An Optical Processor for Matrix-by-Vector Multiplication: an Application to the Distance Geometry Problem in 1D

Simon Hengeveld, N Rubiano da Silva, D Gonçalves, P Souto Ribeiro, Antonio Mucherino

An Optical Processor for Matrix-by-Vector Multiplication: an Application to the Distance Geometry Problem in 1D

S.B. Hengeveld,[1,*] N. Rubiano da Silva,[2,†] D. S. Gonçalves,[3,‡] P. H. Souto Ribeiro,[2,§] and A. Mucherino[1,¶]

[1] *IRISA, University of Rennes 1, Rennes, France*
[2] *Departamento de Física, Universidade Federal de Santa Catarina, CEP 88040-900, Florianóplis, SC, Brazil*
[3] *Department of Mathematics, CFM, Federal University of Santa Catarina, Florianópolis, SC, Brazil*

We present the architecture of a new optical processor specialized in matrix-by-vector multiplication via the manipulation of the light wavefront. This processor can reach up to 1.2 Giga MAC (multiply-accumulate) operations per second using commercially available devices. Moreover, this architecture is compatible with a hardware upgrade with the potential to achieve a processing speed of above Tera MAC per second. We initially present the optical processor and then discuss the use of such a processor for tackling a special class of the one-dimensional Distance Geometry Problem (DGP), which is a well-known NP-hard problem.

I. INTRODUCTION

Optical computing schemes have been recently appearing in the scientific literature for attempting the solution of NP-hard (nondeterministic polynomial time) problems [1] in a processing time consistently shorter than the one attainable by standard computers. Two recent examples are given by the Subset Sum Problem (SSP) [2, 3] and the Traveling Salesman Problem (TSP) [4, 5], for which specific optical computing schemes were proposed. The SSP and the TSP are NP-hard problems, because all existing algorithms for solving them exhibit a computational complexity that scales exponentially with the problem size. We are particularly interested in the Distance Geometry Problem (DGP), which is also an NP-hard problem and is *quite close* to the SSP in the sense that it is possible to reduce, via a polynomial transformation, any instance of the SSP to an instance of the one-dimensional DGP [6]. Our initial motivation comes from the approach to the TSP presented in [5], which is based on using the light wavefront to perform the required mathematical operations of matrix-by-vector multiplications. Moreover, another important and recent result in optical computation, which does not deal directly with NP-hard problems, exhibits an impressive performance by realizing multiply-accumulate (MAC) operations at the rate of 2 THz [7], using frequency combs and programmable transmission devices. When standard transistor-based computers cannot help in solving the hard problems mentioned above, and the technology of quantum computers is not yet ready to tackle them, we believe optical computations can be considered as a valid and particularly efficient alternative.

Here, we present and describe the architecture of an optical processor that is also capable of realizing MAC operations at a speed that, in the best conditions, is comparable to the speed of light. It uses the light wavefront and spatial light modulators (SLM) to implement MAC operations. Our optical processor is specialized on matrix-by-vector multiplication operations and it is in principle, capable of attaining performances in the range of GHz (Giga MAC operations per second). Moreover, the architecture of our optical processor has potential for faster performance, opening the doors for a wide use of this new device. If it attains a high processing speed in terms of MAC operations, then it can be employed to solve problems of bigger size.

In particular, we shall show that a specific class of the DGP can be *exactly* reformulated as a unique matrix-by-vector operation (although the matrix has an exponentially growing number of rows). This indicates that our optical processor can be employed either for solving subtasks involving the solution of more complex problems, or directly for an efficient solution of real-life problems, once a suitable reformulation is supplied.

The matrix-by-vector multiplication operation admits one well-known closed-form formula, and the operation produces a new vector we refer to as the *result* of the multiplication. In spite of this simplicity, however, scientific works have been proposing over the years more and more efficient algorithms that can run on standard computer machines (see for example the works on the software package LAPACK [8], and the more recent proposals to perform the operation with highly sparse matrices [9]). In order to underline the general interest in this kind of linear algebra operations, we finally cite the so-called Power Iteration method used in Google's PageRank algorithm, which is based on the multiplication of the so-called "Google matrix" by a given vector. The advent of modern technologies has more recently motivated the research community to engage not only in the direction of optimizing this matrix operation on classical computers but also in other emerging technologies, such as GPUs (the work in [10] proposes for example an algorithm to perform the matrix-by-vector multiplication on GPU devices).

This paper is organized in two parts. The first (Sec-

* simon.hengeveld@irisa.fr
† nara.rubiano@ufsc.br
‡ douglas.goncalves@ufsc.br
§ p.h.s.ribeiro@ufsc.br
¶ antonio.mucherino@irisa.fr

tion II) is devoted to the description of our new optical processor. The presentation begins with a general and rather informal description, which is followed by a more accurate description of the phenomena behind the computations that our processor performs by using a light beam. We then discuss the computing performance of the processor, in comparison with classical computing. We conclude this part of the paper with some technical details for a practical realization of the optical processor at the hardware level.

The second part (Section III) focuses on a case study, a special class of the DGP in dimension 1 that can be reformulated as a matrix-by-vector multiplication problem. As mentioned above, this is a rather interesting case study for us, because our optical processor can be used for completely solving the problem, and not only for performing a sub-task in a more complex algorithm. Naturally, the reformulation of the DGP as a matrix-by-vector operation does not change the problem complexity (which remains NP-hard), but we will show that our optical processor is potentially able to solve instances for large dimensions in reasonable time scales.

Finally, Section IV concludes the paper with some directions for future works.

II. A MATRIX-BY-VECTOR OPTICAL PROCESSOR

We propose in this work the architecture of a new processor capable of performing the matrix-by-vector multiplication by exploiting the properties of a light beam. As already mentioned in the Introduction, in spite of its simplicity, this basic linear algebra operation is often found at the core of more complex procedures for solving several practical problems. We will focus in Section III on an NP-hard problem that can be reformulated as one unique matrix-by-vector multiplication.

The problem we solve with our optical processor asks to calculate the n-dimensional vector $\mathbf{r}$ as the product between a $n \times m$ matrix $\mathbf{M}$ and a m-dimensional vector $\mathbf{y}$. We make use of light polarization and transverse spatial distribution of phases in a light beam to realize the multiplication of the several matrix and vector elements, by following the well-known formula:

$$\forall i \in \{1, 2, \ldots, n\}, \qquad r_i = \sum_{j=1}^{m} M_{ij} y_j,$$

where r_i is the i^{th} element of the vector $\mathbf{r}$, y_j is the j^{th} element of $\mathbf{y}$, and M_{ij} is the matrix element of position (i, j) in $\mathbf{M}$.

We will give a general description of the optical processor in the next section; we will then discuss the technical details about its functioning in Section II B. Section II C will subsequently discuss how our optical processor can outperform classical transistor-based computing processors. Finally, Section II E will give some technical details about the realization of the proposed processor at hardware level.

A. Basic description of optical processor

The very first task that is performed by our optical processor is to prepare the vector $\mathbf{y}$ for performing the multiplication operation (see Figure 1, top row of optical diagram). A light beam polarized in the diagonal direction is sent to a Spatial Light Modulator (SLM) that encodes the vector elements y_j in a stripe-like phase mask. The numerical values in the vector are mapped onto gray levels of the SLM pixels, which in turn imprint phase shifts to the incoming light according to these levels. Notice that the SLM acts only in the horizontal polarization component, so that the phase applied by the SLM becomes a phase difference between vertical and horizontal polarization components. As a result, the polarization state is locally modified within the wavefront. The outgoing light beam passes through a Half-Wave Plate (HWP) and is filtered by a Polarizing Beam Splitter (PBS), which selects only the diagonal polarization component. This operation converts the phase modulation into an amplitude modulation. Therefore, the vector elements are encoded in the amplitude transverse spatial distribution of the beam in the image plane of a lens.

The combination of phase-only SLMs with before- and after-polarization projections is a possible method to convert the SLM-imprinted phase pattern into the amplitude distribution of the light beam [11]. One advantage of using the amplitude-related encoding instead of phase is that amplitude information can be straightforwardly measured through the intensity distribution and recorded as an image using, e.g., a CCD (charge-coupled device) camera.

The processing of a matrix-by-vector multiplication by our optical processor is entirely displayed in Fig. 1 (optical diagram). The input vector encoded in the light beam prepared by the first SLM is imaged onto the surface of a second SLM. Its polarization state was previously rotated to diagonal and the same strategy used in the first step is repeated. The second SLM is programmed with gray levels that encode the matrix elements. The exit beam has a spatial modulation in polarization, corresponding to the multiplication of each vector element y_j by each matrix row element M_{ij}. To complete the matrix-by-vector operation, the elements $M_{ij}y_j$ of each row finally need to be accumulated. This is implemented by a cylindrical lens focusing along the appropriated row direction. The polarization is filtered again keeping the horizontal component, and we obtain the resulting vector elements r_i as an intensity map, which is subsequently recorded using a CCD camera or photodiodes.

The output signal is thus composed by n "pixels", corresponding to each of the n elements of the vector $\mathbf{r}$. The numerical values for every element r_i can be retrieved by inverting the gray level mapping used at the very begin-

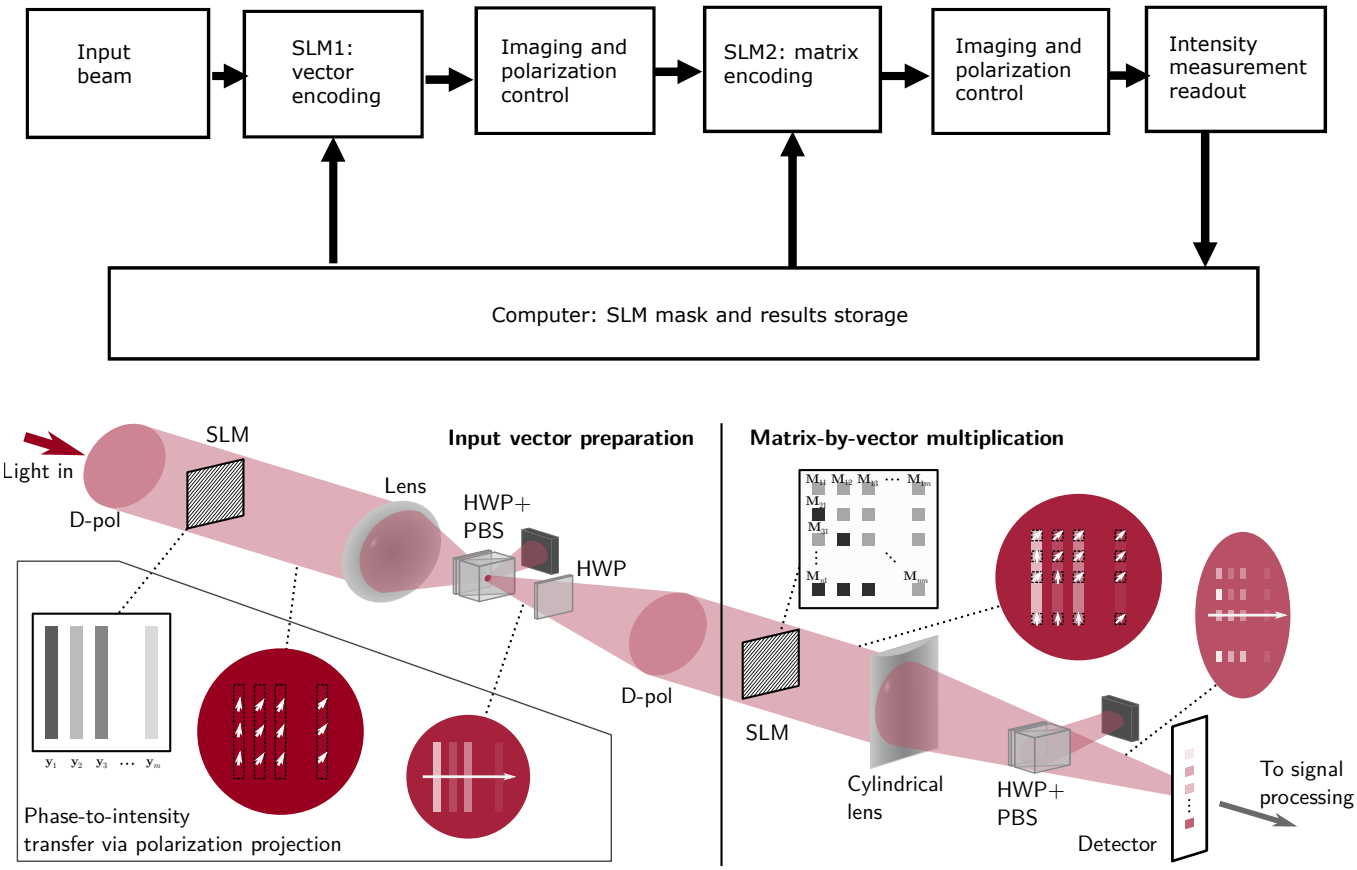


FIG. 1. An optical processor to perform multiplication-accumulation (MAC) operations. Block diagram: upper figure. Optical diagram: lower figure. An input vector is prepared using a first SLM; the vector is then multiplied by a matrix using a second SLM (which encodes the matrix elements) and a cylindrical lens. The resulting vector is detected using a CCD camera, or photodiodes. The insets show the gray level distribution on the SLMs, and the beam transverse profile at different stages (arrows indicate polarization, color level indicate intensity).

ning of the process while preparing the vector $\mathbf{y}$.

B. Description of the optical processor in terms of the optical electric field

We will now explicitly show how the intensity measurements in our optical processor are related to the target matrix-by-vector multiplication. For that, we first demonstrate the phase-to-intensity transfer via polarization projection, taking into account the effects of a SLM, a HWP and a PBS on an input field (see lower inset in Fig. 1).

Before processing with the first SLM, let us suppose that the input light field can be represented as an ideal plane wave propagating along the z-direction. Its electric field complex amplitude is given by:

$$\vec{E}^{in} = E_0 \, \vec{\epsilon} \, \mathrm{e}^{-i(\vec{k}\cdot\vec{p}-\omega t)}, \tag{1}$$

where $\vec{\epsilon}$ is the polarization vector, $\vec{k} = k\hat{z}$ is the wave vector in vacuum, ω is the optical frequency, and E_0 is a constant. Assuming a monochromatic field, the temporal phase ωt only contributes to a global phase and plays no role in our scheme; ωt can hence be omitted. We further assume that the SLM is located at the plane $\vec{p} = (x, y, 0)$ and the polarization state is linear diagonal $\vec{\epsilon} = (\hat{x} + \hat{y})/\sqrt{2}$.

The input field is modified by the first SLM, which imprints a phase modulation on the horizontal polarization component. The modulation is given by the $n \times m$ matrix:

$$S = 2 \begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nm} \end{bmatrix},$$

where the generic ϕ_{ij} is the phase added to the horizontal component of the field impinging on the corresponding cell (i, j) and where the field amplitude is $\dfrac{E_0}{nm}$. Therefore,

the field immediately after the first SLM is:

$$\vec{E}_{ij}^{SLM} = \frac{E_0}{nm\sqrt{2}} \, (\hat{x} + \mathrm{e}^{-i2\phi_{ij}} \hat{y}) \tag{2}$$

which represents a spatial modulation on the polarization. In order to transfer such a modulation to the intensity profile, an ad hoc polarization projection needs to be realized. Specifically, we employ a HWP (tilting the polarization in $-45°$, i.e., D→H and A→V) and a PBS (filtering only the H component) to project the diagonal component of the field in Eq. (2), resulting in (see [11]):

$$\vec{E}_{ij}^{out} = \frac{E_0}{2nm} \, \cos(\phi_{ij}) \hat{x}. \tag{3}$$

Equation (3) clearly shows that the gray-level mask S displayed on the SLM surface is transferred to the transverse intensity distribution of the field. In other words, the combination of phase-only SLM, HWP and PBS works as an effective amplitude modulator.

In our optical processor (see Fig. 1), the operation described above is applied two times, sequentially: first for preparing the input vector $\mathbf{y}$, and then for multiplying it by the matrix $\mathbf{M}$. Specifically, a field given by Eq. (3), with ϕ_{ij} relating to the first SLM, has its polarization tilted to the diagonal direction and then used as the input field for the second SLM, whose modulation we call α (instead of ϕ), and to which we assign the same indices i and j over the two SLM dimensions n and m (it is supposed that the two SLMs have the same resolution). The final field is therefore given by:

$$\vec{E}_{ij}^{out} = \frac{E_0}{2\sqrt{2}nm} \, \cos(\phi_{ij}) \cos(\alpha_{ij}) \hat{x}. \tag{4}$$

We note that an imaging lens is used to carefully transfer the spatial structure of the beam from the first to the second SLM, ideally without any distortion, ensuring a proper overlap between the masks (both of size $n \times m$). The total amplitude modulation can be viewed in terms of the matrix S_{amp}, comprising the element-by-element multiplication:

$$S_{amp} = \begin{bmatrix} \cos(\phi_{11})\cos(\alpha_{11}) & \cos(\phi_{12})\cos(\alpha_{12}) & \dots & \cos(\phi_{1m})\cos(\alpha_{1m}) \\ \cos(\phi_{21})\cos(\alpha_{21}) & \cos(\phi_{22})\cos(\alpha_{22}) & \dots & \cos(\phi_{2m})\cos(\alpha_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\phi_{n1})\cos(\alpha_{n1}) & \cos(\phi_{n2})\cos(\alpha_{n2}) & \dots & \cos(\phi_{nm})\cos(\alpha_{nm}) \end{bmatrix}.$$

In the last step of our optical scheme, the beam is focused in the detection plane with a cylindrical lens, so that the rows of the matrix concentrate to a single cell, effectively performing the sum over the elements of each row. The field at each cell i is given by:

$$\vec{E}_i^{DET} = \sum_j \frac{E_0}{2\sqrt{2}nm} \, \cos(\phi_{ij}) \cos(\alpha_{ij}) \hat{x}. \tag{5}$$

Apart from a constant multiplicative term, the transverse spatial distribution of the field is now represented by a column (it could be a line as well) vector V^{DET}:

$$V^{DET} = \begin{bmatrix} \cos(\phi_{11})\cos(\alpha_{11}) + \cos(\phi_{12})\cos(\alpha_{12}) + \ldots + \cos(\phi_{1m})\cos(\alpha_{1m}) \\ \cos(\phi_{21})\cos(\alpha_{21}) + \cos(\phi_{22})\cos(\alpha_{22}) + \ldots + \cos(\phi_{2m})\cos(\alpha_{2m}) \\ \vdots \\ \cos(\phi_{n1})\cos(\alpha_{n1}) + \cos(\phi_{n2})\cos(\alpha_{n2}) + \ldots + \cos(\phi_{nm})\cos(\alpha_{nm}) \end{bmatrix}.$$

The result of the operation is given by the intensities that are measured by a CCD camera:

$$\mathcal{I}_i = \left| \vec{E}_i^{DET} \right|^2 \equiv I_0 |V_i^{DET}|^2, \tag{6}$$

where $I_0 = \left| \frac{E_0}{2\sqrt{2}nm} \right|^2$. Since α_{ij} and ϕ_{ij} shall be chosen such that $\cos(\alpha_{ij}) \geq 0$ and $\cos(\phi_{ij}) \geq 0$, we have $|V_i^{DET}| = V_i^{DET}$. Hence, we can rewrite V^{DET} in terms

of the measured intensities:

$$V^{DET} = \begin{bmatrix} \sqrt{\mathcal{I}_1/I_0} \\ \sqrt{\mathcal{I}_2/I_0} \\ \vdots \\ \sqrt{\mathcal{I}_n/I_0} \end{bmatrix}.$$

Finally, the detected signal given by Eq. (6) can be identified with the multiplication $\mathbf{My}$, where $\mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} \cos(\phi_1) \\ \cos(\phi_2) \\ \vdots \\ \cos(\phi_m) \end{bmatrix}$$

and

$$\mathbf{M} = \begin{bmatrix} \cos(\alpha_{11}) & \cos(\alpha_{12}) & \dots & \cos(\alpha_{1m}) \\ \cos(\alpha_{21}) & \cos(\alpha_{22}) & \dots & \cos(\alpha_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\alpha_{n1}) & \cos(\alpha_{n2}) & \dots & \cos(\alpha_{nm}) \end{bmatrix},$$

by setting, for the first SLM, all elements of a column in S to the same modulation: $\cos(\phi_{i1}) \equiv \cos(\phi_1)$, for $i = 1, \dots, n$, $\cos(\phi_{i2}) \equiv \cos(\phi_2)$, for $i = 1, \dots, n$, and so on. This results in $V^{DET} = \mathbf{My}$:

$$V^{DET} = \begin{bmatrix} \cos(\phi_1)\cos(\alpha_{11}) + \cos(\phi_2)\cos(\alpha_{12}) + \ldots + \cos(\phi_m)\cos(\alpha_{1m}) \\ \cos(\phi_1)\cos(\alpha_{21}) + \cos(\phi_2)\cos(\alpha_{22}) + \ldots + \cos(\phi_m)\cos(\alpha_{2m}) \\ \vdots \\ \cos(\phi_1)\cos(\alpha_{n1}) + \cos(\phi_2)\cos(\alpha_{n2}) + \ldots + \cos(\phi_m)\cos(\alpha_{nm}) \end{bmatrix}.$$

C. Computing performance

In this section, we examine the performance of our optical processor by estimating the time required to perform MAC operations. The multiplication and accumulation operations of the acronym "MAC" exactly reflect the elementary operations (on scalars) forming a matrix-by-vector multiplication: specific matrix elements are multiplied by specific vector elements, and the result is obtained by the accumulation of the partial results. The number of operations required to perform the multiplication of a $n \times m$ matrix by a vector of dimension m basically corresponds to $n \times m$ multiplications and n accumulations.

Two SLMs having the same resolution are employed in our optical processor. Let $n^{SLM} \times m^{SLM}$ be the size over the two dimensions of the pixel grid forming the screen of the two SLMs. In the hypothesis the $n \times m$ matrix $\mathbf{M}$ fits in the SLMs (i.e. $n \leq n^{SLM}$ and $m \leq m^{SLM}$), the matrix-by-vector operation can be performed in a single run. We use the largest dimension of the first SLM (say m^{SLM}) to encode the vector. The processing rate is naturally tremendously high, because it is essentially given by the speed at which light propagates between the two SLMs, on the order of 1 nanosecond (10^{-9} seconds). For each single run of the optical processor, capable of attaining this high speed, the total number of performed operations corresponds to $[n^{SLM} \times m^{SLM} + n^{SLM}(m^{SLM} - 1)]$.

In contrast to this high speed of calculation, we need to point out that the process of loading the data, i.e. the matrix and vector elements in the two SLMs forming our optical processor, can actually have an important impact on the performance. Let f_R be the frequency at which each SLM can completely change the state of all its pixels. We remark that the data can be loaded independently in the two SLMs because the data (the vector in the first SLM, the matrix in the second one) are independent. Therefore, taking into account not only the speed for the calculations, but also the speed to load the necessary data, and supposing that the two SLMs that form the optical processor are equal, the computing rate (in terms of MAC operations) for the processor is $[n^{SLM} \times m^{SLM} + n^{SLM}(m^{SLM} - 1)] \times f_R$. This quantity is expressed in Hz.

Consequently, the performance of the optical processor depend on the values of n^{SLM}, m^{SLM} and f_R for the two SLMs. We give two possible examples, considering commercially available SLMs. First, for the widely used SLMs, for which $n^{SLM} \times m^{SLM}$ corresponds to 920×1080 and $f_R = 50$ Hz, the frequency of our processor will be about 0.24 GHz. With more recent SLMs benefiting of larger screens and slightly faster refresh rates [12], $n^{SLM} \times m^{SLM}$ corresponds to 2464×4160 and $f_R = 58$ Hz, and the frequency of our processor would be of about 1.2 GHz. Due to the relatively low refresh rate on the order of tens of Hz, the expected performance of

our processor is about three orders of magnitude slower than that achieved in [7], which is about 2 Tera MAC operations per second. However, SLM technology is developing very fast and a modulation rate of 5.4 MHz was reported for a solid-state SLM [13]. Therefore, we expect that operation rates in the THz range will become attainable for our processor in the near future.

Another possibility to upgrade the scheme of our processor is to simply set up a panel composed by several SLMs at the first and second level of our processor, instead of only one per level (i.e. only one for preparing the vector, and only another for the matrix). The architecture of this extended processor enables that one unique wavefront can reach all SLMs, so that they are perfectly synchronized during the calculations.

This is certainly a more expensive solution, because the cost of each SLM needs to be multiplied by the total number of employed SLMs in the two panels. However, we believe this solution should be considered as a candidate to solve important problems. Furthermore, the cost of the SLMs may be reduced with newer technologies. Our optical processor has *ad minima* the advantage that it admits a feasible realization, which is already fully understandable from a physical point of view. Moreover, even if a "useful" quantum computer will be constructed in the next years, it is not evident at the moment whether it will be able to provide a speed-up for operations such as a matrix-by-vector multiplication.

D. Comparison with other optical schemes

Reference [7] implements an optical matrix-by-vector multiplication scheme using an architecture based on the manipulation of the time-frequency degrees of freedom of light and achieves the impressive speed of 2 Tera MAC operations per second. Reference [14], introduced in 1986, reports on a scheme based on the spatial degrees of freedom of light. However, at that time the SLM did not exist and the scheme was essentially realized in a single run, handling vector and matrix dimension of 100 in 20 ns. For comparison, in a single run our scheme may exceed dimensions of 1000 using SLMs. A more recent implementation [15] adopts an architecture very similar to ours in the sense that vector and matrix are encoded in programmable phase/amplitude modulation devices. However, they use a digital micro-mirror device (DMD) and one SLM. Moreover, they do not use the polarization and the dimension of vector and matrix that are reported corresponds to 56, well below our scheme. In terms of speed, the repetition rate is also limited by the refresh rate of the SLM. Recalling that the MAC rate depends on the square of the dimension, their scheme is four orders of magnitude slower than ours. Finally, there is a processor based on a silicon chip optical device [16] and the reported performance in terms of MAC operations per second is 10 Giga MAC operations per second, i.e., about ten times faster than the present version of our scheme. However, the perspective for further improvement is not clear.

E. A note about the hardware

The proposed optical processor can be realized with regular commercial devices like a diode laser, SLMs, regular optical components and an ordinary computer for controlling the system. The classical computer needed to control our optical processor does not need to have any particular features (any commercial computer would be able to make the job, because its main task consists in simply switching the masks of the SLMs every ~ 17 ms, and receiving the data from the detector (a regular CCD camera)).

The laser beam power can be in the range of mW, when employing only one SLM for the vector preparation, and one for the matrix multiplication. In the case of the panel, the power requirement increases with P: the requirement depends on the area covered by the wavefront. For example, the power required in order to shine a panel with 25 SLMs is ~ 225 mW, which is easily obtained with rather economic commercial diode lasers. There is no requirement on the spectral bandwidth, but transverse spatial coherence is required, which is however a common feature to laser light.

A more sensible issue concerns the diffraction and the power stability of the laser. In order to avoid errors in the processing, it is important to have good quality optical components and low intensity fluctuations along the optical wavefront. These points must be tested in a realistic setup.

Specification example: HD SLM (1920 $\times$ 1080 pixel), 60Hz refresh rate - Holoeye Pluto 2.1, Thorlabs Exulus HD1, Meadowlark E-Series, 4K SLM - Holoeye GAEA-2 (4160 $\times$ 2464), Thorlabs 4K1 (38400 $\times$ 2160). Diode laser - There is a very large variety of diode lasers with power above 100mW in the visible range. One example is the Thorlabs L637G1, @637nm, 1200 mW power in continuous wave. There is also a large variety of CCD cameras that would fit the requirements. For instance the Thorlabs 340M-USB, VGA resolution and 200 frames per second. This frame rate is about three times faster than the SLM refresh rate. There is no special requirement for the lenses and polarization optics. The control computer should have two HDMI output ports and there is no special requirement for the CPU and memory.

In the second part of our paper (starting from next section), we will focus on a particular case study, where the considered NP-hard problem can be reformulated as one unique matrix-by-vector multiplication.

III. OUR CASE STUDY:

DISTANCE GEOMETRY

The Distance Geometry Problem (DGP) asks whether a simple weighted undirected graph $G = (V, E, d)$ can be realized in the Euclidean space $\mathbb{R}^K$ so that the distance between two realized vertices corresponds to the real-valued weight $d(u, v)$, for all edges $\{u, v\} \in E$ [17]. In other words, we look for vectors $x_1, \ldots, x_N$ in $\mathbb{R}^K$, where $N = |V|$, such that $|x_u - x_v| = d(u, v), \forall \{u, v\} \in E$. We point out that the weights $d(u, v)$ associated with the edges $\{u, v\} \in E$ are supposed to be *exact*, i.e. very precise, in this work.

The DGP is NP-hard [18], and there exists a lot of recent scientific literature on this topic. In dimension $K = 3$, one traditional and widely studied DGP application arises in the context of structural biology [19]. The Sensor Network Localization Problem (SNLP) is basically a DGP that can be defined in dimensions $K = 3$ or 2 [20]. In this work, we will rather restrict ourselves to DGPs arising from applications in dimension $K = 1$ [21]. An example of application in only one dimension is the Angular Synchronization Problem (ASP), where it is required to obtain an estimation (up to a constant additive phase) for a set of unknown angles by using the information about some measurements of their offsets, modulo 2π [22]. The problem of synchronizing the clocks in a distributed network [23] is basically an ASP where the angles are replaced with time measures that are not periodic, making the problem actually correspond to a DGP in dimension 1.

Definition III.1. A simple weighted undirected graph $G = (V, E, d)$ represents a "paradoxical instance" of the DGP in dimension 1 if and only if G is a cycle graph.

Since G is a single cycle, we can enumerate its vertices: $1, \ldots, N = |V| = |E|$ and use the rank k in this *vertex order* for indicating the corresponding vertex v in V.

The paradoxical character of our instances comes from the following remark. While it can be proved that the number of solutions for these instances is always 2 (see [24] for the detailed proof), the removal of a single edge, e.g. $\{1, N\}$, makes the total number of solutions increase to 2^{N-1} (of which only two will satisfy the constraint $|x_1 - x_N| = d(1, N)$).

From here on we shall use the short notation $d_{ij} = d(i, j)$ to denote the distance (edge weight) between vertices i and j in a given vertex order.

A. The Branch-and-Prune algorithm

Before introducing our reformulation of the paradoxical DGP in dimension 1 (see next section) as a unique matrix-by-vector multiplication, we briefly present the Branch-and-Prune (BP) algorithm [25], which is our starting point for the reformulation. Let G be a graph representing an instance of the paradoxical DGP (see Def. III.1).

The BP algorithm starts by positioning the vertex 1 in the origin of the real line. Let $x_1 = 0$ be this position. Then, since by hypothesis the distance $d_{k-1,k}$ is known for every $k > 1$, we can compute the two possible positions for all other subsequent vertices k by applying the formula:

$$x_k = x_{k-1} + s_k d_{k-1,k}, \quad \forall k = 2, \ldots, N, \tag{7}$$

where the value of s_k is $+1$ when computing the first position, and it is instead set to -1 in the second computed position x_k. Since there are potentially two distinct positions for x_k for *every* previously computed position for x_{k-1}, the set of all vertex positions admits the structure of a tree, where layers contain the various positions for the same vertex k, and a path from the root node (the vertex 1) to any of the leaf nodes represents a candidate realization to our instance of the DGP (a more detailed description is given in Appendix A). This part of the BP algorithm is named the *branching* phase.

The *pruning* phase of the algorithm consists instead in exploiting additional distance information, that was not used during the previous phase, with the aim of verifying the feasibility of computed vertex positions w.r.t. the full distance information. If some of these additional distances are not satisfied, then the current position, and hence the entire tree branch having as a root this position, can be pruned away from the tree.

It is important to remark that, in case of paradoxical DGP instances, these additional distances are actually absent until the layer N of the tree is reached. It is only there where two distances (and not only one) are available to find out the potential positions for the vertex N, the last in the vertex order. The first distance, say $d_{N-1,N}$, can be used for branching, by applying Eq. (7) as in the previous steps. Then, the second distance, say $d_{1,N}$, can be used for pruning purposes: all positions that were generated during the branching phase such that $|x_1 - x_N| \neq d_{1,N}$ can actually be removed from the tree.

Moreover, for a paradoxical DGP instance, we can add a fictive vertex, name it "$N+1$", and add it to the original graph G. Then, remove the original edge $\{1, N\}$ and add the new edge $\{N, N+1\}$ with the same weight $d_{N,N+1} = d_{1,N}$. This way, it is straightforward to verify that Eq. (7) can be applied over all tree layers, in the given vertex ordering from 2 to $N + 1$, and that the solutions to this DGP are all realizations for which

$$x_1 = x_{N+1}.$$

Notice that, if there exists no realization satisfying this identity, then the DGP instance at hand is infeasible.

B. A matrix-by-vector reformulation

The basic idea behind the reformulation of our paradoxical DGP as a matrix-by-vector multiplication consists in representing a candidate realization as a Boolean

row N-vector, where the values of the elements are either -1 or $+1$. For example, the vector

$$(-1, +1, +1, \ldots, -1, -1),$$

representing a realization, when multiplied by the column vector $\mathbf{y}$, defined as

$$\mathbf{y} = \begin{bmatrix} d_{12} \\ d_{23} \\ \vdots \\ d_{N-1,N} \\ d_{1N} \end{bmatrix},$$

provides the position of the fictive vertex "$N+1$". As indicated above, if $x_1 = 0$, then this realization is feasible if and only if $x_{N+1} = 0$ as well.

In order to take into consideration all possible realizations at once, we define a matrix $\mathbf{M}$ which is capable of holding all possible Boolean vectors encoding these realizations. Every row of this matrix corresponds to one realization: since the total number of potential realizations is 2^{N-1}, $\mathbf{M}$ has, in total, 2^N rows (because of the inclusion of the fictive vertex); moreover, since the number of vertices is N, $\mathbf{M}$ has N columns.

While the construction of the vector $\mathbf{y}$ is trivial (it simply consists of the list of available distances, in the given vertex order), we have derived a simple formula for an efficient construction of the matrix $\mathbf{M}$. To introduce this formula in general for every problem size $N > 0$, let us first of all consider the simple case where only one vertex is contained in the graph G, so that only the last distance d_{1N} is theoretically present (notice that this is a special instance admitting no solutions unless $d_{1N} = 0$). In this case, we have

$$\mathbf{M} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \qquad \mathbf{y} = (d_{1N}), \tag{8}$$

and there are only two potential solutions, represented by the elements of the vector $\mathbf{r}^T = (-d_{1N}, d_{1N})$.

As remarked above, $\mathbf{M}$ is a $2^N \times N$ matrix. In the first (and unique) column for the matrix $\mathbf{M}$ in Eq. (8), we can remark that the element equal to -1 is the one with the row index i that is odd, whereas the element equal to 1 has the even row index (we suppose that the first index is 1; the formulae can be simply adapted to the case where the first index would instead be 0). Let us add now a second vertex to our instance, in a way that it still satisfies the properties of a paradoxical DGP instance. The matrix and vector to be multiplied are now:

$$\mathbf{M} = \begin{pmatrix} -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad \mathbf{y} = \begin{pmatrix} d_{12} \\ d_{12} \end{pmatrix},$$

resulting in $\mathbf{r}^T = (-2d_{12}, 0, 0, 2d_{12})$. Again, this is only a theoretical instance, which always admits two solutions (the ones corresponding to the rows $(-1, 1)$ and $(1, -1)$, for every possible value for the distance d_{12}). Even if only theoretical, this instance allows us to have a clearer idea on the general formula for the construction of the matrix $\mathbf{M}$. It is easy to verify that the same rule identified above is still true for the new rows of the matrix, as far as the first column is concerned. This rule is equivalent to saying that the element value is -1 if and only if $(i-1) \bmod 2$ is zero. For the second column, a similar rule can be identified: the element is -1 if and only if $(i-1)/2 \bmod 2 = 0$, where $(i-1)/2$ is the *integer division*. Notice the fact that the row indices are divided by 2 in the rule concerning the second column.

From these rules the following general formula can be derived:

$$M_{ij} = \begin{cases} -1 & \text{if } (i-1)/2^{j-1} \bmod 2 = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Concerning the encoding of such matrix $\mathbf{M}$ in the SLM, we observe that only two gray levels are necessary because the elements of the matrix $\mathbf{M}$ may have only two values in this case: -1 or 1. However, these values, as well as the elements of the distance vector $\mathbf{y}$, need to be mapped to the interval $[0, 1]$ in order to be represented by $\cos(\alpha_{ij})$ and $\cos(\phi_j)$, respectively (see Section IIB). This can be accomplished by the transformations

$$M_{ij} \leftarrow (M_{ij} + 1)/2, \quad y_j \leftarrow \frac{y_j}{d_{\max}},$$

where $d_{\max}$ is the maximum distance. The corresponding paradoxical DGP has a solution if the output vector $\mathbf{r}$ has a component

$$r_i = \frac{\sum_{j=2}^{N+1} d_{k-1,k}}{2d_{\max}}.$$

We can remark that the definition of every element of the matrix does not depend on any of the others, so that every submatrix of $\mathbf{M}$ can be easily constructed. This property of our general formula has a very important, positive impact on our optical processor. We can notice, in fact, that the variant of our processor discussed in Section II C, where a SLM is replaced by a panel grid of P SLMs, the several SLMs in the panel do not need to exchange data and they can all work in perfect parallelism and synchronization. Notice that, if a matrix and vector forming our paradoxical DGP instance can entirely fit in the two panels, then the exponential number of possible realizations is verified "in one shot" by one unique light beam. The exponential complexity is in this way treated by only one cycle of our optical processor.

IV. CONCLUSION

In conclusion, we present a new architecture for the realization of an optical processor. It is based on the

modulation of an optical wavefront by spatial light modulators and it is able to perform fast matrix-by-vector multiplication. We show that it can be realized in practice with commercially available devices and achieve a processing speed in the range of GHz and has a considerable potential for attaining impressive rates above Tera MAC operations per second.

We also show that the one-dimensional distance geometry problem can be cast as a problem of matrix-by-vector multiplication. In this way it could be solved using the proposed optical processor for dimensions as high as 30 with the currently available devices and the perspective of reaching even higher dimensions with faster switching SLMs that are currently being developed.

Optical computation schemes for NP-hard problems (such as the one proposed and studied in this work) show that classical approaches exploring the wave properties of light can be helpful in order to access and realistically treat hard problems in large scale.

ACKNOWLEDGMENTS

Funding was provided by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Fundação de Amparo à Pesquisa do Estado de Santa Catarina (FAPESC), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Instituto Nacional de Ciência e Tecnologia de Informação Quântica (INCT-IQ 465469/2014-0), and the ANR French funding agency (project MULTIBIOSTRUCT, ANR-19-CE45-0019).

[1] Note1, we avoid going through the rigorous definitions of NP-hardness, because it is out of scope of this publication. Roughly speaking, a problem is NP-hard if it is as hard as any other problem in the so-called "NP" class. We refer to [26] for a precise definition.

[2] M. Oltean and O. Muntean, Natural Computing **8**, 321 (2009), ISSN 1572-9796, 0708.1964.

[3] X.-Y. Xu, X.-L. Huang, Z.-M. Li, J. Gao, Z.-Q. Jiao, Y. Wang, R.-J. Ren, H. P. Zhang, and X.-M. Jin, Science Advances **6**, eaay5853 (2020), ISSN 2375-2548.

[4] T. Haist and W. Osten, Optics Express **15**, 10473 (2007), ISSN 1094-4087.

[5] N. T. Shaked, S. Messika, S. Dolev, and J. Rosen, Applied Optics **46**, 711 (2007), ISSN 2155-3165.

[6] C. Lavor, L. Liberti, N. Maculan, and A. Mucherino, Computational Optimization and Applications **52**, 115 (2012).

[7] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, et al., Nature **589**, 52 (2021), ISSN 1476-4687.

[8] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen (1990).

[9] N. Bell and M. Garland, in Proceedings of the conference on high performance computing networking, storage and analysis (2009), pp. 1–11.

[10] A. Monakov, A. Lokhmotov, and A. Avetisyan, in International Conference on High-Performance Embedded Architectures and Compilers (Springer, 2010), pp. 111–125.

[11] E. C. Paul, M. Hor-Meyll, P. H. S. Ribeiro, and S. P. Walborn, Scientific Reports **4**, 5337 (2014), ISSN 2045-2322.

[12] Note2, see for example the SLM GAEA-2 10 MEGAPIXEL PHASE ONLY LCOS-SLM (REFLECTIVE) by Holoeye Photonics AG.

[13] J. Park, B. G. Jeong, S. I. Kim, D. Lee, J. Kim, C. Shin, C. B. Lee, T. Otsuka, J. Kyoung, S. Kim, et al., Nature Nanotechnology **16**, 69 (2021), ISSN 1748-3395, URL `https://doi.org/10.1038/s41565-020-00787-y`.

[14] K. C. Byron, Patent: Optical matrix-vector multiplication (1986).

[15] J. Spall, X. Guo, T. D. Barrett, and A. I. Lvovsky, Opt. Lett. **45**, 5752 (2020), URL `http://www.osapublishing.org/ol/abstract.cfm?URI=ol-45-20-5752`.

[16] L. Yang, R. Ji, L. Zhang, J. Ding, and Q. Xu, Opt. Express **20**, 13560 (2012), URL `http://www.osapublishing.org/oe/abstract.cfm?URI=oe-20-12-13560`.

[17] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino, SIAM Review **56**, 3 (2014).

[18] J. Saxe, Proceedings of 17^{th} Allerton Conference in Communications, Control and Computing pp. 480–489 (1979).

[19] T. Malliavin, A. Mucherino, C. Lavor, and L. L., Journal of Chemical Information and Modeling **59**, 4486 (2019).

[20] P. Biswas, T. Lian, T. Wang, and Y. Ye, ACM Transactions in Sensor Networks **2**, 188 (2006).

[21] A. Mucherino, Studies in Computational Intelligence **717**, 123 (2018).

[22] A. Singer, Applied and Computational Harmonic Analysis **30**, 20– (2011).

[23] A. Giridhar and P. Kumar, in 45^{th} IEEE Conference on Decision and Control 2006 (IEEE, 2006), pp. 4915—-4920.

[24] L. Liberti, B. Masson, J. Lee, C. Lavor, and A. Mucherino, Discrete Applied Mathematics **165**, 213 (2014).

[25] L. Liberti, C. Lavor, and N. Maculan, International Transactions in Operational Research **15**, 1 (2008).

[26] M. Garey and D. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness (Freeman and Company, New York, 1979).

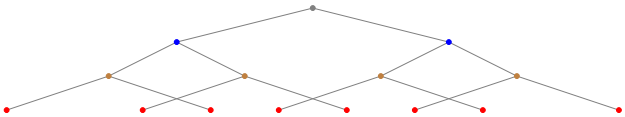
APPENDIX A

In the following, we will give a detailed example of a DGP paradoxical instance in dimension 1, and the use

of the BP algorithm for its solution. The drawing below shows the expected solution to our paradoxical DGP example.

We suppose that the vertices are sorted by considering the following vertex order: gray $\longrightarrow$ blue $\longrightarrow$ brown $\longrightarrow$ red. We also suppose that the distance between every vertex and its preceding vertex (except the gray vertex) is available, as well as the distance between the gray and the red vertex (in other words, this instance satisfies the assumptions in Def. III.1).

In order to find possible solutions to our instance, the BP algorithm initially places at the center of the coordinate system (in the position of coordinate 0) the first vertex, and then it generates the candidate positions for the subsequent vertices by exploiting the available distance information. For example, there exist two possible positions for the blue vertex, and, for every selected position of the blue vertex, there are other two positions for the brown vertex, and so on.



The tree above shows all the possible positions, for every vertex in the DGP instance, that can be constructed by using the entire distance information except the distance between the gray (the first) and the red (the last) vertex. The fact that our instance is in dimension 1 allows us to represent this tree with a drawing where the distances between possible positions for the same vertex are preserved in all parallel axes passing through the vertex positions (it is supposed that the axis passing through the only position for the gray vertex is parallel to all others). In this way, the *entire* set of possible positions for all the vertices in the instance can be obtained by projecting all these positions on one unique line:

Notice that we have marked with a larger circle the positions belonging to the expected solution represented above. The interesting question is: how to select our solutions from the set of given solutions? Every branch of the tree above gives in fact a set of possible positions for the 4 vertices in our instance that satisfies all distances that we have considered (up to now).

The selection of the actual solution set can be in fact performed by exploiting the distance between the gray and the red vertex (which has not been used yet). As it is possible to remark from the drawing above depicting one expected solution, the distance between the gray and red vertex is rather small, so that only two branches of the tree are likely to satisfy this distance constraint. The final solution set therefore contains two solutions, the expected one, plus its symmetric with respect to the position of the first vertex:

As already remarked in the Introduction (and this is the reason why we named the instances defined in Def. III.1 the *paradoxical instances*), the instance considered in our example has a particular difficult structure. In fact, while the final solution set only contains two symmetric solutions, the number of candidate solutions, only one layer before reaching the tree leaf nodes, is 2^{N-1}. In our example this exponential number is rather small, but it can become huge with a little larger instances. With the introduction of the fictive vertex "N+1" (see Section III B), we can reduce the pruning phase of the algorithm to an additional branching phase, with a verification on the position of this fictive vertex. This brings to a total complexity of 2^N, that our optical scheme can ideally lower down to 1 when the matrix-by-vector multiplication can perform all operations detailed in this Appendix in one unique run.