PerVision: An integrated Pervasive Computing/Computer Vision Approach to Tracking Objects in a Self-Sensing Space[1]

Hicham El-Zabadani, Sumi Helal, William Mann and Mark Schmaltz
Pervasive Computing Laboratory
Computer & Information Science & Engineering Department
University of Florida, Gainesville, FL 32611, USA
{hme,helal,mssz} @ cise.ufl.edu
wmann@phhp.ufl.edu

Abstract

We propose a novel approach to self-sensing spaces, in which classical computer vision algorithms are empowered by opportunities presented by the pervasive space. Our approach, which we call PerVision, extends classical object recognition and tracking algorithms by adding a self-assessment/adjustment loop in which sensors and actuators of the pervasive space are used to vary scene parameters to minimize errors in the recognition process. We present the PerVision concept and algorithms in the context of locating and tracking dumb objects such as furniture in a smart house. This work is a continuation of previous research in which we introduced the Smart Plug concept to locate, track, and remotely interact with appliances and electrical devices. Collectively, PerVision and Smart Plugs take us a few steps closer to realizing the ambitious vision of completely self-sensing spaces.

1. Introduction

The pervasive computing laboratory at the University of Florida is exploring a synthesis of several disciplines including (1) sensor network research, (2) real world modeling, (3) Computer Vision, and (4) self-organizing system principles, to realize the vision of self sensing spaces (SSS). Our vision is to create new capabilities in which smart spaces such as homes sense themselves and their residents; create their respective real world models; and enact an accurate mapping between the real world model elements and the physical world. If realized, this vision will enable many applications that rely on remote monitoring and intervention.

By enabling spaces to automatically detect their main aspects (e.g., floor plans, type of rooms etc.), and for objects (e.g., furniture pieces, appliances, etc.) to also be identified automatically, it will be possible to create on-the-fly, incremental, real-world models of the pervasive space. The identification and self-sensing encompasses sensing of the software components needed to interact - within the model - with the model elements. This renders an automatically created, interactive real-world model of the physical space. The model can be used in many applications, most notably in remote monitoring and intervention in the case of older people and those with disability needy of assistance. In this application, intervention occurs by activating actuators, which are mapped in the real world model to actual objects in the user's home. Intervention is limited however to objects equipped with actuators. For instance, while it is possible to monitor a chair and its current location in a house, it will not be possible to remotely move the chair from one place to another. It will be possible however to monitor the location and status (on/off) of a set of stove

[1] This is a publication of the Rehabilitation Engineering Research Center on Technology for Successful Aging, funded by the National Institute on Disability and Rehabilitation Research of the Department of Education under grant number H133E010106. The opinions contained in this publication are those of the grantee and do not necessarily reflect those of the Department of Education.

burners, as well as intervene to turn them off or on.

In a previous work [1], we presented Smart Plugs, an idea implemented in the Gator Tech Smart House [2] that enabled the self-sensing of all appliances and electronic devices. Smart Plugs also enabled the creation of an interactive real-world model through which these appliances could be interacted with. In this paper, we focus on sensing dumb objects, especially furniture. Such sensing is very important to making the real-world model "realistic" and usable by applications.

We realized early on we need to use some artificial intelligence techniques to sense dumb objects. Of all the branches of artificial intelligence, perhaps one of the most diverse is that of computerized vision systems. The range of applications for this relatively new, and still growing technology, is vast [3]. From mundane tasks like production line quality control and video surveillance to exciting new technologies such as self-sensing spaces needed by many applications in pervasive systems.

However, our early prototypes of vision based prototypes for sensing furniture in the GTSH delivered highly inconsistent performance and to a large extent proven to be unreliable. We observed though that the context in which vision algorithms are being used is unique and provides a rare opportunity to make the best out of what computer vision is able to provide. We observed that the pervasive space itself can offer assistance and can guide the basic vision algorithms to achieve the recognition and tracking goals. This is possible because the pervasive space is able to control the sensor/actuator network in the GTSH. By modifying the basic vision algorithms to self-score the resultant error in the recognition process, and to accordingly attempt to correct the score (lower the error below a predefined threshold) by issuing commands to the sensor/actuator network, it was possible to adjust scene parameter and optimize conditions for effective recognition.

In this paper we present PerVision, the pervasive computing based vision algorithm and give an experimental evaluation for this approach in an actual smart house setting. Section 2 quickly reviews the basic computer vision algorithms used in object recognition and tracking. Section 3 introduces PerVision and shows the use of the pervasive computing space capabilities including the RFID, smart floor, photo sensors, and automated (motorized) blind sensors and actuators. The extended vision algorithm (PerVision) is presented in section 4. Section 5 presents a case study and an experimental evaluation of PerVision, performed in the Gator Tech Smart House. Section 6 discusses current limitations and Section 7 concludes the paper and describes our ongoing and future work.

2. Computer Vision: Brief Overview

Computer vision is the branch of artificial intelligence that focuses on providing computers with the functions typical of human vision. To date, computer vision has produced important applications in fields such as industrial automation, robotics, biomedicine, and satellite observation of Earth. In the field of industrial automation alone, its applications include guidance for robots to correctly pick up and place manufactured parts, nondestructive quality and integrity inspection, and on-line measurements [4-9].

Computer vision is also playing a very important role in the domain of elder care. This is done by utilizing pervasive technologies, including vision, to develop systems for well-being monitoring, where behavioral data is used to determine trends in the quality of\life of frail and disabled older persons. The ideas of well-being and well-being monitoring cover a potentially huge area, involving a range of conceptual, methodological and instrumentation issues [10, 11].

Computer vision is the study of methods that can be used for allowing computers to understand images. The term "understand" refers to extracting specific information from images for a specific purpose, either to present it to a human operator, or for controlling a process. The image data used in a computer vision system is often

represented as gray-scale or color digital image in either 2-D or 3-D format.

Since a camera can be seen as a light sensor, there are various methods in computer vision that deal with light and images. For example, it is possible to extract information about motion in fluids and waves by analyzing images of these phenomena. Moreover, a subfield of computer vision deals with the physical process which forms a camera image given a scene of objects, light sources, and lenses in the camera.

A third field which plays an important role is biology, or biological vision system. Over the last century, large amount of studies on eyes, neurons, and the brain structures devoted to processing of visual stimuli in both humans and various animals. These studies have led to a broad, yet complicated, description of how "real" vision systems operate in order to solve certain vision related tasks.

Beside the above mentioned views on computer vision, many of the related research topics can also be studied from a purely mathematical point of view. Finally, a significant part of the field is devoted to the implementation aspect of computer vision; how can existing methods be implemented in various combinations of software and hardware, or how these methods can be modified in order to gain processing speed without loosing too much performance.

The typical tasks of computer vision could be characterized as:

- **Object Recognition:** detecting the presence and/or position of known objects in an image. (e.g., searching for digital images by their content or recognizing human faces and their location in photographs.)
- **Tracking:** tracking known objects through an image sequence. (e.g., tracking a single person walking through a shopping center.)
- **Scene interpretation:** creating a model from an image/video. (e.g., creating a model of the surrounding terrain from images, which are being taken by a robot-mounted camera.)
- **Ego positioning:** determining position and motion of the camera itself. (e.g., navigating a robot through a museum.)

The major functions of a typical computer vision system are image acquisition, preprocessing, feature extraction, and registration. The image or image sequence is acquired with an imaging system. Often the imaging system has to be calibrated before being used. In the preprocessing step, the image is being treated with "low-level" operations. The aim of this step is to do noise reduction on the image and to reduce the overall amount of data. Feature extraction means further reducing the data to a set of features, which ought to be invariant to disturbances such as lighting conditions, camera position, noise and distortion. The aim of the registration step is to establish correspondence between the features in the acquired set and the features of known objects in a model-database and/or the features of the preceding image. The registration step has to bring up a final hypothesis.

3. Pervasive Spaces for Computer Vision

The goal of our research is to be able to create a real time snapshot of the smart house at anytime that reflects the actual layout of salient furniture items in the house. We use this snapshot to create a realistic real-world model of the house. The only condition is that this process should be automatic when it comes to adding new furniture to the house. We can not rely on any human intervention every time a new furniture object is brought to the house. The smart house should be able to find this new object and track it automatically. We found it to be very difficult, if not impossible, to rely on computer vision alone to achieve this goal. We also realized that by integrating the vision system with the smart house, it will be possible to automate the recognition and tracking processes.

3.1 Using the Pervasive Space

A pervasive space contains a network of more or less specialized computational devices that interact among themselves and with the users. These devices, which are distributed in the physical space have a high degree of autonomy than we are used to, and they increasingly serve as a medium for cooperation and communication among humans. Some of these devices are moveable and possibly collocated with humans, while others are stationary. Since the devices are located in many places and some of them are able to move, they must be sensitive to their context or environments. These habitats are typically a combination of three spaces: the physical, the informational and the conceptual space [12].

Our pervasive space is called the Gator Tech Smart House, which is the culmination of more than five years of research in pervasive and mobile computing [2, 13]. The project's goal is to create assistive environments such as homes that can sense themselves and their residents and enact mappings between the physical world and remote monitoring and intervention services [14].

In this pervasive space, there exist many assistive technologies used to help the resident live safely. However, only two are related to this project: (1) the smart floor and (2) RFID technologies.

3.1.1 RFID Tags Describing Dumb Objects

The core of any RFID system is the Tag or Transponder that can be attached to or embedded within objects. An RFID reader sends out a radio frequency wave to the Tag and the Tag broadcasts back its stored data to the reader. The system works basically as two separate antennas, one on the Tag, and the other on the reader. The data collected from the Tag can either be sent directly to a host computer through standard interfaces, or it can be stored in a portable reader and later uploaded to the computer for data processing [15]. The main use of RFID in the Gator Tech Smart House is to enable the house to know exactly when and what is being brought into the house through the front door.

As shown in figure 1, the smart house is equipped with a gate RFID system that detects any object having an RFID tag. Figure 1 shows a chair entering the house where the reader reads the XML information stored on the tag and sends it to the smart house computer.

Figure 1. A chair detected by the RFID system in the smart house.

```
- <RFID>
  <Id>17830928EF498</Id>
  <Name>Dinning Chair</Name>
  <Description>Dining table chair with
    two arms</Description>
  <Height>35</Height>
  <Width>12</Width>
  <Length>14</Length>
  <Weight>21</Weight>
  <Area>168</Area>
  <Model>URL of the chair's model</Model>
  <MinHUE>9</MinHUE>
  <MaxHue>16</MaxHUE>
</RFID>
```

Figure 2. XML representation of a dining chair stored on the RFID tag.

The RFID tag attached to the furniture entering the house (e.g., dining chair) contains information related to that piece of furniture. This information will be used later to assist in locating the new furniture. Figure 2 shows an example XML description of a dining chair. The weight value stored in the RFID tag of the chair will be used to detect its initial approximate location using the smart floor, which we discuss in the next section. The MinHUE and MaxHue values will be used to give the vision system an idea of the expected color range of the chair. The model is simply an image representing the chair that will be used in the remote monitoring application.

3.1.2 The Smart Floor

The Gator Tech Smart House has a 2-inch residential-grade raised floor comprised of a set of blocks, each approximately 1.5 square feet in area. This raised surface simplifies the process of running cables, wires, and devices throughout the house. In addition, each block is equipped with a pressure sensor in the center as shown in Figure 3. The main benefit of such a smart floor is that it can detect any slight pressure anywhere on that block. In fact, we had to add resistors to the sensor cables to reduce sensitivity and eliminate fluctuations in the readings [16].

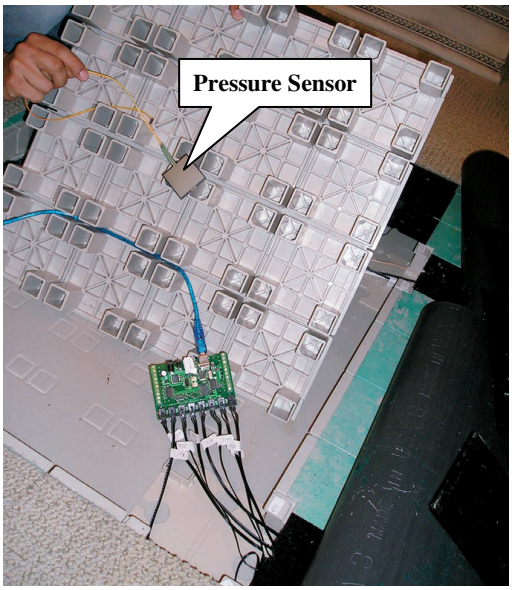
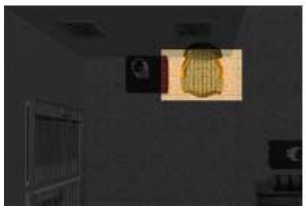


Figure 3. The pressure sensor used with each block in the raised floor.

The smart floor in the Gator Tech Smart House is used to locate the resident and help her/him in her/his daily activities. However, in this project we use the smart floor to assist the vision system with approximate location of the new piece of furniture when it first enters the smart house. Since we can detect any changes in the pressure on the smart floor tiles, we can select the tiles where the object is placed. Let us assume that two tiles experienced changes in the pressure after the front door RFID reader detects the advent of the chair to the house. The system knows the location of each tile in the smart floor. Knowing the weight of the chair (from the XML data stored on the RFID tag), we can make sure that the change on those two tiles is the result of placing that specific chair on top of them. As shown in Figure 4-a, someone brings the chair into the house. The RFID reader is the first device that discovers that new chair. After a period of time, the system runs an algorithm that will search for the chair guided by the smart floor. Figure 4-b shows how two tiles where the chair is placed are found. Finally, the ceiling-mounted camera in the center of the room where those two tiles are located takes an image, crops it using the coordinates of each tile, then shows only what lies above those two tiles as shown in Figure 4-c. This image will be used subsequently by the vision algorithm to locate the chair anytime it is moved. The smart house is also equipped with photo sensors that determine the level of light in each room. After taking the image, the value of the photo sensor in the room where the chair is located is saved. This value will be used later for intensity correction and error analysis.

a- The chair is placed in the desired location

b- The smart floor locates the tiles where pressure is changed



c- Having the coordinates of the pressed tiles, the camera can now zoom in to take a picture of what lies above those tiles

Figure 4. Using the smart floor to approximately locate a new furniture object.

4. The PerVision Algorithm

The preceding section described the first step towards locating any furniture in the house. This is done every time a new object is brought to the house. The idea is to get a top view of that object using the same camera under the same lighting condition. We will call this operation creating a signature of that object as discussed in Section 4.1. The object's signature will be used in PerVision: the vision algorithm that we will describe in Section 4.2 to locate that object.

4.1 Creating Signature

Figure 5 shows the algorithm applied to the resultant image from Section 3.1.

1. After locating any new furniture object using the smart floor, the camera takes an image which we call a source image.
2. It then crops it using the coordinates from the smart floor to obtain a top view for that furniture object alone.
3. Blurring this image is useful because it reduces variance at spatial frequencies below those that characterize a gross object, yet can preserve color information throughout the image.
4. The next step is to extract the Hue values of each pixel.
5. We then create a Hue histogram of that image.
6. Finally, we use the Gaussian distribution to get the min and max Hue values which correspond to: $\mu-2\sigma$ and $\mu+2\sigma$ where μ is the mean value and σ is the standard deviation. These operations are performed only once, then saved with the profile of the furniture object.

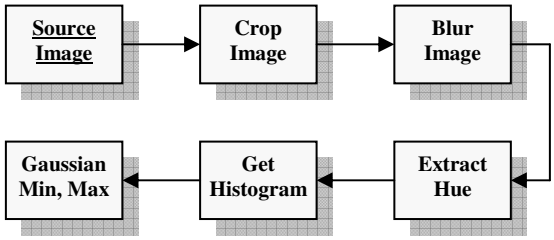


Figure 5. Performing one-time operations on the result image from section 3.1

4.2 Locating Furniture Using Color Restriction

In order to locate furniture in the smart house, we had to use color restriction algorithms. We can let a color (multispectral or hyperspectral) image **a** have L spectral bands, which causes each pixel value $\mathbf{a}(\mathbf{x})$ to have L spectral values, i.e., $a(x) = (a_1, a_2, \ldots, a_L)$, where x is a point in an MxN-pixel domain X. If each of the L band images $a_K \in \mathbf{Z}_m^{\mathbf{x}}$ has m graylevels, then **a** can be expressed in terms of the Cartesian product $\prod$, as follows:

$$\mathbf{a} = \left\{ (\mathbf{x}, \mathbf{a}(\mathbf{x})) : \mathbf{a}(\mathbf{x}) \in \prod_{k-1}^{L} \mathbf{Z}_m, \mathbf{x} \in \mathbf{X} \right\} \quad (1)$$

This formalism provides powerful insights that support design of pixel-level segmentation algorithms for multispectral imagery. For example, one can employ the characteristic function χ to compare each pixel value in each band with the extrema of the corresponding interval in an L-element spectral signature $\mathbf{s}=([s_{1L},s_{1U}], [s_{2L},s_{2U}], ,[s_{LL},s_{LU}])$. Note that the symbol L in s_{kL} denotes the lower bound of the real-valued interval, and should not be confused

with the number of spectral bands *L*, which is in italic typeface. Since one can also express s as the Cartesian product:

$$\mathbf{s} \in \prod_{k-1}^{L} \mathbf{Z}_m \tag{2}$$

it is possible to produce a segmentation **b** of image **a** constrained by signature **s**, which portrays the number of bands in each pixel value **a**(**x**) that match the signature **s**. For example, given the set of test values **G** and the characteristic function:

$$\mathbf{b} = \chi_{\mathrm{G}}(\mathrm{a}) = \left\{ (\mathbf{x}, \mathbf{b}(\mathbf{x})) : \mathbf{b}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{a}(\mathbf{x}) \in \mathbf{G} \\ 0 & \text{otherwise} \end{cases}, \mathbf{x} \in \mathbf{X} \right\} \tag{3}$$

consider the following expression:

$$\mathbf{b}(\mathbf{x}) = \sum_{k=1}^{L} \chi_{[s_{k\mathrm{L}}, s_{k\mathrm{U}}]}(\mathbf{a}(\mathbf{x})), \mathbf{x} \in \mathbf{X} \tag{4}$$

where each pixel $0 \leq \mathbf{b}(\mathbf{x}) \leq L$, with a higher number indicating a better match. It is also possible to combine the band-specific matches described above by performing logical operations such as *and*, *or*, *xor* (exclusive or). For example, logical *and* can be implemented as

$$\mathbf{b}(\mathbf{x}) = \prod_{k=1}^{L} \chi_{[s_{k\mathrm{L}}, s_{k\mathrm{U}}]}(\mathbf{a}(\mathbf{x})), \mathbf{x} \in \mathbf{X} \tag{5}$$

since the characteristic function χ returns a one or zero.

Let us discuss how color restriction can be implemented in practice. Firstly, note that the RGB representation of a digital color image is not intuitive: customarily, humans do not consciously think of colors in terms of how much red, green, or blue comprises a particular color. Rather, we learn colors by their names, such as red, brown, yellow, etc. In scientific terminology, these color names are called hues.

This brings us to our second observation, namely, that there is a more convenient color representation called *hue*, *saturation*, *value* (HSV) that makes color selection more human friendly. For example, consider the color palette shown in Figure 6. The rainbow display is actually a spectral representation of color, which can be expressed in terms of the HSV and RGB values. The spectral color (horizontal axis of the rainbow display) is called the *hue*, while the "depth" of the color is called the *saturation* (vertical axis). For example, the color *white* when fully saturated is shown at the top of the grayscale next to the rainbow display, while the unsaturated version, called *black* is pointed to by the arrowhead. The brightness of the color (also called *luminance*) is indicated by the number that denotes *value*. Because HSV and RGB representations are equivalent, it is relatively easy to convert between them.

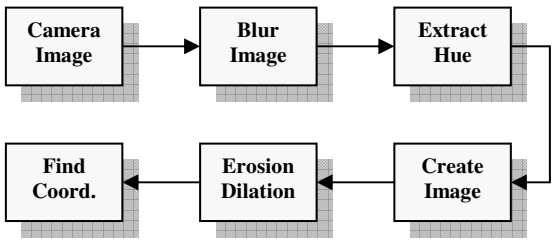


Figure 6. PerVision - Locating a furniture object using the Min and Max

Now that we have a unique signature for each furniture item in the house, we can start locating those objects one after the other. To be able to locate those objects we will use the PerVision algorithm depicted in Figure 6.

1. The first step is to get real-time image from the ceiling mounted camera.
2. Blurring is also applied to this image.
3. The next step is to extract the Hue values of each pixel in that image.
4. Using the Min and Max values from the previous section, we now create a new image where every pixel in the old image outside the range [Min, Max] is not included. This will result in a black and white image where the white pixels corresponds to the furniture object we are looking for.
5. There will be some noise in the image which will be removed using Erosion and Dilation.

6. Finally we can get the coordinates of that polygon which will correspond to the real location of that furniture object.

5. Case Study: The Gator Tech Smart House

Based on the PerVision algorithm described in the previous section, we built an application that locates furniture in the Gator Tech Smart House [2]. This application is used to test the algorithm's efficiency, precision, and usability. Figure 8 shows a snapshot of the application, where four windows can be seen. The top-left image is the live feedback from the camera. In this example, the application is trying to locate the table. Since the table is already in the house, the application now knows its signature given by the RFID tag.

The bottom-right window shows the result of the algorithm where a round white area represents the table in this image. This is the result of color restriction and morphological operations such as erosion and dilation. The top-right window shows the location of the table in terms of x and y coordinates. Finally, the bottom-left window shows the remote monitoring application view where a model of the table is shown in the exact location where the real table is.

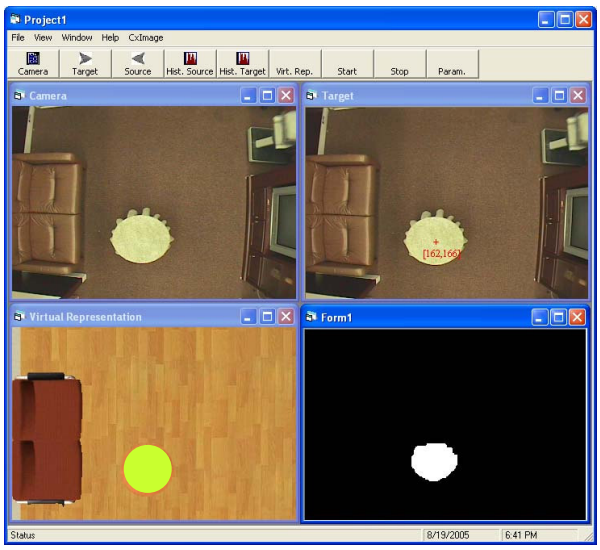


Figure 7. Demo application that locates furniture in the Gator Tech Smart House

5.1 Self Assessment

Like any vision algorithm, the probability of getting erroneous results is nontrivial. In this project we try to lower this probability by maximizing the use of the sensors and actuators in our pervasive space. First, we should note that the demo application not only locates furniture, but also assigns a score each time the recognition is done. The score's value ranges from 0 to 100 with 100 being the highest score. This score is computed based on several factors. The first one is the area of the white shape corresponding to the furniture item we are trying to locate. The closer that area is to the area of the furniture item from the XML information stored on the RFID tag, the higher the score.

$$S_1 = 100 - \left(\frac{\left| A_{result} - A_{original} \right|}{A_{original}} \times 100 \right) \qquad (6)$$

As shown in Equation 6, S_1 is the result of comparing the two areas. A_{result} is the area of the white shape and $A_{original}$ is the area stored on the RFID tag.

The second factor is based on the accuracy of the results of our location algorithm. The system compares the result with readings from the smart floor. The maximum score occurs when the result of the location algorithm is the same as the reading from the smart floor.

$$S_2 = 100 - \left(\frac{\sqrt{(X_s - X_v)^2 + (Y_s - Y_v)^2}}{MAX} \times 100 \right) \qquad (7)$$

In Equation 7, S_2 is the score resulting from comparing the new coordinates with the approximate ones obtained from the smart floor readings. X_s and Y_s are the coordinates obtained from the smart floor and X_v,Y_v are the ones obtained from the vision algorithm. MAX is the maximum distance that those two coordinates can have between them.

$$S = (0.7 \times S_1) + (0.3 \times S_2) \qquad (8)$$

Finally, we compute S which is the final score obtained by weight averaging both S_1 and S_2 as shown in Equation 8. The weight of S2 is lower than S1 due to the fact that the smart floor only gives an approximate result.

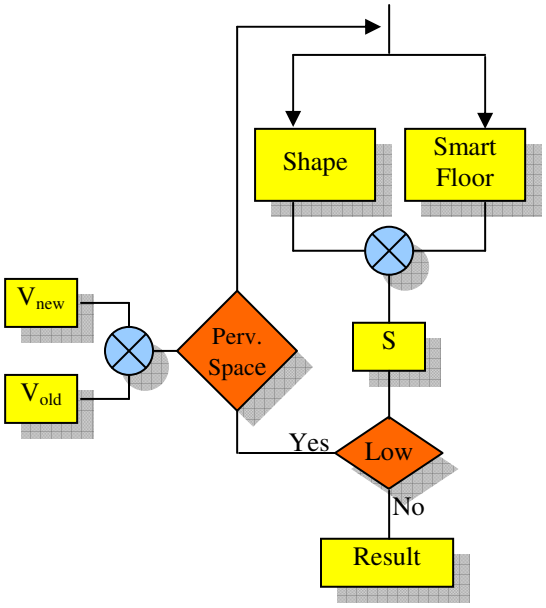


Figure 8. Self assessment and correction diagram

What happens when the score is low? Recall from Section 3.1.2, after taking an image of the chair for the first time in the smart house, the application stored the value V_{old} of the photo sensor in that room. In the case of a low score value, the application compares the current photo sensor value V_{new} with V_{old}. If there is any significant difference, the application uses the pervasive space to change the current luminance in that room so that it matches V_{old}. This is done by controlling the blinds in the specific (either opening or closing) or controlling the lights (if the lights are off). Figure 9 shows the diagram that explains the process of computing the score by checking the shape and comparing the result to the smart floor readings. If the score (S) is low, then V_{old} is compared to V_{new} and the pervasive space is used to change luminance until V_{old} and V_{new} are similar. This process is executed iteratively until we get a score equal to or higher than a predefined threshold.

5.2 Error Analysis and Experimental Results

We tested this system in the Gator Tech Smart House using the previous demo application [2]. The goal of this experiment is to show how a pervasive space could improve the recognition and tracking of various objects in the house. We used three objects: two chairs and one table. Figure 10 shows the floor plan of the kitchen where we did the experiments. The red shapes correspond to the blinds and the blue shapes correspond to light sources.

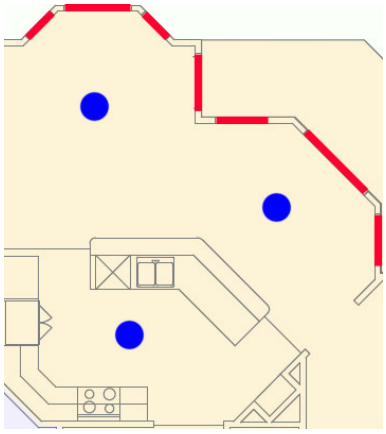


Figure 9. Floor plan of the kitchen

In this experiment, the system looks for the three objects one at a time. Since these objects are already in the house, the system knows their signatures (created when first brought into the house). By running the PerVision algorithm, we obtained the results shown in the following tables.

Table 1. Results for Chair1

Lights	Blinds	Photo Sensor	Score
Off	Open	375	68
Off	Closed	370	76
On	Open	612	91
On	Closed	612	97

Table 2. Results for Chair2

Lights	Blinds	Photo Sensor	Score
Off	Open	370	X
Off	Closed	358	X
On	Open	605	86
On	Closed	594	83

Table 3. Results for Table1

Lights	Blinds	Photo Sensor	Score
Off	Open	370	70
Off	Closed	350	74
On	Open	597	95
On	Closed	585	81

Tables 1, 2 and 3 show the results of recognizing and locating chair1, chair2, and table1 respectively. To get the best possible recognition result (score), the system tries four different combinations: (1) lights off/blinds open, (2) lights off/blinds closed, (3) lights on/blinds open, and (4) lights on/blinds closed. The system at this point chooses the result where the score is the highest. For chair1, the score was the highest when lights were on and blinds were closed. However, chair2 got the highest score when lights were on and blinds were open. Notice that the system could not recognize chair2 when the lights were off. Finally, table1 got the highest score when lights were on and blinds were open. The recognition system does not need to go over all combinations. The only reason we did all combinations here is just to show how controlling a pervasive space could improve the result of the recognition algorithm.

6. Limitations and Observations

In this project, we try hard to achieve the best results that guarantee a reliable location tracking algorithm. However, this algorithm still faces some limitations which we list them below:

- **Covered Objects:** In this case, a furniture object is covered with another object. For example, a table covered with a table cloth. This will block the PerVision algorithm from identifying this object. A solution could be to rely solely on the smart floor result with a low score assigned to the result.
- **Multi-Colored Objects:** The best result achieved using this algorithm is when looking for single color objects. The more colors used in objects, the harder the recognition would become.
- **Hidden Objects:** Of course, it would be impossible to locate hidden objects. Consider a chair lying below a table. The vision algorithm will not be able to locate the chair. The smart floor can help in this case to identify the chair.
- **Split Objects:** In some cases, an object could be captured by two different cameras with half in one camera and the other half in the other. This problem could be solved if the cameras' viewing areas are intersected.

The following observations were made while experimenting with the system and the PerVision algorithm described in this paper:

- Fluorescent light works better than the regular incandescent light. The colors are more vivid under fluorescent light.
- Sun light negatively affects the performance of the recognition algorithm even when blinds are closed. The optimal result the algorithm gets is when sun light is totally blocked.
- The carpet color should be carefully picked. Dark single color carpets work best. This is due to the fact that it is easier for the algorithm to distinguish between a dark carpet and the furniture.

7. Conclusion and Future Work

We described a novel approach to using computer vision to recognize and track dumb objects in a pervasive space. We have shown how RFID technology and smart floors could be used to assist recognition algorithm locate furniture in

a smart house. We described the extended vision algorithm we call PerVision (pervasive-assisted Vision) and showed its self-assessment features. We have conducted an experimental evaluation of PerVision in a real setting, which is the Gator Tech Smart House at the University of Florida. We have shown how the algorithm performance is significantly improved. We discussed current challenges and limitations and summarized our experience in implementing dumb object tracking.

We continue to work on improving our algorithm and in engaging other sensors and actuators in the self-assessment/adjustment loop of the algorithm. The objective is to realize a fully automated self-sensing space that can recognizes and track furniture and other objects.

References

[1] H. Elzabadani, A. Helal, B. Abdulrazak, and E. Jansen, "Self-Sensing Spaces: Smart Plugs for Smart Environments," presented at Third International Conference On Smart homes and health Telematic (ICOST), Sherbrooke, Canada, 2005.

[2] "The Gator Tech Smart House at the Oak Hammock Retirement Community at the University of Florida. Info & Virtual Tour available at: www.harris.cise.ufl.edu/gt.htm."

[3] D. Cook and S. Das, *Smart Environments: Technology, Protocols, and Applications*: John Willey & Sons, Inc., 2005.

[4] P. Chang and J. Krumm, "Object recognition with color cooccurrence histograms," *Computer Vision and Pattern Recognition*.

[5] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach* Prentice Hall, 2002.

[6] T. Gevers and A. W. Smeulders, "Color-based object recognition " *Pattern Recognition*, vol. 32, pp. 453-464, 1999.

[7] C. Papageorgiou and T. Poggio., "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, pp. 15–33, 2000.

[8] A. Torralba, K. P. Murphy, W. T. Freeman, and M. Rubin, "Context-based vision system for place and object recognition," presented at Intl. Conf. Computer Vision, 2003.

[9] Y. Tsin, R. T. Collins, V. Ramesh, and T. Kanade, "Bayesian Color Constancy for Outdoor Object Recognition," presented at IEEE Conference on Computer Vision and Pattern Recognition, 2001.

[10] A. Sixsmith, "Pervasive Computing and the Well-Being of Older Persons," presented at ICADI, 2003.

[11] S. Giroux, A. Ayers, and H. Pigot, "An Infrastructure for Assisted Cognition and Telemonitoring," presented at ICADI, 2003.

[12] L. Kecheng, R. Clarke, and P. Andersen, *Organizational Semiotics: Evolving a Science of Information Systems*, vol. 227. Montreal: Kluwer, 2002.

[13] A. Helal, W. Man, H. Elzabadani, Y. Kaddoura, E. Jansen, and J. King, "Gator Tech Smart House: A Programmable Pervasive Space," *IEEE Computer magazine*, pp. 64-74, 2005.

[14] A. Helal, "Programming Pervasive Spaces," *IEEE Pervasive Computing magazine*, vol. 4, 2005.

[15] K. Romer, T. Schoch, F. Mattern, and T. Dubendorfer, "Smart Identification Frameworks for Ubiquitous Computing Applications," presented at First IEEE International Conference on Pervasive Computing and Communications, 2003.

[16] Y. Kaddoura, A. Helal, and J. King, "Cost-Precision Tradeoffs in Unencumbered Floor-Based Indoor Location Tracking," presented at Third International Conference On Smart homes and health Telematic (ICOST), Sherbrooke, Canada, 2005.