Surveillance in a Smart Home Environment

Ryan Stewart Patrick
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Sciences Commons

Repository Citation
Patrick, Ryan Stewart, "Surveillance in a Smart Home Environment" (2010). Browse all Theses and Dissertations. 352.
https://corescholar.libraries.wright.edu/etd_all/352

This Thesis is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
Surveillance in a Smart Home Environment

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Science

By

RYAN STEWART PATRICK
B.S., The College of New Jersey, 2008

2010
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Ryan Patrick ENTITLED *Surveillance in a Smart Home Environment* BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

Nikolaos Bourbakis, Ph. D.
Thesis Director

Thomas Sudkamp, Ph. D.
Department Chair

Committee on Final Examination

Nikolaos Bourbakis, Ph. D.

Soon Chung, Ph. D.

Yong Pei, Ph. D.

John A. Bantle, Ph.D.
Interim Dean, School of Graduate Studies
A system for assisting the elderly in maintaining independent living is currently being designed. When mature, it is expected that this system will have the ability to track objects that a resident may lose periodically, detect falls within the home, and alert family members or health care professionals to abnormal behaviors.

This thesis addresses the early stages of this system’s development. It presents a survey of the work that has previously been completed in the area of surveillance within a home environment, information on the physical characteristics of the system that is being designed, early results related to this system, and guidance on the future work that will have to be completed.
3.3 SURF Point Correlation ......................................................... 28
3.4 Good Features to Track ....................................................... 30
3.5 Indirect CAMShift Tracking .................................................. 30
3.6 CAMShift in a Different Color space ...................................... 33
3.7 Kalman Filter ................................................................. 34
3.8 Particle Filter ............................................................... 35
3.9 Conclusions ................................................................. 35

4 Future Work ................................................................. 38
4.1 Data Processing .............................................................. 38
4.2 Information from Multiple Cameras ..................................... 41
4.3 Sudden Lighting Changes .................................................. 41

References ................................................................. 42
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Views from the ICDSC Smart Homes Data Set</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>System Elements</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Shadow in the Foreground</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Frame from ICDSC 2009 Smart Homes Data Set</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Effects of Motion Blur</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Mug in the “Background”</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>Magazine in the “Background”</td>
<td>19</td>
</tr>
<tr>
<td>2.6</td>
<td>Views from the TUM Kitchen Data Set</td>
<td>20</td>
</tr>
<tr>
<td>2.7</td>
<td>Creation of a Background Image</td>
<td>22</td>
</tr>
<tr>
<td>2.8</td>
<td>CAMShift Tracking Board</td>
<td>23</td>
</tr>
<tr>
<td>2.9</td>
<td>CAMShift Tracking Cabinet</td>
<td>24</td>
</tr>
<tr>
<td>2.10</td>
<td>CAMShift Tracking Background</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>CAMShift Tracking with Changing Scale</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Incorrect Template Matching</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Template Matching Based on All Views</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Segmentation of Skin and Cutting Board</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Determination of the Arms</td>
<td>32</td>
</tr>
<tr>
<td>3.6</td>
<td>Value - Saturation - Hue Image of Foreground</td>
<td>33</td>
</tr>
<tr>
<td>3.7</td>
<td>Particle Filter Tracking Pick-up</td>
<td>36</td>
</tr>
<tr>
<td>3.8</td>
<td>Particle Filter Tracking Movement</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Video Frame</td>
<td>39</td>
</tr>
<tr>
<td>4.2</td>
<td>Same Frame after Foreground Segmentation</td>
<td>40</td>
</tr>
</tbody>
</table>
List of Tables

1.1 Element Importance ................................................. 5
1.2 Object Locators ..................................................... 6
1.3 Evaluation of Object Locators ..................................... 6
1.4 Generic Tracking Systems .......................................... 7
1.5 Evaluation of Generic Tracking Systems ......................... 8

2.1 Single Linksys Camera Transmission Rates ....................... 12

3.1 Kalman Transition Matrix ........................................... 34
I would like to acknowledge the many people whose previous work and current assistance contributed to this thesis. Whenever I hit a dead end and had no idea which direction to go next, Dr. Nikolaos Bourbakis was there to make suggestions. When I was completely stumped by the inner workings of Logitech’s WiLife camera system, Alexandros Pantelopoulos was able to lend his experience in electrical engineering to help me understand the system’s operation. I would like to thank Alexandros Karargyris for his suggestions on how improve the image acquisition and processing that was required for this thesis. I would also like to express my gratitude to Brian Jackson for his suggestions on tracking objects more reliably.

I would like to also thank the people who provided general support for my thesis. Without the assistance of Donetta Bantle, navigating the bureaucracy of graduate school would have been much more difficult; without the camaraderie of Rob Keefer, Athanasios Tsitsoulis, Mike Mills, Victor Agbugba, Dimitrios Dakopoulos, Allan Rwabutaza, and Giuliano Manno, the hours spent in the lab would have been more monotonous; without the technical support of Matt Kijowski, the setup of our network of cameras would have been much more frustrating; and without the support of the other Computer Science and Engineering faculty, staff, and, especially, teaching assistants who helped me adjust to life at Wright State.

I would especially like to thank my family. For two years, they put up with me living far from home and starting conversations about my research with, “I tried something different, and thought I fixed the problem, but...”. Without their unconditional support, the completion of this thesis would not have been possible.
1

Survey

As part of this work, we evaluated similar systems that were designed in the last decade. We also evaluated systems that were related to our area of work. That survey [Patrick and Bourbakis 2009] is reproduced here.

In the last 10 years, research in the field of automated multiple camera surveillance has grown dramatically. [Stuart et al. 1999] began to experiment with methods for tracking objects within the view of a camera and transferring information about tracked objects from one camera to another. While [Stuart et al. 1999] only provided results on a simulation of a scene that was monitored by several, non-overlapping cameras, several ideas, such as the notion of object “trajectories”, came out of this work.

While the initial contributions of [Stuart et al. 1999] specifically addressed methods for the surveillance of traffic in outdoor environments, interest in the automation of surveillance in indoor environments grew from the prevalence of existing surveillance systems in public and private buildings. Indoor surveillance posed new challenges, and provided new benefits that were not present in outdoor surveillance. Indoor environments are generally protected from factors, such as wind and water, that outdoor surveillance equipment would need to be robust to. However, the sudden illumination changes that are not present in an outdoor environment, must be adequately dealt with indoors.

A specialization of the indoor surveillance problem is the problem of surveillance in smart homes and smart rooms. While general surveillance systems attempt to use each camera to monitor a broad area, thus limiting the number of required cameras, the goal of surveillance in smart homes and rooms is to efficiently capture details that may be important to the user. [Chen et al. 2008] and [Aghajan 2009] illustrate this point well. In [Chen et al. 2008], five cameras are used to monitor two hallways and one room. Only one pair of cameras has overlapping views, and that overlap is only provided by an open door that is not guaranteed to be persistently open.
Alternatively, [Aghajan 2009] monitors one hallway and two rooms with a total of eight cameras. Beyond the numerical difference, the systems in [Chen et al. 2008] and [Aghajan 2009], and environments they monitor, are very different. [Chen et al. 2008] appears to use a system of cameras that are mounted to the ceiling and, therefore, are located parallel to the ground. The ground plane dominates the view that each camera has and each scene is generally illuminated by artificial light. Conversely, the scene and system in [Aghajan 2009] does not appear to be as predictable. While many of the cameras appear to be mounted on the wall or ceiling and have a view of the scene that is similar to the cameras in [Chen et al. 2008], camera 5 appears to be positioned at an oblique angle. The scene also appears to be lit by a combination of natural and artificial light. To further complicate matters, both the natural and artificial light appear to be intense enough to cause parts of the scene to be washed out. In addition all of the other differences, very few of the cameras in [Aghajan 2009] have an adequate view of the ground plane. Many other planes (tables, chairs, counter tops, cabinets, and a sofa) are visible, but many of the cameras have their view of ground largely occluded. The eight camera views from [Aghajan 2009] are shown in Figure 1.1.

![Figure 1.1: Views from the ICDSC Smart Homes Data Set](image)
1.1 Object Tracking in Smart Homes

We focused our survey on video surveillance in smart homes around the central problem of monitoring the location of items that an occupant may forget the location of. While this problem has been worked on through the use of radio frequency identification (RFID) tags [Kidd et al. 1999], we looked primarily at systems that used vision to track items within a home. Due to the limited number of systems that satisfy that narrow requirement, we also looked at systems that could be extended to provide a more complete solution to this problem. That broader scope went on to include systems that used a single camera to locate objects within a smart room and systems that used multiple cameras to provide general indoor surveillance.

1.2 Methodology

We assigned values based on how much systems deviated from the ideal for each element. In the cost element, more sophisticated hardware (PTZ cameras, stereo cameras, etc.) negatively affected a system’s value. Likewise, systems whose expense increased because of large storage or processing requirements received lower values for cost. Friendliness was determined by the interface and images that were presented to the user. Presentations that highlighted important information simply were assigned higher values. Values for the range element were based on how much of a scene a system was designed to cover. Systems that were confined to areas within rooms were assigned lower values in that category than systems that provided a view of a wide area. Calibration values were assigned based on how easily a system could be made operational. Systems that required the intricate calibration of cameras or other hardware received lower values. System complexity had some relation to system cost. More expensive systems generally had more sophisticated hardware. Systems that required computational power that would be considered extraordinary to the average consumer were assigned values that were lower than systems that could be run on hardware that a consumer can be expected to already possess. Systems that could continue to work through physical problems that a home environment may present, such as jostling, received higher values for the robustness element. Systems that could be more easily reconfigured after the addition or subtraction of cameras received higher values for their scalability. Systems whose performance did not deteriorate over time, under expected circumstances, had greater values in the lifetime category. The realtime value was arrived at by how quickly a system would be expected to respond to a request. A system that would need certain conditions to be met first would not have as high a value in that category as a system that could respond immediately. Reliability was affected by how well the software could respond to changes in the physical scene or the hardware. The number that was assigned for synthesis reflected
1.3. SURVEY ELEMENTS

how well a system joined information from multiple views. Three dimensional representations would receive higher values than two dimensional representations, and two dimensional representations would receive higher values than systems with no synthesized representation.

1.3 Survey Elements

First, we proposed a number of elements that users, engineers, and software developers would be concerned with in the production, deployment, and use of a surveillance system for a smart home. Figure 1.2 defines each of the elements that were used in the evaluation and an example of how a system would be ideal with respect to each element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(E1) Cost</td>
<td>The cost of the hardware and software that is used to implement a system. The cost of the hardware would include cameras and processors. The processor could be central to a consumer-level system that could be really reliable or as part of a dedicated system. An ideal system would receive the highest value of its element, meaning the processor would be either a consumer-level processor or in a consumer-level system, or a system of more power than costs less than the calculation of its cost of the processor and a consumer-level system.</td>
</tr>
<tr>
<td>(E2) Friendliness</td>
<td>The ease with which the users could interact with the system to locate an object of interest. The user would most prefer an interface to be friendly. An ideal system would receive a value of 1 since through a friendly interface, the end user can easily view all objects. This may be done through the system view possible of an object, with the user highlighted interactively.</td>
</tr>
<tr>
<td>(E3) Range</td>
<td>The worked area that can be covered by the system. An ideal system would cover the entire area of the monitored area. Ideal users would not exist in the system.</td>
</tr>
<tr>
<td>(E4) Calibration</td>
<td>The amount of work needed to when the camera are physically moved, until the system is fully operational. The ideal system would receive a value of 0. After the camera are moved, the system would begin to image objects to be tracked and to begin tracking.</td>
</tr>
<tr>
<td>(E5) System Complexity</td>
<td>The complexity of the entire system's hardware. An ideal system would receive a value of not having more than simple must trauma.</td>
</tr>
<tr>
<td>(E6) Software Complexity</td>
<td>The ability of the system's software to adapt or change in the monitored area and possible camera failures. An ideal system would be able to support camera failures and would not be interrupted by the number of objects that the system is to support.</td>
</tr>
<tr>
<td>(E7) Robustness</td>
<td>The system's ability to deal with changes in physical factors such as wind or light. The system would have an ability to operate in the area. The ideal system would allow any object to be in the field of view at all locations.</td>
</tr>
<tr>
<td>(E8) Scalability</td>
<td>The system's ability to scale up or down for the system. An ideal system could scale seamlessly to the abilities of several cameras.</td>
</tr>
<tr>
<td>(E9) Lifetime</td>
<td>The duration of time the system could operate before the system could function. An ideal system would be able to continuously function for long periods of time. The ideal system would not be disrupted by a camera failure or issues with the camera.</td>
</tr>
<tr>
<td>(E10) Realtime</td>
<td>A system's ability to provide a user or an object an object's location at any time and system the world without the user noticing. An ideal system could support a query from a tracked object that is located anywhere on the monitored area and in view if the object is being moved at the time of the query.</td>
</tr>
<tr>
<td>(E11) Reliability</td>
<td>The ability of the system's overall system to be constantly available and ready to use. An ideal system would also be able to function in a real-time environment, meaning the system would still work if the system was disconnected.</td>
</tr>
<tr>
<td>(E12) Self-Starter</td>
<td>If the system was disconnected, the system would still work if it was connected to the system. An ideal system would be able to function without human intervention.</td>
</tr>
<tr>
<td>(E13) Synthesis</td>
<td>The ability of the system to be transparent with other systems and how well the system could be integrated into the system. An ideal system would receive a value of 0 since the ideal system would not be able to integrate or be transparent to other systems.</td>
</tr>
<tr>
<td>(E14) Alternative Power</td>
<td>The ability of the system to be able to operate without electricity from the general power grid. An ideal system would receive a value of not having an alternative power source.</td>
</tr>
</tbody>
</table>

Figure 1.2: System Elements

Each element’s importance to a specific group that would interact with the system was assigned
1.3. **SURVEY ELEMENTS**

A number between 1 and 10. An assignment of 1 would indicate that the particular group did not see the element as important in any way and an assignment of 10 would indicate that a particular group saw the element as being of the utmost importance to them. Because a surveillance system in a smart home could potentially be used to monitor the well-being of an occupant and report changes in their condition to a health care provider, each element was also assigned a value for how important doctors and health care providers felt that element was to them. The average element importance was used to compare the relative importance of certain elements to others and to find elements that had universal importance.

<table>
<thead>
<tr>
<th>Element</th>
<th>User</th>
<th>Engineer</th>
<th>Software Developer</th>
<th>Doctor / Healthcare Professional</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>6.75</td>
</tr>
<tr>
<td>E2</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>E3</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>8</td>
<td>8.25</td>
</tr>
<tr>
<td>E4</td>
<td>5</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>5.25</td>
</tr>
<tr>
<td>E5</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>7.75</td>
</tr>
<tr>
<td>E6</td>
<td>4</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>7.5</td>
</tr>
<tr>
<td>E7</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>E8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>E9</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>E10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>9.5</td>
</tr>
<tr>
<td>E11</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>E12</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>E13</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>8.75</td>
</tr>
<tr>
<td>E14</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
<td>7.75</td>
</tr>
<tr>
<td>Average</td>
<td>8.71</td>
<td>9.43</td>
<td>7.71</td>
<td>8.57</td>
<td>8.61</td>
</tr>
</tbody>
</table>

Table 1.1: Element Importance

We then used a similar scale to evaluate the object locating systems and the general purpose, multiple camera surveillance systems. Values of 1 to 10 indicate how close each system is satisfying the ideal for a particular feature. Values of 0 correspond to features that none of the systems exhibited and they were not included in the calculation of the average value that was assigned to each system. Because all of the systems could not be properly evaluated together, the systems that performed object tracking in a network of multiple cameras were separated from the systems which performed general tracking with multiple cameras. The systems that located objects are presented in Table 1.2 and evaluated in Table 1.3, while the general surveillance systems are presented in
1.4 Discussion of Similar Systems

Table 1.4 and evaluated in Table 1.5.

1.4 Discussion of Similar Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>[Campbell and Krumm 2000]</td>
</tr>
<tr>
<td>S2</td>
<td>[Cucchiara et al. 2005]</td>
</tr>
<tr>
<td>S3</td>
<td>[Fleck et al. 2006]</td>
</tr>
<tr>
<td>S4</td>
<td>[Nelson and Green 2002]</td>
</tr>
<tr>
<td>S5</td>
<td>[Williams et al. 2007]</td>
</tr>
<tr>
<td>S6</td>
<td>[Xie et al. 2008]</td>
</tr>
</tbody>
</table>

Table 1.2: Object Locators

<table>
<thead>
<tr>
<th>Element</th>
<th>Average</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>7.2</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>Friendliness</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Range</td>
<td>6.6</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Calibration</td>
<td>7.8</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>System Complexity</td>
<td>7.8</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Software Complexity</td>
<td>8</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Robustness</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Scalability</td>
<td>5.8</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Lifetime</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Realtime</td>
<td>8.8</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Reliability</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Self-Start</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Synthesis</td>
<td>5.2</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Alternative Power</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>7.43</td>
<td>6.83</td>
<td>7.83</td>
<td>7.25</td>
<td>7.08</td>
<td>8.17</td>
<td>7.67</td>
</tr>
</tbody>
</table>

Table 1.3: Evaluation of Object Locators

The single camera system presented in [Campbell and Krumm 2000] appears to perform exceptionally well for object tracking within one camera view. It effectively locates and highlights objects.
that it has been instructed to track, and it does so with hardware that could be easily obtained by the average consumer. Parameters that would be needed to tune the performance of the system could be set in a user-friendly manner and the system can effectively learn the appearance of tracked objects with minimal user interaction. Such a method appears to be a good base for a system that tracks objects within a smart home. With the addition of some multiple camera cooperation elements of from [Xie et al. 2008] and [Cucchiara et al. 2005], the benefits of the single camera tracking in [Stuart et al. 1999] may have the ability to be enhanced.

Systems, such as those in [Nelson and Green 2002] and [Williams et al. 2007], that used Pan-Tilt-Zoom (PTZ) cameras seem to be effective in the task of robustly tracking an object that is within the camera’s field of view, but are less than ideal because of the additional cost of each camera. Furthermore, the decision in [Nelson and Green 2002] to restrict monitoring to small areas where an object is expected to be is not robust to the addition, or movement, of furniture. If a camera was dedicated to monitor the location of objects that were placed on a table, and that table were moved out of the camera’s view, the camera would have to be moved as well.

Because of the problems presented by creating systems that are exclusively designed with the goal of tracking objects within a smart home, it would seem ideal that object tracking be done with only the images that are used for the broader tracking tasks within a smart home. If methods were developed for tracking relatively small objects with the same, static cameras that would be used for tasks such as fall detection, object locating could become more robust to changes that are common within a home environment.

\section{Discussion of Generic Systems}

<table>
<thead>
<tr>
<th>System</th>
<th>Citation</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>[Black et al. 2002]</td>
</tr>
<tr>
<td>G2</td>
<td>[Chen et al. 2008]</td>
</tr>
<tr>
<td>G3</td>
<td>[Khan and Shah 2003]</td>
</tr>
<tr>
<td>G4</td>
<td>[Krumm et al. 2000]</td>
</tr>
<tr>
<td>G5</td>
<td>[Nguyen et al. 2002]</td>
</tr>
<tr>
<td>G6</td>
<td>[Velipasalar and Wolf 2005]</td>
</tr>
</tbody>
</table>

Table 1.4: Generic Tracking Systems

In the broader context of tracking people and objects within a smart home, much can be learned from the work presented in [Chen et al. 2008], [Velipasalar and Wolf 2005], and [Fleck et al. 2006].
1.5. DISCUSSION OF GENERIC SYSTEMS

<table>
<thead>
<tr>
<th>Element</th>
<th>Average</th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>7.83</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Friendliness</td>
<td>7.17</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Range</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Calibration</td>
<td>7.83</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>System Complexity</td>
<td>8.5</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Software Complexity</td>
<td>8.17</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Robustness</td>
<td>7.67</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Scalability</td>
<td>8.83</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Lifetime</td>
<td>7.83</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Realtime</td>
<td>8.17</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Reliability</td>
<td>7.83</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Self-Start</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Synthesis</td>
<td>7.67</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Alternative Power</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Average</td>
<td>8.04</td>
<td>8</td>
<td>8.33</td>
<td>8.33</td>
<td>7.42</td>
<td>7.75</td>
<td>8.42</td>
</tr>
</tbody>
</table>

Table 1.5: Evaluation of Generic Tracking Systems
1.6. Conclusions

The entry/exit zones and methods for adapting to sudden changes in illumination are two proposals from [Chen et al. 2008] that appear to be directly applicable to tracking in smart homes. The authors’ discussion of a priori initialization of known links between cameras and closed/open links in unmonitored regions seem directly applicable to the home. When a surveillance system is installed in a home, this information is easily obtained and can greatly reduce the time needed for a system to become operational. The inclusion of information about closed zones could also be used to refine an object locating service’s response if the exact location of an object is not known. If the system can tell the user that the object is in a closed link between cameras, the area that the user would need to physically search in would be greatly reduced. If the methods for learning field of view lines in [Chen et al. 2008] and [Fleck et al. 2006] were combined with the learning of entry/exit zones and a tracking algorithm that did not necessitate an unobstructed view of the ground plane, immensely robust tracking may be possible in all monitored areas of a smart home.

1.6 Conclusions

This paper reviewed systems that are currently used to for the specific task of tracking objects in a smart home and systems whose methods could be used to track objects within a smart home. While no one system has been ideal, many system contribute methods that can become important parts of a more effective system. There is still research to be done into robustly tracking the wide variety of possible objects that one camera may see, and into methods that would allow multiple cameras to share the information that they gather amongst themselves. With advances in both research areas and the integration of results, it may eventually be possible to provide the occupants of smart homes with a near-ideal system for keeping track of the objects that they value the most.
2 Systems

Throughout our research, we encountered difficulties that required us to use different data sets.

2.1 Our System

We initially attempted to design, implement and use our own system to create data for our software.

2.1.1 Our Hardware

We initially thought to approach this problem by building a small-scale version of a smart room within our lab. We purchased two Linksys WVC54GCA Wireless-G Internet Home Monitoring cameras. The AC-powered cameras can produce individual JPEG-compressed frames or an MJPEG stream of multiple frames and transmit over a wired or wireless network. The cameras also contain open-source firmware [Cisco 2010] [Pastor 2009] that could potentially be used to distribute vision tasks that are currently centralized.

In addition to the two Linksys cameras, we wanted an infrared camera that could perform in a dark environment when the conventional cameras would be hindered by the low lighting conditions. At first, we purchased a Logitech WiLife Indoor Security camera that we believed to have infrared capabilities. The camera attached to an AC power supply via a camera cable that resembled a phone line, and an additional AC-powered receiver was provided with a USB plug that would be connected to a computer.

Unfortunately, the Logitech camera presented many problems. The camera did not have the ability to capture infrared video built in to its hardware, and infrared video could only be captured with an infrared illuminator that had to be purchased at an additional cost. Furthermore, the method that was used to transmit video from the camera was not conducive to simple data acquisition.
2.1. OUR SYSTEM

Initially, we believed camera transmitted video wirelessly in the same way that the Linksys cameras did. While the camera’s documentation insisted that video could only be viewed in the proprietary application that accompanied the camera (an assertion that was echoed by support staff at Logitech), we believed that the video was between the power supply and the receiver, and simply converted by the receiver to resemble video that would be received from a generic, USB webcam. Monitoring of the transmissions between the power supply and receiver seemed to suggest that this hypothesis was correct, and patents for the camera [Willes et al. 2005] seemed provide more evidence that the camera could transmit video wirelessly in the MJPEG format. Evidence that this was untrue came when more information about technology related to Broadband over Power Lines (BPL) was discovered [Logitech 2008]. The camera appeared to transmit its video through electrical wiring.

With the desire to use an infrared, network camera that behaved in a similar manner to the Linksys cameras that we were already using, we found the AirLink101 SkyIPCam500W Wireless Night Vision Network Camera [AirLink 2008]. Like the Linksys cameras, this camera had the ability to transmit an MJPEG video stream wirelessly, or through a wired Ethernet connection. While it functioned in a similar way to the Linksys cameras, it also had six built-in, infrared sensors that could be activated automatically by a low-lighting sensor.

2.1.1.1 Image Acquisition

Learning how to acquire images and video from the cameras in OpenCV was not as simple as expected. While OpenCV allows for the creating of a CvCapture object that can be used attach to a video and grab individual frames, we eventually concluded (contrary to some assertions) that such an object could not be used on an MJPEG stream. After looking at the firmware of the Linksys cameras, we found that individual JPEG frames could be requested from those cameras, but OpenCV did not have built-in functions that would allow for a JPEG image that was stored in memory to be converted to OpenCV’s IplImage format without saving the image to the disk and loading back in to memory with cvLoadImage. Eventually, we found a way by which a compressed JPEG image that was stored in memory could be converted to an IplImage through use of the Independent JPEG Group’s JPEG Image Library [IJG 2010].

While converting a JPEG image to an IplImage object in memory saved time and fatigue on the disk, only being able to request and receive individual frames from the Linksys cameras limited the cameras that we could use and reduced our rate of capture from two Linksys cameras to about three frames per second (from each camera). In order to increase that collection rate, we needed to reduce the overhead of making one HTTP request for each frame that we wanted each camera to transmit.
Finding a simple method for obtaining the MJPEG streams did not have a simple solution. Our first instinct was to use the program wget\[GNU 2009b\] to non-interactively begin downloading the stream, then begin reading and parsing that file. However, downloading a stream to a named file, then reading it simultaneously was not a viable solution. The program curl\[haxx 2010b\] performed many of the same tasks as wget, but its default action was to dump the downloaded data to stdout, instead of to a named file. In addition, curl had a library (libcurl) that could be used to download directly from within a C program, and a function that would generate C code for given command-line execution\[haxx 2010a\]. Unfortunately, use of libcurl did not seem to solve the problem of parsing, processing, and discarding the MJPEG stream as it was received.

Eventually, while searching through the stdio.h file of the GNU C Library[GNU 2009a], we stumbled across the function popen[GNU 2009c]. The function took two strings (a command and an access mode) and returned a file pointer. The function forks a child process, has that process execute system(command), and returns the output through a pipe to the file pointer. By executing

\[
popen(<\text{MJPEG stream URL}>,r);
\]

we were able to treat the MJPEG stream as if it were a normal video file, and parse out the individual JPEG frames. Where requesting individual frames from the Linksys cameras only allowed us to achieve a frame rate of approximately three frames per second (on both wired and wireless networks), accessing the MJPEG stream increased our data collection from one camera to approximately 10 frames per second on a wireless network and approximately 20 frames per second on a wired network.

<table>
<thead>
<tr>
<th>Network</th>
<th>Mode</th>
<th>Frames / Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless</td>
<td>Snapshot Request</td>
<td>3.0157</td>
</tr>
<tr>
<td></td>
<td>MJPEG Stream</td>
<td>10.7181</td>
</tr>
<tr>
<td>Wired</td>
<td>Snapshot Request</td>
<td>2.7382</td>
</tr>
<tr>
<td></td>
<td>MJPEG Stream</td>
<td>20.0803</td>
</tr>
</tbody>
</table>

Table 2.1: Single Linksys Camera Transmission Rates

Unlike the OpenCV function cvQueryFrame, this method, as implemented, could not simply grab the most recent frame from the MJPEG stream. If a frame was requested several seconds after the stream had been attached to, the frame returned would be the first frame received from the stream. A threaded implementation may behave more similarly to cvQueryCapture.
2.1. OUR SYSTEM

2.1.1.2 Synchronization

While popen allowed us to capture video in a simple manner, it required that a child process be created for each video stream that was to be accessed. If the streams were accessed sequentially, by the main program, $n + 1$ processes would be required to collect frames from $n$ cameras. However, if threads were used, to prevent one malfunctioning stream from disrupting the processing of the other streams, $2n + 1$ processes would need to be executing for the duration of the program’s execution.

We operated on the assumption that our system could not handle any malfunctioning streams and the system would want to begin processing frames immediately. Therefore, after we began capturing each video stream, we sequentially processed one frame that was parsed out of each of the streams. With only two cameras (requiring three concurrent processes), the usual delay between the displaying images from the same instant in time was tolerable. However, with the addition of a third camera (required the addition of another concurrent process), the system could not provide anything that resembled synchronization. While the first two camera streams that were accessed appeared to be received within a reasonable time of one another, the third would lag far behind the other two.

2.1.1.3 Image Quality

Some of the synchronization problems were likely the result of our demands for frames of the highest quality and a maximum frame rate of 30 frames per second. These demands were made because of how we wanted to track the movement of objects. Instead of waiting for activity in an area to cease [Nelson and Green 2002] or tracking through recognition [Xie et al. 2008] [Li et al. 2004], we wanted to track objects continuously from an initial, standardized position. Continuous tracking, in a sizable, complex area, with cameras that did not have the ability to pan, tilt, or zoom, would require both high resolution frames and a fairly fast frame rate.

To meet our demands, the Linksys cameras had to transmit individual frames that exceeded 60 kilobytes each, and the infrared camera had to transmit individual frames that exceeded 27 kilobytes each. Assuming that each camera could transmit only 10 frames per second over the wireless network, the central node that processed the video would still have needed to process about 1,470 kilobytes of data for each second that the system was operational, just to acquire the video frames.

2.1.2 Background/Foreground Segmentation

While many algorithms have been proposed (and a few have been implemented by the developers of OpenCV), most background/foreground segmentation algorithms require time to learn a scene’s
background from a fixed vantage point. Because we did not have a permanent, static setup for our system, we had to cobble together rough background subtraction and thresholding in order to produce an approximation of background/foreground segmentation.

### 2.1.2.1 Incompleteness

Our implementation of background subtraction led to a trade-off between segmenting every foreground pixel as a member of the foreground and segmenting every background pixel (including shadows and reflections on the background) as a member of the background. Because our system focused on tracking objects that began on a table in the center of the lab, where shadows that may be cast on the floor were unlikely to be seen by the cameras, we erred on the side of including too many pixels in the foreground. This led to occasions where a shadow would appear as a part of the foreground.

![Figure 2.1: Shadow in the Foreground](image.png)
2.2. ICDSC SMART HOMES DATA SET

2.1.2.2 Background Over Time

Our background subtraction method was designed to solve one of the problems that modern foreground segmentation algorithms create for our specific situation. Modern foreground segmentation algorithms are designed to adjust to gradual changes in lighting in the scene and gradually incorporating stationary objects into their background model. While (with the gradual and sudden changes in lighting in our scene) we find adjustments to lighting changes useful for segmenting foreground objects from the background, our application centers around tracking objects that remain stationary for long periods of time. By performing simple background subtraction between a relatively static scene and one background frame, we are able to include both moving objects and static objects that are of interesting to us, over the duration of our video samples.

2.2 ICDSC Smart Homes Data Set

During our survey of existing surveillance systems in smart homes, we found the website of the IEEE International Conference on Distributed Smart Cameras (ICDSC) 2009[“ICDSC” 2009]. The conference organizers invited participants to submit papers that addressed open-ended problems in one of two datasets. One of the datasets was a set of videos where one person was recorded performing a number of common tasks. The videos were captured by eight synchronized (but uncalibrated) cameras that were set up to monitor areas of a kitchen, a living room, and the hallway connecting the two rooms. None of the papers that were submitted to the conference addressed that dataset.

2.2.1 Image Quality

The dataset, while synchronized and extensive, was flawed in many ways. The captured frames had a width of 320 pixels and a height of 240 pixels. While that resolution may have been useful for a number of vision tasks, the compression of the frames made them appear particularly blurred.

The combination of the quality of the cameras and the lighting of the environment also created areas of some frames where interesting objects that could have been tracked had their initial positions occluded by exceptionally bright lighting (such as the coffee mug on the counter). Beyond the problems created by the quality of individual frames, the frame rate of 10 frames per second and the quality of the cameras contributed to exceptional motion blur.
Figure 2.2: Frame from ICDSC 2009 Smart Homes Data Set
Figure 2.3: Effects of Motion Blur
2.2.2 Background/Foreground Segmentation

The attention that was not paid to items that were contained in the background images that were supplied with the dataset also caused problems with correctly identifying interesting objects to track. In two instances, objects that would have been useful to track throughout their movement through the environment were included in the background images of the environment.

Figure 2.4: Mug in the “Background”
Figure 2.5: Magazine in the “Background”
2.3 The TUM Kitchen Data Set

The Kitchen Data Set from the Technical University of Munich consisted of 21 sets of videos that recorded an individual setting a table in a kitchen setting [IAS 2009]. The data set was intended for research in recognizing human actions from video [Bourbakis et al. 2000], but we used it for our tracking purposes. Each set of recordings included video captured from four synchronized cameras that were placed above a kitchen environment. While the cameras did not monitor disjoint areas of the scene, the video provided usable video for the task of tracking objects from multiple vantage points.

![Figure 2.6: Views from the TUM Kitchen Data Set](image)

Unlike the previous data sets, the TUM Kitchen Data Set provided useful video frames. The frames had a width of 384 pixels and a height of 288 pixels and the video had a frame rate of 25 frames per second.
2.3. THE TUM KITCHEN DATA SET

2.3.1 Background/Foreground Segmentation

Like with the previous data sets, we elected to use simple background subtraction to distinguish foreground objects from background objects. We decided to track a large, white cutting board as it was moved from the top of the stove to the wooden table. Because the data set was created to focus on human motion, objects in the scene (including our object of interest) were simply assumed to be part of the background. To alter the data for our purposes, we had to manually replace the region of the first frame that included the cutting board in its starting position with the same region at a later frame in the video (when the cutting board has been moved) using photo editing software.

As with the data from our own system of cameras, we initially performed background subtraction with the intention of erring on the side of classifying too many pixels as foreground pixels. We wanted to segment as much of the cutting board, and person who moved it, as possible. We believed that the effects of shadows on surrounding background surfaces would be minimal, and that they would not affect the overall performance of our object tracking.

Unfortunately, including some shadows in the foreground was enough to disrupt some of the tracking algorithms, like the continuously adaptive mean-shift (CAMShift) algorithm, that relied on adjusting one hypothesized location and size of a tracked object. If the cutting board was occluded, but the person’s shadow on a white cabinet was near the board, the CAMShift tracker would instead track the cabinet, even after it moved into the background.
Figure 2.7: Creation of a Background Image
Figure 2.8: CAMShift Tracking Board
Figure 2.9: CAMShift Tracking Cabinet
Figure 2.10: CAMShift Tracking Background
3

Object Tracking

3.1 CAMShift Tracking of the Cutting Board

At first glance, we thought that our basic problem of tracking the cutting board in a single, un-occluded camera could be solved by CAMShift tracking [Bradski 1998]. OpenCV had an implementation available [Bradski and Kaehler 2008] that seemed like it would have worked robustly for tracking the cutting board, if given the initial position of the cutting board (the location of the only foreground object in the first frame) and the current frame (with the background pixels zeroed out). The CAMShift had the added benefit of adjusting to an object’s size in the frame. This would allow the tracker to keep tracking the object, even as it moved closer to and farther away from the camera.

However, CAMShift’s reliance on hue and saturation information means that a white object (with an indistinguishable hue and low saturation) does not produce a well defined color histogram and back projection of the histogram, that can be reliably tracked. Therefore, we initially moved away from using the CAMShift tracker.

3.2 Template Matching Tracking of the Cutting Board

Because it appeared that algorithms that provided robust tracking did so with hue and saturation information, we spent a brief time attempting to track the white cutting board with brute force template matching. We thought that, even though template matching is inefficient, and OpenCV’s implementation of template matching only returns the point that is at the center of the template’s location on the image, it might be enough to track the cutting board’s movement throughout the video sequence.

Unfortunately, when template matching was faced with even partial occlusions and white cabinets that were brought into the foreground by the shadows cast upon them and our enthusiastic
Figure 3.1: CAMShift Tracking with Changing Scale
3.3. SURF POINT CORRELATION

background subtraction algorithm, the region that was matched with the template of the board often belonged to an object that was not the cutting board.

![Incorrect Template Matching](image)

**Figure 3.2: Incorrect Template Matching**

In an attempt to account for how the orientation and size of the cutting board affects the performance of template matching, we decided to not just attempt to match templates from a given camera in frames from that same camera, but to match templates from all cameras with each frame from each camera. While results improved for some of the frames from some of the cameras, this method took four times longer and the increase in the number of templates provided the template matching algorithm more opportunities to find the wrong location with a higher confidence.

### 3.3 SURF Point Correlation

Our next attempt at tracking was to use Speed Up Robust Feature (SURF) [Bay et al. 2008] points, which are similar to Scale-Invariant Feature Transform (SIFT) points[Lowe 1999], to correlate views of the cutting board in multiple views. The hope was that by correlating points in the four views,
3.3. SURF POINT CORRELATION

Figure 3.3: Template Matching Based on All Views
we could find a projective transformation between each camera and be able to track an occluded object using the position of the object in the unoccluded cameras.

We planned to use cvExtractSURF to find the keypoints in each image, then correlate the points from each camera to find all of the matches. However, the initial size, orientation, and occlusion of the board in some of the cameras prevented the finding of all but one correlation (between a single point in one view and a single point in only one other camera). Because of the variation of the views of the board’s initial size, SURF point correlation could not work in our situation.

### 3.4 Good Features to Track

We attempted to use the fact that the board should have four distinct corners to track the position of the board by tracking the position of the corners. Since the board was the only foreground object in the first frame of the videos, we used cvGoodFeaturesToTrack to find the positions of a maximum of four corners in the foreground image. From there, we believed that we could track those points through the movement of the board. As it turned out, the best four initial feature points were quickly replaced by four stronger points in subsequent frames and we moved away from attempting to perform tracking in this way.

### 3.5 Indirect CAMShift Tracking

After we were unable to track the cutting board directly, we attempted to track it indirectly. Our thinking was that the cutting board could only be moved by a person, and we could tell that a person was touching (and, therefore, preparing to move) the board when the bounding box that enclosed the view of the cutting board was intersected by a bounding box that surrounded the arm of the person who was about to move the board. To ensure that the person was really touching the board (and the arm was not merely occluding the board in one camera), we insisted that each camera, in which the board and arm in question were visible, had to show the two bounding boxes intersecting. Because skin had a hue and saturation level that could be distinguished by the CAMShift algorithm, we believed that this method would be more successful than attempting to only track the white cutting board.

Our plan to track human limbs against a fairly static background appeared similar to the method employed in [Wilson and Salgian 2008]. Instead of taking the time to attempt to train a classifier for human skin [Kakumanu et al. 2004], we simply sampled some of the images of skin from our dataset and identified the range of Hue/Saturation/Value (HSV) values that represented skin. From there, we filtered out everything from the foreground image, with the exception of the skin and the board.
Figure 3.4: Segmentation of Skin and Cutting Board
3.5. INDIRECT CAMSHIFT TRACKING

To avoid the potential problem of the system concluding that the board is being touched, when a subset of cameras detect the left (but not the right) hand touching the board, while a different subset of cameras detect the opposite, we decided to initially detect and track the specific left or right arm from the frame in which the person enters the scene and extends his arms.

Soon, we found that we had made our original problem more complex. When the board was visible, it could be seen quite well in the frame. Conversely, the combination of foreground and skin segmentation had the effect of occasionally erode the view of an arm to a level where it could not be identified. Furthermore, attempting to indirectly track one object, by directly tracking two, amplified the problems of tracking through occlusion. In time, we abandoned our plan to track the board indirectly.

Figure 3.5: Determination of the Arms
When we resolved ourselves to using CAMShift tracking to directly track the cutting board, we needed to find a color space where white and shades of gray could be uniquely identified in the back projection of the color histogram. We had some success by filtering out all of the foreground pixels that were not the same color as the cutting board, but that solution appeared to be a very delicate solution. In changing the color space, we attempted to invert the values of the red and blue channels of the Red/Green/Blue (RGB) frames, and leave only the true value of the green channel. This produced a bright green cutting board, a blue person, and a magenta floor. However, this was not enough for the CAMShift tracker to work.

Eventually, we attempted to swap the hue and value channels in the HSV frames, before using CAMShift tracking. The change was successful enough and was used as the basis for later tracking.

Figure 3.6: Value - Saturation - Hue Image of Foreground
3.7 Kalman Filter

While unoccluded tracking in one camera could be solved with the CAMShift tracker, it is unlikely that an object that is being moved within a room will always be visible. To continue to track the cutting board through occlusion, we needed a to estimate the board’s position when it was not visible. For that, we initially attempted to use OpenCV’s Kalman filter object.

The Kalman filter attempts to learn the dynamics of a linear system, and make a single, accurate hypothesis about a model’s state at a given time. We felt that these constraints would still allow us, with proper occlusion detection, to track the cutting board through an occlusion and reacquire it with the CAMShift tracker after the board re-emerged from the occlusion. We believed that it was sufficient to model motion during the short periods of occlusion as having a constant, linear direction, and a constant speed.

The Kalman filter requires dynamic parameters that will be estimated to determine the state of a given system, measurements that will determine how to adjust the estimate, and a transition matrix that determines how the dynamic parameters of one state affect the dynamic parameters of the next state. For our dynamic parameters, we settled on using the x/y coordinates of the center of the box that outlined the object that the CAMShift tracker was following, and the x/y velocities of that point. Each measurement we made would consist of the x/y coordinates of the center of the tracking box. The constant 1/25 was used to represent the change in time from one frame (or state) to the next because the video was recorded at 25 frames per second. The transition, therefore, was represented as

\[
\begin{bmatrix}
1 & 0 & 0.04 & 0 \\
0 & 1 & 0 & 0.04 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Table 3.1: Kalman Transition Matrix

Without proper occlusion detection, the Kalman filter was unsuccessful. While the filter learned the dynamics of the system, the non-linear motion of the board that took place before the occlusion biased the model when the occlusion occurred. As the board was being occluded, the model was hypothesizing that the board was continuing on an unrelated path.
3.8 Particle Filter

The Kalman filter’s restrictions did not allow us to accurately track the cutting board through occlusions. Even if it were successful, using it with CAMShift tracking would have required additional logic (and delicacy) to detect when an occlusion was occurring and sizing/positioning of the CAMShift tracking window (which would have shrunken as the board was being occluded) to reacquire the object.

A particle filter takes the idea of the Kalman filter and expands it to included an arbitrary number of hypotheses. Instead of using measurements to update the parameters (which the Kalman filter is supremely confident of), particle filters use measurements to update the confidence that it has in each of its hypotheses. OpenCV includes an implementation of the Condensation algorithm [Isard and Blake 1998], but the documentation on how to use it is limited. We used a particle filter implementation from Rob Hess at Oregon State University [Hess 2006] to test particle filter tracking on our VSH frames. This implementation of a particle filter used the backprojection of the object’s hue histogram to find the most likely location of the object in the video frame. Unlike OpenCV’s implementation of CAMShift tracking, the particle filter handled ambiguous hues (caused by poor saturation) by placing them in a special category. While the tracking was successful through some occlusions, it did not work as well if the initial position of the cutting board was too far away from the camera.

3.9 Conclusions

In retrospect, using the Kalman filter to track the object in the backprojection of the hue channel may have bee more successful than using a particle filter (in our specific instance). While the particle filter jumped around if the object was occluded, the Kalman filter’s estimation of the object’s location and velocity would have only been minimally altered by seemingly random measurements that affected the particle filter.

Overall, particle filters are probably the best tools to use for tracking objects within one camera. While they can act unpredictably when given measurements that occur while objects are occluded, they show a great ability to reacquire their targets when the targets emerge from their occlusions.
Figure 3.7: Particle Filter Tracking Pick-up
Figure 3.8: Particle Filter Tracking Movement
4

Future Work

There is still much work to be done with this project.

4.1 Data Processing

The first step towards creating a surveillance system for a smart home will need to be taken in the area of video processing. Transmitting every frame, from every camera, to a central node, and expecting that node to do all of the processing is impossible. Having each camera maintain its own background model and perform its own foreground segmentation would greatly reduce the amount of data that needs to be transferred within the camera network. Figure 4.1 and Figure 4.2 show a frame before and after foreground segmentation. After foreground segmentation, the JPEG frame’s size is reduced by 84.944% (49071 bytes to 7388 bytes). Additional privacy preserving steps, such as silhouetting or background replacement could be performed on each camera to circumvent network packet sniffing.

Many of the tasks of tracking could also be decentralized if processing were to take place on the cameras themselves. If the ICDSC Smart Homes Data Set is an indication of how a surveillance system in a smart home would be set up (with four cameras in each room), many cameras in a home would be idly updating their background models and maintaining the position of stationary objects for large amounts of time. Based on the assumption that this system would be deployed in a home with one (or two) elderly residents, the majority of cameras would be monitoring a static environment, while the minority would be following the effects that the residents would have on tracked objects. In such a case, distributing the workload related to object tracking and identification could eliminate the need for a central processing station.
4.1. DATA PROCESSING

Figure 4.1: Video Frame
Figure 4.2: Same Frame after Foreground Segmentation
4.2 Information from Multiple Cameras

While this work was able to present methods for performing a degree of tracking in a home environment, it did not do so with information from multiple cameras. In the past, sharing information between multiple cameras required that the individual cameras be calibrated both with respect to their internal parameters and the relation to the scene that they were monitoring, and that the entire system of cameras be calibrated with respect to the physical relation that camera had to the others.

In a smart home, manual calibration is not a reasonable expectation. Camera positions may be altered as the needs of the residents change; furniture may be repositioned as preferences of residents change; and camera orientations may be changed as the result of random events that occur in everyday life. Fortunately, research is being done into using assumptions about the shape of man-made objects to automatically discover the environment that a camera would monitor [Pflugfelder and Bischof 2005]. If combined with methods for automatically learning inter-camera relations [Khan and Shah 2003], it may be possible to create a network that automatically adapts to physical alterations.

4.3 Sudden Lighting Changes

While we did not address dynamic background models, adaptation to gradual lighting changes has been addressed in the literature already. Adaptation to sudden lighting changes is still an open problem. While there have been methods proposed to adapt normal surveillance cameras to sudden lighting changes, the introduction of cameras with infrared capabilities will require adaptation for the sudden changes between color spaces that occur when cameras like our infrared camera detect low lighting conditions.
References


AIRLINK. 2008. Skyipcam500w wireless night vision network camera user’s manual.


CISCO. 2010. Wvc54gca firmware.


FLECK, S., BUSCH, F., BIBER, P., AND STRABER, W. 2006. 3d surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3d. In CVPRW ’06: Proceedings
4.3. SUDDEN LIGHTING CHANGES


GNU. 2009a. Gnu c library - gnu project.


GNU. 2009c. Pipe to a subprocess - the gnu c library.

HAXX. 2010a. curl - how to use.

HAXX. 2010b. curl and libcurl.

Hess, R. 2006. Particle filter object tracking.


IJG. 2010. Independent jpeg group.


4.3. SUDDEN LIGHTING CHANGES


Stuart, H. P., Russell, S., and Ostl, M. 1999. Tracking many objects with many sensors. In In Int. Joint Conf. on Artificial Intelligence (IJCAI. 1160–1171.


4.3. SUDDEN LIGHTING CHANGES

