Quick and energy-efficient bayesian computing of binocular disparity using stochastic digital signals

Alexandre Coninx, Pierre Bessière, Jacques Droulez

Quick and energy-efficient bayesian computing of binocular disparity using stochastic digital signals

Alexandre Coninx[a,*], Pierre Bessière[a], Jacques Droulez[a]

[a] *ISIR CNRS/UPMC, 4 place Jussieu 75005 Paris, France*

Abstract

Reconstruction of the tridimensional geometry of a visual scene using the binocular disparity information is an important issue in computer vision and mobile robotics, which can be formulated as a bayesian inference problem. However, computation of the full disparity distribution with an advanced bayesian model is usually an intractable problem, and proves computationally challenging even with a simple model. In this paper, we show how probabilistic hardware using distributed memory and alternate representation of data as stochastic bitstreams can solve that problem with high performance and energy efficiency. We put forward a way to express discrete probability distributions using stochastic data representations and perform bayesian fusion using those representations, and show how that approach can be applied to diparity computation. We evaluate the system using a simulated stochastic implementation and discuss possible hardware implementations of such architectures and their potential for sensorimotor processing and robotics.

Keywords: Bayesian inference, stochastic computing, sensorimotor processing, energy efficiency, hardware implementation, binocular disparity

1. Introduction

Using two cameras in a stereoscopic setup to reconstruct the tridimensional geometry of a visual scene, in a way similar to that performed by human stereopsis, is an important issue in computer vision, with major applications to autonomous robotics (and more specifically autonomous driving (Geiger et al., 2012)). That issue has been an active research topic since at least 40 years, and a wide range of methods and algorithms have been proposed (Scharstein & Szeliski, 2002; Lazaros et al., 2008) and evaluated on standardized benchmarks (Geiger et al., 2013; Scharstein et al., 2014).

Several works have shown that the binocular disparity computation can efficiently be formulated as a bayesian inference problem (Belhumeur, 1996; Su

*Corresponding author
Email address: alexandre.coninx@isir.upmc.fr (Alexandre Coninx)

et al., 2012). The disparity value for each pixel is then expressed as a discrete probability distribution, which can be computed through a probabilistic model using likelihood values specified from the image data. However, computing the full disparity distribution on whole images proves challenging and compuationally demanding. That's why most works on binocular disparity using bayesian models instead reduce the output to a single disparity value per pixel (often using the maximium a-posteriori likelihood estimator). That approach simplifies the computation and allows to reformulate it as an energy minimization problem that can be solved efficiently by classic optimization techniques such as dynamic programming (see Belhumeur, 1996, for an exemple).

However, it means that despite using a probabilistic formalism, the computation yields deterministic disparity values and not disparity distributions, despite the latter representation being richer and offering many benefits, especially for robotics and sensorimotor systems. Full disparity distributions can accurately represent cases where stereopsis is not sufficient to completely determinate the world geometry, such as ambiguous pixels with multiple matches, or pixels with no matches (e.g. due to occlusions). Such probabilistic representations can also directly be used by Bayesian mapping and navigation methods such as the bayesian occupation filter (Coue et al., 2006), and more generally by probabilistic and bayesian robotics techniques (Thrun et al., 2005; Lebeltel, 2006; Bessière et al., 2008). Bayesian inference also provides a powerful framework to express assumptions and prior knowledge about the structure of the world (for example the location of the ground or other known objects) as prior probability distributions.

Stochastic computing is a field dedicated to using intentionally stochastic computers using non-Von Neumann architectures, distributed memory and specific data representations to perform probabilistic reasoning. More specifically, the BAMBI project is a research effort to develop stochastic machines implementing bayesian inference (bayesian machines) (Alves et al., 2015). In this paper, we show how those bayesian machines can be used to efficiently compute full binocular disparity distribution, paving the way towards stochastic autonomous robots and other sensorimotor systems.

In the remainder of this article, we will first give an overview of the related work in section 2, both about stochastic computing and quick binocular disparity computation. We will then describe our bayesian binocular disparity computation model in section 3. Section 4 will be dedicated to the description of the stochastic computer implementing that model, focusing first on the general principles of computation using stochastic bitstream and second to their application to the bayesian disparity computation. The evaluation of that system and its results will be presented in section 5 and further discussed in section 6. We will then conclude in section 7 by summing up the implications of that work for the design of bayesian sensorimotor systems using stochastic components and discussing the future prospects of that topic.

2. Previous work

2.1. Computing with stochastic bitstreams

The general idea of stochastic computations with temporal coding can be traced back to the seminal works of Von Neumann (1956) and Gaines (1969) who highlighted the interest of such data representations, but their approaches were not widely pursued due to the rapid development of more efficient deterministic computers. The topic has recently received a renewed attention due to the development of probabilistic and bayesian models in computer science and engineering – and more specifically for sensorimotor and cognitive systems – and the limitations of classic computers to implement those models.

Several approaches to probabilistic computing have been put forward. Some of them rely on Markov chain Monte Carlo sampling to perform approximate inference (Mansinghka, 2009; Jonas et al., 2014). Closer to our work, Vigoda (2003) proposes to perform exact inference using probability distributions encoded as analog signals. Recent work conducted in the framework of the BAMBI project have proposed using digital signals with temporal coding to perform bayesian inference, and a proof-of-concept to solve a simple sensorimotor problem has been put forward (Faix et al., 2015). In this paper, we apply the same principles to a more computationally challenging bayesian model to highlight their benefits for robotics and sensorimotor systems.

2.2. Disparity computation

As it provides a way to estimate the depth information using data from standard digital cameras, the binocular disparity problem has received a wide attention since the beginnings of computer vision. Existing approaches have been summarized in reviews (Scharstein & Szeliski, 2002; Lazaros et al., 2008), which show that most methods follow the same general structure which can be divided in three steps:

1. Computing a *matching cost*, which is a positive value associated to each possible pair of matching pixels[1] The matching cost is a dissimilarity measure: the least likely the pixels are to match, the higher it is. The cost is computed locally, typically by comparing the luminance or color of individual pixels. The most common matching cost is the squared difference of pixel values (Scharstein & Szeliski, 2002), but some other techniques preprocess the image with operators such as the gradient (Scharstein, 1994) or use banks of linear spatial filters (Jones & Malik, 1992).
2. Applying an optional *cost aggregation*, which performs spatial integration of the pixel-wise information provided by cost values. The main goal of that step is to take into account the fact that most points of the disparity

[1] Most algorithms use rectified image pairs, which allows to only consider pixels on corresponding rows for matching, and limit the disparity to a maximum value D_{max} corresponding to a minimum distance. D_{max} depends on image resolution, camera focal length and visual environment; typical values are 50 to 100 pixels.

map are locally smooth and therefore neighbouring pixels have correlated disparity values. The simplest form of cost aggregation relies on averaging cost values for a given disparity across a given neighborhood.

3. An *optimization* step, which uses the (aggregated) cost to compute the final disparity image. This step can be limited to simply selecting the disparity value associated to the lowest cost in a winner-takes-all way. But it can also involve global computations to optimize the disparity map with regard to a given world model (e.g. smoothness, plane surfaces, etc. (Belhumeur, 1996)), using techniques such as dynamic programming, in which cases it can complement or replace cost aggregation.

2.2.1. Bayesian disparity computation

Several of the existing works (Belhumeur, 1996; Su et al., 2012) use the bayesian inference framework to describe this process. For example, Belhumeur (1996) proposes to reconstruct the scene geometry S from the left and right images I_l and I_r using a bayesian model:

$$P(S|I_l, I_r) \propto P(S) \cdot P(Il, Ir|S) \tag{1}$$

with $P(S)$ being a prior specifying the expected shape (smooth, etc.) of the world and $P(Il, Ir|S)$ a data term computed from the matching cost. Computing $P(Il, Ir|S)$ therefore corresponds to the matching cost computation step, there is no cost aggregation step, and computing and integrating the prior constitutes the optimization step. Belhumeur uses squared difference to compute the cost and proposes three increasingly complex world models to define the prior, but the computation of the full posterior probability distribution – which has cardinality $(D_{max} + 1)^{w \times h}$ – is intractable.

He therefore uses an energy formalism and defines $E[S] = -\log(P(S) \cdot P(Il, Ir|S))$, which allows to compute $\hat{S} = \arg\max_S P(S|I_l, I_r)$ by minimizing $E[S]$, and shows that a simplified form of this optimization problem can be solved by dynamic programming. As mentioned in section 1, despite that algorithm being based on bayesian inference, it only yields a single disparity value for each pixel.

2.2.2. Supervised techniques

The development of public image pairs datasets provided with a disparity baseline such as the KITTI dataset (Geiger et al., 2013) or the Middlebury dataset (Scharstein et al., 2014) have made it possible to treat disparity computation as a supervised machine learning problem. Some algorithms use deep convolutional networks to learn the matching cost (Žbontar & LeCun, 2014; Mayer et al., 2015), and perform cost aggregation and optimization using other techniques.

Those techniques currently top the KITTI leaderboard[2]. Although they are

[2] `http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo`,

extremely accurate on benchmarks, their efficiency depend on the existence of a relevant supervised training dataset. Besides, they are computationally very intensive, using high-end CPUs and GPUs and sometimes requiring a computing time of several minutes per frame. Those features would make applying those techniques in a mobile robotics context challenging.

2.2.3. Sampling approach

An approach that is directly relevant to our positioning is the method proposed by Jonas et al. (2014) as an application of his aforementioned hardware architecture for approximate inference. In that work, he models the disparity distribution using a Markov random field, and use his hardware architecture using Gibbs sampling to sample the posterior distribution. Although this approach is efficient and allows to use a computationally intensive bayesian disparity model with global optimization, it uses a unique, centralized pseudo-random number generator as source of entropy and lacks some of the features of our system, such as the high parallelism and the robust computation of the full disparity with a very low number of clock cycle.

3. Bayesian disparity computation model

3.1. Overview

The goal of the disparity computation is to estimate the tridimensional geometry of a visual scene from two rectified images taken from two identical cameras with focal length f distant from a known baseline distance B. If an object projects into the left camera's image plane I^l at position x and in the right camera's image plane I^r at position $x - d$, its depth Z from the cameras can be computed by $Z = \frac{B \cdot f}{d}$ (see fig. 1). The goal of a disparity algorithm is therefore to identify matching pixels in the two images to compute the disparity.

3.2. Model description

As mentioned in section 2.2.1, the main obstacle to computing full disparity distributions is the very high cardinality of the considered distribution: integrating smoothness constraints in the probabilistic model requires to perform inference on distributions of size $(D_{max}+1)^{N_{\text{pixels}}}$, where N_{pixels} is the number of pixels in the domain on which the optimization is performed. If the optimization is performed on the whole image or on entire rows or columns (as is the case in (Belhumeur, 1996)) the problem becomes completely intractable, but even smaller integration neighborhoods are problematic. In order to avoid that issue, we will perform all of the spatial information integration as image preprocessing operations, and only perform completely local bayesian operations on tractable distributions.

---

consulted 24/03/2016

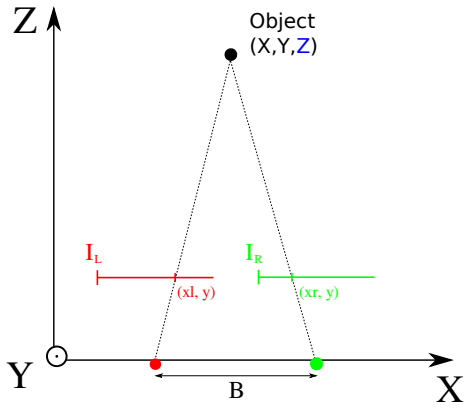


Figure 1: Object observed by two cameras in a stereoscopic vision system. Matching corresponding pixels in the two images is key to computing the disparity d and therefore the depth Z.

Our stereo matching method relies on the preprocessing of images using linear convolution filters to extract relevant features. Other algorithms have used linear spatial filters for disparity computation Jones & Malik (1992), although they use the feature information from those filter in a different way that is not based on. The relevance of using such linear spatial filters as a preprocessing step is also highlighted by recent works using deep neural networks to compute disparity (Žbontar & LeCun, 2014; Mayer et al., 2015), which use a convolutional layer (with filters trained through supervised learning) as their input.

The feature maps output by the filters are used to compute feature matching costs for pixel pairs corresponding to the possible disparities. Those costs are used to compute probabilistic likelihood functions, that are then combined using naive bayesian fusion.

In our method, we only use three simple square spatial filters of size 5 pixels to process images of width W and height H:

- One simple linear averaging filter m;
- One linear horizontalgradient filter g_H;
- One linear vertical gradient filter g_V.

For each of the three filters $f \in \{m, g_H, g_V\}$, we compute the left and right feature maps by applying the filter to the left and right images: $f^l = I^l * f$ and $f^r = I^r * f$. Due to the size of the convolution filters, those feature maps have width $W_f = W - 4$ and height $H_f = H - 4$.

For each pair of feature maps and each possible disparity value we compute a matching cost, using the simple squared difference:

$$C_f(x, y, d) = (f^l(x, y) - f^r(x - d, y))^2 \tag{2}$$

for $d \in [\![0; D_{max}]\!]$ and $x \geq D_{max}$.

The cost shown by 2 measures the dissimilarity between the pixels at coordinate (x, y) in the left image and $(x-d, y)$ in the right image for the feature f. In order to use a bayesian inference framework, we use these costs to compute likelihood probability values:

$$p(f^r(x-d,y)|f^l(x,y),[D(x,y)=d]) = p_0 + (1-p_0)e^{-\frac{C_f(x,y,d)}{2\sigma_f^2}} \tag{3}$$

Equation 3 expresses the likelihood of observing the value $f^r(x-d, y)$ in the right feature map if the value $f^l(x, y)$ is observed in the left feature map and the disparity at coordinates (x, y) is d. That probabilistic formulation allows us to specify a base probability p_0 of the features matching even if the cost is high (which can happen when the two images locally differ for reasons unrelated to the problem, for example because of specular reflections), and a parameter σ_f representing the expected inaccuracy of the cost measurement (a small value of σ_f results in a null or very small cost being required to give a high likelihood value).

In the following, we will drop the (x, y) and $(x-d, y)$ spatial coordinates in equations for better readability. We can compute the disparity distribution from the likelihoods using naive bayesian fusion:

$$p([D=d]|I^l, I^r) \propto p([D=d]) \prod_{f \in \{m, g_H, g_V\}} p(f^r|f^l, [D=d]) \tag{4}$$

where $p(f^r|f^l, [D=d])$ are the the likelihoods computed by eq. 3 and $p(D)$ is a prior on the disparity distribution, which can either be set to uniform or be used to represent prior information about the world (for example, if we know the world contains a flat floor with no holes, the prior probability of disparities corresponding to objects under the floor can be set to zero). $p(D|I^l, I^r)$, is the posterior disparity distribution, which can be used in further probabilistic computation or estimated using the maximum a-posteriori (MAP) estimator: $d^* = \underset{d \in [\![0; D_{max}]\!]}{\arg\max}\ p([D=d]|I^l, I^r)$

4. Stochastic implementation of the model

Previous work (Faix et al., 2015) has shown that naive bayesian fusion could be performed by stochastic machines. In this section, we will describe that structure of a bayesian machine architectured as a matrix of stochastic operators and explain how it can be used to implement the probabilistic binocular disparity computation detailed in section 3.2.

4.1. Stochastic bayesian fusion: the bayesian machine

4.1.1. Probabilities as stochastic bitstreams

Our stochastic computational architecture represents data using *stochastic bitstreams*. Stochastic bitstreams are random digital binary signals that express

a probability value (p-value)) by the proportion of bits set to 1 in a given signal (Fig. 2a). Generating a stochastic bitstream b encoding probability p is therefore done by using a random number generator outputing random bits set to 1 with a probability p. Conversely, extracting the value of p from b and storing it as a floating point or fixed point number requires to integrate information from b on an extended duration to count the proportion of bits set to 1, the precision of the recovered p value increasing with the integration time.

If two probability values p_1 and p_2 are encoded by two uncorrelated stochastic bitstreams b_1 and b_2 and those two signals are input to a logic AND gate, the probability p_{out} of the output signal s_{out} to be in state 1 at a given time is given by :

$$\begin{aligned} p_{out} &= P([s_{out} = 1]) \\ &= P([s_1 = 1] \wedge [s_2 = 1]) \\ &= P([s_1 = 1]) \cdot P([s_2 = 1]|[s_1 = 1]) \\ &= P([s_1 = 1]) \cdot P([s_2 = 1]) \\ &= p_1 \cdot p_2 \end{aligned}$$

The stochastic signal data representation allows to perform probability product with a simple logic circuit.

4.1.2. Representation of discrete random variables: the stochastic bus

A discrete random variable V with cardinality M can be represented by a set of M stochastic bitstreams $b_1, \ldots, b_M$, which we will name a *stochastic bus of width* M. The j-th bitstream b_j encodes a probability $p_j = C \cdot P([V = V_j])$. C is a bus normalization constant chosen to facilitate data encoding and processing : since $\sum_j P([V = V_j]) = 1$, we have $\sum_j p_j = C$. A useful choice is $C_{max} = \frac{1}{\max_j P([V=V_j])}$, which allows to represent the most probable value V_j^{max} by $p_j^{max} = 1$ and maximizes the p-values of other signals on the bus.

Stochastic buses can be instanciated by a set of M random number generators outputting the individual bitstreams. Similarly, a set of M counters can be used to recover the unnormalized probability distribution $C \cdot P(V)$.

That data representation implies that the average number of bits before observing a "1" on the j-th signal of the bus is $T_{avg} = \frac{1}{C \cdot P([V=V_j])}$. That number, which directly determines the number of bits necessary to get an accurate reconstruction of the distribution using counters, depends on the shape of the distribution and on the value of C, which is modified by the computations done on the bus and can often not be easily controlled or computed. This creates two problems. First, the number of bits necessary to reconstruct the distribution with a given desired precision can't be easily anticipated. Second, in some cases – especially if C is low – that number may be very high, which leads to poor performance of the stochastic machine (which we call the *time dilution problem*).

The first problem can be adressed by integrating the data until a given number of "1" bits have been observed on a signal, instead of during a fixed number of bits. This can easily be achieved using counters overflow. If a stochastic bitstream of width M is connected to counters with a maximum value n_{max}, we can run the signals until one of the M counters (with index (j_{max}) overflows. If the computation is stopped at that moment, the counter with index j_{max} stores the value n_{max} corresponding to the p-value $p_j^{max} = 1$, and the other counters store values n_j corresponding to p-values $p_j = \frac{n_j}{n_{max}}$.

That process allows to renormalize the distribution with regard to the maximum probability value p_j^{max}, and to read it as a set of fixed-point numbers with precision depending on n_{max}. Furthermore, the index of the overflowing counter immediately gives the index of the most probable value, which implements the maximum a-posteriori estimator.

4.1.3. Bayesian inference with stochastic bitstreams: the bayesian machine

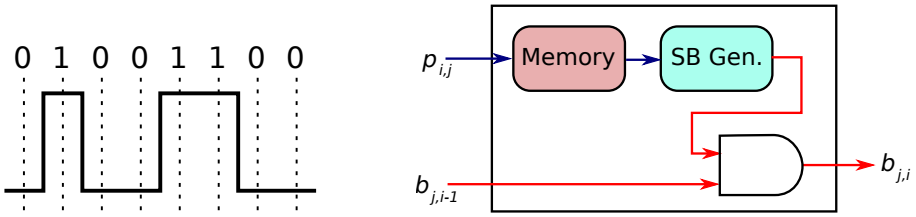
One of the most common bayesian computing techniques is naive bayesian fusion (Bessière et al., 2013) : computing the posterior probability distribution on a searched variable S, knowing a prior distribution $P(S)$ and the conditional distributions $P(K_i|S)$ on some known variables $K_1, \ldots, K_N$. If the K_i variables are conditionally independant given S, the inference is computed by :

$$P(S|K_1, \ldots, K_N) = \frac{1}{Z} P(S) \prod_{i=1}^{N} P(K_i|S) \tag{5}$$

where Z is a normalization constant.

This distribution can be computed using stochastic bitstreams by representing both the prior $P(S)$ and the data terms $P(K_i|S)$ with stochastic buses of width M, corresponding to the cardinality of S. After the bitstreams $b_{j,0}$ ($j \in \{1, \ldots, M\}$) encoding the prior values $P(S = S_j)$ (with a bus normalization constant C_0) are generated, the data terms can be integrated using simple computational modules comprised of a memory, a random generator and a logic AND gate as described in fig. 2b. For each line $j \in \{1, \ldots, M\}$ in the stochastic bus and for each data term $i \in \{1, \ldots, N\}$, the memory stores the value $p_{i,j} = C_i \cdot P(K_i|S = S_j)$ (where C_i is the bus normalization constant associated with data term i), the random generator generates a stochastic bitstream encoding probability $p_{i,j}$, and the AND gate perform the probability product between that signal and the signal $b_{j,i-1}$ from the previous element, outputting signal $b_{j,i}$.

The resulting architecture performs bayesian inference using a matrix of stochastic operators, with a number of rows equal to the cardinality M of variable S and a number of columns equal to the number of data terms N (see fig. 3). The output stochastic bus, comprised of the signals $b_{j,N}$ for $j \in \{1, \ldots, M\}$, encodes the posterior probability distribution $P(S|K_1, \ldots, K_n)$, with a bus normalization constant $C_{out} = \prod_{i=0}^{N} C_i$.



(a) Stochastic bitstream encoding a probability value $p = \frac{3}{8}$

(b) Computational module implementing probability product

Figure 2: Computing probability products with stochastic bitstreams. Fig. 2a shows 8 bits of a stochastic bitstream, with 3 bits set to 1, therefore encoding a p-value $p = \frac{3}{8}$. Fig. 2b shows a computational module performing probability product as part of a bayesian naive fusion operation: if the input signal has a p-value $PV(b_{j,i-1})$, the output signal $b_{j,i}$ encodes a p-value $PV(b_{j,i}) = PV(b_{i-1,j}) \cdot C_i \cdot P(K_i|[S = S_j)$.

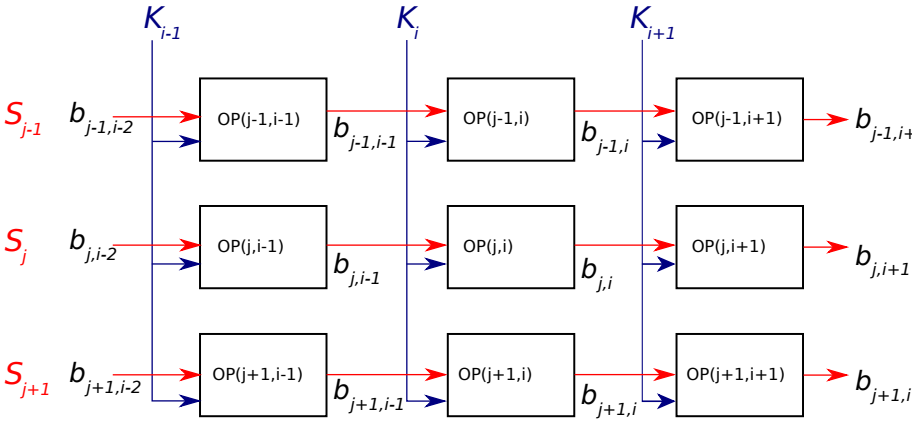


Figure 3: Architecture computing naive bayesian fusion with stochastic signals. Each OP(i,j) element is an instance of the module described in fig. 2b. The leftmost input signals $b_{j,0}$ constitute a stochastic bus encoding the prior distribution $P(S)$, and the rightmost output signals $b_{j,N}$ constitute a stochastic bus encoding the posterior distribution $P(S|K_1, \ldots, K_N)$.

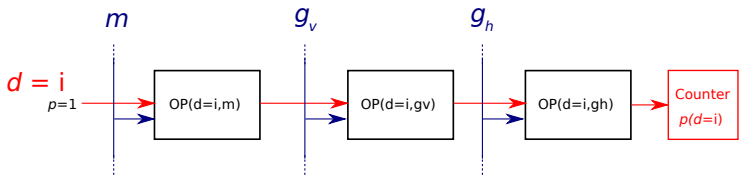


Figure 4: Sequence of computational elements computing the stochastic signal corresponding to disparity value i. Each box contains an instance of the module described in fig. 2b. A set of $D_{max}+1$ such structures allows us to compute the disparity distribution (see fig. 6)

4.2. Stochastic disparity computation

4.2.1. General description

The architecture described in section 4.1 can be used to implement the disparity computation model described in section 3. The search variable is the disparity D, which takes values in $[\![0; D_{max}]\!]$ and therefore has cardinality $D_{max}+1$, and the data terms are the three likelihood values computed from the features through equation 3.

We therefore use such a matrix of stochastic operators with $N = 3$ and $M = D_{max} + 1$ to compute a stochastic bus representation of the posterior disparity representation. In the following, we will use a uniform disparity prior ($P([D = d]) = \frac{1}{D_{max}+1} \forall i \in [\![0; D_{max}]\!]$), which can efficiently be represented by a stochastic bus with all signals constantly equal to 1 ($C_0 = D_{max} + 1$). Each of the data terms are integrated as described above in section 4.1, and the full disparity distribution for a pixel can be estimated using counters (see fig. 4). If the posterior disparity distribution is unimodal and clearly indicates a disparity value, that value can be estimated by the maximum a-posteriori estimator by simply getting the index of the first overflowing counter, as suggested in section 4.1.2.

4.2.2. Addressing issues: occlusions and low-contrast areas

Although the previous architecture allows for efficient computation when the output distribution is unimodal and indicates a clear disparity value or a small range of values, we must adapt it to take into account some issues that arise from the fact that disparity values cannot always be computed. We will describe those problems and their consequence on the architecture, and then put forward a solution.

In some cases such as occlusion (see fig. 5), some pixels in the left image have no matching pixel in the right one and the matching costs will therefore be high for every possible disparity value. In our bayesian model, it means the values computed by equation 3 will be small for all $d \in [\![0; D_{max}]\!]$, which in our stochastic architecture translates to very low p-values for all output signals. For example, in the limit case of a pixel (x, y) where the matching cost $C_f(x, y, d)$ is infinite for all disparity values $d \in [\![0; D_{max}]\!]$ and for each feature $f \in \{m, g_H, g_V\}$ in equation 3, we have $p(f^r(x - d, y)|f^l(x, y), [D(x, y) = d]) = p_0$, $\forall d \in [\![0; D_{max}]\!]$,

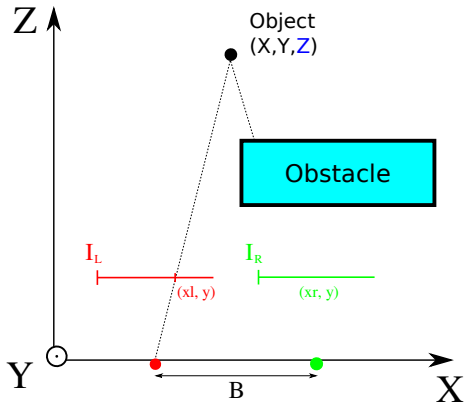


Figure 5: Occlusion example: If an obstacle is positioned such as it hides a distant object from one of the two cameras, the corresponding pixels can't be matched and disparity cannot be computed.

$\forall f \in \{m, g_H, g_V\}$. In our stochastic computation, all the signals in the output stochastic bus is have a p-value of p_0^3. This corresponds to a uniform distribution – which is correct since no information could be inferred about the disparity value from the data – but that distribution is encoded with a very low bus normalization constant $C = (D_{max} + 1) \cdot p_0^3$, which is problematic because of the time dilution problem mentioned in section 4.1.2. For $p_0 = 0.02$, for example, it means that an average of one every 125000 bits will be set to 1, and the machine has to be run for an average of one million cycles simply to fill a 8-bits counter, which is very inefficient.

In some other cases, such as large uniform areas with no texture or distinctive features, the opposite problem arises: many (or possibly all) disparity values are possible match and therefore have low matching costs. The likelihood values $p(f^r(x-d,y)|f^l(x,y),[D(x,y)=d])$ then have values close to 1 for all disparity values $d \in [\![0; D_{max}]\!]$, and all the the signals in the output stochastic bus will have a p-value close to 1, which encodes a high-entropy, close to uniform distribution with a high bus normalization constant. Again, this is a correct result and the high bus normalization constant means the time dilution problem does not arises; that output can efficiently be converted to numerical values or used in further stochastic computations. However, such high-entropy distributions are ill-suited to the use of the maximum a-posteriori estimator, which will return a random result among the possible disparity values.

A way to solve both those problems is to explicitly model the case where a pixel can't satisfyingly be matched, either because of occlusions or because of a lack of contrast, with an extra signal on the stochastic bus encoding a probability $p_{nomatch}$:

$$P(nomatch(x,y)) = p_{nm0} + (1 - p_{nm0})e^{-\frac{(g_V^l(x,y))^2}{2\sigma_{nm}^2}} \tag{6}$$

The first term in the equation is a probability $p_{nm0} \gg p_0^3$ that determines the time until which an occluded pixel is detected as not matching. It should be low enough that if the pixel can be correctly matched, the stochastic signal of the corresponding disparity value j has a p-value $p_j > p_{nm}$, but high enough that if, as described above, no match is possible because of an occlusion, the "no match" signal fills its counter and stops the computation in a reasonable time, while detecting an absence of match.

The second term of equation 6 handles the poorly contrasted areas, which have been found to be characterized by low values of the vertical gradient[3] $g_V^l(x, y)$. Weakly contrasted areas will therefore have a $P(nomatch(x, y))$ value very close to 1, and the corresponding stochastic signal will very quickly fill the counter and detect an absence of match before a spurious match attributed to the behavior of the MAP estimator can be detected.

The final architecture for our disparity computation stochastic machine is shown in fig. 6. With the extra "no match" signal, it has a dimension $N = 3$ and $M = D_{max} + 2$.

5. Stochastic model evaluation

5.1. Model implementation

In order to evaluate the benefits of using a stochastic disparity computation system, we will compare two implemetations of the bayesian disparity algorithm described in section 3.2

- A reference implementation performing the computation as floating point operations.
- A simulated stochastic implementation, using pseudo-random number generators (PRNG) and bitwise boolean logic operations to simulate the behaviour of the bayesian machine described in section 4.2.

Both implementations are programs written in C++ and run on a desktop computer equipped with an Intel Xeon E3-1271 v3 64-bit CPU. The reference implementation use FPU computations using 64 bit floating point numbers. The simulated stochastic implementation uses the Mersenne twister 19937 PRNG provided by the GNU implementation of C++11 to generate stochastic bitstreams, and the 64-bit bitwise boolean AND operation to perform probability product.

5.2. Results

The reference implementation ran in about 1.25 seconds per frame, which is the order of magnitude of the "fast" disparity algorithms from the state of

[3]Note that the square of the gradient value of the left image itself is used, and not a matching cost associated to the gradient as in equation 3. $P(nomatch(x, y))$ is therefore high is the gradient is close to zero, that is in weakly contrasted areas.

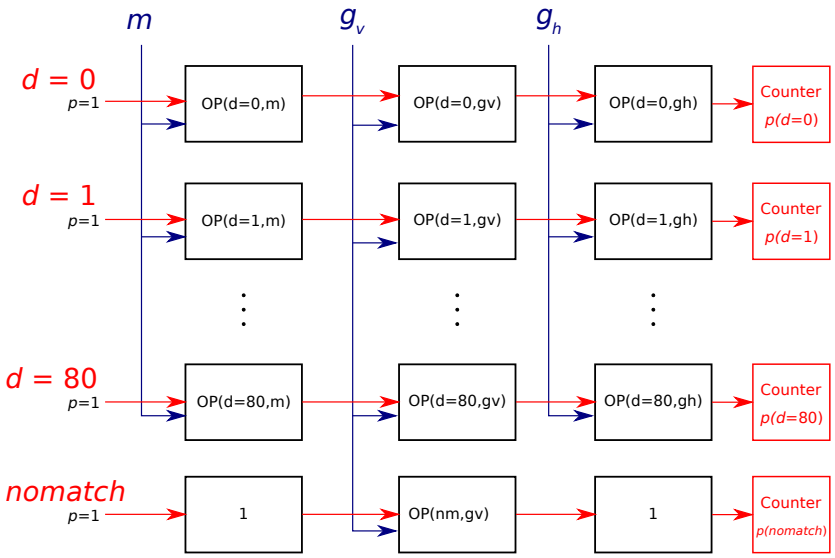


Figure 6: Bayesian stochastic machine implementing the disparity computation. Each "OP" box contains an instance of the module described in fig. 2b. Always-on signals corresponding to the uniform prior are input in the left, the feature matching likelihood values are integrated by the computational modules, and the disparity distribution is output as a stochastic bus on the right and converted to a fixed-point numeric representation by the counters. If the disparity can be computed, the counter linked to the corresponding signal will overflow first, otherwise the "no match" channel will overflow.

the art. The simulated stchastic implementation ran in 25 to 110 seconds per frame (depending on the frame and on the size of the output counters). That low performance is due to the overhead of simulating stochastic machines using non-stochastic hardware; the performance of the stochastic system is better estimated by the number of simulated clock cycles used to compute a frame (see below).

5.2.1. Dataset and model parameters

We collected stereo image pairs using a PointGrey BumbleBee2 BB2-03S2C-25 wide-angle color stereoscopic camera, with focal length $f = 2.5mm$, baseline distance $B = 120mm$ and resolution 640×480 at 25 frames per second. The images were rectified using the Triclops proprietary PointGrey middleware. The camera was mounted on a TurtleBot 2 mobile robot base, which was manually controlled in an office environment to collect data. A total of 6301 frames were captured, corresponding to 4 minutes and 10 seconds of video.

The 24 bits color images captured were converted to 8 bits luminance images, with pixel values in $[\![0; 255]\!]$. The preprocessing described in section 3.2 therefore generate feature maps with pixel values in $[\![0; 255]\!]$ for the averaging filter and $[\![-127; 127]\!]$ for the gradients. The D_{max} value was set to 80, which corresponds to a minimum distance of 42 centimeters. A simple grid search performed during preliminary experiments allowed us to select good values of the other parameters, summarized in table 1.

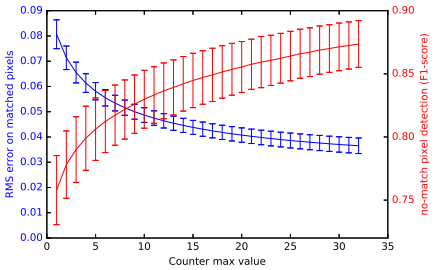
Parameter	Value
D_{max}	80
p_0	0.02
σ_m	10
σ_{g_V}	10
σ_{g_H}	10
p_{nm0}	0.01
σ_{nm}	8

Table 1: Model parameter values used for the evaluation

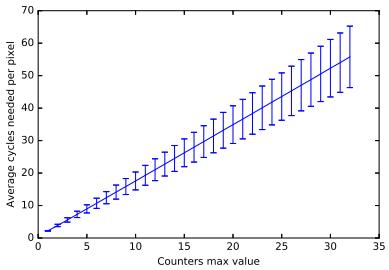
The feature maps were used to compute the likelihoods as described in equation 3, and those likelihoods were used both in the reference implementation and in the simulated stochastic implementation to compute the disparity distribution.

5.2.2. Disparity computation accuracy

A feature of stochastic computing using stochastic bitstreams is progressive precision: the longer the information from a bitstream is integrated, the more precisely the corresponding p-value can be estimated. In the context of the stochastic bus framework described in section 4.1.2, it means that precision increases with the size of the counters used to estimate the distribution: the higher



(a) Global RMS error (for matched pixels) and F1-score (for non-match detection)



(b) Average run time per pixel

Figure 7: Performance of the simulated stochastic disparity processor. Fig. 7a shows that the detection of "no-match" pixels and the quality reconstructed distribution exponentially improves with counter max value. Fig. 7b depicts the linear relationship between counter max value and simulated run time (in number of simulated clock cycles to compute the disparity distribution for one pixel). Data shown are the mean and standard deviation on a subset of 126 images obtained by regular sampling of the full dataset.

the counters' maximum value, the closer to the reference implementation the resulting distribution is expected to be. We therefore used the simulated stochastic implementation with variable counter sizes to quantify that phenomenon.

The stochastic disparity processor described in section 4.2 performs two functions: detecting the "no match" pixels, and computing the disparity distribution on matched pixels. The performance of the "no-match" pixels discrimination task can be assessed by computing the F1-score between the set of pixels identified as "no-match " by the reference implementation and the simulated stochastic implementation. The performance of the disparity distribution computation task can be assessed by measuring the RMS error between the distributions estimated from the simulated stochastic implementation and the reference implementation[4].

Figure 7a shows that indeed, the F1-score exponentially increases and the RMS error exponentially decreases with the counter max value. For example, with 16-bits counters the RMS error is below 0.05 and the F1-score above 80%.

Figure 7b shows the relationship between the counter max value and the average time (in number of clock cycles) the simulated stochastic machine has to run before filling a counter. As expected, it grows linearly with maximum value of the counter: as all signals are uncorrelated, each extra "1" required to fill the counter generates a constant overhead.

[4]Using the KL-divergence has also been considered, but proved problematic because of the frequent occurence of 0 as a p-value in the distributions estimated from the simulated stochastic implementation.

Fig. 8 shows an example of reconstructed disparity image with the reference implementation, and with the simulated stochastic implementation using two maximum counter values, 1 and 16. The disparity image from the stochastic system with the larger counters is visually close to the reference. The image obtained using 1-bit counters, while clearly noisier and lower quality, still correctly describes the general tridimensional structure of the scene and could possibly be used to drive a robust robot control system. According to the data from fig. 7b, the stochastic computation of the disparity distribution requires 2.21 ± 0.09 clock cycles per pixel for 1-bit counters and 27.97 ± 4.58 clock cycles per pixel for counters with a maximum value of 16. Both those values compare favorably to the floating point computations performing to the same operations, which requires at least 81×3 floating point number products and a maximum search on a 81-value vector.

6. Discussion: speed, energy and hardware implementation considerations

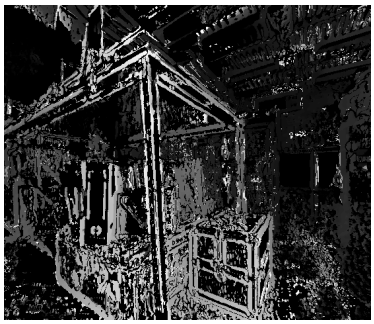
The above results show that our stochastic computational system can successfully implement a bayesian binocular disparity algorithm and compute full disparity distribution with good accuracy, using stochastic bitstreams and a reduced number of computation cycles.

However, the stochastic bitstream-based computational system described in section 4.1 supposes the use of fast, efficient sources of stochastic signals, that could be integrated at a large scale in a hardware component, jointly with AND gates, memories and counters, to implement the architecture seen in section 4.1.3. In this paper, we used a simulated implementation using Mersenne twister PRNGs to evaluate the potential of this approach in the absence of such components. But recent advances in new nanodevices based on spintronics, such as the superparamagnetic tunnel junction (SMTJ) (Locatelli et al., 2015), bear the promise that such generators could be available in the short or medium term. Experimental SMTJ devices have been shown to be able to generate high-quality stochastic bitstreams at a frequency of 500MHz with a very low power consumption of 50W. Those components can be built with CMOS technology using an area equivalent to 12 bytes of SRAM (Querlioz, 2016), making them suitable to large scale integration with the other components needed to build the stochastic machines described above.

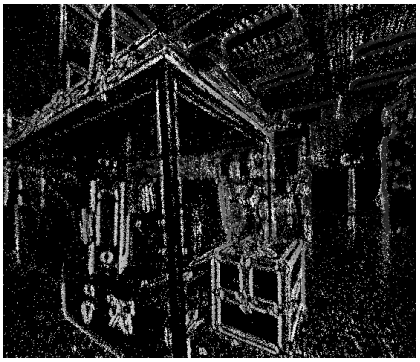
Using those figures as guidelines, we can compute the order of magnitude of the speed and power consumption of the disparity computation systen described in section 4.2 and evaluated in simulation in section 5. The system requires 246 random signal generators, which would have a total power consumption of 12.3 mW. Using counters with a maximum value of 16, which has been shown in section 5.2.2 to be an adequate tradeoff between speed and accuracy, we need an average of 27.97 clock cycles per pixel, with $(640 - 4 - 80) \times (480 - 4)$[5],

[5]Each dimension of the original 640×480 images is reduced by 4 pixels by the prefiltering

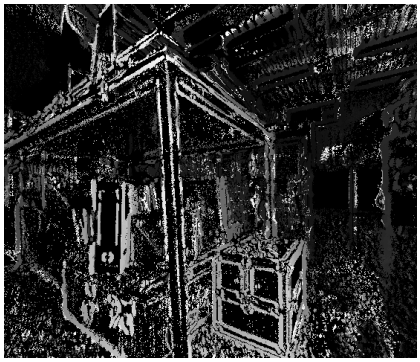
(a) Left image



(b) Reference disparity image (64 bits floating point computation)



(c) Stochastic computation, counter max value 1



(d) Stochastic computation, counter max value 16

Figure 8: Disparity images obtained by applying the MAP estimator to the output distribution for an exemple frame. Fig. 8a is the rectified frame from the left camera. Fig. 8b is the reference image obtained by classic 64 bits floating point computation. Fig. 8c and 8d show the images obtained by the simulated stochastic implementation, respectively with max counter values 1 and 16. Black pixels are "no-match" pixels; white pixels have disparity to $d = D_{max} = 80$, and gray pixels have a luminance proportional to their disparity value in $[\![0; D_{max}]\!]$

which represents an average total of 7402428.32 clock cycles per image. At a frequency of 500MHz, the architecture would therefore be able to process about 67.5 image pairs per second.

Those computations are only rough estimations; more specifically the energy consumption computation ignores the energy cost of the AND gates, memories and counters also necessary to implement the circuit, and the performance does not take into account the overhead induced by reinitializing the machine between each pixel (resetting the counters and loading the data memories). Our bayesian algorithm also makes use of preprocessed images using spatial filters; the cost (both computational and energetic) of that preprocessing should be taken into account into any global evaluation of the system. But many methods exist to perform such spatial filtering (using general-purpose CPUs, GPUs, FPGAs, dedicated hardware, etc.) with various cost, performance and energy-efficiency characteristics; further work will explore ways through which such filtering could be done using stochastic computations. Similarly, the cost computation step is currently performed using classic floating-point computation, the opportunity to use stochastic computations instead is currently being studied.

On the other hand, those computations are assumed to be performed sequentially for each pixel on a unique instance of the systems described in section 4.2, using only 246 of the computing modules described in fig. 2b. But our bayesian machine architecture is parallel by design, and a higher number of those modules would allow for parallel processing of many pixels and increased performance, at the cost of higher circuit size and energy consumption.

7. Conclusion

We have put forward an architecture to compute a class of bayesian inference problems with probabilistic hardware using stochastic bitstreams, and evaluated that system in simulation on the example of binocular disparity computation, demonstrating high performance and energy-efficiency. Although the work described in this paper uses simulations of hypothetical stochastic machines using experimental hardware devices and can therefore only offer rough estimations of the performance of those systems, it is our belief that those results clearly highlight the potential of bayesian computation using stochastic bitstreams for sensorimotor processing, especially in applications with tight constraints on computational and energy resources such as mobile robotics, embedded systems or distributed sensors.

The proposed architecture allows to solve many sensorimotor fusion and processing problems, yielding full distributions expressed as bus of stochastic bitstreams, with a low power consumption and reduced computational resources. The parallel and distributed nature of our architecture could allow to easily address a variety of different problems using the same arrays of generic

as seen in section 3.2, and the horizontal dimention is further reduced by D_{max} since the distribution can't be computed for the D_{max} first pixels of each row as shown by equation 2.

components, in a way similar to FPGAs. Furthermore, the progressive precision of the stochastic bitstream data representation allows to easily adjust the speed/accuracy or power/accuracy tradeoffs by changing the signal integration time (determined by the counters maximum values), making it possible, for example, to maintain degraded operation with lower accuracy in low energy conditions.

Future work will entail continued collaboration with teams developing stochastic signal generator devices, in order to evaluate our architecture on real hardware. Efforts will also be dedicated to extending the breadth of the computations implemented by stochastic bitstream based systems, combining the disparity computation to other sensory computations (such as optical flow) to create an occupancy map, which could then be used for obstacle avoidance and robot navigation, paving the way to a completely stochastic robot sensorimotor controller.

Acknowledgements

This work was performed within the EU Future and Emerging Technologies BAMBI project (FP7-ICT-2013- C, project number 618024). It was also partly supported by ANR Labex SMART (ANR-11-LABX-65).

References

Alves, J. D., Ferreira, J. F., Lobo, J., & Dias, J. (2015). Brief Survey on Computational Solutions for Bayesian Inference. In *Workshop on Unconventional computing for Bayesian inference at IROS2015*. Hamburg.

Belhumeur, P. N. (1996). A Bayesian Approach to Binocular Stereopsis. *International Journal of Computer Vision*, *19*, 237–260.

Bessière, P., Ahuactzin, J.-M., Mekhnacha, K., & Mazer, E. (2013). *Bayesian Programming*. Chapman and Hall/CRC. URL: `ftp://ftp.inrialpes.fr/pub/emotion/bayesian.../Bayesian-Programming.pdf`.

Bessière, P., Laugier, C., & Siegwart, R. (2008). *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems* volume 46 of *Springer Tracts in Advanced Robotics*. Berlin, Heidelberg: Springer Berlin Heidelberg. URL: `http://link.springer.com/10.1007/978-3-540-79007-5`. doi:`10.1007/978-3-540-79007-5`. `arXiv:arXiv:1011.1669v3`.

Coue, C., Pradalier, C., Laugier, C., Fraichard, T., & Bessiere, P. (2006). Bayesian Occupancy Filtering for Multitarget Tracking: An Automotive Application. *The International Journal of Robotics Research*, *25*, 19–30. doi:`10.1177/0278364906061158`.

Faix, M., Lobo, J., Laurent, R., Vaufreydaz, D., & Mazer, E. (2015). Stochastic Bayesian Computation for Autonomous Robot Sensorimotor Systems. In *Proceedings of the IROS2015 workshop on Unconventional computing for Bayesian inference* (pp. 27–32).

Gaines, B. (1969). Stochastic computing systems. *Advances in information systems science*, (pp. 37–172). URL: http://link.springer.com/chapter/10.1007/978-1-4899-5841-9{_}2. doi:10.1145/1465482.1465505.

Geiger, a., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, *32*, 1231–1237. URL: http://ijr.sagepub.com/cgi/doi/10.1177/0278364913491297. doi:10.1177/0278364913491297.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for Autonomous Driving? The \textsc{KITTI} Vision Benchmark Suite. *Computer Vision and Pattern Recognition)*, .

Jonas, E., Tenenbaum, J. B., & Wilson, M. a. (2014). *Stochastic Architectures for Probabilistic Computation by*. Ph.D. thesis Massachssets Institute of Technology.

Jones, D. G., & Malik, J. (1992). A Computational framework for determining stereo correspondence from a set of linear spatial filters. *Image and Vision Computing*, *10*, 699—-708. doi:10.1613/jair.301. arXiv:9605103.

Lazaros, N., Sirakoulis, G. C., & Gasteratos, A. (2008). Review of Stereo Vision Algorithms: From Software to Hardware. *International Journal of Optomechatronics*, *2*, 435–462. doi:10.1080/15599610802438680.

Lebeltel, O. (2006). *Programmation Bayésienne des Robots*. Ph.D. thesis Université de Grenoble.

Locatelli, N., Vincent, A. F., Mizrahi, A., Friedman, J. S., Vodenicarevic, D., Kim, J.-V., Klein, J.-O., Zhao, W., Grollier, J., & Querlioz, D. (2015). Spintronic Devices as Key Elements for Energy-Efficient Neuroinspired Architectures. *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, *1*, 994–999.

Mansinghka, V. (2009). *Natively Probabilistic Computation*. Ph.D. thesis Massachusetts Institute of Technology. URL: http://dspace.mit.edu/handle/1721.1/47892.

Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2015). *A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation*. Technical Report arXiv preprint 1512.02134. URL: http://arxiv.org/abs/1512.02134. arXiv:1512.02134.

Querlioz, D. (2016). Review of IEF's work - Modelling of superparamagnetic MTJs. In *BAMBI-FET second year annual meeting*. Paris.

Scharstein, D. (1994). Matching images by comparing their gradient fields. *Proceedings of 12th International Conference on Pattern Recognition*, *1*, 4–7. doi:10.1109/ICPR.1994.576363.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., & Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8753*, 31–42. doi:10.1007/978-3-319-11752-2_3.

Scharstein, D., & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, *47*, 7–42. doi:10.1023/A:1014573219977.

Su, C. C., Bovik, A. C., & Cormack, L. K. (2012). Statistical model of color and disparity with application to Bayesian stereopsis. *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, (pp. 169–172). doi:10.1109/SSIAI.2012.6202480.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Vigoda, B. (2003). *Analog Logic : Continuous-Time Analog Circuits for Statistical Signal Processing*. Ph.D. thesis Massachusetts Institute of Technology. URL: http://pubs.media.mit.edu/pubs/papers/03.07.vigoda.pdf.

Von Neumann, J. (1956). Probabilistic logics and the synthesis of reliable organisms from unreliable components. URL: http://books.google.com/books?hl=en{&}lr={&}id=QaruU73YWGkC{&}oi=fnd{&}pg=PA110{&}dq=PROBABILISTIC+LOGICS+AND+THE+SYNTHESIS+OP+RELIABLE.+ORGANISMS+PROM+UNRELIABLE+COMPONENTS{&}ots=AdOY3cy2Nu{&}sig=i8ODJDGUrK51AETEzojzVtx5LwM. doi:10.1128/AEM.00314-09.

Žbontar, J., & LeCun, Y. (2014). Computing the Stereo Matching Cost with a Convolutional Neural Network. *arXiv preprint arXiv:1409.4326*, . doi:10.1109/CVPR.2015.7298767. arXiv:1409.4326.