Robotic Vision and Graphical Display Based on the IBM PC

M. Feeley and N.F. Stewart

INTRODUCTION

In this paper we shall discuss the design of a robotic vision system, based on the IBM Personal Computer, and intended to treat input from one or more TV cameras. The system will eventually form part of a larger system for the treatment of data from various sensors, including the TV cameras. These systems replace the vision part of a robotic system which was described in an earlier paper by the second author alone [22], and documented in detail in the reports[5,6,7,9, 10,11,15,16]. The purpose of the vision system is to permit development and testing of image processing algorithms for robotics applications.

System organization and available hardware with be discussed first, followed by a description of the proposed software. Design and programming of the image processing software has only just begun, as part of a project course [6]. Since the purpose of the system is development and testing of algorithms, it is expected that the software will be continually changing.

SYSTEM ORGANIZATION AND AVAILABLE HARDWARE

The robotic system described in [22] involves two small experimental arms with controllers, a console, and a vision system, connected to a Host computer (an LSI 11/23). It was mentioned there that the preliminary vision system (a 96 x 128 frame-grabber with 8 gray levels, sharing 12K of RAM with an Intel 8086) would be replaced by an iAPX 88/20 system. In fact, the vision system described here will be based on an IBM PC (to be delivered March 83) with the 8087 co-processor installed, and this constitutes an iAPX 88/20 system with high level programming and disk operating system support. The PC has 128K RAM, and there is room on the memory card to expand to 320K.

A Panasonic WV-1350-A vidicon camera is available for use in the vision system, and two Hitachi KP-120 Solid State cameras, for use in experiments in stereoscopic vision, have been ordered. It will be possible to use the two Hitachi cameras together, or the vidicon camera alone, by using both or only one of the two frame grabbers inserted in the PC. We decided to build two frame grabbers, rather than sharing one between two cameras, in order to be able to digitize simultaneous views of the scene when we are using the two Hitachi cameras together. (The alternative of sharing the framegrabber requires that the image from the first camera be moved before digitization of the second image begins. However, to transfer the contents of a digital image from the frame grabber memory to the ordinary memory in the PC takes well over 100 msec, which is too large a delay if the scene is changing, even slowly.) Each frame grabber is based on a flash chip and 82K of static RAM, and digitizes the standard video signal produced by each of the three cameras mentioned above. The digitized image contains 256x320 pixels, of which only 244 rows contain useful picture information (the other rows contain video synchronisation information). Each pixel has 64 gray levels, requiring 6 bits per pixel, which are stored 6 bits per byte. The entire image requires therefore 256x 320 = 81,920 bytes of memory, which is directly addressable by the PC.

The frame grabbers were built using static RAM in order to minimize design and construction time; however, as a consequence, they are quite bulky. In fact, each frame grabber takes up one of the five slots of the IBM PC, and at that, it was necessary to piggy-back the memory chips to house them all on a single full-size board. (With the release of the 8KX8 static RAM chips in summer 1983 it will be possible to build a double frame grabber with 164K static RAM using only a single slot, but we wanted to begin immediately; moreover, since the memory board purchased with the computer also contains serial and parallel I/O ports, using two slots for frame grabbers is not an inconvenience.)

The frame grabber digitizes the image by columns so that any 256 columns of the 256 x 320 image fit into a contiguous 64K segment of memory. Thus, the whole image is available to the programmer, but programs using only a 256 x 256 window run very efficiently. (Moreover, it is easy to maintain program compatibility if the camera is replaced by one with 256 x 256 digital output, such as the GE TN 2500 CID camera, provided that

the image is stored by columns.)

The design of the two frame grabbers is identical, and construction of the first of the two is fairly well advanced (it should be complete by the end of March). The design was done in our laboratory by M. Jacques, with an important contribution by G. Hurteau.

The reasons for choosing the IBM PC, as a development system for vision algorithms, were three-fold. Firstly, we want to be able to do fast floating point arithmetic, in order to be able to implement, for example, methods involving integral convolution [13] or principal component calculations [4]. The IBM PC executes a floating point multiply in 19 μsec., which is somewhat slower than the PDP-11/34, (and slower than the MC68881 announced for late 1983 [23]), but considerably faster than the LSI-11/23 with the KEF11-AA floating point option [2, Appendix B].

A second reason for choosing the PC was cost. We wanted an inexpensive system which we could afford, and also, which can plausibly be transferred to the factory floor. Of course, we are interested in studying some algorithms which can not be executed in real time on the PC, such as algorithms for the stereo matching problem [19], and for such problems it will be necessary to transfer the images to a CDC Cyber computer for off-line processing. However, in robotics, it is also important to see what can be done on a small system acceptable in an industrial context [24].

A third reason for our choice is that we have used (and are using) another IBM PC (with the 8087) for the development of the robot controllers [15], with very satisfactory results.

As described in the next section, the software will be written using IBM's Pascal and Macro-assembler. This is satisfactory for the treatment of vision input. Sending the signal to initiate a grab can be done from an assembly language procedure or from IBM Pascal, as can the interrogation of the flag set by the frame grabber when the digitization is complete. Critical sections of algorithms can be written as assembly language procedures. On the other hand, in the context of input from a larger variety of sensors (for example, we have two Leuze position detectors and a Polaroid range finder), it would be much more attractive to use an interrupt based system. We have some experience with Modula-2 [11,25], a high level (Pascal extension) language permitting interrupt handling, and our tentative plan is to base an Integrated Sensory System on an LSI-11/02 using Modula-2, running under the RT11 Operating System. The ISS would be connected to the host computer by a 9600 bps serial link, and would handle sensory information from various sensors, amongst them the PC based Vision System. In this way we can essentially write all of the software in Pascal, while still having an interrupt based system when it is needed, and fast floating point when it is needed.

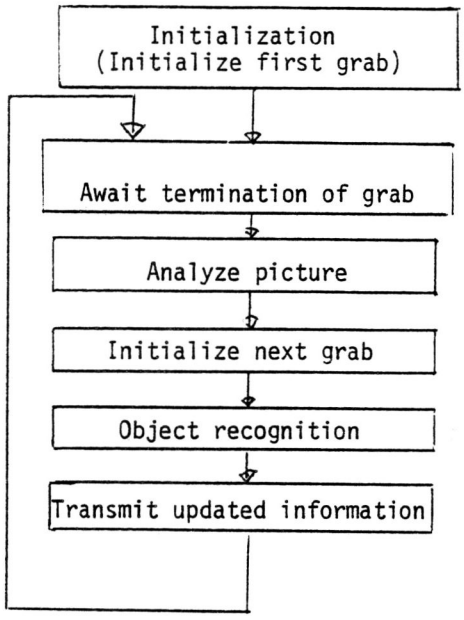
SOFTWARE AND HOST VISION SYSTEM PROTOCOL

We are primarily concerned with the case when the system has a great deal of prior knowledge about the likely contents of the scene being viewed [14]. This is consistent with, and to some extent a consequence of, our decision to study systems suitable in an industrial context. For the same reasons, at least the first versions of the software will be restricted to a very small set of object types, and will be based on simple but fast line-finding schemes.

The software for processing visual information will in fact be divided between the Host and the Vision System. The Vision System takes care of the digitization of the image, analysis of the image, and recognition of objects in the scene. The Host uses this information, sent by the Vision System, to update its internal description of the environment. Such information can be used, for example, to calculate the relative position of objects for use in object avoidance algorithms [17].

The system referred to above, being implemented as a course project [6], will be based on a Hough transform line-finding method [21], with extrapolation of the path of moving objects in time, in order to avoid reprocessing of the entire image at every step by tracking moving objects. In fact, a window will be associated with each object identified, and only the windows likely to contain an object will be reanalyzed at each step.

The system will normally be in an infinite loop in which it digitizes and analyses the image, recognizes objects, and transmits updates of the scene description to the host, breaking this loop only to execute instructions from the Host:



Note: Treatment of commands from the Host is done concurrently, by means of a polling process (i.e. periodic calls to a procedure which processes such commands).

Amongst the commands accepted from the Host will be:

FIXED <object name>
MOBILE.CR <object name>
MOBILE.CI <object name>

The vision system itself identifies the objects in the scene. The purpose of the FIXED command is to permit the Host to signal to the Vision System that it knows about a certain object, that the object is not going to move, and that therefore it need neither track the object nor consider sending an update about the object's position. (For example, the squares of a chess board.) The other possibility is that the object is mobile, but here there are two subcases, depending upon whether the object is Currently Relevant to what the Host is doing, or Currently Irrelevant. If the Host (perhaps temporarily) does not need information about the object, the command MOBILE.CI should be sent to the Vision System. The system will continue to track the object (and it must do this even if the object is always "currently irrelevant", in order to prevent its interpretation as a new object), but it will not send the information to the Host, thereby reducing the communication load.

Analysis of the image will involve extraction of feature lines by means of method based on the Hough transform and a digital approximation to the gradient, resulting in a list of feature lines [20,21]. The first version of the software is limited to a small class of recognizable objects, defined by a separate procedure for each type of object (cube, sphere,...). In later versions of the system we intend to include the possibility of a LEARN command, which would permit the Host to request the Vision System to describe any new objects which satisfy constraints specified in the command [1,12]. We also envisage a DESCRIBE command, which would return a low level description of the object.

The Host will maintain a list of objects in the environment, specifying the (unique) name of the object, type or size parameters associated with the object, object position, speed and orientation, and so on. We expect to be able to maintain synchronization between the two systems' clocks to within a fraction of a second.

As mentioned above, processing of the entire image at each cycle is avoided by tracking objects, and analyzing only the appropriate part of the image. However, in order to detect new objects appearing on the scene, the whole image must be sampled periodically.

EXTENSIONS

As mentioned above, our tentative plan is to extend the system by linking it with an LSI-11/12, available in our laboratory. Additional sensors available or ordered include two Leuze position detectors based on a modulated LED light source, with a filter for ambient light, a Polaroid ultrasound rangefinder, and a Hitachi VK-C2000 solid state colour TV camera. In addition, a laser range finder is being built by L.-P. Demers and A. Foisy as a student project [3]. It is not clear whether this device will have adequate precision, since it is being built entirely with equipment borrowed here and there, but even if not, it may at least serve as a prototype for a more accurate device. Finally, one of us [8] has with a colleague designed and implemented a graphical simulator for the 6E arms used in our system. The system runs on the CDC Cyber and permits wire-frame display of the two arms, and objects from a small class (cube, sphere), including the possibility of perspective changes. It is important in the context of off-line programming of robots to be able to simulate a taught sequence of moves before moving the manipulator itself [18], even if the simulation is slow. (In fact, slow-motion simulation is more desirable than actual speed simulation, provided that it is not too slow.) Implementation of such a simulator on a small system, which would be subordinate to the Host, appears to be feasible, but at present we do not have the necessary human resources to do so.

ACKNOWLEDGEMENTS

D. Fortin and G. Hurteau have collaborated closely on most aspects of the design of the system described and proposed here (but responsibility lies entirely with the authors). The work was supported in part in the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Capson, D., Pena, M. and Kitai, R. A silhouette vision system for robots. Proceedings, Canadian Conference on Robotics, Mississauga, 20-21 September, 1982, pp.73-84.

[2] DEC (Digital Equipment Corporation). Microcomputers and Memories (LSI-11 Handbook), 1981.

[3] Demers, L.-P. and Foisy, A. Laser rangefinder, IFT 6111 course project, Université de Montréal, 1983.

[4] Duda, R.O. and Hart, P.E. Pattern Classification and Scene Analysis, Wiley, 1973.

[5] Feeley, M. Transformations de coordonnées pour le bras SMART ARMS 6E. Document de travail #133, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[6] Feeley, M. Système d'analyse d'image. Document de travail #134, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[7] Feeley, M., Fortin, D. and Hurteau, G. Aubusson-expansion possible. Document de travail #138, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[8] Feeley, M. and Gagnon, F. Système d'émulation d'environnement robotique (Manuel d'usager et Manuel d'implantation). Unpublished manuscript, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[9] Fortin, D. Aubusson-Description du système. Document de travail #130, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[10] Fortin, D. Aubusson-Manuel d'usager et description technique. Document de travail #131, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[11] Fortin, D. Modula 2-Implantation au L.I.A. Document de travail #135, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[12] Geschke, C.C. A system for programming and controlling sensor-based robot manipulators. IEEE Transactions PAMI, Vol. PAMI-5, No. 1, 1983, pp. 1-7.

[13] Hildreth, E. Edge detection for computer vision system. Mechanical Engineering, August 1982, pp. 48-53.

[14] Horn, B.K.P. Derivation of invariant scene characteristics from images. AFIPS Conference Proceedings, Volume 49, 1980, pp. 371-376.

[15] Hurteau, G. RUM-Robot Controller University of Montreal. Document de travail #132, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1983.

[16] Hurteau, G. Turn your IBM PC into a real number cruncher. Document de travail #136, Dépt. IRO, Université de Montréal, 1983.

[17] Lozano-Perez, T. and Wesley, M.A. An algorithm for planning collision-free paths among polyhedral obstacles. CACM, Volume 22, No. 10, 1979, pp. 560-570.

[18] Kretch, S.J. Robotic Animation. Mechanical Engineering, August 1982, pp. 32-35.

[19] Nishihara, H.K. and Larson, N.G. Towards a real time implementation of the Marr and Poggio stereo matcher. SPIE Vol. 281, Techniques and applications of image understanding, 1981, pp. 299-304.

[20] O'Gorman, F. and Clowes, M.B. Finding picture edges through collinearity of feature points. IEEE Transactions on Computers, April 1976, pp. 449-456.

[21] Rosenfeld, A. and Kak, A.C. Digital Picture Processing, Second Edition, Volumes 1 and 2. Academic Press, 1982.

[22] Stewart, N.F. A multiple robot system with vision. Proceedings, Canadian Conference on Robotics, Mississauga, 20-21 September, 1982, pp. 85-90.

[23] Stockton, J.F. Growth of processor family boosts system options. Computer Design, February 1983, pp. 71-80.

[24] Warnecke, H.J., Schweizer, M. and Haaf, D. Programmable assembly with tactile sensors and visual inspection. Proc. First International Conference on Assembly Automation, 1980, pp. 23-32.

[25] Wirth, N. Modula-2. ETH Zurich, March 1980.