P Systems Implementation of Dynamic Programming Stereo

Georgy Gimel'farb
Radu Nicolescu
Sharvin Ragavan
Department of Computer Science,
University of Auckland,
Auckland, New Zealand

CDMTCS-401
April 2011 / Revised: December 2011

P Systems Implementation of Dynamic Programming Stereo

Georgy Gimel'farb, Radu Nicolescu and Sharvin Ragavan
Department of Computer Science
The University of Auckland, Private Bag 92019
Auckland, New Zealand
{g.gimelfarb, r.nicolescu}@auckland.ac.nz,
srag010@aucklanduni.ac.nz

December 31, 2011

Abstract

Designing parallel versions of sequential algorithms has attracted renewed attention, due to recent hardware advances, including various general-purpose multi-core and many-core processors, as well as special-purpose FPGA implementations. P systems consist of networks of autonomous cells, such that each cell transforms its input signals in accord with its symbol-rewriting rules and feeds the output results into its immediate neighbours. Inherent massive intra- and inter-cell parallelisms make P systems a prospective theoretical testbed for designing efficient parallel and parallel-sequential algorithms. This paper discusses the ability of P systems to implement the symmetric dynamic programming stereo (SDPS) matching algorithm, which explicitly accounts for binocular or monocular visibility of 3D surface points. Given enough cells, the P system implementation speeds up the inner algorithm loop from $O(nd)$ to $O(n + d)$, where n is the width of a stereo image and d is the disparity range. The implementation also gives an insight into to a more general SDPS algorithm that allows a possible multiplicity of solutions to the ill-posed optimal stereo matching problem.

Keywords: Membrane computing, Parallel systems, Stereo matching, Symmetric dynamic programming stereo (SDPS).

1 Introduction

Essentially, a P system is a network of data processing cells, inspired by the structure and interaction of living cells [1, 6, 10, 12, 13, 14, 15, 16]. Each cell transforms input and local symbols in accord with rewriting rules and sends some of the resulting symbols out to its immediate neighbours. Rules of the same cell can be applied in parallel (if possible) and all the cells evolve in parallel, in a distributed *synchronous* mode. The underlying network is a *digraph* or a more specialized version, such as a directed acyclic graph (*dag*) or a *tree* (which is the most studied case). Advanced scenarios consider cases when cells or arcs can dynamically appear, disappear or move.

Several previous papers suggested that P systems offer a theoretically efficient testbed for the design of parallel versions of various sequential image analysis tasks, e.g. segmentation [4, 5, 17]. This paper discusses a P system based implementation of the *Symmetric Dynamic Programming Stereo* (SDPS) [8]. A P system model for stereo matching, which reveals and separates the inherently parallel and sequential stages of SDPS, was introduced first in our paper [9]. Below, the latter is extended and elaborated further. Section 2 defines and details a P model. The P implementation of SDPS is discussed in Section 3 and its impact on this algorithm (as well as on other dynamic programming solutions of ill-posed problems with multiple equivalent solutions) is outlined in Section 4. Conclusions are given in Section 5.

2 General P Model

Dinneen et al.'s basic definition of *simple P modules* [6] covers many common P systems, such as *cell-like* (based on trees), *hyperdag* (based on dags), *tissue* and *neural* P systems (based on directed graphs). This definition is further generalised below, by introducing new features, which appear useful for modelling SDPS.

Definition 1. A simple P module with duplex channels is a system $\Pi = (O, K, \Delta)$, where O is a finite non-empty alphabet of symbols; K is a finite set of cells and Δ is an irreflexive binary relation on K, representing a set of structural *parent-child* arcs between cells (essentially a digraph), with duplex communication capabilities.

Each cell, $\sigma_i \in K$, has the initial configuration $\sigma_{i0} = (Q_i, s_{i0}, w_{i0}, R_i)$, and the current configuration $\sigma_i = (Q_i, s_i, w_i, R_i)$, where:

- Q_i is a finite set of states;
- $s_{i0} \in Q_i$ is the initial state;
- $s_i \in Q_i$ is the current state;
- $w_{i0} \in O^*$ is the initial multiset of symbols;
- $w_i \in O^*$ is the current multiset of symbols;
- R_i is a finite ordered set of multiset rewriting rules of the form:

$$s\ x \rightarrow_\alpha s'\ x'\ (u)_{\beta_\gamma} \ldots |z,$$

  where $s, s' \in Q$, $x, x', u, z \in O^*$, $\alpha \in \{\texttt{min}, \texttt{max}\}$, $\beta \in \{\uparrow, \downarrow, \updownarrow\}$ and $\gamma \in K \cup \{\forall\}$. Except the two states, s and s', all other ingredients are optional and can be omitted. For example, if $u = z = \lambda$ (the empty multiset of symbols), then this rule can be abbreviated as $s\ x \rightarrow_\alpha s'\ x'$.

All cells evolve *synchronously.* A cell evolves by applying one or more rules that may change its content and state and may send symbols to its neighbours. For cell $\sigma_i = (Q_i, s_i, w_i, R_i)$, a rule $s\ x \rightarrow_\alpha s'\ x'\ (u)_{\beta_\gamma} | z \in R_i$ is applicable if $s = s_i$, $xz \subseteq w_i$. The application of a rule has two sub-steps: (i) its left-hand side is processed, x is removed from the current content, the next state s' is decided; (ii) its right-hand side is processed, the cell transits to the decided target state s', x' is added to the current content and u is sent as specified by the transfer operator β_γ (as further described below). Multiset z is a *promoter*: it enables the rule, but does not otherwise participate in its application and remains unchanged. The rules are applied in weak priority order [14]: (a) higher priority applicable rules are applied before lower priority rules, and (b) a lower priority rule can be applied after other rules only if it reaches the same target state.

In this paper, we use the rewriting operators $\alpha \in \{\texttt{min}, \texttt{max}\}$, and the transfer operators $\beta_\gamma \in \{\downarrow_\forall, \downarrow_j, \uparrow_j\}$, where $\sigma_j \in K$. Other (not used in this paper) operators are described [7]. The rewriting operator, $\alpha = \texttt{min}$ indicates that the rule is applied once; while $\alpha = \texttt{max}$ indicates that the rule is applied as many times as possible. Multiset u represents a message, sent to digraph neighbours, up, down or up and down *structural arcs*, as specified by the β_γ operators.

If the right-hand side of the rule contains $(u)_{\downarrow_\forall}$, then, for each application of this rule, a copy of multiset u is sent to each cell in $\Delta(i)$, i.e. to each structural *child* of σ_i. If the right-hand side of the rule contains $(u)_{\downarrow_j}$, then, for each application of this rule, a copy of multiset u is sent to cell $\sigma_j \in K$, if $j \in \Delta(i)$, i.e. if σ_j is a structural *child* of σ_i (the mode $\downarrow_\forall$ is also known as $\downarrow_{\texttt{repl}}$). However, if $j \notin \Delta(i)$, then message u is silently lost. Similarly, if the right-hand side of the rule contains $(u)_{\uparrow_j}$, then, for each application of this rule, a copy of multiset u is sent to cell $\sigma_j \in K$, if $j \in \Delta^{-1}(i)$, i.e. if σ_j is a structural *parent* of σ_i; otherwise, if $j \notin \Delta^{-1}(i)$, then message u is silently lost.

A state is called *quiescent* if no rules can be applied for empty cells in this state (i.e. an empty quiescent cell does not evolve until some symbol appears in this cell, e.g., is received from this cell's neighbours).

Although the definition does *not* enforce this, we typically require a *fixed number* of *fixed size* rule sets. This is a strong requirement, which enables the design of scalable algorithms, which can solve problem instances of any size, without increasing the alphabet size or the number of rules. One could imagine a virtual cell factory, which, using a fixed number of cell blueprints, creates any number of required cells, which are then allocated different positions in a structural digraph.

Extensions. We extend the basic simple P module concept with the following three features, which are useful in complex scenarios, such as SDPS.

1. Arcs can have *labels* to be used in transfer operators, instead of cell labels. For example, the occurrence of $(u)_{\downarrow_k}$ in the right-hand side of a σ_i rule indicates that message u is to be sent from cell σ_i to cell σ_j, where k is the label of arc $(\sigma_i, \sigma_j) \in \Delta$, also denoted by $\sigma_i \xrightarrow{k} \sigma_j$. Otherwise, if cell σ_i does not have an outgoing arc labelled k, message u is silently lost. Although not globally unique, arc labels are locally unique (unique in each local neighbourhood).

2. A rule can send several messages (not just one), each one to a different neighbour, e.g., the occurrence of $(u)_{\downarrow_k}(u')_{\downarrow_{k'}}$ in the right-hand side of a rule indicates that messages u, u' are to be sent via arcs labelled k, k', respectively.

3. A pair of neighbouring nodes can be connected by several labelled arcs (not just one), i.e. the supporting structural digraph becomes a *directed multigraph* with labelled arcs.

These extensions offer additional support for the design of scalable algorithms (without increasing the alphabet size or the number of rules), as many cells can reuse the same labels for their outgoing arcs.

Example 2. Consider a simple P module Π, with four cells, σ_1, σ_2, σ_3, σ_4, sharing the same three states, s_0 (initial state), s_1, s_2, and the same ordered set of four rules:

1. $s_0\ a \rightarrow_{\texttt{min}} s_1\ a'\ (e)_{\downarrow_\forall}\ (f)_{\downarrow_u}\ (g)_{\downarrow_v}\ (h)_{\downarrow_w}$
2. $s_0\ b \rightarrow_{\texttt{max}} s_1\ b\ b'$
3. $s_0\ c \rightarrow_{\texttt{min}} s_2\ c'$
4. $s_0\ d \rightarrow_{\texttt{max}} s_1\ d'$

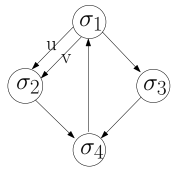


Figure 1: P module based on a multigraph with labelled arcs.

Assume that these four cells are arranged in the multigraph structure shown in Figure 1, where cells σ_1 and σ_2 are connected by two arcs, labelled u and v, and all cells are initially empty, except cell σ_1, which contains the symbol multiset ab^2cd^2. In this scenario, all rules are applicable for cell σ_1. First, rule 1 is applied once and sets the target state to s_1. Next, rule 2 is applied exactly twice, because the right-hand side symbols b are produced after all rules are applied. However, rule 3 is not applied, because its right-hand side indicates a different target state, s_2. Finally, rule 4 is applied twice. In the final configuration of the system, after one step: σ_1 is in state s_1 and contains multiset $a'b^2b'^2cd'^2$; σ_2 is in state s_0 and contains multiset efg; σ_3 is in state s_0 and contains multiset e; σ_4 is in state s_0 and is empty.

3 SDPS: P system design

Sample scenario. The stereo matching scenario in Example 3 will be used throughout the rest of this paper.

Example 3. Two cameras, L and R, in a (simplistic) planar configuration in Figure 2 view a line object, defined by points $\{p_i \mid i \in [0,6]\}$. Each camera defines a (non-rectangular) coordinate system, $\mathbb{X} = [0,5]$. Assuming that all the points in this plane share the same y-coordinate $y = 0$, our points have the following coordinates, in a $(x_{\texttt{left}}, y, x_{\texttt{right}})$ system: $p_0 = (0,0,0)$, $p_1 = (1,0,0)$, $p_2 = (2,0,1)$, $p_3 = (3,0,2)$, $p_4 = (3,0,3)$, $p_5 = (4,0,4)$ and $p_6 = (5,0,5)$. Points may be classified by their *visibility state*: binocular (B)—seen by both cameras, monocular-left (M_L)—seen by the left camera only and monocular-right (M_R)—seen by the right camera only; e.g., p_0 is M_L and p_4 is M_R. Note that, under an assumed single continuous surface, the monocular visibility is caused by self-occlusions. The point p_0 can be seen from the left camera only (so can be called monocular left by visibility), and the point p_4 is monocular right (can be seen from the right camera only); all other points are binocular (can be seen by both camera). Such sequences are known as profiles.

Profiles can be represented in an alternate notation, which will be used throughout the rest of paper, where the $x_{\texttt{right}}$ coordinate is replaced by the difference to be subtracted from $x_{\texttt{left}}$ to obtain $x_{\texttt{right}}$. This difference, $d = x_{\texttt{left}} - x_{\texttt{right}}$, is called parallax, or disparity, and can be used to estimate depth distances. By this transformation, in the $(x_{\texttt{left}}, y, d)$ system, our sequence becomes: $p_0 = (0,0,0)$, $p_1 = (1,0,1)$, $p_2 = (2,0,1)$, $p_3 = (3,0,1)$, $p_4 = (3,0,0)$, $p_5 = (4,0,0)$, $p_6 = (5,0,0)$. Clearly, points with larger disparity are closer to the cameras than points with smaller disparity; in this scenario, points p_1, p_2, p_3, with disparity 1, are closer to the cameras than the other points, with disparity 0.

Note that the projection of the profile on the left coordinate system gives an unambiguous representation of the full profile. In our case, this projection yields a reduced disparity sequence: 0, 1, 1, 1, 0, 0; between the two points sharing the same x-coordinate, p_3 and p_4, the projection selects the one closer to the cameras (which is also the one with larger disparity). The projection ignores the monocular right points, but these can be unambiguously reconstructed from the reduced disparity sequence.

Binocular stereo matching determines the disparities associated with the points that are in visual field of two cameras, solely based on the pixel values recorded. Our scenario assumes that the pixel intensity values are 15, 10, 30, 50, 15, 10 for the left camera and 10, 30, 50, 50, 15, 10 for the right camera. In general, this is a complex problem, as different points may appear with the same grey scale intensity, or the same point may appear with different grey intensities in the two cameras. In fact, mathematically, the problem is ill-posed, as there always exist ambiguities, i.e. scenarios which, for the same pixel intensities, offer more than one valid interpretation. We want first to regularise this problem and then solve it as fast as possible, using all parallel computing facilities available.

Concepts. Given a pair of rectified images, let $\mathbb{C} = \mathbb{XYD}$ be a discrete space of 3D points, $\mathbf{p} = (x, y, d)$, of a single optical (visible) continuous surface, reduced to the left image plane $\mathbb{XY}$, with integer planar (x, y)-coordinates, $x \in \mathbb{X} = [0, n-1]$, $y \in \mathbb{Y} = [0, m-1]$ and specified by integer disparities, $d \in \mathbb{D} = [d_{\min}, d_{\max}]$, of corresponding points, (x, y) and $(x - d, y)$, depicting $\mathbf{p}$ in the left and right image, respectively. We

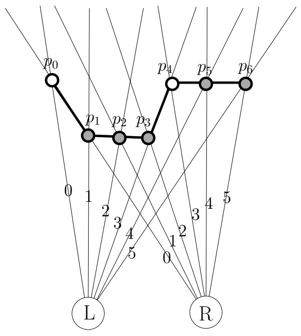


Figure 2: Sample configuration for our algorithm.

further assume that this correspondence is left-total, right-total and order preserving.

Each 2D profile relating to a conjugate pair of scanlines with the same y-coordinate, $y \in \mathbb{Y}$, is given by a sequence of points $\mathbf{d}_y = \{(x, y, d) \mid x \in \mathbb{X}, d \in \mathbb{D}\}$, such that, for each two successive points, $\mathbf{p}' = (x', y', d')$ and $\mathbf{p} = (x, y, d)$, either: (i) $x = x' + 1$ and $d \in \{d', d' + 1\}$ or (ii) $x = x'$ and $d = d' - 1$.

A profile, $\mathbf{d}_y$, can be unambiguously (reversibly) represented by a *reduced* profile, $\hat{\mathbf{d}}_y$: its projection on the left image space. Specifically, for each $x \in \mathbb{X}$, we only keep its highest associated disparity value. It is clear that $\hat{\mathbf{d}}_y$ can be simply described by a sequence of disparity values, $\hat{\mathbf{d}}_y = \{d_i \mid i \in \mathbb{X}\}$, and:

$$|\mathbb{X}| = |\hat{\mathbf{d}}_y| \leq |\mathbf{d}_y| \leq 2 \cdot |\mathbb{X}|.$$

Let $s \in \{B, M_L, M_R\}$ indicate visibility of a 3D point. Consider two successive profile points, extended with explicit visibility states: $\mathbf{v'}_{s'} = (x', y, d', s')$ and $\mathbf{v}_s = (x, y, d, s)$. Column "Constraints" of Table 1 summarizes the visibility constraints assumed for each profile. In fact, the visibility constraints define a relation Ω, describing the *forward transition* between two successive points of a profile. Tables 2 and 3 show the *forward transition* relation, Ω, and its inverse, the *backward transition* relation, Ω^{-1}; these two relations are further used in our construction.

Figure 3 shows an intuitive representations of the above constraints, as transitions between visibility states.

$$\begin{tikzcd}
M_L \arrow[loop above, "6"] \arrow[r, bend left, "3"] & B \arrow[loop above, "1"] \arrow[l, bend left, "4"] \arrow[r, bend right, "7"'] & M_R \arrow[loop above, "8"] \arrow[l, bend right, "2"'] \arrow[ll, bend left=50, "5"]
\end{tikzcd}$$

Figure 3: Allowed transitions between visibility states.

Given $y \in \mathbb{Y}$, let $g_L(x, y)$ and $g_R(x, y)$ denote grey intensities along two corresponding scanlines, for $x \in \mathbb{X}$. Without loss of generality, we assume that these two lines have the same length, $|\mathbb{X}|$. A simplified version of SDPS relates the point-wise signal dissimilarity:

- to a thresholded absolute difference $\delta(x, y, d) = \max\{0, |g_L(x, y) - g_L(x - d, y)| - \theta\}$, $\theta \geq 0$, between the corresponding signals for a binocularly visible point, $\mathbf{v}_B$, or

Table 1: Visibility constraints and dissimilarity scores between two successive points: $\mathbf{v}'_{s'} = (x', y, d', s')$ and $\mathbf{v}_s = (x, y, d, s)$.

#	s'	s	Constraints	$\varphi_y(\mathbf{v}_s \vert \mathbf{v}'_{s'})$	Arc
1	B	B	$x = x' + 1, d = d'$	$\delta(x, y, d)$	b
2	M_R	B	$x = x' + 1, d = d'$	$\delta(x, y, d)$	c
3	M_L	B	$x = x' + 1, d = d' + 1$	$\delta(x, y, d) \times 0.5$	a
4	B	M_L	$x = x' + 1, d = d'$	$\delta_{\texttt{occl}}$	b
5	M_R	M_L	$x = x' + 1, d = d'$	$\delta_{\texttt{occl}}$	c
6	M_L	M_L	$x = x' + 1, d = d' + 1$	$\delta_{\texttt{occl}} \times 0.5$	a
7	B	M_R	$x = x', d = d' - 1$	$\delta_{\texttt{occl}} \times 0.5$	d
8	M_R	M_R	$x = x', d = d' - 1$	$\delta_{\texttt{occl}} \times 0.5$	e

Table 2: The forward transition relation, Ω, determined by the visibility constraints.

$\mathbf{v}'_{s'}$	$\Omega(\mathbf{v}'_{s'})$	Arc
(x', y, d', M_L)	$(x' + 1, y, d' + 1, M_L)$	a
(x', y, d', M_L)	$(x' + 1, y, d' + 1, B)$	a
(x', y, d', B)	$(x' + 1, y, d', M_L)$	b
(x', y, d', B)	$(x' + 1, y, d', B)$	b
(x', y, d', M_R)	$(x' + 1, y, d', M_L)$	c
(x', y, d', M_R)	$(x' + 1, y, d', B)$	c
(x', y, d', B)	$(x', y, d' - 1, M_R)$	d
(x', y, d', M_R)	$(x', y, d' - 1, M_R)$	e

Table 3: The backward transition relation, Ω^{-1}, determined by the visibility constraints.

$\Omega^{-1}(\mathbf{v}_s)$	$\mathbf{v}_s$	Arc
$(x - 1, y, d - 1, M_L)$	(x, y, d, M_L)	a
$(x - 1, y, d - 1, M_L)$	(x, y, d, B)	a
$(x - 1, y, d, B)$	(x, y, d, M_L)	b
$(x - 1, y, d, B)$	(x, y, d, B)	b
$(x - 1, y, d, M_R)$	(x, y, d, M_L)	c
$(x - 1, y, d, M_R)$	(x, y, d, B)	c
$(x, y, d + 1, B)$	(x, y, d, M_R)	d
$(x, y, d + 1, M_R)$	(x, y, d, M_R)	e

- to a heuristic regularising score, $\delta_{\texttt{occl}} \geq 0$, for a monocularly visible point, $\mathbf{v}_{M_L}$ or $\mathbf{v}_{M_R}$.

Column "$\varphi_y(\mathbf{v}_s|\mathbf{v'}_{s'})$" of Table 1 shows, in the first line of each table cell, the rules for integrating point-wise dissimilarities between successive profile points. For example, transition #1, $B \Rightarrow B$, adds a cost of $\delta(x, y, d)$. Due to unequal numbers of points in profile variants to be compared by their total signal similarity, these point-wise dissimilarities are corrected by taking into account their actual unit or half-unit shifts in the Cyclopean space. The second line of each table cell in same column indicates when this correction is applied and the correction factor. For example, transition #7, $M_R \Rightarrow M_R$, indicates the correction factor of 0.5, thus its transition cost is $0.5 \cdot \delta_{\texttt{occl}}$.

For the first points in a profile, we take:

1. $\varphi_y(\mathbf{v}_B) = \delta(x, y, d)$;
2. $\varphi_y(\mathbf{v}_{M_L}) = \delta_{\texttt{occl}}$;
3. $\varphi_y(\mathbf{v}_{M_R}) = \delta_{\texttt{occl}}$.

See [8] for the full version of SDPS, which adapts the matching to possible local contrast and offset deviations of the corresponding signals.

SDPS computations. For a given $y \in \mathbb{Y}$, SDPS *minimizes* the dissimilarity score between the two y-conjugate scanlines:

$$\mathbf{d}^*_y = \arg\min_{\mathbf{d}_y} \Phi_y(\mathbf{d}_y), \tag{1}$$

$$\Phi_y(\mathbf{d}_y) = \varphi_y(\mathbf{v}_{0,s_0}) + \sum_{i=1}^{|\mathbf{d}_y|-1} \varphi_y(\mathbf{v}_{i,s_i}|\mathbf{v}_{i-1,s_{i-1}}), \tag{2}$$

where the current disparity profile, $\mathbf{d}_y$, is indexed over $[0, |\mathbf{d}_y| - 1]$.

The *forward pass* computes potentially minimal dissimilarity scores $F_y(\mathbf{v}_s)$ and backward transitions $T_y(\mathbf{v}_s)$, by sequential pass along $x \in \mathbb{X}$, over all $d \in \mathbb{D}$. If $\Omega^{-1}(\mathbf{v}_s) \neq \emptyset$, then:

$$F_y(\mathbf{v}_s) = \min_{\mathbf{v'}_{s'} \in \Omega(\mathbf{v}_s)} \{F_y(\mathbf{v'}_{s'}) + \varphi_y(\mathbf{v}_s|\mathbf{v'}_{s'})\}, \tag{3}$$

$$T_y(\mathbf{v}_s) = \arg\min_{\mathbf{v'}_{s'} \in \Omega^{-1}(\mathbf{v}_s)} \{F_y(\mathbf{v'}_{s'}) + \varphi_y(\mathbf{v}_s|\mathbf{v'}_{s'})\}. \tag{4}$$

Otherwise, if $\Omega^{-1}(\mathbf{v}_s) = \emptyset$, then:

$$F_y(\mathbf{v}_s) = \varphi_y(\mathbf{v}_s), \tag{5}$$

$$T_y(\mathbf{v}_s) = \Box \ (\textit{undefined}). \tag{6}$$

For each $y \in \mathbb{Y}$, the *backward pass* computes minimal profiles $\mathbf{d}_y$, in reverse order:

$$\text{initially}: \quad \widehat{\mathbf{v}_s} = \arg\min_{\mathbf{v}_s}\{F_y(\mathbf{v}_s)\} \tag{7}$$

$$\text{while } T_y(\widehat{\mathbf{v}_s}) \neq \Box: \quad \widehat{\mathbf{v'}_{s'}} = T_y(\widehat{\mathbf{v}_s}) \tag{8}$$

If we do not apply the corrections mentioned in Table 1, the above equations take simpler forms. For example, Equation 3 takes the general forms:

$$\begin{aligned} F_y(\mathbf{v}_B) &= \min\{F_y(\mathbf{v}'_B), F_y(\mathbf{v}'_{M_R}), F_y(\mathbf{v}'_{M_L})\} \\ &\quad + \delta(x,y,d), \\ &\quad \text{where } \{\mathbf{v}'_B, \mathbf{v}'_{M_R}, \mathbf{v}'_{M_L}\} = \Omega^{-1}(\mathbf{v}_B); \end{aligned} \tag{9}$$

$$\begin{aligned} F_y(\mathbf{v}_{M_L}) &= \min\{F_y(\mathbf{v}'_B), F_y(\mathbf{v}'_{M_R}), F_y(\mathbf{v}'_{M_L})\} \\ &\quad + \delta_{\mathtt{occl}}, \\ &\quad \text{where } \{\mathbf{v}'_B, \mathbf{v}'_{M_R}, \mathbf{v}'_{M_L}\} = \Omega^{-1}(\mathbf{v}_{M_L}); \end{aligned} \tag{10}$$

$$\begin{aligned} F_y(\mathbf{v}_{M_R}) &= \min\{F_y(\mathbf{v}'_B), F_y(\mathbf{v}'_{M_R})\} + \delta_{\mathtt{occl}}, \\ &\quad \text{where } \{\mathbf{v}'_B, \mathbf{v}'_{M_R}\} = \Omega^{-1}(\mathbf{v}_{M_R}). \end{aligned} \tag{11}$$

Design issues. Without loss of generality, our P system was designed for one pair of conjugate scanlines ($|\mathbb{Y}| = 1$), which arguably is the most challenging parallelisation task. This core solution can be further extended to a stereo pair of images ($|\mathbb{Y}| > 1$) in a straightforward manner, using "trivial" parallel processing of each pair of conjugate scanlines. We also take $d_{\text{min}} = 0$, so the disparities range over the integer interval $\mathbb{D} = [0, d_{\text{max}}]$, and we encode integer numbers by repeating symbols, starting with one occurrence for zero (as in traditional λ-calculus), e.g., the base symbol a gives the following encodings: $0 \to a$, $1 \to a^2$, $2 \to a^3$, etc.

A bird's eye view of our P system shows several types of cells:

- Input cells, codenamed L, that hold pixel values of the left scanline;
- Input cells, codenamed R, that hold pixel values of the right scanline;
- Output cells, codenamed D, that will hold, without loss of generality, disparities for optimal profiles, in the reduced format (projected on the left image plane);
- Work cells, codenamed W, that compute the forward and backward phase of our dynamic programming algorithm, keeping track of the visibility states (M_L, B, M_R);
- Work cells, codenamed M, that help the selection of global minimum.

The critical visibility constraints and dissimilarity integration rules, indicated in Table 1, are implemented by way of arcs linking the W work cells, as indicated by column "Arc". For example, transition #.1, $B \Rightarrow B$, will be supported via arc labelled a. Further details are given in Section 3.1.

For simplicity, in this paper we do not apply the optional corrections indicated in Table 1, i.e. the four cases involving multiplications by 0.5. We can thus use the simpler formulas given by Equations 9, 10 and 12, where the indicated $F_y(\mathbf{v}'_{s'})$ costs come as messages, via arcs a, b, c, d and e. However, if required, the corrections can be supported, by rewriting rules which divide given counters by two. For example, the rewriting formula $a^2 \to_{\mathtt{max}} a$ halves a counter represented by a symbols.

Example 4. Consider again our basic scenario, from Example 3, where: $n = 6$, $\mathbb{X} = [0,5]$, $\mathbb{Y} = \{0\}$. Assuming $d_{max} = 3$, $\mathbb{D} = [0,3]$, $\theta = 0$ and $\delta_{\texttt{occl}} = 18$, the above SDPS algorithm determines that the global minimum dissimilarity score is 36, which is achieved by two optimal profiles:

$$\begin{aligned} \mathbf{d} &= \{(0,0,0),(1,0,1),(2,0,1),(3,0,1),(3,0,0),\\ &\quad (4,0,0),(5,0,0)\},\\ \mathbf{d}' &= \{(0,0,0),(1,0,0),(2,0,0),(3,0,0),\\ &\quad (4,0,0),(5,0,0)\},\\ \hat{\mathbf{d}} &= \{0,1,1,1,0,0\} \ (\textit{reduced form}),\\ \hat{\mathbf{d}}' &= \{0,0,0,0,0,0\} \ (\textit{reduced form}). \end{aligned} \tag{12}$$

The actual computations are not shown here, but appear in the sequel, as executions steps of our P system version.

3.1 Cells, initialisation and arcs

Cells. We use a P system, Π, consisting of five major components (groups of cells sharing the same rule sets):

- L: a list of $|\mathbb{X}|$ cells, $\sigma^L_i, i \in \mathbb{X}$, which represent the *left* scan line, initialized with the left pixel values, encoded in base a. In the scenario of Example 4, these cells are initialized, in order, with the following multisets: a^{16}, a^{11}, a^{31}, a^{51}, a^{16}, a^{11}.

- R: a list of $|\mathbb{X}|$ cells, $\sigma^R_i, i \in \mathbb{X}$, which represent the *right* scan line, initialized with the right pixel values, encoded in base c. In the scenario of Example 4, these cells are initialized, in order, with the following multisets: c^{11}, c^{31}, c^{51}, c^{51}, c^{16}, c^{11}.

- D: a list of $|\mathbb{D}|$ cells, $\sigma^D_i, i \in \mathbb{X}$, which represent a *disparity* profile, initially empty. During the last phase of our P program, these cells will contain optimal disparity values, in base d. In the scenario of Example 4, for profile $\hat{\mathbf{d}}$, these cells will contain, in order, the following multisets: d^1, d^2, d^2, d^2, d^1, d^1.

- W: an array of $|\mathbb{X}|\times|\mathbb{D}|$ cells, $\sigma^W_{ij}, i \in \mathbb{X}, j \in \mathbb{D}$, which represent the *main workspace*. Each cell σ_{ij} encloses three logical (virtual) *subcells*, respectively holding *monocular-left* values (M_L), *binocular* values (B), and *monocular-right* values (M_R). The initialisation of these cells is described later on, in this section.

- M: a list of $|\mathbb{X}|$ cells, $\sigma^M_j, j \in \mathbb{D}$, which represent a secondary workspace, which identifies the global *minimum* score, among $d_{\max}+1$ possible scores. Initially, these cells are empty, except the bottom cell, σ^M_0, which starts with one copy of symbol, k.

Figure 4 shows all these cells, without initialisation, for our sample scenario indicated in Example 4.

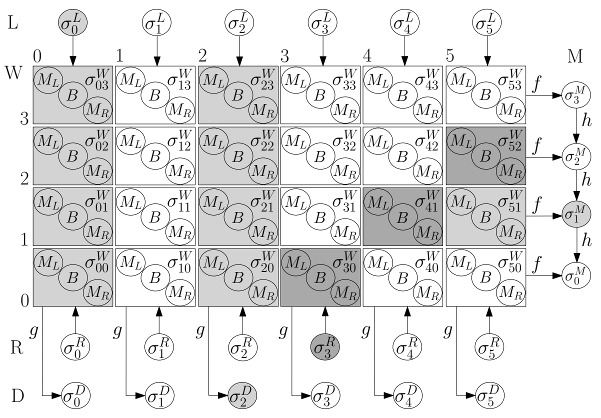


Figure 4: Cells of Π and arcs between major components.

Workspace initialisation. The W-cells are initialised as follows:

- Each cell above the first S/W to N/E diagonal, σ_{ij}^W, $i \in \mathbb{X}$, $j \in \mathbb{D}$, $j > i$, contains one copy of symbols w'_a, w'_b and w'_c, interpreted as the *infinite* values of its M_L, B and M_R subcells, respectively.

- Each other top-row cell, $\sigma_{i,d_{\max}}^W$, $i \in \mathbb{X}$, $i \geq d_{\max}$, contains one copy of symbols w'_a and w'_c (interpreted the same way as above).

- Each cell on or below the first S/W to N/E diagonal, σ_{ij}^W, $i \in \mathbb{X}$, $j \in \mathbb{D}$, $j \leq i$, contains θ copies of symbol t, interpreted as the dissimilarity *threshold.*

- The top-left cell, $\sigma_{0,d_{\max}}^W$, contains one copy of symbol k, which *triggers* the whole computation.

- Each other W-cell contains $\delta_{\texttt{occl}}$ copies of symbol o_a and $\delta_{\texttt{occl}}$ copies of symbol o_c, which indicate the *occlusion* weight parameters for its M_L and M_R subcells, respectively.

Figure 5 shows a fully initialised P system Π, for the scenario of Example 4, just before it starts.

Note. As much as possible, *symbols a, b, c* will be associated with M_L, B, M_R data, respectively. However, this convention does *not* apply to arc *labels.*

Arcs. The cells are linked by arcs in a *labelled directed multigraph*, Δ. We have *cross-component* arcs between our five major components: $L, R \to W \to M, D$ and *internal* arcs in our two "workhorse" components: $W \to W$, $M \to M$. The arcs are incrementally constructed by the following enumeration:

- Each L-cell is parent of its corresponding W-column, in the S direction: $\sigma_i^L \to \sigma_{ij}^W$, $i \in \mathbb{X}$, $j \in \mathbb{D}$.

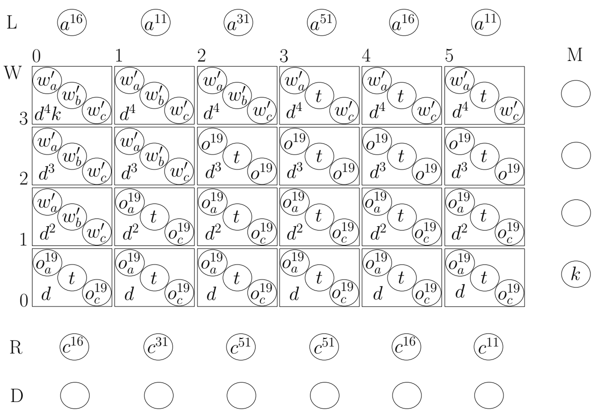


Figure 5: Initialised P system Π (arcs omitted).

- Each R-cell is parent of a corresponding W-diagonal, running S/W to N/E: $\sigma_i^R \rightarrow \sigma_{i+j,j}^W$, $i \in \mathbb{X}$, $j \in \mathbb{D}$, $i + j \in \mathbb{X}$.

- Each W-cell before the last column and below the top row is parent, via an a-labelled arc, of the W-cell in the N/E direction: $\sigma_{ij}^W \overset{a}{\rightarrow} \sigma_{i+1,j+1}^W$, $i \in \mathbb{X} \setminus \{n-1\}$, $j \in \mathbb{D} \setminus \{d_{\max}\}$.

- Each W-cell before the last column is *twice* parent, via b- and c-labelled arcs, of the W-cell following it, in the E direction: $\sigma_{ij}^W \overset{b,c}{\rightarrow} \sigma_{i+1,j}^W$, $i \in \mathbb{X} \setminus \{n-1\}$, $j \in \mathbb{D}$.

- Each W-cell above the bottom row is *twice* parent, via d- and e-labelled arcs, of the W-cell below it, in its S direction: $\sigma_{ij}^W \overset{d,e}{\rightarrow} \sigma_{i,j-1}^W$, $i \in \mathbb{X}$, $j \in \mathbb{D} \setminus \{0\}$.

- Each rightmost column W-cell is parent, via an f-labelled arc, of its corresponding M-cell, in the E direction: $\sigma_{n-1,j}^W \overset{f}{\rightarrow} \sigma_j^M$, $j \in \mathbb{D}$.

- Each W-cell is parent, via a g-labelled arc, of its corresponding D-cell, in the S direction: $\sigma_{ij}^W \overset{g}{\rightarrow} \sigma_i^D$, $i \in \mathbb{X}$, $j \in \mathbb{D}$.

- Each M-cell above the bottom is parent, via an h-labelled arc, of its underlying M-cell, in the S direction: $\sigma_j^M \overset{h}{\rightarrow} \sigma_{j-1}^M$, $j \in \mathbb{D} \setminus \{0\}$.

Example 5. Even for our simple example, a full picture of all these arcs is impossible, because of its sheer complexity. However, we can suggest a representative fragment of these arcs.

Using highlighting, Figure 4 suggests the following cross-component arcs ($L, R \rightarrow W \rightarrow M, D$):

- $\sigma_0^L \rightarrow \sigma_{03}^W, \sigma_{02}^W, \sigma_{01}^W, \sigma_{00}^W$;

- $\sigma_3^R \rightarrow \sigma_{30}^W, \sigma_{41}^W, \sigma_{52}^W$;

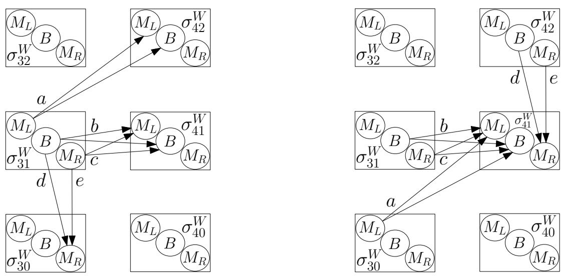


Figure 6: Typical arcs between W-cells. Here arc representations are extended, to indicate the logical message flows, from logical subcell to logical subcell.

- $\sigma_{51}^W \xrightarrow{f} \sigma_1^M$;
- $\sigma_{23}^W, \sigma_{22}^W, \sigma_{21}^W, \sigma_{20}^W \xrightarrow{g} \sigma_2^D$.

Figure 4 also shows internal arcs in the M component:

- $\sigma_3^M \xrightarrow{h} \sigma_2^M$,
- $\sigma_2^M \xrightarrow{h} \sigma_1^M$,
- $\sigma_1^M \xrightarrow{h} \sigma_0^M$.

Figure 6 shows internal arcs between W-cells. In this figure, the arc tails and heads are extended, to suggest the logical message flows, between logical subcells. For example, arc $\sigma_{31}^W \xrightarrow{b} \sigma_{41}^W$, although "physically" just links these two cells, logically, it channels data from σ_{31}^W's subcell B to σ_{41}^W's subcells M_L and B.

Figure 6, left, shows internal arcs, from the "point-of-view" of a sending cell, σ_{31}^W:

- $\sigma_{31}^W \xrightarrow{a} \sigma_{42}^W$;
- $\sigma_{31}^W \xrightarrow{b,c} \sigma_{41}^W$;
- $\sigma_{31}^W \xrightarrow{d,e} \sigma_{30}^W$.

Figure 6, right, shows internal arcs, from the "point-of-view" of a receiving cell, σ_{41}^W:

- $\sigma_{30}^W \xrightarrow{a} \sigma_{41}^W$;
- $\sigma_{31}^W \xrightarrow{b,c} \sigma_{41}^W$;
- $\sigma_{42}^W \xrightarrow{d,e} \sigma_{41}^W$.

Remark. Arcs a, b, c, d and e, which link logical subcells (B, M_L, M_R), correspond to profile transitions between visibility states (B, M_L, M_R), as described in Tables 1, 2 and 3. For example, arc b, which links logical subcell B to logical subcells B and M_L, corresponds to transitions #1 and #4.

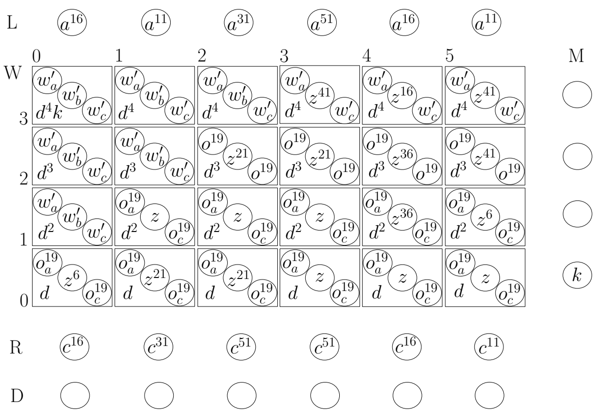


Figure 7: Fully initialised P system Π, just before the start of the *forward pass* phase (II).

Construction. Except actual pixel values of the left and right scanlines, which are specific to each problem instance, the system can be constructed and initialised before its actual usage, starting from (i) two essential parameters: the scanline length, $|\mathbb{X}|$, and the disparity range, $|\mathbb{D}|$; and (ii) two other, less critical parameters (where defaults may be assumed): the dissimilarity threshold, θ, and the occlusion weight, $\delta_{\mathtt{occl}}$. One could imagine that, given these parameters, a single "ur-cell" will multiply and create all required cells and arcs. This construction should require $O(|\mathbb{X}|+|\mathbb{D}|)$ steps only, however, it needs further extensions to the usual P systems framework and we do not develop it here.

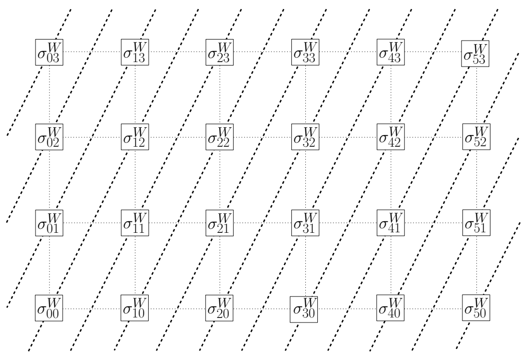
3.2 Evolution—bird's eye view

At a very high level, the system Π evolves through four phases: (I) an initialisation completion phase; (II) a forward pass phase; (III) a global minimum phase; (IV) a backward pass phase; all closely related to the above description of SDPS.

(I) *Initialisation completion.*
Each W-cell on or below the first S/W to N/E diagonal, σ_{ij}^W, $i \in \mathbb{X}$, $j \in \mathbb{D}$, $j \leq i$, determines, in its binocular subcell B, a preliminary score, encoded in base z, which is the thresholded absolute difference between the pixel values received from the left scanline and the right scanline, via $L, R \to W$ arcs. At the end of this preliminary phase, all W-cells have values for all their subcells, M_L, B and M_R. Figure 7 shows a snapshot of the sample system Π, at the end of phase (I).

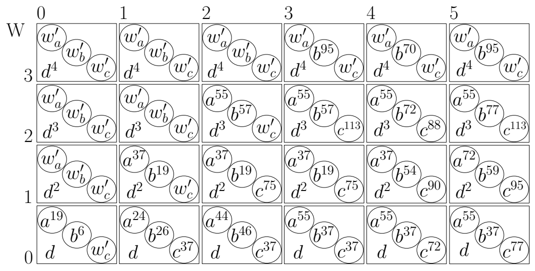
(II) *Forward pass.*
The forward pass phase proceeds on slope 2 diagonals, starting from the top-left cell (initially marked with k). All cells on the same diagonal work in parallel. In our sample scenario, the computational *wave* moves, in order, over the diagonals $\{\sigma_{03}^w\}$, $\{\sigma_{02}^w\}$, $\{\sigma_{01}^w, \sigma_{13}^w\}$, $\{\sigma_{00}^w, \sigma_{12}^w\}$, $\{\sigma_{11}^w, \sigma_{23}^w\}$, $\{\sigma_{10}^w, \sigma_{22}^w\}$, $\{\sigma_{21}^w, \sigma_{33}^w\}$, $\ldots$,
$\{\sigma_{40}^w, \sigma_{52}^w\}$, $\{\sigma_{51}^w\}$ and $\{\sigma_{50}^w\}$. Figure 8 indicates this progression of the computational wave.

During the forward pass, in the general case, each cell σ_{ij}^W determines:



σ^W_{03} σ^W_{13} σ^W_{23} σ^W_{33} σ^W_{43} σ^W_{53}
σ^W_{02} σ^W_{12} σ^W_{22} σ^W_{32} σ^W_{42} σ^W_{52}
σ^W_{01} σ^W_{11} σ^W_{21} σ^W_{31} σ^W_{41} σ^W_{51}
σ^W_{00} σ^W_{10} σ^W_{20} σ^W_{30} σ^W_{40} σ^W_{50}

Figure 8: Progression on the computational wave during the *forward pass* phase (II): starting from the top-left cell, σ^W_{03}, on slope 2 diagonals.



0 1 2 3 4 5
W
3
w'_a w'_b d^4 w'_c
w'_a w'_b d^4 w'_c
w'_a w'_b d^4 w'_c
w'_a b^{95} d^4 w'_c
w'_a b^{70} d^4 w'_c
w'_a b^{95} d^4 w'_c
2
w'_a w'_b d^3 w'_c
w'_a w'_b d^3 w'_c
a^{55} b^{57} d^3 w'_c
a^{55} b^{57} d^3 c^{113}
a^{55} b^{72} d^3 c^{88}
a^{55} b^{77} d^3 c^{113}
1
w'_a w'_b d^2 w'_c
a^{37} b^{19} d^2 w'_c
a^{37} b^{19} d^2 c^{75}
a^{37} b^{19} d^2 c^{75}
a^{37} b^{54} d^2 c^{90}
a^{72} b^{59} d^2 c^{95}
0
a^{19} b^{6} d w'_c
a^{24} b^{26} d c^{37}
a^{44} b^{46} d c^{37}
a^{55} b^{37} d c^{37}
a^{55} b^{37} d c^{72}
a^{55} b^{37} d c^{77}

Figure 9: Final scores of the subcells of the W-cells at the end of the *forward pass* phase (II).

1. In its B and M_L subcells, the sum between its previous scores, as completed in phase (I), and the minimum value of those received:

   - via arc a: the M_L score of $\sigma^W_{i-1,j-1}$, if $i > 0$, $j > 0$;
   - via arc b: the B score of $\sigma^W_{i-1,j}$, if $i > 0$;
   - via arc c: the M_R score of $\sigma^W_{i-1,j}$, if $i > 0$.

   Essentially, these operations corresponds to Formulas 9 and 10.

2. In its M_R subcell, the sum between its previous score and the minimum score of those received:

   - via arc d: the B score of $\sigma^W_{i,j+1}$, if $j < d_{\max}$;
   - via arc e: the M_R score of $\sigma^W_{i,j+1}$, if $j < d_{\max}$.

   Essentially, these operations corresponds to Formula 12.

Figure 9 shows the final scores of the W-cells at the end of the forward pass phase.

Additionally, each W-cell keeps *pointers* to the origin of the *minimum* received scores, which are *symbols*, p_a, p_b, p_c, p_d, p_e, indicating *labels* of the corresponding via arc, a, b, c, d, e, which brought in the minimum scores. These pointers are shown in Figure 11 and

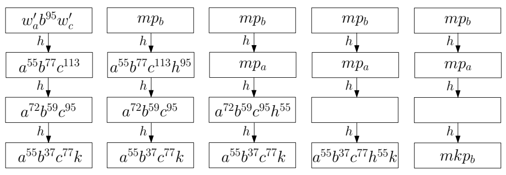


Figure 10: Determination of the global minimum by M-cells, phase (III).

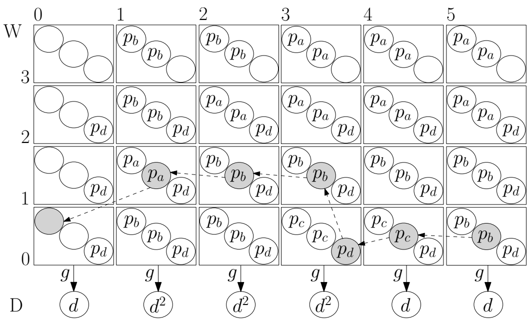


Figure 11: The *backward pass* traversal, phase (IV), and final values of D-cells.

will be used in the backward pass phase. When the wave reaches cells in the rightmost column of W, their final scores are sent to the corresponding M-cells, via f-labelled arcs.

(III) *Global minimum.* Using two sub-phases, a top-down sweep followed by a bottom-up one, the M-cells determine the global minimum score and trigger the backward pass, using the reverse direction of the f-labelled arcs.

Figure 10 illustrates the evolution of the M-cells for our sample scenario.

(IV) *Backward pass.* The backward pass phase follows back the pointers stored in the forward pass, during the evaluation of the minimum score. W-cells traversed during the backward pass send their *disparity* positions to the corresponding D-cells, via g-labelled arcs, which in the end return the problem's solution. Figure 11 shows the backward pass traversal using the backward pointers; D-cells contain disparity values of one optimal profile, $\mathbf{d}$.

We can either stop with the *first* optimal profile, or explore all optimal profiles. The P-rules presented here explore *all* optimal profiles in a depth-first-search manner. However, an alternative solution uses breadth-first-search, which enables a parallel evaluation of a "best" optimal profile (on additional criteria, not discussed here).

3.3 Rules

Each group of cells has its own state and rule set. Alternatively, all cells could start from the same initial state, s_0, and differentiate later, as required; however, we do not detail this extra path here.

(*L*) An L-cell starts from state s_{10}, broadcasts its a-encoded left pixel value, $g_L(x, y)$, to all its W-children (a W-column), and ends in state s_{11}:

1. $s_{10}\ a \rightarrow_{\texttt{max}} s_{11}\ a\ (l)_{\downarrow\forall}$

Grand total for rule set (L): 1 P step (all L-cells in parallel).

(R) An R-cell starts from state s_{20}, broadcasts its c-encoded right pixel values, $g_R(x,y)$, and ends in state s_{21}:

1. $s_{20}\ c \rightarrow_{\texttt{max}} s_{21}\ c\ (r)_{\downarrow\forall}$

Grand total for rule set (R): 1 P step (all R-cells in parallel).

(W) A W-cell starts from state s_{30} and evolves successively through three major subsets, (W_1), (W_2) and (W_3), idling at their boundaries. Rule sets are presented in their logical activation order. Therefore, rule set (M) will be presented inserted between (W_2) and (W_3).

(W_1) Starts from state s_{30}, computes $\delta(x,y,d)$, as the z-encoded thresholded absolute difference between the received left and right pixel values (a-encoded and c-encoded, respectively), and then idles in state s_{40}:

1. $s_{30}\ l \rightarrow_{\texttt{max}} s_{31}\ |_{w'_b}$
2. $s_{30}\ r \rightarrow_{\texttt{max}} s_{31}\ |_{w'_b}$
3. $s_{30}\ l\ r \rightarrow_{\texttt{min}} s_{31}\ v$
4. $s_{30}\ l\ r \rightarrow_{\texttt{max}} s_{31}$
5. $s_{30}\ l \rightarrow_{\texttt{max}} s_{31}\ v$
6. $s_{30}\ r \rightarrow_{\texttt{max}} s_{31}\ v$
7. $s_{31}\ t\ v \rightarrow_{\texttt{min}} s_{40}\ z$
8. $s_{31}\ t\ v \rightarrow_{\texttt{max}} s_{40}$
9. $s_{31}\ t \rightarrow_{\texttt{max}} s_{40}$
10. $s_{31}\ v \rightarrow_{\texttt{max}} s_{40}\ z$
11. $s_{31}\ \rightarrow_{\texttt{min}} s_{40}$

Grand total for rule set (W_1): 1 P step (all W-cells in parallel).

(W_2) Conceptually, this rule set corresponds to the *forward pass* and is further partitioned in its own subsets: $(W_{2.0})$, $(W_{2.1})$, ..., $(W_{2.5})$.

Table 4 shows traces of the *forward pass* phase of the first three and last three diagonals (rounds) of the W-cells.

$(W_{2.0})$ All W-cells idle in state s_{40}, until they receive the wave triggering symbol k—as indicated at $(W_{2.5})$ below—after which they continue through state s_{41}. The top-left W-cell, $\sigma_{0,d_{\max}}$, makes an exception: it continues without delay, because it already has one k, from the initialisation process.

1. $s_{40}\ k \rightarrow_{\texttt{min}} s_{41}\ k$

Total for rule set $(W_{2.0})$: 1 P step, for each W-cell, *after* receiving the wave triggering symbol, k.

($W_{2.1}$) A W-cell transits from state s_{41} to state s_{42}, computes $\min\{F_y(\mathbf{v'}_B), F_y(\mathbf{v'}_{M_R}), F_y(\mathbf{v'}_{M_L})\}$, as the x-encoded minimum value of those received via arcs a, b and c (a-, b-, c-encoded, respectively), with due attention to infinite boundary conditions, and stores the proper *backward pointer*, p_a, p_b or p_c.

1. $s_{41} \rightarrow_{\texttt{min}} s_{42}\ p_b \ |_{w_a\ w_b\ w_c}$
2. $s_{41}\ w_a\ w_c \rightarrow_{\texttt{min}} s_{42}$
3. $s_{41} \rightarrow_{\texttt{min}} s_{42}\ p_a \ |_{w_b\ w_c}$
4. $s_{41}\ a \rightarrow_{\texttt{max}} s_{42}\ x \ |_{w_b\ w_c}$
5. $s_{41}\ w_b\ w_c \rightarrow_{\texttt{min}} s_{42}$
6. $s_{41}\ a\ b \rightarrow_{\texttt{max}} s_{42}\ x \ |_{w_c\ a\ b}$
7. $s_{41}\ a \rightarrow_{\texttt{max}} s_{42}\ p_b \ |_{w_c\ a\ b}$
8. $s_{41}\ b \rightarrow_{\texttt{max}} s_{42}\ p_a \ |_{w_c\ a\ b}$
9. $s_{41}\ a \rightarrow_{\texttt{max}} s_{42} \ |_{w_c\ a\ b}$
10. $s_{41}\ b \rightarrow_{\texttt{max}} s_{42} \ |_{w_c\ a\ b}$
11. $s_{41}\ a\ b \rightarrow_{\texttt{min}} s_{42}\ x \ |_{w_c}$
12. $s_{41} \rightarrow_{\texttt{min}} s_{42}\ p_b \ |_{w_c}$
13. $s_{41}\ b \rightarrow_{\texttt{max}} s_{42}\ x \ |_{w_c}$
14. $s_{41}\ w_c \rightarrow_{\texttt{min}} s_{42}$
15. $s_{41}\ a\ b\ c \rightarrow_{\texttt{max}} s_{42}\ x \ |_{a\ b\ c}$
16. $s_{41}\ a\ b \rightarrow_{\texttt{min}} s_{42}\ p_c \ |_{a\ b\ c}$
17. $s_{41}\ b\ c \rightarrow_{\texttt{min}} s_{42}\ p_a \ |_{a\ b\ c}$
18. $s_{41}\ a\ c \rightarrow_{\texttt{min}} s_{42}\ p_b \ |_{a\ b\ c}$
19. $s_{41}\ a \rightarrow_{\texttt{min}} s_{42}\ p_b \ |_{a\ b\ c}$
20. $s_{41}\ b \rightarrow_{\texttt{min}} s_{42}\ p_c \ |_{a\ b\ c}$
21. $s_{41}\ c \rightarrow_{\texttt{min}} s_{42}\ p_b \ |_{a\ b\ c}$
22. $s_{41}\ a \rightarrow_{\texttt{max}} s_{42} \ |_{a\ b\ c}$
23. $s_{41}\ b \rightarrow_{\texttt{max}} s_{42} \ |_{a\ b\ c}$
24. $s_{41}\ c \rightarrow_{\texttt{max}} s_{42} \ |_{a\ b\ c}$
25. $s_{41}\ a\ b\ c \rightarrow_{\texttt{max}} s_{42}\ x$
26. $s_{41}\ b\ c \rightarrow_{\texttt{max}} s_{42}\ x$
27. $s_{41}\ b \rightarrow_{\texttt{min}} s_{42}\ p_c$
28. $s_{41}\ c \rightarrow_{\texttt{min}} s_{42}\ p_b$
29. $s_{41}\ b \rightarrow_{\texttt{max}} s_{42}$
30. $s_{41}\ c \rightarrow_{\texttt{max}} s_{42}$
31. $s_{41} \rightarrow_{\texttt{min}} s_{42}\ p_c$

Total for rule set ($W_{2.1}$): 1 P step, for each W-cell.

($W_{2.2}$) A W-cell transits from state s_{42} to state s_{43}, computes $\min\{F_y(\mathbf{v'}_B), F_y(\mathbf{v'}_{M_R})\}$, as the y-encoded minimum value of those received via arcs d and e (d-, e-encoded, respectively), with due attention to infinite boundary conditions, and stores the proper *backward pointer*, p_d or p_e.

1. $s_{42}\ d \rightarrow_{\texttt{max}} s_{43}\ y \ |_{w_e}$
2. $s_{42}\ w_e \rightarrow_{\texttt{min}} s_{43}\ p_d$
3. $s_{42} \rightarrow_{\texttt{min}} s_{43}\ p_d \ |_{d\ e}$
4. $s_{42}\ d\ e \rightarrow_{\texttt{max}} s_{43}\ y$
5. $s_{42}\ d \rightarrow_{\texttt{min}} s_{43}\ p_e$
6. $s_{42}\ e \rightarrow_{\texttt{min}} s_{43}\ p_d$
7. $s_{42}\ d \rightarrow_{\texttt{max}} s_{43}$
8. $s_{42}\ e \rightarrow_{\texttt{max}} s_{43}$
9. $s_{42} \rightarrow_{\texttt{min}} s_{43} \ |_{w'_a\ w'_c}$

Total for rule set $(W_{2.2})$: 1 P step, for each W-cell.

$(W_{2.3})$ A W-cell transits from state s_{43} to state s_{44}, computing: (i) $F_y(\mathbf{v}_B)$, as the b-encoded sum of the z-encoded $\delta(x, y, z)$ with the x-encoded minimum determined by $(W_{2.1})$; and (ii) $F_y(\mathbf{v}_{M_L})$, as the a-encoded sum of the o_a-encoded $\delta_{\texttt{occl}}$ with the x-encoded minimum determined by $(W_{2.1})$.

1. $s_{43}\ x \rightarrow_{\texttt{min}} s_{44}\ |_{o_a}$
2. $s_{43}\ o_a \rightarrow_{\texttt{max}} s_{44}\ a\ |_{o_a}$
3. $s_{43}\ z \rightarrow_{\texttt{max}} s_{44}\ b\ |_{o_a}$
4. $s_{43}\ x \rightarrow_{\texttt{max}} s_{44}\ a\ b\ |_{o_a}$
5. $s_{43}\ o_a \rightarrow_{\texttt{min}} s_{44}\ a$
6. $s_{43}\ w_b \rightarrow_{\texttt{min}} s_{44}$
7. $s_{43}\ z \rightarrow_{\texttt{min}} s_{44}$
8. $s_{43}\ z \rightarrow_{\texttt{max}} s_{44}\ b$
9. $s_{43}\ x \rightarrow_{\texttt{max}} s_{44}\ b$
10. $s_{43} \rightarrow_{\texttt{min}} s_{44}$

Total for rule set $(W_{2.3})$: 1 P step, for each W-cell.

$(W_{2.4})$ A W-cell transits from state s_{44} to state s_{45}, computing $F_y(\mathbf{v}_{M_R})$, as the c-encoded sum of the o_c-encoded $\delta_{\texttt{occl}}$ with the y-encoded minimum determined by $(W_{2.2})$.

1. $s_{44}\ o_c \rightarrow_{\texttt{min}} s_{45}\ w'_c\ |_{w_d}$
2. $s_{44}\ o_c \rightarrow_{\texttt{max}} s_{45}\ |_{w_d}$
3. $s_{44}\ w_d \rightarrow_{\texttt{min}} s_{45}$
4. $s_{44}\ y \rightarrow_{\texttt{min}} s_{45}\ |_{o_c}$
5. $s_{44}\ o_c \rightarrow_{\texttt{max}} s_{45}\ c\ |_{o_c}$
6. $s_{44}\ y \rightarrow_{\texttt{max}} s_{45}\ c\ |_{o_c}$
7. $s_{44}\ o_c \rightarrow_{\texttt{min}} s_{45}\ c$
8. $s_{44} \rightarrow_{\texttt{min}} s_{45}$

Total for rule set $(W_{2.4})$: 1 P step, for each W-cell.

$(W_{2.5})$ At this stage, a W-cell transits from state s_{45} to state s_{60}, forwards the wave trigger symbol k and its computed values to its neighbouring W-cells, and, if the cell is in the last column, to its neighbour M-cell:

- via arc a: one k
- via arcs a and f: the a-encoded $F_y(\mathbf{v}_{M_L})$
- via arc b and f: the b-encoded $F_y(\mathbf{v}_B)$
- via arc c and f: the c-encoded $F_y(\mathbf{v}_{M_R})$
- via arc d: one k plus a d-encoded equivalent of the b-encoded $F_y(\mathbf{v}_B)$
- via arc e: an e-encoded equivalent of the c-encoded $F_y(\mathbf{v}_{M_R})$

Infinity symbols are used, if actual values are not available, i.e. for boundary cells. Note that k is properly sent, over arcs a and d, but not over arcs b and c: this ensures that the computing wave moves as required, as a slope 2 diagonal. A W-cell receiving k will break out its idling and resume with rule set $(W_{2.0})$. Although not required, the given rules keep copies of the sent symbols, for visualisation purposes.

1. $s_{45}\ k \rightarrow_{\mathtt{min}} s_{60}\ (k)_{\downarrow_a}\ (k)_{\downarrow_d}$
2. $s_{45}\ k \rightarrow_{\mathtt{max}} s_{60}$
3. $s_{45}\ a \rightarrow_{\mathtt{max}} s_{60}\ a'\ (a)_{\downarrow_a}\ (a)_{\downarrow_f}$
4. $s_{45}\ b \rightarrow_{\mathtt{max}} s_{60}\ b'\ (b)_{\downarrow_b}\ (d)_{\downarrow_d}\ (b)_{\downarrow_f}$
5. $s_{45}\ c \rightarrow_{\mathtt{max}} s_{60}\ c'\ (c)_{\downarrow_c}\ (e)_{\downarrow_e}\ (c)_{\downarrow_f}$
6. $s_{45}\ w'_a \rightarrow_{\mathtt{min}} s_{60}\ w'_a\ (w_a)_{\downarrow_a}\ (w'_a)_{\downarrow_f}$
7. $s_{45}\ w'_b \rightarrow_{\mathtt{min}} s_{60}\ w'_b\ (w_b)_{\downarrow_b}\ (w_d)_{\downarrow_d}\ (w'_b)_{\downarrow_f}$
8. $s_{45}\ w'_c \rightarrow_{\mathtt{min}} s_{60}\ w'_c\ (w_c)_{\downarrow_c}\ (w_e)_{\downarrow_e}\ (w'_c)_{\downarrow_f}$

Total for rule set $(W_{2.5})$: 1 P step, for each W-cell.

Total for all six rule sets (W_2): 6 P steps, for each W-cell, *after* receiving the wave triggering symbol, k.

During this forward pass phase, all W-cells on the same wave (slope 2 diagonal) work in parallel, achieving maximum parallelism on the disparity axis, $\mathbb{D}$. Thus, for all W-cells, grand total for rule set (W_2): $O(|\mathbb{X}|)$ P steps.

(M) M-cells start from state s_50 and determine the overall minimum of all the values received via arcs f, as indicated in rule set $(W_{2.5})$. The work is divided into two subphases:

1. $s_{50} \Rightarrow s_{51} \Rightarrow s_{52} \Rightarrow s_{53}$: Firstly, a top-down sweep, over arcs h. If the current W-cell holds the minimum so far, then the proper backward pointer is recorded, as p_a, p_b or p_c. The sweep ends at the bottom M-cell, which was initially marked with one k.
2. $s_{53} \Rightarrow s_{54}$ Secondly, a bottom-up sweep, over the reverse direction of arcs h. This sweep stops on the first M-cell holding a backward pointer, which indicates the global minimum. Next, the existing backward pointer, p_a, p_b or p_c, is used to send the "right" backtrack token, a, b or c, respectively.

1. $s_{50}\ b \rightarrow_{\mathtt{min}} s_{51}\ p_b\ m\ q\ |_{w'_c}$
2. $s_{50}\ b \rightarrow_{\mathtt{max}} s_{51}\ q\ |_{w'_c}$
3. $s_{50}\ w'_a\ w'_c \rightarrow_{\mathtt{min}} s_{51}$
4. $s_{50}\ a\ b\ c\ h \rightarrow_{\mathtt{max}} s_{51}\ q$
5. $s_{51}\ a \rightarrow_{\mathtt{max}} s_{52}\ |_a$
6. $s_{51}\ b \rightarrow_{\mathtt{max}} s_{52}\ |_b$
7. $s_{51}\ c \rightarrow_{\mathtt{max}} s_{52}\ |_c$
8. $s_{51}\ h \rightarrow_{\mathtt{max}} s_{52}\ |_h$
9. $s_{51}\ a\ b\ c \rightarrow_{\mathtt{min}} s_{52}$
10. $s_{51}\ b\ c \rightarrow_{\mathtt{min}} s_{52}\ p_a$
11. $s_{51}\ b \rightarrow_{\mathtt{min}} s_{52}\ p_c$
12. $s_{51}\ a \rightarrow_{\mathtt{min}} s_{52}\ p_b$
13. $s_{51}\ c \rightarrow_{\mathtt{min}} s_{52}\ p_b$
14. $s_{51}\ h \rightarrow_{\mathtt{min}} s_{52}\ m$
15. $s_{51}\ q \rightarrow_{\mathtt{max}} s_{52}\ (h)_{\downarrow_h}$
16. $s_{52}\ k \rightarrow_{\mathtt{min}} s_{53}\ k$

17. $s_{53}\ k\ p_a \rightarrow_{\mathtt{min}} s_{54}\ (a)_{\uparrow_f}\ |_m$

18. $s_{53}\ k\ p_b \rightarrow_{\mathtt{min}} s_{54}\ (b)_{\uparrow_f}\ |_m$

19. $s_{53}\ k\ p_c \rightarrow_{\mathtt{min}} s_{54}\ (c)_{\uparrow_f}\ |_m$

20. $s_{53}\ k \rightarrow_{\mathtt{max}} s_{54}\ (k)_{\uparrow_h}$

Total for rule set (M): 3 or 4 P steps for each M-cell, thus, for all cells, a grand total of $O(|\mathbb{D}|)$ P steps.

(W_3) This rule set start from state $s_6 0$ and implements the final, *backward pass*, phase. It starts from one of the M-cells and is further relayed by the activated W-cells, in the reverse order of the optimal path. The existing backward pointers, stored during the forward pass, p_a, p_b or p_c, are used to further relay the "right" backtrack tokens, a, b or c, respectively. At the same time, but only if it has received an a or a b, the current W-cell sends its g-encoded disparity position to its corresponding D-cell, via arc g. This ensures that, for each column, only the largest disparity is recorded. Thus, when the backtrack ends, the D-cells will contain a d-encoded reduced profile (projected on the $\mathbb{X}$ axis).

1. $s_{60}\ g \rightarrow_{\mathtt{max}} s_{61}\ g\ (d)_{\downarrow_g}\ |_a$
2. $s_{60}\ g \rightarrow_{\mathtt{max}} s_{61}\ g\ (d)_{\downarrow_g}\ |_b$
3. $s_{60}\ a \rightarrow_{\mathtt{min}} s_{61}\ a\ (a)_{\uparrow_a}\ |_{p_a}$
4. $s_{60}\ a \rightarrow_{\mathtt{min}} s_{61}\ a\ (b)_{\uparrow_b}\ |_{p_b}$
5. $s_{60}\ a \rightarrow_{\mathtt{min}} s_{61}\ a\ (c)_{\uparrow_c}\ |_{p_c}$
6. $s_{60}\ b \rightarrow_{\mathtt{min}} s_{61}\ b\ (a)_{\uparrow_a}\ |_{p_a}$
7. $s_{60}\ b \rightarrow_{\mathtt{min}} s_{61}\ b\ (b)_{\uparrow_b}\ |_{p_b}$
8. $s_{60}\ b \rightarrow_{\mathtt{min}} s_{61}\ b\ (c)_{\uparrow_c}\ |_{p_c}$
9. $s_{60}\ c \rightarrow_{\mathtt{min}} s_{61}\ c\ (b)_{\uparrow_d}\ |_{p_d}$
10. $s_{60}\ c \rightarrow_{\mathtt{min}} s_{61}\ c\ (c)_{\uparrow_e}\ |_{p_e}$

Total for rule set (W_3): 1 P step, for each W-cell along the optimal profile, sequentially. Thus, a grand total of $O(|\mathbb{X}|)$ P steps.

3.4 All optimal solutions.

The above system can be enhanced to efficiently explore all optimal solutions, either sequentially, using a depth-first-search method (DFS), or in parallel, using a breadth-first-search method (BFS). The actual extension rules are not presented here. We refer the reader to our papers [3, 11], where we analyse a variety of efficient methods for DFS and BFS in P systems, leveraging their parallel and distributed facilities.

3.5 Runtime complexity.

The asymptotic runtime complexity of our P model, $O(n+d)$ P steps, which is arguably optimal, is summarized by Theorem 6. In contrast, the best implementation on existing parallel hardware is limited, by hardware resources, to $O(nd/q)$, where q is the number of available processors.

Theorem 6. The P system described in Section 3 completes in $O(n+d)$ P steps.

The proof follows from synthesising the sequential and parallel execution patterns of the above rule sets.

4 Impact of P modelling onto SDPS

We designed a massively parallel synchronous P model for implementing a critical part of a symmetric dynamic programming stereo matching algorithm [8]. Our model processes in parallel all potentially optimal similarity scores that trace candidate decisions for all the disparities associated with each current x-coordinate. The theoretical performance of our P model is conceptually comparable to that of a physical parallel processor with an unlimited number of processing elements.

This P modelling exercise has enabled us to not only identify a small bug in an existing practical SDPS implementation (in the C programming language), but also (and what is much more important) *refactor* this algorithm, based on our cell structure. The result is a more robust and flexible version, which allows us to fine tune its parameters and enhance its capabilities, without rewriting it from scratch.

Main impact of the P model onto SDPS stems from the cell structure capability to keep track (at least, in principle) of multiple equivalent optimum solutions. Traditional DP based optimisation assumes implicitly only a single globally optimum solution and thus stores only a single potentially optimal backward transition, e.g., $T_y(\mathbf{v}_s)$ in Section 3, even if there exist more than one equivalent choice. At the same time, the binocular stereo matching belongs to a class of fundamentally ill-posed inverse optical problems and typically has a rich set of equivalent solutions (i.e. visible 3D scene surfaces) resulting in the same globally maximum similarity between the images of a stereo pair depicting the scene. Heuristic regularisation of the stereo matching problem with the non-zero occlusion score allows for reducing the number of equivalent solutions, but still the unique solution is not guaranteed and the traditional DP performs an arbitrary selection from the set of the remaining equivalent solutions.

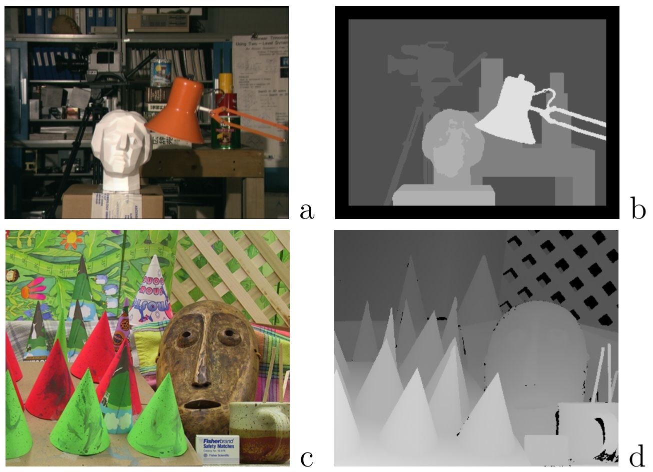


Figure 12: Left stereo images (a,c) and the true disparity maps (b,d) for the "Tsukuba Heads" and "Cones" scenes [2].

Refactoring SDPS in Section 3 accounts for the multiple solutions by storing in each cell at the forward stage all the equivalent potentially optimal backward transitions and marking at the backward stage all the cells included into the equivalent optimal solutions.

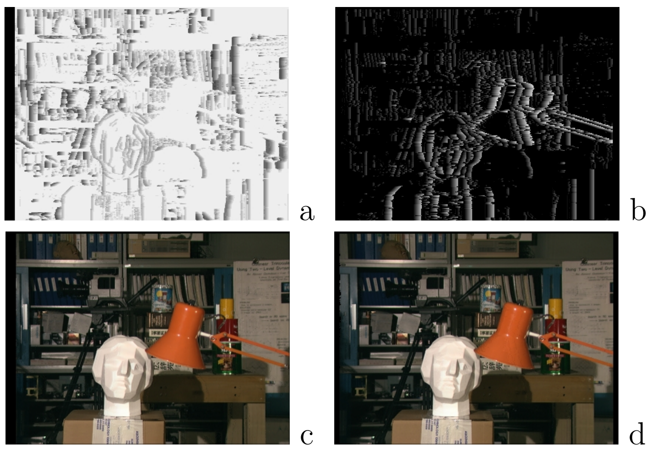


Figure 13: Top (a,c) and bottom (b,d) bounding surfaces and corresponding Cyclopean images, reconstructed for the distance threshold $\theta = 16$ with no regularisation (the occlusion score $\delta_{\texttt{occl}} = 0$) and projected to the left image plane.

In spite of a combinatorial number of the latter, the refactored SDPS allows us to explore the set of the equivalent solutions, e.g. to output its "envelope", which consists of two bounding extreme surfaces. Because each spatial cell $\mathbf{p} = (x, y, d)$ in SDPS has only eight possible backward transitions, three or two per visibility state s, each cell in the refactored SDPS will store at most eight transitions, rather than a single one in the traditional version. Generally, the same refactoring can be applied to the DP solutions of other ill-posed optimisation problems to account for multiple equivalent solutions (at the expense of the extra cell memory).

Figure 12 shows left images of the Middlebury stereo pairs "Tsukuba Head" and "Cones" together with their true disparity maps [2]. Figures 13 and 14 demonstrate the grey-coded top and the bottom bounding surfaces of the disparity maps' envelopes for non-regularised matching and the corresponding Cyclopean images of the surfaces computed with the refactored SDPS and projected onto the left image planes. The images produced by these very different equivalent optimal surfaces are virtually the same as the initial stereo images in Figure 12. Figure 15 illustrates the pixel-wise spans between the envelope surfaces. This and other available structural information about the multiple equivalent solutions could be useful in developing more accurate stereo matching techniques.

5 Conclusions

We presented a massively parallel P model for implementing a critical part of SDPS. Our solution is based on simple P modules, with extensions that allow our model to adapt to complex cases such as SDPS. Our model processes all potentially optimal similarity scores that trace candidate decisions, for all the disparities associated with each current x-coordinate. We plan to further extend the solution, from pairs of scanlines to stereo pairs of full images. To avoid complex and lengthy arguments, we did not discuss in

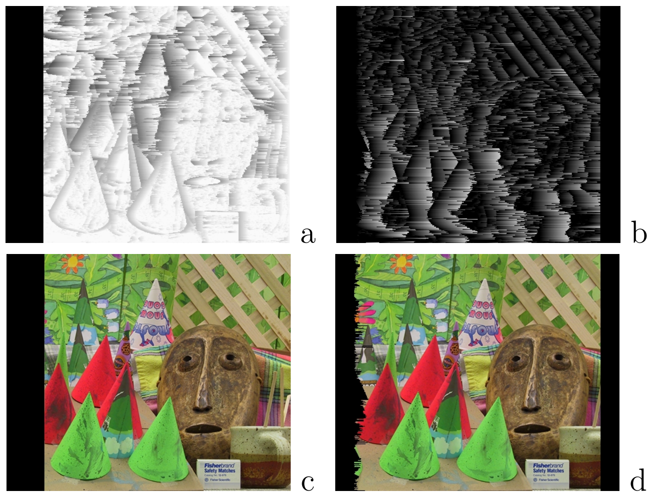


Figure 14: Top (a,c) and bottom (b,d) bounding surfaces and corresponding Cyclopean images, reconstructed for the distance threshold $\theta = 16$ with no regularisation (the occlusion score $\delta_{\mathtt{occl}} = 0$) and projected to the left image plane.

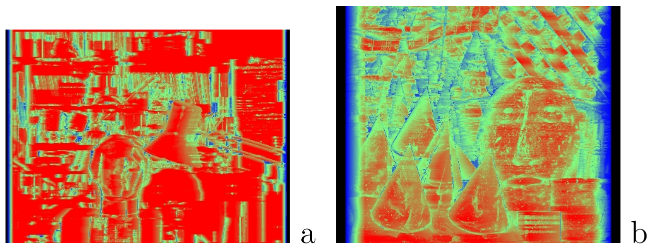


Figure 15: Spans of the disparity envelopes in Figs. 13 and 14 in the blue(0)-to-red(the maximum span) range coding.

this paper how all the cells have been created and initialised. We are investigating an advanced model, which starts as one single "ur-cell" and then, given essential parameters, such as $|\mathbb{X}|$ and $|\mathbb{D}|$, grows until it reaches the proper size and shape, required by SDPS solution. We are also working on improved P system simulators, that are able to leverage the parallel features of this design on a variety of parallel hardwares: multi-cores, many-cores and clusters.

Acknowledgments

The authors wish to acknowledge the contributions of P. Delmas, M.J. Dinneen, Y.-B. Kim, J. Morris, D. Nicolescu and the assistance received via the University of Auckland FRDF grants 9843/3626216 and 3624475.

References

[1] The P systems webpage.

[2] Middlebury stereo vision webpage, 2001.

[3] Tudor Bălănescu, Radu Nicolescu, and Huiling Wu. Asynchronous P systems. *International Journal of Natural Computing Research (IJNCR)*, 2(2):1–18, 2011.

[4] Javier Carnero, Daniel Díaz-Pernil, Helena Molina-Abril, and Pedro Real. Image segmentation inspired by cellular models using hardware programming. In Rocío González-Díaz and Pedro Real-Jurado, editors, *3rd International Workshop on Computational Topology in Image Context*, pages 143–150, 2010.

[5] Hepzibah A. Christinal, Daniel Díaz-Pernil, and Pedro Real. P systems and computational algebraic topology. *Mathematical and Computer Modelling*, 52(11-12):1982–1996, 2010.

[6] Michael J. Dinneen, Yun-Bum Kim, and Radu Nicolescu. A faster P solution for the Byzantine agreement problem. In Marian Gheorghe, Thomas Hinze, and Gheorghe Păun, editors, *Conference on Membrane Computing*, volume 6501 of *Lecture Notes in Computer Science*, pages 175–197. Springer-Verlag, Berlin Heidelberg, 2010.

[7] Michael J. Dinneen, Yun-Bum Kim, and Radu Nicolescu. P systems and the Byzantine agreement. *Journal of Logic and Algebraic Programming*, 79(6):334–349, 2010.

[8] Georgy L. Gimel'farb. Probabilistic regularisation and symmetry in binocular dynamic programming stereo. *Pattern Recognition Letters*, 23(4):431–442, 2002.

[9] Georgy L. Gimel'farb, Radu Nicolescu, and Sharvin Ragavan. P systems in stereo matching. In Real et al. [18], pages 285–292.

[10] Radu Nicolescu, Michael J. Dinneen, and Yun-Bum Kim. Towards structured modelling with hyperdag P systems. *International Journal of Computers, Communications and Control*, 2:209–222, 2010.

[11] Radu Nicolescu and Huiling Wu. BFS solution for disjoint paths in P systems. In Cristian Calude, Jarkko Kari, Ion Petre, and Grzegorz Rozenberg, editors, *Unconventional Computation - 10th International Conference, UC 2011, Turku, Finland, June 6-10, 2011. Proceedings*, volume 6714 of *Lecture Notes in Computer Science*, pages 164–176. Springer Berlin / Heidelberg, 2011.

[12] Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[13] Gheorghe Păun. *Membrane Computing: An Introduction.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[14] Gheorghe Păun. Introduction to membrane computing. In Gabriel Ciobanu, Mario J. Pérez-Jiménez, and Gheorghe Păun, editors, *Applications of Membrane Computing*, Natural Computing Series, pages 1–42. Springer-Verlag, 2006.

[15] Gheorghe Păun and Mario J. Pérez-Jiménez. Solving problems in a distributed way in membrane computing: dP systems. *International Journal of Computers, Communications and Control*, 5(2):238–252, 2010.

[16] Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.

[17] Francisco Peña-Cantillana, Daniel Díaz-Pernil, Ainhoa Berciano, and Miguel A. Gutiérrez-Naranjo. A parallel implementation of the thresholding problem by using tissue-like p systems. In Real et al. [18], pages 277–284.

[18] Pedro Real, Daniel Díaz-Pernil, Helena Molina-Abril, Ainhoa Berciano, and Walter G. Kropatsch, editors. *Computer Analysis of Images and Patterns - 14th International Conference, CAIP 2011, Seville, Spain, August 29-31, 2011, Proceedings, Part II*, volume 6855 of *Lecture Notes in Computer Science*. Springer, 2011.

Table 4: The traces of the *forward pass* phase of the first three and last three diagonals (rounds) of the W-cells (ρ – round; τ – step).

ρ	0	1	2	3	n-2	n-1	n
τ	σ_{03}	σ_{02}	σ_{13}	σ_{12}	σ_{52}	σ_{51}	σ_{50}
			σ_{01}	σ_{00}	σ_{40}		
0	$s_4\ w_lw_bw_r$	$s_6\ w_lw_bw_r$	$s_6\ w_lw_bw_r$	$s_6\ w_lw_bw_r$	$s_6\ o_l^{19}z^{41}o_r^{19}$	$s_6\ o_l^{19}z^6o_r^{19}$	$s_6\ o_l^{19}zo_r^{19}$
			$s_6\ w_lw_bw_r$	$s_6\ o_l^{19}z^6o_r^{19}$	$s_6\ o_l^{19}zo_r^{19}$		
1	$s_5\ w_lw_bw_r$	$s_6\ w_lw_bw_rw^dw^e$	$s_6\ w_lw_bw_rw^aw^bw^c$	$s_6\ w_lw_bw_rw^aw^bw^cw^dw^e$	$s_6\ o_l^{19}z^{41}o_r^{19}a^{37}b^{72}c^{88}d^{95}w^e$	$s_6\ o_l^{19}z^6o_r^{19}a^{55}b^{54}c^{90}d^{77}e^{113}$	$s_6\ o_l^{19}zo_r^{19}b^{37}c^{72}d^{59}e^{95}$
			$s_6\ w_lw_bw_rw^dw^e$	$s_6\ o_l^{19}z^6o_r^{19}w^dw^e$	$s_6\ o_l^{19}zo_r^{19}b^{37}c^{37}d^{54}e^{90}$		
2	$s_7\ w_lw_bw_r$	$s_8\ w_lw_bw_rw^dw^e$	$s_8\ w_lw_bw_rw^aw^bw^c$	$s_8\ w_lw_bw_rw^aw^bw^cw^dw^e$	$s_8\ o_l^{19}z^{41}o_r^{19}a^{37}b72c^{88}d^{95}w^e$	$s_8\ o_l^{19}z^6o_r^{19}a^{55}b^{54}c^{90}d^{77}e^{113}$	$s_8\ o_l^{19}zo_r^{19}b^{37}c^{72}d^{59}e^{95}$
			$s_8\ w_lw_bw_rw^dw^e$	$s_8\ o_l^{19}z^6o_r^{19}w^dw_e$	$s_8\ o_l^{19}zo_r^{19}b^{37}c^{37}d^{54}e^{90}$		
3	$s_{26}\ w_lw_bw_r$	$s_{10}\ w_lw_bw_rw^dE$	$s_{10}\ w_lw_bw_rw^aw^bw^c$	$s_{10}\ w_lw_bw_rw^aw^bw^cw^dE$	$s_9\ o_l^{19}z^{41}o_r^{19}a^{37}b72c^{88}d^{95}E$	$s_{10}\ o_l^{19}z^6o_r^{19}a^{55}b^{54}c^{90}i^{77}E$	$s_{10}\ o_l^{19}zo_r^{19}b^{37}c^{72}i^{59}E$
			$s_{10}\ w_lw_bw_rw^dE$	$s_{10}\ o_l^{19}z^6o_r^{19}w^dE$	$s_{10}\ o_l^{19}zo_r^{19}b^{37}c^{37}i^{54}E$		
4		$s_{11}\ w_lw_bw_rw^dE$	$s_{11}\ w_lw_bw_rw^aw^bw^c$	$s_{11}\ w_lw_bw_rw^aw^bw^cw^dE$	$s_{11}\ o_l^{19}z^{41}o_r^{19}a^{37}b72c^{88}i^{95}E$	$s_{11}\ o_l^{19}z^6o_r^{19}a^{55}b^{54}c^{90}i^{77}E$	$s_{11}\ o_l^{19}zo_r^{19}b^{37}c^{72}i^{59}E$
			$s_{11}\ w_lw_bw_rw^dE$	$s_{11}\ o_l^{19}z^6o_r^{19}w^dE$	$s_{11}\ o_l^{19}zo_r^{19}b^{37}c^{37}i^{54}E$		
5		$s_{12}\ w_lw_bw_rw^dp_d$	$s_{12}\ w_lw_bw_rw^aw^bw^c$	$s_{12}\ w_lw_bw_rw^aw^bw^cw^dp_d$	$s_{12}\ o_l^{19}z^{41}o_r^{19}a^{37}b72c^{88}i^{95}p_d$	$s_{12}\ o_l^{19}z^6o_r^{19}a^{55}b^{54}c^{90}i^{77}p_d$	$s_{12}\ o_l^{19}zo_r^{19}b^{37}c^{72}i^{59}p_d$
			$s_{12}\ w_lw_bw_rw^dp_d$	$s_{12}\ o_l^{19}z^6o_r^{19}w^dp_d$	$s_{12}\ o_l^{19}zo_r^{19}b^{37}c^{37}i^{54}p_d$		
6		$s_{13}\ w_lw_bw_rw^dp_d$	$s_{16}\ w_lw_bw_rw^bAC$	$s_{16}\ w_lw_bw_rw^bw^dp_dAC$	$s_{16}\ o_l^{19}z^{41}o_r^{19}i^{95}j^{37}p_dBC$	$s_{16}\ o_l^{19}z^6o_r^{19}i^{77}j^{54}p_dAC$	$s_{13}\ o_l^{19}zo_r^{19}b^{37}c^{72}i^{59}p_d$
			$s_{13}\ w_lw_bw_rw^dp_d$	$s_{13}\ o_l^{19}z^6o_r^{19}w^dp_d$	$s_{13}\ o_l^{19}zo_r^{19}b^{37}c^{37}i^{54}p_d$		
7		$s_{17}\ w_lw_bw_rw^dp_d$	$s_{17}\ w_lw_bw_rw^bAC$	$s_{17}\ w_lw_bw_rw^bw^dp_dAC$	$s_{17}\ o_l^{19}z^{41}o_r^{19}i^{95}j^{37}p_dBC$	$s_{17}\ o_l^{19}z^6o_r^{19}i^{77}j^{54}p_dAC$	$s_{17}\ o_l^{19}zo_r^{19}i^{59}j^{37}p_dC$
			$s_{17}\ w_lw_bw_rw^dp_d$	$s_{17}\ o_l^{19}z^6o_r^{19}w^dp_d$	$s_{17}\ o_l^{19}zo_r^{19}i^{54}j^{37}p_dB$		
8		$s_{18}\ w_lw_bw_rw^dp_d$	$s_{18}\ w_lw_bw_rw^bAC$	$s_{18}\ w_lw_bw_rw^bw^dp_dAC$	$s_{18}\ o_l^{19}z^{41}o_r^{19}i^{95}j^{37}p_dBC$	$s_{18}\ o_l^{19}z^6o_r^{19}i^{77}j^{54}p_dAC$	$s_{18}\ o_l^{19}zo_r^{19}i^{59}j^{37}p_dC$
			$s_{18}\ w_lw_bw_rw^dp_d$	$s_{18}\ o_l^{19}z^6o_r^{19}w^dp_d$	$s_{18}\ o_l^{19}zo_r^{19}i^{54}j^{37}p_dB$		
9		$s_{19}\ w_lw_bw_rw^dp_d$	$s_{19}\ w_lw_bw_rw^bp_b$	$s_{19}\ w_lw_bw_rw^bw^dp_bp_d$	$s_{19}\ o_l^{19}z^{41}o_r^{19}i^{95}j^{37}p_ap_d$	$s_{19}\ o_l^{19}z^6o_r^{19}i^{77}j^{54}p_bp_d$	$s_{19}\ o_l^{19}zo_r^{19}i^{59}j^{37}p_bp_d$
			$s_{19}\ w_lw_bw_rw^dp_d$	$s_{19}\ o_l^{19}z^6o_r^{19}w^dp_d$	$s_{19}\ o_l^{19}zo_r^{19}i^{54}j^{37}p_bp_d$		
10		$s_{20}\ w_lw_bw_rw^dp_d$	$s_{20}\ w_l^2w_b^2w_rp_b$	$s_{20}\ w_l^2w_b^2w_rw^dp_bp_d$	$s_{21}\ u_l^{55}u_b^{77}o_r^{19}i^{95}p_ap_d$	$s_{21}\ u_l^{72}u_b^{59}o_r^{19}i^{77}p_bp_d$	$s_{21}\ u_l^{55}u_b^{37}o_r^{19}i^{59}p_bp_d$
			$s_{20}\ w_lw_bw_rw^dp_d$	$s_{21}\ u_l^{19}u_b^6o_r^{19}w^dp_d$	$s_{21}\ u_l^{55}u_b^{37}o_r^{19}i^{54}p_bp_d$		
11		$s_{22}\ w_lw_bw_rw^dp_d$	$s_{22}\ w_lw_bw_rp_b$	$s_{22}\ w_lw_bw_rw^dp_bp_d$	$s_{22}\ u_l^{55}u_b^{77}o_r^{19}i^{95}p_ap_d$	$s_{22}\ u_l^{72}u_b^{59}o_r^{19}i^{77}p_bp_d$	$s_{22}\ u_l^{55}u_b^{37}o_r^{19}i^{59}p_bp_d$
			$s_{22}\ w_lw_bw_rw^dp_d$	$s_{22}\ u_l^{19}u_b^6o_r^{19}w^dp_d$	$s_{22}\ u_l^{55}u_b^{37}o_r^{19}i^{54}p_bp_d$		
12		$s_{23}\ w_lw_bw_r^2p_d$	$s_{23}\ w_lw_bw_rp_b$	$s_{23}\ w_lw_bw_r^2p_bp_d$	$s_{24}\ u_l^{55}u_b^{77}o_r^{19}i^{95}p_ap_d$	$s_{24}\ u_l^{72}u_b^{59}o_r^{19}i^{77}p_bp_d$	$s_{24}\ u_l^{55}u_b^{37}o_r^{19}i^{59}p_bp_d$
			$s_{23}\ w_lw_bw_r^2p_d$	$s_{23}\ u_l^{19}u_b^6o_r^{19}w_rp_d$	$s_{24}\ u_l^{55}u_b^{37}o_r^{19}i^{54}p_bp_d$		
13		$s_{25}\ w_lw_bw_rp_d$	$s_{25}\ w_lw_bw_rp_b$	$s_{25}\ w_lw_bw_rp_bp_d$	$s_{25}\ u_l^{55}u_b^{77}u_r^{113}p_ap_d$	$s_{25}\ u_l^{72}u_b^{59}u_r^{95}p_bp_d$	$s_{25}\ u_l^{55}u_b^{37}u_r^{77}p_bp_d$
			$s_{25}\ w_lw_bw_rp_d$	$s_{25}\ u_l^{19}u_b^6w_rp_d$	$s_{25}\ u_l^{55}u_b^{37}u_r^{72}p_bp_d$		
14		$s_7\ w_lw_bw_rp_d$	$s_7\ w_lw_bw_rp_b$	$s_7\ w_lw_bw_rp_bp_d$	$s_7\ u_l^{55}u_b^{77}u_r^{113}p_ap_d$	$s_7\ u_l^{72}u_b^{59}u_r^{95}p_bp_d$	$s_7\ u_l^{55}u_b^{37}u_r^{77}p_bp_d$
			$s_7\ w_lw_bw_rp_d$	$s_7\ u_l^{19}u_b^6w_rp_d$	$s_7\ u_l^{55}u_b^{37}u_r^{72}p_bp_d$		
15		$s_{26}\ w_lw_bw_rp_d$	$s_{26}\ w_lw_bw_rp_b$	$s_{26}\ w_lw_bw_rp_bp_d$	$s_{26}\ u_l^{55}u_b^{77}u_r^{113}p_ap_d$	$s_{26}\ u_l^{72}u_b^{59}u_r^{95}p_bp_d$	$s_{26}\ u_l^{55}u_b^{37}u_r^{77}p_bp_d$
			$s_{26}\ w_lw_bw_rp_d$	$s_{26}\ u_l^{19}u_b^6w_rp_d$	$s_{26}\ u_l^{55}u_b^{37}u_r^{72}p_bp_d$		