Controlling Camera and Lights for Intelligent Image Acquisition and Merging

Olena Borzenko, Yves Lespérance, Michael Jenkin
York University
Department of Computer Science and Engineering
4700 Keele Street, Toronto, ON M3J 1P3, CANADA
{olena, lesperan, jenkin}@cs.yorku.ca

Abstract

Docking craft in space and guiding mining machines are areas that often use remote video cameras equipped with one or more controllable light sources. In these applications, the problem of parameter selection arises: how to choose the best parameters for the camera and lights? Another problem is that a single image often cannot capture the whole scene properly and a composite image needs to be rendered. In this paper, we report on our progress with the CITO Lights and Camera project that addresses the parameter selection and merging problems for such systems. The prototype knowledge-based controller adjusts lighting to iteratively acquire a collection of images of a target. At every stage, an entropy-based merging module combines these images to produce a composite. The result is a final composite image that is optimized for further image processing tasks, such as pose estimation or tracking.

1. Introduction

Remote video cameras have found a wide range of applications. They are used to guide semi-autonomous mining machines and vehicles, dock craft in outer space, control endoscopes in medicine, etc. To overcome the poor lighting conditions common to such application areas, the remote camera is often associated with one or more (typically fixed) light sources that can be controlled. The task may be performed directly by a human operator, such as in the case of guiding an endoscope, or it may be performed by a software agent with or without human intervention. In either case, the operator manipulates the light conditions and camera parameters to appropriately illuminate and capture portions of the image that are critical to the task at hand.

Choosing an appropriate illumination is an extremely complex problem. Maximizing one illuminant may place portions of the scene in high relief, while at the same time casting shadows over other portions of the image that must be viewed in order to solve the required task. Interactions between the illuminants and gain control within the camera itself complicate the task even further. Perhaps the most common version of this problem is the lighting problem portrait photographers encounter. How should the scene be lit and the camera controlled in order for the camera to best capture the subject? What we mean by "best" depends significantly on the specific task at hand.

In the machine vision domain, the task becomes even more complex. To adequately capture the whole scene in one acquisition is often impossible. If one assumes that the scene is static, a solution is to obtain images taken under different illumination or camera conditions and then to combine them into a single composite image. (For instance, during space docking, spacecraft often “freezes” to make the scene static and thus localize a target precisely. To improve the performance of a vision algorithm or a human operator, a sequence of images may be taken and combined into a high-quality composite before further processing). Not only must the different illumination conditions be chosen in some intelligent manner, but some process is also required to render the combination. Choosing appropriate lights and camera parameters as well as presenting the resulting set of images as a composite to a user or further system component, are the two problems that we address within the Lights and Camera (L&C) project.

The primary problem we address here is the control problem: given an appropriate evaluation function that estimates how well or appropriately regions of the scene are illuminated in one or a collection of images, how should the various parameters of the lights and camera system should be controlled in order to capture all portions of the scene ‘sufficiently well’? A second issue is a rendering problem. Given the collection of images obtained under different illumination conditions, how should the images be combined in order to produce a single image that can be presented to a user or further system component for later processing?

To address the high-level decision tasks of capture control and rendering, we have developed a prototype knowledge-based controller, programmed in the IndiGolog (IG) agent programming language[8]. The controller maintains high-level knowledge of how to interpret numerical image evaluation results, how to improve the image capture, and how to determine whether enough different lighting conditions have been captured. For combining images obtained under varying lighting conditions as well as for evaluating the amount of detail of the image, we have adopted an entropy-based approach (see [6] for details of the approach and [9] for a more general introduction to entropy-based image merging.)

The Lights and Camera project is a collaboration between researchers at York University and with our industrial partner, MD Robotics (MDA Space Missions). The docking latch target (Figure 8) and the fastener from the Hubble telescope service box (Figure 1) that we use in our experiments were contributed by MD Robotics.

The rest of the paper is organized as follows. In Section 2, we discuss related work on image acquisition and merging as well as knowledge-based control for robotics and vision. Section 3 gives an overview of the IndiGolog agent programming language and its applicability to the problem of adaptive control for image acquisition. The Lights and Camera testbed is described in Section 4. The next two sections zero in on the main components of the testbed, namely, Section 5 discusses the implementation of our entropy-based merging module while Section 6 elaborates on the strategies adopted in the L&C intelligent controller. We then proceed with a discussion of the results obtained so far (Section 6), and conclude the paper with a summary (Section 7) and an outline of future research directions (Section 8).

2. Previous Work

There has been very little work done on the development of strategies for the controlled illumination of scenes prior to image capture. Perhaps the most advanced systems are based on Panasonic's Super Dynamic II technology. These systems combine two views from a single static camera (without controllable illumination) to overcome low and variable light conditions. The technology is quite impressive for it can be implemented within the camera itself or as part of larger system. Nevertheless, it does not address issues related to actually controlling the illuminants (it only deals with the existing illumination), nor does it propose particularly sophisticated mechanisms for combining imagery taken under different illumination conditions (it only deals with combining parts of the image captured under different capture parameters).

Software based systems have been proposed where multiple views of the same scene are acquired using different camera settings and integrated into one high dynamic range (HDR) image [4]. The most popular approaches to HDR imaging are temporal exposure change [4] (when the camera and the scene are static) and generalized mosaicing [1] (when the imaging system rotates, but the scene remains constant). All these systems operate using a "brute force" approach where images are acquired under all possible settings. Although these methods reduce dependency on the level of external lighting, they still assume adequate external illumination and do not address the problem of illumination control. To combine acquired images into a composite, several hardware and software-based methods have been developed, among them recovering camera's response function [4], the contrast-compression method [18], and entropy maximization [9], [6] (also see [6] for a detailed discussion of image merging techniques).

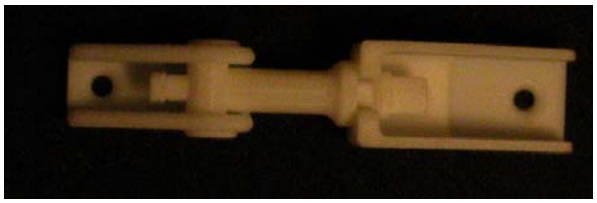

Figure 1. Space Component Illuminated with a Controllable Light

There has been significant research concerning the use of AI/knowledge-based techniques for vision and robotics control. Several mobile robot platforms equipped with suites of sensors have been endowed with AI/reasoning capabilities and demonstrate these by conducting museum/lab tours, mail delivery, etc. AI systems have also been deployed in space. Deep Space-1 (DS-1), launched in 1998, was the first spacecraft to use AI software to coordinate all the hardware and software activities of different parts of the spacecraft [12]. The executive "AI kernel" was written in Lisp and in addition to DS-1, is being used for satellite telecommunication and space-based interferometers. The agent programming language Golog was used in a very successful museum guide robot developed in Germany [1]. Several approaches to adaptive and knowledge driven operation of vision systems have been proposed. For example, Shekah et al. [16] packaged basic image processing algorithms into multi-layer "smart" modules that encapsulate knowledge of how to run the associated algorithm, evaluate its results, and tune its parameters. First, vision tasks are decomposed into a hierarchical plan. Then rules expressing possible approaches, constraints, adjustment and failure handling are used to complete the plan. Automatic solving of a specific problem entails the selection and ordering of modules, running them, and tuning parameters on training images. Robertson [15] uses the concept of procedural reflection to allow monitoring and modification of the computational state of an image processing filter.

The adaptive operation of the vision system, incorporating the adaptable filters, is comparable to the control of a closed-loop system.

To address the Lights and Camera control problem, we build on the results of a previous project on High-Level Control of Vision Sensing Systems. In the previous project, we developed a testbed for studying high-level control of vision systems, and designed two prototype high-level controllers for vision systems involving multiple vision processing modules with adjustable parameters (one for the testbed and one for a vision system belonging to MD Robotics). The testbed controller was built using IndiGolog and a small library of high-level control components was developed.

3. IndiGolog

Our approach to the development of high-level controllers for vision systems is based on the use of the high-level model-based agent programming language IndiGolog [1], a successor of Golog [10] and ConGolog [16].

In these languages, the programmer provides a declarative specification of the domain – actions and their preconditions and effects on fluents, i.e. the dynamic aspects of the state – and develops complex control programs in terms of these. Programs can also generate new plans by searching for a sequence of actions that will achieve certain goals. ConGolog adds concurrency, priorities, and interrupts to Golog, which makes it easier to write reactive controllers. IndiGolog extends ConGolog to support the inclusion of planning/search components within an overall deterministic program that is to be executed incrementally in conjunction with sensing of the environment. The IndiGolog planning/search mechanism automatically monitors the execution of the generated plan and re-plans when the current plan is no longer appropriate due to sensed changes in the environment. The mechanism also supports a simple form of contingent planning where the dynamic environment is modeled as a simple deterministic reactive program [4]. An agent programming language such as IndiGolog is an ideal basis for developing high-level controllers for vision systems. Unlike expert-system shells, it is intended for dealing with dynamic situations. It supports both planning and high-level reactivity, in contrast to classical planners on the one hand and purely reactive architectures on the other. The York IndiGolog interpreter that we use in the project is implemented in Quintus Prolog and SWI Prolog.

4. Lights and Camera Testbed

The L&C testbed consists of a controllable camera and three controllable lights that are used to iteratively acquire a collection of images of a visual target. The adjustable parameters are:

- intensities for the 3 lights, ranging from 0 to 7 (8 values each),
- shutter speed for the camera (15 values).

So how many different combinations of images can be constructed of a single static scene using this relatively simple and limited imaging platform? Taking into account the fact that we consider image combinations rather than single images, the search space size for the task is

$$2^{8 \cdot 8 \cdot 8 \cdot 15} = 2^{7680} > 8 \cdot 10^{2311}.$$

That is, there are more than 75 billion possible image combinations even with as few as three images in a given set. Clearly, it is not possible to try every possible set of images. There is a need for heuristic parameter selection that adapts to already acquired images and lighting conditions inferred from them.

The software architecture of the Lights and Camera system consists of the Image Server, the Merging Module(s) and the high-level Intelligent Controller (see Figure 2). Communication between different modules occurs by message passing via TCP/IP sockets. Each message is encoded as an ASCII string which consists of words separated by empty space characters. For example, in order to change the value of an adjustable parameter, such as the intensity of the first light, the Controller sends a "set" message which contains the new value of the parameter that should be used by the Merging Module; whenever a module receives a "set" message, it sends back an "ok" message to notify the sender.

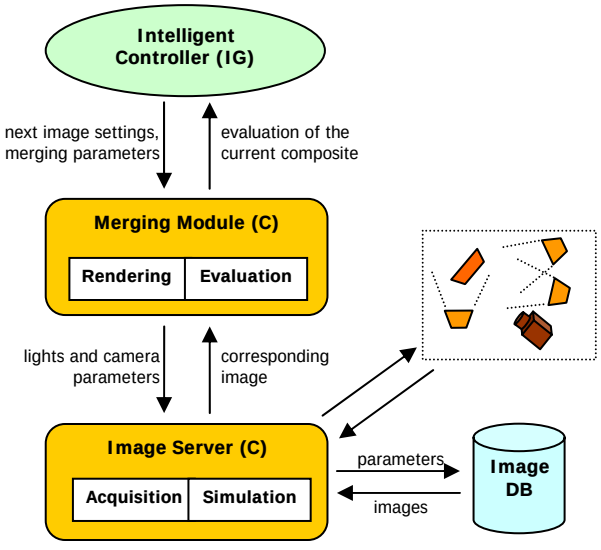

Figure 2. Lights and Camera System Architecture

The Image Server, implemented in C, operates a digital camera and 3 lights, changing values for parameters such as light intensities, camera aperture, shutter, and

focal length. It acquires images from the camera and stores them along with the description of conditions under which they were taken. Stored images can also be used in simulation mode, when no actual acquisition takes place, but previously stored images are made available to other modules. This helps separate the acquisition and merging stages to facilitate experimentation.

The Merging Module implements an image merging algorithm and an evaluation measure of image goodness. Following a request from the Controller as to what the next light and camera parameters are, it queries the Image Server for the corresponding image. It adds the new image to the existing collection of images taken under different operating conditions, renders a composite, and estimates the amount of detail in the composite in terms of entropy. The result is then sent to the Controller, closing the feedback loop. According to the decision taken by the Controller module, the current image is either kept in or discarded from the composite.

The task of the L&C Intelligent Controller is to optimize the final composite image for further image processing tasks, such as pose estimation or tracking. The control task (see Figure 3) is done by selecting a parameter configuration, getting a new image, adding it to the image set, evaluating the updated composite, and deciding whether its quality improved compared to the previous step - if not, the last image is marked as discarded and is replaced at the next iteration. This is repeated until the desired quality or maximum number of iterations is reached.

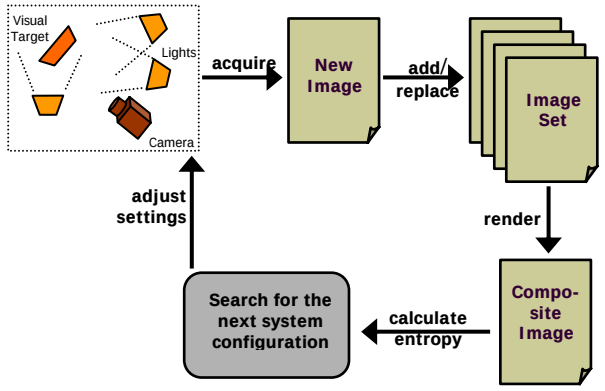

Figure 3. Image acquisition and rendering

5. Image Merging

Merging a set of images taken under varying lighting conditions and camera parameters into a single image, requires the development of a system through which the "best" elements in each source image can be accurately identified. These elements must then be merged and blended into a final image that is a "best" (under some metric) representation of the collection of images.

Currently we use entropy to measure the amount of detail provided by each picture [9]. The entropy is defined as the number of bits necessary to represent a given input given the probability of that input appearing in a stream. High entropy indicates large variance in the given pixels, while low entropy indicates that the pixels are fairly uniform, and hence little detail can be derived from them. Therefore, when applied to groups of pixels within the source images, entropy provides a way to compare the same element from the different source images and decide which provides the most detail.

A set of images is merged into a single image using a local entropy-weighting method [6]. This approach allows a collection of images to be collapsed into a single image in an efficient manner. The entropy of the resulting merged image is used as an estimate of its quality.

The entropy-based evaluation and merging method is implemented in our system as a separate module. It is used by the L&C controller to combine sets of images and evaluate resulting composites. For a detailed discussion of our approach to image merging, see [6].

6. Intelligent Controller

Given the current conditions of the environment, such as natural lighting or lack thereof, the task of the intelligent controller is to acquire and merge a collection of images under such system conditions that their composite is optimized for further image processing. The task is complicated by the size of the search space as well as the complexity of the effects that changes in the system parameters produce on the image content.

To develop good heuristics for parameter selection, we conducted experiments where we merged pairs and triples of images and compared the entropy of the individual images with that of the corresponding composite. The experiments showed that combining images with similar lighting does not generally lead to substantial increases in entropy. However, merging sets of images taken under somewhat different light conditions yields composites that exhibit adequate entropy gains. The intuition is that varying lighting conditions usually bring up new features that were not previously noticeable in the image, thus contributing to the overall level of detail. At the present stage of the project, we only use one shutter speed for all images.

Based on these empirical results, we designed a simple model to estimate the similarity of images in terms of the common information that they might share (see Figure 4). The idea is to divide the hyperspace of light settings into subspaces in such a way that images from one given group carry approximately the same information. By reasoning about the similarity of the information in images we reduce the dimensionality and size of the search space.

```
 % image categories
brightness(low,L1,L2,L3):-
                cumulative_intensity(I,L1,L2,L3),
                is_low_intensity(I),!.
brightness(high,L1,L2,L3):-
                cumulative_intensity(I,L1,L2,L3),
                is_high_intensity(I),!.
brightness(medium,L1,L2,L3):-
                cumulative_intensity(I,L1,L2,L3),
                not is_low_intensity(I),
                not is_high_intensity(I).
directionality(even,L1,L2,L3):- L1 = L2, L2 = L3,!.
directionality(left,L1,L2,L3):- L1 >= L2, L1 > L3,!.
directionality(right,L1,L2,L3):- L3 >= L2, L3 > L1,!.
directionality(middle,L1,L2,L3):- L2 >= L1, L2 >= L3,!.
directionality(sides,L1,L2,L3):- L1 > L2, L1 = L3.

 % similarity of two images
same_brightness([L1,L2,L3],[N1,N2,N3]):-
                brightness(Brightness,L1,L2,L3),
                brightness(Brightness,N1,N2,N3).
same_directionality([L1,L2,L3],[N1,N2,N3]):-
                directionality(Directionality,L1,L2,L3),
                directionality(Directionality,N1,N2,N3).
img_similarity(Img1,Img2,1.0):-
                same_brightness(Img1,Img2),     % and
                same_directionality(Img1,Img2),!.
img_similarity(Img1,Img2,0.5):-
                same_brightness(Img1,Img2);     % or
                same_directionality(Img1,Img2),!.
img_similarity(Img1,Img2,0.0).
```

Figure 4. Definition of image similarity (Prolog)

As seen in Figure 4, we based the categorization on overall brightness (low, medium, high) and directionality (left, right, middle, even, sides) of the light. Specifically, an image has even directionality of light if the intensities of all three lights are the same, etc. We give the highest similarity value of 1.0 to a pair of images if they have the same directionality and brightness of light. If the images share only the same directionality or the same brightness, we assign a medium level of similarity to the pair. Finally, if the images do not fall under the same category for either brightness or directionality, we say they are *not similar* and use the value of 0.0.

We distinguish lighting distributed evenly and that coming mostly from the middle light. While the benefits of such a categorization may not be evident for planar targets facing the camera, the distinction becomes especially important for more complex, three-dimensional targets, such as a docking latch, for which the three lights may influence the visibility of disjoint sets of features.

The degree of similarity of a new image to the existing set of images is calculated by averaging similarity values for the new image and each of the images in the existing set. The new image is said to be *not similar* to the set if the calculated degree of similarity is below a specified threshold. In an analogous fashion, we calculate how similar the new image is to the set of already discarded images. We maintain separate thresholds for similarity to the composite image set and similarity to the set of discarded images, though in the current controller we use the same value for both thresholds.

The controller keeps track of the images that were added to or discarded from the composite by storing the corresponding parameters and entropy evaluation in IG *fluents* (dynamic predicates). For example, the value of the `imgsetPara(entropy,k,light1)` fluent is the intensity of the first light that was used to acquire the k-th image for the composite in the entropy-based merging module. To be able to use fluents, one should declare and initialize them. When the control program is running, the values of fluents can be changed by actions which become possible under certain conditions. For example, the values for `imgsetPara` and `imgsetResult` fluents are only updated when the Merging Module has confirmed the addition of a new image and calculated the new entropy of the composite. In addition, a fluent can be used to specify the effects of actions or to form definitions of complex conditions.

The high-level control procedure for image acquisition and rendering is given on Figure 5.

```
proc(runController(Host, Port),
    [ %%% captures and merges a set of images,
      % optimizing the composite

    connectTo(merging_module, Host, Port),

    initialize(composite),

    while(and(insufficient_entropy(composite),
              is_not_full(imgset)),
      [
      adjust_composite(entropy)
      ]),

     disconnectFrom(merging_module)
]).
```

Figure 5. Main Control Procedure (IndiGolog)

At the first step of the algorithm, the settings for the initial image are picked at random. The agent then iteratively chooses light settings for the next image acquisition, looking for images that would contribute sufficient additional information with respect to the existing image set (see Figure 6).

```
proc(adjust_composite(entropy),
    [   %%% adds/replaces an image
        % to improve overall entropy

    add_img(imgset, not_similar),

    if(insufficient_gain(entropy, composite),
        discard_img(imgset),
    % else
        confirm_img(imgset))
    ]).
```

Figure 6. One Iteration of the Feedback Loop

If the entropy of the composite did not increase sufficiently due to the newly added image, the image is

marked as discarded, parameters under which it was taken are marked as unfavourable, and the search for a new addition to the set is initiated.

When searching for the best additional image, the controller restricts its choice to the images that are not similar to the set of images added to the current composite as well as to the set of images discarded at earlier steps. The main part of the `add_image(imgset, not_similar)` procedure is the search for new light settings that would produce such an image (see Figure 7). First, the controller picks some settings at random and sets them as current. Then, it computes the degree of similarity of this image to each of the image sets. Finally, it decides whether the image is similar to either of the image sets by evaluating the complex conditions `is_not_similar(Set, current_img)`. If the current image does not satisfy the condition of non-similarity, the procedure is repeated until an acceptable image is found.

```
proc(find_img(imgset, not_similar, [L1,L2,L3]),
    [ %%% searches for an image
      % that is not similar to the existing
      % and discarded image sets
    pi([l1,l2,l3],              % pick settings
      [
      ?(random_img(l1,l2,l3)),  % at random
      set_lights(l1,l2,l3),
      compute_similarity(imgset,current_img),
      compute_similarity(discarded,current_img),
                                % test similarity
      if(and(is_not_similar(imgset, current_img),
             is_not_similar(discarded, current_img)),
        [                       % if acceptable,
                                % return result
        ?(L1 is l1), ?(L2 is l2), ?(L3 is l3)
        ],
      % else            % continue search otherwise
        [
        find_img(imgset, not_similar, [L1,L2,L3])
        ])
      ])
    ]).
```

Figure 7. Search for Images that are `not_similar`

In evaluations of whether a complex condition holds, the controller refers to the current values of fluents specified in the declaration of the condition. For example, in order to evaluate the `is_similar(Set, current_img)` complex condition, the controller checks if the value of the `similarity(Set, current_img)` fluent is less than or equal to the `low_similarity(Set)` threshold for a given set.

Augmenting the composite in the main control loop continues until a maximum number of images has been added to the image set or the entropy of the current composite reaches a predetermined level.

As the project progresses, we intend to vary the value of shutter speed in addition to controlling light intensity levels. This poses the question of how to adjust shutter speed in an optimal way from acquisition to acquisition. Presumably, the agent would still be able to base its reasoning upon the categorization of images described earlier. For example, it may notice that most of the images that significantly contributed to the composite were captured under highly bright light. The agent may conclude that the bright light revealed object features otherwise invisible under the current camera settings. In this case, it may decide to use lower shutter speed for later image acquisitions. The resulting images may enhance these object features in the composite and even introduce new ones. Experimentation with image acquisition and merging across a range of shutter speed values will give grounds for the final design of parameter searching scheme.

7. Preliminary Results

The average running time for the control algorithm for image acquisition and merging varied between 4 and 6 minutes (in simulation mode, using pre-acquired images). In general, the higher we set the values for sufficient composite entropy, for minimum required entropy gain for each step, or for image similarity threshold, the longer it took the controller to run.

Below we illustrate a sample run of the control algorithm, aiming at the entropy of 13.8 and using the similarity threshold values of 0.5 and minimum entropy gain of 0.2.

First, the initial light settings are picked at random and the first image is acquired (see Figure 8). Then the controller searches for an image that would be sufficiently different from the already existing one. It finds the new settings “5, 1, 6” (intensity of left, middle, right lights) and acquires a corresponding image (Figure 9), which has the degree of similarity of 0.5 (it shares the same brightness level as the initial image, but not the directionality of light) and is, therefore, acceptable. After the two images are merged, the evaluation of the composite shows an adequate entropy gain. Therefore, the second image is kept in the set (see Figure 9).

In the next step, the controller performs yet another search for an image that could potentially improve the entropy of the existing set. Its choice appears on (Figure 10). When this third image is added, the entropy of the composite actually goes down (which in this particular case can be expected based on mere visual inspection). Therefore, the new image is discarded. The same happens for the next choice of an image “1, 3, 8” which has with medium intensity and right directionality of light, and it is also discarded.

However, the next image with light settings “3, 6, 2”, when added to the image set, indeed improves the composite entropy compared to its previous level (see Figure 11). In fact, the entropy of the composite reaches the

Figure 8. Initial image is captured. Lights: 5, 4, 1, directionality: left, brightness: high.

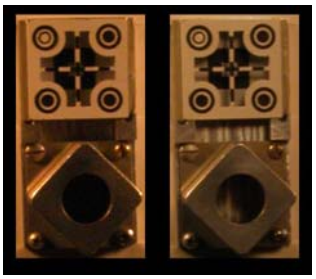

Figure 9. Next image (on the left) is added. Lights: 5, 1, 6, directionality: right, brightness: high. The new composite (on the right) has higher entropy than the original.

Figure 10. The third Image (on the left) is added. Lights: 0, 2, 0, directionality: middle, brightness: low. Corresponding composite (on the right) has not improved. Image is marked as discarded.

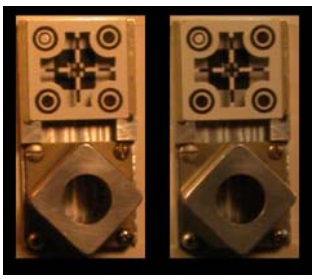

Figure 11. The last image (on the left) replaces the discarded one. Lights: 3, 6, 2, directionality: middle, brightness: high. Entropy of the final composite (on the right) has reached the desired level.

required level of 14.0. Hence, the control algorithm does not look for further improvements.

We have been performing tests to study the controller's performance and find ways to improve it. When running the experiments, we also paid attention to how well entropy performed as a measure of image quality. It appears that, for image processing tasks such as edge-based model-matching, the entropy of a composite image may not always be the best evaluation measure. For example, it would assign higher scores to composites on which a shadow was present simply because that enriched the information content of the images.

8. Conclusion

Developing techniques that provide robotic vision systems with autonomy is very important for many applications, for instance for space robotics. In this paper, we have described a specific problem where intelligent control can be used: acquiring multiple images of a target under different lights and camera parameters so that a high-quality composite image can be rendered. We have proposed an approach to solving the problem based on characterizing the information in regions of images in terms of entropy. We have described an overall architecture for a system that performs this task. One key component of our system is an intelligent controller implemented in the agent programming language IndiGolog. We have described the main features of this controller as well as given examples of the knowledge that it uses in making decisions. We have also described some of the early system test results.

Clearly, having a methodology and tools that support the design of such intelligent autonomous controllers is a very important objective for robot vision. We need techniques that allow important domain knowledge to be represented in a perspicuous and easily extended way, so that it can be reasoned with, and exploited in making control decisions. Our work on the Lights and Camera control system is part of an effort at developing an IndiGolog-based toolkit for designing intelligent vision application. This toolkit will include a library of vision algorithms, search methods, and control frameworks that will speed up development. We believe that such techniques and tools would be an important contribution to the field.

9. Future Work

One of the directions for future work is to explore various control strategies for parameter selection. This will include designing a more complex model to estimate the amount of shared information in two images.

Another direction for future research is the search for a better evaluation measure of an image. In fact, we are already developing an edge-based model-matching mod-

ule to estimate the pose of a docking latch given the current composite. We intend to use the certainty of match returned by the module as a new evaluation measure. We will also look at using a combined entropy/match quality measure.

In the case of a highly autonomous vision system, it may be beneficial to adjust the definition of concepts in the knowledge base of a controller. Specifically, the controller could use a reinforcement learning framework, such as in [17], to automatically correct definitions of brightness or directionality as well as entropy threshold values in order to adapt to varying environment conditions or a different target.

It is unlikely that a single control algorithm will be appropriate for all possible camera and light situations, so we are also working on a set of tools for the development of high-level controllers for the application.

Acknowledgements

We would like to acknowledge Wei Xu, Mark Obsniuk, and Arjun Chopra for helping to create and maintain the experimental testbed as well as for their ongoing work on the project, Andrew German for his work on image merging, and Nava David for creating the experimental image database. The financial support of OCE/CITO, NSERC Canada, and MDA Space Missions is gratefully acknowledged.

References

[1] Aggarwal, M.; Ahuja, N., "High dynamic range panoramic imaging", *Proceedings of the Eighth IEEE International Conference on Computer Vision*, Volume 1, pp. 2-9, July 2001.

[2] Brajovic, V., "Brightness Perception, Dynamic Range and Noise: A Unifying Model for Adaptive Image Sensors", *CVPR04 (II)*, pp. 189-196, 2004.

[3] Burgard, W., Cremers, A. B., Fox, D., Hahnel D., Lakemeyer, G., Schultz, D., Steiner W., and Thrun, S., "The Interactive Museum Tour-Guide Robot", *Proceedings of AAAI-98*, pp. 11-18, 1998.

[4] Debevec, P. E. and Malik, J., "Recovering High Dynamic Range Radiance Maps from Photographs", *Proceedings of SIGGRAPH 1997*, ACM Press / ACM SIGGRAPH, pp. 369-378, 1997.

[5] Frieden, B. R., "Restoring with Maximum Likelihood and Maximum Entropy", *J. Optical Society America*, vol. 62, pp. 511–518, 1972.

[6] German, A., Jenkin, M., Lespérance, Y., "Entropy-based image merging". *Proceedings of the Second Canadian Conference on Computer and Robot Vision*, Victoria, BC, May 2005.

[7] De Giacomo, G., Lespérance, Y., and Levesque, H.J., "ConGolog, a concurrent programming language based on the situation calculus", *Artificial* Intelligence, 121, 109-169, 2000.

[8] De Giacomo, G., and Levesque, H. "An incremental interpreter for high-level programs with sensing", in Levesque, H. J., and Pirri, F., eds., *Logical Foundations for Cognitive Agents: Contributions in Honour of Ray Reiter*, Berlin: Springer, pp. 86-102, 1999.

[9] Goshtasby, A. A., "High Dynamic Range Reduction via Maximization of Image Information", http://www.cs.wright.edu/~agoshtas/hdr.html, 2003.

[10] Jasiobedzki P, Anders C, "Computer Vision for Space Robotics, Applications, Role, and Performance", SPRO '98 I IFAC Workshop on Space Robotics, Canadian Space Agency, pp. 96-103, 1998.

[11] Levesque, H. J., Reiter, R., Lespérance,Y., Lin, F., and Scherl, R. B., "GOLOG: A Logic Programming Language for Dynamic Domains, Journal of Logic Programming", 31, 59-84, 1997.

[12] Nayar, S. K., Branzoi, V., "Adaptive Dynamic Range Imaging: Optical Control of Pixel Exposures over Space and Time", *ICCV03*, pp. 1168-1175, 2003.

[13] Pal, C., Szeliski, R., Uyttendaele, M., Jojic, N., "Probability models for high dynamic range imaging", *CVPR04* (II), pp. 173-180, 2004.

[14] Pell, B., Bernard, D.E., Chien, S.A., Gat, E., Muscettola, N., Nayak, P.P., Wagner, M.D. and Williams, B.C., "An Autonomous Spacecraft Agent Prototype", *Autonomous Robots*, 5(1), March, 1998.

[15] Robertson, P, Brady, M., "Adaptive Image Analysis for Aerial Surveillance", *IEEE Intelligent Systems*, pp.30-36, May/June 1999.

[16] Shekhar, C., Moisan, S., Vincent, R., Burlina, P., and Chellapa, R., "Knowledge-based control of vision systems", *IVC* 17, pp. 667-683, 1999.

[17] Taylor, G. W., "A Reinforcement Learning Framework for Parameter Control in Computer Vision Applications", *Proceedings of the First Canadian Conference on Computer and Robot Vision*, London, ON, May 2004.

[18] Tumblin, J. and Turk, G., "LCIS: A Boundary Hierarchy for Detail-Preserving Contrast Reduction," *Proceedings of SIGGRAPH*, ACM Press / ACM SIGGRAPH, 83-90, 1999.