Parallelizing Tracking Algorithms

María Carina Roldan[1], Marcelo Naiouf [2], Armando De Giusti [3]
mcroldan@vittal.com.ar, {mnaiouf, degiusti}@lidi.info.unlp.edu.ar

LIDI. Laboratorio de Investigación y Desarrollo en Informática [4]
Facultad de Informática. UNLP

Abstract

In several applications, the trajectory of an entity, a feature or an object has to be tracked over a sequence of image frames. When the processing is to be performed in real time, there are important constraints leading to the parallelization of tracking algorithms.

This paper presents the results of a concrete implementation, which deals with the particular case of simple objects moving in an context reachable by the vision element (video camera). The steps involved in the solution development are detailed, specially in relation to their parallelization by using a computer heterogeneous network and MPI (Message Passing Interface) support.

Finally, an analysis of the different algorithms behavior is carried out together with the obtained results assessment, which allows knowing the performed parallelization efficiency, and determining under which conditions this solution turns out to be the best one.

Key words:
Parallel Algorithms. Tracking. Computer Vision. Heterogeneous Multiprocessors. MPI.

Introduction

There exists a growing interest in the processes automation based in the visual perception of the world around us, all of which requires images acquisition and processing. This is related to the field of Computer Vision [Har92][Jai95]; such systems intend to copy human vision processes into tasks of high complexity regarding exactness and time, or of excessively routine.

The objective of a computer vision system is to create a model of the real world from images, recovering useful information about a scene from its 2D projection. There exist developed techniques in other fields used for recovering data from images. Some of them are Image Processing (generally in the first stages in order to upgrade particular data and suppress noise) [Gon92][Bax94] [Jai89], Computer Graphics (where images are generated from geometric primitives), Pattern Recognition (classifying numeric and symbolic data) [Sch92], etc.

Computer vision systems encompass different applications, among which we can mention medical diagnosis by computed tomography images or product quality control, from food to industrial pieces [Law92]. Another type is related to an environment three-dimensional structure recovery, i.e. *scene movements recognition and interpretation*; these applications are useful in cars, planes or robots navigation. A third group includes the analysis and management of large data volumes obtained by satellites used for weather forecast, the study of the planet behavior, agriculture, etc.

Object tracking in an sequence of image frames is particularly associated to:
- Objects recognition and classification.
- Tracking of the same objects through that sequence.

---

[1] Graduate in Computer Sciences Licenciature.
[2] Full-time Associate Professor.
[3] Principal Researcher of CONICET. Full-time Chair Professor.
[4] LIDI – Faculty of Computer Sciences. UNLP - 50 y 115 1st Floor, (1900) La Plata, Argentina. TE/Fax +(54)(221)422-7707. http://lidi.info.unlp.edu.ar

It is also necessary to operate in real time, implying the use of specialized hardware and robust algorithms in order to reduce failure possibilities; the systems must be flexible enough in order to rapidly adapt themselves to changes in the process. The temporal constraint naturally leads to the idea of a problem solution based on parallelism in order to fulfill the requirements [Akl97][Bri95][Tin98][Zom96].

In this paper, the stages related to computing vision systems are analyzed, such as the tracking mechanism and object recognition in the image sequence obtained with a video camera. A first sequential solution is developed, consisting in an algorithm set that allows capturing objects data and their movement from the sequence. In the second stage, algorithms are parallelized both in the recognition process and in the tracking process. This is implemented over an heterogeneous network with a MPI programming support. The found solutions allow making a comparison and assessment of the performance, whose results are shown at the end of the article.

Tracking in Computer Vision

The objective of a computer vision system is to create a model of the real world from images. For this purpose, a closely related field is that of digital image processing including subject matters such as enhancement, compression and correction of blurred or out of focus images. Computer vision algorithms take images as inputs and produce outputs as a representation of the object contour found in an image.

Image processing algorithms are useful in the first stages of a vision system, usually to emphasized data and suppress noise. The objective of the *enhancement* is to emphasize the features for further analysis or for a display in an output device. *Segmentation* identifies the semantically significant components in an image, and groups the points belonging to the same; the proceeding is simplified if the images are sharp, reason why the enhancement generally constitutes an important previous step. The segmentation result is a set of regions or objects which can be identified by a location or a set of features.

An object *recognition* system finds real world items from an image captured using already known models. The recognition is related to the segmentation, and implies the consideration of the object representation manner, the important features to be detected, the way of comparing the features with those of the models, etc. Recognition complexity depends on several factors, such as for example whether the conditions in which the image was taken (illumination, background, point of view) are not similar to those under which the model was defined. Another factor is the occlusion: if there is only one object, it can be entirely visible; but as the quantity increases, the possibility of occlusion increases as well, and the recognition process may be hindered.

Dynamic Vision

Although the first computer vision systems were principally related to static scenes, solutions that analyzed dynamic scenes have been designed for several applications. The input of the same is a sequence of frames taken from a real moving scene. Each frame represents an image of the scene in a time instant; the changes occurring in the same may be due to the movement of the camera, of the objects, or to modifications in the illumination or in the objects shape or size. The system must detect the changes, determine the movements characteristics, recover objects structure and recognize them.

There are four possibilities for analyzing a dynamic scene: stationary camera and stationary objects (SCSO), stationary camera and moving objects (SCMO), moving camera and stationary objects (MCSO), moving camera and moving objects (MCMO). Depending on the case, different techniques will be needed. SCMO has received the greatest attention; the objective usually consists in detecting the movement, recognizing the moving objects and computing the characteristics of those movements.

The *correspondence* process tries to identify the same object in two or more frames and determine changes in its location (i.e., its movement). An image point $p_i = (x_i, y_i)$ is meant to be coupled with a point $p_j = (x_j, y_j)$ of the following one in a temporarily ordered sequence. The disparity between these two points is given by the displacement vector $d_{ij} = (x_i - x_j, y_i - y_j)$. The result of the correspondence between two consecutive frame points is a set of conjugated pairs.

In order to solve the correspondence problem we should consider: which criterion is used to determine whether two points correspond to each other; which the characteristics to be compared are; which constraints, if any, are imposed to the displacement vectors. As parameters for answering these questions, it is essential to bear in mind the following properties:
- *Similarity:* it is a measure for determining the likelihood of two pixels. It is given by a similarity between the objects, determined from the features found during the segmentation.
- *Movement Consistency*: due to inertia, a physical entity movement cannot change instantaneously. If the sequence is obtained with such a frequency that there are not decisive changes in two consecutive frames, the images reflect the movement smoothness for most of the objects.

Tracking

Given an image sequence over which one or more trajectories are tracked. If there is only one object, the tracking problem can be easily solved. In the presence of several entities moving independently, the use of constraints based in the nature of the objects and their movements will be required. Considering the consistence property in the movement, the *path coherence* can be formulated: it implies that an object movement in any point of a frame sequence will not change abruptly. Thus, the formulation of a solution to the correspondence problem is simplified since it can be implied that (a) a given point location, (b) a point scalar speed, and (c) a point moving direction remain relatively without change from one frame to another one.

The *deviation function,* or *trajectory function,* is used in order to assess the movement properties in a sequence. Its input is a route and the return value must be inversely proportional to the route smoothness (or directly proportional to the path deviation degree).

Let a trajectory be $T_i = \langle P_i^1, P_i^2, P_i^3, \cdots, P_i^n \rangle$, where P_i^k represents a point in the k-th image. If the coordinates of P_i^k are given by X_i vector of the frame, the coordinates of the k-th frame can be represented as X_{ik}. Vectorialy, $T_i = \langle X_{i1}, X_{i2}, X_{i3}, \cdots, X_{in} \rangle$.

d_i^k deviation in the path - of the point in the k-th frame - is given by $D_i^k = f\left(\overline{X_{ik-1}X_{ik}}, \overline{X_{ik}X_{ik+1}}\right)$, where f is the path coherence function (which gives an idea of how much coherent an object movement passing through theses two points could be). The deviation for all the trajectory is defined as $D_i = \sum_{k=2}^{n-1} d_i^k$.

If there are m points in a sequence of n frames, which results in m trajectories, all trajectories deviations should be considered, given by $D(T_1, T_2, T_3, \cdots, T_m) = \sum_{i=k}^{m} \sum_{k=2}^{n-1} d_i^k$.

Then, the correspondence problem is solve by minimizing the total deviation D (or, what would be the same, maximizing the movement smoothness), in order to find the right trajectory set.

If the camera sampled frequency is high enough, the change in the direction and the speed of any moving point in temporarily consecutive frames is smooth. This is described with the deviation function $f\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = w_1(1-\cos q) + w_2\left(1 - 2\frac{d_1 d_2}{d_1 + d_2}\right)$

Vectorialy: $f\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = w_1\left(1 - \frac{\overline{X_{ik-1}X_{ik}X_{ik}X_{ik+1}}}{\|X_{ik-1}X_{ik}\|\,\|X_{ik}X_{ik+1}\|}\right) + w_2\left(1 - 2\frac{\sqrt{\|X_{ik-1}X_{ik}\|\,\|X_{ik}X_{ik+1}\|}}{\|X_{ik-1}X_{ik}\| + \|X_{ik}X_{ik+1}\|}\right)$

The first term is the vectorial product of the displacement vectors and represents the *direction coherence* (the vectorial product is the angle between two imaginary segments between two points k-1,k and k+1): the smaller the angle, the greater the deviation. The second term considers the geometrical and arithmetical mean of the magnitude, and represents the *speed coherence.* w_1 and w_2 are weights chosen to assign different importance to direction and speed changes. They can be chosen from a range from 0.00 to 1.00, only if their sum is 1.

One of the major difficulties with the sequences is the occlusion, i.e., the total or partial disappearance of objects, or the appearance of others. The changes in data due to movements and scene illumination variations can lead to incorrect correspondences. Trajectories can nevertheless be obtained even in the presence of occlusion by forcing them to fulfill some of the local constraints, and allowing them to remain incomplete where necessary.

One path coherence algorithm limitation is that it assumes that in all of the frames, the same points set is available. When searching for a trajectory set, it should be permitted to obtain incomplete trajectories indicating occlusion, objects appearance, or temporal objects disappearance due to bad quality data. Furthermore, certain constraints should be imposed to the possible maximum displacement and the maximum local deviation.

Given a P^j set of m_j points for each of the n frames, it can be found the maximum set of complete or partially complete trajectories that minimizes the sum of local deviations for all of the obtained trajectories, provided that the maximum deviation for any of the trajectories does not surpass $\phi_{\max}$, and that the displacement between any pair of successive frames for any trajectory is always inferior than $d_{\max}$. In order to justify the points left due to occlusion, ghost points are used (hypothetical points used as a filler to extend all the routes over a given frame set). In order to make use of the notion of ghost points, the displacement and local deviation values must be defined for a route containing such points.

- In order to compute the displacement for T_i in the *k*-th frame, the displacement function is defined:

$$Disp\left(P_i^k, P_i^{k+1}\right) = \begin{cases} EuclideanDistance(P_i^k, P_i^{k+1}) & if\ both\ points\ are\ real \\ d_{\max} & in\ other\ case \end{cases}$$

This implies that a ghost point always moves in a fixed quantity $d_{\max}$.

- In order to compute the local deviation for T_i in the *k*-th frame, the deviation function is defined:

$$Desv\left(P_i^{k-1}, P_i^k, P_i^{k+1}\right) = \begin{cases} 0 & if\ P_i^{k-1}\ is\ a\ ghost\ point \\ f(P_i^{k-1}, P_i^k, P_i^{k+1}) & if\ the\ three\ points\ are\ real \\ f_{\max} & in\ other\ case \end{cases}$$

This local deviation function definition is equivalent to the path coherence function if the three points are real. It introduces a penalty of f_{max} in case there is no real point for T_i in the current frame or the following one, and does not penalize if the trajectory begins in the frame under consideration.

The following set of steps roughly shows the trajectory building process.

Initialization:

1. For each point m_k in P^k, k=1, ..., n-1 (P^k set of points of the frame k), determine the closest neighbor in P^{k+1} within distance $d_{\max}$. Solve arbitrarily in case of multiple alternatives.
2. Form initial trajectories chaining the closest neighbors found in the successive frames. Extend all the incomplete trajectories using ghost points in order to lead them through *n* frames.
3. For each trajectory in step 2, form an additional route only with ghost points.

Interchange cycle:

For each frame from k = 2 to n-1

For i = 1 up to m-1

For j = i+1 up to m

If d_{max} constraints are fulfilled, compute

$$G_{ij}^{k} = \left[f\left(P_i^{k-1}, P_i^{k}, P_i^{k+1}\right) + f\left(P_j^{k-1}, P_j^{k}, P_j^{k+1}\right) \right] - \left[f\left(P_i^{k-1}, P_i^{k}, P_j^{k+1}\right) + f\left(P_j^{k-1}, P_j^{k}, P_i^{k+1}\right) \right]$$

Take pair *ij* with maximum gain (G_{ij})

If the gain is superior to zero then

Interchange the points of the frame *k+1*, P_i^{k+1} with P_j^{k+1}.

Termination:

Repeat the interchange cycle until there is no more frames left.

The number of trajectories built in initialization step 2 depends on the data quality. Only *m* routes will be formed and none of them will have ghost points in the ideal case –where the same *m* number of points is consistently present in all the frames. In the worst of the cases, when the points of some of the frames do not have any correlation to the points of any other frame, the number of routes formed can be as large as the total sum of all the points of all the frames. In general, the route number will be at least m_{max} where m_{max} = max(m_1, m_2,... ,m_n) and the different routes will have different quantities of ghost points.

A case with several objects is always implied. In the particular case in which a known object must be tracked, the solution for economizing the process time is to check in each image only the part where the object location is estimated, in function of its location in the previous frames, and considering the estimated object movement. On the other hand, in this route monitoring algorithm, the objects are considered as a set of indivisible entities. A more interesting situation is that in which the searched object/s model is known, and as the image is checked, the segments that do not fit with any of these models can be discarded.

Problem presentation and Sequential Solution

Considering the system as applicable to reality, the data source must come directly from a device capturing images in real time. With the purpose of development and assessment, data was completely available before the application execution. Images were taken with a camcoder. The obtained analogical video was digitalized and converted into a sequence of images. In order to generate a code in which the management of images was simple and clear, pgm format images were used, codified in the grey scale form 0 to 255.

The following algorithm gives a first idea about the principal system running cycle:

While (there are frames)
Take a frame
Identify the objects of the frame
Identify and build up the found objects trajectories

What does “while there are frames” mean? In the case of images coming from a real time capturing device, it is equivalent to “while the images are being captured”. For already available images, it means “while there still are unprocessed images”.

Since sample images were captured specifically with this end, they had no damage or bad quality. However, they did not show a clear difference between objects and background (due to the lack of illumination during the capturing), and besides, before an increasing movement of the objects, some frames of the sequence were blurred. This raised the need of enhancing images in order to facilitate recognition. This was carried out first by calculating an spatial average in order to upgrade the definition, and then by highlighting the points belonging to the objects by means of a contrast enhancement function.

For the object recognition, the images were run from left to right and from top to bottom. Since the implementation was focused in the tracking rather than in objects recognition, it was implied that these were distinguishable enough in the background, and that the interest features were minimum: location, area,

average intensity (grey or color level). These features were enough to characterize them, since any scene objects were searched and not objects with specific characteristics. The technique used for the identification of the objects within an image combines thresholding with labeling.

Once the routing of all the images was finished, data was debugged, discarding all segments small enough or weak in color that they do not require much consideration - since they may be just stains or shadows. One constraint imposed to images was that the objects could not be in touch with each other. When the process was finished, a list of the objects found in the scene with the features identifying them was obtained as a result.

The implementation is general, in the sense that in the scene there are several objects and that they are not indivisible one from the other. For more specific cases, the solution can be improved by checking only the part in which the object may be located in function of its location in the previous frames. Having in mind that the images come from reality, it makes no sense searching an object far from where it appeared in the preceding frame.

Objects Correspondence and Tracking

It is implied that the sequence corresponds to a filming where the camera is fixed, and that the objects movement is parallel to the image plane. For the trajectories building, the initial algorithm was modified (which supposed that all the sequence was available from the beginning) for the case in which images are incorporated in real time.

While (there are frames)
Take a frame
Identify the objects of the frame
Update the found objects trajectories

Lets suppose that the current sequence frame is *k* and that in each of the previous ones, P1, P2 . . .Pk-1 objects were identified. Up to the moment there are built *n* routes that go from 1 frame to k-1. The process must associate frame objects one-to-one with one of the trajectories under construction. The algorithm that solves the problem is the following:

For each existent object in the frame k+1, create a ghost trajectory
// Compute the maximum global deviation
For i=1 up to m-1
For j=i+1 up to m
If d_{max}, constraints are fulfilled in the interchange, compute

$$G_{ij}^{k} = \left[f\left(P_i^{k-1}, P_i^{k}, P_i^{k+1}\right) + f\left(P_j^{k-1}, P_j^{k}, P_j^{k+1}\right)\right] - \left[f\left(P_i^{k-1}, P_i^{k}, P_j^{k+1}\right) + f\left(P_j^{k-1}, P_j^{k}, P_i^{k+1}\right)\right]$$

Take pair ij *as maximum gain (G_{ij})*

If the gain G_{ij} is greater than zero then
Interchange the points of the frame k+1, P_i^{k+1} *with* P_j^{k+1}.
Save the result of this step in the trajectory file

In order to implement the deviation function, some decisions should be taken:
- The method to be used for the distance between two points. In this case, *city-block*: $d = |i_1 - i_2| + |j_1 - j_2|$ was used.
- The weights to be used for the direction coherence and the speed coherence in the formula, w_1 and w_2. In this case, there was no preference for any of the measures, thus $w_1 = w_2 = 0{,}5$.

Trajectory-related data is maintained in a trajectory file. In order to simplify the interchange algorithm and to save computing time, route data related to the last three frames is kept in memory. The order in which the objects are saved is that resulting from the tracking algorithm application. In the case of the ghost points as part of a trajectory, they are also saved with an invalid row and column value (-1,-1), distinguishing them from a common point.

The implementation also allows plain data (with text format) – which constitutes the found trajectories – to be represented graphically. An image is generated that, for each identified object, displays with a trace the trajectory performed by it all throughout the images sequence. The following algorithm shows the building of the routes images:

Let MAX be the object quantity of the first frame.
For each of these objects, generate an image associated to its trajectory.
For each frame
For I=1 up to MAX
    If the point exists in the frame
        Generate an line from the object point in the previous image, up to
            the object point in the current image
    If the point does not exist in the frame (ghost)
        Project the position and draw in another color or leave it in white
For each new object appearing in the frame
    Increase MAX in 1
    Create a new trajectory image.

Parallel Solution

The Parallelization was performed over an heterogeneous computer network and support MPI [GDB95] [Mil98] [Mor94] [MPI1] [MPI2] [MPI3] [PBM] [Sim97] [Sni96] [Ste96] [Wel96]. The operation dealt with was the contrast enhancement by means of a spatial average. For the parallel code, a root or master process distributes per columns the image among all the processes, and compiles the results building the final image.

Object recognition process combines thresholding with labeling. The parallelization is based in the same idea but introduces some details. The root distributes the columns equitably among the processes and performs part of the general operation. Each process runs over the subimage extracting the candidate objects data. Then, each of them send data to the root in such a way that the result of the object recognition process in the image is focused in a single node. In order to solve the case of objects belonging to more than one subimage, each process identifies the candidates "stick" to the left edge of its subimage, and then it "forgets" that it has found that object; it is delegated to the left process and discarded from its own set of objects. When receiving the process data of the right neighbor, it verifies if it coincides with the edge of some of the right-edge adjacent objects. If this is so, the process unifies data, adding the received object data with that of the own object. Finally, each process sends the found object data to the master, finishing with the process.

Once image objects are identified, the trajectories are to be built. Remember that in the sequential solution - being P_1, P_2, ..., P_k the objects sets found in an k image sequence – for each image P_i objects were taken and "hooked" in the trajectories built for the previous images following the algorithm. Data related to the different routes is saved in a file as the process advances. In main memory, there is always information about the last three frames by performance and easiness of processing. Reviewing the given solution, the sequential process for performing the object tracking can be summarized as follows: it is a repetitive cycle in which, as new sequence frames are available, the trajectories built for the previous frames are updated with the identified objects in the new frame. That is, let the instance be one where frame i has to be processed. There are N incomplete trajectories identified in the $i-1$ of the previous frames. The following step consists in "hooking" the x objects identified in the frame i in the previous routes. In order to do this, it is necessary to perform several comparing operations for determining which object display rendering the best trajectories set is.

As a result of these operations, it is probable that the order of some of the found objects has to be interchanged. If this is so, the cycle is repeated until there are no more interchanges. If *n* is the maximum between the built trajectory quantity and the number of objects found in the new frame, the quantity of comparing or verification operations to be performed in each cycle is *(n-1) + (n-2) + ... + 1*. It can be proved that this sum is equal to $\sum_{i=1}^{n-1}\left[(n-1)^2+n-1\right]/2$:

Remember that each mentioned operation is a function that, given two triads of points or objects (where each corresponds to one to three consecutive images), determines a possible trajectory verifying which combination renders the best total deviation (whether both of the given routes, or those resulting from interchanging the third point of the triads). When the result is favorable for the triads with the interchanged point, the mentioned interchange must be carried out. The quantity of interchanges and necessary cycles obviously depends on the sample data set.

A parallelization consists in distributing equitably that quantity of operations among the processes. If the number of operations is not an exact multiple of the number of processes, the remaining operations will be distributed one for each of the first process. As for the rest, all of them would perform the same functions, with a local copy of the most recent image data. Here are some examples of the quantity of operations according to the number of processes: for a small number of objects, the distribution of the operations would be an insignificant quantity of processes to be performed for each one, adding up to the communication times. Before this, the best is to discard this possibility and search a more appropriate alternative.

For it, the solution is considered from a lower refinement level. The problem in its general expression consists in identifying the objects in a sequence in order to track their route. Trajectory updating can be carried out only if the object identification is completed in at least three images of the sequence. When there is more than one process, instead of parallelizing the internal algorithm of the route monitoring, the two central tasks are separated from the different processes. That is, on the one hand, there is a set of processes that carry out the identification, whereas a separated process is in charge of updating the different routes. With this, the trajectories updating can be superimposed with the object identification phase in the following image. If that time is short and keeps the idle responsible process for a long time, this part of the process can be carried out by one of the first group processes.

Results Assessment

One of the original objectives was to obtain acceptable results for real time problems. The speedup and efficiency of the parallel solution were analyzed using different quantity of processors and processes, and different image sizes.

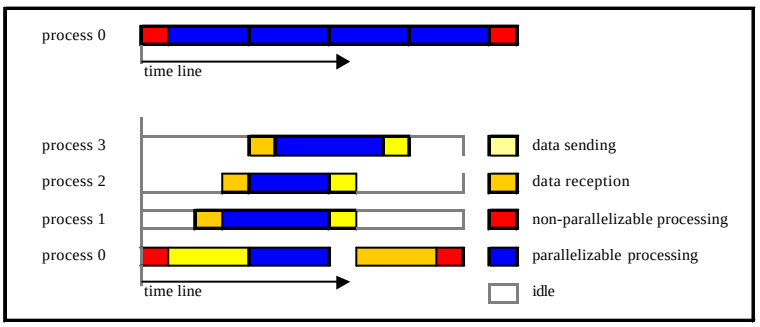
Ideally, efficiency is meant to be 1. If the ideal is taken as parameter, it can be measured how far the implementation of that ideal is. In the best of the cases, the time for executing a solution decreases proportionally to the number of processes. This is, if N = number of processes and T_N = time for using the problem employing N processes, then as *N* increases, T_N should increase as well. In the ideal case: $T_N = T_1 / N$ o $N = T_1 / T_N$. This last formula gives a *speedup* marker. In general, $T_1 / T_N \leq N$. The quotient between the real speedup in N processes and the ideal gives a measure for the *efficiency*. Formally speaking, *efficiency = real / ideal = real speedup in N processes / N.* Low efficiency values represent a waste of the processing power [Hwa93] [Kum94][Lei92].

In general, for each part of the parallel solution, the total runtime can be divided in two components. On the one hand, the part of the problem parallelized, where the addition of processors actually reduces the times, even really close to the ideal. Given $\boldsymbol{T_p}$ as the time used for executing this part of the problem in the sequential solution. On the other, the part of the problem inherently sequential or non-parallelizable, which is not going to be more rapid by adding processors. Naming $\boldsymbol{T_S}$ the time that takes to execute this part of the problem, $T_1 = T_S + T_p$

T_S is an inferior limit for the processing time, thus in the best of the cases, $T_N = T_S + (T_p / N)$. The fraction of the total runtime corresponding to the non-parallelizable part, T_S / T_1, imposes a constraint in the reduction of the times independently of the number of processes. The greater this fraction, the less efficient the parallel solution would be. It is important to bear in mind that this factor is not a constant, but it varies with the problem size: as it changes, T_S / T_1 factor changes as well. Typically, T_p increases much faster than T_S. As a consequence, a complete assessment should be extended over at least two parameters: the problem size and the processors number. To this, it may be added the fact that the performance can be seriously affected both by input/output operations and by the communication quantity among the processes.

In order to determine the characteristics that the image sequences should contain, in a way that the assessment could contemplate all the possible situations, the algorithms of each part of the system were analyzed in first place. From there, different testing cases were defined and the necessary images sequences were generated. Then the algorithms were assessed over each of these cases varying the quantity of processes and the machines involved.

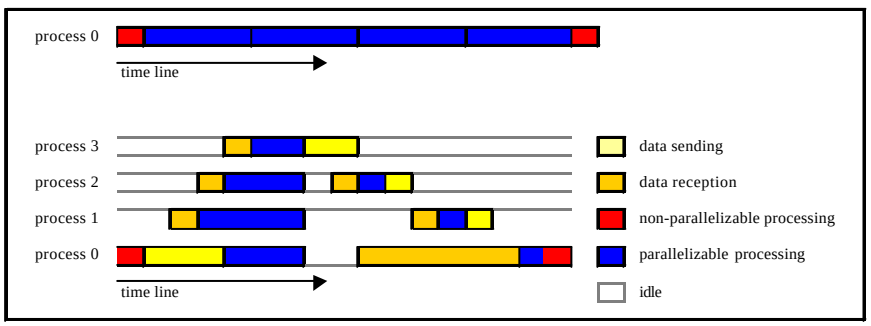
In the enhancement, T_S includes images obtaining, opening and closing of files and format recognition (e.g., find out its dimensions). T_p includes computing the value for each point of the new image. In the case of more than one process, this time is no purely processing time, but it carries the sending and reception times of data among the processes. With a temporal graphic:



In this case, the size of the problem is directly proportional to the image size which is being processed. If we consider that each process receives, apart from the data columns to be processed, both adjacent columns, it can be noticed that as there are more processes, more data traffic is generated. But at the same time, the addition of processes implies a decrease in the processing time per process. Thus, it is necessary to search an equilibrium between these two points.

In order to measure the solution performance for the enhancement, testing cases were generated with different image sizes (problem size) and different number of processes. The results showed that, independently of the type of images, the times used up by the enhancement tasks were similar. What it could be first observed was that a small size image sequence is not the best candidate for being enhanced in parallel. In this case, the times used up by a single process to enhance the operation were better than those resulting form the parallelization. On the other hand, for a larger size of image, the speedup was of 22% over the base time when incorporating a second process; it is not an optimal result, but is nevertheless positive. The same did not happen when a third process was added, with which the processing times were again increased.

Secondly, the recognition process was assessed. Here, T_S includes the image obtaining and the format recognition, and T_p is the time used up by the objects identification; it can be observed - in the case of the parallel solution - that this time carries itself data sending among the processes, plus the eventual “assembly” of objects which have remained divided in the distribution. In a temporal graphic :



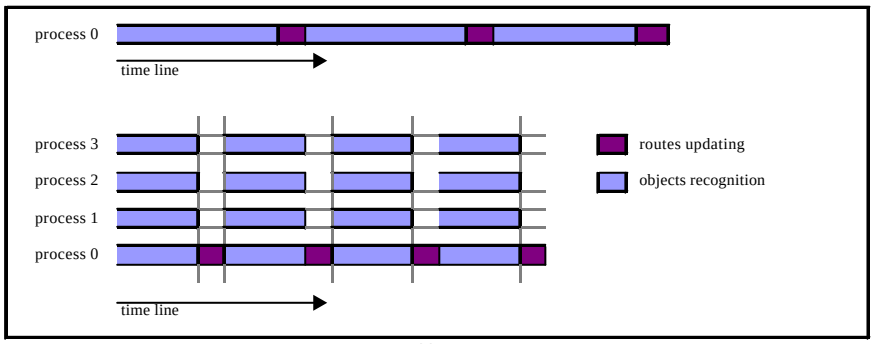
In this case, the size of the problem depends on two main factors:
- the image size. The more rows and columns are, the more points or pixels have to be checked in order to find the objects data.
- the particular “data set”. The quantity, size and display of the objects directly influence in the processing time, and the data quantity which will have to “travel” between the processes.

For example, a large problem is characterized by high resolution images, or with several objects, or with objects whose shape is enlarged in such a way that when dividing the image, they can be “partitioned”. With respect to the processes quantity, as with the previous case, when two more columns are distributed per process, the traffic increases with the number of processes, but each of them has less number of data to be checked. The disadvantage is that when the image is more partitioned, there exists the possibility of partitioning the objects even more, all of which adds time of “assembly” of objects in the end.

In order to measure the solutions performance suggested for the recognition, testing cases were generated with images resulting in different problem sizes and with different quantity of processes. The different tests were grouped according to the mentioned characteristics, in a way to make the assessment more interesting. In this way, results were obtained for tests grouped according objects quantity and for tests grouped by the size of them. Objects display was not considered as a grouping criterion since we are dealing with sequences in which objects move, and thus within a same test this characteristic is not fixed – objects are moving in a sector which may be very different from their location at the end of the sequence.

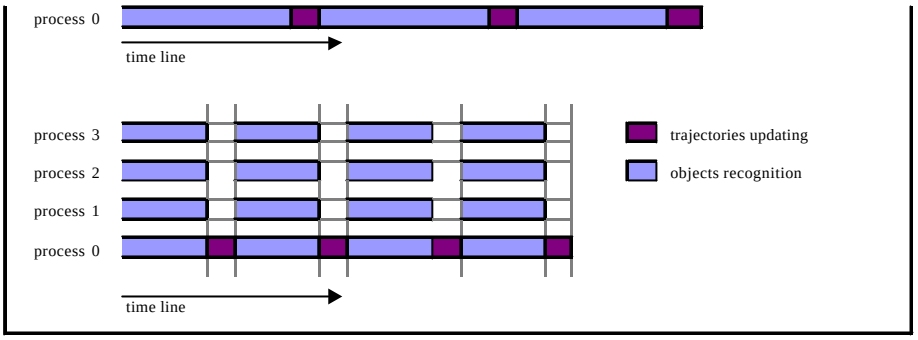
The tests regarding the recognition algorithm reveal better results than the previous case, though always in the case of larger sized images. For smaller images sequences, in most of the cases, the times increase with the process quantity (except when the images contain large objects, but the gain is relatively small and the efficiency is so reduced that the final results are actually poor). Instead, for larger size images, the incorporation of processes upgrades the times, principally in the cases in which the images contain large objects. Also, the speedup attained when incorporating a third process is even better that the one obtained when going from one process to two processes.

Finally, it remains assessing the object tracking process. The given solution, represented temporarily, renders the following situation:



This is similar to the recognition case. The difference is given by the processing which must perform the master or root after each image is analyzed. For this, in order to deal with the problem dimension, the same concepts posed for that case are valid: object quantity, display, etc. Also, since the quantity of operations to be performed (as regards the trajectories updating) depends on the objects movement characteristics, speed and disparity with which they move will also influence in the total runtime of both sequential and parallel solutions.

The task of trajectory updating, as it was already assessed, represents a very small fraction of all the involved tasks. Because of this, it would make no sense dedicate a process exclusively for it. It could be done in the presented case as an upgrading: generated images showing graphically the trajectories drawn by the moving objects. In that case, a separated process would update the routes and would also generate those images, obtaining the following graphic:



In order to measure the tracking process performance, images sequences that vary the problem dimension were generated: two or three objects sequences, sequences with more than three objects, sequences with slow and even moving objects, sequences with rapid and uneven moving objects.

As in the recognition case, the tests were grouped according to the mentioned characteristics. On the one side, tests grouped by movements speed and coherence (slow and even movement, moderated and even speed, high speed and even movement, uneven speed and coherence), and on the other, tests grouped by objects quantity.

It is important to bear in mind that tracking algorithm tests refer to the integral test which goes from the enhancement of each individual frame to the final result. It is not the monitoring exclusive algorithm test.

These tests encompass all the images processing cycle. As the route maintenance exclusive portion is a very small part of the algorithm, and it is a task of a single process, it is probable that the result will not be reverted in respect to the previous tests about objects enhancement and recognition over the images. Actually, this is what happened.

Summarizing,

- For small images (in particular, 320 x 240) it is not suitable parallelizing the algorithms since, in proportion, the communication time is greater than the one used up by each of them in order to perform the different tasks. Working with more than one process lower the times only when the images objects were large, but the speedup resulted so low that, even these cases, does not deserve consideration.

- For larger images (in particular 720 x 540), the results were acceptable, not so much in the initial algorithm of images enhancement, but in particular in that of object recognition and, of course, in the joint test of all the system. Specially, the results were better in the cases in which the images had large objects in relation to the frame size. Finally, when analyzing the algorithms according to the types of the movements of the objects, the results were better though not by a great difference in relation to the others.

Conclusions

After having completed the implementation and the assessments of the described algorithms, the result obtained fulfilled the presented expectations, namely:

- A solution for identifying objects in an images sequence and identifying the trajectories of their movements was fully developed.

- The cases in which the parallel solution is good were identified as well as those cases in which is preferable adopting the traditional solution.

On the other hand, all throughout the project, upgradings or alternatives to the implemented solution were presented. They are here summed up together with some additional suggestions:

a) *As regards the project:*

- Generate additional images reflecting the scene objects movement.

- Identify the objects based in those features distinguishing them. In this paper, the objects were indivisible. This upgrading is completed with a modification in the tracking algorithm appearing in the objects not only due to its position in the consecutive frames but also to the features individualizing them.

- In the case that a single object has to be tracked, modify the recognition algorithm in order to check only one part of the image depending on where the object has been found in the previous frames. This upgrades the solution, without wasting the processing time.

- Modify or upgrade the presented algorithms in order to render a better response according to the different sequences to be assessed. The project is implemented in a such way that it is easy to change any of its parts by another one, responding better to the requirements (e.g. it may be desired to change the image recognition algorithm, or to work with another image format).

b) *As regards performance*

- Assess how often it is necessary to perform a selection of the threshold for the segmentation. Given the natural smoothness in the images movements, it is no necessary to recompute that value for each frame since significant changes will not be obtained from one image to the following one. However, the necessary frequency can depend on the speed of the objects movements.

- Implement an alternative in order to handle the auxiliary data (labels and objects) in memory instead of objects. This option would be better in time, but it would require the availability of machines with enough memory.

- Assess the possibility of skipping frames in the cases where it is beforehand known that the objects in the scene move really slow; this allows obtaining the same data with better times.

- For the case where it is known that the objects are of considerable size, implement a variant that does not check every pixel but takes one every two or more so as to reduce the processing. It is expected that the results do not change, but the times probably do.

- Vary the distribution of the tasks among the different processes; for example, that the enhancement is carried out by two processes, while the recognition task is performed by three. This distribution has to do with taking advantage of the conclusions obtained from the performed tasks. On the other hand, if more machines are available, other possibility is to distribute the task in a way that a group performs the enhancement and another group of processes in the other machines performs the recognition.

Also, the implementation can be migrated towards an homogeneous multiprocessor (for example, the transputers hypercube available in the Laboratory of Parallel Processing of the Faculty of Computer Sciences UNLP), with the objective of comparing the performance on both platforms.

Bibliography

[Akl97] Akl S, “Parallel Computation. Models and Methods”, Prentice-Hall, Inc., 1997.
[Bax94] G. A. Baxes, "Digital Image Processing. Principles and Applications", John Wiley & Sons Inc., 1994.
[Bri95] Brinch Hansen, P., “Studies in computational science: Parallel Programming Paradigms”, Prentice-Hall, Inc., 1995.
[Gon92] R. C. González, R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Comp., 1992.
[GDB95], GDB/RBD, "MPI Primer / Developing with LAM", The Ohio State University, 1995.
[Gup93] Gupta A., Kumar V., “Performance properties of large scale parallel systems”, Journal of Parallel and Distributed Computing, November 1993.
[Har92] R. M. Haralick, L. G. Shapiro, “Computer and Robot Vision”, Addison-Wesley Publishing Company, 1992.
[Hwa93] K. Hwang, "Advanced Computer Architecture. Parallelism, Scalability, Programmability", McGraw Hill, 1993.
[Jai89] A. Jain, "Fundamentals of Digital Image Processing", Prentice Hall Inc., 1989.
[Jai95] R. Jain, R. Kasturi, B. G. Schunck, “Machine Vision”, McGraw-Hill International Editions, 1995.

[Kum94] Kumar V., Grama A., Gupta A., Karypis G., “Introduction to Parallel Computing. Design and Analysis of Algorithms”, Benjamin/Cummings, 1994.
[Law92] H. Lawson, "Parallel processing in industrial real time applications", Prentice Hall 1992.
[Lei92] F. T. Leighton, “Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes”, Morgan Kaufmann Publishers, 1992.
[Mil98] Miller R., Stout Q. F., “Algorithmic Techniques for Networks of Processors”, CRC Handbook of Algorithms and Theory of Computation, M. J. Atallah, ed, 1998.
[Mor94] H. S. Morse, “Practical Parallel Computing”, AP Professional, 1994.
[MPI1] http://www.mcs.anl.gov/pub/mpi. Information about implementations and MPI in general - Argonne National Laboratory.
[MPI2] http://www.erc.msstate.edu/mpi. Information about implementations and MPI in general – Mississippi State University.
[MPI3] ftp://info.mcs.anl.gov/pub/mpi. Implementation of MPICH of the Argonne National Laboratory and the Mississippi State University.
[PBM] www.acme.com/software/pbmplus Image file format conversion package.
[Sch92] R. Schalkoff, “Pattern Recognition. Statistical, Structural and Neural Approaches”, 1992
[Sim97] Sima D, Fountain T, Kacsuk P, “Advanced Computer Architectures. A Design Space Approach”, Addison Wesley Longman Limited, 1997.
[Sni96] Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J., "MPI: The Complete Reference", The MIT Press, 1996
[Ste96] Steenkiste P., “Network-Based Multicomputers: A Practical Supercomputer Architecture”, IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 8, August 1996, pp. 861-875
[Tin98] Tinetti F., De Giusti A., "Procesamiento Paralelo. Conceptos de Arquitectura y Algoritmos", Editorial Exacta, 1998.
[Wel96] Welsh M., Kaufman L., "Running LINUX", O’Reilly & Associates, Inc, 1996 (Second Edition)
[Zom96] Zomaya A., “Parallel Computing. Paradigms and Applications”, Int. Thomson Computer Press, 1996.

.