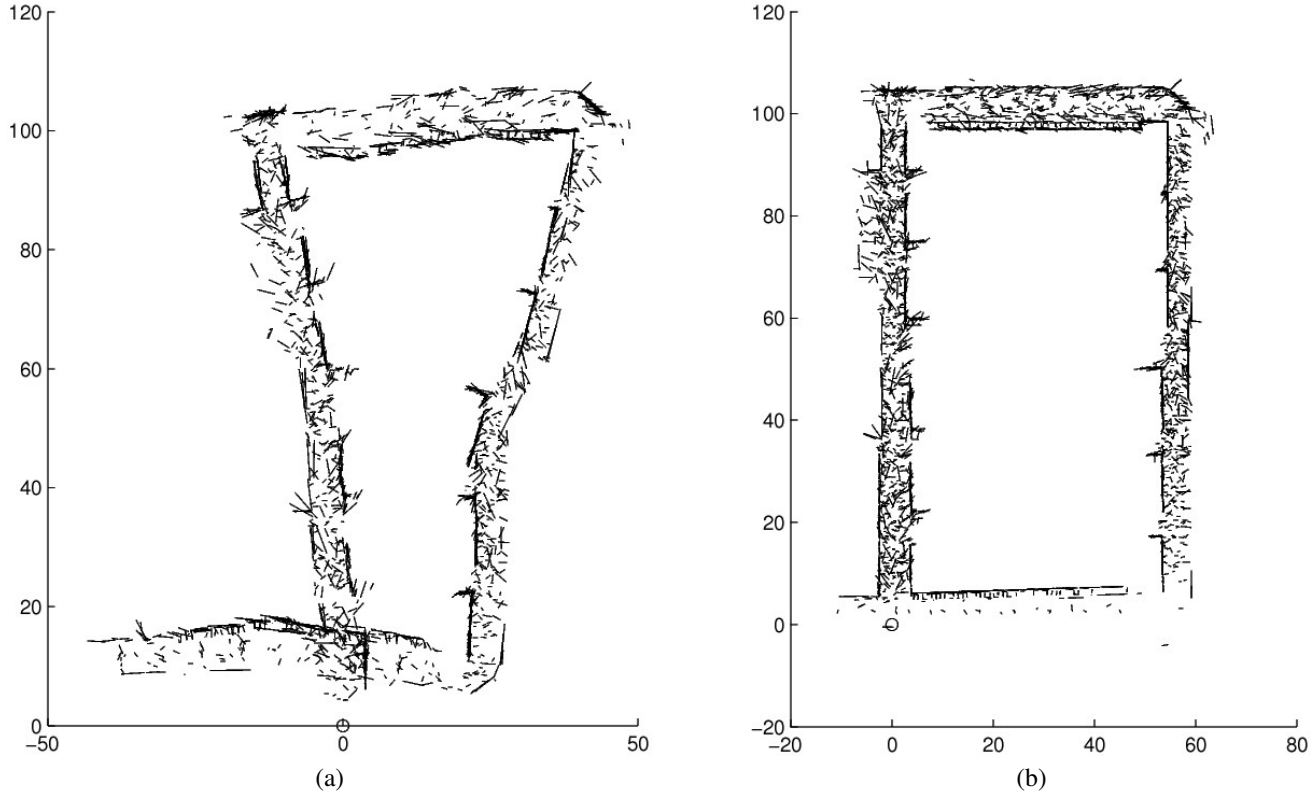


Fig. 3. Map constructed by VL-SLAM (a), compared to the the map assuming perfect knowledge of the robot's trajectory (b).

[16] S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21(8):735–758, 2002.

[17] R. Sim and G. Dudek. Learning generative models of scene features. *International Journal of Computer Vision*, 60(1):45–61, 2004.

[18] R. Sim, P. Elinas, M. Griffin, and J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.

[19] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[20] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 2004. To appear.

[21] Y. Yagi, Y. Nishizawa, and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor copis. *IEEE Transactions on Robotics and Automation*, 11(5):634–648, 1995.

[22] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

SLAM WITH KLT POINT FEATURES

Yoichi Nakaguro,[1] *Matthew Dailey,*[2] *Stanislav Makhanov*[1]

[1]Sirindhorn International Institute of Technology, Thammasat University
P.O. Box 22, Thammasat-Rangsit Post Office, Pathumthani 12121 Thailand

[2]Computer Science and Information Management, Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120 Thailand

E-mail: ynaka96@yahoo.com, mdailey@ait.ac.th, makhanov@siit.tu.ac.th

ABSTRACT

In the simultaneous localization and mapping (SLAM) problem, a mobile robot must localize itself in an unknown environment using its sensors and at the same time construct a map of that environment. While SLAM utilizing costly (expensive, heavy and slow) laser range finders as a sensor has been very successful in both indoor and outdoor environments, large-scale SLAM with cost-effective vision-based sensors has yet to be realized. In this paper, we evaluate the effectiveness of one possible low-cost vision-based approach to SLAM. We take 3D points constructed from Kanade-Lucas-Tomasi (KLT) image feature points in trinocular camera images as the basic landmarks for SLAM. We demonstrate the feasibility of KLT-based SLAM by conducting an experiment in a real indoor environment.

1. INTRODUCTION

Simultaneous localization and mapping (SLAM) is one of the fundamental problems in robotics. The problem is for a mobile robot, while moving around in some unknown environment, to use its sensors to construct a map of that unknown environment. SLAM is difficult mainly because the robot cannot determine its position precisely. It might have access to some positioning sensors such as wheel encoders, GPS, or a compass, but still, some kind of environmental feedback will always be necessary to help correct the error that inevitably exists in these sensor readings. The main sensors used in SLAM for this kind of feedback are laser range finders and video cameras.

We are interested in SLAM for constructing metric maps of large scale environments such as office buildings and mine fields. Laser range finders have been particularly successful sensors for these kinds of environments — see, for example, [14] — because lasers are extremely accurate. On the other hand, they are also heavy, expensive, and slow. In our work, we focus on the use of cameras as sensors due to their high speed, small size, and low cost.

Vision-based SLAM is an actively developing research area, but thus far most of the existing systems construct either occupancy grids or topological maps (see [5] for a survey), and these approaches are inappropriate for large scale metric mapping. For large scale metric maps, the simplest approach is to represent the world with a sparse collection of *landmarks*. These landmarks could be distinctive-looking 3D points or more complex objects such as lines, curves, corners, and so on. There have been several SLAM systems based on visual landmarks that work in small environments [1, 3, 4, 8, 15, 16], but thus far, there has been no successful robust, large-scale demonstration of vision-based SLAM.

Towards the goal of achieving large-scale metric vision-based SLAM, there has been some recent work on applying the efficient Rao-Blackwellised particle filter (RBPF) [9] as the underlying estimation algorithm and a stereo vision head as the sensor [2,12]. Both of these systems use Thrun et al.'s FastSLAM algorithm [14] for the RBPF to create sparse landmark maps organized by k-D trees for efficient search and modification.

We are particularly interested in combining multiple information sources, for example, line segments and distinctive points, to achieve robust large-scale vision-based SLAM at minimal cost. For point features, however, SIFT is computationally expensive; it requires construction of a scale space representation of each image, multiple convolutions, and extraction of a rich descriptor of the local image statistics around each point of interest. Combined with the computational complexity of maintaining many robot path estimates in the RBPF, systems based on SIFT and FastSLAM are going to be expensive or slow for several years to come.

In this paper, we explore the use of the KLT interest point detector [11] with trinocular stereo vision and the FastSLAM algorithm. On the one hand, KLT feature locations can be sensitive to noise, but on the other hand, they are quite lightweight in comparison with SIFT. We find that with the help of rather strict epipolar line constraints on the images obtained by a trinocular camera system, it is possible to choose only reliable points from a set of KLT points in an image set and use them to reconstruct 3D geometric point landmarks in an environment. We ran the FastSLAM algorithm with the 3D landmark point observations and verified the consistency of the result by comparing the performances with different number of particles used in the particle filter.

2. KLT-BASED FASTSLAM

Here we describe the application of FastSLAM [14] to the problem of vision-based SLAM with KLT point features as observations.

2.1. The FastSLAM algorithm

FastSLAM [14] is an elegant solution to the SLAM problem that maintains a full posterior over possible robot paths (as opposed to a maximum a posteriori estimate) using the RBPF [9]. The posterior distribution over possible robot paths is represented by a set of samples or particles, where each particle at time t represents one possible robot path up to time t, one possible series of data association assumptions for the sensor measurements up to time t, and a stochastic landmark map [13] based on those assumptions. Since each particle represents a particular robot path and a particular series of data association decisions up to time t, the observed landmarks are conditionally independent, so the posterior over landmark positions can be represented simply as a list of landmark estimates with associated uncertainties. The assumption of a particular robot path and particular series of data associations allows a representation of the map that is linear in the number of landmarks (the classical stochastic map is quadratic in the number of landmarks due to the correlations introduced by uncertain robot positions).

In this paper we adapt Thrun et al.'s "FastSLAM 1.0" [14] algorithm to the vision-based SLAM problem. At each time t, we seek a recursive estimate of

$$p(s_{0:t}, \Theta_t \mid u_{1:t}, z_{1:t}) \tag{1}$$

where $s_{0:t}$ is the robot's path from time 0 to time t, Θ_t is a map containing a set of landmarks, $u_{1:t}$ is a set of robot actions, and $z_{1:t}$ is a set of sensor observations. Each element s_i of $s_{0:t}$ is a vector describing the robot's pose at time i; we use a six degree of freedom representation for s_i.

The idea of the RBPF is to represent the posterior (Eq. 1) with a discrete set of M_t samples or particles

$$\left\{ \left\langle s_{0:t}^{[m]}, \Theta_t^{[m]} \right\rangle, \text{where each index } m \in 1, \ldots, M_t \right\}. \tag{2}$$

$s_{0:t}^{[m]}$ is a specific robot path associated with particle m, and $\Theta_t^{[m]}$ is the stochastic landmark map associated with particle m, derived from $s_{0:t}^{[m]}$, the robot actions $u_{1:t}$, and the observations $z_{1:t}$.

FastSLAM 1.0 uses sequential importance resampling, also known in the computer vision literature as the "condensation" algorithm [7]. At each time t, for each particle m, we sample from the *proposal distribution*

$$p(s_t \mid s_{t-1}^{[m]}, u_t)$$

to obtain a temporary set of particles for time t. Then, for each temporary particle m, we compute the *importance weight*

$$w^{[m]} \propto p(z_t \mid s_t^{[m]}, \Theta_{t-1}^{[m]})$$

and update the particle's map with z_t assuming $s_t^{[m]}$ to get $\Theta_t^{[m]}$. The importance weights are normalized to sum to 1, then we sample M_t particles, with replacement, from the temporary particle set according to the normalized weights. The result is a new set of particles (Eq. 2) that represents the posterior (Eq. 1) at time t.

Our approach is identical to planar FastSLAM 1.0 [14] except that we use a six degree of freedom motion model $p(s_t \mid s_{t-1}, u_t)$ and a 3D point sensor model $p(z_t \mid s_t, \Theta_{t-1})$ using landmarks derived from KLT features on a trinocular stereo vision rig. We now describe the trinocular stereo sensor in detail.

2.2. Trinocular KLT as a sensor model for FastSLAM

In FastSLAM, the sensor model is fully described by the conditional probability $p(z_t \mid s_t, \Theta_{t-1}, n_t)$, explicitly conditioning on n_t, the set of correspondences between observations z_t and landmarks stored in Θ_{t-1}. The distribution is assumed to be a deterministic measurement function $f(\Theta_{t-1}, s_t, n_t)$ corrupted by Gaussian noise.

In our case, the observations are sets of 3D points in robot-relative coordinates, estimated by triangulation with a trinocular stereo vision rig. Our 3D point extraction procedure begins by obtaining 2D KLT (Kanade-Lucas-Tomasi) corner features [11] from each of three calibrated images simultaneously captured by the trinocular camera rig. We then find sets of corresponding features across the three images and triangulate to obtain an estimate of the putative feature's 3D position relative to the robot.

The basic idea of using KLT as a 2D feature detector is to find points with a complex local gradient field. Complexity of the gradient field is measured by the smaller eigenvalue of the matrix

$$Z = \begin{pmatrix} g_x^2 & g_{xy} \\ g_{xy} & g_y^2 \end{pmatrix}$$

in which the quantities are integrals of the squared gradient (in the case of g_x^2 and g_y^2) or the integral of the product of x and y gradients (g_{xy}) in a neighborhood around the point of interest. A point is selected as a KLT feature if the smaller eigenvalue λ_2 of Z is a local maximum and above some threshold λ. The motivation is that image points meeting the criterion have a local gradient structure that cannot be described by a single eigenvector (as would be the case for a simple edge), but have a more complex corner-like structure that should be easy to detect under various imaging conditions.

After extracting a set of KLT feature points from each of the three images acquired at time t, we attempt to find triples of corresponding points as a necessary step prior to triangulation. For each KLT point $p_{1,i}$ detected in image 1, we search image 2 for potentially corresponding points. For each point $p_{2,j}$ in image 2 close enough to the epipolar line corresponding to $p_{1,i}$, we triangulate using the calibrated intrinsic and extrinsic parameters of the camera rig to predict the putative object feature's appearance in image 3. If a suitable KLT point $p_{3,k}$ exists in image 3, we consider the triple

$(p_{1,i}, p_{2,j}, p_{3,k})$ a candidate match and continue searching for other possible matches for $p_{1,i}$. If no consistent triples or more than one consistent triple is found for $p_{1,i}$, we throw it out. On the completion of this simple correspondence algorithm, we have a set of corresponding triples of 2D points that can then used for 3D estimation. Typically we begin with about 200 KLT points in each image and end up with about 20 corresponding triples.

The last step of obtaining a sensor measurement is to estimate a 3D landmark in robot-relative coordinates given each triple of corresponding 2D KLT points. For each correspondence (p_1, p_2, p_3), we obtain an initial estimate of the 3D position P by triangulating from p_1 and p_2, then we use the Levenburg-Marquardt nonlinear least squares optimization algorithm [10] to find the 3D position P maximizing the likelihood of the 2D observations (p_1, p_2, p_3) assuming spherical Gaussian error in the measured image coordinates. We also obtain an estimate of confidence in the 3D point landmark position P by propagating the assumed measurement error through the maximum likelihood estimation procedure using the standard first-order approximation [6].

After 2D feature detection, correspondence estimation, and triangulation, we obtain a set of 3D point landmark observations with associated error covariance matrices. The set of landmarks with covariances makes up z_t, the robot's observation at time t, which is input to FastSLAM. From this point on, our system is identical to Thrun et al.'s FastSLAM 1.0 algorithm [14].

3. EXPERIMENTAL METHODS

To test KLT-based FastSLAM, we performed an experiment in the Image and Vision Computing Laboratory at SIIT. The room is a typical laboratory with desks, bookshelves and computers. Figure 1 shows an image set captured in the lab with the 10cm-baseline trinocular camera rig that was used in the experiment.

In this experiment, rather than mounting the rig on a robot, we simulated robot motions by manually moving a camera tripod. The simulated robot's position s_t in world coordinates at time t is defined as a vector with six degrees of freedom $s_t = (x, y, z, \phi, \theta, \psi)^T$. Here the x and y axes span a plane parallel to the floor of the lab, and z is the vertical distance of the reference camera's origin from the ground plane. The remaining three variables represent the robot's orientation. ϕ, θ and ψ stand for pitch, roll and yaw of the camera rig, respectively. During the experiment, due to the flat floor, z, pitch, and roll was always equal to zero throughout the experiment.

The camera rig cannot move itself, so in the experiment we roughly pushed or rotated the rig by hand from its original position to the next destination position in order to emulate a real robot move. Since each move of the rig is not perfect, the rig normally reaches a position slightly away (in terms of x, y and yaw, we do not measure z, pitch and roll since they are assumed to be zero in the experiment) from its destination position. So we treated the difference of the original position and the desired destination position as

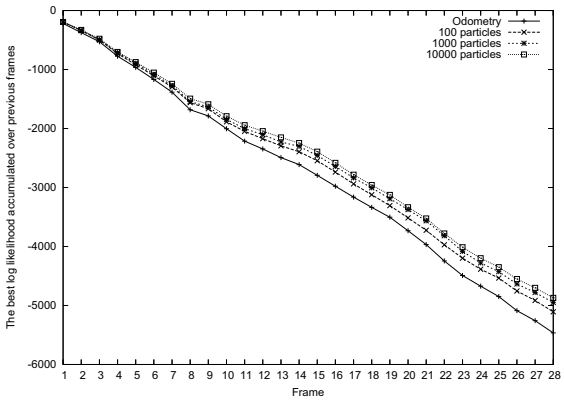


Fig. 2. Log likelihood of observation sequence given the model.

robot odometry, and the difference of the original position and the actually reached position as a true move. To make the experiment simpler, we composed camera rig odometry so that each odometric move involves only translation or only rotation. More specifically, odometry is of the form $(x, y, 0, 0, 0, 0)^T$ for translation, and $(0, 0, 0, 0, 0, \psi)^T$ for rotation.

The actual path of the camera rig consisted of 29 positions $D_0, D_1, \ldots, D_{28}$ marked on the floor of the lab. At first the rig was positioned at D_0, which we defined to be the origin of the world coordinate system. The rig was then moved to each destination. Along the way, at each position, we measured the true position $T_1, T_2, \ldots, T_{28}$ of the rig and captured a trinocular image set. The simulated odometry measurements $O_1, O_2, \ldots, O_{28}$ were computed as $O_i = D_i - T_{i-1}$.

In this indoor experiment the robot's path was approximately composed of a 4 meter forward translation from O_1 to O_{10} (roughly 0.4 meters per move), a 180 degree rotation from O_{11} to O_{18} (roughly 22.5 degrees per move), and finally a 4 meter forward translation from O_{19} to O_{28} (roughly 0.4 meters per move).

Image sets (29 frames including the initial state) and odometry (28 six dimensional vectors) were collected in the lab. They were used as the input for KLT-Based FastSLAM to estimate the path of the camera rig and generate a 3D metric map of the lab. We ran the algorithm with 100, 1000 and 10000 particles. In order to compare the algorithm's performance against a baseline, we also ran the same mapping algorithm purely using odometry as the estimate of the camera position.

4. RESULTS

Log likelihood is a measure of accuracy of the current landmark observation given the previous observation. It is given

(a) (b) (c)

Fig. 1. Trinocular image set captured in the lab. (a) Reference image. (b) Horizontally aligned image. (c) Vertically aligned image.

by

$$\begin{aligned} & ln\left(p(z_t \mid s_t^{[m]}, \Theta_{t-1}^{[m]})\right) \\ \sim \quad & -\frac{1}{2} ln\left|2\pi Q_t^{[m]}\right| - \frac{1}{2}(z_t - \hat{z}_t^{[m]})^T Q_t^{[m]-1}(z_t - \hat{z}_t^{[m]}) \end{aligned}$$

with the covariance

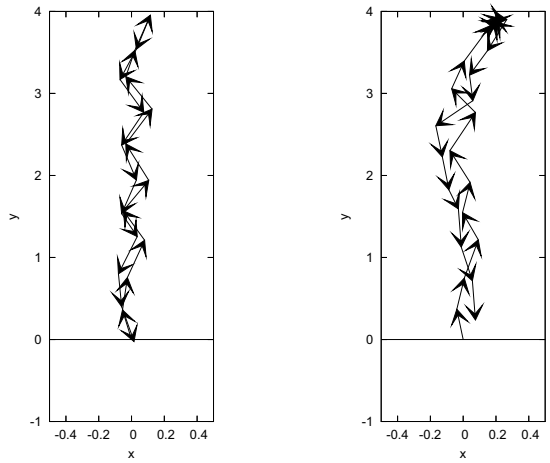
$$Q_t^{[m]} = G_t^{[m]T} \Sigma_{t-1}^{[m]} G_t^{[m]} + R_t$$

, where $\hat{z}_t$ is an estimation of the new observation z_t, Σ_{t-1} is the covariance of the landmark before the new observation is made, G_t is the Jacobian of the sensor model with respect to the landmark, and R_t is the covariance of the Gaussian noise of the new observation [14].

For each particle of each sequence of observation, we calculated the accumulated log likelihood, which is an addition of log likelihood over all the past sequences. It tells the degree of consistency of the map recorded in a particle. For each sequence of observation, we chose the particle that has the best (largest) value of accumulated log likelihood. The result is shown in Figure 2. As the number of the particle used in the FastSLAM algorithm increases, the accumulated log likelihood becomes better. The result tells that the particle filter is working properly in the experiment, i.e. with more particles, the better localization of the camera rig and estimate of landmark positions for each observation sequence is achieved.

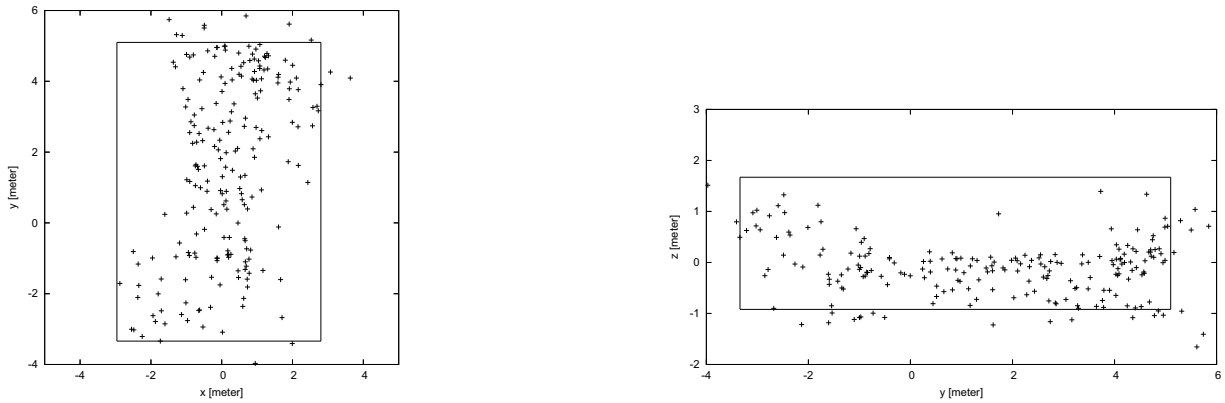
Figure 3 is 2D projections of the generated 3D map of the lab using 1000 particles. Only KLT point landmarks that were observed more than twice over all the observation sequences are plotted since landmarks observed only once tend to be noisy observations. Point landmarks in the map captured the actual distribution of edges and corners of objects seen in the lab.

In Figure 4, estimated path of the camera rig is shown.



(a) (b)

Fig. 4. Path of the camera rig projected onto the $x - y$ plane. Each move is represented as a vector. The rig was put at $(x, y) = (0, 0)$ initially and was moved 28 times while taking an image set after each move. (a) True path of the camera rig. (b) Estimated path of the camera rig using 1000 particles.



(a) (b)

Fig. 3. Projection of the 3D metric map into 2D planes. The boundary of the lab is shown as a rectangle in the figures. (a) KLT point landmarks projected into $x - y$ plane, the top view of landmarks. (b) KLT point landmarks projected into $y - z$ plane, the side view of landmarks.

5. CONCLUSION

In this paper, we have demonstrated the feasibility of KLT-based FastSLAM on a data set collected in a real indoor environment. We confirmed the positive effect of increasing the number of particles by looking at the accumulated log likelihood of particles per each sequence of observation. The distribution of 3D KLT point landmarks in the generated map globally represented the real distribution of edge and corner points of objects seen the lab.

However, there are noticeably many noisy cluttering landmark points in the 3D map which will hamper the navigation task based on the map. This happened mainly due to the fact that the calibration of the cameras on the rig was not ideally done in the time of the experiment. Poorly calibrated camera parameters give noisy estimation of the 3D landmark position derived from the 2D pixel coordinates of the landmark in each image of an image set.

The estimated path did not show any significant improvement against the path based on odometry nor on true measurement. One possible reason is that the odometry used in the experiment was so close to the truth that it was beyond the capability of the estimation algorithm to get the better estimate of the path. To verify it, we need more experiments with varying degrees of odometry error against the true measurement.

In the future work, we plan to analyze to what extent the calibration of the camera rig is affecting the resultant accuracy of estimation of landmark positions. We also seek to see the better estimation of camera rig positions over odometry by testing with various degrees of erroneous odometry. Finally, we plan a direct comparison of KLT and SIFT features as landmarks in SLAM.

6. ACKNOWLEDGMENTS

We thank Nico Hochgeschwender for help with software development. This research was supported by Thailand Research Fund grant MRG4780209 to MND.

7. REFERENCES

[1] N. Ayache and 0.D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[2] M. Dailey and M. Parnichkun. Simultaneous localization and mapping with stereo vision. In *Proceedings of the IEEE International Conference on Automation, Robotics, and Computer Vision (ICARCV)*, 2006. To appear.

[3] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1403–1410, 2003.

[4] A. Davison, Y. Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles*, 2004.

[5] G.N. DeSouza and A.C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, 2002.

[6] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. University Press, Cambridge, UK, 2000.

[7] M. Isard and A. Blake. Condensation — conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[8] D.J. Kriegman, F. Triendl, and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Transactions on Robotics and Automation*, 5(6):792–803, 1989.

[9] K. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems (NIPS)*, 1999.

[10] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1988.

[11] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Patter Recognition (CVPR '94)*, pages 593–600, 1994.

[12] R. Sim, P. Elinas, M. Griffin, and J.J. Little. Vision-based SLAM using the Rao-Blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR)*, 2005.

[13] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[14] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 2004. To appear.

[15] Y. Yagi, Y. Nishizawa, and M. Yachida. Map-based navigation for a mobile robot with omnidirectional image sensor copis. *IEEE Transactions on Robotics and Automation*, 11(5):634–648, 1995.

[16] Z. Zhang and O. Faugeras. *3D Dynamic Scene Analysis*. Springer-Verlag, 1992.

Towards Real-Time Hand Tracking In Crowded Scenes

Matthew N. Dailey and Nyan Bo Bo
Sirindhorn International Institute of Technology (SIIT)
Thammasat University
Pathumthani, Thailand 12121
Email: {mdailey,nyanbobo}@siit.tu.ac.th

Abstract— **Being able to detect and track human hands is one of the keys to understanding human goals, intentions, and actions. In this paper, we take the first steps towards real-time detection and tracking of human hands in dynamic crowded or cluttered scenes. We have built a prototype hand detection system based on Viola, Jones, and Snow's dynamic detector, which was originally constructed to detect and track pedestrians in outdoor surveillance imagery. The detector combines motion and appearance information to rapidly classify image sub-windows as either containing or not containing a hand. A preliminary evaluation of the system indicates that it has promise.**

I. Introduction

If we are ever to realize the goal of autonomous mobile robots able to interact with us in everyday life, we will have to overcome many obstacles. One of the most significant is the current lack of technology for perceiving and interpreting the structure of the world and the agents acting in it.

In this paper, we focus on a particular problem relevant to mobile robot applications in personal services, health care, and security: detecting and tracking human hands. A robot able to find and track human hands in real time would be able to accomplish many tasks. It could accept gesture-based commands from humans [**?**], [**?**], interact socially with humans, help patients in and out of bed, and so on. It could also detect and/or respond to security incidents, such as shoplifting, pick-pocketing, and assault.

Over the last 15 years or so, a great deal of research has focused on the problem of hand tracking. To date, the vast majority of systems have been aimed at empowering human computer interaction or sign language recognition. The early hand tracking systems relied on uncluttered static backgrounds, high resolution imagery, and manual initialization. These systems could track hands reliably and accurately, so long as the constraining assumptions held.

One of the first such systems was DigitEyes [**?**], which is able to track a human hand with 27 degrees of freedom at 10 frames per second (fps), given an already-initialized model. DigitEyes predicts the appearance of the hand in two stereo images given a previous state estimate then searches for the nearest matching features in the actual input. The actual measured feature positions are then used to obtain a maximum likelihood estimate of the hidden kinematic state by linearizing the transform from state to image appearance around the state estimate from the previous time. As long as there is no occlusion and the image features do not move too much between frames, the system can track a single hand with amazing accuracy.

Ahmad's tracker [**?**] was perhaps the first system to perform 3D hand tracking in real time with arbitrary background clutter. The tracker first performs color segmentation then applies a variety of classical computer vision techniques to identify the palm of the hand, its planar orientation, and the orientation of the fingers. It estimates depth changes using changes in the size of the palm. The system only works robustly when the hand is approximately parallel to the image plane.

Segen and Kumar built a more robust system [**?**] that is capable of tracking a hand in good imaging conditions through four different gestures. It provides a 10 degree of freedom estimate of hand's position: five for the 3D position and orientation of the thumb, and five for the 3D position and orientation of the index finger.

Many more hand tracking systems have appeared in recent years, but nearly all of them rely on fairly detailed models of the hand. Some, such as Triesch and von der Malsburg's [**?**], take a pattern recognition approach and are quite robust to clutter, but the systems still all assume a fairly high resolution view of the hand. This assumption is fine for "virtual mouse" and sign language applications, but it is unrealistic in the applications we have in mind. When our fictitious security robot spots the "slipping the gold watch into the pocket" gesture, for instance, the hand being observed might only be, say, five pixels wide in the camera image.

Systems like Pfinder [**?**] are probably more applicable to our task. Pfinder finds and tracks entire human bodies first, then finds the body parts. The body is assumed to be made up of a collection of colored blobs. A human's hands are normally easy to segment from the rest of the body using color, under favorable imaging conditions. We could in principle use a similar approach to first find the humans in the scene then find the hands of those humans.

But the task of finding humans in video streams is itself an extremely difficult problem, and the technique would preclude finding the hands of people that are mostly occluded by objects or other people. Also, reliable color information may not always be available, especially in low-light or surveillance camera applications. We take an entirely different approach in this paper. The goal is to detect multiple hands in a cluttered, crowded scene, *without first detecting the human bodies*.

Classifying, say, a 5×5 blob in a gray scale image as a hand and not a face or a piece of paper taped to the wall might at first seem to be impossible. Indeed, we do not know of any existing system capable of performing the task. However,

there is some hope: *motion* is an extremely salient source of contextual information that can help us distinguish "hand blobs" from other blobs. The power of motion as a cue to the identity of a moving object is well known, especially for human and animal motion. For instance, Polana and Nelson [**?**] find that the spatio-temporal profiles of low-level features can be used to automatically classify a scene as a human walking, running, jumping, and so on. Perhaps hands in typical crowded scenes have similarly characteristic motion profiles and can be detected according to their motion.

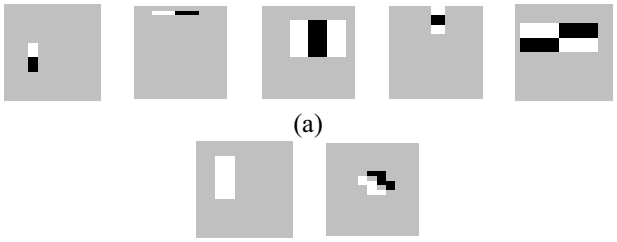
In this paper, we report on preliminary experiments applying Viola, Jones, and Snow's dynamic object detector [**?**] to the task of hand tracking in crowded scenes potentially containing many humans. The results thus far are promising, but there is much left to be done. First we describe the system, then we present preliminary results on a single video sequence, then we conclude with future directions for the research.

II. The Dynamic Object Detector

Viola and Jones created a great deal of excitement in the computer vision and machine learning communities when they first demonstrated the fastest-ever human face detection system [**?**]. The idea of the V&J detector, as in many other object detection systems, is to sweep a search window over the input image at multiple scales, and classify each window as either a member of the class or not a member of the class. The difficulty is twofold: the classifier must be extremely robust, to handle noise, partial occlusion, and background clutter, and it must be extremely fast, in order to allow classification of each possible window at frame rate. As an indication of the scale of the problem, for a 640×480 image with a 20×20 window moved two pixels on each test, at 25 different scales increasing by 10% at each step, a system must calculate a function $h : \Re^{400} \mapsto \{0, 1\}$ 349,740 times on every camera frame.

To solve the speed problem, the V&J detector is restricted to a class of filters that can be computed in constant time, regardless of the spatial extent of the filter, and is also arranged in a cascade that rapidly rejects most negative windows without much computation. The filters involve comparing sums of pixel intensities in rectangle-shaped regions of the image (see Figure 1(a) for example filters). To evaluate this filter, the sum of the image pixels falling within the black box is subtracted from the sum of the image pixels falling in the white box. Sums of image pixels within rectangles can be computed in constant time, regardless of the size of the rectangle, if an "integral image" [?] is precomputed, requiring just a few operations per pixel.

Filters like those in Figure 1 can be turned into classifiers if we apply a threshold and parity, for example, "if the difference in the sum of the pixels in the black box and white box is more (alternatively, less, according to the parity) than 20, then the image contains an example of our object." No single such classifier is good enough for any interesting object, but it is possible for multiple such "weak" classifiers to



(a)

(b)

Fig. 1. (a) Example V&J static filters. (b) Example VJ&S dynamic filters.

"vote" for whether the window contains an object. If enough of the weak classifiers vote for presence of the object, the classifier returns the value 1; otherwise it returns 0.

In the Viola and Jones scheme, many classifiers based on rectangle-feature filters are learned from a large set of examples using the AdaBoost algorithm [**?**]. AdaBoost is a greedy method for finding a good combination of weak classifiers. First one applies the weak learning algorithm to find the optimal filter for the set of training examples, using brute-force search over all possible filters if necessary. Then the training set is reweighted such that misclassified examples get bigger weights. Then a new classifier is trained on the reweighted examples. The final classifier is a weighted combination of the votes of the weak classifiers. Viola and Jones found that with a large training set (thousands of face and non-face images), AdaBoost could learn a very good detector containing 200 filters similar to those shown in Figure 1(a). But to speed up the amortized time to classify each image subwindow, they instead split the ensemble into several stages, rejecting windows early if they are extremely unlikely to contain the object of interest. This system has generated enormous interest in the the computer vision community. In fact, it has already been applied to hand gesture detection in high-resolution imagery [**?**].

Unfortunately, the V&J detector alone is not good enough for our particular application of detecting and tracking multiple hands in crowded scenes. Within the range of resolutions we are interested in, a hand could be simply a blob, five pixels or so wide. A local, static detector would produce far too many false positives to be of direct use. To find and track hands reliably with a low false positive rate, we need more context, either spatial or temporal. Viola, Jones, and Snow [?] have recently extended the V&J detector into the time domain, and we thought the new algorithm was a promising candidate for our hand detector.

The VJ&S dynamic detector uses the same techniques as the V&J detector, but in addition to static "appearance" filters applied directly to an image, a few *difference images* are also considered as candidates for filtering. The AdaBoost-based training procedure is now allowed to choose not only filters applied directly to $\mathcal{I}_t$ (the original image at time t) but also to five difference images (as listed in [?]):

$$\begin{aligned} \Delta_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1}) \\ U_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \uparrow) \end{aligned}$$

$$
\begin{aligned}
L_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \leftarrow) \\
R_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \rightarrow) \\
D_t &= \mathrm{abs}(\mathcal{I}_t - \mathcal{I}_{t+1} \downarrow)
\end{aligned}
$$

where the arrows indicate a shift of one pixel up, down, left, or right. This means a classifier could be built with, say, a rectangle feature in the difference image Δ to indicate a region of motion, or a rectangle feature with a low threshold specifying a lack of a response in the U image to specify an object apparently moving upwards. To further improve the ability of the weak learners to utilize these difference images, Viola, Jones, and Snow add filters like the ones shown in Figure 1(b) to the set of candidates for the weak learning algorithm. The additional motion information is sufficient to learn an extremely high-accuracy detector for pedestrians in video sequences [?].

Now that we have described the VJ&S dynamic detector, in the next section we describe its application to human hand detection.

III. Hand Detection With VJ&S

We have built a prototype hand detection system based on the VJ&S dynamic detector and performed a preliminary study of its efficacy on a real-world data set. The system is comprised of two main software modules: the *training* system, which does its work off-line, and the *run-time* system, which attempts to detect hands in a live video stream using a classifier produced by the training system.

A. Training

The first step in training is to collect a large set of example image pairs containing hands in natural poses. We take a straightforward approach: simply capture video of everyday human activity with a digital camera, ideally in several locations under various lighting conditions. Once the video data is collected, a human operator must manually select regions containing human hands from the image sequence. As this is a labor-intensive process, and a large number of examples are needed for training, we have created a simple graphical tool to automate as much of the process as possible. The main criterion is that the exact same window in both images of the pair must contain the hand. Subject to this constraint, we then try to ensure that the hands are roughly centered in the selection window, and that the hand should take up around 60% of the pixel area within the selection window. Sometimes these constraints conflict, as when there is a large amount of motion between two successive frames; then the selection window tends to contain the hand at one extreme end in the first image, and the other extreme end in the second image.[1] We repeat this process for every visible hand approximately 5 or more pixels in width, for every image pair in the sequence. We do not allow image pairs to overlap.

The learning algorithm also requires a set of negative examples, i.e., windows in image pairs that do not contain hands. We create an initial set of negative examples by simply choosing random window locations and sizes from the training video sequence then manually verifying that randomly-chosen windows do not happen to contain hands.

[1] We plan to correct this problem with a faster frame rate in the future.

Algorithm TRAIN-CASCADE
- Given: $\mathcal{P}$, a set of positive examples
  - $\mathcal{N}_0$, an initial set of negative examples
- Returns: C, a cascade of ensemble classifiers
- $C \leftarrow H_0 \leftarrow$ ADABOOST($\mathcal{P}, \mathcal{N}_0$)
- $i \leftarrow 0$
- Repeat:
  - Test C on new, known-to-be-negative examples
  - $\mathcal{N}_i \leftarrow$ The top k false positives from test
  - $H_i \leftarrow$ ADABOOST($\mathcal{P}, \mathcal{N}_i$)
  - $C \leftarrow C$ with H_i appended to the cascade
  - $i \leftarrow i + 1$
- Until performance is "good enough"
- Return C

Fig. 2. Cascade training algorithm.

Algorithm ADABOOST
- Given: $\mathcal{P}$, a set of positive examples
  - $\mathcal{N}$, a set of negative examples
  - WEAKLEARN, a weak learning algorithm
- Returns: H, an ensemble of weak classifiers
- Initialize weights w_i for each example i uniformly
- For $t \leftarrow 1$ to T:
  - Normalize weights to sum to 0.5 for $\mathcal{P}$ and $\mathcal{N}$
  - $h_t \leftarrow$ WEAKLEARN($\mathcal{P}, \mathcal{N}, \mathbf{w}$)
  - $\epsilon_t \leftarrow$ the weighted error for h_t given $\mathbf{w}$
  - $\beta_t \leftarrow \epsilon_t / (1 - \epsilon_t)$
  - For correctly classified examples i,
    - $w_i \leftarrow \beta_t w_i$
- Return H :

$$
H(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0 & \text{otherwise} \end{cases}
$$

Fig. 3. AdaBoost learning algorithm [?].

Towards constructing an extremely efficient classifier that can be quickly evaluated on a given window in an image pair, following Viola, Jones, and Snow [?], we cascade several ensembles as described in Figure 2.

The result is a cascade C of ensemble classifiers H_i, each consisting of several weak classifiers based on rectangle filters. The training method means that we train a classifier on the initial training set, get a set of false positives from the trained classifier, then use those false positives to train the next stage of the cascade. Given a set of positive and negative examples, we train h_i using the AdaBoost algorithm, described in Figure 3.

The weights $\mathbf{w}$ indicate the importance of each training example. Early on in training (small t), simple individual weak classifiers will have fairly low error, making β_t small,

so that examples correctly classified early on will have less weight later, and therefore have less influence on subsequent calls to the weak learner. This means the later weak learners focus on the "hard" examples that are not correctly classified early on. The final result, H, is simply a weighted majority vote by each of the weak classifiers.

The weak learning algorithm, as mentioned before, is simply to try all possible rectangle filters and thresholds, and determine which combination of filter and threshold minimizes the weighted error.

B. Run-time

The run-time system is responsible for evaluating a classifier cascade on every possible window in a given image pair. We begin with the image pair at its original resolution, calculate the needed difference images and integral images, then slide the search window over the image. We first classify a given window according to the first ensemble in the classifier cascade, H_0. This entails calculating the prediction of each of the weak classifiers h_i in H_0 and weighting their votes.

Rather than use the vote threshold recommended by AdaBoost directly, however, following [?], we adjust the threshold to meet a specific target true positive rate at each stage, thus ensuring that very few windows containing hands will be rejected prematurely, at the cost of potentially accepting more false positives. If the candidate window location passes stage 0 of the cascade, we then evaluate ensemble $H_1, H_2, \ldots$ on the window until it is either rejected by one stage or accepted by every stage of the cascade. The run-time system records each location in the image predicted to be positive by the classifier cascade, eliminating any weak positives that are overlapped by stronger positives. Finally, when every possible window location at the current scale has been tested, we down-scale the entire image and sweep the window over the image again. Though the window is still the same size, this corresponds to a larger window in the original image pair.

Once every possible window location in the input image pair has been classified, the run-time system returns the list of potential hand locations to the running application.

IV. Experimental Results

We have implemented the system described in the previous section and performed a preliminary evaluation using a single training video sequence captured in a cafeteria at SIIT. Using an inexpensive IEEE 1394 digital camera, we captured 8-bit 640×480 grayscale video at 4 fps for 10 minutes. We paired consecutive frames of the sequence (1198 pairs of images) and manually selected the image regions containing suitable human hands as described in the previous section. We cropped the hand regions from each image pair and scaled the resulting small pairs to a size of 20×20 pixels. After exhaustively scanning the data set, we had 1060 pairs of 20×20 images containing human hands (many pairs contained several suitable locations, but many others contained no suitable locations). For the initial set of negative examples $\mathcal{N}_0$, we selected 1060 random patches random from the original video sequence, with width varying uniformly between 20 and 80 pixels.

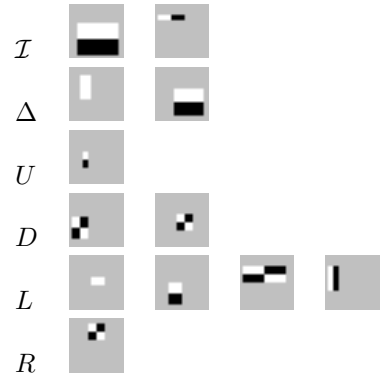


Fig. 4. Filters in the first-stage hand classifier.

TABLE I
Cascade training and validation results.

Cascade Stage	# of Weak Classifiers	Training Set Accuracy	Validation TP rate	Validation FP rate
1	12	0.916	0.953	0.094
2	91	0.882	0.953	0.229
3	193	0.963	0.953	0.297
4	171	0.537	0.953	0.294

From the 1060 negative and 1060 positive training examples, we reserved 10% (212 image pairs) for testing and an additional 10% (212 image pairs) as a validation set to deterimine score thresholds. The remaining 80% of the data was fed to the cascade learning algorithm (Figure 2). Our current cascade has depth four; for each stage, we trained an ensemble consisting of 200 weak classifiers using AdaBoost (Figure 3, $T = 200$), but then used the number of classifiers that produced the lowest ensemble false positive rate for a true positive rate of 0.95. Figure 4 shows the filters selected by AdaBoost for the first-stage of the cascade, and Table **??** shows the ensemble size and false positive rates for each stage of our cascade.

After each stage was trained and appropriate score cutoffs were determined, we extracted new negative examples for the training and validation sets using false positives from the current classifier. Specifically, using the first stage ensemble containing 12 weak classifiers and the score cutoff determined from the first stage validation set, we ran the resulting classifier on selected training images until the number of positively classified windows predicted by the classifier was at least 350. We then manually classified those positives into true positives and false positives, and used the false positives to construct the training and validation sets for the next stage of the classifier. So each stage of the classifier has the same positive examples in its training set and validation set as the previous stage; however, each stage has a new (more difficult) set of negative examples in its training and validation sets. The final classifier cascade achieves an accuracy of 0.882 on the final test set (which was never seen at any point during training); it properly classified 86/106 (81%) of the positive

TABLE II
TEST SET PERFORMANCE AT EACH STAGE OF THE CASCADE.

Cascade Size	False Positives	False Negatives
1	10/106	13/106
2	8/106	15/106
3	8/106	15/106
4	5/106	20/106

test items as positives, but improperly classified 5/106 (5%) of the negative test items as positives. Table **??** shows the test set performance and how it changes with each layer of the cascade. We find that the overall accuracy on the test set does not change much with each additional level of the cascade; the levels seem mostly to be trading off false positives for false negatives.

After training the four-stage cascade, we evaluated it more thoroughly on ten full image pairs from the test set. Again, these image pairs had never been seen during training (though of course the background was the same). The current implementation of the run-time system evaluates the classifier cascade C on 349,740 windows, from a size of 20×20 to (nearly) the entire image. Unfortunately, since our current classifier is only approximately 90% accurate on its test set, we can guess that it will have a high false positive rate (indeed we can guess that if the test set is representative of all windows in the image, we should get on the order of 35,000 false positives). The situation is not that severe, however, because, as described earlier, when two overlapping regions at the same scale are positive according to the classifier, we only retain the window with the highest confidence. Figure 5 shows our preliminary results from the run-time system on two image pairs in the test sequence. Only the top 10 positives are shown for each pair (the system actually identified 88 positives for the first pair and 131 positives for the second pair). Clearly, the system's false positive rate is too high for direct use, but even so, it does detect most of the hands in the scene, and it is readily apparent why it makes the mistakes it does. In the next section, we detail how we plan to improve the system in future research.

V. CONCLUSION

We have only begun to tackle a very difficult computer vision problem. Though these preliminary results are encouraging, currently, the hand detection system makes far too many mistakes to be of practical use. There are four main factors contributing to the mistakes:

1) At four fps, we observe too much movement of the hands in consecutive frames. Selecting a large enough region to encompass both appearances of the hand, followed by down-scaling to 20×20, often shrinks the hands to unnecessarily small sizes and introduces too much variability, since the hands end up at opposite extremes of the crop window.
2) A first-rate detector will require a much larger training set. Face detectors like the V&J detector require thousands of examples to robustly detect human faces, which have a much more regular structured than human hands in arbitrary poses. Compared to the standards of face detection research, then, our training set is rather small.
3) The lack of dramatic improvements on the test set with each additional stage of the classifier indicates overtraining. That is, each additional stage seems to improve performance on its validation set, but the newly learned features do not apparently generalize very well to the test set.
4) In many cases, it might be practically impossible to distinguish a 20×20 grayscale image containing a human hand from other similarly blob-shaped, moving objects. Particularly in light of factor 2 above, we believe a human observer would also have a great deal of difficulty with the classification task.

The first problem is simple to solve with a faster frame rate. We need not run the full detection algorithm at high frame rates (4–10 Hz should be sufficient for most applications), but a 30 fps capture rate would mean less hand travel between frames and make the training system's job easier.

The second and third problems can be solved with some effort. We plan to acquire several video sequences under a variety of conditions in order to improve the classifier's robustness to noise. It may also be possible to generate new synthetic training data from the existing data by adding small random rotation, translation, and scale transforms to the data and including the transformed items in the training set. With enough high-quality training data, we should be able to prevent the overtraining problem.

The only potentially serious problem is the last problem. To overcome any inherent ambiguity in the stimulus, we may have to add additional information, such as color, or make the set of possible filters richer, or use higher resolution imagery, or even try to find the humans *before* finding their hands. These solutions, however, will all increase the system's training time, run time, and/or complexity.

Our ultimate long-term goal is to deploy the system as part of a mobile robotics application. Once we have the system performing robustly with a fixed camera, we plan to explore ways to achieve similar results with a moving platform. The current approach obviously will not work, since the difference images rely on a non-moving background. It might be possible to correct for camera motion prior to detection, but a more practical approach would be to have the robot stay still long enough for an initial detection of the hands in the scene, then use more dynamic techniques like tracking and active vision while moving.

ACKNOWLEDGMENTS

We thank the students of SIIT for being our test subjects. Cristopher Voldum and Yang-Cheng Fan helped with the software development for this project. MND is supported by Thailand Research Fund grant MRG4780209.

Fig. 5. Preliminary results on test data from the hand detector. Each row shows a consecutive pair of test images, with the top 10 most likely hands highlighted with boxes.

REFERENCES

[1] M. Becker, E. Kefalea, E. Maël, C. von der Malsburg, M. Pagel, J. Triesch, J. C. Vorbrüggen, R. P. Würtz, and S. Zadel, "GripSee: A gesture-controlled robot for object perception and manipulation," *Autonomous Robots*, vol. 6, pp. 203–221, 1999.

[2] C. Shan, Y. Wei, T. Tan, and F. Ojardias, "Real time hand tracking by combining particle filtering and mean shift," in *IEEE International Conference on Face and Gesture Recognition*, 2004, pp. 669–674.

[3] J. M. Rehg and T. Kanade, "Visual tracking of high DOF articulated structures: An application to human hand tracking," in *Third European Conference on Computer Vision*, 1994, pp. 35–46.

[4] S. Ahmad, "A usable real-time 3D hand tracker," in *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, 1995, pp. 1257–1261.

[5] J. Segen and S. Kumar, "Human-computer interaction using gesture recognition and 3D hand tracking," in *Proceedings of the IEEE International Conference on Image Processing*, 1998, pp. 188–192.

[6] J. Triesch and C. von der Malsburg, "A system for person-independent hand posture recognition against complex backgrounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 12, 2001.

[7] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[8] R. Polana and R. Nelson, "Low level recognition of human motion (Or how to get your man without finding his body parts)," in *Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, 1994, pp. 77–82.

[9] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, 2003, pp. 734–741.

[10] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[11] Y. Freund and R. E. Shapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 5, no. 1, pp. 119–139, 1997.

[12] M. Kölsch and M. Turk, "Robust hand detection," in *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.

ROBUST HAND TRACKING IN LOW-RESOLUTION VIDEO SEQUENCES

Nyan Bo Bo[1] Matthew N. Dailey[2] Bunyarit Uyyanonvara[1]

[1]Sirindhorn International Institute of Technology
Thammasat University
Klong Luang, Pathumthani, Thailand
email: {nyanbobo,bunyarit}@siit.tu.ac.th

[2]Computer Science and Information Management
Asian Institute of Technology
Klong Luang, Pathumthani, Thailand
email: mdailey@ait.ac.th

ABSTRACT

Automatic detection and tracking of human hands in video imagery has many applications. While some success has been achieved in human-computer interaction applications, hand tracking would also be extremely useful in security systems, where it could help the system to understand and predict human actions, intentions, and goals. We have designed and implemented a prototype hand tracking system, which is able to track the hands of moving humans in low resolution video sequences. Our system uses grayscale appearance and skin color information to classify image subwindows as either containing or not containing a human hand. The prototype's performance is quite promising, detecting nearly all visible hands in a test sequence with a relatively low error rate. In future work we plan to improve the prototype and explore methods for interpreting human gestures in surveillance video.

KEY WORDS
Computational Intelligence, Hand Tracking, Computer Vision, Image Processing.

1 Introduction

Just as monitoring a person's face is key to effective communication with that person, monitoring a person's hands is key to understanding what that person is *doing*. Towards endowing intelligent systems with the ability to monitor and understand human behavior, in this paper, we propose and evaluate an object detection system capable of finding hands in video imagery without requiring any close-up, high-resolution analysis.

Human hand detection in video sequences would enable several useful applications. We are primarily interested in security applications, where human figures in the scene could be at a fairly large distance distance from the camera and therefore would appear at a fairly low resolution. In security applications, it would be useful to track people in the scene and perform automated analysis of their actions, e.g., by detemining if they are walking, running, punching someone, and so on. The key to many of these behaviors is the ability to track the hand.

Over the last 15 years, the problem of hand tracking has become an attractive area for research [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Many early hand tracking systems relied on uncluttered static backgrounds, high resolution images and manual initialization. Though state of the art systems are becoming more robust, most have been targeted towards sign language and gesture recognition, where the assumption of high-resolution imagery holds. In these cases, motion information between two consecutive frames makes hand tracking easier since most of the body parts will be stationary except the hands. In the general case, motion information is less useful for tracking hands, since all of the body parts may be moving from frame to frame.

Some systems like Pfinder [5] imitate the way humans look for hand in images — instead of directly detecting hands in the image, Pfinder tries to find human bodies first and then finds body parts such as hands. However, finding the human in an image is itself a difficult problem, so in our work we attempt to bypass this problem entirely by finding hands without any attempt to model the human context.

Some hand tracking systems use skin color information to segment hands from the background and then track segmented hands over a frame sequence. The face and hand tracking system for sign language recognition by Soontranon and Chalidabhongse [7] first segments the image into skin and non-skin regions using a Gaussian model for skin pixels in CbCr space. Then they obtain the connected components of the skin pixel image and track the connected components from frame to frame assuming translation only. They use face detection to prevent false positives due to skin pixels on the face. A similar approach is used by [10, 12] to detect and track hands for human-robot interaction and human-computer interaction.

In recent years, face detection has been one of the great success stories in computer vision; new techniques are achieving excellent performance in real time [13, 14, 15, 16, 17]. Hand tracking is more difficult, however, since the face contains structural information such as eyes, mouth and so on, even in low resolution images. Much less structural information, except the gaps between fingers, is present in hand image, however, so hands look more or less like elliptical blobs of skin in low resoultion images.

Barreto and colleagues [9] applied the Viola and Jones face detector [15] (later improved by Lienhart and Maydt [18]) to upright hand detection. Ong and colleagues

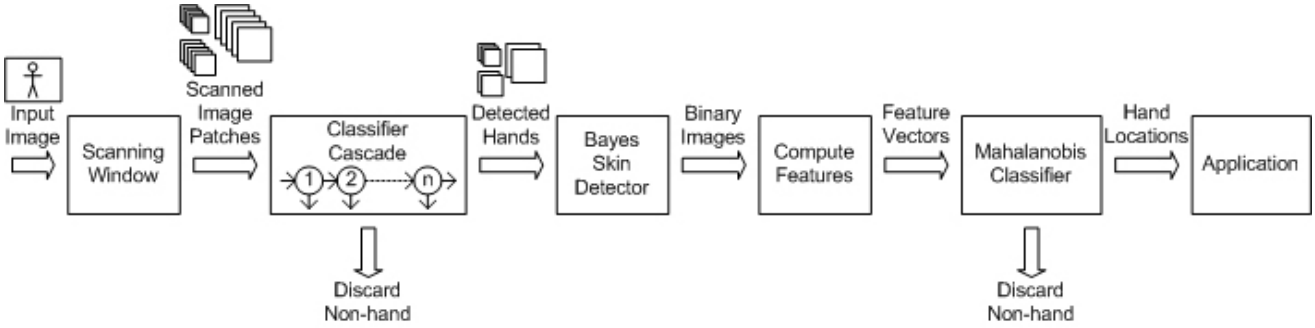


Figure 1. Hand detection system architecture.

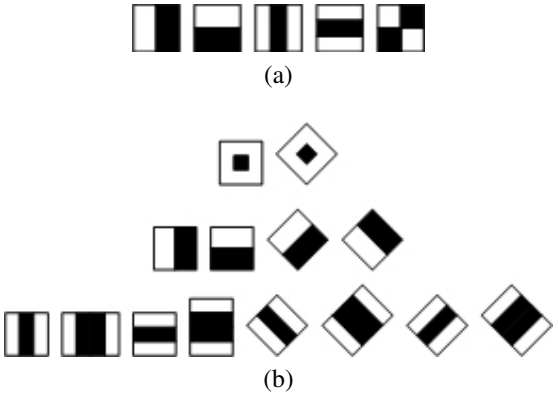


Figure 2. Haar-like features used to construct weak classifiers in the boosted classifier cascade. (a) Viola and Jones features. (b) Lienhart and Maydt features.

[8] use a similar approach, but instead of a linear cascade to detect a single hand posture, they construct a tree-structured classifier to not only detect hands but also to classify hand posture.

Thus far, hand detection systems using general object recognition techniques like those from the face detection literature are tailored to detecting distinct sets of specific gestures. Our goal, however, is to detect and track multiple hands in arbitrary postures in relatively low-resolution video sequences. Our approach uses grayscale appearance information to reject most non-hand image regions very quickly, then uses skin color information to reject the remaining non-hand image patches. We have conducted preliminary experiments with the proposed system, and the results are encouraging. In the rest of the paper, we describe the system and its evaluation in detail.

2 Hand Detection

Figure 1 depicts our hand detection system's architecture schematically. A scan window sweeps over the input image at multiple scales. Each resulting image patch is classified as either hand or non-hand by a boosted classifier cascade

Algorithm TRAIN-CASCADE
Given: $\mathcal{P}_0$, a set of k positive examples
$\mathcal{N}_0$, an initial set of k negative examples
α_p, the desired true positive rate per stage
α_f, the desired false positive rate per stage
Returns: C, a cascade of ensemble classifiers
$C \leftarrow H_0 \leftarrow$ ADABOOST($\mathcal{P}_0, \mathcal{N}_0, \alpha_p, \alpha_f$)
$i \leftarrow 0$
Repeat:
Test C on new, known-to-be-negative examples
$\mathcal{N}_i \leftarrow$ The top k false positives from test
$\mathcal{P}_i \leftarrow (\mathcal{P}_{i-1} -$ positives dropped by $H_{i-1})$
$H_i \leftarrow$ ADABOOST($\mathcal{P}_i, \mathcal{N}_i, \alpha_p, \alpha_f$)
$C \leftarrow C$ with H_i appended to the cascade
$i \leftarrow i + 1$
Until performance is "good enough"
Return C

Figure 3. Cascade training algorithm. The algorithm utilizes the ADABOOST routine, which, given a training set $\langle \mathcal{P}, \mathcal{N} \rangle$, finds a combination of weak threshold classifiers obtaining a true positive rate of at least α_p and a false positive rate at most α_f on that training set.

[15, 18]. To further reduce false positive detections using a priori knowledge of hand color and geometry, each positive detection from the classifier cascade is further processed by a skin detection module, a feature extractor, and a simple classifier based on Mahalanobis distance to the "average" hand. We describe each of the modules in turn.

2.1 Boosted classifier cascade

The core of our object detection system is the cascade of boosted classifiers originally proposed by Viola and Jones [19, 15] and later modified by Lienhart and Maydt [18, 20]. The cascade reduces processing time while preserving classifier accuracy through the use of a sequence of classifiers tuned for reasonably low false positive rates but extremely high detection rates. The cascade quickly rejects most de-

tection windows unlikely to contain the object of interest and spends more compute time on the detection windows most likely to contain the object of interest.

Each stage in the cascade uses the "boosting" ensemble learning method [21] to induce a strong nonlinear classification rule that is a linear combination of the "votes" of multiple weak threshold classifiers, each considering the output of a single Haar wavelet-like filter at a fixed location in the detection window. Viola and Jones' original method [19] uses Freund and Shapire's "discrete" Adaboost algorithm [21] with a set of five types of Haar-like features for the weak threshold classifiers (Figure 2(a)). Lienhart and colleagues' method [18, 20] uses the "gentle" Adaboost algorithm and additional Haar-like features (Figure 2(b)). Here we refer to both types of classifier as a *V&J cascade*.

The first step in constructing a V&J cascade for object detection is to obtain a large training set of images containing or not containing the object of interest. We then extract an initial set $\mathcal{P}_0$ of positive detection windows and an initial set $\mathcal{N}_0$ of negative detection windows and execute the procedure TRAIN-CASCADE, detailed in Figure 3.

2.2 Bayesian skin detector

Our skin detector is a Bayesian maximum likelihood classifier based on color histograms [22, 23]. The classifier estimates the class $S \in \{skin, nonskin\}$ of a single pixel based only on its observed color x measured in some color space. This simple method is extremely efficient and surprisingly effective. We let

$$\hat{s} = \arg\max_s P(X = x \mid S = s),$$

where the likelihood $P(X \mid S)$ is modeled by a color histogram estimated from training data we obtained in a pilot study. Our color histograms have two dimensions, namely the hue and saturation axes of the HSV color space, which we quantize into $16^2 = 256$ bins.

2.3 Feature extraction

The output of the Bayesian skin detector is a binary image in which one value represents skin pixels and the other value represents non-skin pixels. We have found that a few simple features extracted from this binary image allow surprisingly accurate classification. The particular features we extract are:

1. The *area* (in pixels) of the largest connected component of skin pixels.
2. The length of the *perimeter* of the largest connected component of skin pixels.
3. The *eccentricity* of the largest connected component of skin pixels.
4. The number of pixels in the largest skin component intersecting the detection window *boundary*.

Clearly, when the area feature is especially large or especially small, the given image patch is unlikely to contain a hand. The perimeter length and eccentricity features provide additional information about the shape of the detected skin "blob." Finally, since a properly detected hand will only intersect the boundary of the detection window at the wrist, the boundary feature provides information about how wrist-like the boundary is.

When combined with a reasonable classifier, these four features are sufficient to correct most of the V&J cascade's mistakes. We next describe our classifier.

2.4 Mahalanobis classifier

Given a feature vector $\boldsymbol{x}$ consisting of the area, perimeter, eccentricity, and boundary features, we must determine whether $\boldsymbol{x}$ represents a true hand or a false positive. We tackle this problem by applying a threshold θ to the dissimilarity of the given feature vector $\boldsymbol{x}$ from the mean feature vector $\boldsymbol{\mu}$. Our dissimilarity measure is the Mahalanobis distance

$$d(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}).$$

Here the mean hand feature vector $\boldsymbol{\mu}$, covariance matrix Σ, and distance threshold θ are estimated from a training set.

Once we obtain a final classification for each possible detection window, the positively detected hands could then be forwarded to another component in an integrated application, for example a gesture recognition module. In the current paper we simply evaluate the efficacy of the proposed algorithm on a series of video segments.

3 Experimental Methods

Here we describe an experiment in which we captured video sequences of humans walking in an indoor environment then evaluated the hand detection system on those video sequences.

3.1 Data acquisition

For purposes of training and testing the hand detection system, we captured four video sequences of four different people walking in and out of a moderately cluttered laboratory environment. The video sequences were captured at 15 frames per second with an IEEE 1394 Web camera, and each sequence lasted approximately three minutes.

After video acquisition, we manually located all visible hands in every image of all four sequences, for a total of 2246 hands. We designated the first 2000 as training examples and reserved the remaining 246 for testing.

Our criteria for positioning the detection window on the hand was that that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of selection window (see Figure 4 for examples).

As already described, the cascade learning algorithm, at step i, requires a set $\mathcal{N}_i$ of negative examples that do

Figure 4. Example training images scaled to 24×24.

not contain hands for training. For this purpose, we randomly selected eight frames from the training data that did not contain and hands. The OpenCV implementation of the V&J cascade (see below) scans these eight images to produce new negative examples for training at each stage.

3.2 Boosted classifier training

To train the classifier cascade, we used Linehart and Maydt's approach [18] as implemented in OpenCV [24]. With 2000 positive hand examples and 8 background images for negative examples, we trained 30 stages using GentleBoost, $\alpha_p = 0.995$, and $\alpha_f = 0.5$. This took two weeks on a 3 GHz Pentium 4 PC with 1 GB of RAM.

We found that the classifier's training set performance peaked at 26 stages, so we used only the first 26 stages, comprising 763 weak classifiers, in subsequent analysis.

3.3 Skin detector training

To train the skin detector, we selected 10 images containing one or more humans from a set of independent video sequences captured under various lighting conditions at several different locations. We manually marked the skin pixels in each image, extracted the hue (H) and saturation (S) of the resulting 70,475 skin and 1,203,094 non-skin pixels, quantized the H-S values into 16 bins, and constructed two 2D histograms: one for skin pixels, and one for non-skin pixels.

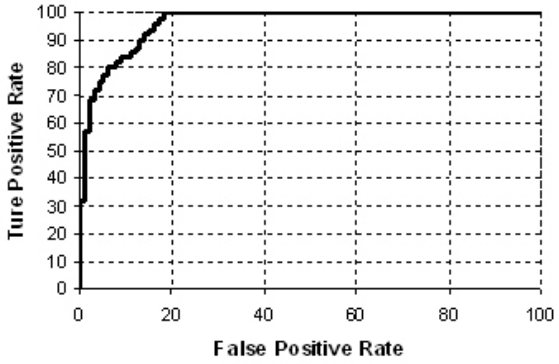


Figure 5. ROC curve between true positive rate and false positive rate

3.4 Gathering data to train the post-processor

We ran OpenCV's performance testing utility, which had been modified to produce true positive and false positive image patches, on all labeled images from the training set with a detection window scale factor of 1.1. The resulting image patches were scaled to the standard size of 24×24. Then, we randomly selected 1000 true positive and 1000 false positive image patches for the training process of post-processor system.

3.5 Parameter estimation for the post-processor

To estimate the parameters of the Mahalanobis distance-based post processor, we applied skin detection to the 2000 training patches, eliminated all connected skin components smaller than 36 pixels, and filled in holes with a morphological opening operator. We then extracted features (area, eccentricity, perimeter, and boundary pixel count) for the largest connected skin blob in each of the 2000 patches. We randomly split the true positives into two groups, using the first 500 for mean and covariance calculation and using the remaining 500 to determine the best Mahalanobis distance threshold. Using the ROC curve in Figure 5 to explore the tradeoff between true positives and false positives, we selected the point where the false positive rate was as low as possible (18%) while maintaining a 100% true positive rate.

4 Results

To analyse the performance of our system, we selected 4 frames from a video sequence which had never been used in the training process, and fed it to our system. We manually classified the detections at each stage of the system as a false positive or true positive. We found that the classifier cascade, by itself, performed relatively poorly, but that the post processing system was extremely effective in eliminating false positives produced by the classifier cascade.

Image	Number of True Positives	Number of False Positives
1	1	22
2	2	35
3	1	19
4	1	18

Table 1. Test results without post-processing

Image	Number of True Positives	Number of False Positives
1	1	2
2	2	5
3	1	6
4	1	2

Table 2. Test results with post-processing

Table 1 shows that without the post-processing system, the false positive rate is too high for practical application. Table 2, on the other hand, shows that when we add post processing to our system, the false positive rate decreases rapidly.

Image 1 and 2 in Figure 6 illustrate example detection results from our complete system. In both images, we observe that regions on the person's head, especially in the chin and neck areas, are detected as hands by the system. The most probable reason for this kind of false positive is that we did not include such image patches as negative examples during training of the boosted classifier cascade. We only used background images to generate negative examples. Unfortunately, not even the post-processing system can reject this kind of false positive because the shape of the skin blobs in those regions are very similar to the shape of skin blobs in patches actually containing hands. We believe that including human body parts other than hands as negative training examples will eliminate these types of false positives.

5 Conclusion

From the literature, we know that hand trackers incorporating Adaboost and Haar-like features perform quite well in applications like sign language recogntion, in which the video is relatively high resolution and the hand gestures are rather constrained. We have found, on the other hand, that the approach suffers from high false positive rates when applied to less constrained scenes. However, as we have shown in this paper, these limitations can be reduced by incorporating domain-specific knowledge in the form of a post processing system.

One important limitation of the work described here is that both the training and testing data were captured in the same simple lab environment. The reported performance is therefore optimistic. In future experiments, we plan to test the approach more rigorously by training on a wider variety of scenes and testing on a completely novel scene.

In any case, the system is a good starting point for a practically applicable low resolution, real-time hand tracker. The current results are encouraging, and with some improvement we will be able to integrate it with a complete gesture recognition or activity recognition system.

Acknowledgments

We thank the members of the Image and Vision Computing Laboratory at the Sirindhorn International Institute of Technology for participating in our data collection efforts. This research was partially supported by Thailand Research Fund grant MRG4780209 to MND.

References

[1] J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Third European Conference on Computer Vision*, pages 35–46, 1994.

[2] J. Segen and S. Kumar. Human-computer interaction using gesture recognition and 3D hand tracking. In *Proceedings of the IEEE International Conference on Image Processing*, pages 188–192, 1998.

[3] S. Ahmad. A usable real-time 3D hand tracker. In *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, pages 1257–1261, 1995.

[4] J. Triesch and C. von der Malsburg. A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12), 2001.

[5] C. R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[6] M. Kölsch and M. Turk. Robust hand detection. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, 2004.

[7] N. Soontranon, S. Aramvith, and T. H. Chalidabhongse. Face and hands localization and tracking for sign language recognition. In *International Symposium on Communications and Information Technologies*, pages 1246–1251, 2004.

[8] Eng-Jon Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 889–894, 2004.

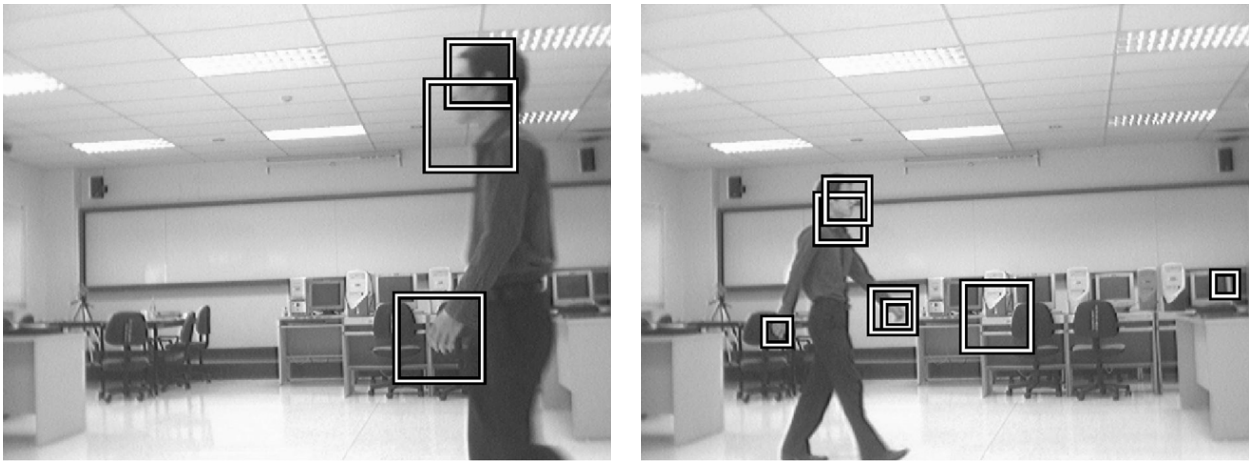


Image 1 Image 2

Figure 6. Preliminary results on test data from the hand detector.

[9] J. Barreto, P. Menezes, and J. Dias. Human-robot interaction based on haar-like features and eigenfaces. In *Proceedings of the 2004 IEEE Conference on Robotics and Automation*, pages 1888–1893, 2004.

[10] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler. A real-time hand gesture system based on evolutionary search. In *Genetic and Evolutionary Computation Conference*, 2005.

[11] S. Wagner, B. Alefs, and C. Picus. Framework for a portable gesture interface. In *Proceeding of the 7th International Conference on Automatic Face and Gesture Recognition*, pages 58–63, 2006.

[12] Kye Kyung Kim, Keun Chang Kwak, and Su Young Chi. Gesture analysis for human-robot interaction. In *The 8th International Conference on Advanced Communication Technology*, pages 1824–1827, 2006.

[13] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(20):23–28, 1998.

[14] D. Roth, M. Yang, and N. Ahuja. A SNoW-based face detector. In *Advances in Neural Information Processing Systems (NIPS)*, pages 855–861, 2000.

[15] P. A. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[16] I. Fasel, B. Fortenberry, and J. Movellan. A generative framework for real time object detection and classification. *Computer Vision and Image Understanding*, 98:182–210, 2005.

[17] E. Hjelmås and B. K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, pages 236–274, 2001.

[18] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, volume 1, pages 900–903, 2002.

[19] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[20] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs, 2002.

[21] Y. Freund and R. E. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 5(1):119–139, 1997.

[22] B. D. Zarit, B. J. Super, and F. K. H. Quek. Comparison of five color models in skin pixel classification. In *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems*, pages 58–63, 1999.

[23] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. *International Journal of Computer Vision*, 46(1):81–96, 2002.

[24] Intel Corporation. OpenCV Computer Vision Library (software). Open source software available at `http://sourceforge.net/projects/opencv/`.

NATURAL-POSED HAND DETECTION IN LOW-RESOLUTION IMAGES

Nyan Bo Bo[1] (*Corresponding Author*)
M.Sc. Candidate (Information Technology)
Email : nyanbobo@gmail.com
Mobile : +66-81-402-7959

Matthew N. Dailey[2]
Ph.D. (Computer Science and Cognitive Science)
Email : mdailey@ait.ac.th

Bunyarit Uyyanonvara[1]
Ph.D. (Image Processing)
Email : bunyarit@siit.tu.ac.th

[1]Information Technology Program
School of Information and Computer Technology
Sirindhorn International Institute of Technology
Thammasat University
131 Moo, Thiwanont Road, Bangkadi
Pathumthani 12000 Thailand

[2]Computer Science and Information Management
School of Engineering and Technology
Asian Institute of Technology
Klong Luang
Pathumthani 12120 Thailand

Key Words
Hand detection and tracking, Boosted classifier cascade, Skin detection, Mahalanobis classifier, Artificial intelligence, Computer vision and image processing

Abstract

Robust real-time hand detection and tracking in video sequences would enable many applications in areas as diverse as human-computer interaction, robotics, security and surveillance, and sign language-based systems. In this paper, we introduce a new approach for detecting human hands that works on single, cluttered, low-resolution images. Our prototype system, which is primarily intended for security applications in which the images are noisy and low-resolution, is able to detect hands as small as 24×24 pixels in cluttered scenes. The system uses grayscale appearance information to classify image sub-windows as either containing or not containing a human hand very rapidly at the cost of a high false positive rate. To improve on the false positive rate of the main classifier without affecting its detection rate, we introduce a post-processor system that utilizes the geometric properties of skin color blobs. When we test our detector on a test image set containing 106 hands, 92 of those hands are detected (86.8% detection rate), with an average false positive rate of 1.19 false positive detections per image. The rapid detection speed, the high detection rate of 86.8%, and the low false positive rate together ensure that our system is useable as the main detector in a diverse variety of applications requiring robust hand detection and tracking in low-resolution, cluttered scenes.

Introduction

If it were possible to detect and track human hands in video sequences, a variety of useful applications would be possible. These applications include human-computer interaction, human-robot interaction, gesture and sign-language recognition, intelligent security systems and more. Over the last 15 years, the problem of hand tracking has

become an attractive area for research in the field of computer vision. Many early hand tracking systems relied on uncluttered static backgrounds, high resolution imagery, and manual initialization. Most of the modern hand tracking systems are oriented towards sign language recognition, human-computer interaction, and human-robot interaction. In these applications, it is possible to make the very useful assumption that only hands are moving while the rest of the scene is stationary. The problem can be further simplified by assuming that there will be only two hands, since there should be only one person performing sign language or gestures in the scene. Nowadays, the systems are becoming more robust, but they generally still require high resolution imagery.

We are primarily interested in hand detection because monitoring person's hand is a key to predict what that person is doing. In security applications, it would be very useful to detect and track hands of people in the scene and perform automated analysis of their actions, e.g., by determining if they are walking, running, punching someone, or even identifying any object they are holding. Detecting and tracking hands in security applications is more challenging than in human-computer interaction because most surveillance cameras provide noisy images, with human figures quite far away and therefore appearing at a fairly low resolution. The resolution of hands in those images may be as small as 24×24 pixels; detecting such small hands in static images is a very challenging task. Another difficulty is that motion information is less useful since there may be many people in the scene, and their entire bodies may be moving from frame to frame as they move in front of the security camera.

Some early hand tracking systems like Pfinder proposed by Wren et al. (1997) attempt to follow the way humans look for the hand in images. Instead of directly detecting hands in an image, Pfinder looks for human bodies first and then easily

segments out hands from the rest of the body by using skin color. However, since detecting humans in a cluttered video sequence is itself a very difficult problem, and the human body could easily be partially occluded in the scene, we try to bypass the human detection problem in our work by finding hands directly, without any attempt to find the entire human body first.

Most of the modern hand tracking systems fall into one of two main categories. The first approach uses skin color information to segment hands from the background and then tracks segmented hands between frames using a tracking algorithm. The face and hand tracking system for sign language recognition proposed by Soontranon et al. (2004) first segments the image into skin and non-skin regions using an elliptical model for skin pixels in CbCr space. Then face detection is used to locate the face skin blob ideally leaving only the skin blobs of hands. The system constructs a template for each hand then in subsequent frames, finds the region best matching that template using a minimum mean-squared error cost function. A similar approach is used by Wachs et al. (2005) to detect and track hands for human-robot interaction. The system proposed by Varona et al. (2004) also takes this approach but their system is extended to track hands and faces in 3D for a virtual reality application. The hand tracker of Shamaie and Sutherland (2005) does not use skin color information so it works on monochrome video sequences. Hands are extracted from the background using a blob analysis algorithm then tracked using a dynamic model from control theory. Unfortunately, these approaches relying on tracking are not suitable when the goal is to extract hands from single images.

However, in a second approach, a detection window is scanned over the image and each of the scanned image patches are classified as hand or non-hand. In contrast to

the first approach, this approach can be used to detect hands in static images. Barreto et al. (2004) applied improved version of Viola and Jones (2001) face detector by Lienhart and Maydt (2002), to detect hands for a human-robot interaction application. Their hand detection system works quite well at various scales and with different backgrounds under various illumination conditions. The hand tracking system proposed by Ong and Bowden (2004) uses a similar approach, but they construct a tree-structured classifier, instead of a linear cascade, not only to detect hands but also to classify hand posture. Both of these systems require high-resolution imagery. The hand detector by Caglar and Lobo (2006) also detects hands in high resolution static images, in this case making use of the geometric properties of the hand without the use of skin color or motion information. Their proposed system is robust to the size and the orientation of hands with the limitation that one or more fingers must be visible.

The goal of our proposed system is to detect and track multiple hands in arbitrary postures in relatively low-resolution video sequences. Our approach uses grayscale appearance information to reject most of the non-hand image regions very rapidly and then uses the shape of skin color regions to reject most of the remaining non-hand image patches. We conducted a thorough evaluation on our proposed system and found that its detection rate was 86.8% and that its false positive rate was 1.19 false detections per image on average. The system's speed and accuracy will enable many useful applications that are based on hand detection and tracking.

Materials and Methods

Hand Detector

A block diagram of our hand detection system is shown in Figure 1. A scan window is scanned over the input image at different scales and each of the resulting image patches is fed into a classifier cascade which rapidly determines whether the image patch is a hand. Our classifier cascade eliminates more than 95% of the non-hand regions in a given image. However, due to the large number of candidate regions in one image, to be practical, the false positive rate must be further reduced. To serve this need, we add a postprocessor to the system in order to further reduce false positive detections. The postprocessor takes advantage of a priori knowledge of hand's color and geometry. Skin detection, feature extraction, and Mahalanobis classification are the essential building blocks of our postprocessor.

Scanning Window

When an image is presented to our hand detection system (Figure 1), a detection window is scanned over the image at multiple scales, and each resulting image patch is passed to the boosted classifier cascade. The scanning process is to begin with a 24×24 detection at the image's original scale. After every possible image patch at that scale has run through the classifier cascade, the image is scaled down by 90% and the process is repeated until a minimum image size (maximum detection window size) is reached.

Boosted Classifier Cascade

Viola and Jones (2001, 2004) originally proposed the cascade of boosted classifiers as a real-time general object detector and applied it to face detection. They showed that the system could robustly detect faces in static images independent of the background. The system runs in real-time since the feature detector is limited to a class of Haar-like filters that can be computed in constant time with the help of integral images, regardless of the spatial extent of the filters. The speed of the system is increased even further by arranging the classifiers in a cascaded fashion, so that the early stages reject most of the image patches unlikely to contain the object of interest. The cascade therefore only spends significant compute time on the image patches most likely to contain the object of interest.

Each stage in the cascade is constructed from a set of simple Haar-like filters using Freund and Shapire's (1997) AdaBoost algorithm. AdaBoost builds a strong nonlinear classifier from multiple weak threshold classifiers, in this case each using a Haar-like filter, a threshold, and a weight, all of which are selected by AdaBoost to minimize the weighted error for the whole stage over the training set, while maintaining the desired detection rate. Viola and Jones (2001, 2004) used the four types of Haar-like filters shown in Figure 2 (a). The filters can take on arbitrary positions and sizes within an 24×24 image patch. The output of each filter is simply the difference between the average pixel value within the clear rectangular regions and the shaded rectangular regions.

Recently, Lienhart and Maydt (2002) modified Viola and Jones (2001, 2004) detector. Their system adds additional rotated Haar-like filter types, as shown in Figure 2 (b). On a particular test set, they found that their modified system gave 10% fewer

false positives than the original system for certain detection rates. The empirical analysis of detection cascade of boosted classifier by Lienhart et al. (2002) compared Discrete AdaBoost (the algorithm used by Viola and Jones [2001]), Real AdaBoost, and Gentle AdaBoost and found that classifiers trained with Gentle AdaBoost performed the best.

We apply Lienhart and colleagues' methods, as implemented in the OpenCV, Open Computer Vision Library (2006), to the hand detection problem, using all the filter types in Figure 2 (b), 24×24 image patches, and the Gentle AdaBoost learning algorithm.

Since boosting algorithms are supervised learning algorithms, a large number of labeled positive and negative examples must be input to the training process. Besides the examples, some learning parameters must be specified. The most important parameter is the desired true positive and false positive rate for each stage of the cascade. We train one stage at a time until that stage achieves the specified true positive and false positive rates. Then a new stage is begun, and the process continues until some stopping criterion is reached. Only the positive examples that are correctly classified by the previous stages and the negative examples that are incorrectly classified by the previous stages are used to train each new stage.

Skin Detector

There are many approaches to segmenting regions with similar color and texture from other regions. To extract skin color blobs from images, color information is the obvious choice. The skin detector for our system need not be extremely robust but it should be fast.

The Bayesian maximum likelihood classifier based on color histograms, as presented by Zarit et al. (1999), meets all of these needs. Based on their results and our own follow-up study, we selected the HS (hue and saturation) color model. Histograms used in the skin detector have two dimensions, namely hue and saturation. Each axis of the plane is quantized into 16 bins, so that each histogram will have $16^2 = 256$ bins. We selected 16-bin quantization based on comparison experiments with different bins counts of 8, 16, 32, and 64. We found that 16 bins along each axis gave the best performance. The reasons for excluding the intensity component from the histogram are to eliminate the effect of non-uniform illumination and to save computational cost. We construct histograms for skin and non-skin pixels from a large training set. The histogram counts are used to construct a discrete class-conditional likelihood for a Bayesian maximum likelihood classifier which we then use to determine whether a given pixel is most likely skin or not skin.

Each image patch which is classified as a hand by the cascade is scaled to a standard size 24×24 pixels and then fed to the skin detector, which produces a binary image, in which the value 1 represents a putative skin pixel and the value 0 represents a non-skin pixel.

Features Extractor and Mahalanobis Classifier

The shape and relative size of the skin blob within the detection window give useful information for discriminating image patches containing hand from those not containing hand. We extract four simple features from the binary skin image that are surprisingly useful for accurate classification:

1. The area of the largest connected component of skin pixels.

2. The length of the perimeter of the largest connected component of skin pixels.
3. The eccentricity of the largest connected component of skin pixels.
4. The number of pixels on the boundary of the largest connected component of skin pixels that intersect the detection window boundary.

The area feature is simply the number of pixels in the largest connected skin component; it is normalized by the total number of skin pixels ($24 \times 24 = 576$) in the image patch. It is very obvious that the given image patch is unlikely to contain a hand if the area feature is very large or very small. The perimeter feature is the total number of pixels on the perimeter of the largest connected skin component; it is normalized in the same way as the area feature. The eccentricity feature is the eccentricity of the ellipse having the same second moments as the largest connected skin component, i.e., the ratio of the distance between the foci of the ellipse and its major axis length. The eccentricity is between 0 and 1, with 0 indicating a circle and 1 indicating a line segment. This feature helps to discriminate face skin regions, which tend to be quite round, from true hand skin regions, which tend to be more eccentric. Finally, the boundary feature helps to discriminate between arm skin regions, which tend to intersect the boundary of the detection window in two places, from true hand skin regions, which only intersect the detection window at the wrist. The boundary feature provides information about how wrist-like the boundary is.

No matter how good those four features are, they will not be efficiently utilized for classification without a suitable classifier. We prefer classifiers that are simple with few parameters to tune. We find that a simple classifier based on Mahalanobis distance is a reasonable choice. Each image patch can be represented by a feature vector consisting

of the area, perimeter, eccentricity, and boundary features. To classify a given feature vector x as a true hand or not a hand, we calculate the Mahalanobis distance

$$d(x) = (x - \mu)^T \Sigma^{-1} (x - \mu)$$

between the feature vector x and the mean feature vector μ, then we classify x as a hand if $d(x)$ is less than some threshold θ. Here the mean hand feature vector μ, the covariance matrix Σ, and the distance threshold θ are estimated from the training set.

Once classification for each possible detection windows is done, the positively detected hands are fed to the final module, the grouping, filtering, and averaging module. Further reduction of false positives is done there.

Grouping, Filtering, and Averaging Module

Our Mahalanobis classifier produces a few very sparsely distributed false positives and densely distributed true detections around the actual targets. Since it produces several true detections around each of the actual detections, grouping and averaging is necessary to ensure only one detection for each target. A group which contains less than some number of detections can be disposed of on the assumption it is a false positive. We use the existing implementation of this technique in the OpenCV. The positively detected hands output from this module could then be forwarded to another component in an integrated application, for example a gesture recognition module. But, in this article we simply evaluate the performance and efficiency of the proposed algorithm on a series of video sequences. We now describe our experiments in detail.

Data Acquisition

For the purpose of training, testing and evaluation of the proposed hand detection system, we captured 12 video sequences in a moderately cluttered laboratory environment. Four people volunteered to be models, and we captured three video sequences for each person. In the first sequence, each model walked away from the camera then came back to the starting position, in a direction parallel to the camera angle. In the second sequence, each model walked back and forth across the field of view in a direction perpendicular to the camera angle, at three different distances from the camera. In the last sequence, each model walked diagonally across the field of view, starting from a position to the right or left of the camera then returned to the start position, and repeated the procedure beginning from the other side of the camera.

We captured the video sequences at 15 frames per second with an inexpensive IEEE1394 Web camera at a resolution of 640×480 pixels. Each sequence lasted approximately 30 seconds. After video capture, all visible hands not smaller than the standard size of 24×24 pixels in every image of all 12 sequences were manually located. A total of 2246 hand locations were obtained. Our criteria for locating the selection window on the hand was that the hand should be roughly at the center of the window while taking up about 50% of the pixel area of the selection window. Some examples are shown in Figure 3.

Of the 12 video sequences, 11 were used to train the system and the remaining sequence was reserved for testing and evaluating the complete hand detection system. To train the boosted classifier cascade, we used 2000 hands as positive examples, and negative examples were automatically extracted from a set of background images. As background images, we used four randomly selected images from the video sequences

that did not contain any human. We created an additional set of background images using six randomly selected images containing humans. From each image, we cut out two large regions that did not containing hands but did contain other body parts such as faces and arms. From the test image sequence, we selected 99 images, each of containing at least one hand not smaller than 24×24 pixels. These 99 images contained a total of 106 proper hands. All of our test evaluation calculations are based on those 106 proper hands.

We also prepared a holdout set by randomly selecting 100 images from the 11 training video sequences. This holdout set was used to monitor system performance as well as to tune system parameters.

Boosted Classifier Training

To train the classifier cascade, we used Lienhart and colleagues' approach, implemented in OpenCV. We used the previously-described 2000 manually located hands from the eleven training video sequences as positive examples and the 16 previously-described background images.

The important parameters of the training process are the minimum hit rate (true positive rate) and maximum false alarm rate (false positive rate). Every stage in the cascade must satisfy these criteria on the training set. We used 100% for the hit rate and 60% for the false alarm rate. This means when adding a new stage to the classifier, the training system keeps adding additional weak classifiers to that stage until it correctly classifies all of the positive training examples with at most a 60% false alarm rate. Lienhart and colleagues' training system extracts the desired number of negative examples, 4000 for our experiment, by scanning a window with different scales over the

background images. After training one stage of the classifier, the negative examples which are correctly classified are disposed of and the system extracts a sufficient number of new negative examples. We use the Gentle AdaBoost variant of AdaBoost and the full Haar-like feature set of Lienhart and Maydt (2002).

The performance of the cascade is tested on the holdout set every time a new stage is constructed and added to the cascade. The results of the training process will be discussed in more detail in the Results and Discussion section.

Skin Detector Training

To train our skin detector, we selected 10 images containing one or more humans from a set of independent video sequences captured under various lighting conditions at several different locations. Skin pixels on those images were manually marked and the resulting 70,475 skin and 1,203,094 non-skin pixels were fed to the skin detector training process. The training process computes the hue (H) and saturation (S) for each pixel and quantizes each value into one of 16 bins. From the quantized values of skin pixels, one 2D histogram is constructed, and another is constructed from the quantized values of the non-skin pixels. Both histograms are constructed by simply counting the number of pixels which belong to same bin, and they are normalized by the total number of pixels used to construct the histogram.

Mahalanobis Classifier Training

The purpose of the Mahalanobis classifier is to eliminate the false detections made by the boosted classifier cascade while still maintaining a high detection rate. As the detection window is scanned over every image in the training set, the boosted

classifier outputs both true positive and false positive image patches. We found 78,658 true positives on our training set then randomly selected 6,000 true positives for computing the mean feature vector μ and covariance matrix Σ for the Mahalanobis classifier.

After we obtain μ and Σ for the Mahalanobis classifier, we need to find the optimum threshold. To do so, we scanned a detection window over every image in the holdout set and separated the detected image patches into false positives and true positives using the known hand locations for the holdout set. We extracted the Mahalanobis classifier's four features from each detected image patches and calculated the Mahalanobis distance between the feature vector of each image patch and the mean feature vector μ. As the class for each image patch is known, we plotted the ROC curve as shown in Figure 4. At this point, a detection rate of less than 100% is acceptable because the classifier cascade typically produces multiple true detections around each hand. Examining the ROC curve, we found that a Mahalanobis distance of 2.9 is a reasonable threshold since this threshold gives a very low false positive rate (6%) while giving an acceptable true positive rate (60%) on the image patches output by the classifier cascade.

Parameter Tuning for the System

Once all the required building blocks for hand detection are in place, we need to specify one last parameter, i.e., the minimum number of nearby positive image patches required for the Group, Filter, and Average block. In practice, this parameter must be tuned to achieve a good detection rate. To tune this parameter, we assembled all of the building blocks into a complete system then tested it on the holdout set with various

values for neither parameter. We found that a minimum of 4 neighboring patches produced the optimal result: 81.8% of the hands in the holdout set were detected and the false positive rate was also relatively low, an average of 1.55 false positives per image.

Testing the Complete System

We tested the complete hand detection system on the test set that was never used in any part of the training process. As previously described we used 99 images containing 106 hands in known locations. The detailed results of the test are discussed in the next section.

Results and Discussion

During the training process, we monitored the performance of the cascade and found that 12 stages of strong classifiers gave the optimum performance. The 12-stage classifier had a 97.5% detection rate on the holdout set, while having a reasonably low false positive rate of about 0.3% on the holdout set. A false positive rate of 0.3% may seem quite low but in fact this means we had an average of 1,000 false positive detections per image because one image contains more than 300,000 possible image patches. Clearly, these results indicate that a post processor is necessary to further eliminate false positives if the system is to be useable in practical applications.

When we tested our system on the test set, we found that the boosted classifier cascade frequently detected incorrect body parts such as arms, as shown in the left half of Figure 5 (b). However, the skin detection images shown in the right halves of Figures 5 (a) and 5 (b) show that the Mahalanobis classifier's boundary feature can distinguish between these cases. We found that most of the remaining false positives contained

either too few skin pixels or sparsely distributed skin pixels. These cases are easily eliminated by the Mahalanobis classifier's area feature since it operates on the largest connected skin component.

Our final hand detector detects 92 hands (86.8%) of the 106 hands in the test set, with an acceptable false positive rate of 1.19 false detections per image on average. A detection rate of 86.8% will enable many applications based on hand detection, such as human action recognition systems for security. Images (a) and (b) in Figure 6 illustrate example detections by our complete system, and all detected hands in the test set are shown in Figure 7. In the example, all hands were detected in both images, and only one false detection occurred in each image. The false detection of the desktop computer in the middle of the image is present in almost every image because the computer's color and texture are in fact similar to that of a hand. This kind of false positive detection on a stationary object will be eliminated if we add motion information between two consecutive frames in the video sequence.

Conclusion

From the literature, we know that hand detectors incorporating AdaBoost and Haar-like features perform quite well in applications like sign-language recognition, in which images are relatively high resolution with less cluttered background and constrained hand gesture. These approaches suffer from high false positive rates and low detection rates when applied to detect less constrained hands in low resolution and cluttered images. However we find that these limitations can be overcome with the help of a simple but efficient post processing system – in our experiments, the prototype hand detection system achieved excellent performance on its test set. One important

limitation of this work is that both the training and testing image sequences were captured in the same environment. This means that the performance of our current system is likely background dependent; if so, the reported performance is optimistic. However, the current results are encouraging, and in future work we plan to explore integrating our system with gesture recognition and human action recognition systems.

Acknowledgement

We thank the member of the Image and Vision Computing Laboratory at the Sirindhorn International Institute of Technology for the participation in our data collection efforts. This research was partially supported by Thailand Research Fund grant MRG4780209 to Matthew N. Dailey.

References

Barreto, J., Menezes, P. and Dias, J. 2004. Human-robot interaction based on haar-like features and eigenfaces. Proceedings of the 2004 IEEE Conference on Robotics and Automation, 2004,1888-1893.

Caglar, M.B. and Lobo, N. 2006. Open hand detection in a cluttered single image using finger primitives. Proceeding of the 2006 Computer Vision and Pattern Recognition Workshop, June 17-22, 2006.

Freund Y. and Shapire, R.E. 1997. A decision-theoretic generalization of online learning and an application to boosting. Journal of Computer and System Sciences 5(1):119-139.

Intel Corporation. Open Computer Vision Library (software). 2006. Open source software available at http://sourceforge.net/projects/opencv/.

Lienhart, R. and Maydt, J. 2002. An extended set of Haarlike features for rapid object detection. Proceedings of the IEEE International Conference on Image Processing, 2002, 900-903.

Lienhart, R., Kuranov, A. and Pisarevsky, V. 2002. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, Microprocessor Research Lab, Intel Labs.

Ong, E. and Bowden, R. 2004. A boosted classifier tree for hand shape detection. Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004, 889-894.

Shamaie, A. and Sutherland, A. 2005. Hand tracking in bimanual movements. Image and Vision Computing 23(13):1131-1149.

Soontranon, N., Aramvith, S. and Chalidabhongse, T.H. 2004. Face and hands localization and tracking for sign language recognition. International Symposium on Communications and Information Technologies, 2004, 1246-1251.

Varona, J., Buades, J.M. and Perales, F.J. 2005. Hands and face tracking for VR applications, Computers & Graphics 29(2):179-187.

Viola P.A. and Jones, M.J. 2004. Robust real-time face detection. International Journal of Computer Vision 57(2):137-154.

Viola P.A. and Jones, M.J. 2001. Rapid object detection using a boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001, 511-518.

Wachs, J., Stern, H., Edan, Y., et al. 2005. A real-time hand gesture system based on evolutionary search. Genetic and Evolutionary Computation Conference, 2005.

Wren, C.R., Azarbayejani, A., Darrell, T. and Pentland, A. 1997. Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7): 780-785.

Zarit, B.D., Super, B.J. and Quek, F.K.H. 1999. Comparison of five color models in skin pixel classification. International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-Time Systems, 1999, 58–63.

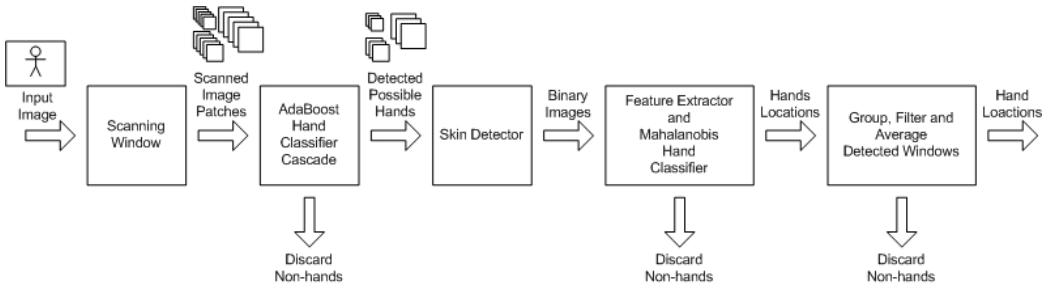
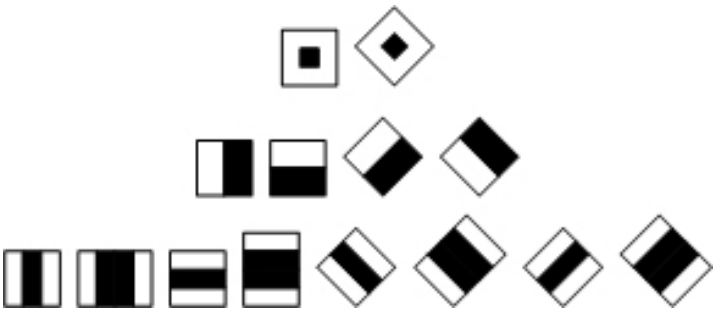


Figure 1, Hand detection system architecture.

(a)



(b)

Figure 2, Haar-like features used to construct weak classifies in the boosted classifier cascade. (a) Features used by Viola and Jones. (b) Features used by Lienhart and colleagues.

Figure 3, Example training images Scaled to 24x24.

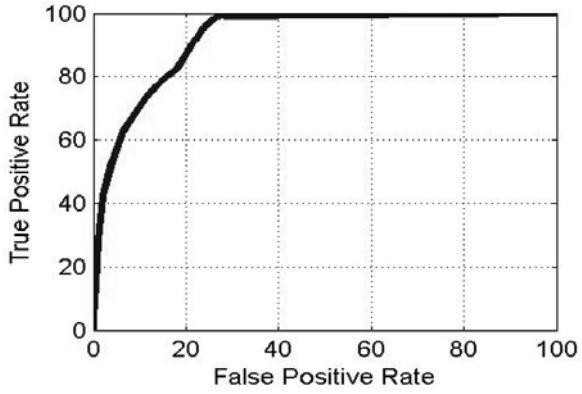
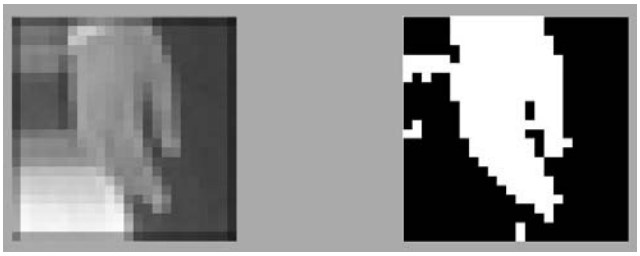
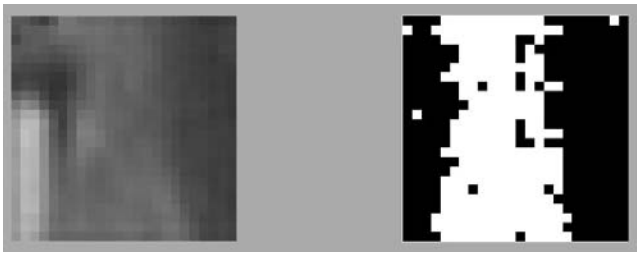


Figure 4, ROC cure between true positive and false positive for different threshold on Mahalanobis distance. True positive and false positive rate are calculated based on number of true detection and false detection input to the Mahalanobis Classifier.



(a) Properly detected hand.



(b) Non-hand body part detected as hand.

Figure 5, Original (left) image patches detected as hand by boosted hand classifier and binary images patches (right) after skin detection.

(a)

(b)

Figure 6, Example detection results of our proposed hand detection system.

Figure 7, Hands detected by our complete hand detector system. All detections are scaled down to standard size 24x24 pixels for easy visualization.

A Family of Quadratic Snakes for Road Extraction

Ramesh Marikhu[1] Matthew N. Dailey[2] Stanislav Makhanov[3] Kiyoshi Honda[4]

[1] Information and Communication Technologies, Asian Institute of Technology
[2] Computer Science and Information Management, Asian Institute of Technology
[3] Sirindhorn International Institute of Technology, Thammasat University
[4] Remote Sensing and GIS, Asian Institute of Technology

Abstract. The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures from satellite imagery. Quadratic snakes allow global interactions between points along a contour, and are well suited to segmentation of linear structures such as roads. However, a single quadratic snake is unable to extract disconnected road networks and enclosed regions. We propose to use a family of cooperating snakes, which are able to split, merge, and disappear as necessary. We also propose a preprocessing method based on oriented filtering, thresholding, Canny edge detection, and Gradient Vector Flow (GVF) energy. We evaluate the performance of the method in terms of precision and recall in comparison to ground truth data. The family of cooperating snakes consistently outperforms a single snake in a variety of road extraction tasks, and our method for obtaining the GVF is more suitable for road extraction tasks than standard methods.

1 Introduction

The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures and regions of interest from satellite imagery. In particular, *automated road extraction* would boost the productivity of technicians enormously. This is because road networks are among the most important landmarks for mapping, and manual marking and extraction of road networks is an extremely slow and laborious process. Despite years of research and significant progress in the computer vision and image processing communities (see, for example, [1, 2] and Fortier et al.'s survey [3]), the methods available thus far have still not attained the speed and accuracy necessary for practical application in GIS tools.

Among the most promising techniques for extraction of complex objects like roads are *active contours* or *snakes*, originally introduced by Kass et al. [4]. Since the seminal work of Kass and colleagues, techniques based on active contours have been applied to many object extraction tasks [5] including road extraction [6].

Rochery et al. have recently proposed *higher-order active contours*, in particular *quadratic snakes*, which hold a great deal of promise for extraction of linear

structures like roads [7]. The idea is to use a quadratic formulation of the contour's geometric energy to encourage anti-parallel tangents on opposite sides of a road and parallel tangents along the same side of a road. These priors increase the final contour's robustness to partial occlusions and decrease the likelihood of false detections in regions not shaped like roads.

In this paper, we propose two heuristic modifications to Rochery et al.'s quadratic snakes, to address limitations of a single quadratic snake and to accelerate convergence to a solution. First, we introduce the use of a *family* of quadratic snakes that are able to split, merge, and disappear as necessary. Second, we introduce an improved formulation of the image energy combining Rochery et al.'s oriented filtering technique [7] with thresholding, Canny edge detection, and Xu and Prince's Gradient Vector Flow (GVF) [8]. The modified GVF field created using the proposed method is very effective at encouraging the quadratic snake to snap to the boundaries of linear structures. We demonstrate the effectiveness of the family of snakes and the modified GVF field in a series of experiments with real satellite images, and we provide precision and recall measurements in comparison with ground truth data. The results are an encouraging step towards the ultimate goal of robust, fully automated road extraction from satellite imagery.

As a last contribution, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Matlab platform. The system is freely available as open source software [9].

2 Experimental Methods

2.1 Quadratic snake model

Here we provide a brief overview of the quadratic snake proposed by Rochery et al. [7]. An *active contour* or *snake* is parametrically defined as

$$\boldsymbol{\gamma}(p) = \left[x(p)\ y(p)\right]^T, \tag{1}$$

where p is the curvilinear abscissa of the contour and the vector $\left[x(p)\ y(p)\right]^T$ defines the Cartesian coordinates of the point $\boldsymbol{\gamma}(p)$. We assume the image domain Ω to be a bounded subset of $\mathbb{R}^2$.

The energy functional for Rochery et al.'s *quadratic snake* is given by

$$E_s(\boldsymbol{\gamma}) = E_g(\boldsymbol{\gamma}) + \lambda E_i(\boldsymbol{\gamma}), \tag{2}$$

where $E_g(\boldsymbol{\gamma})$ is the *geometric energy* and $E_i(\boldsymbol{\gamma})$ is the *image energy* of the contour $\boldsymbol{\gamma}$. λ is a free parameter determining the relative importance of the two terms.

The geometric energy functional is defined as

$$E_g(\boldsymbol{\gamma}) = L(\boldsymbol{\gamma}) + \alpha A(\boldsymbol{\gamma}) - \frac{\beta}{2} \iint \boldsymbol{t}_{\boldsymbol{\gamma}}(p) \cdot \boldsymbol{t}_{\boldsymbol{\gamma}}(p')\ \Psi\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right)\ dp\ dp', \tag{3}$$

where $L(\boldsymbol{\gamma})$ is the length of $\boldsymbol{\gamma}$ in the Euclidean metric over Ω, $A(\boldsymbol{\gamma})$ is the area enclosed by $\boldsymbol{\gamma}$, $\boldsymbol{t}_{\boldsymbol{\gamma}}(p)$ is the unit-length tangent to $\boldsymbol{\gamma}$ at point p, and $\Psi(z)$, given the

distance z between two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of each term. Clearly, for positive β, $E_g(\boldsymbol{\gamma})$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The interation function $\Psi(z)$ is a smooth function expressing the radius of the region in which parallel tangents should be encouraged and anti-parallel tangents should be discouraged. $\Psi(z)$ incorporates two constants: d, the expected road width, and ϵ, the expected variability in road width. During snake evolution, weighting by $\Psi(z)$ in Equation 3 discourages two points with anti-parallel tangents (the opposite sides of a putative road) from coming closer than distance d from each other.

The image energy functional $E_i(\boldsymbol{\gamma})$ is defined as

$$\begin{aligned} E_i(\boldsymbol{\gamma}) = & \int \boldsymbol{n}_{\boldsymbol{\gamma}}(p) \cdot \nabla I(\boldsymbol{\gamma}(p))\ dp \\ & - \iint \boldsymbol{t}_{\boldsymbol{\gamma}}(p) \cdot \boldsymbol{t}_{\boldsymbol{\gamma}}(p')\ \nabla I(\boldsymbol{\gamma}(p)) \cdot \nabla I(\boldsymbol{\gamma}(p'))\ \Psi(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|)\ dp\ dp', \end{aligned} \tag{4}$$

where $I : \Omega \rightarrow [0, 255]$ is the image and $\nabla I(\gamma(p))$ denotes the 2D gradient of I evaluated at $\gamma(p)$. The first linear term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand to enclose dark regions surrounded by light roads.[5] The quadratic term favors nearby point pairs with two different configurations, one with parallel tangents and parallel gradients and the other with anti-parallel tangents and anti-parallel gradients.

After solving the Euler-Lagrange equations for minimizing the energy functional $E_s(\gamma)$ (Equation 2), Rochery et al. obtain the update equation

$$\begin{aligned} \boldsymbol{n}_{\boldsymbol{\gamma}}(p) \cdot \frac{\partial E_s}{\partial \boldsymbol{\gamma}}(p) = & -\kappa_{\boldsymbol{\gamma}}(p) - \alpha - \lambda \|\nabla I(\boldsymbol{\gamma}(p))\|^2 + G(\boldsymbol{\gamma}(p)) \\ & + \beta \int \boldsymbol{r}\left(\boldsymbol{\gamma}(p), \boldsymbol{\gamma}(p')\right) \cdot \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\ \ \Psi'\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right) dp' \\ + 2\lambda \int \boldsymbol{r}\left(\boldsymbol{\gamma}(p), \boldsymbol{\gamma}(p')\right) \cdot \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\ \ & \left(\nabla I(\boldsymbol{\gamma}(p)) \cdot \nabla I(\boldsymbol{\gamma}(p'))\right)\ \ \Psi'\left(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|\right) dp' \\ & + 2\lambda \int \nabla I(\boldsymbol{\gamma}(p')) \cdot \left(\nabla \nabla I(\boldsymbol{\gamma}(p)) \times \boldsymbol{n}_{\boldsymbol{\gamma}}(p')\right)\ \Psi(\|\boldsymbol{\gamma}(p) - \boldsymbol{\gamma}(p')\|)\ dp', \end{aligned} \tag{5}$$

where $\kappa_{\boldsymbol{\gamma}}(p)$ is the curvature of $\boldsymbol{\gamma}$ at $\boldsymbol{\gamma}(p)$ and $G(\boldsymbol{\gamma}(p))$ is the "specific energy," evaluated at point $\boldsymbol{\gamma}(p)$ (Section 2.2). $\boldsymbol{r}\left(\gamma(p), \gamma(p')\right) = \frac{\gamma(p) - \gamma(p')}{\|\gamma(p) - \gamma(p')\|}$ is the unit

[5] For dark roads in light background, we negate all the terms involving image, including $G(\boldsymbol{\gamma}(p))$ in Equation 5. In the rest of the paper, we assume light roads on a dark background.

vector pointing from $\gamma(p)$ towards $\gamma(p')$. $\nabla\nabla I(\gamma(p))$ is the Hessian of I evaluated at $\gamma(p)$.

α, β, and λ are free parameters that need to be determined experimentally. d and ϵ are specified a priori according to the desired road width. Following Rochery et al., we normally initialize our quadratic snakes with a rounded rectangle covering the entire image.

2.2 Oriented filtering

We use Rochery's oriented filtering method [10] to enhance linear edges in our satellite imagery. The input image is first convolved with oriented derivative-of-Gaussian filters at various orientations. Then the minimum (most negative) filter response over the orientations is run through a ramp function equal to 1 for low filter values and -1 for high filter values. The thresholds are user-specified. An example is shown in Fig. 1(b).

2.3 GVF energy

Rather than using the oriented filtering specific image energy $G(\boldsymbol{x})$ from Section 2.2 for snake evolution directly, we propose to combine the oriented filtering approach with Xu and Prince's Gradient Vector Flow (GVF) method [8]. The GVF is a vector field $V^{\mathrm{GVF}}(\boldsymbol{x}) = \left[u(\boldsymbol{x})\ v(\boldsymbol{x})\right]^T$ minimizing the energy functional

$$E(V^{\mathrm{GVF}}) = \int_{\Omega} \mu(u_x^2(\boldsymbol{x}) + u_y^2(\boldsymbol{x}) + v_x^2(\boldsymbol{x}) + v_y^2(\boldsymbol{x})) + \|\nabla\tilde{I}(\boldsymbol{x})\|^2\ \|V(\boldsymbol{x}) - \nabla\tilde{I}(\boldsymbol{x})\|^2\ d\boldsymbol{x}, \tag{6}$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $v_x = \frac{\partial v}{\partial x}$, $v_y = \frac{\partial v}{\partial y}$, and $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term inside the integral encourages a smooth vector field whereas the second term encourages fidelity to $\nabla\tilde{I}$. μ is a free parameter controlling the relative importance of the two terms.

Xu and Prince [8] experimented with several different methods for obtaining $\nabla\tilde{I}$. We propose to perform Canny edge detection on G (the result of oriented filtering and thresholding, introduced in Section 2.2) to obtain a binary image $\tilde{I}$ for GVF, then to use the resulting GVF V^{GVF} as an additional image energy for quadratic snake evolution. The binary Canny image is ideal because it only includes information about road-like edges that have survived sharpening by oriented filters. The GVF field is ideal because during quadratic snake evolution, it points toward road-like edges, pushing the snake in the right direction from a long distance away. This speeds evolution and makes it easier to find suitable parameters to obtain fast convergence. Fig. 1 compares our method to alternative GVF formulations based on oriented filtering or Canny edge detection alone.

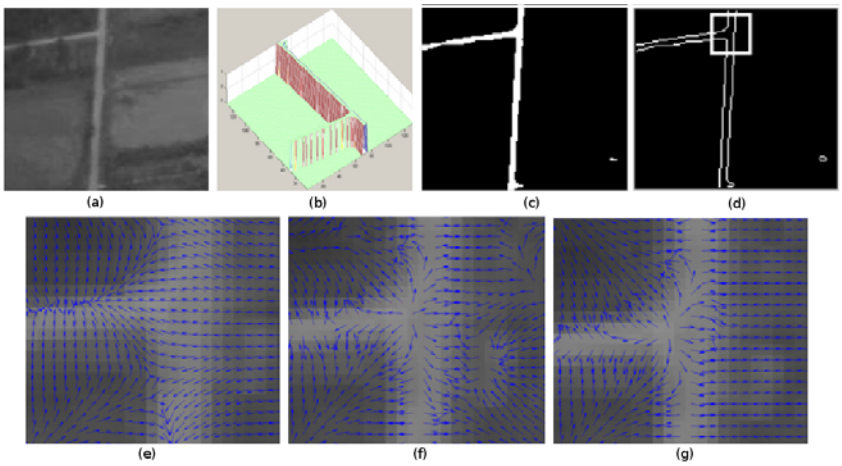


Fig. 1. Comparison of GVF methods. (a) Input image. (b) $G(\boldsymbol{x})$ obtained from oriented filtering on $I(\boldsymbol{x})$. (c) Image obtained from $G(\boldsymbol{x})$ using threshold 0. (d) Canny edge detection on (c), used as $\tilde{I}$ for GVF. (e-f) Zoomed views of GVFs in region delineated in (d). (e) Result of using the magnitude of the gradient $\nabla(G_\sigma * I)$ to obtain $\tilde{I}$. (f) Result of using Canny edge detection alone to obtain $\tilde{I}$. (g) GVF energy obtained using our proposed edge image. This field pushes most consistently toward the true road boundaries.

2.4 Family of quadratic snakes

A single quadratic snake is unable to extract enclosed regions and multiple disconnected networks in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, merge, and disappear as necessary.

In our formulation, due to the curvature term $\kappa_\gamma(p)$ and the area constant α in Equation 5, specifying the points on γ in a counterclockwise direction creates a *shrinking snake* and specifying the points on γ in a clockwise direction creates a *growing snake*.

An enclosed region (loop or a grid cell) can be extracted effectively by initializing two snakes, one shrinking snake covering the whole road network and another growing snake inside the enclosed region.

On the one hand, our method is heuristic and dependent on somewhat intelligent user initialization, but it is much simpler than level set methods for the same problem [7], and, assuming a constant number of splits and merges per iteration, it does not increase the asymptotic complexity of the quadradic snake's evolution.

Splitting a snake We split a snake into two snakes whenever two of its arms are squeezed too close together, i.e. when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other

in both directions of traversal around the contour. d^{split} should be less than 2η, where η is the maximum step size.

Merging two snakes Two snakes are merged when they have high curvature points within a distance d^{merge} of each other, the two snakes' order of traversal (clockwise or counterclockwise) is the same, and the tangents at the two high curvature points are nearly antiparallel. High curvature points are those with $\kappa_\gamma(p) > 0.6\kappa_\gamma^{\max}$ where $\kappa_\gamma^{\max}$ is the maximum curvature for any point on γ. High curvature points are taken to ensure merging only occurs if two snakes have the semi-circular tip of their arms facing each other. Filtering out the low curvature points necessitates computing angle between the tangents at two points only for the high curvature points.

When these conditions are fulfilled, the two snakes are merged by deleting the high curvature points and joining the snakes into a single snake while preserving the direction of traversal for the combined snake.

Deleting a snake A snake γ is deleted if it has low compactness ($\frac{4\pi A(\boldsymbol{\gamma})}{L(\boldsymbol{\gamma})^2}$) and a perimeter less than L^{delete}.

2.5 Experimental design

We analyze extraction results on different types of road networks using the single quadratic snake proposed by Rochery et al. [7] and the proposed family of cooperating snakes. The default convergence criterion is when the minimum $E_s(\boldsymbol{\gamma})$ has not improved for some number of iterations.

Experiments have been performed to analyze the extraction of tree-structured road networks and those with loops, grids and disconnected networks.

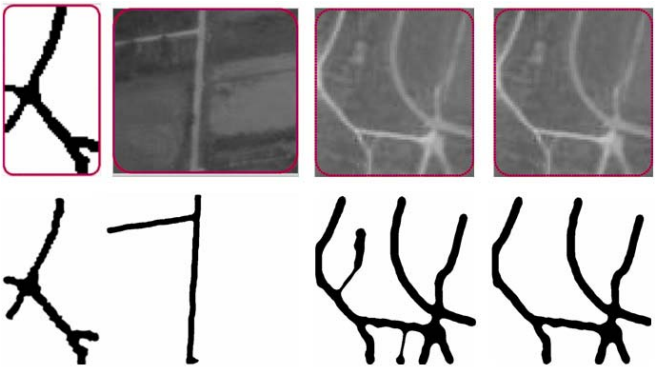
We then analyze the effectiveness of GVF energy obtained from the proposed edge image in Experiment 4. For all the experiments, we digitize the images manually to obtain the ground truth data necessary to compute precision and recall.

3 Results

We have obtained several parameters emperically. For splitting a snake, d^{split} should be less than `d`. k to be chosen depending on how far the two splitting points should be to ensure that the snakes formed after splitting have at least k points.

In order to ensure that merging of snakes takes place only among the arms with the semi-circular tips facing each other, the tangents at the high curvature points are checked for antiparallel threshold of $130\pi/180$..

The compactness should be greater than 0.2 to ensure that linear structured contours are not deleted.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	480	0.937	0.8605	0.75	0.35	4	3
2	single	308	0.9483	0.8503	0.15	0.2	3.5	4
3	single	473	0.7091	0.9074	0.1	0.2	4	3
4	family	450	0.8011	0.9115	0.1	0.2	4	3

Fig. 2. Evolution of quadratic snake on roads with tree structure.Each column displays an image with initial contour in red and the extracted road network below it.

3.1 Experiment 1: Simple (tree-structured) road networks

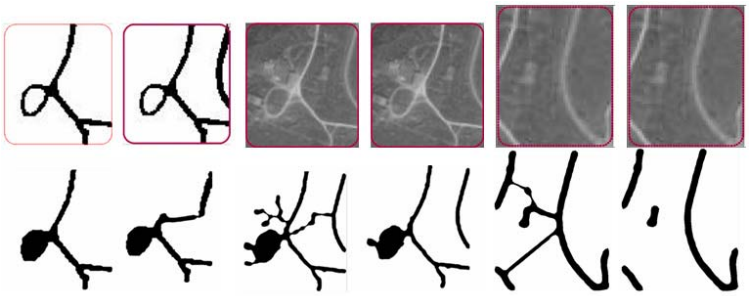
A single quadratic snake is well suited for tree-structured road networks as the snake will not need to change its topology during evolution (Figure 2). A family of snakes enable faster and better road extraction as non-road regions are eliminated using splitting and deletion of snakes.

3.2 Experiment 2: Road networks with single loop and multiple disconnected networks

The family of quadratic snakes are able to extract disconnected networks with high accuracy (Figure 3) but are not able to extract enclosed regions automatically as the snakes are not able to develop holes inside it in the form of growing snakes.

3.3 Experiment 3: Complex road networks

A road network is considered complex if it has multiple disconnected networks and enclosed regions and large number of branches. With the appropriate user initialization (Figure 4), the snakes are able to extract the road networks with high accuracy and in less time.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	1080	0.6364	0.9833	0.75	0.35	4	3
2	single	1300	0.5638	0.7322	0.75	0.35	4	3
3	single	182	0.5473	0.9114	0.1	0.2	4	2
4	family	182	0.6344	0.9833	0.1	0.2	4	2
5	single	219	0.6823	0.8673	0.1	0.2	3	2.5
6	family	163	0.8582	0.8731	0.1	0.2	3	2.5

Fig. 3. Evolution of quadratic snake on roads with loops and disconnected networks. Each column displays an image with initial contour in red and the extracted road network below it.

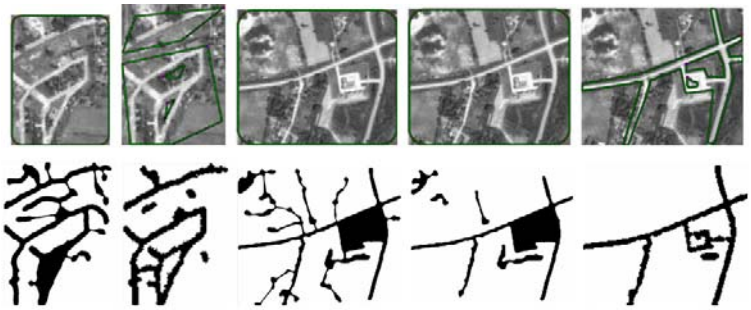
3.4 Experiment 4: GVF energy to enable faster evolution

The Gradient Vector Flow Field [8] boosts the evolution process as we can see from the number of iterations required for each evolution in Experiment 4 with and without the use of GVF energy. From the evolution in the fifth column, we see that the snake was able to extract the network with greater detail. Also, from the evolution in the last column, we see that it is necessary for the quadratic image energy to enable robust extraction and thus the GVF weight and λ need to be balanced appropriately.

4 Discussion and Conclusion

In Experiment 1, we found that the our modified quadratic snake is able to move into concavities to extract entire tree-structured road networks with very high accuracy. Experiment 2 showed that the family of quadratic snakes is effective at handling changes in topology during evolution, enabling better extraction of road networks. Currently, loops cannot be extracted automatically.

We demonstrated the difficulty in extracting complex road networks with multiple loops and grids in Experiment 3. However, user initialization of a family of contours enable extraction of multiple closed regions and help the snake to avoid road-like regions. The level set framework could be used to handle change in topology enabling effective extraction of enclosed regions. Rochery et al. [10] evolved the contour using the level set methods introduced by Osher and Sethian.



Column	Snakes	Iter	Precision	Recall	α	β	λ	d
1	single	519	0.5301	0.9067	0.1	0.25	3	3.5
2	user defined	166	0.7675	0.9117	0.1	0.25	3	3.5
3	single	450	0.3851	0.8171	0.05	0.05	2	4
4	family	350	0.5058	0.7987	0.05	0.05	2	4
5	user defined	104	0.799	0.8951	0.1	0.25	4	4

Fig. 4. Evolution of quadratic snake on roads with enclosed regions. Each column displays an image with initial contour in green and the extracted road network below it.

However, our method is faster, conceptually simpler, and a direct extension of Kass et al.'s computational approach.

In Experiment 4, we found that faster and robust extraction is achieved using oriented filtering and GVF energy along with image energy of the quadratic snakes. Our proposed edge image obtained from oriented filtering is effective for computing GVF energy to enhance the process of extraction. We also found that our method for obtaining the GVF outperforms standard methods.

Finally, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Matlab platform. The system is freely available as open source software [9].

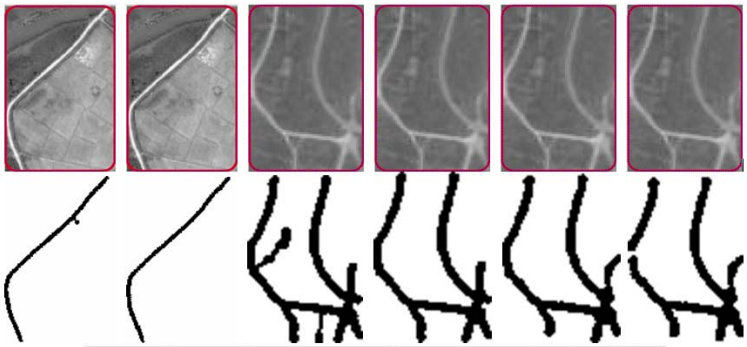
Future work will focus on possibilities to automate the extraction of enclosed regions. Digital elevation models could be integrated with image energy for increased accuracy.

Acknowledgments

This research was supported by Thailand Research Fund grant MRG4780209 to MND. RM was supported by a graduate fellowship from the Nepal High Level Commission for Information Technology.

References

1. Fischler, M., Tenenbaum, J., Wolf, H.: Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique. Computer Graphics and Image Processing **15** (1981) 201–223



Column	Snakes	Iter	Precision	Recall	GVF Wt	α	β	λ	d
1	single	670	0.8789	0.9171	0	0.2	0.15	4	2.5
2	single	390	0.9066	0.9129	1	0.2	0.15	4	2.5
3	single	950	0.7321	0.9467	0	0.05	0.2	4	3
4	family	650	0.8279	0.9382	0	0.05	0.2	4	3
5	family	430	0.8017	0.9094	1	0.05	0.2	2	3
6	family	400	0.7925	0.8998	1	0.05	0.2	0	3

Fig. 5. Evolution of quadratic snake on roads with enclosed regions. Each column displays an image with initial contour in green and the extracted road network below it.

2. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(1) (1996) 1–14
3. Fortier, A., Ziou, D., Armenakis, C., Wang, S.: Survey of work on road extraction in aerial and satellite images. Technical Report 241, Université de Sherbrooke, Quebec, Canada (1999)
4. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1**(4) (1987) 321–331
5. Cohen, L.D., Cohen, I.: Finite-element methods for active contour models and balloons for 2-D and 3-D images. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(11) (1993) 131–147
6. Laptev, I., Mayer, H., Lindeberg, T., Eckstein, W., Steger, C., Baumgartner, A.: Automatic extraction of roads from aerial images based on scale space and snakes. Machine Vision and Applications **12**(1) (2000) 23–31
7. Rochery, M., Jermyn, I.H., Zerubia, J.: Higher order active contours. International Journal of Computer Vision **69**(1) (2006) 27–42
8. Xu, C., Prince, J.L.: Gradient Vector Flow: A new external force for snakes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (1997) 66–71
9. Marikhu, R.: A GUI environment for road extraction with quadratic snakes (2007) Matlab software, available at http://www.cs.ait.ac.th/∼mdailey/snakes.
10. Rochery, M.: Contours actifs d'order supérieur et leur application à la détection de linéiques dans des images de télédétection. PhD thesis, Université de Nice, Sophia Antipolis — UFR Sciences (2005)

Multiple Quadratic Snakes for Road Extraction ⋆

Matthew N. Dailey [a,*], Stanislav Makhanov [b], and Ramesh Marikhu [a]

[a] *Computer Science and Information Management*
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120 Thailand

[b] *Sirindhorn International Institute of Technology*
Thammasat University
131 M. 5, Tiwanont Road, Bangkadi, A. Muang, Pathumthani 12000 Thailand

Abstract

We propose a family of quadratic cooperating snakes, which are able to split, merge, and disappear as necessary, for segmentation of roads in satellite imagery. We combine the multiple snake framework with a preprocessing method based on oriented filtering, Canny edge detection, and Gradient Vector Flow (GVF). We evaluate the performance of the method in terms of precision and recall in comparison to ground truth data. The family of cooperating snakes consistently outperforms a single snake in a variety of road extraction tasks.

Key words: Active contours, segmentation, road extraction

1 Introduction

The geographic information system industry would benefit from flexible automated systems capable of extracting linear structures and regions of interest from remote sensing imagery. In particular, *automated road extraction* would boost the productivity of technicians enormously. This is because road networks are among the most important landmarks for mapping, and manual marking and extraction of road networks is an extremely slow and laborious process.

1.1 Related work

Towards the ultimate goal of fully automated road extraction, there has been a great deal of progress in the computer vision and image processing communities on partially automating the process. For example, Geman and Jedynak [1] proposed statistical modeling of the responses of simple nonlinear oriented ridge filters to track a given road from a seed point and direction. The method is extremely accurate, even on extremely difficult imagery.

In fully automatic road extraction, we are required to detect all of the roads of a particular range of widths in a given region of an input image. The typical approach to solving this problem combines a local neighborhood analysis step that generates feature points or calculates local likelihoods, followed by im-

⋆ This is a revised and extended version of a paper presented in ACCV 2007.

* Corresponding author. Tel: +66 2 524 5712 Fax: +66 2 524 5721.
Email addresses: mdailey@ait.ac.th (Matthew N. Dailey),
smakhanov@siit.tu.ac.th (Stanislav Makhanov), marikhu@yahoo.com (Ramesh Marikhu).

position of global constraints to link possible road points and eliminate false positives by minimizing a global cost function. Typical global cost minimization techniques include dynamic programming [2,3], active contours or snakes [4], and Markov random fields [5,6].

After 30 years of research on road extraction in the computer vision and image processing communities (see [7] for a review), there is still no system attaining the speed, robustness, and level of automation necessary for practical application on arbitrary imagery. There are very good methods for tracking single roads (e.g. [1]), but it is very difficult to reliably extract complete road networks in the presence of variability in shape, radiometry, connectivity, and geometry.

Among the most promising techniques for extraction of complex objects like roads are *active contours* or *snakes*, originally introduced by Kass et al. [8]. Since the seminal work of Kass and colleagues, techniques based on active contours have been applied to many object extraction tasks [9].

Despite their popularity, the classical parametric snake model and its variations have several major drawbacks. Chief among them is the lack of topological flexibility. When there are several objects in the image to capture, the model requires multiple snakes which have to be manually initialized to be close to the contour of each object. The initialization can be done, at best, semi-automatically, which is often time-consuming and prone to misplacement. The number of snakes is usually fixed; they cannot merge, split, or disappear. Besides this topological inflexibility, individual snakes can intersect themselves and separate snakes can collide with each other. This is due to the inability of traditional snakes to repel parts of other snakes and repair self intersections

and loops. Remedying these problems requires geometric constraints to ensure that multiple snakes do not intersect, and these geometric constraints are difficult to implement. The problem is further exacerbated when nested snakes are initialized inside one another.

To deal with these topological issues, Wong et al. [10] introduced an adjustable "blow force" to detect convex and concave shapes and to avoid self intersections. Ivins and Porrill [11] introduced a "repulsion force" discouraging contours from intersecting themselves. However, neither of these techniques deal with multiple snakes or topology changes. Samadani [12] was perhaps the first to break an active contour into several pieces, using heuristic techniques based on "energy of deformation." Durkovich et al. [13] presented a heuristic rule to split a snake into several contours whenever two parts of a snake approach each other.

Ngoi and Jia [14] applied a positive/negative contour scheme to prevent self looping and to allow a splitting into multiple contours. A "positive" active contour is initialized as a point inside the object then expelled towards the object's boundary by negative charges enclosed by the contour. This outward deformation is constrained by positive charges outside the boundary. After each iteration, the contour points are subjected to a check for self-intersections. A self-intersection is detected if the minimum distance between two non-adjacent control points is less than three pixels. When a self-intersection is detected, a positive contour can either split into two positive contours (in the case of detection of another adjacent object) or a positive and a negative contour (in the case of detection of an internal region of the object).

In order to determine which part of a contour should be split to form two

contours, Choi et al. [15] classify the segments of a snake into "contour" and "non-contour" segments, calculating the surrounding image forces along a segment. If the surrounding image forces of a point are smaller than a threshold, it is deemed a non-contour point; otherwise, it is deemed a contour point. A sequence of contour points or non-contour points forms a contour or non-contour segment. Critical points are defined as the end points of a contour segment adjacent to non-contour segments. The method decides when to split or merge contours by evaluating the distance between the critical points.

Delingnette and Montagnat [16] proposed a new topology operator for automatically creating or merging active contours.

Evaluating these existing approaches for merging and splitting snakes, Ji and Yan [17] write that

> these approaches are high in computational cost since they require checking the potential self looping/connectivity change at every iteration ... Also, these approaches suggest that the moving speed of all snake points should be equivalent, therefore being unable to deal with more complex objects (e.g. long tube-like shapes) due to the nature of their test criteria (e.g. one based on the minimum distance between two non adjacent control points (p. 149)).

The authors overcome these limitations with a very complicated but apparently robust merging algorithm which employs polygon analysis as well as geometrical analysis of intersection points of colliding snakes. The algorithm involves analysis of many cases and requires verification of many geometric conditions associated with relative positions of points.

It should be noted that the heuristic techniques previously described attempt to *prevent* self-looping. However, it is possible to make use of self-loops by splitting useful loops into separate contours and eliminating useless loops. The "T-snakes" technique proposed by McInerney and Terzopoulos [18] and later improvements like the "dual T-snakes" technique [19] are based on iterative re-parameterization of the original contour. They are able to make the use of self loops, but the approach allows only "rigid" deformations limited by a superimposed "simplicial grid."

Rochery et al. have recently proposed a parametric model for *higher-order active contours*, in particular *quadratic snakes*, for extraction of linear structures like roads [20]. The idea is to use a quadratic formulation of the contour's geometric energy to encourage anti-parallel tangents on opposite sides of a road and parallel tangents along the same side of a road. These priors increase the final contour's robustness to partial occlusions, decrease the likelihood of false detections in regions not shaped like roads, and help to prevent self-looping, since different segments of a contour with anti-parallel tangents repel each other in the absence of image forces.

Finally, to address the topological flexibility problem with traditional active contours, Caselles et al. [21] and Malladi et al. [22] independently introduced "geometric active contour models." These models are based on curve evolution theory and level set methods [23]. Rochery et al. [20] have also proposed a level set method for their quadratic snakes. Level set methods for snakes introduce a higher dimensional hypersurface in which the snake is embedded as the zero level set of the hyper surface. The geometric approach has two advantages over traditional parametric representations. First, the curve can automatically break or merge as the hypersurface evolves. Second, since the hypersurface is

represented as a mathematical function, it admits straightforward and efficient numerical adaptation schemes.

However, geometric snake models have several inherent drawbacks compared to parametric models. First, the level set representation makes it difficult, if not impossible, to impose arbitrary geometric or topological constraints on the evolving contour via the higher dimensional hypersurface [18]. Second, they do not readily admit specification of a user-defined external force. Finally, the geometric active contour models may generate shapes having inconsistent topology with respect to the actual object, when applied to noisy images characterized by large boundary gaps [24]. Rochery et al.'s system [20] requires extensive optimization to achieve reasonable run times.

Li et al. [25], in reference to the problem of topological adaptation, write "in light of the ... inherent weaknesses of geometric active contour models, it is worthwhile to seek solutions within the parametric model realm."

1.2 Our approach

The *quadratic multiple snake model* developed in this paper presents a compromise between geometric snakes' ability to split and merge easily and parametric snakes' flexibility to incorporate arbitrary constraints. We use quadratic constraints [20] both to avoid self-intersections and loops and as a means to encourage capture of thin elongated objects such as roads, rivers, canal systems, pipes, and vascular systems. We develop efficient split and merge algorithms employing straightforward conditions on the closeness of non-adjacent contour points. In the model, separate snakes can repel each other but are still capable

of approaching an object from opposite sides. The split and merge algorithms make it possible to extract highly complex networks of roads and other linear structures. The model thus provides the topological adaptability of geometric models without sacrificing the simplicity, efficiency, or flexibility of traditional parametric models.

The ability of our cooperating snakes to split, merge and disappear combined with their self-repelling feature makes it possible to easily segment several objects without knowing the number of objects in advance and without initializing the snake close to the desired contour. As an introductory example, Figure 1 shows how our snakes can segment a "broken bar" consisting of two pieces. Figure 1(a) shows an initial configuration in which the snake is relatively far from the object of interest and defined by only a few points. Figure 1(b–c) shows two successful iterations of the snake evolution algorithm. Note that the "peninsula" (highlighted by the window in Figure 1(b)) splits from the snake then disappears in the next iteration. In fact, the snakes' ability to split and disappear also accelerates convergence to a minimum. Figure 1(d) shows a split at a point with high curvature, and Figure 1(e) shows the final configuration at convergence.

In addition to the multiple snake model, to accelerate convergence to a solution, we introduce an improved external force combining oriented filtering with Canny edge detection and Xu and Prince's Gradient Vector Flow (GVF) [26]. The modified GVF field created using the proposed method is very effective at encouraging the quadratic snake to snap to the boundaries of linear structures.

A second introductory example, illustrated in Figure 2, shows a horseshoe

Fig. 1. A quadratic snake splitting and converging around two island-like objects. (a) Initial configuration. (b–c) An arm (highlighted by the square) develops, splits off, and is deleted. (d) Configuration just before a split. (e) Final configuration.

shape used to verify the ability of the snake to converge into to deep concavities and to ignore noise. The image is distorted by a grid of curves that would distract the snake from the object were it not for the GVF force encouraging the snake to snap to the boundaries of road-like objects. The combination of oriented filtering, Canny edge detection, and the GVF external force makes it possible for the snake to ignore the obstructing grid entirely and attach itself to the object of interest despite the large gradients located far from the desired boundary.

These simple examples demonstrate that the variational formulation governed by the quadratic energy functional not only allows the snake to split and merge without creating loops within one snake or intersections between different

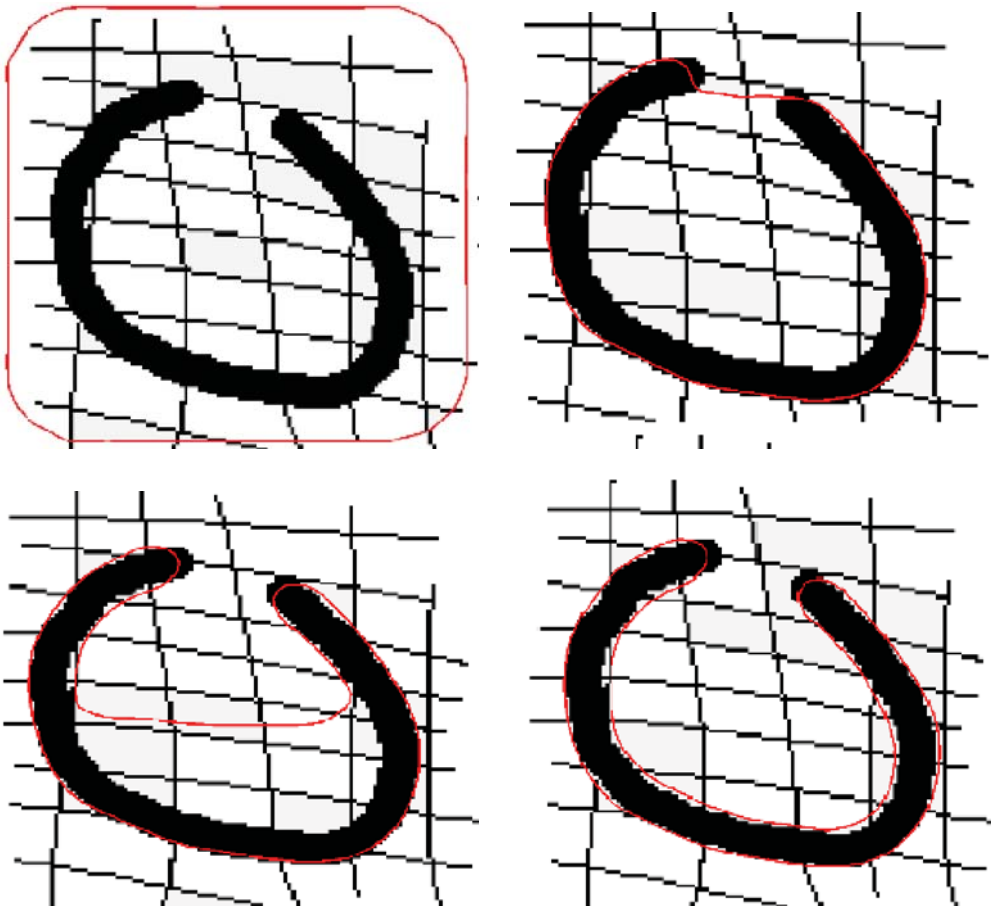


Fig. 2. A quadratic snake converging into a deep concavity despite the presence of noise.

snakes, but also makes it simple to incorporate the GVF, allowing the snake to capture the shape of complex objects with deep, narrow concavities.

In this paper, we demonstrate the effectiveness of the family of snakes and the modified GVF field in a series of experiments with real satellite images, and we provide precision and recall measurements in comparison with ground truth data. The results are an encouraging step towards the ultimate goal of fully automated road extraction from satellite imagery.

As a last contribution, we have developed a complete GUI environment for satellite image manipulation and quadratic snake evolution, based on the Mat-

lab platform. The system is freely available as open source software from `http://www.cs.ait.ac.th/~mdailey/snakes`.

2 Method

2.1 Quadratic snake model

This section provides a brief overview of the quadratic snake proposed by Rochery et al. [20]. An *active contour* or *snake* is parametrically defined as

$$\gamma(p) = \begin{bmatrix} x(p) & y(p) \end{bmatrix}^T, \tag{1}$$

where p is the curvilinear abscissa of the contour and the vector $\begin{bmatrix} x(p) & y(p) \end{bmatrix}^T$ defines the Cartesian coordinates of the point $\gamma(p)$.

The energy functional is given by

$$E_s(\gamma) = E_g(\gamma) + \lambda E_i(\gamma), \tag{2}$$

where $E_g(\gamma)$ is the *geometric energy* and $E_i(\gamma)$ is the *image energy* of the contour γ. λ is a free parameter determining the relative importance of the two terms.

To apply the method to road extraction, we define the geometric energy functional to be

$$E_g(\gamma) = L(\gamma) + \alpha A(\gamma) - \frac{\beta}{2} \iint \mathbf{t}(p) \cdot \mathbf{t}(p') \; \Psi\left(\|\gamma(p) - \gamma(p')\|\right) \; dp \; dp', \tag{3}$$

where $L(\gamma)$ is the Euclidean length of γ, $A(\gamma)$ is the area enclosed by γ, $\mathbf{t}(p)$ is the unit-length tangent to γ at point p, and $\Psi(z)$, given the distance z between

two points on the contour, is used to weight the interaction between those two points (see below). α and β are constants weighting the relative importance of the terms. Clearly, for positive β, $E_g(\gamma)$ is minimized by contours with short length and parallel tangents. If α is positive, contours with small enclosed area are favored; if it is negative, contours with large enclosed area are favored.

The interaction function $\Psi(\cdot)$ is a smooth function expressing the radius of the region in which parallel tangents should be encouraged and anti-parallel tangents should be discouraged:

$$\Psi(z) = \begin{cases} 1 & \text{if } z < d - \epsilon, \\ 0 & \text{if } z > d + \epsilon, \\ \frac{1}{2}\left(1 - \frac{z-d}{\epsilon} - \frac{1}{\pi}\sin\pi\frac{z-d}{\epsilon}\right) & \text{otherwise.} \end{cases} \tag{4}$$

In application to road extraction, d is the expected road width and ϵ expresses the expected variability in road width. During snake evolution, weighting by $\Psi(z)$ in Equation 3 discourages two points with anti-parallel tangents (the opposite sides of a putative road) from coming closer than distance d from each other.

The image energy functional $E_i(\gamma)$ is defined as

$$\begin{aligned} E_i(\gamma) = & \int \mathbf{n}(p) \cdot \nabla I(\gamma(p))\ dp \\ & - \iint \mathbf{t}(p) \cdot \mathbf{t}(p')\ \nabla I(\gamma(p)) \cdot \nabla I(\gamma(p'))\ \Psi(\|\gamma(p) - \gamma(p')\|)\ dp\ dp', \end{aligned} \tag{5}$$

where $I : \Omega \to [0, 255]$ is an image and $\nabla I(\gamma(p))$ is the gradient of I evaluated at $\gamma(p)$.

The first (linear) term favors anti-parallel normal and gradient vectors, encouraging counterclockwise snakes to shrink around or clockwise snakes to expand

to enclose dark regions surrounded by light roads.[1] The second (quadratic) term favors nearby point pairs with two different configurations, one with parallel tangents and parallel gradients and the other with anti-parallel tangents and anti-parallel gradients.

After solving the Euler equations for minimizing the energy functional $E_s(\gamma)$ (Equation 2), ignoring flow in the direction tangent to γ, we can obtain the update equation

$$\begin{aligned}\mathbf{n}(p) \cdot \frac{\delta E_s}{\delta \gamma}(p) = &-\kappa(p) - \alpha - \lambda \|\nabla I(\gamma(p))\|^2 \\ &+ \beta \int \mathbf{r}\left(\gamma(p), \gamma(p')\right) \cdot \mathbf{n}(p') \;\; \Psi'\left(\|\gamma(p) - \gamma(p')\|\right) dp' \\ + 2\lambda \int \mathbf{r}\left(\gamma(p), \gamma(p')\right) \cdot \mathbf{n}(p') \;\; &(\nabla I(\gamma(p)) \cdot \nabla I(\gamma(p'))) \;\; \Psi'\left(\|\gamma(p) - \gamma(p')\|\right) dp' \\ &+ 2\lambda \int \nabla I(\gamma(p')) \cdot (\nabla \nabla I(\gamma(p)) \mathbf{n}(p')) \;\; \Psi(\|\gamma(p) - \gamma(p')\|) \; dp'. \quad (6)\end{aligned}$$

In the equation, $\kappa(p)$ is the curvature of γ at $\gamma(p)$.

$$\mathbf{r}\left(\gamma(p), \gamma(p')\right) = \frac{\gamma(p) - \gamma(p')}{\|\gamma(p) - \gamma(p')\|}$$

is the unit vector pointing from point $\gamma(p)$ towards $\gamma(p')$. $\nabla \nabla I(\gamma(p))$ is the 2×2 Hessian of I evaluated at $\gamma(p)$. α, β, and λ are free parameters that need to be determined experimentally. d and ϵ are specified a priori according to the desired road width.

2.2 GVF external force

The term $\alpha A(\gamma)$ in Equation 3 leads to the constant term $-\alpha$ in Equation 6. This "balloon force" [9] increases the capture region around objects, but

[1] For dark roads on a light background, we simply negate the terms involving the image. In the rest of the paper, we assume light roads on dark background.

its effect is uniform throughout the image. This makes it difficult to specify a value for α that is appropriate in all regions of the image.

Xu and Prince [26,27] have proposed to use, rather than a global balloon force, a smooth, diffuse gradient field as a local external force with the traditional linear snake. They find that this technique, Gradient Vector Flow (GVF), improves the traditional snake's convergence to a minimum energy configuration.

We propose the use of GVF with quadratic road extraction snakes.

2.2.1 GVF

The GVF is a vector field

$$V^{\mathrm{GVF}}(\mathbf{x}) = \begin{bmatrix} u(\mathbf{x}) \; v(\mathbf{x}) \end{bmatrix}^T$$

minimizing the energy functional

$$\begin{aligned} E(V^{\mathrm{GVF}}) = \int_\Omega & \mu(u_x^2(\mathbf{x}) + u_y^2(\mathbf{x}) + v_x^2(\mathbf{x}) + v_y^2(\mathbf{x})) \\ & + \|\nabla \tilde{I}(\mathbf{x})\|^2 \, \|V(\mathbf{x}) - \nabla \tilde{I}(\mathbf{x})\|^2 \, d\mathbf{x}, \end{aligned} \tag{7}$$

where

$$u_x = \frac{\partial u}{\partial x}, \quad u_y = \frac{\partial u}{\partial y}, \quad v_x = \frac{\partial v}{\partial x}, \quad v_y = \frac{\partial v}{\partial y},$$

and $\tilde{I}$ is a preprocessed version of image I, typically an edge image of some kind. The first term inside the integral encourages a smooth vector field whereas the second term encourages fidelity to $\nabla \tilde{I}$. μ is a free parameter controlling the relative importance of the two terms.

We obtain $\tilde{I}$ using oriented filtering and Canny edge detection (see Figure 3). We use elongated Laplacian of Gaussian filters that emphasize road-like structures, deemphasize non-road-like structures, and, to a certain extent, fill

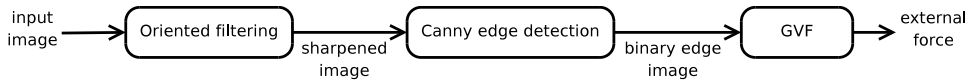


Fig. 3. Schematic of procedure to obtain the GVF external force.

in short gaps where a road has low contrast with the background. The resulting binary Canny image is ideal because it only includes information about road-like edges that have survived sharpening by the oriented filters. The GVF field on top of the sharpened edge image is ideal because it points toward the road-like edges from a long distance, and, during snake evolution, it pushes the snake in an appropriate direction. This speeds evolution and makes it easier to find suitable parameters to obtain fast convergence.

2.2.2 Oriented filtering

Using oriented filters for contour detection, contour completion, and restoration of edges corrupted by noise is a recurring idea in image processing and computer vision (see, e.g., [28–32]). The oriented filters most frequently used are 2D Gabor filters [33] and directional 2nd-derivative-of-Gaussian filters. Gabor filters are thought to be good models of the response of simple cells in primary visual cortex [34]. When paired symmetric (even) and antisymmetric (odd) oriented filter responses are combined by summing their squares, they are thought to be good models of the response of complex cells in primary visual cortex [35]. Perona and Malik [29] advocate these paired "energy filters" for their ability to detect not only step edges but also ridge edges at specific scales.

The ability of Gabor filters and 2nd-derivative-of-Gaussian filters to detect ridge edges makes them ideal for identifying roads in satellite imagery. Our oriented filtering method is the same as that of Rochery et al. [20]. We use

the linear response of even 2nd-derivative-of-Gaussian filters tuned to detect roads at particular scales. We obtain a sharpened image Q defined by

$$Q(\mathbf{x}) = \min_{\theta \in \Theta} \left\{ (\mathcal{F}_\theta * I)(\mathbf{x}) \right\}, \tag{8}$$

where $*$ denotes convolution and the kernel $\mathcal{F}_\theta$ is given by

$$\mathcal{F}_\theta = R_\theta \nabla^2 N_{\sigma_x, \sigma_y}. \tag{9}$$

N_{σ_x,σ_y} is a 2D Gaussian with variance σ_x^2 in the x direction and σ_y^2 in the y direction, R_θ is a matrix rotating N_{σ_x,σ_y} by angle θ, and ∇^2 is the 2D Laplacian. σ_x is chosen according to the desired length of the filter along the road contour whereas σ_y is chosen according to the desired width of the road to detect. We use angles

$$\Theta = \{0, \frac{\pi}{8}, ..., \frac{7\pi}{8}\}.$$

If the roads in I are darker than their surroundings, the maximum rather than the minimum convolution result is used to compute $Q(\mathbf{x})$.

An example of the convolution and minimum response selection procedure is shown in Figure 4(a–j). The filters respond well to long straight edges in the image. This has the effect of emphasizing road-like gradients, deemphasizing non-road-like gradients, and, to a certain extent, filling in short gaps where a road has low contrast with the background.

2.2.3 Obtaining the GVF field

After oriented filtering, we obtain the Canny edge image $\tilde{I}$ from the sharpened image Q. An example is shown in Figure 4(k). This is the input to the GVF relaxation procedure [26]. An example of the resulting GVF field V^{GVF} is shown in Figure 4(l). We precalculate V^{GVF} before snake evolution begins, then dur-

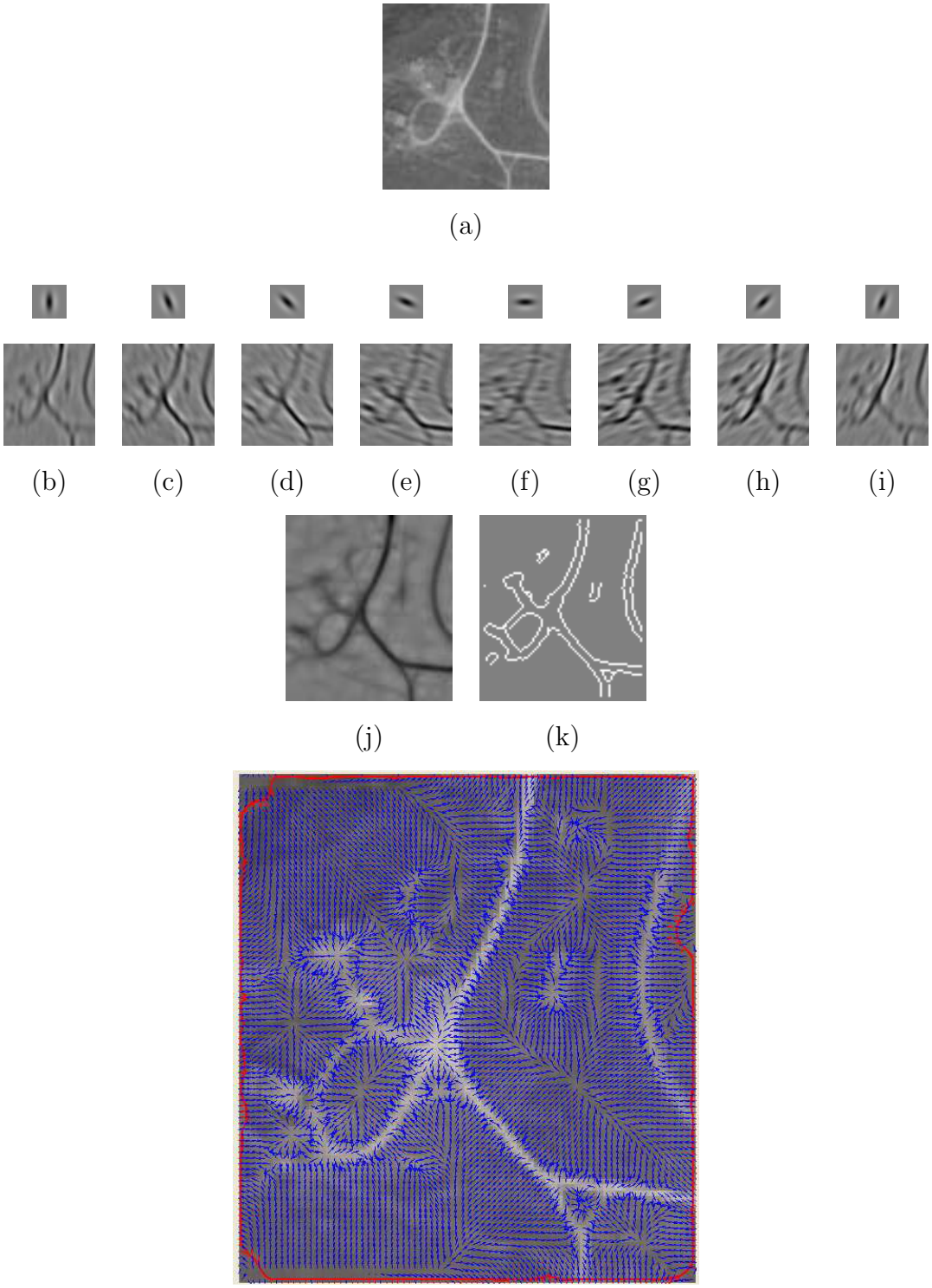


Fig. 4. Image processing for obtaining the GVF. (a) Original image. (b–i) Convolution results. (j) Sharpened image Q. (k) Canny edge image $\tilde{I}$ derived from Q. (l) GVF V^{GVF} based on $\tilde{I}$.

ing evolution, for each point $\gamma(p)$ in Equation 6, we project $V^{\text{GVF}}(\gamma(p))$ onto $\mathbf{n}(\gamma(p))$ and add the resulting force directly to the update equation. Clearly, this encourages the snake to snap to the road edge contours, where ideally $\|V^{\text{GVF}}(\gamma(p))\| = 0$.

2.3 Family of quadratic snakes

A single quadratic snake is unable to extract enclosed regions and multiple disconnected networks in an image. We address this limitation by introducing a *family* of cooperating snakes that are able to split, merge, and disappear as necessary.

In our formulation, due to the curvature term $\kappa(p)$ and the area constant α in Equation 6, specifying the points on γ in a counterclockwise direction creates a *shrinking snake* and specifying the points on γ in a clockwise direction creates a *growing snake*.

An enclosed region (loop or a grid cell) can be extracted effectively by initializing two snakes, one shrinking snake covering the whole road network and another growing snake inside the enclosed region.

2.3.1 Splitting a snake

We split a snake into two snakes whenever two of its arms are squeezed too close together, i.e. when the distance between two snake points is less than d^{split} and those two points are at least k snake points from each other in both directions of traversal around the contour. d^{split} should be less than 2η, where η is the maximum step size.

2.3.2 Merging two snakes

The merging algorithm selects points having high curvature and merges two snakes when 1) two selected points are closer than a prescribed minimal merging distance d^{merge}, 2) the traversal direction (clockwise or counterclockwise) of the two snakes is the same, and 3) the tangents at the two high curvature points are nearly antiparallel. High curvature points are those with $\kappa_\gamma(p) > 0.6\kappa_\gamma^{\max}$, where $\kappa_\gamma^{\max}$ is the maximum curvature for any point on γ. When these conditions are satisfied, the two snakes are combined into a single snake by deleting the high curvature points and merging at the holes.

Limiting the merge decision to high curvature points ensures that merging only occurs if two snakes have semi-circular tips of their arms facing each other. It might seem that merging at low curvature points should also be permitted. However, as already explained, snakes normally repel each other due to the quadratic term in the internal energy (Equation 3). Consequently, low curvature segments can approach each other when high-gradient features allow the external energy to overcome the geometric energy. When this occurs for low curvature segments, the two snakes are most likely positioned on different sides of a road and merging should not be allowed. There are several other (rare) cases when snakes face each other at low curvature parts. However they should not be merged in those cases either.

Considering only the high curvature points also saves computational costs. In particular, the merging procedure requires computation of the angle between tangents only for the selected points. The number of those points usually does not exceed 10% of the total number of points.

The conditions that the traversal direction of two snakes should be the same

and that the tangents at the two high curvature points should be antiparallel reflect the fact that in our system, nested snakes form a tree structure. We initialize all the snakes at the first level with the same direction of traversal. The second level has the opposite direction of traversal and so on. When two snakes from the same level merge, we assign the resulting snake the same direction. Snakes from two consecutive levels do not merge. Growing and shrinking behavior is controlled by the area constant (α) and the weight on the geometric energy (β).

2.3.3 Deleting a snake

A snake γ is deleted if it has perimeter less than L^{delete}.

2.4 Experimental design

We present four experiments aimed at evaluating the effectiveness of the proposed cooperating snake model for road extraction. In Experiment 1, we explore the ability of the model to extract simple tree-structured road networks that do not require multiple snakes. Experiment 2 moves to more complex tree-structured networks with distracting structures. These networks require contours able to split, merge, and disappear in order to ignore noise. Experiment 3 tests the model's ability to capture disconnected networks. In Experiment 4, we evaluate the model's ability to extract networks with cycles with the help of user initialization. Finally, in Experiment 5, we determine the effect of the GVF external force on the model's evolution.

In each experimental condition, we hand-tune the free parameters to achieve

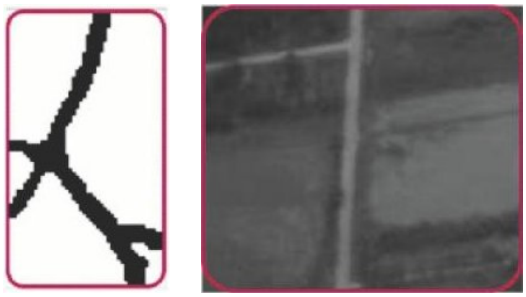
good results. We terminate contour evolution whenever the energy $E_s(\gamma)$ fails to decrease for some number of iterations.

As a baseline for comparison, we use the parametric representation of the single quadratic snake model proposed by Rochery et al. [20]. To evaluate the results, we hand-digitized ground truth images and used them to calculate precision (the proportion of detected pixels that are road pixels according to the ground truth), recall (the proportion of road pixels that are detected), and F_1 (the harmonic mean of precision and recall) for each solution.

3 Results

3.1 Experiment 1: Simple tree-structured networks

We ran a single quadratic snake on the synthetic image shown in Figure 5(a) and the real image shown in Figure 5(b). Our best extraction results are shown in Figure 5(c) and Figure 5(d). The precision and recall results are shown in Table 1. Experiment 1 demonstrates that a single quadratic snake is well suited to simple tree-structured road networks, when the contour does not need to change topology during evolution.



(a) (b)

(c) (d)

Fig. 5. Experiment 1 extraction results. (a) Original synthetic image. (b) Original real image. (c) Road network extracted from the image of (a). (d) Road network extracted from the image of (b).

Table 1

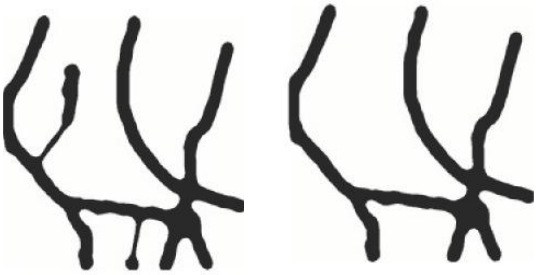
Experiment 1 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Synthetic simple tree	5(c)	480	0.937	0.860	0.897
Real simple tree	5(d)	308	0.948	0.850	0.896

3.2 Experiment 2: Complex tree-structured networks

Beginning with the image shown in Figure 6(a) containing a relatively complex tree-structured road network with distracting road-like structures, we ran a single quadratic snake and the cooperating multiple snake model. The best extraction results for the two models are shown in Figure 6(b) and Figure 6(c), respectively. Table 2 shows the detailed precision and recall results. The cooperating multiple snake model is better able to handle the distracting structures mainly because the evolving snake splits to enclose each road-like structure then the small isolated contours are deleted.

(a)



(b) (c)

Fig. 6. Experiment 2 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with a single snake. (c) Road network extracted from the image of (a) with cooperating snakes that can split, merge, and disappear.

Table 2

Experiment 2 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Complex tree, single snake	6(b)	473	0.709	0.907	0.796
Complex tree, cooperating snakes	6(c)	450	0.801	0.912	0.853

3.3 Experiment 3: Disconnected networks

Beginning with the image shown in Figure 7(a) containing multiple disconnected road networks, we ran a single quadratic snake and the cooperating multiple snake model. The best extraction results for the two models are shown in Figure 7(b) and Figure 7(c), respectively. Table 3 shows the extraction performance details. The cooperating multiple snake model is able to extract the multiple separate road networks, whereas the single snake does its best to model the network with a single contour.

(a)

(b) (c)

Fig. 7. Experiment 3 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with a single snake. (c) Road network extracted from the image of (a) with cooperating snakes that can split, merge, and disappear.

Table 3

Experiment 3 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Disconnected network, single snake	7(b)	219	0.682	0.867	0.764
Disconnected network, cooperating snakes	7(c)	163	0.858	0.873	0.865

3.4 Experiment 4: Networks with cycles

The multiple cooperating snake model cannot extract road networks with loops in a fully automatic fashion, but in our implementation it is simple for the user to manually initialize a separate contour inside each loop. We compared the ability of the single snake, the multiple cooperating snake model with a single initial contour, and the multiple cooperating snake model with user-defined initialization to extract road networks from the images containing loops in Figure 8(a–b). Our best results are shown in Figure 8(c–f). The multiple cooperating snake model obtains excellent results with user-specified initial conditions.

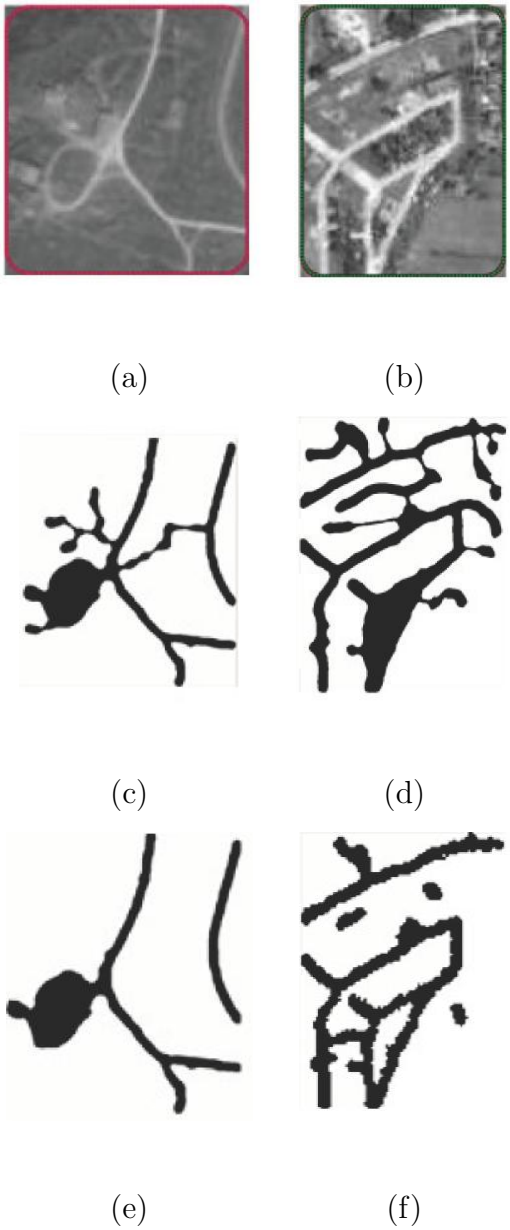


Fig. 8. Experiment 4 extraction results. (a–b) Original images. (c–d) Road networks extracted from images (a–b) with a single snake. (e) Road network extracted from image (a) with cooperating multiple snakes. (f) Road networks extracted from image (b) with cooperating multiple user-defined snakes.

Table 4

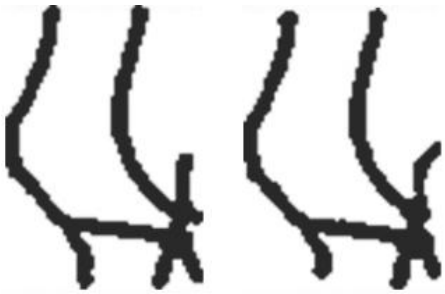
Experiment 4 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Simple loop, single snake	8(c)	182	0.547	0.911	0.684
Complex loop, single snake	8(d)	519	0.530	0.907	0.669
Simple loop, cooperating snakes	8(e)	182	0.6344	0.9833	0.771
Complex loop, user-initialized snakes	8(f)	166	0.768	0.912	0.834

3.5 Experiment 5: GVF external force

Our GVF external force is based on an edge map acquired through oriented filtering, and Canny edge detection. We precomputed the GVF vector field then ran the multiple quadratic snake model on the image shown in Figure 9(a), with and without the GVF force. Our best results for the two experimental conditions are shown in Figure 9(b) and Figure 9(c). The precision and recall results are shown in Table 5. The snakes converge faster in the GVF condition with a slight decrease in precision and recall, although visually the two extracted road networks are of comparable quality.

(a)



(b) (c)

Fig. 9. Experiment 5 extraction results. (a) Original image. (b) Road network extracted from the image of (a) with the multiple snake model and no GVF external force. (c) Road network extracted from the image of (a) with the multiple snake model and the GVF external force.

Table 5

Experiment 5 extraction performance.

Condition	Figure no.	No. of iterations	Precision	Recall	F_1
Cooperating snakes without GVF	9(b)	650	0.828	0.938	0.880
Cooperating snakes with GVF	9(c)	430	0.802	0.909	0.852

4 Discussion

Experiments 1–5 demonstrate the effectiveness of the proposed multiple cooperating snake model on a variety of road networks. In Experiment 1, we found that a single quadratic snake was sufficient to extract simple tree-structured road networks in synthetic and real imagery. Experiment 2 demonstrated that cooperating snakes converge faster and more accurately than a single snake when the image contains a more complex tree-structured road network with distracting noise. In Experiment 3, we found that the cooperating snake model is effective for extracting disconnected road networks, and in Experiment 4, we found that it is also appropriate for complex networks with cycles, if user-aided initialization is used. Finally, Experiment 5 demonstrated that incorporating an external image force derived from oriented filtering, Canny edge detection, and the GVF provides faster convergence to a minimum-energy configuration.

Our empirical study shows that quadratic snakes avoid self-intersections by incorporating into the energy functional the constraint that contour segments with anti-parallel tangents should repel each other. They are nevertheless still able to enter long and narrow concavities and to approach each other close enough to extract road networks in satellite images.

Our parametric specification of the family of cooperating snakes is simple and efficient compared to level set methods. If a family of snakes is represented by N discrete points, a naive implementation of the evolution requires $O(N^2)$ time per iteration, since Equation 6 must be applied to each of the N points and it involves an integral over all N points. Likewise, naive implementation of the split and merge algorithms developed in Section 2.3 consider at most each

pair of points, so assuming at most a constant number of splits and merges per iteration, the complexity remains $O(N^2)$. However, it is possible to reduce the runtime to $O(N)$. The contour update and split/merge algorithms only perform comparisons with other contour points within a fixed local region, so it is possible to preindex the contour points in the image domain such that at most a constant number of other points are considered for each point on the contour.

One limitation of our approach is that we require manual initialization of the contours to obtain good results when the road network contains loops. The level set method's main strength is its ability to handle such loops without any special treatment. However, our method could be extended to handle this case, if we added autodetection of "holes" in a converged family of contours or if we initialized with many small growing snakes inside a shrinking outer snake, either randomly or in a grid pattern.

A more serious limitation of our approach is the need to determine free parameters such as α, β, and λ empirically. In constrained applications such as road extraction, it should be possible to develop a database of useful parameter settings for particular image resolutions and road network types. But sensitivity to parameter settings is the Achilles' heel of all active contour models; unless this problem is solved, the technique's applicability to real-world GIS problems will be limited.

5 Conclusion

The proposed method is an implementation of multiple active contours in a variational framework based on a quadratic energy functional. Our model performs better than conventional snakes and single parametric quadratic snakes on road extraction tasks. The combination of oriented LoG filters, Canny edge detection, and the GVF provides effective preprocessing of noisy and distorted images and an appropriate external force for the quadratic snake's energy functional. The multiple snake configuration that minimizes the total energy functional accurately snaps to the boundaries of objects. The energy functional's quadratic terms, which encourage parallel tangents and discourage anti-parallel tangents, and its sigmoid interaction function, are designed to extract curvilinear ribbon-like structures such as roads, canals, and pipelines from digital images. We have shown that the scheme is effective at extracting road networks from a series of satellite images, but some calibration of the free parameters is required to achieve good results.

In future research we plan to focus on automatic initialization of contours to handle networks with cycles and reducing the method's run time to the point that it is practical for application in GIS applications. It may also be possible to develop higher order active contour models for other important segmentation problems involving extraction of other shapes such as ellipses and polygons.

6 Acknowledgments

This work was supported by Thailand Research Fund grant MRG4780209 to MND. RM was supported by a graduate fellowship from the Nepal High Level Commission for Information Technology. Ran Zask assisted with software development. We are grateful to Kiyoshi Honda for helpful comments on this research.

References

[1] D. Geman, B. Jedynak, An active testing model for tracking roads in satellite images, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (1) (1996) 1–14.

[2] M. Fischler, J. Tenenbaum, H. Wolf, Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique, Computer Graphics and Image Processing 15 (1981) 201–223.

[3] M. Barzohar, D. Cooper, Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (7) (1996) 707–721.

[4] P. Fua, Y. Leclerc, Model driven edge detection, Machine Vision and Applications 3 (1990) 45–56.

[5] C. Regazzoni, G. Foresti, S. Serpico, An adaptive probabilistic model for straight edge-extraction within a multilevel MRF framework, in: International Geoscience and Remote Sensing Symposium, 1995, pp. 458–460.

[6] F. Tupin, H. Maître, J.-F. Mangin, J.-M. Nicolas, E. Pecersky, Detection of linear features in SAR images: Application to road network extraction, IEEE

Transactions on Geoscience and Remote Sensing 36 (2) (1998) 434–453.

[7] M.-F. Auclair-Fortier, D. Ziou, C. Armenakis, S. Wang, Survey of work on road extraction in aerial and satellite images, Tech. Rep. 247, Departement de mathematiques et d'informatique, Universitede Sherbrooke (2000).

[8] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, International Journal of Computer Vision 1 (4) (1987) 321–331.

[9] L. D. Cohen, I. Cohen, Finite-element methods for active contour models and balloons for 2-d and 3-d images, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (11) (1993) 131–147.

[10] Y. Wong, P. Yuen, C. Tong, Segmented snake for contour detection, Pattern Recognition 31 (11) (1998) 1669–1679.

[11] J. Ivins, J. Porrill, Active region models for segmenting regions and colors, Image and Vision Computing 13 (5) (1995) 431–438.

[12] R. Samadani, Changes in connectivity in active contour models, in: Proceedings of the Workshop on Vision Motion, 1989, pp. 337–343.

[13] R. Durkovich, K. Kaneda, H. Yamashita, Dynamic contour: A texture approach and contour operations, Visual Computing 11 (1995) 227–289.

[14] K. Ngoi, J. Jia, An active contour model for color region extraction in natural scenes, Image and Vision Computing 17 (3) (1999) 955–966.

[15] W. Choi, K. Lam, W. Siu, An adaptive contour model for highly irregular boundaries, Pattern Recognition 34 (2) (2001) 323–331.

[16] H. Delingnette, J. Montagnat, New algorithm for controlling active contour shape and topology, in: Sixth European Conference on Computer Vision (ECCV), Vol. 2, 2000, pp. 381–395.

[17] L. Ji, H. Yan, Robust topology-adaptive snakes for image segmentation, Image and Vision Computing 20 (2002) 147–164.

[18] T. McInerney, D. Terzopoulos, T-snakes: Topology adaptive snakes, Medical Image Analysis 4 (2) (2000) 73–91.

[19] G. Giraldi, E. Strauss, A. Oliveira, Dual-T-Snakes model for medical imaging segmentation, Pattern Recognition Letters 24 (2003) 993–1003.

[20] M. Rochery, I. H. Jermyn, J. Zerubia, Higher order active contours, International Journal of Computer Vision 69 (1) (2006) 27–42.

[21] V. Caselles, F. Catte, T. Coll, F. Dibos, A geometric model for active contours, Numerische Mathematik 66 (1993) 1–31.

[22] R. Malladi, J. Sethian, B. Vemuri, Shape modeling with front propagation: A level set approach, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (1995) 158–175.

[23] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.

[24] C. Xu, D. Pham, J. Prince, Image segmentation using deformable models, in: Handbook of Medical Imaging Volume 2: Medical Image Processing and Analysis, SPIE Press, 2000, pp. 129–174.

[25] C. Li, J. Liu, M. Fox, Segmentation of external force field for automatic initialization and splitting of snakes (2005).

[26] C. Xu, J. L. Prince, Gradient Vector Flow: A new external force for snakes, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1997, pp. 66–71.

[27] C. Xu, J. Prince, Snakes, shapes, and gradient vector flow, IEEE Transactions on Image Processing 7 (3) (1998) 359–369.

[28] H. Knutsson, R. Wilson, G. Granlund, Anisotropic nonstationary image estimation and its applications: Part I — restoration of noisy images, IEEE Transactions on Communications COM-31 (3) (1983) 388–397.

[29] P. Perona, J. Malik, Detecting and localizing edges composed of steps, peaks, and roofs, in: International Conference on Computer Vision, 1990, pp. 52–57.

[30] W. Freeman, E. Adelson, The design and use of steerable filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (9) (1991) 891–906.

[31] C. Steger, An unbiased detector of curvilinear structures, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (2) (1998) 113–125.

[32] S. Konishi, A. Yuille, J. Coughlan, S. Zhu, Statistical edge detection: Learning and evaluating edge cues, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (1) (2003) 57–74.

[33] J. G. Daugman, Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters, Journal of the Optical Society of America A 2 (1985) 1160–1169.

[34] J. P. Jones, L. A. Palmer, An evaluation of the two-dimensional Gabor filter model of receptive fields in cat striate cortex, Journal of Neurophysiology 58 (6) (1987) 1233–1258.

[35] F. Heitger, L. Rosenthaler, R. von der Heydt, E. Peterhans, O. Kübler, Simulation of neural contour mechanisms: From simple to end-stopped cells, Vision Research 32 (5) (1992) 963–981.