SCHOOL OF ENGINEERING – STI
Signal Processing Laboratory LTS4

Master Thesis

Dense disparity estimation for spherical images based on belief propagation

Josep Cuscó Mach
June 2009

Supervisor
Prof. Pascal Frossard

Assistant
Vijayaraghavan Thirumalai

Acknowledgments

I would like to thank all those who contributed to this project and made me feel at home during my stay abroad. Whatever little thing they did, even they might think they have not done anything special, it was really worth.

And in particular, I would like to thank Telecom BCN for giving me the opportunity of developing my thesis in a incredible city in a lovely country; EPFL for its charming welcome and for giving me all the facilities; Prof. Pascal Frossard for letting me work at his group and being my supervisor; Zafer Arican for his help; and very specially Vijayaraghavan Thirumalai for all his support. Thanks Vijay.

Abstract

A lot of applications in computer vision are based on a pixel-labelling problem, such as stereo matching, image restoration or object segmentation. In the last years great advances have been achieved in dense disparity estimation, being Graph Cuts and Belief Propagation two of the most outstanding algorithms. Particularly, Belief Propagation has some characteristics which make it very interesting to deal with, i.e. powerful message passing and high flexibility.

Furthermore, working with omnidirectional cameras, instead of standard cameras, a smaller number of images would be needed because of their wider field of view and it would allow reconstructing the 3D scene in an easier way.

This project aims to adapt the Belief Propagation algorithm to spherical stereo images. In addition, as working with spherical images, we should take into account that these images will be projected on a sphere, being then the pixels at different distances between them. Thus, the project also aims to improve the algorithm adding a weighting function which considers the distance between the points on the sphere.

The project contains the general description of the proposed framework as well as an analysis and evaluation of the results obtained after its implementation.

Contents

Acknowledgments 2
Abstract 3
Contents 4
State of the art 5
Introduction 6
Epipolar geometry *6*
Rectification of spherical images *7*
Markov Random Field model *8*
Belief propagation 9
Modification in the sphere 11
Introducing distance weights 13
Distance between points in the sphere *13*
Applying the weights *15*
Implementation 16
The software *16*
Code operation *16*
Results and discussion 19
Synthetic omnidirectional image *19*
Natural omnidirectional image *22*
Conclusions 24
Bibliography 26
Appendix 27
Tables of results *27*

State of the art

Dense disparity estimation is one of the active research areas in computer vision and finds its application in image based rendering or object recognition. Given a pair of stereo images, the goal of the dense disparity estimation is to compute the depth map by assigning a disparity value to each pixel in the reference image.

Over the past few years there have been exciting advances in the development of algorithms for solving early vision problems such as stereo matching for approximate inference in Markov random fields (MRF's), which are powerful tools for modelling vision problems. Two of the most impressive results are in the domain of graph cuts [1, 2] and belief propagation [3, 4, 5]. These algorithms may become the basis for new and powerful vision algorithms [6]. In the realm of stereo, the top contenders for the best stereo shape estimation, on the most common comparison data, either use graph cuts [1,7] or belief propagation [4]. Both algorithms yield highly accurate approximate solutions to MRF's, producing comparable labellings, but were intractable to solve with reasonable speed until rather recently, being often too slow for practical use comparing to local methods which were faster although they achieved poorer results.

Several algorithms have been proposed in the literature for computing the depth map for the standard stereo cameras. However, as the standard cameras capture the scene in a limited view, it usually requires an efficient distribution of cameras to reconstruct the entire view of the 3D scene. On the other hand, omnidirectional cameras capture the scene in a wide field of view, and hence it is more suitable for scene representation since they certainly represent a great advantage in terms of accuracy and efficiency. Omnidirectional cameras have been already widely used in robot navigation and video surveillance [8, 9, 10]. In the stereo problem, the wide field of view permits to reconstruct the associated 3D structure by processing a significantly smaller number of multi-view images, compared to the number of views that would be needed in a standard camera network performing the same task.

Thus, putting together both situations definitely motivated the development of the present thesis. Hence, the goal of the project is to develop a dense depth estimation algorithm for the omnidirectional stereo cameras by borrowing ideas from the techniques developed for the standard stereo cameras. Being the graph cut and belief propagation two of the better performing algorithms in the field, the decision was taken to develop the project based on the belief propagation because its message passing principle offers a high flexibility and, in addition, the graph cut had been already adapted to the sphere. The developed scheme is tested on omnidirectional stereo images and, finally, its performance is compared with that of the graph cut algorithm.

Introduction

This section explains some essential background to understand the problem addressed in this project. For comprehending the stereo matching scenario, basis in epipolar geometry both in the planar and in the spherical case are needed. Moreover, we will see how for performing disparity estimation in a spherical framework we will use a rectification step, which allows using the available planar techniques, such as belief propagation, in the sphere. Finally, we will see the formulation of a Markov Random Field, which is a robust and unified model for early vision problems such as stereo matching.

Epipolar geometry

We will cover the theoretical basis for two-view geometry [11], since we are working in the standard stereo case with two images of the same scene. The two images can be acquired simultaneously or sequentially, for example moving the camera in relation to the scene. Considering the planar case, each view has an associated camera matrix, P, P', where ' indicates entities associated with the second view, and a 3-space point X is imaged as $x=PX$ in the first view, and $x'=P'X$ in the second. Image points x and x' correspond because they are the image of the same 3-space point. In the stereo matching problem, we will address the question of, given an image point x in the first view, how to find the position of the corresponding point x' in the second image.

The epipolar geometry only depends on the camera parameters and position and is independent of the structure of the scene. The intrinsic projective geometry between two views is encapsulated in the fundamental matrix F, which is a 3x3 matrix of rank 2. The image points x, x' of a 3-space point X satisfy the relation $x'^{T}Fx=0$.

As shown in fig.1, the image points x, x', the 3-space point X and the camera centres C, C' are coplanar, lying all of them in a common plane denoted π and called epipolar plane. Obviously, the rays back-projected from x and x' intersect at the space point X.

Supposing that we only know x, we may ask how the corresponding point x' is constrained. We do not know the position of the point X in space but we know it must lie in the ray back-projected from the camera centre C and the known image point x. From the statement above, this ray forms an epipolar plane with the second camera centre C' and the image of X in the second view, x', must lie in this plane too. The plane intersects with the second view defining a line l', which is called the epipolar line. As the image point x' must lie on both the defined epipolar plane and the view plane, we can affirm that it will lie on their intersection, hence in the epipolar line l'. Thus, the search of the corresponding point to x in the second view, x', has been limited to a search on a straight line. This restriction is a good benefit since we do not need to cover the entire image plane to find the correspondence between points.

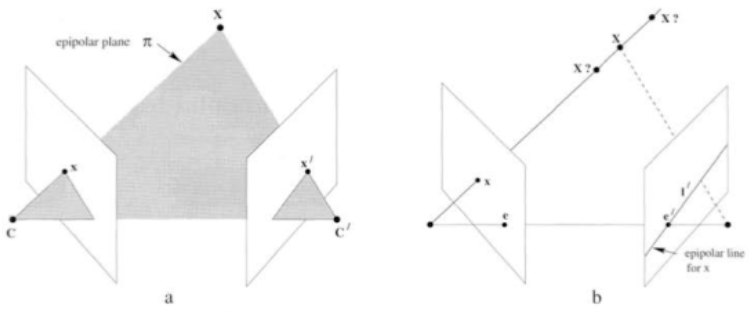

Fig.1: Epipolar geometry in the planar case.

Every point in one view has a corresponding epipolar line in the other view and, moreover, all the epipolar lines intersect at the epipole, which is the projection of the other camera in the view. In the same way, all the epipolar planes intersect at the baseline, which is the line defined by the two camera centres. The camera baseline intersects each image plane at the epipoles *e*, e'. Any plane π containing the baseline is an epipolar plane and intersects the image planes at the corresponding epipolar lines *l* and *l'*. As the position of the 3D point varies, the epipolar planes rotate about the baseline, forming an epipolar pencil.

When moving to the spherical case, the situation is reformulated taking into account that we now deal with spherical images. A spherical image is obtained by the projection of all the visible points from the centre of the sphere, where the camera centre is. The projection of a 3-space point *P* in the sphere is the intersection of the sphere surface with the line joining the point *P* and the centre of the sphere.

We can analyze the epipolar geometry for spherical stereo applying the same reasoning stated above for the planar case. Imagine the two-camera scenario where both cameras capture the same scene. The point in space *P* and the two camera centres, *C*, *C'*, form an epipolar plane. The projection of the point *P* in both spherical images, p_l, p_r, lie in the epipolar plane. The epipolar plane intersects at the two images at the respective epipolar lines, where the projections of the space point, p_l, p_r, lie. However, now the epipolar lines have becomes great circles over the sphere surface instead of straight lines, as can be seen in fig.2. Yet knowing the image of a 3-space point in one of the views the search for the corresponding point in the other view is still restricted to the corresponding epipolar line, but the fact that it has turned into a circle makes the problem much more complicated, as it has become a multidimensional search and we can not apply the standard stereo matching algorithms. Nonetheless, there is a technique that allows converting the stereo correspondence in a spherical framework into a one-dimensional problem, the rectification.

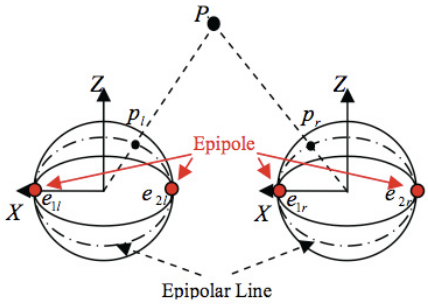

Fig.2: Epipolar geometry in the spherical case

Rectification of spherical images

Rectification is a very important step in stereo vision, both using standard and spherical camera images. It aims at reducing the stereo correspondence estimation to a one-dimensional search problem. It basically consists in image warping, which is computed such that epipolar lines coincide with the scan lines, facilitating the implementation of disparity estimation algorithms and allowing faster computations.

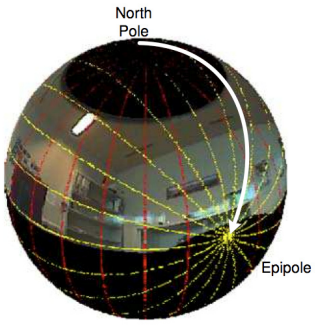

Fig.3: The rectification approach by rotation

In the spherical framework, considering that epipoles are similar to the coordinate poles and epipolar great circles intersecting on epipoles are like longitude circles, it is possible to rotate the spherical image pair such that the epipoles coincide with the coordinate poles (fig.3). In this way, epipolar great circles coincide with the longitudes and the disparity estimation becomes a one-dimensional problem.

At the end of the rectification step, two rectified stereo spherical images (fig.4) are obtained in the form of rectangular images on an equiangular grid, where epipolar great circles have become straight lines.

In consequence, it allows extending disparity estimation algorithms developed for standard images to spherical images.

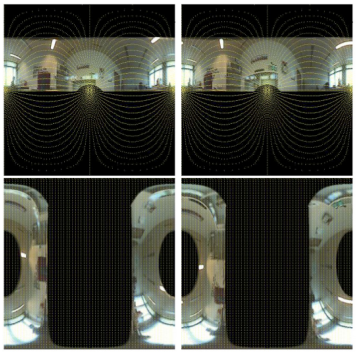

Fig.4: A pair of spherical stereo images before and after the rectification process.

Markov Random Field model

Given a pair of stereo images, the goal of the dense disparity estimation algorithms is to find the disparity of each pixel in the reference image. Most stereo algorithms perform four basic steps: matching cost computation, cost or support aggregation, disparity optimization and disparity refinement. The first three steps can be accomplished by modelling the disparity image as a Markov Random Field (MRF).

The disparity of each pixel in the disparity image is a random variable, denoted x_p for the variable at the pixel location p. Each variable can take one of N discrete states, which represent the possible disparities at that point. For each possible disparity value there is a cost associated to matching the pixel to the corresponding pixel in the other image at that disparity value. This cost, typically based on intensity differences between the two pixels, y_p, is reflected in a compatibility function $\Phi(x_p,y_p)$, which relates how compatible is a disparity value with the intensity differences observed in the image and is usually called local evidence. Smaller differences correspond to higher compatibilities.

A MRF aggregates support for the candidate disparities by introducing a second compatibility function between neighbouring variables. Traditionally, only adjacent variables are considered as neighbours. Therefore, the compatibility function is of the form $\Psi(x_p,x_n)$, where the locations p and n are adjacent. This is known as a pair-wise MRF, which are typically used in stereo problems because considering more neighbours quickly makes inference on the field computationally intractable. Although the compatibility function only considers adjacent variables, each variable can still influence every other variable via the pair-wise connections.

Then, the joint probability of the MRF can be written as

$$P(x_1,\ldots,x_N,y_1,\ldots,y_N)=\prod_{(i,j)}\Psi(x_i,x_j)\prod_{p}\Phi(x_p,y_p)$$

The disparity optimization step requires to choose an estimator for $x_1,\ldots,x_N$. The two most common estimators are the Maximum A Posteriori (MAP) which is the labelling that maximizes the equation, and the Minimum Mean Square Error (MMSE) in which the estimate of each x_i is the mean of the marginal distribution of x_i.

Belief propagation

Belief propagation is an iterative inference algorithm for graphical models such as MRF which is based on a message passing principle that propagates messages in the network. It is an energy minimization algorithm, thus it aims to reduce an energy function representing the labelling quality. This energy function is split into two terms: the data energy and the smoothness energy, which is weighted by a parameter λ, being the total energy $E = E_d + \lambda E_s$.

The belief propagation algorithm provides some features that make it very interesting to work with. Two important properties are due to its powerful message passing. Firstly, it is asymmetric. The entropy of the messages from high-confidence nodes to low-confidence nodes is smaller than the entropy from low-confidence nodes to high-confidence nodes. Secondly, it is adaptive. The influence of a message between a pair of nodes with larger divergence would be weakened more. Therefore, the message passing principle makes it an algorithm with high flexibility. Moreover, there is another characteristic that can be key in evolving the belief propagation in contrast to other algorithms such as graph cuts: the smoothness cost function in belief propagation does not need to be metric.

There are two possible implementations of the belief propagation algorithm with different message update rules: max-product and sum-product. The max-product computes the MAP estimate to find the lowest energy solution, whereas the sum-product computes the marginal probability distribution for each node through the MMSE. In this project, the max-product version is used to find the approximate minimum cost labelling.

The message update schedule determines when a message sent to a node will be used by that node to compute messages for the node's neighbours. Our schedule is to propagate messages in one direction and update each node immediately. The advantage of this method is that information is quickly propagated across the network, requiring only one iteration to propagate the information from one side of the image to the other. In contrast, in synchronous schedules the nodes first compute the message to send and once all of them have computed it, the messages are delivered and used to compute the next round, needing as much iterations as the image dimension for the information to reach the other side. This feature causes the belief propagation algorithm to converge very quickly. When the max-product algorithm converges on a graph with loops, it returns an approximate solution for the most likely labelling of the graph.

Although the algorithm is normally defined in terms of probability distribution [3], an equivalent computation can be performed with negative log probabilities, where the max-product becomes a min-sum. Using this formulation it is less sensitive to numerical artefacts and it uses the energy function more directly.

The algorithm works by passing messages around the graph defined by the four-connected image grid. Each message is a vector of dimension given by the number of possible labels. Let m_{pq}^t be the message that node p sends to a neighbouring node q at time t. When using negative log probabilities all entries in m_{pq}^t are initialized to zero and at each iteration new messages are computed in the following way:

$$m_{pq}^t(f_q) = \min_{f_p} \left(V(f_p, f_q) + D_p(f_p) + \sum_{p \in \aleph(p) \setminus q} m_{sp}^{t-1}(f_p) \right)$$

where $p \in \aleph(p) \setminus q$ denotes the neighbours of p other than q. $D_p(f_p)$ is the cost of assigning a label f_p to pixel p, which is referred as the data cost and corresponds to the matching cost computation, thus penalizing solutions which are inconsistent with the observed data. We compute this term using the Birchfield and Tomasi cost function [12] as it is insensitive to image sampling. $V(f_p,f_q)$ is the cost of assigning labels f_p, f_q to two neighbouring pixels, thus enforcing the spatial coherence, and corresponds to the support aggregation. The cost $V(f_p,f_q)$ is generally based in a measure of the difference between the labels f_p, f_q rather than on the value of the particular pair of labels. Thus, it is defined as $V(f_p,f_q) = min\ (|f_p-f_q|^k, V_{max})$, where k can take the values 1 or 2 depending on whether we consider absolute or squared differences and V_{max} is a maximum value to truncate the cost.

After T iterations a belief vector is computed for each node in the following way:

$$b_q(f_q) = D_q(f_q) + \sum_{p \in \aleph(q)} m_{pq}^T(f_q)$$

Finally, the label f_q^* that minimizes $b_q(f_q)$ is independently selected at each node.

Modification in the sphere

The key point for adapting the algorithm designed for planar standard images to omnidirectional spherical images is to consider the continuity present in the latter.

The first step that should be taken into account is that we need to submit the spherical images to a rectification process for being able to apply the algorithm on them. Going back to the point where the image rectification step was explained, we can recapitulate that, on it, the images suffer a rotation. Thus, although the original spherical image pair has periodicity in the horizontal direction, once the images have been rectified the continuity comes to be in the vertical direction. Then, we must consider as if the first and last row in the rectified images were next to each other. Therefore, in this approach, the pixels in these two rows will be considered vertically as neighbours although they are apart in the rectified images.

This means that the relation between the pixels in the first and last row should be taken into consideration also in the message passing, so they will send information directly between them too. Hence, now all the pixels in the input image will have four neighbours except those in the right and left edges which will have only three, as the continuity exists only in the vertical direction but not in the horizontal, which would correspond to the space between the epipoles in the original spherical image. In other words, the angular length of the rectified image is 2π radians in the vertical direction whereas it is π radians in the horizontal direction.

As we work with rectified images, we can apply the belief propagation as the planar technique and the fundamental operation will be essentially the same. The modification comes mainly in the message passing step, where the algorithm behaves in a different way when it gets to the bordering nodes in the vertical direction due to the already mentioned continuity. The smoothness energy computing is also modified in order to take into account the continuity in the sphere.

As in the planar case, we use the max-product message updating scheme, so we will process one direction after the other, using the recently received information for computing the next round of messages and therefore having an accelerated propagation. Then, firstly the rows are processed. In respect to the horizontal direction nothing has changed, so it performs in the same way as it did. The first node computes the message to send to its right, and the message is sent and received by the closest neighbour at its right side, which makes use of that information to compute the message to send to the next node. This process is repeated for all the nodes in the row until it gets to the last node. Then, this last node begins a backward pass, starting the same procedure but in the reverse sense, so sending messages to the left side. When it reaches the last node at the left side, the row is done. The next rows are processed until they are all done. Then it starts processing the columns.

It commences in the first node in the left column. The node computes the message to send down, which is received by the node below. This node computes the next message to send, taking into account the new information it has just received, and the next node receives the message. This is done until the last node in the column receives the message from the node above. At this point, instead of starting the backward pass, this node still computes the message to send downwards. However, this message will be sent to the first node in the column, so it also receives a contribution from its upper direction. Now, it is this point, the first one in the column, who starts the backward pass. With the information it has received from above and

the information it had previously received from its left and right (except for those nodes in the corners which only receive messages from one of their two horizontal sides) in the rows processing, it computes the message to send above. This message will be sent to the last node in the column, as it is the one next to it in the spherical projection. Then, the last node in the column continues with the backward pass, computing the next message and sending it upwards. The following nodes keep on with the process until it reaches the first node in the column. At this point, the column is done and the algorithm jumps to the next column, repeating the same process until all the columns are done. See fig.5.

Hence, the nodes in the first and last row which are connected by the continuity in the spherical image, receive now contribution from their upper and lower neighbours, which did not happen in the implementation designed for working on standard images.

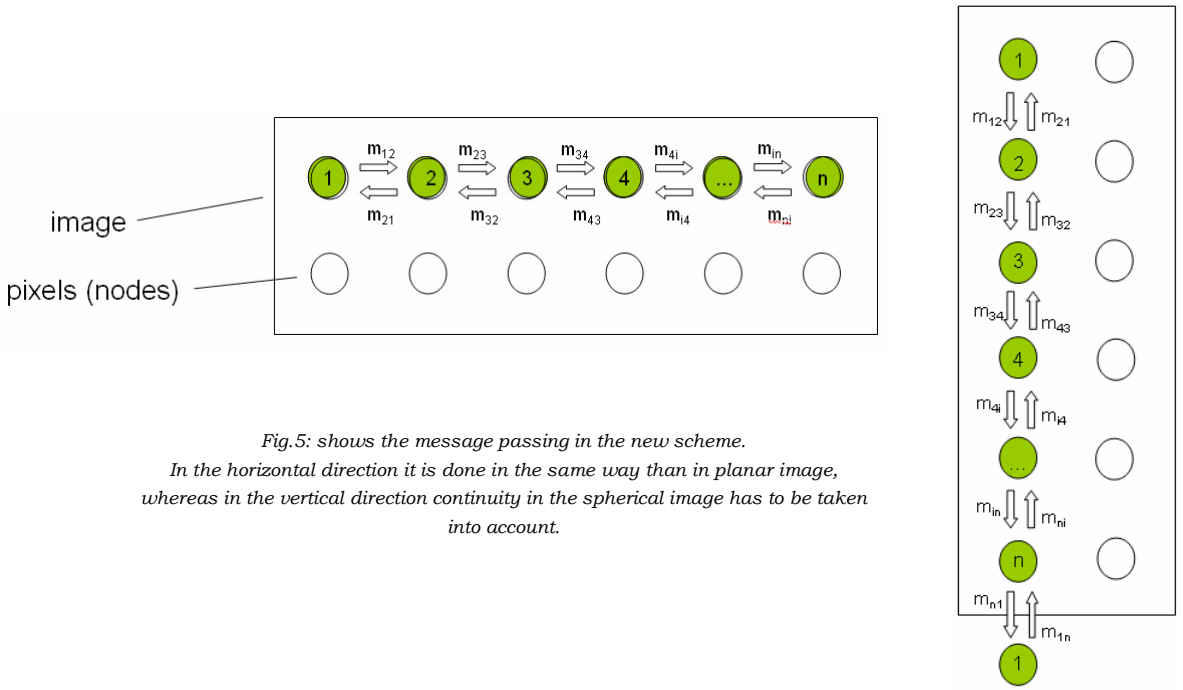

Fig.5: shows the message passing in the new scheme.
In the horizontal direction it is done in the same way than in planar image, whereas in the vertical direction continuity in the spherical image has to be taken into account.

Introducing distance weights

After having adapted the belief propagation algorithm to be applied on spherical images, we should take some profit from it. Knowing that points in the sphere are not equally separated between them, the idea arose to relate this fact to the messages sent between the pixels. Thus, the influence of the messages received by a node would depend on how apart the sender is from the receiver. The information coming from a pixel which is closer would have more relevance than that coming from a pixel which is further.

Thanks to the flexibility offered by the smoothness term in the belief propagation algorithm, the idea is to modify the smoothness function in order to introduce a weight based on the inverse of the distance between the pixels on the sphere.

Distance between points in the sphere

The shortest distance between two points on the surface of a sphere measured along a path on the surface of the sphere, as opposed to going through the sphere's interior, is called great circle distance or also orthodromic distance. This is how the distance between the nodes will be measured, as in the spherical projection the pixels lay on the surface of the sphere.

On the sphere there are no straight lines as it is the case in a Euclidean space, where the distance is measured as the length of a straight line from one point to the other. In a non-Euclidean space, straight lines are replaced with geodesics, and on the sphere the geodesics are the great circles. Great circles on the sphere are those circles whose centres coincide with the centre of the sphere.

Between any two points on a sphere which are not directly opposed to each other there is a unique great circle and the two points partition the great circle into two arcs. The length of the shorter arc is the great-circle distance between the two points. Between two points which are directly opposed to each other there is an infinite number of great circles but all them have the same length, half the circumference of the circle, or πr, where r is the radius of the sphere.

There exist different expressions to mathematically represent the great circle distance. Nevertheless, although the formulas themselves are exact for a sphere, some of them present a certain lack of accuracy due to rounding errors, which seems to be especially important for small distances. Actually, this will be our case as we will deal with the points between which the messages are sent and these are just next to each other. Having images composed of many pixels the distance between them will become really small. However, there is a special case of the Vincenty formula (which is a method to compute distances in ellipsoids) that is mostly accurate for all distances.

Let $\phi_s, \lambda_s; \phi_f, \lambda_f$ be respectively the geographical latitude and longitude of two points; $\Delta\phi, \Delta\lambda$ their differences and $\Delta\hat{\sigma}$ the spherical angular difference, then:

$$d = r\Delta\hat{\sigma} = r \cdot \arctan\left(\frac{\sqrt{(\cos\phi_f \sin\Delta\lambda)^2 + (\cos\phi_s \sin\phi_f - \sin\phi_s \cos\phi_f \cos\Delta\lambda)^2}}{\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos\Delta\lambda}\right)$$

Where *d* is the great circle distance and *r* is the radius of the sphere. In our case, as we will be interested in using the distance as a weight, we are not really interested in the real distance but

in the relation between them. For this reason, the radius of the sphere, *r*, is set to 1 to simplify the computation.

The algorithm works with rectified images which are squared, so they have the same number of pixels in the vertical and horizontal direction. However, the image is a representation of a spherical image and will be projected on the sphere. On the sphere's surface the pixels will not be at the same distance from each other since the perimeter of each circle corresponding to the rows where the pixels lie is different, being shorter at the poles and longer at the equator, but the number of pixels in each row is the same (fig.6). This means that pixels in a row will be closer to each other near the poles than near the equator. As a result, the distances obtained between two consecutive pixels in the same row will describe a sinusoidal shape along the sphere. In contrast, in the vertical direction all the circles where the pixels lie have the same length, so the pixels will be equally distributed in this direction. In a particular row or column the separation between its pixels is constant.

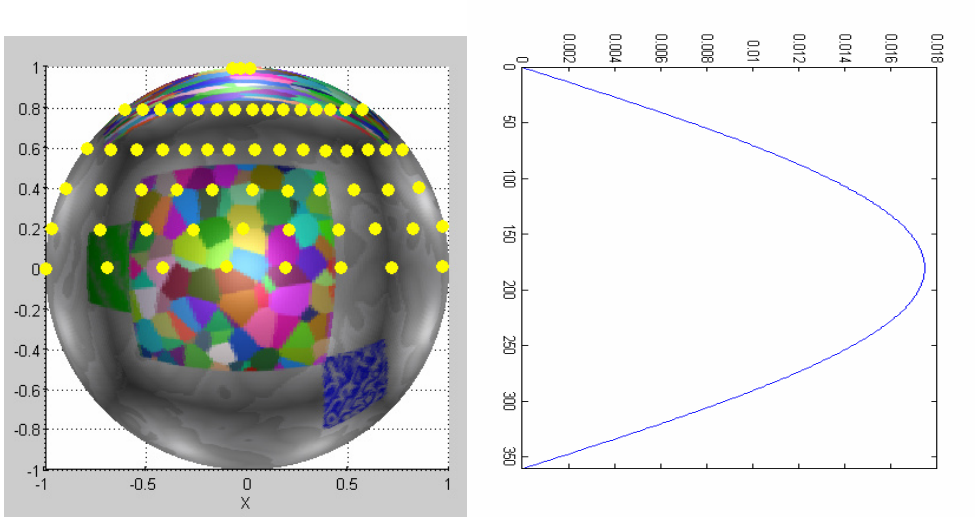

Fig.6: In the sphere, nodes in rows are equally separated in contrast to nodes in columns, which have different separations depicting a sinusoidal shape.

The formula depicted above was used to compute the distances in the first stage of the weight introduction. However, later on, we realized that in our case we are interested in the distance between two consecutive points, therefore we could find other expressions much simpler and, in addition, it would even reduce the rounding errors.

Let w, h be respectively the number of pixels of the width and height of a spherical image. Then, for the vertical direction all the parallel circles corresponding to the rows of pixels will be at the same distance between them, going from the north to the south pole. The distance between the two poles on the sphere's surface will be the half of the perimeter of the circle, πr, or π considering the radius of the sphere equal to 1, since in this direction we only consider one side of the sphere as the periodicity is on the other direction, which will wrap the sphere. Thus, the vertical distance between two pixels in the same column can be simplified to $d_v = \pi / h$.

In a similar way it is also possible to simplify the expression for the distance between pixels in the horizontal direction. If we look at the sphere, the angle θ ($0 \leq \theta \leq \pi$) is the angle indicating the height at which we consider the circle where the points in a row lie. The radius of this circle is $r' = r \sin\theta$, where r is the radius of the sphere, but as it is considered to be 1 then $r' = \sin\theta$. So, being the points equally distributed along the perimeter of this circle, the distance between two consecutive points is $d_h = \frac{2\pi \sin\theta}{w}$.

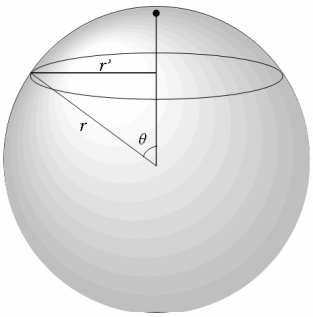

Fig.7: Radius of the circle containing a row.

Applying the weights

The aim of the weighting is to give more importance to the information one node is sending if the node is closer to the receiver in respect to the others. In consequence, the decision was taken to set the value of the weights as the inverse of the distance between the two points involved in a particular message, $w = 1/d$. Moreover, as we are interested in the relation between the weights rather than in their value itself, the horizontal weights, which are constant, were set to 1 and the horizontal weights became the quotient between the vertical and the horizontal weight, w_v / w_h. With this simple action, as the weights have a lot of decimal numbers we could avoid some computation and the running time of the algorithm was considerably reduced.

At some time, other attempts were done to evaluate what happened if the weights were taken in a different way, trying to improve the results. For example, the squared and cubed value of the previous weight was tried though no successful results were achieved. Nevertheless, the key point in the weighting stage has been where to exactly put the weights rather than their value and several approaches were attempted.

For applying the weights properly, the first to be considered was to fulfil the properties of the belief propagation algorithm, especially the condition saying that beliefs for all the labels at a particular node must sum to 1. At first, an attempt was done to put the weights directly into the messages, achieving in this way a faster propagation and probably higher contribution. However, the constraints to fulfil the condition exposed made us change our focus and finally the weights were placed in the smoothness energy term. At this place is where there is normally a gradient weight with a penalty multiplying the smoothness cost depending on the gradient value. In our case, the new weights have nothing to do with the gradient, but only with the distance between the points in the sphere. Placed here, the cost of assigning two particular labels to two particular pixels will be modified by the weights for the pixels belonging to the neighbourhood, becoming the smoothness energy term defined by:

$$E_s = \sum_{\{p,q\} \in \aleph} \omega_{pq} \cdot V\left(\left| l_p - l_q \right|\right)$$

As the algorithm works with a pair of rectified images and we compute the distances over the sphere surface, we need to know where the points in the sphere have their corresponding position in the rectified image. This is found by means of the same technique used in the rectification, i.e. a spherical rotation.

Implementation

The software

The development of the present project has been done from the work by several authors who elaborated a Markov Random Field minimization software [13]. From a pair of stereo images, it computes the disparity map corresponding to the left image assigning to each pixel a label of the matching disparity level. To achieve our goals, modifications have been done to their code which is available at http://vision.middlebury.edu/MRF and contains the implementation for running different algorithms such as Graph Cut, Belief Propagation, Iterated Conditional Modes and Tree Re-Weighted Message Passing. The Belief Propagation software was provided by Marshall Tappen [14].

Code operation

The program is executed from a command line prompt. The instruction to execute it must indicate the location of the two stereo images and the filename to store the disparity map. In the optional field, it is also possible to specify the algorithm to be used, as well as the value for some algorithm parameters or options such as writing a log file or running the program in quiet mode. Otherwise, the default values will be assumed.

```
Usage: ./mrfstereo [options] imL imR dispL

  - Reads imL and imR (in png or pgm/ppm format)
  - Runs MRF stereo
  - Writes dispL (in png or pgm/ppm format), disparities corresponding to imL

 Options:
    -n nD          disparity levels, by default 16 (i.e. disparites 0..15)
    -b             use Birchfield/Tomasi costs
    -s             use squared differences (absolute differences by default)
    -t trunc       truncate differences to <= 'trunc'
    -a MRFalg      0-ICM, 1-Expansion (default), 2-Swap, 3-TRWS, 4-BPS, 5-BPM, 9-all
    -e smoothexp   smoothness exponent, 1 (default) or 2, i.e. L1 or L2 norm
    -m smoothmax   maximum value of smoothness term (2 by default)
    -l lambda      weight of smoothness term (20 by default)
    -g gradThresh  intensity gradient cue threshold
    -p gradPenalty if grad < gradThresh, multiply smoothness (2 by default)
    -o outscale    scale factor for disparities (full range by default)
    -w             write parameter settings to dispL.txt
    -x             write timings to dispL.csv
    -q             quiet (turn off debugging output)
```

Fig.8: Usage of the MRF software.

Once it has been called, the program performs the following steps:

- Reading of the input images and parameters.
- Disparity Space Image (DSI) computation.
- Algorithm initialization.
- Energy computation and algorithm optimization.
- Disparity obtaining and writing of output files.

Some of these steps have a different implementation depending on which algorithm the program is running on. Since this project works on the Max-Product version of the Belief Propagation algorithm, only this implementation is explained in detail.

The program gets the paths of the two input images from the command line. The input images are a rectified stereo pair. The images are read and some features such as width and height are extracted. The shape of both images must be the same, otherwise the program exits.

The values of the different parameters used during the execution of the program are also read from the arguments in the command line. In the case that a parameter is not explicitly specified in the input arguments, a previously set by default value will be used.

The next step is to compute the Disparity Space Image (DSI). It computes the cost in colour differences between the images. Depending on the input parameters it can be done in simple absolute differences or by means of the Birchfield and Tomasi cost. At the same time, squared and truncated differences can be applied. Finally, the computed cost for assigning a label l to a pixel p is assigned to the data cost.

At that point the smoothness cost function is set up. The smoothness cost depends on labels for all edges and it is symmetric. In addition, for 2D grid graphs spatially varying weights for the gradient can be added. This happens if the *gradThresh* parameter is positive. In that case, the weighting cues are set to *gradPenalty* depending on whether the value of the gradient is lower than the threshold.

Then the Max-Product Belief Propagation algorithm is initialized. In this step the nodes and messages structure is created and the messages are set to their initial value, zero.

Next, the energy of the current labelling is computed, both for the data and smoothness energy terms. At this moment there are significant values only in the data term, as it has the costs previously assigned when computing the DSI. The smoothness term is zero, owing to the fact that its cost has not been evaluated yet. The value of the energies and the current timing is printed to a log file if this option is enabled.

Here, it starts the optimization step, which will deal with the smoothness energy. The next steps will be repeated for several iterations during which the algorithm will optimize the labelling for the disparities, in other words, it will minimize the energy.

It is at this point when the Belief Propagation algorithm comes on the scene. By means of the message passing method the beliefs at the nodes will be propagated between them, and from the beliefs at each node the labels will be assigned.

Each node will have a defined neighbourhood. In our case it will be composed of each node's four closest neighbours. So in this way a node will have four adjacent points from which it will receive information and to which it will send information as well. These points will be those nodes placed just next to the node we are considering at its right, left, up and down directions. Nevertheless, it will not be true for the nodes at the edges of the image, which will have three neighbours.

It is important to remark again the difference between the two main variants of the Loopy Belief Propagation (LBP). We are using the Max-Product LBP implementation, which is designed to find the lowest energy solution, whereas the other main variant, the Sum-Product LBP, does not directly search for a minimum-energy solution but computes the marginal probability distribution for each node in the graph. The authors implemented two different variants of LBP:

the Max-Product LBP is an updated version of that proposed in [14], and the Sum-Product variant is derived from the Tree Re-Weighted Message Passing implementation described in [13].

The most significant difference between the two implementations is, precisely, in the schedules for passing messages on grids. In the Max-Product implementation messages are passed along rows and then along columns. When a row or column is processed the algorithm starts at the first node and passes messages in one direction. Once the algorithm reaches the end of a row or column, messages are passed backwards along the same row or column. Instead, in the Sum-Product implementation, the nodes are processed in scan-line order, with a forward and backward pass. In the forward pass, each node sends messages to its right and bottom neighbours, and in the backward pass messages are sent to the left and upper neighbours.

There exists the possibility of adjusting the contribution of a received message with a parameter α. The value for the new message at, for example, the left of a node becomes $m^i[left] = \alpha m^i_{received}[left] + (1-\alpha) m^{i-1}[left]$, where the parameter α multiplies the value of the message just received and $(1-\alpha)$ multiplies the value the node had previously stored as information coming from its left side. By default α is set to 0.8.

As described in the sphere adaptation chapter (pp.11-12), the process of the message passing for all the rows and all the columns. After all of them have been processed, it is the moment to evaluate the beliefs that have been propagated through the network. For this purpose, the value of the belief is checked for all the possible labels at each node and the label corresponding to the maximum belief at that node is assigned to it. The belief is computed from the messages each node has got from the four directions and the local evidence, which comes from the data cost.

```
beliefVec[i] = receivedMsgs[UP][i] + receivedMsgs[DOWN][i] +
               receivedMsgs[LEFT][i] + receivedMsgs[RIGHT][i] - localEv[i];
```

Fig.9: Computing the belief at a particular node for a particular label i.

Thus, in the Max-Product implementation each node chooses independently the label with highest belief, in contrast to the Sum-Product implementation where the labelling is computed from messages. Next, being the new labelling done, it's time to update both data and smoothness energy terms.

The steps corresponding to computing and propagating the messages, updating the labelling and the new energy computation, are repeated for every iteration in the optimization loop. At the first iteration it is done, we will get the initial smoothness energy since in the first energy computation it was zero due to the fact that we had only computed the DSI corresponding to the data cost. Normally, the energies should decrease after each iteration as we are working on an energy minimization algorithm.

As LPB is used in a graph with loops, the messages may circulate indefinitely. To avoid that situation, there are two possible conditions that exit from the optimization process loop: the total energy maintains the same value for ten consecutive iterations or the algorithm reaches a previously set maximum number of iterations. By default the maximum number of iterations is set to 250 for the Max-Product LBP implementation.

Finally, when any of the conditions to exit the loop is fulfilled, it comes to the last step. It gets the disparities from the last labelling in the optimization step and writes the result in a disparity map, which is the final output image file.

Results and discussion

This section evaluates the performance of the dense disparity estimation algorithm adapted to the sphere, for both synthetic and natural omnidirectional images. It also compares the results obtained with those achieved with a graph cut algorithm adapted to the sphere.

Synthetic omnidirectional image

Firstly, we report the results obtained working with a synthetic omnidirectional image. Fig.10 shows the spherical stereo pair and the rectified images which are the input for our algorithm. Note that in the spherical images the periodicity is in the horizontal direction, whereas after the rectification step the periodicity has been moved to the vertical direction. Since this image has been created for this purpose and a ground truth disparity map is available, it will be possible to analyze the results in a numerical way in addition to a visual assessment, thus having a mathematical method to evaluate the performance.

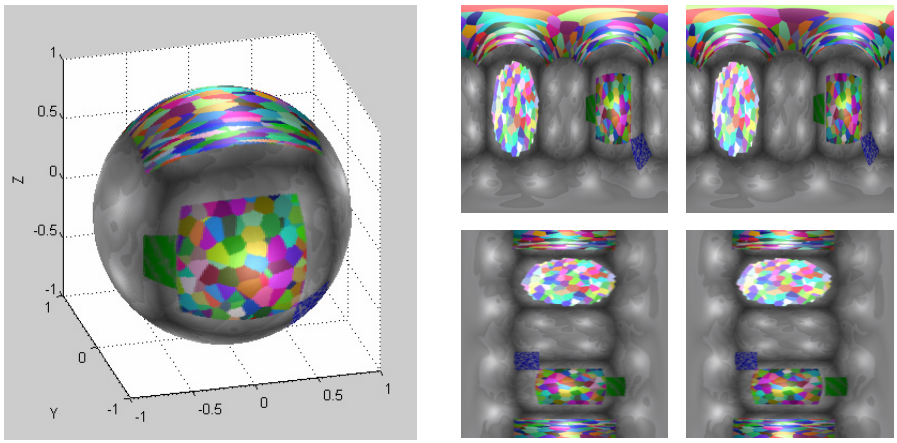

Fig.10: Projection of the synthetic omnidirectional image in the sphere (left). Spherical stereo image pair (top-right). Rectified stereo image pair (bottom-right).

The metrics defined to evaluate the performance of our algorithm, besides the logical visual appearance, are:

- PSNR of the disparity map in relation to the ground truth.
- The weighted percentage of bad matching pixels computed as $\omega_i |d_O - d_{GT}| > \delta$, where δ is assigned the values 0.5 or 1 and ω is a sinusoidal weight to give more importance to the pixels near the equator than to those near the poles.
- The PSNR of the warped image obtained using the left original image of the stereo pair and the obtained disparity map, in relation to the right original image of the stereo pair.
- The percentage of holes in the obtained warped image.

In general, the best values for each metric are obtained for different combinations of the input parameters. Thus, there is not an optimal value which behaves well in all the metrics. The following results are obtained for a combination of the input parameters, different for each scheme, for which we obtain an overall performance in all the metrics considered.

Consider the disparity maps obtained for both the unweighted and weighted scheme, as well as the ground truth, shown in fig.11. In general, the disparity maps do not look bad, although we can make some appreciations. In the non-weighted scheme there are some darker stains in the white boxes at the top and bottom. However, the most remarkable fact is that the grey circles present at the centre of the ground truth image do not appear. The PSNR computed on this

disparity map is 24.11 dB. Instead, in the weighted version these circles start to appear, since thanks to the weights there is more contribution in the horizontal direction in this area. Yet, this contribution also affects at the vertical lines on both right and left sides, which do not look now so straight as in the non-weighted or in the ground truth. Moreover, some dark points appear in the white box under the circles, though in the boxes at the top and bottom the weight contribution has helped to have more verticality as it is in the ground truth. All in all, the PSNR of the disparity map in the weighted scheme is 24.26 dB, thus it has slightly increased comparing to the unweighted. However, here the PSNR of the disparity map is not really an appropriate metric to evaluate the results, since it has been obtained from a discrete ground truth image. If we had had a continuous ground truth, the results would have been more reliable.

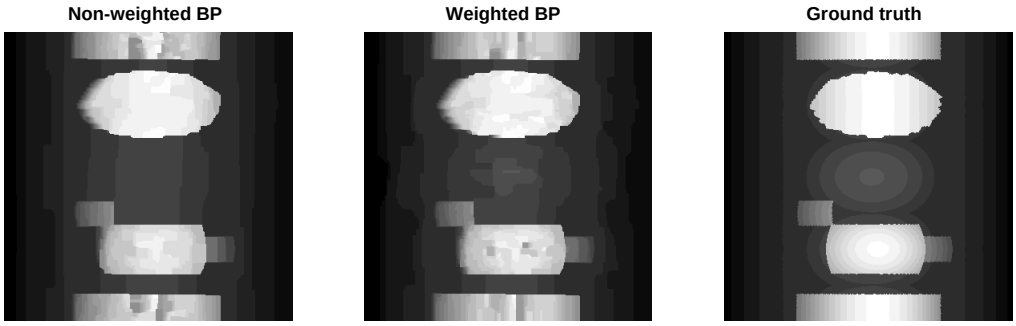

Fig.11: Disparity maps obtained using the unweighted and weighted schemes of the BP algorithm and parameters for which they are obtained and the ground truth

In fig.12 we compare the performance of our belief propagation algorithm with a graph cut version which was also adapted to the spherical framework and which is a non-weighted version. Note that, visually, the result obtained for the unweighted belief propagation is almost identical to that obtained with the unweighted graph cut algorithm. As in the unweighted belief propagation, in the graph cut version the circles in the centre of the disparity map do not appear. Thus, in the weighted version of the belief propagation is where we get a better result in this sense. Taking into account the numerical measurements, we have that for the unweighted versions the belief propagation performs slightly better than the graph cut both in PSNR and in number of bad matching pixels, computed here for δ=1. The PSNR of the belief propagation is 24.11 dB whereas for the graph cut is 24.02 dB, and the weighted percentages of bad matching pixels are 2.28 % and 2.30 % respectively. Comparing the weighted version of the belief propagation algorithm with the unweighted graph cut, there is a more significant difference in

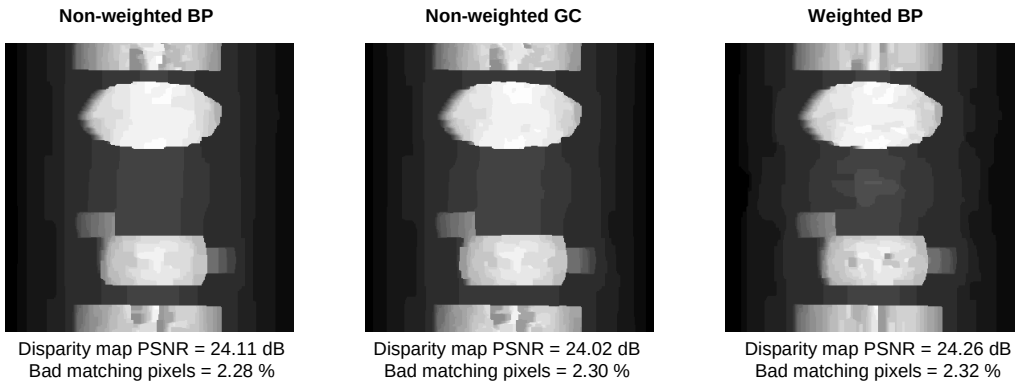

Fig.12: Disparity maps obtained using an unweighted scheme both for belief propagation and graph cuts, and the weighted belief propagation scheme.

PSNR quality than in the case of comparing both belief propagation schemes, since it increases the PSNR in 0.24 dB instead of 0.15 dB. However, in the weighted scheme the percentage of bad matching pixels has vaguely increased, as it is 2.32 %.

Comparing the energy reduction curve (fig.13) for both unweighted versions, we can appreciate how for the graph cut it decreases faster, in less iterations, whereas for the belief propagation it longs a longer time. However, this might be not a fact due to the algorithm nature, but more related to the condition for which both algorithms stop their optimization loop. Whereas in the graph cut implementation the loop is exited at the first time that the energy does not decrease, in the belief propagation implementation the condition to exit the loop is when the value of the energy remains constant for ten consecutive iterations, which might be somewhat too strict compared with the graph cut condition. In both cases there is also a previously set maximum number of iterations to stop the algorithm, in order no to be running it indefinitely. If we wanted the belief propagation algorithm to be faster, the condition to exit the optimization loop could probably be relaxed because the value of the energy is more or less stabilized after some iterations. In any case, the final energy value for both graph cut and belief propagation schemes is approximately the same.

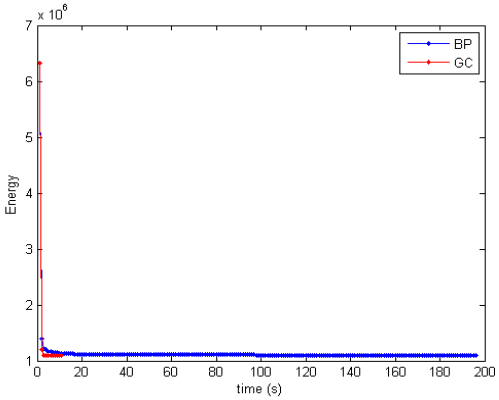

Fig.13: Energy reduction curve for unweighted belief propagation and graph cut algorithms.

The error location (fig.14) for both belief propagation and graph cut schemes is almost the same, being mostly distributed near the regions which are occluded in one view in respect to the other. Visually, the only noticeable difference is a little line of pixels corresponding to the upper border of the figure lying just below the central circles in the case of a threshold δ=1. In the case of a threshold of δ=0.5 there are more differences in the location of little groups of points, though in general both have a very similar shape.

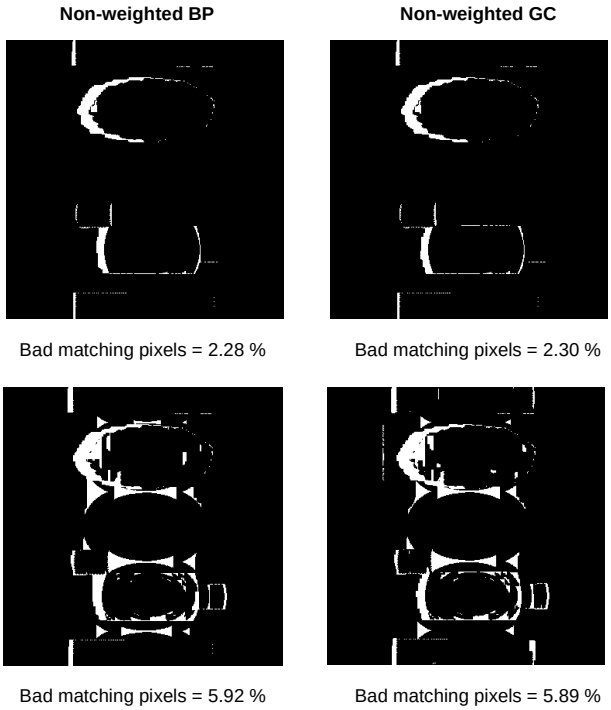

Fig.14: Error location for δ=1 (top) and for δ=0.5 (bottom).

Another way to evaluate the performance consists in obtaining the warped image from one image of the original pair and the disparity map computed by our algorithm. By means of this operation, we obtain the image corresponding to the other view of the stereo pair (fig.15). However, as there are some occluded regions, the image obtained will have some holes. For computing the PSNR on this image the hole areas are excluded.

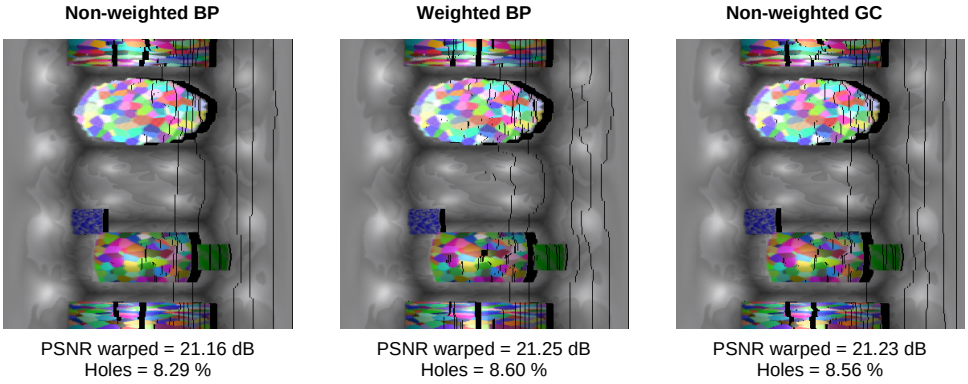

Fig.15: Warped images obtained for the three schemes.

Comparing the images obtained using this technique there are visually no big differences. The PSNR excluding holes can be slightly improved, although it means increasing the number of holes. There is usually a compromise between these two metrics. If we want to decrease the number of holes in the warped image normally the PSNR decreases. And vice versa, if the parameters are chosen to increase the PSNR quality generally the percentage of holes also increases.

Natural omnidirectional image

The performance of the developed algorithm is also tested on a natural omnidirectional image. Fig.16 shows the spherical image pair and the rectified images.

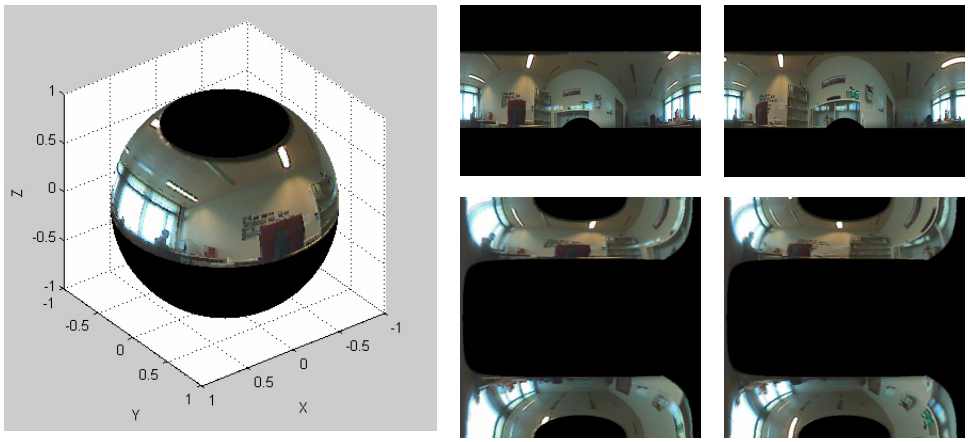

Fig.16: Projection of the natural image in the sphere (left).
Spherical stereo natural image pair (top-right). Rectified stereo natural image pair (bottom-right).

Working with this image pair we get a surprising result. Note in fig.17 that in the disparity maps obtained for the belief propagation algorithm a big strip line appears joining the two images areas from up to down. Even in the weighted version the horizontal contribution in the centre is not enough to avoid having this line. Yet, modifying the weights to have a bigger influence from the nodes on the right and left sides than using the exposed weight $w = 1/d$, it is possible to

make it disappear, though it would probably interfere in the proper computation of the rest of the disparity map. However, I must say that for the original provided implementation of the belief propagation for the planar case, hence before any modification by our part, this strip already appeared. Thus, we think this is not a problem of the spherical adaptation. Although it is a spherical image and the original code is designed for the planar case, using this image on it the strip should not appear, since it is located in the centre where it has not much to do with the spherical continuity present in the edges of the rectified image. Thus, it might be a problem that arises for this implementation for this particular image. However, using the graph cut algorithm adapted to the sphere the strip does not appear. Although we have no ground truth for this image to compute the metrics and numerically evaluate the performance of the algorithms we can affirm, from our visually assessment, that for this particular natural image the graph cut implementation works clearly better than the developed belief propagation.

Conclusions

In this project, the dense disparity estimation from omnidirectional images has been addressed. In the first part, we aimed at adapting the belief propagation algorithm to the sphere in order to work with omnidirectional images, which show an advantage in front of the standard images due to a wider field of view. The results achieved are comparable to those achieved with the graph cut algorithm adapted to the sphere, obtaining very similar results and normally even improved.

In the second part, we intend to improve the performance by introducing a weighting function in the smoothness term of the energy, which is based on the distance between pixels in the spherical image. Unfortunately, in this section the results obtained do not fulfil the expectations that were previously put on it. To make use of the distance between the nodes in the message passing process as a method to have an influence on the reliability of the information received seemed to be a good idea hoping for a significant enhancement. However, the impact of the applied weights was not that much, being the achieved improvement rather slight. Nonetheless, we think that a major improvement could be obtained by normalizing the weights properly. For this purpose we should know the conditions applied to the messages for satisfying the belief propagation statement saying that the sum of the beliefs at a node must sum to one.

On the other hand, I would like to mention that for running the algorithm a large number of parameters can be adjusted, i.e. the variables in the data cost function, the weight of the smoothness energy compared to the data energy, the maximum smoothness value or the gradient threshold and penalty among others. Furthermore, there is no explicit way to know how to find the optimal values for obtaining the best results, but rather, it is an empirical 'keep on trying' problem where the parameters are set arbitrarily and then the obtained result is evaluated. Although I have done my best, I am pretty sure there are other parameter combinations for which the results could be better than those shown.

Another fact I would like to expose is related with how the numerical results have been measured. There is always a trade-off between the different metrics considered. If the parameters are chosen in a way that improves one of the metrics usually the others are worsen. For example, if the PSNR value of the disparity map increases comparing to an other scheme, then the PSNR of the warped image decreases. And the same happens in the reverse sense. In the same way, decreasing the number of holes in the warped image decreases its PSNR. Most of the results shown here are those providing an overall performance, which could show improvements in all the metrics although these were very slight. However, if we were interested in a specific metric it would be possible to perform the search of the best parameters focused on achieving the best results on it. For example, in the situation of having the means to fill the holes in the warped image one would probably be more interested in obtaining the best quality in the warped image than in the number of holes, thus in this case the best result for the warped PSNR is 21,88 dB whereas in the one considering an overall performance it was 21,25 dB.

There are a number of possible extensions for the work developed here which might be interesting to consider in a hypothetic future improvement of the scenario. A few ideas that arose during the development refer to optimize the method by normalizing the weights based on the distance, developing a new smoothness cost function making use of the belief propagation

flexibility, finding a method to fill the holes in the warped images by means of the spherical geometry or extending to multiple view reconstructing the scene from several images.

Bibliography

[1] Y. Boykov, O. Veksler, R. Zabih. “Fast approximate energy minimization via graph cuts”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23 (11), pp. 1222-1239, November 2001.

[2] V. Kolmogorov, R. Zabih. “What energy functions can be minimized via graph cuts?”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147-159, 2004.

[3] J.S. Yedidia, W.T. Freeman, Y. Weiss. "Understanding belief propagation and its generalizations". *Exploring Artificial Intelligence in the New Millennium*, Chap. 8, pp. 239-236, January 2003.

[4] J. Sun, H. Shum and N. Zheng. “Stereo matching using belief propagation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.25, no. 7, pp. 787-800, 2003.

[5] Weiss and W.T. Freeman. “On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs”. *IEEE Trans. Information Theory, Special Issue on Codes on Graphs and Iterative Algorithms*, 47(2), pp. 723–735, 2001.

[6] P. Felzenszwalb, D. Huttenlocher. “Efficient belief propagation for early vision”. *CVPR*, pp. 261-268, 2004.

[7] S. Birchfield, C. Tomasi. “Multiway cut for stereo nd motion with slanted surfaces”. *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pp. 489-495, 1999.

[8] J. Gaspar, N. Winters and J. Santos-Victor. “Vision-based navigation and environmental representations with an omnidirectional camera”. *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 890-898, 2000.

[9] N. Winters, J. Gaspar, G. Lacey and J. Santos-Victor. “Omni-directional vision for robot navigation”. *Proc. IEEE Workshop on Omnivis00*, 2000.

[10] S. Hrabar and G.S. Sukhatme. “A comparison of two camera configurations for optic-flow based navigation of a UAV through urban canyons”. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2673–2680, September 2004.

[11] R. Hartley, A. Zisserman. “Multiple view geometry in computer vision”. *Cambridge University Press*. Chap. 9-14, pp.237-360, 2000.

[12] S. Birchfield and C. Tomasi. “A pixel dissimilarity measure that is insensitive to image sampling”. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, num. 4, pp. 401-406, April 1998.

[13] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. “A comparative study of energy minimization methods for Markov random fields with smoothness-based priors”. *Ninth European Conference on Computer Vision (ECCV 2006)*, vol. 2, pages 16-29, Graz, Austria, May 2006.

[14] M.F. Tappen, W.T. Freeman. “Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters”. *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV)*, pp. 900-907, 2003.

[15] Z. Arican, P. Frossard. “Dense disparity estimation from omnidirectional images”. 2007.

[16] S. Li, K. Fukumori. “Spherical stereo for the construction of immersive VR environment”. *Proceedings of the IEEE Virtual Reality*, 2005.

[17] A. Torii, A. Imiya, N. Ohnishi. “Two -and three- view geometry for spherical cameras”.

[18] S. Li, “Real-time spherical stereo”. *18th International Conference on Pattern Recognition*, vol. 3, pp. 1046-1049, 2006.

[19] D. Scharstein, R.Szeliski. “A taxonomy and evaluation of two-frame stereo correspondence algorithms”. *International Journal of Computer Vision*, vol. 47, nos. 1-3, pp. 7-42, 2002.

Appendix

Tables of results

Unweighted BP	PSNR disp_map (dB)	Bad matching pixels				PSNR warped (dB)	Holes (%)
		sinus weighted		not weighted			
		delta=1	delta=0.5	delta=1	delta=0.5		
m1l5	20,6236	3,1427	8,1505	3,4738	11,1590	21,8326	12,9776
m1l10	21,2759	2,7377	7,7307	2,9830	10,6443	21,6761	11,4252
m1l20	21,8807	2,4421	7,5386	2,7137	10,9390	21,4497	10,2392
m1l30	22,3197	2,3634	7,5548	2,6211	10,7492	21,3741	9,8858
m2l5	21,5049	2,6080	6,7894	2,8426	9,5725	21,6473	11,7855
m2l10	22,2373	2,3819	6,5772	2,5818	9,0856	21,4170	10,4097
m2l20	23,1490	2,3133	6,3465	2,4954	8,3773	21,3665	9,4244
m2l30	23,1475	2,2747	5,7369	2,4360	7,7477	21,2889	8,8611
m3l5	21,8131	2,5100	6,6258	2,7215	9,3966	21,5913	11,5401
m3l10	22,8983	2,3542	6,4483	2,5571	8,8488	21,3809	10,1674
m3l15	23,4905	2,3063	6,4074	2,4985	8,6528	21,2995	9,4290
m3l20	23,8198	2,2315	6,1620	2,4252	8,1674	21,2648	9,0270
m3l25	23,7890	2,2353	5,6165	2,4298	7,6219	21,2316	8,7924
m3l30	23,8444	2,2400	5,7137	2,4313	7,6991	21,1835	8,5995
m3l35	23,8649	2,2515	5,6165	2,4437	7,6019	21,1963	8,4769
m4l5	22,0620	2,4983	6,6250	2,7099	9,3958	21,5704	11,4977
m4l10	23,2588	2,3773	6,3858	2,5864	8,7863	21,3838	10,0378
m4l15	23,8337	2,3225	6,3580	2,5262	8,6034	21,2848	9,2623
m4l20	24,0212	2,2569	6,0270	2,4645	8,0324	21,2750	8,9005
m4l25	23,9337	2,2438	5,5918	2,4398	7,5972	21,2235	8,6782
m4l30	23,9978	2,2693	5,6667	2,4653	7,6520	21,1887	8,5201
m4l35	24,0659	2,2639	5,7215	2,4630	7,7068	21,1637	8,3858
m4l40	23,9952	2,2778	5,7160	2,4769	7,6605	21,1646	8,3187
m5l5	22,1835	2,4892	6,6026	2,7068	9,3735	21,5396	11,4159
m5l10	23,5793	2,3742	6,3920	2,5910	8,7924	21,3500	9,8981
m5l15	24,0111	2,2932	6,3349	2,5062	8,5802	21,3002	9,1983
m5l16	24,0928	2,2940	6,2346	2,5054	8,3488	21,2970	9,0741
m5l17	24,1392	2,2917	6,1404	2,5031	8,2392	21,2765	8,9807
m5l18	24,1405	2,2955	6,1204	2,5100	8,1906	21,2705	8,9475
m5l19	24,1701	2,2940	6,1373	2,5093	8,2076	21,2717	8,8897
m5l20	24,2286	2,2816	6,2315	2,4969	8,2369	21,2607	8,8287
m5l21	24,2404	2,2909	6,4282	2,5062	8,4336	21,2418	8,7654
m5l22	24,2049	2,2948	6,2454	2,5023	8,2508	21,2387	8,7539
m5l23	24,1248	2,2971	6,0077	2,5046	8,0131	21,2241	8,7099
m5l24	24,1394	2,3009	6,0463	2,5085	8,0517	21,2031	8,6543
m5l25	24,0931	2,2978	5,7685	2,5054	7,7739	21,2024	8,6335
m5l26	24,1256	2,2978	5,7778	2,4938	7,7631	21,1955	8,5856
m5l27	24,1496	2,2971	5,8835	2,4931	7,8688	21,1764	8,5108
m5l28	24,1529	2,2978	5,8503	2,4938	7,8356	21,1787	8,4931
m5l29	24,1372	2,3017	5,8573	2,4977	7,8426	21,1746	8,4707

m5l30	24,1466	2,3017	5,8781	2,4977	7,8634	21,1698	8,4514
m5l32	24,0952	2,2917	5,8410	2,4877	7,8264	21,1654	8,4113
m5l34	24,1158	2,2840	6,0170	2,4830	8,0023	21,1539	8,3364
m5l36	24,1104	2,2785	5,9151	2,4776	7,9005	21,1589	8,2878
m5l38	24,0810	2,2762	5,9097	2,4753	7,8951	21,1523	8,2654
m5l40	24,0767	2,2994	5,9931	2,4985	7,0375	21,1558	8,2377
m6l5	22,3487	2,4830	6,7137	2,6975	9,4846	21,5269	11,3634
m6l10	23,6944	2,3858	6,4267	2,5980	8,8272	21,3548	9,8534
m6l15	24,1542	2,3071	6,2840	2,5193	8,5293	21,2911	9,1127
m6l16	24,1755	2,3071	6,1682	2,5193	8,2824	21,2846	9,0355
m6l17	24,2847	2,3079	6,3048	2,5216	8,4035	21,2574	8,9344
m6l18	24,2736	2,3094	6,3449	2,5262	8,4151	21,2528	8,8758
m6l19	24,2743	2,3048	6,3495	2,5224	8,4198	21,2532	8,8310
m6l20	24,2769	2,2917	6,2840	2,5093	8,2894	21,2544	8,7963
m6l21	24,1348	2,3094	6,5432	2,5270	8,5486	21,2098	8,7384
m6l22	24,1068	2,3156	6,3295	2,5332	8,3349	21,2118	8,7269
m6l23	24,1297	2,3156	6,1273	2,5332	8,1327	21,2036	8,6744
m6l24	24,1504	2,3179	6,1335	2,5355	8,1389	21,1921	8,6281
m6l25	24,1328	2,3164	5,9128	2,5340	7,9182	21,1915	8,5887
m6l26	24,1622	2,3156	5,8765	2,5255	7,8619	21,1873	8,5355
m6l27	24,1790	2,3164	5,9884	2,5262	7,9738	21,1791	8,4931
m6l28	24,1797	2,3171	5,9761	2,5270	7,9614	21,1798	8,4869
m6l29	24,1719	2,3171	5,9792	2,5270	7,9645	21,1718	8,4622
m6l30	24,2083	2,3194	6,1705	2,5293	8,1559	21,1607	8,4205
m6l32	24,1827	2,3210	6,1998	2,5193	8,1852	21,1560	8,3835
m6l34	24,1969	2,3194	6,2052	2,5208	8,1906	21,1504	8,3156
m6l36	24,1780	2,3063	6,0841	2,5069	8,0694	21,1527	8,2793
m6l38	24,1826	2,3002	6,1211	2,5008	8,1065	21,1522	8,2261
m6l40	24,1450	2,3318	6,1512	2,5324	8,0957	21,1487	8,2052
m7l5	22,6175	2,5046	6,7600	2,7191	9,5309	21,5079	11,3071
m7l10	23,7946	2,3650	6,5008	2,5772	8,9012	21,3548	9,8241
m7l15	24,2425	2,3110	6,3565	2,5231	8,6019	21,2732	9,0826
m7l20	24,1935	2,3071	6,3117	2,5247	8,3171	21,2229	8,7593
m7l25	24,1751	2,3326	5,9205	2,5502	7,9259	21,1880	8,5231
m7l30	24,2653	2,3202	6,1605	2,5378	8,1458	21,1559	8,3611
m7l35	24,2310	2,3241	6,1659	2,5255	8,1512	21,1485	8,2554
m7l40	24,2129	2,3472	6,1443	2,5478	8,0887	21,1527	8,1759
m8l25	24,1626	2,3272	5,9174	2,5448	7,9228	21,1877	8,5224
m9l25	24,2306	2,3333	6,1566	2,5509	8,1620	21,1782	8,5085
m10l25	24,2432	2,4043	6,2793	2,6219	8,2847	21,1790	8,4977
m15l25	24,6432	2,4298	7,8850	2,6489	9,8904	21,1888	8,3711
m20l25	24,6603	2,4637	8,2531	2,6829	10,2585	21,1473	8,3804
nobm5l5	22,3251	2,4846	7,5664	2,8071	9,2577	21,5390	12,0833
nobm5l10	23,3052	2,3465	6,8148	2,5532	8,2546	21,4433	10,5239
nobm5l15	23,5815	2,2917	6,3719	2,4992	7,7508	21,3717	9,7114
nobm5l20	23,7065	2,2060	6,2508	2,4136	8,0494	21,3084	9,3140
nobm5l25	23,6210	2,2315	6,1767	2,4367	8,1458	21,2895	9,0610
nobm5l30	23,4475	2,1944	6,1968	2,3920	8,5278	21,2911	8,8403

nobm5l35	23,4760	2,1759	6,2485	2,3619	8,5826	21,2542	8,6713
nobm5l40	23,4234	2,2083	6,2346	2,3943	8,4190	21,2404	8,5864
with gradient							
m4l20g2p4	23,2666	2,2948	6,5039	2,5579	8,6481	21,2664	9,7407
m4l20g4p4	22,8629	2,3665	6,1667	2,5694	8,7392	21,3206	9,4923
m4l20g6p4	23,1504	2,2269	6,4568	2,4012	9,0448	21,3544	9,1867
m4l20g10p4	23,2881	2,0309	6,2500	2,1975	8,3981	21,3032	8,8488
m4l20g15p4	23,0888	2,0517	5,7353	2,2137	7,6798	21,3378	8,7423
m4l20g20p4	23,0948	2,0586	5,5949	2,2130	7,5394	21,3330	8,6435
m4l20g4p2	23,6480	2,2994	6,1605	2,5285	8,6258	21,2336	9,1821
m4l20g4p4	22,8629	2,3665	6,1667	2,5694	8,7392	21,3206	9,4923
m4l20g4p6	22,6777	2,3881	6,2292	2,5748	8,7384	21,3217	9,5957
m4l20g4p10	22,5263	2,3951	6,3356	2,5633	8,8603	21,3400	9,7191
m4l20g4p15	22,4237	2,4390	6,5100	2,5972	8,9931	21,3319	9,7847

Weighted BP	**PSNR**	**Bad matching pixels**				**PSNR**	**Holes**
	disp_map	**sinus weighted**		**not weighted**		**warped**	**(%)**
	(dB)	**delta=1**	**delta=0.5**	**delta=1**	**delta=0.5**	**(dB)**	
m1l5	20,5571	3,2724	8,3349	3,5556	11,1566	21,8771	13,1698
m1l10	21,1897	2,7824	7,6551	3,0733	11,0872	21,7559	11,6080
m1l20	21,8172	2,4367	7,7361	2,6991	11,2292	21,5293	10,4375
m1l30	22,2945	2,3395	7,3565	2,6181	10,3225	21,4232	9,9252
m2l5	21,4217	2,6219	6,8789	2,8596	9,3981	21,7311	11,9853
m2l10	22,2070	2,3310	6,4275	2,5316	8,9367	21,4820	10,4745
m2l20	22,9605	2,2785	6,4491	2,4637	8,7593	21,3893	9,5355
m2l30	23,1080	2,3179	6,0980	2,4668	8,1289	21,3443	9,0471
m3l5	21,7669	2,4877	6,8657	2,7160	9,3441	21,6294	11,6281
m3l10	22,6627	2,3326	6,3904	2,5378	8,8264	21,4393	10,2809
m3l20	23,6878	2,2840	6,3819	2,4838	8,5069	21,2970	9,1698
m3l30	23,8564	2,2600	6,0085	2,4483	8,0656	21,2463	8,7137
m4l5	21,9453	2,4985	6,8387	2,7315	9,3140	21,6140	11,5239
m4l10	23,0836	2,3372	6,3742	2,5463	8,8102	21,4405	10,1327
m4l15	23,6130	2,3241	6,4028	2,5270	8,6813	21,3223	9,4282
m4l20	24,0203	2,3148	6,4205	2,5293	8,5455	21,2733	8,9745
m4l25	24,1308	2,2762	6,2407	2,4830	8,2978	21,2605	8,7230
m4l30	24,0314	2,2724	5,9699	2,4668	8,0270	21,2156	8,5401
m4l35	24,0678	2,2631	5,4753	2,4576	7,5039	21,2093	8,4614
m5l5	22,1158	2,4807	6,8156	2,7176	9,2909	21,5873	11,4390
m5l10	23,2012	2,3951	6,3488	2,6080	8,7847	21,4355	10,0355
m5l15	23,8525	2,3364	6,4290	2,5494	8,7076	21,3106	9,2963
m5l20	24,2004	2,3171	6,2438	2,5417	8,3688	21,2852	8,8750
m5l25	24,2553	2,3210	6,2137	2,5301	8,2708	21,2466	8,5988
m5l30	24,1827	2,3140	6,0463	2,5085	8,1034	21,2172	8,4985
m5l40	24,1518	2,3657	5,7639	2,5633	7,7485	21,1819	8,3441

m6l5	22,2138	2,4769	6,7986	2,7091	9,2739	21,5788	11,3681
m6l10	23,5995	2,4066	6,4097	2,6196	8,8457	21,3982	9,8619
m6l20	24,3220	2,3380	6,2469	2,5625	8,3719	21,2472	8,7870
m6l30	24,2303	2,3758	6,2153	2,5802	8,2724	21,1903	8,4468
m6l40	24,1714	2,4105	5,9537	2,6103	7,9383	21,1792	8,2832
m7l5	22,3937	2,4961	6,8765	2,7284	9,3519	21,5730	11,3596
m7l10	23,6528	2,4120	6,5895	2,6250	9,0255	21,3960	9,7940
m7l20	24,3600	2,3580	6,3202	2,5764	8,4452	21,2335	8,7330
m7l30	24,2171	2,4637	6,4012	2,6682	8,4583	21,1701	8,3866
m7l40	24,0822	2,5401	6,1782	2,7477	8,1628	21,1637	8,2083
m8l25	24,2642	2,5201	6,4444	2,7307	8,5015	21,1956	8,4915
m9l25	24,2765	2,5340	6,5046	2,7446	8,5617	21,1901	8,4923
m10l25	24,3804	2,5386	6,7215	2,7492	8,7785	21,1948	8,4815
m15l25	24,3154	3,3781	9,1806	3,5910	11,2377	21,0562	8,1775
m20l25	24,2615	3,4498	9,6551	3,6628	11,7122	20,9914	8,1867
nobm5l5	22,0325	2,5216	8,0648	2,7500	9,9599	21,5834	12,4599
nobm5l10	23,2370	2,3951	7,0617	2,5856	8,6358	21,4557	10,5664
nobm5l15	23,5797	2,3302	6,6944	2,5231	8,1512	21,3823	9,8356
nobm5l20	23,7141	2,3032	6,3148	2,4985	7,8017	21,3310	9,3526
nobm5l25	23,7342	2,2685	6,2909	2,4606	8,0100	21,2844	9,0965
nobm5l30	23,7206	2,2731	6,2909	2,4614	8,0741	21,2623	8,9074
nobm5l35	23,4523	2,2284	6,0779	2,4120	8,2647	21,2836	8,7708
nobm5l40	23,3903	2,2333	5,9468	2,4221	8,1181	21,2860	8,7114
with gradient							
m4l20g2p4	23,4582	2,3418	6,0818	2,5478	8,5000	21,2395	9,4221
m4l20g4p4	22,7835	2,2793	6,2006	2,4560	8,7338	21,3100	9,3318
m4l20g6p4	22,9210	2,0841	6,5270	2,2315	9,0077	21,3672	9,1119
m4l20g10p4	23,1405	1,9846	6,0324	2,1281	8,2832	21,3163	8,7346
m4l20g15p4	22,9680	1,9830	5,8403	2,1204	7,7847	21,3309	8,5849
m4l20g20p4	22,9509	1,9738	5,4120	2,1165	7,3565	21,3342	8,5355
m4l20g4p2	23,8489	2,1875	5,7875	2,3858	8,2955	21,2572	8,9498
m4l20g4p4	22,7835	2,2793	6,2006	2,4560	8,7338	21,3100	9,3318
m4l20g4p6	22,6832	2,3457	6,1705	2,5085	8,7060	21,2882	9,4390
m4l20g4p10	22,3995	2,4514	6,1813	2,5525	8,7546	21,3074	9,5355
m4l20g4p15	22,3411	2,4861	6,2022	2,5787	8,7106	21,3224	9,7029