Medical Image Analysis
Professor Ganapathy Krishnamurthi
Department of Engineering Design
Indian Institute of Technology Madras

Lecture 20

Demons Part 2

(Refer Slide Time: 00:14)

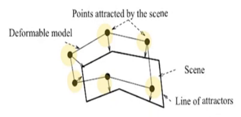


Figure 1: Deformable model with attractors

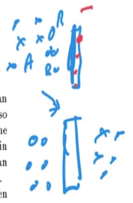
3.2 Demons

Let us now present the point of view of demons. We will suppose that we can determine the object boundary points, as in the case of attractors, but also the local orientation (inside-outside) of the contour for each point, both for the deformable model and for the scene. Those assumptions are easily fulfilled in the case of medical images, where an iso-contour generally suffices to define an object boundary and where the gradient defines the normal to that contour.

This time, it is the scene object contour which is sampled into a given number of points, with the associated orientation, as presented in figure 2. We will see later that alternatively we can sample the whole scene image with a regular grid of points, instead of sampling a single contour. A demon is associated to each of those contour points, in a sense very similar to Maxwell's demons[2].

Each demon acts locally, to push the deformable model in the normal direction of the scene contour, but the orientation of the push depends on the nature (inside or outside) of the current estimate of the model at that point.

[2]Maxwell's demons were introduced in physics to model the paradoxical idea that a semi-permeable membrane could violate the second law of thermodynamics (about entropy). Situated in the membrane, those demons are supposed to sort locally two different types of molecules of a mixed gas, to put one type on one side and the other type on the other side of the membrane, hence giving a decrease in entropy. As each demon needs to consume some energy to sort, the global entropy of the system is still increased, which suppresses the paradox.



Hey, so, welcome back. So, we are to try and understand why this is called the demons algorithm. So, the idea behind that is that, for medical images, we usually have these edges of these organs or anatomies or features of interest. And these are actually ISO contours and we saw that I showed that in the previous video.

Now, if we have these contours, then on the boundaries of these contours, we can position the so called demons like shown here, it just points on the contour. So, in this case, there are two contours one is in the author calls it the scene rather is the deformable model, the scene is basically the fixed image and the deformable the model is the moving image.

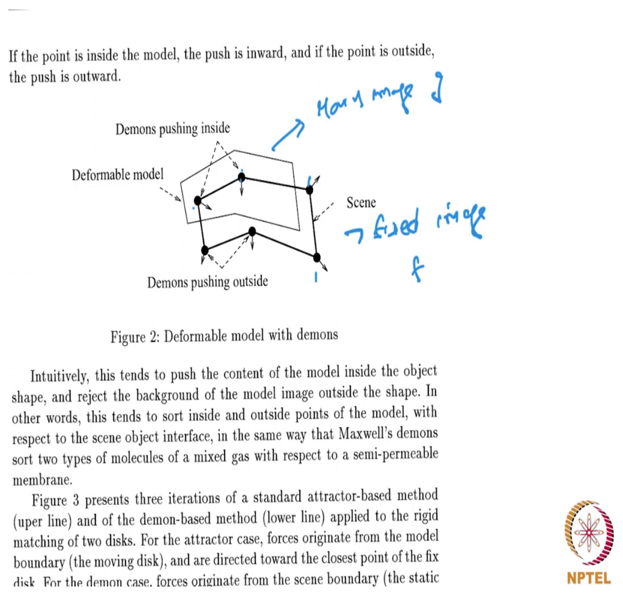
So, we denote them by f and g are respectively. So, the idea is, we can sample the entire image, we have the pixels in a picture, in a medical image, we can look at all the pixels or we can just look at the edges, edges are easily obtained by edge detection, and we can choose points on the edges, you can do, we can find the edge and do points on the edge or we have a segmentation we already know where the edges are.

So, then we position these points, we have these grid points to call the control points or the demons points, so why are they called demons because there is a very similar concept, one systematic concept, the demons concept was introduced in physics, the idea is if you have like a sort of semi permeable membrane, so for instance, something like, like a membrane here, and then you have different types of molecules, A and B, circles and process, and then there are these demons sitting here, because it is semi permeable, you can go in this direction, which tries to separate out, just the A from the B.

So basically on one side, you have the A's the other side, it only gets the B's too. So after a period of time, you will have one set of the permeable membrane, you have all these x's, the other side, you have these o’s. And it turns out that this is a violation of the second law of thermodynamics.

So but then it turns out that that is, that is, that is addressed by considering that the demon needs to can consume energy to solve this, we got to study this, from here to there corresponds to a decrease in entropy, which simply cannot happen. So, the idea of this demons is exactly what he uses here. So, he calls these points on the contours of objects of interest as demons and they serve the purpose of them over the demon that is basically to decide, to push objects inside an object or we will see that in the next slide.

(Refer Slide Time: 03:17)



If the point is inside the model, the push is inward, and if the point is outside, the push is outward.

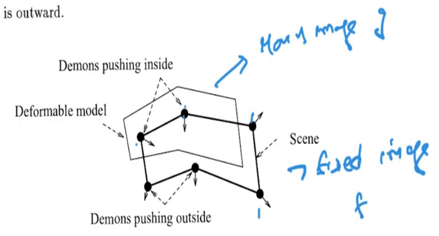


Figure 2: Deformable model with demons

Intuitively, this tends to push the content of the model inside the object shape, and reject the background of the model image outside the shape. In other words, this tends to sort inside and outside points of the model, with respect to the scene object interface, in the same way that Maxwell's demons sort two types of molecules of a mixed gas with respect to a semi-permeable membrane.

Figure 3 presents three iterations of a standard attractor-based method (uper line) and of the demon-based method (lower line) applied to the rigid matching of two disks. For the attractor case, forces originate from the model boundary (the moving disk), and are directed toward the closest point of the fix disk. For the demon case, forces originate from the scene boundary (the static

So, if you look at this picture, this this corresponds to what the demons algorithm does. So, the scene here is nothing but the fixed image f and this is the moving image g. So, we have the corresponding points. So, you have these dark spots here, which correspond to your demon points. And then of course, we assume that is when you start out at the first iteration, there is some degree of overlap.

So, if you look at this, all the points in the scene or the pixels that are inside the model image or the moving image contours, they tend to push inside while those which are outside the model tend to push outside.

So this is kind of like trying to squeeze your moving image into the fixed image. So it and the demons decide which points to squeeze in and which points to squeeze out. So that is what that is why it is called the demons algorithm. Because these control points or the demon points as you can call them, decides I know how to which what piece of the model comes in and what piece in the model stays out.

(Refer Slide Time: 04:42)

Figure 4 are three other iterations, and shows that the behaviors of the two methods become similar (in that particular case) when coming close to the final solution.

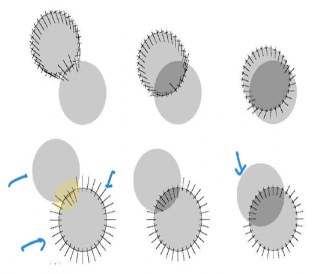


Figure 3: Three iterations of the attractor-based method (upper line) and of the demon-based method (lower line).

3.3 3D grids of demons

For medical images, iso-intensity contours are closely related to the shapes of the objects, because intensity represents density. Let f (resp. g) be the (3D) image intensity function in the scene (resp. in the model). We associate a demon to each voxel P of the scene image where the gradient norm $|\vec{\nabla} f|$ is not null: in that case, an iso-surface comes through that voxel P, whose implicit equation is $f = f(P)$, and whose oriented normal is $\vec{\nabla} f(P)$. The demon in P

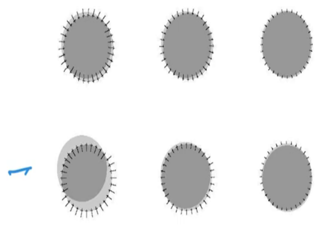


Figure 4: Another set of iterations of the attractor-based method (upper line) and of the demon-based method (lower line).

$-\vec{\nabla} f(P)$ if $f(P) > g(P)$ (see figure 5). Hence a whole 3D grid of demons acts to deform the model.

4 A simple implementation of a demon-based deformable model

We present now an iterative algorithm to perform the non-rigid matching, which can be considered to be an archetype for demon-based methods. We suggest also several opened slots in the basic scheme, which are:

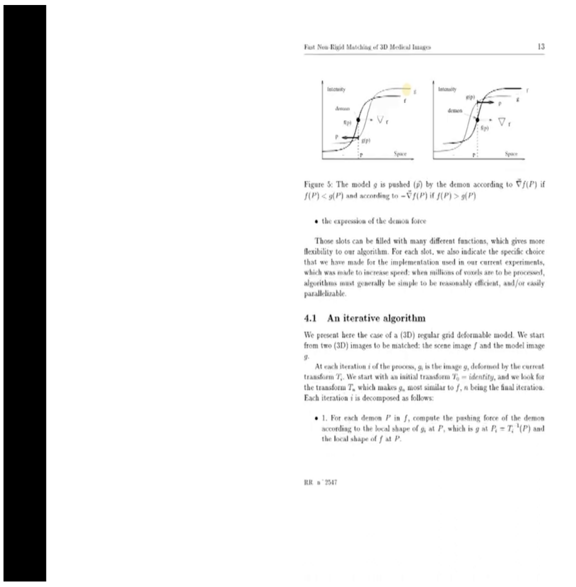
So the f has provided some fixed examples of since if you look at this example here and this basically the second row, you have all these spikes basically show you are more than one second. These spikes basically show here, these ones, they show your gradient pointed along which basically, it is pulling out to it, except in the region of overlap here with this region overlap here, where it is where it is actually dragging the other disc inside.

So, we are trying to match these two discs. So, this is the moving disc, here the moving disc and this is the fixing disc. The fixed disc initially all the gradients are pointing out in the regions which are overlapped basically that is the fixed image is inside the moving image you try to push the drag the moving image inside.

So, you see the first few iterations were in the, if you can call this as the moving image kind of diffuses into the fixed image. So, you can do that and of course, depending on the orientation of the gradient say how do you decide where to how to push, when you should push in when you push out depends on the orientation of the gradient.

So, for instance, how do we decide the demon in p, pushes the model image according to gradient of ∇f, if $f(P) < g(P)$ and you see here, let me let us see this example. So it is completely done. So here it is been completely moved in. And as the author states, according to ∇f, if $f(P) > g(P)$.

(Refer Slide Time: 06:42)



Fast Non-Rigid Matching of 3D Medical Images 13

Figure 5: The model g is pushed ($\vec{p}$) by the demon according to $\vec{\nabla} f(P)$ if $f(P) < g(P)$ and according to $-\vec{\nabla} f(P)$ if $f(P) > g(P)$

- the expression of the demon force

Those slots can be filled with many different functions, which gives more flexibility to our algorithm. For each slot, we also indicate the specific choice that we have made for the implementation used in our current experiments, which was made to increase speed: when millions of voxels are to be processed, algorithms must generally be simple to be reasonably efficient, and/or easily parallelizable.

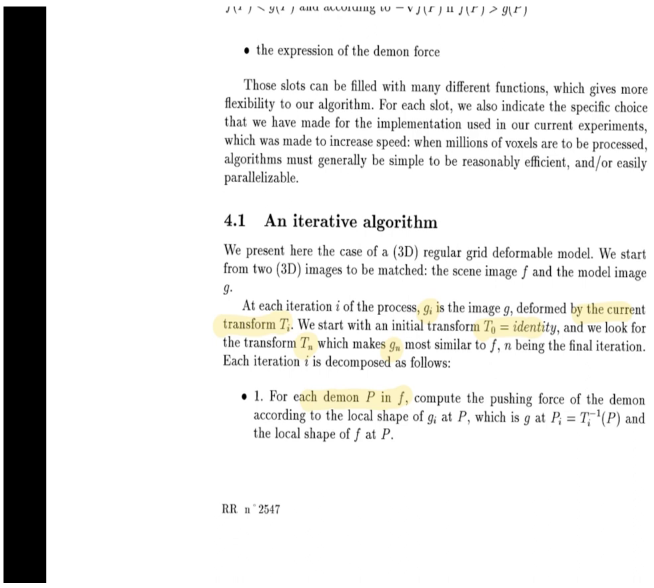
4.1 An iterative algorithm

We present here the case of a (3D) regular grid deformable model. We start from two (3D) images to be matched: the scene image f and the model image g.

At each iteration i of the process, g_i is the image g, deformed by the current transform T_i. We start with an initial transform $T_0 = identity$, and we look for the transform T_n which makes g_n most similar to f, n being the final iteration. Each iteration i is decomposed as follows:

- 1. For each demon P in f, compute the pushing force of the demon according to the local shape of g_i at P, which is g at $P_i = T_i^{-1}(P)$ and the local shape of f at P.

RR n°2547



$f(P) < g(P)$ and according to $-\vec{\nabla} f(P)$ if $f(P) > g(P)$

- the expression of the demon force

Those slots can be filled with many different functions, which gives more flexibility to our algorithm. For each slot, we also indicate the specific choice that we have made for the implementation used in our current experiments, which was made to increase speed: when millions of voxels are to be processed, algorithms must generally be simple to be reasonably efficient, and/or easily parallelizable.

4.1 An iterative algorithm

We present here the case of a (3D) regular grid deformable model. We start from two (3D) images to be matched: the scene image f and the model image g.

At each iteration i of the process, g_i is the image g, deformed by the current transform T_i. We start with an initial transform $T_0 = identity$, and we look for the transform T_n which makes g_n most similar to f, n being the final iteration. Each iteration i is decomposed as follows:

- 1. For each demon P in f, compute the pushing force of the demon according to the local shape of g_i at P, which is g at $P_i = T_i^{-1}(P)$ and the local shape of f at P.

RR n°2547

But there is another figure which actually tells you that one says this right here. So, the model g is pushed by the demon according to a ∇f again, here f and g are these intensity contours, you can think of any dimension f(P) < g(P) then you have pushing to the right in according to this picture, and according to -∇f , f(P) > g(P).

So, this is the simple to that can be done. And we saw this one implementation is what we saw earlier in terms of the horns and flow, where we then made the constraint on v to v along the proportional the gradient of f. And x p there are expression of the update scheme.

So, the idea so, he also describes how this is actually to be implemented. So, here is the explanation for the iterative algorithm. So, at each iteration, I of this process of we saw, where we each iteration we are to estimate the velocity v.

So, g is the, in this case g is the moving image which he calls the model and f is the scene image which is basically the fixed image and g_i, g subscripts i or nothing but the image g defined by the current transform T_i. So, i iteration index. So, we start with adding to the transform and we look for a transform Tn such that gn which is after the application of transform T into g, g_n is the image we get. So, that gn is most similar to f. The n being the number of iterations so, it is hyper parameter mystics. So, each iteration I is decomposed as false.

So, for each demon in P in f is in this case, in many of cases, demon P is basically at the point P and it basically all the pixels in the image, are the most relevant pixels in the image sometimes there is a lot of black space around the image air, you can crop that out in both the moving and fixing image and you can just use whatever is remaining. So, most of the pixels in f.

So, for all for all the control points or what I call control points demon P in f, compute the pushing force of the demon, according to the local shape g_i at P. So, because how do you do, how do we know what is the gi because initially, the transformation is identity. So, you start off with a rough image. So you just have a, look at the corresponding you assume they are in correspondence and then you just work with that.

Otherwise, you have to calculate the inverse T T inverse of P. So for every pixel in f, this very similar to what we saw for rigid body, so very every pixel in f, you come you figure out the corresponding location in g and you use that to calculate the force. Therefore, in our case, we actually do not do the force. We actually directly compute the velocity. If you lose the horns and flow, you just calculate v. And we approximately say that since delta T is 1, we just used that as a displacement.