Stereo Matching and Graph Cuts

Chapter · November 2008

DOI: 10.5772/5888 · Source: InTech

3 authors, including:

Michel Devy
Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS)
123 PUBLICATIONS **1,320** CITATIONS

SEE PROFILE

Raja Chatila
Sorbonne Université
202 PUBLICATIONS **4,978** CITATIONS

SEE PROFILE

21

Stereo Matching and Graph Cuts

Ayman Zureiki [1,2], Michel Devy [1,2] and Raja Chatila [1,2]
[1]*CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse,*
[2]*Université de Toulouse; UPS, INSA, INP, ISAE; LAAS-CNRS : F-31077 Toulouse, France*

1. Introduction

The stereo matching aims to find corresponding entities between two (or more) images, i.e. entities that are projections of the same 3D object in the scene. Constraints used in stereo matching can be classified into two categories: local constraints, which rely only on a pixel and on some pixels in its surrounding, and global constraints, which must be verified by the whole pixels of a line or of the image. The local methods aim to find a matching for a given pixel without taking into account neighbour pixels correspondences. Global methods try to define a global model of the observed scene and to minimize a global cost function. They try to find the correspondences once for all pixels in one line or for all pixels in the image.
Graph cuts techniques have been recently used to solve the stereo matching problem involving global constraints. These methods transform the matching problem to a minimization of a global energy function. The minimization is achieved by finding out an optimal cut (of minimum cost) in a special graph. Different methods were proposed to construct the graph. However, when applied to minimize a global cost function in stereo vision, all of them consider for each pixel, all possible disparities between minimum and maximum values. Our contribution is a new method for constructing a reduced graph: only some potential values in the disparity range are selected for each pixel. These values can be provided by a preliminary matching process using only local constraints. We will detail how this method allows to make wider the disparity range, and at the same time to limit the volume of the graph, and therefore to reduce the computation time.
In this chapter, the stereo matching problem is defined. A brief state of the art about stereo matching is presented. The stereo matching can be solved by either local methods or global ones. Among the global methods, we give a short introduction of dynamic programming techniques to be a logic introduction of the Graph Cuts methods. We recall the definition of Graph, flow, cut, and the different algorithms to solve the problem of maximum flow. A general formalism of Relabelling problem is used to express the stereo matching as a minimization of an energy function. The implementation of both complete graph (Roy & Cox, 1998) and reduced graph (our contribution) are detailed. The two methods are compared from experimental results.

2. Stereo matching

The stereo vision is a tool to find 3D information on a scene perceived by two images (or more) acquired at the same moment from different points of view. The stereo

Source: Stereo Vision, Book edited by: Dr. Asim Bhatti,
ISBN 978-953-7619-22-0, pp. 372, November 2008, I-Tech, Vienna, Austria

reconstruction is based on the aptitude to find in each image the projection of the same object in the scene. In fact, depth information of an object is related from one side to the disparity (difference of projections of the object in both images), and on the other side, to the relative position of both cameras (baseline) and to the image resolution focal length, etc.). Thus, the stereoscopic reconstruction raises two different problems. The first is the disparity calculation, which is attached to the problem of stereo correspondence (matching). The second problem is the ability to inverse the projective geometry problem. In other words, the tridimensional reconstruction, or how to exploit disparity knowledge and relative position of the two sensors to find 3D information. The works of (Faugeras, 1993), on the projective geometry have established a solid basis for the 3D reconstruction problem. For the matching problem, there is no method sufficiently reliable, robust and effective that allows a simple use of stereo vision as a sensor of depth measurement.

Binocular stereo vision uses two images acquired by two cameras. A preliminary phase of calibration is needed to estimate the different parameters of a stereo rig: the parameters of the projection model for each camera (pinhole geometric model) and the spatial relationship between the two cameras. This knowledge allows us to calculate the 3D coordinates of a point from its projections in the two images by a simple triangulation.

Stereo matching is one of the most studied topics in computer vision since more than half a century (Julesz, 1962). Stereo matching aims at finding in the left and right images, features, which are the projections of the same entity in the 3D scene. In general, the matching problem can be seen as a minimization problem. Local approaches try to minimize separately many energy functions, representing local matching costs supposed independent between different entities to be matched: a local cost depends on similarity constraints between these entities. Global approaches try to minimize a unique energy function taking into account all matching costs: this global cost integrates local matching costs, and also compatibility costs expressing how consistent are matchings computed on a line or on the whole image. A detailed taxonomy of stereo correspondence algorithms is proposed in (Scharstein & Szeliski, 2002). The authors classify stereo matching methods with respect to four criteria: (i) The local matching cost, (ii) The aggregation area while computing the local cost, (iii) The optimization method, (iv) The method performed to refine matching results.

The method developed at LAAS since 1995 (Grandjean & Lasserre, 1995) is a modified version of the dense or pixel-based algorithm described by (Faugeras et al., 1993). This method is suitable for robotic applications, especially because it satisfies real-time constraints. It can be classified under local methods, because there is no a global optimization step to make consistent matchings of adjacent pixels. Here we propose to apply such a global optimization method as a second step of our matching algorithm, to take advantage of the global minimization while remaining within real-time constraints as much as possible. We consider that left and right images are already rectified: it allows to limit the research area in the right image, for the match of a pixel in the left image.

2.1 Local stereo matching methods

Local stereo matching methods search separately for the best match of each pixel strating from one image (e.g. the left one) without taking into account the matches of other ones. The matching cost between two pixels is based on similarity measurements of the local intensity function. Intuitively, the projections of the same 3D point will have (naturally) similar

intensities in the two images. In fact, the Lambertian model (Horn, 1986) assumes that the object surface reflects uniformly the light in all directions. Using this model, we can suppose that the corresponding pixels in both images are similar, and indeed, their neighbours are also similar, assuming that view fields between the two cameras are very close (no occlusions). A correlation measurement can calculate a degree of similarity between two point sets. Local methods try to find a match p2 in the right image for a point p1 in the left one. The correlation measurement uses information given by p1, p2 and their neighbour pixels. The pixel p1 and its neighbours form a first point set, and the point p2 and its neighbours constitute the other point set. A correlation score evaluates the similarity between these data sets.

Many local approaches use comparison windows centred on the considered pixel. Among the most known measurements, we can find: Sum of Squared Differences (SSD) (Cox et al., 1996), Sum of Absolute Differences (SAD) (Hirschmüller, 2001), Zero-mean Normalized Cross-Correlation (ZNCC) (Chen & Medioni, 1999), (Sára, 2002), etc.

Local stereo correspondence methods are in general fast algorithms, so can be used for real-time applications. However, they are exposed to many failure sources, in particular occlusions or variations of intensity between the two images. In fact, these situations can produce many false matches. In addition, because of the absence of any constraint between matches, adjacent pixels can have very different disparities, which can be particularly remarkable in scenes having vertical lines (edges of an open door for example). Global methods try to overcome these problems.

2.2 Global stereo matching methods

Global approaches try to define a global model of observed scene and to minimize a global cost function. Matches for pixels of one line or or pixels of the whole image, are searched at the same time. In a global method, the matching between a pixel in the left image and a pixel in the right image does not depend only on their neighbours, but also on the matches of their neighbours. Hence, the match of a pixel influences the matches of its neighbour pixels. This influence is modelled by regularization constraints on the matches set. Some methods are based only on the epipolar constraint to transform the bidimensional matching problem into one-dimensional problem, as in dynamic programming (Belhumeur, 1996), (Cox, 1992). Other methods address the bidimensionnal problem by taking into account, inter-lines constraints, i.e. compatibility constraints between matchings provided on every epipolar lines, as in graph cuts algorithms (Boykov et al., 1998), (Ishikawa & Geiger, 1998).

The global regularization aims to reduce the sensibility of stereo correspondence to ambiguities caused by occlusions, poor local texture or fluctuation of illumination. This improvement has a cost, which is the increasing of algorithms complexity, and in consequence, a longer execution time, in addition to some secondary effects due to this regularization (smoothing). We detail two global methods: dynamic programming and graph cuts.

Dynamic Programming:

Dynamic programming, introduced by Richard Bellman (Bellman, 1957), allows resolving optimization problems having an objective function as a sum of monotone non-decreasing functions of resources. In practice, this means that we can infer the optimal solution of a problem using optimal solutions of sub-problems. The dynamic programming applied to

stereo matching searches for a path of minimal cost through a matrix composed of possible matches. To reduce the complexity, this technique is applied on two sets of points of the same epipolar line. Thus, the stereo correspondence is applied successively to find matchings for all pixels of a line of one image with pixels located on its epipolar line in the other image (Ohta & Kanade, 1985).

To obtain a global path cost equal to the sum of the partial-paths costs, it is mandatory to use additive costs. We define the local cost for each point in the research zone as the cost of a local stereo matching (SAD, SSD, etc.). Occlusions can be taken into account, making possible to link a set of image pixels with the same pixel in the other image; penalties are considered for these relations (occlusion costs), which will be added to the global cost of any path in the matrix (Ohta & Kanade, 1985), (Cox et al., 1996). This formulation presents many inconvenient as the sensibility to the occlusion cost, the difficulty to guarantee inter-lines consistency (Bobick & Intille, 1999), and the weak application of constraints on order and continuity, that could be not satisfied.

The dynamic programming can help in finding matchings in poorly textured zones, and in solving some occlusion problems. But this method brings also some weak points, as complexity of calculation, possibility of propagation of a local error through all the research line, and non-consistency of disparity between lines.

Graph Cuts:

The majority of stereo matching dynamic programming methods try to match pixels between epipolar lines in both images without taking into account inter-lines consistency. Hence, they do not use the bidimensionnal nature of the problem. To overcome this drawback, and to take into consideration bidimensionnal continuity constraint, a solution has been proposed using the graph theory. Initially, graph cuts applied to stereo matching were proposed by (Roy & Cox, 1998), and then reformulated by (Veksler, 1999) in which the matching problem is considered as a minimization of an energy function. Afterwards, the iterative graph-cuts algorithms were introduced in (Kolmogorov & Zabih, 2001), (Kolmogorov & Zabih, 2002a), (Kolmogorov & Zabih, 2002b).

The first global method based on graph cuts for stereo correspondence (Roy & Cox, 1998), (Roy, 1999), started from the 1D formulation of the order constraint used by the dynamic programming applied separately to each image line. They tried to find a more general 2D formulation for this constraint to be applied to all lines together. They proposed a *local coherence* constraint, which suggests that the disparity function is locally smooth, which means that neighbour pixels in all directions have similar disparities. They applied this local coherence constraint by defining a disparity matching cost, which depends on intensity variations of matched pixels. In the case of two cameras, the matching cost is the squared difference of intensities.

The next step in the method proposed by Roy and Cox is to estimate the optimal disparity map over the entire image. The matching constraints are expressed in a 3D mesh composed of planes, themselves composed of an image of nodes. There is a plane for each level of disparity, and each node represents a matching between two pixels in original images. The 3D mesh is then transformed into a graph of maximal flow by connecting each node to its four neighbours in the same plane by edges called occlusion edge, and with the two nodes in the neighbour planes with edges called disparity edges. Edges are not oriented. We add two special nodes: a source connected to all nodes in the plane of minimum disparity, and a

sink connected to all nodes in the plane of maximum disparity. The weight of a disparity edge is equal to the mean value of matching costs of the two nodes. For occlusion edges, the weight is multiplied by a constant to control the smoothness of the optimal disparity map. A graph cut will separate the nodes in two sub-sets: the optimal disparity map is constructed by the assignment of each pixel with the bigger value of disparity for which the corresponding node is still connected to the source.
(Ishikawa & Geiger, 1998) pointed out that the method of Roy and Cox can deal only with convex maps. Thereby, it can only take into account linear penalties on disparity, which may lead to not very good results due to an over smoothing of disparities. They proposed a novel graph with oriented edges, and then it is possible to reinforce the constraint of uniqueness and order. However, their method is also weak at discontinuities because of linear penalties.
Sub-optimal algorithms: (Boykov et al., 1998; Boykov et al., 1999) proposed another method to resolve the stereo corresponding using graph cuts. The authors showed that the problem of stereo corresponding could be formulated by a *Markov Random Field* (MRF). They showed that the estimate Maximum A Priori (MAP) of such MRF, could be obtained by a minimal multi-way cut, using a maximum flow. The advantage of such a method is that it accepts non linear penalties for discontinuities, and then it gives more precise disparity maps especially near objects' edges. As the general problem of minimal multi-way cut is NP-complete (Dahlhaus et al., 1994), Boylov et al. decided to introduce an approximated algorithm, which can resolve iteratively some sub-problems until convergence. (Kolmogorov & Zabih, 2001) have continued the works of Boykov trying to ameliorate the energy function to represent more explicitly the occlusion. The sub-optimal approach has a wider application spectrum than the one proposed by Roy and Cox or by Ishikawa and Geiger. However, it is iterative and sub-optimal, and then its convergence speed and the quality of the obtained minimum must be supervised.
In spite of their good results, methods based on graph cuts are disadvantaged by several limitations. Firstly, because of using the bidimensionnal continuity constraint, these methods can cause an excess of regularization (over smoothing), which can appear as the effects of a local filter. Secondly, as the penalty function (cost of attributing different disparities to neighbour pixels) is not always convex, this returns the minimization of energy function to be a NP-complete problem (Kolmogorov & Zabih, 2001), and in consequence, the obtained solution is only an approximation of the exact one.

3. Minimum cut problem

3.1 Graph definition

A graph is a set of vertices connected by edges. A binary relation between vertices, called adjacency relation, defines the connection. A graph **G** be defined as a pair (**V**,**E**), where **V** is a set of vertices, and **E** is a set of edges between the vertices $\mathbf{E} = \{(u,v) \mid u, v \in \mathbf{V}\}$. If the graph is undirected (see figure 1), the adjacency relation is symmetric, or $\mathbf{E} = \{\{u,v\} \mid u, v \in \mathbf{V}\}$ (sets of vertices rather than ordered pairs). In a directed graph (cf. figure 2), the edges are ordered, the edge (u,v) is different from the edge (v,u). When an edge e=(u,v) connects the vertices u and v, these two vertices are adjacent, and called the extremities of the edge e. Two edges are called adjacent if they have in common one vertex. A weighted graph is a graph having a weight, or a number, associated with each edge. Some algorithms require all weights to be non-negative, integral, positive, etc.

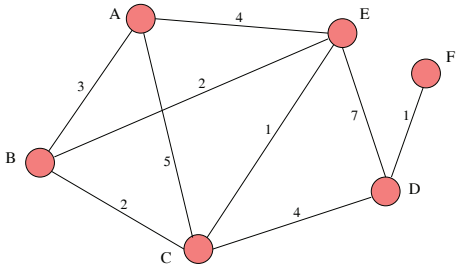

Fig. 1. Weighted Undirected Graph

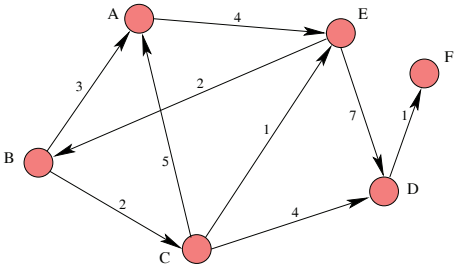

Fig. 2. Weighted Directed Graph

3.2 Graph representation

A graph can be represented by two data structures: adjacency-matrix or adjacency-list.

Adjacency-matrix Representation:

A directed graph with v vertices can be represented by v x v matrix, where the entry at (i,j) is 1 if there is an edge from vertex i to vertex j; otherwise the entry is 0. A weighted graph may be represented using the weight as entry. An undirected graph may be represented using the same entry in both (i,j) and (j,i) or using an upper triangular matrix. Figure 3 illustrates a representation of the directed graph of the figure 2 by an adjacency-matrix.

Adjacency-list Representation:

A directed graph with v vertices can be represented using a set of v lists of vertices. List i contains vertex j if there is an edge from vertex i to vertex j. A weighted graph may be represented with a list of (vertex, weight) pairs. An undirected graph may be represented by having vertex j in the list for vertex i, and vertex i in the list for vertex j. Figure 4 illustrates a representation of the directed graph of the figure 2 by an Adjacency-list. The arrow (->) means a link in a list. In general, the adjacency-list representation is more compact for a sparse graph.

	A	B	C	D	E	F
A	0	0	0	0	1	0
B	1	0	1	0	0	0
C	1	0	0	1	1	0
D	0	0	0	0	0	1
E	0	1	0	1	0	0
F	0	0	0	0	0	0

Fig. 3. Adjacency-Matrix Representation

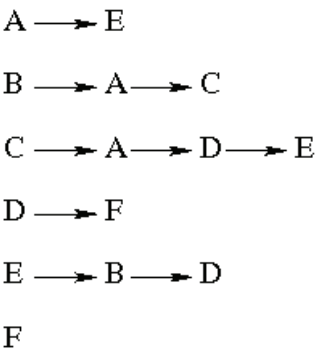

Fig. 4. Adjacency-List Representation

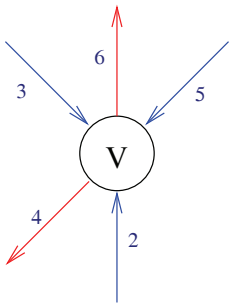

Fig. 5. Entering edges (blue) and leaving edges (red).

3.3 Graph cut definition

Let **G** = (**V**,**E**) denotes a graph, with **V** the set of vertices and **E** the set of edges. A cut is formed by partitioning the nodes of a graph into two mutually exclusive sets **S** and **T**. An

edge between a node in one set and a node in the other is said to be crossing the cut and called a **cut edge**. In weighted graphs, the weight, or the capacity, of a cut is the sum of weights of the edges that have exactly one endpoint in each of the two sets. The problem of finding the minimum weight cut in a graph is called Minimum Cut Problem.

3.4 Minimum s-t cut problem

In a directed graph **G** = (**V**,**E**), the edges having the same beginning vertex v are called leaving edges of v, and the sum of their capacities (weights) is called leaving degree of vertex v. The edges having the same arriving vertex v are called entering edges, and the sum of their capacities (weights) is called entering degree of vertex v. See figure 5.

$$\text{entering degree of v} = \sum_{(u,v)\in E} c(u,v) \tag{1}$$

$$\text{eaving degree of v} = \sum_{(v,w)\in E} c(v,w) \tag{2}$$

We call a **source** s a vertex in a directed graph with entering degree equal to zero, and a **sink** t is a vertex with zero leaving degree. The s-t cut is the problem of finding a cut in the graph to separate the source s from the sink t. A minimum s-t cut is a cut with a minimum weight separating the vertices into two sets, the source s will be in one of them and the sink will be in the other.

3.5 Flows in graph

It is frequently common to introduce flows in a graph by the example of water conduit in pipe networks.

Water flow in networks:

Suppose we have a water source, a sink and a network of pipes relating them. We accept that the source can give (and the sink can receive) an unlimited amount of water. Find the maximum flow is the research for the maximum amount of flow capable to pass from the source into the sink via the network. Supposing the unlimited capacity for the source and the sink, the problem is only constrained by the capacity of the network. The capacity of each pipe is proportional to its diameter. In this network, there will be obstructions in some pipes. To pass from to the source to the sink, water must absolutely take one of these obstructed pipes. In an intuitive case, if all the pipes are full, the flow is equal to the sum of their capacities. In particularly, the obstruction corresponds to a set of pipes, which separates the source from the sink, and the sum of its capacities is equal to the maximum flow capable to pass from the source to the sink. Once the flow is maximal, we are in the situation where all the pipes of obstruction are full (saturated).

Mathematical definition:

A flow in a weighted graph **G**=(**V**,**E**), where each edge e=(i,j) has a real positive capacity c(e), is a map Φ between the edges set **E** and the set of real numbers.

A flow $\Phi : V \times V \rightarrow R$

- $\Phi(i,j) = -\Phi(i,j)$
- $\sum_{i \in V - \{s,t\}} \Phi(i,j) = 0$: Flow conservation: what goes into a vertex is equal to what goes out of it (except for the source and the sink)
- $\Phi(i,j) \leq c(i,j)$ for all $(i,j) \in E$: Flow in an edge is less or equal to its capacity

- $c(i,j) = 0$ for all $(i,j) \notin E$

An edge is saturated if the flow in it is equal to its non-zero capacity:

$$e=(i,j) \text{ is saturated } <==> (c(i,j) = \Phi(i,j) \text{ and } c(i,j)>0)$$

3.6 Residual graph

Let Φ be a flow in the graph $\mathbf{G}=(\mathbf{V},\mathbf{E})$. The residual graph $\mathbf{G_r} = (\mathbf{V}, \mathbf{E_r})$ is constructed in the following way:

$\forall\ (i,j) \in \mathbf{E}$ **:**

$$\text{If } \Phi(i,j) < c(I,j) \text{ then: } (i,j) \in \mathbf{E_r} \text{ and } c_r(i,j) = c(i,j) - \Phi(i,j)$$

$$\text{If } \Phi(i,j) \geq 0 \quad \text{then: } (j,i) \in \mathbf{E_r} \text{ and } c_r(j,i) = \Phi(i,j)$$

Thus, for each edge in the graph **G** with non-zero capacity c(i,j) and with a flow Φ (i,j), there are two edges in the residual graph $\mathbf{G_r}$ with capacities $c_r(i,j) = c(i,j) - \Phi$ (i,j) and $c_r(j,i) = \Phi$ (i,j). Figure 6 represents a maximum flow in a graph, and the construction of the residual graph illustrated in the figure 7.

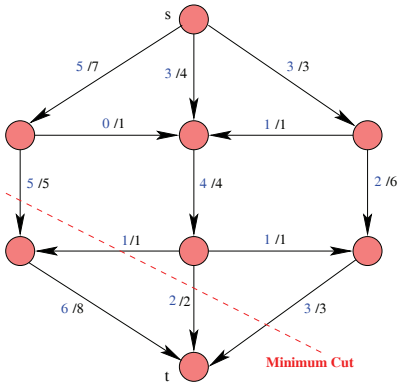

Fig. 6. Minimum Cut in a graph

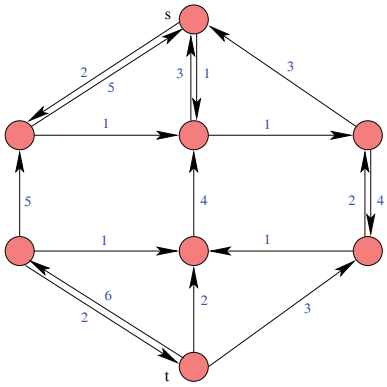

Fig. 7. Residual Graph

Augmenting Path:

An Augmenting Path in the residual graph $\mathbf{G_r}$ is a path from the source s to the sink t composed of residual edges with non-zero residual capacities.

3.7 Solutions of the minimum s-t cuts problem

The minimum s-t cut problem is traditionally resolved using the maximum flow algorithms in networks. Let s be the source and t the sink in a graph **G**, the maximum flow – minimum cut theorem says:

$$\text{Max Flow } (s,t) = \text{ Min Cut } (s,t)$$

This theorem says that the maximum value of a flow in a graph is equal to the minimum value of an s-t cut. (Ford & Fulkerson, 1962) have proved that the flow from the source s to the sink t will cause the saturation of a set of edges dividing the vertices into two sets **S** and **T**, with s in **S** and t in **T**. In other words, the theorem states that the maximum flow in a network is

dictated by its bottleneck. Between any two nodes, the quantity of material flowing from one to the other cannot be greater than the weakest set of links somewhere between the two nodes. This theorem links the value of the maximum flow with the value of the minimum cut, but it does not specify any relation between the minimum cut value and the set of cut edges. However, once we have a maximum flow, we can find the set of edges forming the minimum s-t cut. The algorithm 1 illustrates a procedure to obtain the minimum s-t cut.
The figure 6 illustrates a minimum cut in a weighted directed graph. The weights of edges are in black, and the flow values are in blue. The dashed red line crosses the edges of the minimum cut. In this example, we notice that the value of the maximum flow is 11, and the weight of the minimum cut is 11.

1. Resolve the Maximum s-t Flow problem.
2. Find the set of vertices **S** reachable from the source s in the residual graph.
3. Find the set of vertices **T** non-reachable from the source s in the residual graph.
4. Consider the edges starting from a vertex in **S** and ending in a vertex in **T**.
   The set of these edges forms the minimum s-t cut.

Algorithm 1. An algorithm of minimum s-t cut.

3.8 Maximum flow algorithm

We find in the literatures two approaches for solving the maximum flow problem (Cormen et al., 2001). The first is the augmented path algorithm proposed by (Ford & Fulkerson, 1962), and the second is the Push-Relabel algorithm.

Augmented path algorithm:

(Ford & Fulkerson, 1962) have developed a solution of the maximum flow problem based on the augmented path. In fact, if there is an augmented path in the residual graph $\mathbf{G_r}$, then the flow is not maximal. The algorithm commences by finding an augmenting path, next, it adds the capacity of this path to the flow, and then updates the residual graph $\mathbf{G_r}$. This operation is repeated until there is no augmented path.

G=(V,E) : a graph, **V** is the set of vertices, and **E** is the set of edges.
s is the source, t is the sink
$\forall$ (u,v) $\in$ **E** :
  $\Phi(u,v) \leftarrow 0$
  $\Phi(v,u) \leftarrow 0$
Construct the residual graph $\mathbf{G_r}$
while (there is an augmenting path p in the residual graph $\mathbf{G_r}$ between s and t**)**
begin
  $c_\Phi(p) = \min\{ c_r(u,v) : (u,v) \in p\}$: the capacity of the augmented path.
  for each (u,v) $\in$ p
  begin
    $\Phi(u,v) \leftarrow \Phi(u,v) + c_\Phi(p)$
    $\Phi(v,u) \leftarrow \Phi(u,v)$
  end
  Update the residual graph $\mathbf{G_r}$
end

Algorithm 2. Maximum Flow: the augmented path algorithm.

Push-Relabel algorithm:

The idea of the Push-Relabel algorithm can be clarified by the following example. Suppose that the graph can be represented as follow: the source s is on the top and the sink t is at the height zero (bottom). We send water from the source down to the sink. In each vertex, water in excess will flow to the vertices at smaller height, constrained with the capacity of the edge relating the two vertices. After each operation, water will be in a lower vertex, and the excess is reduced in the upper vertex. We track water until arriving to the sink t. The algorithm stops when there is no excess in any vertex.

The **Height** of a vertex: We observe that the longest path from the source to the sink contains at maximum V vertices. Thus, we can assign a height to each vertex in the following way:

- Height(s) = V
- Height (t) = 0
- $\Phi(u,v)>0$ ==> Height (u) > Height (v)

In other words, the heights of nodes (except s and t) are adjusted, and flow is sent between vertices, until all possible flow has reached t. Then we continue increasing the height of internal nodes until all the flow that went into the network, but did not reach t, has flowed back into s. In order to introduce the notion of excess, we define first the preflow.

Preflow Definition:

A preflow is similar to a flow with the exception that the total amount which can flow into a vertex can be bigger than the flow out of the vertex (the principle of flow conservation is no longer respected). The notion of preflow is introduced by (Karzanov, 1974). Let **G=(V,E)**be a graph. A preflow is a map verifying:

Un preflow $P : VxV \rightarrow R^+$:

- $\forall (u,v) \in \mathbf{E} : P(u,v) \leq c(u,v)$
- $\forall (u,v) \in \mathbf{E} : P(v,u) = -P(u,v)$
- $\forall u \in V\text{-}\{s\} : \sum_{(w,u)\in E} P(w,u) - \sum_{(w,u)\in E} P(u,w) \geq 0$

Let $\epsilon(u) = P(V,u)$, $\epsilon(u)$ is called the excess of vertex u, which is the difference between the in and out flows of the vertex u. A vertex $u \in V\text{-}\{s,t\}$ is called overflowed if $\epsilon(u) > 0$. The algorithm 3 gives a general implementation of the Push-Relabel algorithm, which uses the Push and the Relabel operations defined below (Goldberg & Tarjan, 1988).

```
G=(V,E) : a graph, V is the set of vertices, and E is the set of edges.
          s is the source, t is the sink
while (there is a vertex v having an excess (ϵ(v) > 0))
begin
    Select the vertex v.
    do a Push operation (Push(v))
    or do a legal Relabel operation (Ralabel(v))
end
```

Algorithm 3. Maximum Flow: General algorithm of Push-Relabel.

Push operation:

A push from u to v means sending a part of the excess flow from u into v. Three conditions must be met for a push to take place:

- $\epsilon(u) > 0$: there is an excess in the vertex u, which means that the flow into this vertex is bigger than the flow out of it.
- $c(u,v) - \Phi(u,v) > 0$: the edge (u,v) is not saturated.
- Height(u) > Height(v). The vertex u is higher than the vertex v.

When these conditions are verified, we send a flow equal to $\min(\epsilon(u), c(u,v) - \Phi(u,v))$.

Relabel operation:

Relabelling a vertex u means increasing its height until it is higher than at least one of the vertices it has available capacity to them. Conditions for a relabel:

- $\epsilon(u) > 0$: There must be an excess in the vertex in relabelling.
- Height(u) ≤ Height(v) for all v such that $c(u,v) - \Phi(u,v) > 0$. The only vertices we have available capacity to are higher.

When we re-label a vertex u, we adjust its height to be equal to the smallest value verifying Height(u) > Height(v) for all the vertices v having $c(u,v) - \Phi(u,v) > 0$.

Complexity of the maximum flow algorithms:

- The complexity of the augmenting path algorithm is O(e*MaxFlow) (Ford & Fulkerson, 1962), where e is the number of edges in the graph, and MaxFlow is the value of the maximum flow.
- The Push-Relabel algorithm is one of the most efficient algorithms to compute a maximum flow. The general algorithm complexity is $O(v^2e)$ (Cormen et al. 2001), where v is the number of vertices in **V** and e is the number of edges in **E**. An implementation with FIFO vertex has a complexity of $O(v^3)$ (Cormen et al. 2001). The theoretical complexity of the algorithm proposed by Goldberg is $O(v\ e \log(v^2/e))$ (Goldberg & Tarjan, 1988).

In our work, we use an implementation of Push-Relabel algorithm proposed by (Goldberg, 1985), which is included in *Boost Graph Library*[1].

Note that (Boykov & Kolmogorov, 2004) have proposed a new algorithm, and they have proved that for typical graphs in vision, their algorithm is about 2 to 5 times faster than the other algorithms including the approach Push-Relabel. Their algorithm belongs to the group of algorithms based on augmenting path. The algorithm starts by building search trees for detecting augmenting paths. In fact, the good performance is due to the reuse of these trees and never restarts building them from scratch.

4. Stereo matching using graph cuts

4.1 Labelling problem

Many problems in vision can be formulated as a labelling problem. In such a problem, we have a set of sites and a set of labels. Sites represent features extracted from the image (pixels, segments, etc.), for which we want to estimate some quantity. Labels represent the quantities associated to these sites: intensity, disparity, region number, etc. Let P={1,2,…,n} be a set of n sites, and $L=\{l_1, \ldots, l_k\}$ be a set of k labels. Labelling is a map from P into L:

$$f : P \dashrightarrow L : \ s_p \dashrightarrow \ f_p = f(s_p)=l_i \tag{3}$$

[1] http://www.boost.org/libs/graph/doc/index.html

Thus, $f(P) = \{f_1,\ldots, f_n\}$. We assign an energy function to the labelling map: a general form of energy function can be written:

$$E(f)=E_{data}(f)+\lambda E_{prior}(f) \quad (4)$$

The first term $E_{data}(f)$ represents the intrinsic data energy, which translates the constraints of associating labels to data. The second term $E_{prior}(f)$ aggregates the extrinsic energies (*prior energy*) which translate the constraints defined by the prior information. The constant λ can control the relative importance of the two terms; bigger values of λ give more importance to prior information.
The energy function $E_{data}(f)$ associates an important cost for association data/label, which are less pertinent.

$$E_{data}(f)=\sum_{p\in P} D_p(f_p) \quad (5)$$

Where $D_p(f_p) \geq 0$ measures the cost of assigning the label f_p to the site p.
The prior energy function $E_{prior}(f)$ will assign an important cost to the associations f_p not compatibles with prior information. The choice of this function depends on the type of the problem, but in general, a type of prior energy expresses the smoothing constraint. This constraint is very famous in computer vision, and it is well adapted when the estimated quality changes slowly everywhere or nearly everywhere: in 3D, this corresponds to the hypothesis that the world is continuous piecewise. This hypothesis is considered by taking prior information of smoothing type E_{smooth}.
To formulate the smoothing energy, we need to model how the pixels interact between them: often it is sufficient to express how a pixel interacts with its neighbours. Let N_p be the set of neighbour pixels to the pixel p, and N be the set of neighbour pairs {p,q}: N is named neighbourhood system. Smoothing energy can be written:

$$E_{smooth}(f)=\sum_{\{p,q\}\in N} V_{\{p,q\}}(f_p,f_q) \quad (6)$$

Where $V_{\{p,q\}}(f_p,f_q)$ is the neighbourhood interaction function: this function attributes high penalties to the pair {p,q} if the pixels p and q have different labels. The form of $V_{\{p,q\}}(f_p,f_q)$ determines the type of prior smoothing. With these notations, the global smoothing energy is the sum of neighbourhood interaction functions of all neighbour pixels. Often we choose $V_{\{p,q\}}(f_p,f_q)$ as:

$$V_{\{p,q\}}(f_p,f_q)=u_{\{p,q\}}\,V(f_p,f_q) \quad \text{with } u_{\{p,q\}} \in R^+ \quad (7)$$

Where V is a homogeneous potential, and $u_{\{p,q\}}$ is a multiplying term depending on the pair of pixels. The figure 8 shows a linear potential, $V(f_p,f_q)= |\, f_p - f_q \,|$.
In stereo vision, the term $u_{\{p,q\}}$ is often a decreasing function of the norm of the gradient between the sites p and q, which favours the coincidence of disparity discontinuities with the contours of the reference image. This choice is translated by the map $u_{\{p,q\}} : u_{\{p,q\}} = U(|I_p - I_q|)$. The term $u_{\{p,q\}}$ represents the penalty of assigning different disparities for neighbour pixels p and q. The value of this penalty must be small for a pair {p,q} which is on a contour, and therefore with a large difference of intensity $|I_p - I_q|$. In practice, we use a decreasing empiric function U(.). The figure 9 illustrates the function $u_{\{p,q\}}$.

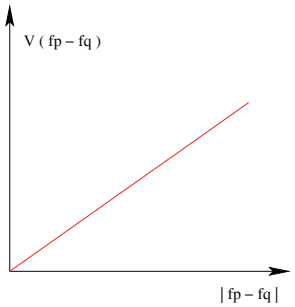

Fig. 8. Function of linear potential

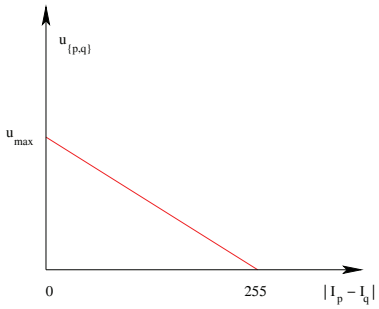

Fig. 9. The function $u_{\{p,q\}}$.

4.2 Energy and graph

The principle of graph cuts methods is to create a graph in such a way a cut represents a function and the weight (cost) of this cut represents the energy value of this function. The value of the minimum cut will be equal to the minimum of the energy function.

Starting from the general formulation of labelling problem, let us consider the linear potential map:

$$V_{\{p,q\}}(f_p,f_q) = u_{\{p,q\}} \ |f_p - f_q| \tag{8}$$

This gives the global energy:

$$E(f) = \sum_{p \in P} D_p(f_p) + \lambda \sum_{\{p,q\} \in N} u_{\{p,q\}} \ |f_p - f_q| \tag{9}$$

We will show how to construct a graph associated to an energy function, and then how to obtain the minimum value of this energy function by finding the minimum cut of the graph. This method is due to (Roy & Cox, 1998), but it profits of reformulations proposed by (Veksler, 1999) who put in evidence the energy to be minimized. Our description of the graph will be based on this formulation. First, we will explain the construction of a full graph, and then we explain our contribution in reducing its size to accelerate the algorithm.

4.3 Building a complete graph

We aim to construct a graph **G=(V,E)**, where the set of vertices **V** has two special vertices: a source s and a sink t, and in which a minimum cut will represent a minimization of the stereo matching cost. The graph is constructed as follow:

- Let k be the number of possible matches (in stereo vision, k is given by the range of disparity, which is related to the minimum and maximum depths in the scene). For example, for a disparity range [0,40] , the value of k is 41.
- To each pixel p in the image, we associate a chain composed of k-1 nodes (sites), noted $p_1, p_2, \ldots, p_{k-1}$. These nodes are connected by edges called *t-links* or disparity edges, and noted $\{t_1^p, t_2^p, \ldots, t_k^p\}$, where $t_1^p = (s, p_1)$, $t_j^p = (p_{j-1}, p_j)$, and $t_k^p = (p_{k-1}, t)$. The t-links of a pixel are a set of edges linking the source to the first node, and then link this node to the next one, until linking the last node in the chain to the sink. For a disparity range of k values, and for each pixel p, we will have k-1 nodes in the chain and k-2 t-

links between these nodes, and two other t-links, the first with the source and the other with the sink. Thus, we have in total k t-links. The capacity (weight) of each of these t-links is equal to the cost of matching of the pixel to a corresponding one in the other image with a disparity equal to the zero-index of the t-link in the chain.

- The capacity of the t-link number j is set to $K_p+D_p(l_j)$, with $D_p(l_j)$ is the cost of matching of the pixel p with a disparity l_j, and K_p is a constant satisfying the constraint 10.

$$K_p > (k-1) \sum_{q \in N_p} u_{\{p,q\}} \tag{10}$$

In fact, the constant Kp is chosen to be bigger than the sum of capacities of penalty edges (n-links, cf. below) which have one extremity in the chain of nodes associated to the pixel p.

- To each pair of neighbour pixels p and q, we link the corresponding chains with edges called *n-links* or penalty edges. The n-link edge at the level $j \in \{1, 2 \ldots k-1\}$, is noted $\{p_j,q_j\}$, has a capacity equals to $u_{\{p,q\}}$. The figure 10 illustrates the nodes related to one pixel, the t-links (which are coloured in blue) and the n-links (in green).

The capacity of an s-t cut of a graph is the sum of capacities of all cut edges. Due to the construction method of this graph, a cut capacity will be composed of two parts: the first is the sum of capacities of the cut t-link edges, and the second is the sum of capacities of cut n-link edges. In fact, the addition of the constant K_p allows us to assure the uniqueness of the cut of each t-link chain, see (Roy & Cox, 1998) for a proof.

Another method can assure the uniqueness of cutting the chain of t-links in exactly one edge is proposed by (Ishikawa, 2000). Their graph is very similar but it is oriented, and they add to the chain of pixel p another inverse chain with infinite capacity.

$$E(cut) = \sum_{p \in \mathrm{Image}} (K_p + D_p(f_p)) + \lambda \sum_{\{p,q\} \in N} u_{\{p,q\}} \, | f_p - f_q | \tag{11}$$

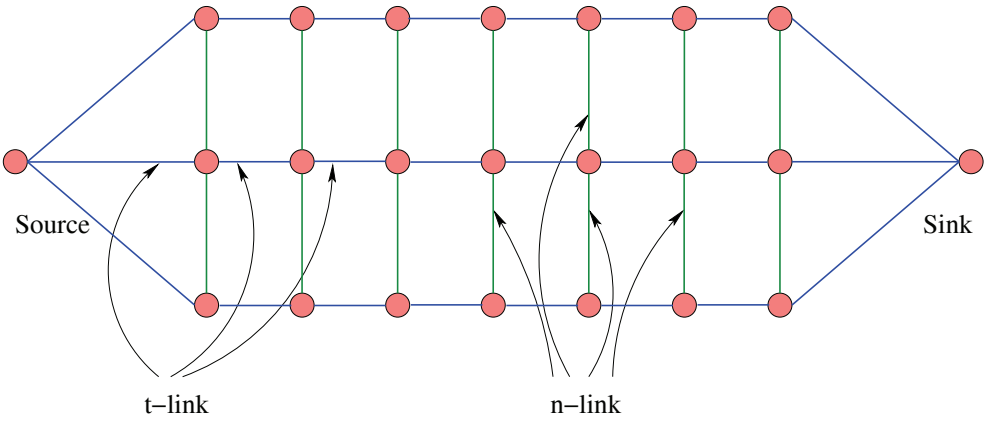

Fig. 10. The t-links and the n-links.

A graph cut consists in splitting the graph into tow parts. The cut t-link edges form the surface of searched depth, and this can allow the assignment of a disparity to each pixel. The problem of graph cut can be solved using the maximum flow (as seen before). To determine the disparity of a pixel p, the minimum cut will pass through one (and only one) t-link of the

chain associated to the pixel p. The index (zero-based) of this edge in the chain allows finding the disparity of this pixel. Hence, for a disparity range [d_{min}, d_{max}], a t-link of index j corresponds to a disparity of (d_{min} +j).

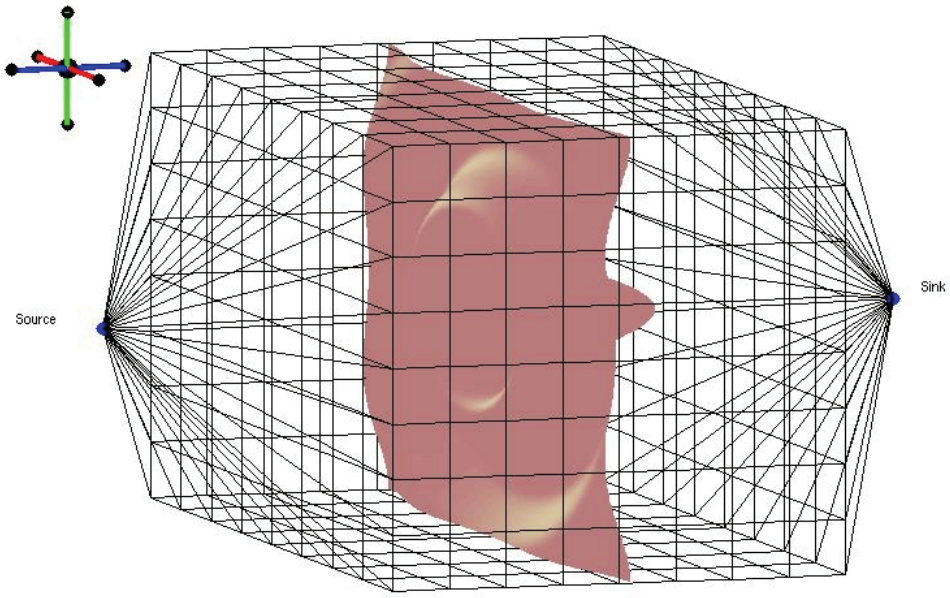

Fig. 11. The disparity surface correspondent to a minimum cut of the graph, as proposed by (Roy & Cox, 1998)

This method enables us to attribute a disparity to each pixel in the image. The disparity surface can be visualized as a surface traversing all the cut t-links in the graph. The figure 11 illustrates a minimum cut and the corresponding disparity surface. We note that this method allow us to obtain integer disparity only, and cannot deal with real disparities (sub-pixel). In fact, the construction of the complete graph requires that the zero-index of a t-link in the chain corresponds to a value in the disparity range [d_{min}, d_{max}], and in consequence, non-integer (real) values of disparity are not acceptable by this method. We will see in the next section (by the reduced graph) how to overcome this problem.

Requirements of this method:

We will calculate the total number of vertices and edges in the complete graph in order to put in evidence two constraints: the constraint of needed memory, and the constraint of execution time.

Let W and H be the width and the height of the image, [0,d_{max}] the disparity range. Let v the vertices number in the graph, and e the number of edges. For each pixel, the number of possible matches is k=d_{max}+1. For each pixel, the corresponding chain has k-1=d_{max} nodes. Thus, the total number of nodes in the graph is (the term +2 is added for the source and the sink):

$$v = W \; H \; d_{max} + 2 \tag{12}$$

In each chain (except at the boundaries), each node is linked with its 6 neighbours by edges. The number of edges in the graph is:

$$e = 6\ W\ H\ d_{max} - 2\ d_{max}\ (W + H) \quad (13)$$

The term $2\ d_{max}\ (W + H)$ is added because there are fewer edges at the boundaries of the graph. To have an idea about the total number of vertices and edges in the complete graph, see the table 1, which is calculated for an image of size 512x512 pixels and for different disparity ranges. In this table, and for a disparity range [0, 40], there are more than $10x10^6$ vertices and more than $60x10^6$ edges. This example illustrates the necessity to have huge memory (RAM). Indeed, because the complexity of maximum flow algorithm (as seen before), the execution time will be very big, especially for large disparity ranges.

	$d_{max} = 30$	$d_{max} = 40$	$d_{max} = 50$
Vertices Number v	7 864 320	10 485 760	13 107 200
Edges Number e	47 124 480	62 832 640	78 540 800

Table 1. Vertices and edges numbers in the complete graph for an image of size 512x512, for different disparity ranges.

4.4 Building a reduced graph

The major problems of the complete graph method presented in the previous section are its greediness to memory and execution time. To overcome these problems, we propose to construct a reduced graph. This reduced graph will contain a reduced number of vertices and edges. This allows us to decrease the amount of needed memory, and to lighten the execution time (Zureiki et al., 2007).

In the reduced graph: for each pixel in the image, we keep only some potential disparity values, resulting from a local method of stereo matching. However, in the method of the complete graph, we keep for each pixel all possible values of disparity (as seen in previous section).

The construction steps of the reduced graph are similar to the complete graph, with some differences:

- By mean of a local matching method (based on local similarity measurement, as SAD for example), we calculate for each left pixel p, the matching costs for all possible values in the disparity range $[d_{min}, d_{max}]$.
- Then, we choose the N best values (for illustration purposes, we choose N=4 without lack of generality). The choice may be done according to different criteria, for example, with a classic ZNCC score; we keep the disparity values around the peak, or the N best local maxima (if they exist), etc. We will note our N selected disparities for the pixel p as $\{d_{1,p}, d_{2,p}, \dots\ d_{N,p}\}$, and the corresponding costs of matching as $\{ D_{p,d_{1,p}}, D_{p,d_{2,p}}, \dots, D_{p,d_{N,p}} \}$.
- To reduce the size of the graph, for each pixel p of the image, instead of keeping a chain of k-1 nodes and k edges, we remove all the nodes and edges except N-1 nodes (and thus, N edges).
- Thus, to each pixel p we associate a novel chain of nodes $\{ p_{L_1}, p_{L_2}, \dots, p_{L_{N-1}} \}$. These nodes are connected by edges (t-links) noted $\{ t_1^p, t_2^p, \dots, t_N^p \}$ with $\{ t_1^p = (s, p_{L_1}), t_2^p = (p_{L_1}, p_{L_2}), \dots, t_N^p = (p_{L_{N-1}}, t) \}$.

- The capacity of the t-link edge i is $C + D_{p,d_{i,p}}$, where C is a constant satisfying the constraint 14. The addition of this constant assures the uniqueness of the cut of the t-links chain in exactly one edge.

$$C > N * \max_{\{p,q\} \in N} (u_{\{p,q\}}) * | d_{\max} - d_{\min} | \tag{14}$$

- For each adjacent pixels p and q, the corresponding chains are related by edges (n-links) at the N-1 levels, with a capacity as in equation 15.

$$\text{n-link capacity at level i} = u_{\{p,q\}} (| d_{i,p} - d_{i,q} | +1) \tag{15}$$

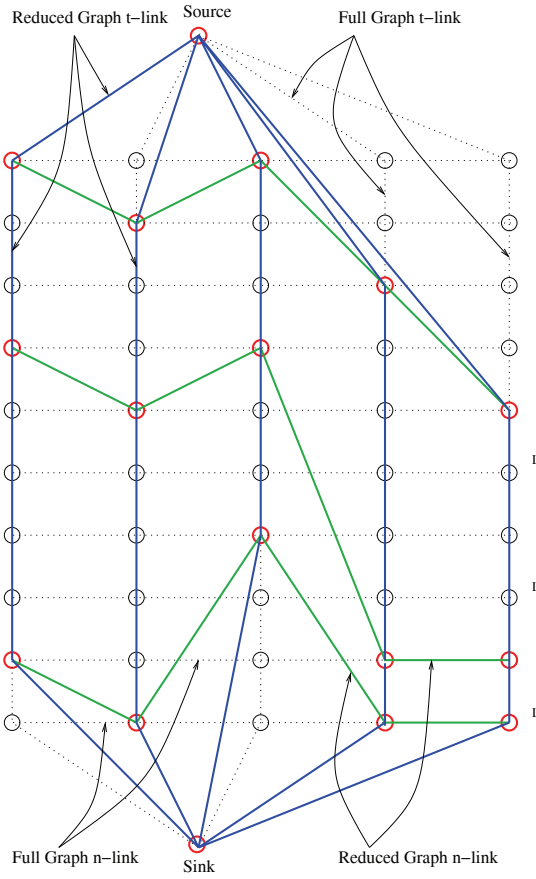

Fig. 12. Construction of the Reduced Graph.

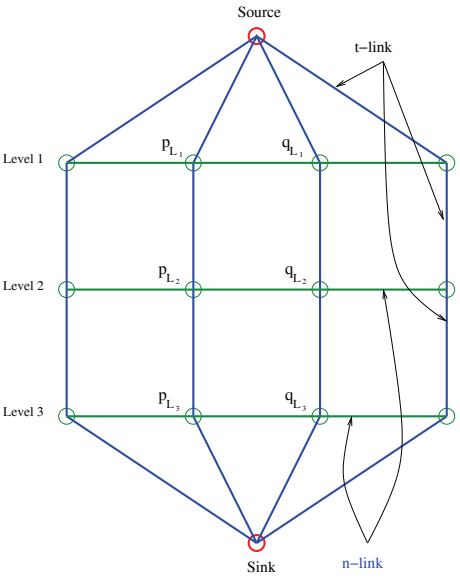

Fig. 13. The Reduced Graph.

The figure 12 illustrates the construction of the reduced graph. The dashed edges are for the complete graph (previous section). The not removed nodes are in red, the new t-links are in blue and the new n-links are in green. The figure 13 illustrates a frontal projection of the reduced graph.

Minimized Energy by the Reduced Graph:
We have seen that the minimized energy by the complete graph contains two terms. The first term represents the intrinsic data energy, which translates the costs of attributing disparities to pixels of the image. The second term aggregates the constraint of continuity (adjacent pixels have similar disparities). The constant λ can control the relative importance of the two terms. Hence, the prior energy appears in the weights associated to the n-link edges in the complete graph. In our reduced graph, we do distinguish between two types of prior information, the first translates the information acquired by the local method and acts in the choice of the selected nodes, while the second (smoothing) interferes in the penalties associated to the n-link edges. Thereby, we exploit the prior knowledge that the disparity of a pixel p has only N possible values (the most probable) in a new way. In fact, we consider that removing non-potential nodes as a novel form of representing this prior knowledge. The minimized energy can be written as:

$$E(f) = \sum_{\substack{p \in P \\ f_p \in \{d_{p,1}, \ldots, d_{p,N}\}}} D_p(f_p) + \lambda \sum_{\substack{\{p,q\} \in N \\ f_p \in \{d_{p,1}, \ldots, d_{p,N}\} \\ f_q \in \{d_{q,1}, \ldots, d_{q,N}\}}} u_{\{p,q\}} \; |f_p - f_q| \tag{16}$$

Requirements of this method:
We will calculate the total numbers of vertices and edges in the reduced graph, using the same notation as previously for W, H, d_{max}, k, etc. Let N be the number of chosen disparities among all the possible values in the disparity range. We can obtain the number of vertices and edges by replacing d_{max} by (N-1) in the equation for the complete graph:

$$v = W\,H\,(N-1) + 2 \tag{17}$$

$$e = 6\,W\,H\,(N-1) - 2\,(N-1)\,(W + H) \tag{18}$$

The table 2 gives the numbers of vertices and edges in the reduced graph for an image of size 512x512, and for any value of disparity range, while using N=4 or N=5. We clarify that the disparity range does not any more influence the size of the graph, but the number of pre-selection N. For N=4, we obtain less than 0.8×10^6 vertices and less than 0.160×10^6 edges. This example illustrates clearly the immense gain in needed memory (RAM).

	N=4	N=5
Vertices Number v	786 432	1 048 576
Edges Number e	153 600	1 564 672

Table 2. Vertices and edges numbers in the reduced graph for an image of size 512x512 with the number of pre-selected disparities N= 4 or 5.

Advantages of the Reduced Graph:
The benefits of reduced graph construction are:

- The graph becomes less voluminous. The vertices number is divided for example by 10 for a disparity range of width 40 and with the number of pre-selection N=4.
- The size of the graph is no more dependent of the disparity range. This allows us to use larger disparity ranges.
- The index of the edge in the t-link chain associated to a pixel p in the complete graph corresponds to a value in the disparity range. This prevents using non-integer

disparities. On the contrary, in the reduced graph, the value associated to the edge is added as an attribute, and there is not any constraint to be an integer or real value. Hence, we can use real value for disparity, and especially for sub-pixel disparities.

- Due to the size of the reduced graph in numbers of vertices and edges, and knowing that the maximum flow algorithms are proportional (of order 2 or 3) to these numbers, the execution time to find the minimum cut in the reduced graph is extremely smaller than the one for the complete graph. Table 3 gives some experimental results for execution time.

5. Implementations and results

In our work, images are acquired by a pre-calibrated stereo rig. In addition, our stereo matching algorithms considers that the left and right images are rectified, which means that epipolar lines are horizontal and have the same line index in the left and right images. Thus, the rectified images come from the pre-calibrated cameras are the input of the graph cuts algorithms. We suppose that the rectification is exact, which means that the disparity depends only on the column index of the pixel: the pixel (u,v) in the left image is matched to the pixel (u, v-d) in the right image. The sub-pixelic resolution is applied only to the value d.
Graph constructions and processings, are designed using the Boost Graph Library (BGL)[2]. Boost is an open source library distributed under Boost Software License[3]. The BGL is a C++ library built by using the principles of generic programming for the construction of data structures and for the implementation of the different algorithms on graphs. For more details, see (Siek et al., 2001). Among the many algorithms for manipulating graphs, BGL makes available in particular an implementation of the algorithm Push-Relabel Maximum Flow proposed by (Goldberg, 1985). This algorithm has a complexity of $O(v^3)$, with v is the vertices number. In fact, this implementation is not the best one in performance, but it allows us to evaluate the execution time for both the complete and reduced graphs. Indeed, for the reduced graph, the influence of algorithm is already reduced because the graph is less voluminous.
Algorithms of construction and of minimum cut have been implemented first on the complete graph, as described in section 4.3. Next, these algorithms of construction and of minimum cut have been adapted for the reduced graph (section 4.4). To evaluate the performance of these methods, we have used the image *sawtooth* (Scharstein & Szeliski, 2002) illustrated on figure 14. For this image, the exact disparity (round truth) is given on figure 15. For the reduced graph, the selection of a limited set of disparities on every pixel is provided by a local matching method based on SAD with window size equals to 7 pixels.
The test is done on a P4 with 3GHz and 512MB of RAM. Table 3 gives the execution time for the image sawtooth. We note that for the size 434x380 and 20 levels of disparities, we could not test the method of the complete graph (memory explosion). On the contrary, with the reduced graph, we obtain the disparity image in less than one minute. Further, on, we tested the two methods with a half-size image (217x190), we can notice a factor of 40 between the complete and reduced graphs in execution time (150 seconds against 4 seconds). In addition, we remark that for the reduced graph, the pre-selection number N influence explicitly the

[2] http://www.boost.org/libs/graph/doc/index.html
[3] http://www.boost.org/LICENSE_1_0.txt

execution time for larger sizes of images, but its influence is nearly neglected for small images. Thus, this table put in evidence the enormous gain in execution time provided by the reduced graph approach. We notice that the complete graph cannot be manipulated on normal machines (with ordinary memory), whereas the reduced graph can be built and analyzed in acceptable time. Therefore, the reduced graph algorithm is faster, in spite of a supplementary phase of calculation of local costs (which can be neglected compared with the execution time of graph cut algorithm).

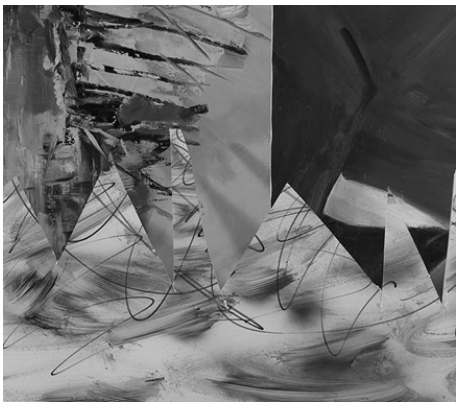

Fig. 14. The sawtooth image.

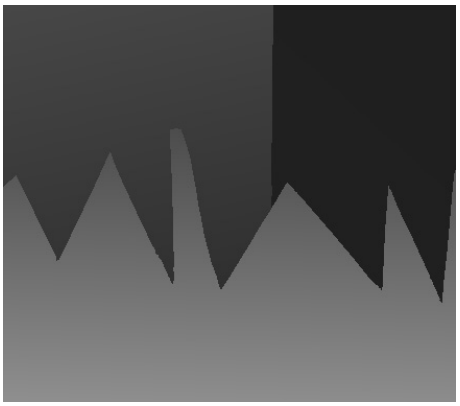

Fig. 15. The sawtooth image: true disparity.

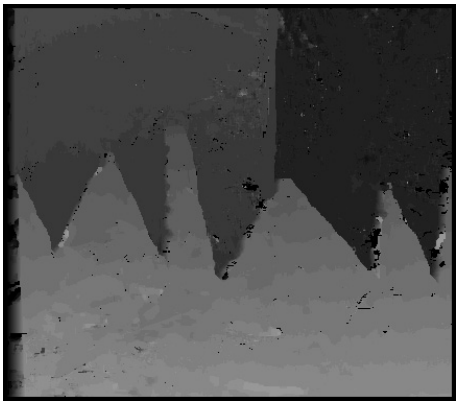

Fig. 16. Disparity image by reduced graph.

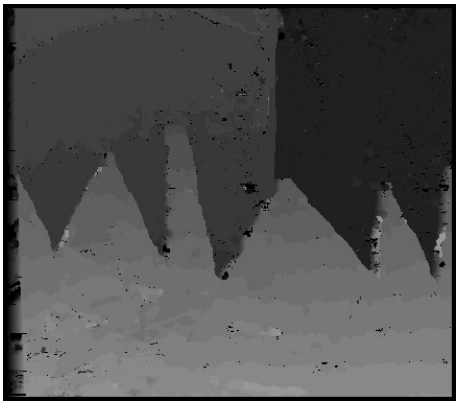

Fig. 17. Disparity image by complete graph.

Image Size	Complete Graph	Reduced Graph	
		N=4	N=5
434x380	NA	15 s	50 s
217x190	150 s	4 s	5 s

Table 3. Execution time in seconds for stereo matching using graph cuts in Complete graph and Reduced graph, for a disparity range [0, 20], and with the number of pre-selected disparities N= 4 or 5.

Figure 16 represents the disparity image obtained by the reduced graph approach, and figure 17 shows results by the complete graph. We can visually appreciate the quality of the obtained disparity images.

Fig. 18. The Flowerpots image.

Fig. 19. The Flowerpots image: true disparity.

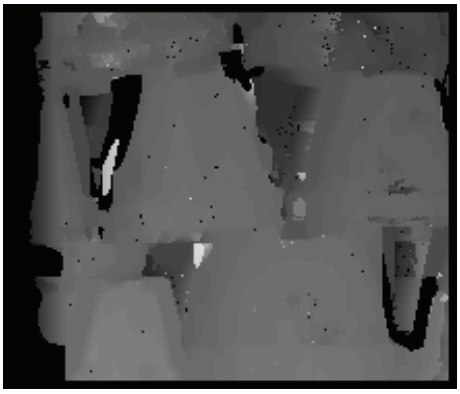

Fig. 20. Disparity image by reduced graph.

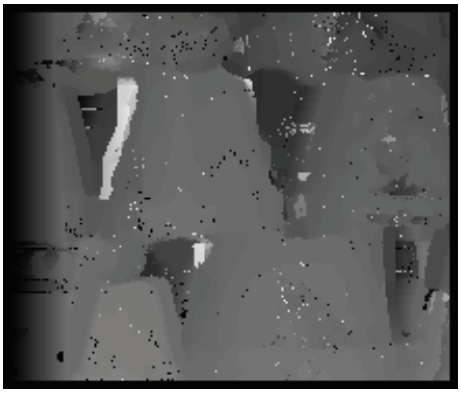

Fig. 21. Disparity image by SAD matching.

To illustrate the improvement of the reduced graph with respect to the local matching method, let us use the image *Flowerpots* [4] (Hirschmüller & Scharstein, 2007) shown in the figure 18. The true disparity of this image is given in figure 19. We used the SAD local method with a window size 7, this gives the disparity image of figure 21. When using our reduced graph method to calculate the disparity image, the result is shown in figure 20. We notice that many false matches in the local method were removed by the global method. In spite of these ameliorations, there still exist some errors in the disparity image. These errors (see figure 20), are in general in zones with very poor texture (or even mono coloured zones). In fact, these are among the most difficult problems for stereo matching.

Although the reduced graph matching method provides several ameliorations, it has some weak points, summarized in three main drawbacks:

[4] http://vision.middlebury.edu/stereo/

- The reduced graph method inherits some problems from the used local method. In fact, if the local method supplies N erroneous values of a pixel disparity, the resulting disparity for this pixel will be absolutely one of these erroneous values.By now, the global step cannot correct errors introduced by the local step.
- There are smoothing effects (like all global methods) due to the regularization. The smoothing will be more important when using bigger value for the penalty cost.
- In spite of its relative good execution time, the reduced graph method is not yet adapted to be used in real-time applications, like robotics. A parallel version of the maximum flow algorithm could be proposed to overcome this problem, taking advantage of a dedicated architecture.

6. Conclusion and perspectives

We described in this chapter, our evaluation of global stereo correspondence methods based on graph cuts. The combination of a local method, able to select a reduced set of possible matches for each pixel, and a global method, based on the graph cuts algorithm, let us to achieve two goals

- Sensibly ameliorate the quality of disparity images obtained only by a local method.
- Avoid the combinatorial explosion of the Graph Cuts method executed without preliminary reduction of the graph.

Some optimizations of our algorithm are currently studied. Note that we work always on pre-rectified images, and we produce an integer disparity image: we will study how to adapt this algorithm in order to find stereo matching on non-rectified images.
The reduced graph approach allows us to limit the execution time and to use larger disparity ranges. We write the code with the aim to obtain results and not to optimize the performance. In addition, we did not use the best maximum flow algorithm. In the next period, we could very easily improve performances, in order to satisfy as much as possible, real time constraints.
Section 4.4 has shown that the reduced graph is able to deal with sub-pixel disparities (real values). Testing this property and obtaining sub-pixel disparity images will be our main objective in our near future works.

7. References

Belhumeur, P. N. (1996). A bayesian approach to binocular stereopsis. *International Journal of Computer Vision*, 19(3):237-260.

Bellman, R. (1957). *Dynamic programming*, Princeton University Press.

Bobick, A. F. & Intille, S. S. (1999). Large occlusion stereo. *International Journal of Computer Vision (IJCV)*, 33(3):181-200.

Boykov, Y. & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1124-1137.

Boykov, Y., Veksler, O. & Zabih, R. (1998). Markov random fields with efficient approximations. *In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 648-655.

Boykov, Y., Veksler, O. & Zabih, R. (1999). Fast approximate energy minimization via graph cuts. *In Proceedings of International Conference on Computer Vision (ICCV)*, volume 1, pp. 377-384.

Chen, Q. & Medioni, G. (1999). A volumetric stereo matching method : Application to image based modeling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. & Stein, C. (2001). *Introduction to Algorithms, Second Edition*, The MIT Press.

Cox, I. J. (1992). Stereo without disparity gradient smoothing: a bayesian sensor fusion solution. *In proceedings of British Machine Vision Conference (BMVC)*, pp. 337-346.

Cox, I. J., Hingorani, S. L., Rao, S. B. & Maggs, B. M. (1996). A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542-567.

Dahlhaus, E., Johnson, D. S., Papadimitriou, C. H., Seymour, P. D. & Yannakakis, M. (1994). The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864-894.

Faugeras, O. (1993). *Three-Dimensional Computer Vision : a Geometric Viewpoint*, MIT press.

Faugeras, O., Vieville, T. et al. (1993). *Real-time correlation-based stereo: algorithm, implementations and applications*. Rapport technique RR-2013, INRIA-Sophia Antipolis.

Ford, L. & Fulkerson, D. (1962). *Flows in Networks*, Princeton University Press.

Goldberg, A. V. (1985). *A new max-flow algorithm*. Rapport technique MIT/LCS/TM-291, Massachusetts Institute of Technology, Cambridge, MA.

Goldberg, A. V. & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the Association for Computing Machinery (JACM)*, 35(4):921-940.

Grandjean, P. & Lasserre, P. (1995). Stereo Vision Improvments. *In IEEE International Conference on Advanced Robotics*, Barcelona (Spain).

Hirschmüller, H. (2001). Improvements in real-time correlation-based stereo vision. *In Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*.

Hirschmüller, H. and Scharstein, D. (2007). Evaluation of cost functions for stereo matching. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Minneapolis, MN, pp. 1-8.

Horn, B. (1986). *Robot Vision*, MIT Press.

Ishikawa, H. (2000). *Global optimization using embedded graphs*. PhD Thesis, New York University. Adviser : Davi Geiger.

Ishikawa, H. & Geiger, D. (1998). Occlusions, discontinuities, and epipolar lines in stereo. *In Proceedings of the 5th European Conference on Computer Vision (ECCV)*, pp. 232-248. Springer-Verlag.

Julesz, B. (1962). Towards the automation of binocular depth perception. *In IFIP Congress*, pp. 439-444.

Karzanov, A. V. (1974). Determining the maximal flow in a network by the method of preflows. *Soviet Mathematics Doklady*, 15:434-437.

Kolmogorov, V. & Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. *In Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 508-515.

Kolmogorov, V. & Zabih, R. (2002a). Multi-camera scene reconstruction via graph cuts. *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pp. 82-96. Springer-Verlag.

Kolmogorov, V. & Zabih, R. (2002b). What energy functions can be minimized via graph cuts? *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pp. 65-81. Springer-Verlag.

Ohta, Y. & Kanade, T. (1985). Stereo by intra- and inter-scanline search using dynamic programming. *In IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 139-154.

Roy, S. (1999). Stereo without epipolar lines: A maximum-flow formulation. *International Journal of Computer Vision*, 34(2-3):147-161.

Roy, S. & Cox, I. (1998). A maximum-flow formulation of the n-camera stereo correspondence problem. *In Proceedings of the International Conference on Computer Vision (ICCV)*, pp. 492-499.

Sára, R. (2002). Finding the largest unambiguous component of stereo matching. *In Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pp. 900-914. Springer-Verlag.

Scharstein, D. & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47.

Siek, J. G., Lee, L.-Q. & Lumsdaine, A. (2001). *Boost Graph Library, The : User Guide and Reference Manual*. Addison-Wesley Professional.

Veksler, O. (1999). *Efficient graph-based energy minimization methods in computer vision*. PhD Thesis, Cornell University. Adviser : Ramin Zabih.

Zureiki, A., Devy, M. & Chatila, R. (2007b). Stereo matching using reduced-graph cuts. *In IEEE International Conference on Image Processing (ICIP)*, San Antonio, Texas (USA).