Optical computing ⋆

Damien Woods [a]
[a] *Department of Computer Science and Artificial Intelligence*
University of Seville, Spain

Thomas J. Naughton [b,c]
[b] *Department of Computer Science*
National University of Ireland, Maynooth, County Kildare, Ireland
[c] *University of Oulu, RFMedia Laboratory, Oulu Southern Institute, Vierimaantie 5, 84100 Ylivieska, Finland*

Abstract

In this survey we consider optical computers that encode data using images and compute by transforming such images. We give an overview of a number of such optical computing architectures, including descriptions of the type of hardware commonly used in optical computing, as well as some of the computational efficiencies of optical devices. We go on to discuss optical computing from the point of view of computational complexity theory, with the aim of putting some old, and some very recent, results in context. Finally, we focus on a particular optical model of computation called the continuous space machine. We describe some results for this model including characterisations in terms of well-known complexity classes.

Key words:
optical computing, optical algorithm, optical implementation, continuous space machine, computational complexity, Fourier transform, search algorithm

⋆ DW acknowledges support from Junta de Andalucía grant TIC-581. TN acknowledges support from the European Commission Framework Programme 6 through a Marie Curie Intra-European Fellowship.
Email addresses: dwoods@us.es (Damien Woods), tomn@cs.nuim.ie (Thomas J. Naughton).
URLs: http://www.cs.us.es/∼dwoods/ (Damien Woods), http://www.cs.nuim.ie/∼tnaughton/ (Thomas J. Naughton).

1. Introduction

In this survey we consider optical computers that encode data using images and compute by transforming such images. We try to bring together, and thus give context to, a large range of architectures and algorithms that come under the term optical computing.

In Section 2 we begin by stating what we mean by the term optical computing, and we discuss some common optical computing architectures. Unlike a number of other areas of nature inspired computing, optical computers have existed (mostly in laboratories) for many years, and so in Section 3 we describe hardware components that are found in many optical computing systems. In Section 4 we describe some of the physical principles that underlie the parallel abilities and efficiencies found in optical computing. These include high fan-in, high interconnection densities, and low energy consumption.

In order to understand the abilities of optics as a computing medium we would argue that computational complexity theory is an indispensable tool. So in Section 5 we collect a number of existing algorithmic results for optical computers. In particular we focus on algorithms and models for matrix-vector multipliers, and some other architectures. We also offer suggestions for future work directions for optical algorithm designers. In Section 6, we take a detailed look at a particular model of optical computing (the continuous space machine, or CSM) that encompasses most of the functionality that coherent optical information processing has to offer. We begin by defining the CSM and a total of seven complexity measures that are inspired by real-world (optical) resources. We go on to discuss how the CSM's operations could be carried out physically. Section 7 contains some example datastructures and algorithms for the CSM. In Section 8 we motivate and introduce an important restriction of the model called the $\mathcal{C}_2$-CSM, and in Section 9 we briefly describe a number of $\mathcal{C}_2$-CSM computational complexity results, and their implications.

2. Brief overview of optical computing architectures

Traditionally, in optical information processing a distinction was made between signal/image processing through optics and numerical processing through optics, with only the latter (and often only the digital version of the latter) being called optical computing [58,49,30,103]. However, it was always difficult to clearly delineate between the two, since it was largely a question of the interpretation the programmer attached to the output optical signals. The most important argument for referring to the latter only as optical computing had to do with the fact that the perceived limits (or at least, ambitions) of the former was simply for special-purpose signal/image processing devices while the ambitions for the latter was general-purpose computation. Given recent results on the computational power of optical image processing architectures [64,100,94], it is not the case that such architectures are limited to special-purpose tasks.

Furthermore, as the field become increasingly multidisciplinary, and in particular as computer scientists play a more prominent role, it is necessary to bring the definition of optical computing in line with the broad definition of computing. In particular, this facilitates analysis from the theoretical computer science point of view. The distinction between analog optical computing and digital optical computing is similarly blurred given the prevalence of digital multiplication schemes effected through analog convolution [49].

Our broad interpretation of the term optical computing has been espoused before [19].

2.1. *Optical pattern recognition*

Arguably, optical computing began with the design and implementation of optical systems to arbitrarily modify the complex valued spatial frequencies of an image. This concept, spatial filtering [68,87,90], is at the root of optics' ability to perform efficient convolution and correlation operations. In a basic pattern recognition application, spatial filtering is called matched filtering, where a filter is chosen that matches (i.e. conjugates) the spectrum of the sought object. Employing this operation for advanced pattern recognition [12,18,44,48], effort focused on achieving systems invariant to scaling, rotation, out-of-plane rotation, deformation, and signal dependent noise, while retaining the existing invariance to translating, adding noise to, and obscuring parts of the input. Effort also went into injecting nonlinearities into these inherently linear systems to achieve wider functionality [47]. Improvements were made to the fundamental limitations of the basic matched filter design, most notably the joint transform correlator architecture [93].

Optical correlators that use incoherent sources of illumination (both spatially and temporally) rather than lasers are also possible [11,72]. The simplest incoherent correlator would have the same basic spatial filtering architecture as that used for matched filtering. While coherent systems in principle are more capable than incoherent systems (principally because the former naturally represents complex functions while the latter naturally represents real functions), incoherent systems require less precise positioning when it comes to system construction and are less susceptible to noise.

Trade-offs between space and time were proposed and demonstrated. These included time integrating correlators [91] (architecturally simpler linear time variants of the constant time space integrating matched filter mentioned already) and systolic architectures [21]. In addition to pattern recognition, a common application for these classes of architectures was numerical calculation.

2.2. *Analog optical numerical computation*

An important strand of image-based optical computation involved numerical calculations: analog computation as well as multi-level discrete computation. Matrix-vector and matrix-matrix multiplication systems were proposed and demonstrated [43,58,91,49,4,30,52]. The capability to expand a beam of light and to focus many beams of light to a common point directly corresponded to high fan-out and fan-in capabilities, respectively. The limitations of encoding a number simply as an intensity value (finite dynamic range and finite intensity resolution in the modulators and detectors) could be overcome by representing the numbers in some base. Significant effort went into dealing with carry operations so that in additions, subtractions, and multiplications each digit could be processed in parallel. Algorithms based on convolution to multiply numbers in this representation were demonstrated [49], with a single post-processing step to combine the sub-calculations and deal with the carry operations.

An application that benefited greatly from the tightly-coupled parallelism afforded by optics was the solving of sets of simultaneous equations and matrix inversion [17,1]. An application that, further, was tolerant to the inherent inaccuracies and noise of analog

optics was optical neural networks [29,20] including online neural learning in the presence of noise [63].

2.3. *Digital optical computing*

The next major advances came in the form of optical equivalents of digital computers [45]. The flexibility of digital systems over analog systems in general was a major factor behind the interest in this form of optical computation [79]. Specific drawbacks of the analog computing paradigm in optics that this new paradigm addressed included no perceived ability to perform general purpose computation, accumulation of noise from one computation step to another, and systematic errors introduced by imperfect analog components. The aim was to design digital optical computers that followed the same principles as conventional electronic processors but which could perform many binary operations in parallel. These systems were designed from logic gates using nonlinear optical elements: semitransparent materials whose transmitted intensity has a nonlinear dependence on the input intensity.

Digital optical computing was also proposed as an application of architectures designed originally for image-based processing, for example logic effected through symbolic substitution [10]. At the confluence of computing and communication, optical techniques were proposed for the routing of signals in long-haul networks [30,103].

3. Optical computing hardware

The three most basic hardware components of an optical information processing system are a source, a modulator, and a detector. A source generates the light, a modulator multiplies the light by a (usually, spatially varying) function, and a detector senses the resulting light. These and others are introduced in this section.

3.1. *Sources*

Lasers are a common source of illumination because at some levels they are mathematically simpler to understand, but incoherent sources such as light-emitting diodes are also used frequently for increased tolerance to noise and when nonnegative functions are sufficient for the computation. Usually, the source is monochromatic to avoid the problem of colour dispersion as the light passes through refracting optical components, unless this dispersion is itself the basis for the computation.

3.2. *Spatial light modulators (SLMs)*

It is possible to encode a spatial function (a 2D image) in an optical wavefront. A page of text when illuminated with sunlight, for example, does this job perfectly. This would be called an amplitude-modulating reflective SLM. Modulation is a multiplicative effect, so an image encoded in the incoming wavefront will be pointwise multiplied by the image on the SLM. Modulators can also act on phase and polarisation, and can be transmissive rather than reflective. They include photographic film, and electro-optic, magneto-optic,

and acousto-optic devices [58,4,49,30,36]. One class of note are the optically-addressed SLMs, in which, typically, a 2D light pattern falling on a photosensitive layer on one side of the SLM spatially varies (with an identical pattern) the reflective properties of the other side of the SLM. A beam splitter then allows one to read out this spatially-varying reflectance pattern. The liquid-crystal light valve [46] is one instance of this class. Other classes of SLMs such as liquid-crystal display panels and acousto-optic modulators allow one to dynamically alter the pattern using electronics. It is possible for a single device (such as an electronically programmed array of individual sources) to act as both source and modulator.

3.3. *Detectors and Nature's square law*

Optical signals can be regarded as having both an amplitude and phase. However, detectors will measure only the square of the amplitude of the signal (referred to as its intensity). This phenomenon is known as Nature's detector square law and applies to detectors from photographic film to digital cameras to the human eye. Detectors that obey this law are referred to as square-law detectors. This law is evident in many physical theories of light. In quantum theory, the measurement of a complex probability function is formalised as a projection onto the set of real numbers through a squaring operation. Square-law detectors need to be augmented with a interferometric or holographic arrangement to measure both amplitude and phase rather than intensity [13], or need to be used for multiple captures in different domains to heuristically infer the phase.

Since it squares the absolute value of a complex function, this square law can be used for some useful computation (for example, in the joint transform correlator [93]). Detectors most commonly used include high range point (single pixel) detectors such as photodiodes, highly sensitive photon detectors such as photomultiplier tubes, and 1D and 2D array detectors such as CCD- or CMOS-digital cameras. Intensity values outside the range of a detector (outside the lowest and highest intensities that the detector can record) are thresholded accordingly. The integration time of some detectors can be adjusted to sum all of the light intensity falling on them over a period of time. Other detectors can have quite large light sensitive areas and can sum all of the light intensity falling in a region of space.

3.4. *Other optical computing hardware*

Lenses can be used to effect high fan-in and fan-out interconnections, to rescale images linearly in either one or two dimensions, and for taking Fourier transforms. In fact, a coherent optical wavefront naturally evolves into its Fresnel transform, and subsequently into its Fourier transform at infinity, and the lens simply images those frequency components at a finite fixed distance.

A mirror changes the direction of the wavefront and simultaneously reflects it along some axis. A phase conjugate mirror [25] returns an image along the same path at which it approached the mirror.

Prisms can be used to for in-plane flipping (mirror image), in-plane rotations, and out-of-plane tilting. A prism or diffraction grating can be used to separate by wavelength the components of a multi-wavelength optical channel. For optical fiber communications ap-

plications, more practical (robust, economical, and scalable) alternatives exist to achieve the same effect [103].

Polarisation is an important property of wavefronts, in particular in coherent optical computing, and is the basis for how liquid crystal displays work. At each point, an optical wavefront has an independent polarisation value dependent on the angle, in the range $[0, 2\pi)$, of its electrical field. This can be independent of its successor (in the case of randomly polarised wavefronts), or dependent (as in the case of linear polarisation), or dependent and time varying (as in the case of circular or elliptical polarisation). Mathematically, a polarisation state, and the transition from one polarisation state to another, can be described using the Mueller calculus or the Jones calculus.

Photons have properties such as phase, polarisation, and quantum state that can be used for computation. For example, quantum computers using linear optical elements (such as mirrors, polarisers, beam splitters, and phase shifters) have been proposed and demonstrated [51].

4. Efficiencies in optical computing

Optical computing is an inherently multidisciplinary subject whose study routinely involves a spectrum of expertise that threads optical physics, materials science, optical engineering, electrical engineering, computer architecture, computer programming, and computer theory. Applying ideas from theoretical computer science, such as analysis of algorithms and computational complexity, enables us to place optical computing in a framework where we can try to answer a number of important questions. For example, which problems are optical computers suitable for solving? Also, how does the resource usage on optical computers compare with more standard (e.g. digital electronic) architectures? The physical principles behind some efficiencies in optical computing are outlined here. Then, in Section 5, we go to describe models and architectures that exploit these efficiencies.

4.1. *Fan-in efficiency*

Kirchoff's Law is well understood in analog electronics as a natural and constant-time means of summing the current at the intersection of an arbitrary number of wires [59]. In optics, the same thing is possible by directing several light beams towards a point detector with a linear response to incident light. Such an optical arrangement sums n nonnegative integers in $O(1)$ addition steps. On a model of a sequential digital electronic computer this would require $n - 1$ addition operations and even many typical (bounded fan-in) parallel models, with n or more processors, take $O(\log n)$ timesteps. Tasks that rely on scalar summation operations (such as matrix multiplication) would benefit greatly from an optical implementation of the scalar sum operation. Similarly, $O(1)$ multiplication and $O(1)$ convolution operations can be realised optically. Very recently, an optics-based digital signal processing platform has been marketed that claims digital processing speeds of tera (10^{12}) operations per second [54].

4.2. *Efficiency in interconnection complexity*

As optical pathways can cross in free space without measurable effect on the information in either channel, high interconnection densities are possible with optics [20]. Architectures with highly parallel many-to-many interconnections between parallel surfaces have already been proposed for common tasks such as sorting [7]. Currently, intra-chip, inter-chip, and inter-board connections are being investigated for manufacturing feasibility [60].

4.3. *Energy efficiency*

Electrical wires suffer from induced noise and heat, which increases dramatically whenever wires are made thinner or placed closer together, or whenever the data throughput is increased [60]. As a direct consequence of their resistance-free pathways and noise-reduced environments, optical systems have the potential to generate less waste heat and so consume less energy per computation step than electronic systems [14]. This has been demonstrated experimentally with general-purpose digital optical processors [39].

5. Optical models of computation and computational complexity

On the one hand, there has been a lot of effort put into designing optical computers to emulate conventional microprocessors (digital optical computing), and to image processing over continuous wavefronts (analog optical computing and pattern recognition). Numerous physical implementations of the latter class exist, and example applications include fast pattern recognition and matrix-vector algebra [36,91]. There have been much resources devoted to designs, implementations and algorithms for such optical information processing architectures (for example see [4,15,30,36,53,56,58,63,78,91,103,28] and their references).

On the other hand, the computational complexity theory of optical computers (that is, finding lower and upper bounds on computational power in terms of known complexity classes) has received relatively little attention when compared with other nature-insired computing paradigms. Some authors have even complained about the lack of suitable models [30,56]. Many other areas of natural computing (e.g. [42,2,55,102,83,38,71,61,62]) have not suffered from this problem. Despite this, we review a number of algorithmically orientated results related to optical computing. We then go on to suggest classes of problems where optical computers might be usefully applied.

5.1. *Some optical models of computation*

Reif and Tyagi [78] study two optically inspired models. One model is a 3D VLSI model augmented with a 2D discrete Fourier transform (DFT) primitive and parallel optical interconnections. The other is a DFT circuit with operations (multiplication, addition, comparison of two inputs, DFT) that compute over an ordered ring. Time complexity is defined for both models as number of (parallel) steps. For the first model, volume complexity is defined as the volume of the smallest convex box enclosing an instance of

the model. For the DFT circuit, size is defined as the number of edges plus gates. Constant time, polynomial size/volume, algorithms for a number of problems are reported including matrix multiplication, sorting and string matching [78]. These interesting results are built upon the ability of their models to compute the 2D DFT in one step. The authors suggest that the algorithm designer and optical computing architecture communities should identify other primitive optical operations, besides the DFT, that might result in efficient parallel algorithms. Barakat and Reif [6], and Tyagi and Reif [77], have also shown lower bounds on the optical VLSI model.

Reif, Tygar and Yoshida [76] examined the computational complexity of ray tracing problems. In such problems we are concerned about the geometry of an optical system where diffraction is ignored and we wish to predict the position of light rays after passing through some system of mirrors and lenses. They gave undecidability and PSPACE hardness results, which gives an indication of the power of these systems as computational models.

Feitelson [30] gives a call to theoretical computer scientists to apply their knowledge and techniques to optical computing. He then goes on to generalise the concurrent read, concurrent write parallel random access machine, by augmenting it with two optically inspired operations. The first is the ability to write the same piece of data to many global memory locations at once. Secondly, if many values are concurrently written to a single memory location then a summation of those values is computed in a single timestep. Essentially Feitelson is using 'unbounded fan-in with summation' and 'unbounded fan-out'. His architecture mixes a well-known discrete model with some optical capabilities.

A symbolic substitution model of computation has been proposed by Huang and Brenner, and a proof sketched of its universality [10]. This model of digital computation operates over discrete binary images and derives its efficiency by performing logical operations on each pixel in the image in parallel. It has the functionality to copy, invert, and shift laterally individual images, and OR and AND pairs of images. Suggested techniques for its optical implementation are outlined.

5.2. *Computational complexity of optical models of computation*

In computer science there are two famous classes of problems called P and NP [69]. P contains those problems that are solvable in polynomial time on a standard sequential computer, while NP is the class of problems that are solvable in polynomial time on a nondeterministic computer. NP contains P, and it is widely conjectured that they are not equal. A direct consequence of this conjecture is that there are (NP-complete) problems for which we strongly believe there is no polynomial time algorithm on a standard sequential computer.

It is known that it is possible to solve any NP problem, and even any PSPACE problem, in polynomial time on optical computers, albeit with exponential use of some other, space-like, resources [98,94,96]. These results were shown on the CSM, a general model of a wide range of optical computers. The lower bound results were shown by generating appropriate Boolean masks, of exponential size, and manipulating the masks via parallel multiplications and additions to simulate space bounded Turing machines in a time-efficient way. The model was designed on the one hand to be close to the realities of optical computing, and on the other hand to be relatively straightforward to analyse from

the point of view of computational complexity theory (e.g. see Section 6). PSPACE upper bounds, and a number of other computational complexity results, have also been shown for the model. In Section 9.1 we discuss the computational abilities of this computational model.

Since these general results, there have been a number of specific examples of optical systems (with exponential resource usage) for NP-hard problems.

Shaked et al. [80–82] design an optical system for solving the NP-hard travelling salesman problem in polynomial time. Basically they use an optical matrix-vector multiplier to generate the (exponentially large) matrix of all possible tours, then they multiply this tour matrix by the vector of intercity weights, and finally the lowest value in the resulting vector corresponds to the shortest tour. Interestingly, they give both optical experiments and simulations. They note that solving travelling salesman problems (or Hamiltonian path problems) with more than 15 nodes is problematic. However they argue that for less nodes (e.g. 5) their system works in real-time, which is faster than digital-electronic architectures. Problems with such bounds on input size (i.e. constant) lie in the class NC^1, and moreover in AC^0. As argued below, perhaps this suggests that the optical computing community should be focusing on problems where optics excels over digital-electronic architectures, such as problems in P or NC, rather than NP-hard problems.

Dolev and Fitoussi [27] give optical algorithms that make use of (exponentially large) masks to solve a number of NP-hard problems on two different architectures. Their goal includes giving specific optical algorithms with good implementation prospects, at least on small input sizes.

In 1992, Collings et al [26] gave an optical neural network implementation of a small instance of the travelling salesman problem. Oltean [67], and Haist and Osten [40], give architectures for Hamiltonian path, and travelling salesman problem, respectively, via light travelling through optical cables. As is to be expected, both suffer from exponential resource use. The paper by MacKenzie and Ramachandran [57] is an example of algorithmic work, and lower bounds, on dynamically reconfigurable optical networks.

5.3. *A possible way forward*

Nature-inspired systems that apparently solve NP-hard problems in polynomial time, while using an exponential amount of some other resource(s), have been around for many years. So the existence of massively parallel optical systems for NP-hard problems, such as those previously described, should not really suprise the reader.

One could argue that it is interesting to know the computational abilities, limitations, and resource trade-offs of such optical architectures, as well as to find particular (tractable or intractable) problems which are particularly suited to optical algorithms. However, "algorithms" that use exponential space-like resources (such as number of pixels, number of images, number of amplitude levels, etc.) are not realistic to implement for large input instances. What happens to highly parallel optical architectures if add the requirement that the amount of space-like resources are bounded in some reasonable way? We could, for example, stipulate that the optical machine use no more than a polynomially bounded amount of space-like resources. If the machine runs in polynomial time, then it is not difficult to see that it characterises P [101] (by characterise we mean that the model solves exactly those problems in P), for a wide range of reasonable parallel and sequential

optical models. Many argue that the reason for using parallel architectures is to speed-up computations. Asking for an exponential speed-up motivates the complexity class NC. The class NC can be thought of as those problems in P that can be solved exponentially faster on parallel computers than on sequential computers. NC is contained in P and it is an major open question whether this containment is strict: it is widely conjectured that this is indeed the case [37].

How does this relate to optics? It turns out that a wide range of optical computers that run for at most polylogarithmic time, and use at most polynomial space-like resources, solve exactly NC [98,94,96] (this can be shown to be a corollary of the PSPACE characterisation cited earlier in Section 5.2). In effect, this means that we have an algorithmic way (in other words, a compiler) to convert existing NC algorithms into optical algorithms that use similar amounts of resources. There is scope for further work here, on the CSM in particular, in order to find exact characterisations, or as close as possible for NC^k for given k. On a technical note, NC can be defined as $\cup_{k=0}^{\infty}\mathrm{NC}^k$, where NC^k is the class of problems solvable on a PRAM that runs for $O(\log n)^k$ time and uses polynomial processors/space, in input length n. Equivalently NC^k can be defined as those problems solvable by circuits of $O(\log n)^k$ depth (parallel time), and polynomial size. From the practical side of things, perhaps we can use these kinds of results to find problems within NC, where optical architectures can be shown to excel. Obvious examples for which this is already known are matrix-vector multiplication (which lies in NC^2), or Boolean matrix multiplication (which is in NC^1). Another example is the NC^1 unordered search problem [101,100]. Another closely related idea is to exploit the potential unbounded fan-in of optics to compute problems in the AC, and TC, (parallel) circuit classes. These are defined similarly to NC circuits except we allow unbounded fan-in gates, and threshold gates, respectively. The results in the above mentioned paper of Reif and Tyagi [78], and Caulfield's observation on the benefits of unbounded fan-in [16], can be interpreted as exploiting this important and efficient aspect of optics.

6. Continuous space machine (CSM)

For the remainder of this paper we focus on an optical model of computation called the CSM. The model was originally proposed by Naughton [64,65]. The CSM is inspired by analog Fourier optical computing architectures, specifically pattern recognition and matrix algebra processors [36,63]. For example, these architectures have the ability to do unit time Fourier transformation using coherent (laser) light and lenses. The CSM computes in discrete timesteps over a number of two-dimensional images of fixed size and arbitrary spatial resolution. The data and program are stored as images. The (constant time) operations on images include Fourier transformation, multiplication, addition, thresholding, copying and scaling. The model is designed to capture much of the important features of optical computers, while at the same time be amenable to analysis from a computer theory point of view. Towards these goals we give an overview of how the model relates to optics as well as giving a number of computational complexity results for the model.

Section 6.1 begins by defining the model. We analyse the model in terms of seven complexity measures inspired by real-world resources, these are described in Section 6.2. In Section 6.3 we discuss possible optical implementations for the model. We then go on to give example algorithms and datastructures in Section 7. The CSM definition is

rather general, and so in Section 8 we define a more restricted model called the $\mathcal{C}_2$-CSM. Compared to the CSM, the $\mathcal{C}_2$-CSM is somewhat closer to optical computing as it happens in the laboratory. Finally, in Section 9 we show the power and limitations of optical computing, as embodied by the $\mathcal{C}_2$-CSM, in terms computational complexity theory. Optical information processing is a highly parallel form of computing and we make this intuition more concrete by relating the $\mathcal{C}_2$-CSM to parallel complexity theory by characterising the parallel complexity class NC. For example, this shows the kind of worst case resource usage one would expect when applying CSM algorithms to problems that are known to be suited to parallel solutions.

6.1. *CSM definition*

We begin this section by describing the CSM model in its most general setting, this brief overview is not intended to be complete and more details are to be found in [94].

A complex-valued image (or simply, image) is a function $f : [0,1) \times [0,1) \to \mathbb{C}$, where $[0,1)$ is the half-open real unit interval. We let $\mathcal{I}$ denote the set of complex-valued images. Let $\mathbb{N}^+ = \{1,2,3,\ldots\}$, $\mathbb{N} = \mathbb{N}^+ \cup \{0\}$, and for a given CSM M let $\mathcal{N}$ be a countable set of images that encode M's addresses. An address is an element of $\mathbb{N} \times \mathbb{N}$. Additionally, for a given M there is an *address encoding function* $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ such that $\mathfrak{E}$ is Turing machine decidable, under some *reasonable* representation of images as words.

Definition 1 (CSM) *A CSM is a quintuple* $M = (\mathfrak{E}, L, I, P, O)$*, where*
$\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ *is the address encoding function,*
$L = ((s_\xi, s_\eta), (a_\xi, a_\eta), (b_\xi, b_\eta))$ *are the addresses: sta, a and b, where* $a \neq b$*,*
I *and* O *are finite sets of input and output addresses, respectively,*
$P = \{(\zeta_1, p_{1_\xi}, p_{1_\eta}), \ldots, (\zeta_r, p_{r_\xi}, p_{r_\eta})\}$ *are the* r *programming symbols* ζ_j *and their addresses* (p_{j_ξ}, p_{j_η}) *where* $\zeta_j \in (\{h, v, *, \cdot, +, \rho, st, ld, br, hlt\} \cup \mathcal{N}) \subset \mathcal{I}$*.*
Each address is an element from $\{0, \ldots, \Xi - 1\} \times \{0, \ldots, \mathcal{Y} - 1\}$ *where* $\Xi, \mathcal{Y} \in \mathbb{N}^+$*.*

Addresses whose contents are not specified by P in a CSM definition are assumed to contain the constant image $f(x,y) = 0$. We interpret this definition to mean that M is (initially) defined on a grid of images bounded by the constants Ξ and $\mathcal{Y}$, in the horizontal and vertical directions respectively. The grid of images may grow in size as the computation progresses. In our grid notation the first and second elements of an address tuple refer to the horizontal and vertical axes of the grid respectively, and image $(0,0)$ is located at the lower left-hand corner of the grid. The images have the same orientation as the grid. For example the value $f(0,0)$ is located at the lower left-hand corner of the image f.

In Definition 1 the tuple P specifies the CSM program using programming symbol images ζ_j that are from the (low-level) CSM programing language [94,100]. We refrain from giving a description of this programming language and instead describe a less cumbersome high-level language [94]. Figure 1 gives the basic instructions of this high-level language. The copy instruction is illustrated in Figure 2. There are also **if**/**else** and **while** control flow instructions with conditional equality tests of the form $(f_\psi == f_\phi)$ where f_ψ and f_ϕ are *binary symbol images* (see Figures 3(a) and 3(b)).

Address *sta* is the start location for the program so the programmer should write the first program instruction at *sta*. Addresses a and b define special images that are

$\mathrm{h}(i_1;i_2)$: replace image at i_2 with horizontal 1D Fourier transform of i_1.

$\mathrm{v}(i_1;i_2)$: replace image at i_2 with vertical 1D Fourier transform of image at i_1.

$*(i_1;i_2)$: replace image at i_2 with the complex conjugate of image at i_1.

$\cdot\,(i_1,i_2;i_3)$: pointwise multiply the two images at i_1 and i_2. Store result at i_3.

$+(i_1,i_2;i_3)$: pointwise addition of the two images at i_1 and i_2. Store result at i_3.

$\rho(i_1,z_\mathrm{l},z_\mathrm{u};i_2)$: filter the image at i_1 by amplitude using z_l and z_u as lower and upper amplitude threshold images, respectively. Place result at i_2.

$[\xi_1',\xi_2',\eta_1',\eta_2'] \leftarrow [\xi_1,\xi_2,\eta_1,\eta_2]$: copy the rectangle of images whose bottom left-hand address is (ξ_1,η_1) and whose top right-hand address is (ξ_2,η_2) to the rectangle of images whose bottom left-hand address is (ξ_1',η_1') and whose top right-hand address is (ξ_2',η_2'). See illustration in Figure 2.

Fig. 1. CSM high-level programming language instructions. In these instructions $i, z_\mathrm{l}, z_\mathrm{u} \in \mathbb{N}\times\mathbb{N}$ are image addresses and $\xi,\eta \in \mathbb{N}$. The control flow instructions are described in the main text.

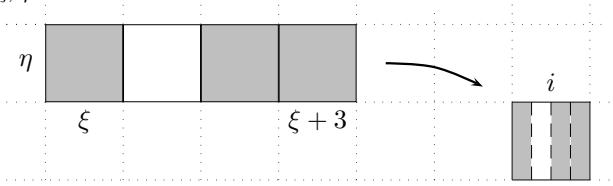

Fig. 2. Illustration of the instruction $i \leftarrow [\xi,\xi+3,\eta,\eta]$ that copies four images to a single image that is denoted i.

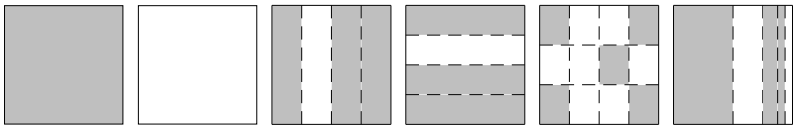

Fig. 3. Representing binary data. The shaded areas denote value 1 and the white areas denote value 0. (a) Binary symbol image representation of 1 and (b) of 0, (c) list (or row) image representation of the word 1011, (d) column image representation of 1011, (e) 3×4 matrix image, (f) binary stack image representation of 1101. Dashed lines are for illustration purposes only.

frequently used by some program instructions. The function $\mathfrak{E}$ is specified by the programmer and is used to map addresses to image pairs. This enables the programmer to choose her own address encoding scheme. We typically don't want $\mathfrak{E}$ to hide complicated behaviour thus the computational power of this function should be somewhat restricted. For example we put such a restriction on $\mathfrak{E}$ in Definition 7. At any given timestep, a configuration is defined in a straightforward way as a tuple $\langle c,e\rangle$ where c is an address called the control and e represents the grid contents.

6.2. *Complexity measures*

In this section we define a number of CSM complexity measures. As is standard, all resource bounding functions map from $\mathbb{N}$ into $\mathbb{N}$ and are assumed to have the usual

properties [5]. We begin by defining CSM TIME complexity in a manner that is standard among parallel models of computation.

Definition 2 *The* TIME *complexity of a CSM M is the number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.*

The first of our six space-like resources is called GRID.

Definition 3 *The* GRID *complexity of a CSM M is the minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.*

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ constant-valued pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$. If we choose a reasonable and realistic S then the details of S are not important.

Definition 4 *The* SPATIALRES *complexity of a CSM M is the minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.*

One can think of SPATIALRES as a measure of the number of pixels needed during a computation. In optical image processing terms, and given the fixed size of our images, SPATIALRES corresponds to the space-bandwidth product of a detector or SLM.

Definition 5 *The* DYRANGE *complexity of a CSM M is the ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.*

In optical processing terms DYRANGE corresponds to the dynamic range of a signal.

We also use complexity measures called AMPLRES, PHASERES and FREQ [94,100]. Roughly speaking, the AMPLRES of a CSM M is the number of discrete, evenly spaced, amplitude values per unit amplitude of the complex numbers in M's images. For example, we would need AMPLRES of 3 to directly store values from the set $\{0, \pm 1/3, \pm 2/3, \pm 1, \pm 4/3, \ldots\}$ as complex values in an image. Thus AMPLRES corresponds to the amplitude quantisation of a signal. The PHASERES of M is the total number (per 2π) of discrete evenly spaced phase values in M's images, and so PHASERES corresponds to the phase quantisation of a signal. For example, we would need PHASERES of 3 to directly store values from the set $\{e^{ix} \mid x \in \{0, 2/3\pi, 4/3\pi\}\}$ as complex values in an image. Finally, the FREQ complexity of a CSM M is the minimum optical frequency necessary for M to compute correctly, this concept is explained further in [100].

Often we wish to make analogies between space on some well-known model and CSM 'space-like' resources. Thus we define the following convenient term.

Definition 6 *The* SPACE *complexity of a CSM M is the product of the* GRID, SPATIALRES, DYRANGE, AMPLRES, PHASERES *and* FREQ *complexities of M.*

6.3. *Optical realisation*

In this section, we outline how some of the elementary operations of the CSM could be carried out physically. We do not intend to specify the definitive realisation of any of the operations, but simply convince the reader that the model's operations have physical interpretations. Furthermore, although we concentrate on implementations employing visible light (optical frequencies detectable to the human eye) the CSM definition does not preclude employing other portion(s) of the electromagnetic spectrum.

A complex-valued image could be represented physically by a spatially coherent optical wavefront. Spatially coherent illumination (light of a single wavelength and emitted with the same phase angle) can be produced by a laser. A SLM could be used to encode the image onto the expanded and collimated laser beam. One could write to a SLM offline (expose photographic film, or laser print or relief etch a transparency) or online (in the case of a liquid-crystal display [104,63,92] or holographic material [24,75]). The functions h and v could be effected using two convex cylindrical lenses, oriented horizontally and vertically, respectively [91,36,63,35].

A coherent optical wavefront will naturally evolve into its own Fourier spectrum as it propagates to infinity. What we do with a convex lens is simply image, at a finite distance, this spectrum. This finite distance is called the focal length of the lens. The constant θ used in the definitions of h and v could be effected using Fourier spectrum size reduction techniques [91,36] such as varying the focal length of the lens, varying the separation of the lens and SLM, employing cascaded Fourier transformation, increasing the dimensions/reducing the spatial resolution of the SLM, or using light with a shorter wavelength.

The function $*$ could be implemented using a phase conjugate mirror [25]. The function $\cdot$ could be realised by placing a SLM encoding an image f in the path of a wavefront encoding another image g [91,36,90]. The wavefront immediately behind the SLM would then be $\cdot(f, g)$. The function $+$ describes the superposition of two optical wavefronts. This could be achieved using a 50:50 beam splitter [91,25,93]. The function ρ could be implemented using an electronic camera or a liquid-crystal light valve [92]. The parameters z_{l} and z_{u} would then be physical characteristics of the particular camera/light valve used. z_{l} corresponds to the minimum intensity value that the device responds to, known as the dark current signal, and z_{u} corresponds to the maximum intensity (the saturation level).

A note will be made about the possibility of automating these operations. If suitable SLMs can be prepared with the appropriate 2D pattern(s), each of the operations h, v, $*$, $\cdot$, and $+$ could be effected autonomously and without user intervention using appropriately positioned lenses and free space propagation. The time to effect these operations would be the sum of the flight time of the image (distance divided by velocity of light) and the response time of the analog 2D detector; both of which are constants independent of the size or resolution of the images if an appropriate 2D detector is chosen. Examples of appropriate detectors would be holographic material [24,75] and a liquid-crystal light valve with a continuous (not pixellated) area [92]. Since these analog detectors are also optically-addressed SLMs, we can very easily arrange for the output of one function to act as the input to another, again in constant time independent of the size or resolution of the image. A set of angled mirrors will allow the optical image to be fed back to the first SLM

in the sequence, also in constant time. It is not known, however, if ρ can be carried out completely autonomously for arbitrary parameters. Setting arbitrary parameters might fundamentally require offline user intervention (adjusting the gain of the camera, and so on), but at least for a small range of values this can be simulated online using a pair of liquid-crystal intensity filters.

We have outlined some optics principles that could be employed to implement the operations of the model. The simplicity of the implementations hides some imperfections in our suggested realisations. For example, the implementation of the + operation outlined above results in an output image that has been unnecessarily multiplied by the constant factor 0.5 due to the operation of the beam splitter. Also, in our suggested technique, the output of the ρ function is squared unnecessarily. However, each of these effects can be compensated for with a more elaborate optical setup and/or at the algorithm design stage.

A more important issue concerns the quantum nature of light. According to our current understanding, light exists as individual packets called photons. As such, in order to physically realise the CSM one would have to modify it such that images would have discrete, instead of continuous, amplitudes. The atomic operations outlined above, in particular the Fourier transform, are not affected by the restriction to quantised amplitudes, as the many experiments with electron interference patterns indicate. We would still assume, however, that in the physical world space is continuous.

A final issue concerns how a theoretically infinite Fourier spectrum could be represented by an image (or encoded by a SLM) of finite extent. This difficulty is addressed with the FREQ complexity measure [100].

7. Example CSM datastructures and algorithms

In this section we give some example data representations. We then to go on to give an example CSM algorithm that efficiently squares a binary matrix.

7.1. *Representing data as images*

There are many ways to represent data as images and interesting new algorithms sometimes depend on a new data representation. Data representations should be in some sense reasonable, for example it is unreasonable that the input to an algorithm could (non-uniformly) encode solutions to NP-hard or even undecidable problems. From Section 8.1, the CSM address encoding function gives the programmer room to be creative, so long as the representation is logspace computable (assuming a reasonable representation of images as words).

Here we mention some data representations that are commonly used. Figures 3(a) and 3(b) are the binary symbol image representations of 1 and 0 respectively. These images have an everywhere constant value of 1 and 0 respectively, and both have SPATIALRES of 1. The row and column image representations of the word 1011 are respectively given in Figures 3(c) and 3(d). These row and column images both have SPATIALRES of 4. In the matrix image representation in Figure 3(e), the first matrix element is represented at the top left corner and elements are ordered in the usual matrix way. This 3×4 matrix

image has SPATIALRES of 12. Finally the binary stack image representation, which has exponential SPATIALRES of 16, is given in Figure 3(f).

Figure 2 shows how we might form a list image by copying four images to one in a single timestep. All of the above mentioned images have DYRANGE, AMPLRES and PHASERES of 1.

Another useful representation is where the value of a pixel directly encodes a number, in this case DYRANGE becomes crucial. We can also encode values as phase values, and naturally PHASERES becomes a useful measure of the resources needed to store such values.

7.2. *A matrix squaring algorithm*

Here we give an example CSM algorithm (taken from [96]) that makes use of the data representations described above. The algorithm squares a $n \times n$ matrix in $O(\log n)$ TIME and $O(n^3)$ SPATIALRES (number of pixels), while all other CSM resources are constant.

Lemma 1 *Let n be a power of 2 and let A be a $n \times n$ binary matrix. The matrix A^2 is computed by a $\mathcal{C}_2$-*CSM*, using the matrix image representation, in* TIME $O(\log n)$, SPATIALRES $O(n^3)$, GRID $O(1)$, DYRANGE $O(1)$, AMPLRES 1 *and* PHASERES 1.

Proof. *(Sketch)* In this proof the matrix and its matrix image representation (see Figure 3(e)) are both denoted A. We begin with some precomputation, then one parallel pointwise multiplication step, followed by $\log n$ additions to complete the algorithm.

We generate the matrix image A_1 that consists of n vertically juxtaposed copies of A. This is computed by placing one copy of A above the other, scaling to one image, and repeating to give a total of $\log n$ iterations. The image A_1 is constructed in TIME $O(\log n)$, GRID $O(1)$ and SPATIALRES $O(n^3)$.

Next we transpose A to the column image A_2. The first n elements of A_2 are row 1 of A, the second n elements of A_2 are row 2 of A, etc. This is computed in TIME $O(\log n)$, GRID $O(1)$ and SPATIALRES $O(n^2)$ as follows.

Let $A' = A$ and $i = n$. We horizontally split A' into a left image A'_L and a right image A'_R. Then A'_L is pointwise multiplied (or masked) by the column image that represents $(10)^i$, in TIME $O(1)$. Similarly A'_R is pointwise multiplied (or masked) by the column image that represents $(01)^i$. The masked images are added. The resulting image has half the number of columns as A' and double the number of rows, and for example: row 1 consists of the first half of the elements of row 1 of A' and row 2 consists of the latter half of the elements of row 1 of A'. We call the result A', and we double the value of i. We repeat the process to give a total of $\log n$ iterations. After these iterations the resulting column image is denoted A_2.

We pointwise multiply A_1 and A_2 to give A_3 in TIME $O(1)$, GRID $O(1)$ and SPATIALRES $O(n^3)$.

To facilitate a straightforward addition we first transpose A_3 in the following way: A_3 is vertically split into a bottom and a top image, the top image is placed to the left of the bottom and the two are scaled to a single image, this splitting and scaling is repeated to give a total of $\log n$ iterations and we call the result A_4. Then to perform the addition, we vertically split A_4 into a bottom and a top image. The top image is pointwise added to the bottom image and the result is thresholded between 0 and 1. This splitting, adding and thresholding is repeated a total of $\log n$ iterations to create A_5. We 'reverse'

the transposition that created A_4: image A_5 is horizontally split into a left and a right image, the left image is placed above the right and the two are scaled to a single image, this splitting and scaling is repeated a total of $\log n$ iterations to give A^2. □

The algorithm highlights a few points of interest about the CSM. The CSM has quite a number of space-like resources, and it is possible to have trade-offs between them. For example in the algorithm above, if we allow GRID to increase from $O(1)$ to $O(n)$ then the SPATIALRES can be reduced from $O(n^3)$ to $O(n^2)$. In terms of optical architectures modelled by the CSM this phenomenon could be potentially very useful as certain resources may well be more economically viable than others. The algorithm is used in the proof that that polynomial TIME CSMs (and $\mathcal{C}_2$-CSMs, see below) compute problems that are in the PSPACE class of languages. PSPACE includes the famous NP class. Such computational complexity results are discussed further in Section 9 below.

There are a number of existing CSM algorithms, for these we point the reader to the literature [64–66,94,96,98–100].

8. $\mathcal{C}_2$-CSM

In this section we define the $\mathcal{C}_2$-CSM. One of the motivations for this model is the need to put reasonable upper bounds on the power of reasonable optical computers. As we've shown elsewhere [97], it turns out that CSMs can very quickly use massive amounts of resources, and the $\mathcal{C}_2$-CSM definition is an attempt to define a more reasonable model, especially towards the goal of providing useful upper bounds on its power.

8.1. $\mathcal{C}_2$-CSM

Motivated by a desire to apply standard complexity theory tools to the model, we defined [94,97] the $\mathcal{C}_2$-CSM, a restricted and more realistic class of CSM.

Definition 7 ($\mathcal{C}_2$-CSM) *A $\mathcal{C}_2$-CSM is a CSM whose computation* TIME *is defined for $t \in \{1, 2, \ldots, T(n)\}$ and has the following restrictions:*
- *For all* TIME *t both* AMPLRES *and* PHASERES *have constant value of 2.*
- *For all* TIME *t each of* GRID, SPATIALRES *and* DYRANGE *is $2^{O(t)}$ and* SPACE *is redefined to be the product of all complexity measures except* TIME *and* FREQ.
- *Operations h and v compute the discrete Fourier transform in the horizontal and vertical directions respectively.*
- *Given some* reasonable *binary word representation of the set of addresses $\mathcal{N}$, the address encoding function $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ is decidable by a logspace Turing machine.*

Let us discuss these restrictions. The restrictions on AMPLRES and PHASERES imply that $\mathcal{C}_2$-CSM images are of the form $f : [0, 1) \times [0, 1) \to \{0, \pm\frac{1}{2}, \pm 1, \pm\frac{3}{2}, \ldots\}$. We have replaced the Fourier transform with the discrete Fourier transform [9], this essentially means that FREQ is now solely dependent on SPATIALRES; hence FREQ is not an interesting complexity measure for $\mathcal{C}_2$-CSMs and we do not analyse $\mathcal{C}_2$-CSMs in terms of FREQ complexity [94,97]. Restricting the growth of SPACE is not unique to our model, such restrictions are to be found elsewhere [34,70,74].

In Section 6.1 we stated that the address encoding function $\mathfrak{E}$ should be Turing machine decidable, here we strengthen this condition. At first glance sequential logspace

computability may perhaps seem like a strong restriction, but in fact it is quite weak. From an optical implementation point of view it should be the case that $\mathfrak{E}$ is not complicated, otherwise we cannot assume fast addressing. Other (sequential/parallel) models usually have a very restricted 'addressing function': in most cases it is simply the identity function on $\mathbb{N}$. Without an explicit or implicit restriction on the computational complexity of $\mathfrak{E}$, finding non-trivial upper bounds on the power of $\mathcal{C}_2$-CSMs is impossible as $\mathfrak{E}$ could encode an arbitrarily complex Turing machine. As a weaker restriction we could give a specific $\mathfrak{E}$. However, this restricts the generality of the model and prohibits the programmer from developing novel, reasonable, addressing schemes.

9. Optical computing and computational complexity

As we saw in Section 5, there are number of optical algorithms that use the inherent parallelism of optics to provide fast solutions to certain problems. An alternative approach is to ask the following question: How does a given optical model relate to standard sequential and parallel models? Establishing a relationship with computational complexity theory, by describing both upper and lower bounds on the model, gives immediate access to a large collection of useful algorithms and proof techniques.

The parallel computation thesis [32,23,22,50,88,70] states that parallel time (polynomially) corresponds to sequential space, for reasonable parallel and sequential models. An example would be the fact that the class of problems solvable in polynomial space on a number of parallel models is equivalent to PSPACE, the class of problems solvable on Turing machines that use at most polynomial space [41,8,31,33,22,34,89,86,84,3,85].

Of course the thesis can never be proved, it relates the intuitive notion of reasonable parallelism to the precise notion of a Turing machine. When results of this type were first shown researchers were suitably impressed; their parallel models truly had great power. For example if model M verifies the thesis then M decides PSPACE (including NP) languages in polynomial time. However there is another side to this coin. It is straightforward to verify that given our current best algorithms, M will use at least a superpolynomial amount of some other resource (like space or number of processors) to decide a PSPACE-complete or NP-complete language. Since the composition of polynomials is itself a polynomial, it follows that if we restrict the parallel computer to use at most polynomial time and polynomial other resources, then it can at most solve problems in P.

Nevertheless, asking if M verifies the thesis is an important question. Certain problems, such as those in the class NC, are efficiently parallelisable. NC can be defined as the class of problems that are solvable in polylogarithmic time on a parallel computer that uses a polynomial amount of hardware. So one can think of NC as those problems in P which are solved exponentially faster on parallel computation thesis models than on sequential models. If M verifies the thesis then it may be useful to apply M to these problems. We also know that if M verifies the thesis then there are (P-complete) problems for which it is widely believed that we will not find exponential speed-up using M.

9.1. $\mathcal{C}_2$-CSM *and parallel complexity theory*

Here we summarise some characterisations of the computing power of optical computers. Such characterisations enable the algorithm designer to know what kinds of problems are solvable with resource bounded optical algorithms.

Theorem 2 below gives lower bounds on the computational power of the $\mathcal{C}_2$-CSM by showing that it is at least as powerful as models that verify the parallel computation thesis.

Theorem 2 ([96,98]) NSPACE($S(n)$) $\subseteq$ $\mathcal{C}_2$-CSM–TIME($O(S^2(n))$)

In particular, polynomial TIME $\mathcal{C}_2$-CSMs accept the PSPACE languages. PSPACE is the class of problems solvable by Turing machines that use polynomial space, which includes the famous class NP, and so NP-complete problems can be solved by $\mathcal{C}_2$-CSMs in polynomial TIME. However, any $\mathcal{C}_2$-CSM algorithm that we could presently write to solve PSPACE or NP problems would require exponential SPACE.

Theorem 2 is established by giving a $\mathcal{C}_2$-CSM algorithm that efficiently generates, and squares, the transition matrix of a $S(n) = \Omega(\log n)$ space bounded Turing machine. This transition matrix represents all possible computations of the Turing machine and is of size $O(2^S) \times O(2^S)$. The matrix squaring part was already given as an example (Lemma 1), and the remainder of the algorithm is given in [96]. The algorithm uses SPACE that is cubic in one of the matrix dimensions. In particular the algorithm uses cubic SPATIALRES, $O(2^{3S})$, and all other space-like resources are constant. This theorem improves upon the time overhead of a previous similar result [94,98] that was established via $\mathcal{C}_2$-CSM simulation of the vector machines [73,74] of Pratt, Rabin, and Stockmeyer.

From the resource usage point of view, it is interesting to see that the older of these two algorithms uses GRID, DYRANGE, and SPATIALRES that are each $O(2^S)$, while the newer algorithm shows that if we allow more SPATIALRES we can in fact use only constant GRID and DYRANGE. It would be interesting to find other such resource trade-offs within the model.

Since NP is contained in PSPACE, Theorem 2, and the corresponding earlier results in [94,98], show that this optical model solves NP-complete problems in polynomial TIME. As described in Section 5, this has also been shown experimentally, for example Shaked et al. [81] have recently given a polynomial time, exponential space, optical algorithm to solve the NP-complete travelling salesperson problem. Their optical setup can be implemented on the CSM.

The other of the two inclusions that are necessary in order to verify the parallel computation thesis have also been shown: $\mathcal{C}_2$-CSMs computing in TIME $T(n)$ are no more powerful than $T^{O(1)}(n)$ space bounded deterministic Turing machines. More precisely, we have:

Theorem 3 ([94,95]) $\mathcal{C}_2$-CSM-TIME($T(n)$) $\subseteq$ DSPACE($O(T^2(n))$)

This result gives an upper bound on the power of $\mathcal{C}_2$-CSMs and was established via $\mathcal{C}_2$-CSM simulation by logspace uniform circuits of size and depth polynomial in SPACE and TIME respectively [95].

These computational complexity results for the $\mathcal{C}_2$-CSM have shown that the model is capable of parallel processing in much the same way as models that verify the parallel

computation thesis (and models that are known to characterise the parallel class NC). These results strongly depend on their use of non-constant SPATIALRES. The algorithms exploit the ability of optical computers, and the CSM in particular, to operate on large numbers of pixels in parallel. But what happens when we do not have arbitrary numbers of pixels? If allow images to have only a constant number of pixels then we need to find new CSM algorithms. It turns out that such machines also characterise PSPACE in polynomial TIME.

Theorem 4 ([101]) PSPACE *is characterised by* $\mathcal{C}_2$-CSM*s that are restricted to use polynomial* TIME $T = O(n^k)$, SPATIALRES $O(1)$, GRID $O(1)$, *and generalised to use* AMPLRES $O(2^{2^T})$, DYRANGE $O(2^{2^T})$.

So by treating images as registers and generating exponentially large, and exponentially small, values we can solve seemingly intractable problems. Of course this kind of CSM is quite unrealistic from the point of view of optical implementations. In particular, accurate multiplication of such values is difficult to implement in optics [101].

To restrict the model we could replace arbitrary multiplication, by multiplication by constants, which can be easily simulated by a constant number of additions. If we disallow multiplication in this way, we characterise P.

Theorem 5 ([101]) $\mathcal{C}_2$-CSM*s without multiplication, that compute in polynomial* TIME, *polynomial* GRID $O(n^k)$, *and* SPATIALRES $O(1)$, *characterise P.*

We can swap the roles of GRID and SPATIALRES, and still obtain a P characterisation:

Theorem 6 ([101]) *CSMs without multiplication, that compute in polynomial* TIME, *polynomial* SPATIALRES $O(n^k)$, *and* GRID $O(1)$, *characterise P.*

Theorems 5 and 6 give conditions under which our optical model essentially looses its parallel abilities and acts like a standard sequential Turing machine.

Via the proofs of Theorems 2 and 3 we can show that $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME solve exactly those problems in the class NC.

Corollary 7 $\mathcal{C}_2$-CSM-SPACE, TIME$(n^{O(1)}, \log^{O(1)} n) = \text{NC}$

Problems in NC highlight the power of parallelism, as these problems can be solved exponentially faster on a polynomial amount of parallel resources than on polynomial time sequential machines. As further work in this area one could try to find alternate characterisations of NC in terms of the $\mathcal{C}_2$-CSM. In particular, one could try to find further interesting trade-offs between the various space-like resources of the model. In the real world this would correspond to computing over various different optical resources. As discussed in Section 5 it would be interesting for optical algorithm designers to try to design (implementable) optical algorithms for NC problems in an effort to find problems that are well-suited to optical solutions.

References

[1] M. A. G. Abushagur and H. J. Caulfield. Speed and convergence of bimodal optical computers. *Optical Engineering*, 26(1):22–27, Jan. 1987.

[2] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, Nov. 1994.

[3] A. Alhazov and M. de Jesús Pérez-Jiménez. Uniform solution to QSAT using polarizationless active membranes. In J. Durand-Lose and M. Margenstern, editors, *Machines, Computations and Universality (MCU)*, volume 4664 of *LNCS*, pages 122–133, Orléans, France, Sept. 2007. Springer.

[4] H. H. Arsenault and Y. Sheng. *An Introduction to Optics in Computers*, volume TT8 of *Tutorial Texts in Optical Engineering*. SPIE Press, Bellingham, Washington, 1992.

[5] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity, vols I and II*. EATCS Monographs on Theoretical Computer Science. Springer, Berlin, 1988.

[6] R. Barakat and J. H. Reif. Lower bounds on the computational efficiency of optical computing systems. *Applied Optics*, 26(6):1015–1018, Mar. 1987.

[7] F. R. Beyette Jr., P. A. Mitkas, S. A. Feld, and C. W. Wilmsen. Bitonic sorting using an optoelectronic recirculating architecture. *Applied Optics*, 33(35):8164–8172, Dec. 1994.

[8] A. Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, Dec. 1977.

[9] R. N. Bracewell. *The Fourier transform and its applications*. Electrical and electronic engineering series. McGraw-Hill, second edition, 1978.

[10] K.-H. Brenner, A. Huang, and N. Streibl. Digital optical computing with symbolic substitution. *Applied Optics*, 25(18):3054–3060, Sept. 1986.

[11] D. P. Casasent and G. P. House. Comparison of coherent and noncoherent optical correlators. In *Optical Pattern Recognition V*, Proceedings of SPIE vol. 2237, pages 170–178, Apr. 1994.

[12] D. P. Casasent and D. Psaltis. Position, rotation, and scale invariant optical correlation. *Applied Optics*, 15(7):1795–1799, 1976.

[13] H. J. Caulfield, editor. *Handbook of Optical Holography*. Academic Press, New York, 1979.

[14] H. J. Caulfield. The energetic advantage of analog over digital computing. In *OSA Optical Computing Technical Digest Series*, volume 9, pages 180–183, 1989.

[15] H. J. Caulfield. Space-time complexity in optical computing. In B. Javidi, editor, *Optical information-processing systems and architectures II*, volume 1347, pages 566–572. SPIE, July 1990.

[16] H. J. Caulfield. Space-time complexity in optical computing. *Multidimensional Systems and Signal Processing*, 2(4):373–378, Nov. 1991. Special issue on optical signal processing.

[17] H. J. Caulfield and M. A. G. Abushagur. Hybrid analog-digital algebra processors. In *Optical and Hybrid Computing II*, Proceedings of SPIE vol. 634, pages 86–95, Orlando, Florida, Apr. 1986.

[18] H. J. Caulfield and R. Haimes. Generalized matched filtering. *Applied Optics*, 19(2):181–183, Jan. 1980.

[19] H. J. Caulfield, S. Horvitz, and W. A. V. Winkle. Introduction to the special issue on optical computing. *Proceedings of the IEEE*, 65(1):4–5, Jan. 1977.

[20] H. J. Caulfield, J. M. Kinser, and S. K. Rogers. Optical neural networks. *Proceedings of the IEEE*, 77:1573–1582, 1989.

[21] H. J. Caulfield, W. T. Rhodes, M. J. Foster, and S. Horvitz. Optical implementation of systolic array processing. *Optics Communications*, 40:86–90, 1981.

[22] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, Jan. 1981.

[23] A. K. Chandra and L. J. Stockmeyer. Alternation. In *17th annual symposium on Foundations of Computer Science*, pages 98–108, Houston, Texas, Oct. 1976. IEEE. (Preliminary Version).

[24] F. S. Chen, J. T. LaMacchia, and D. B. Fraser. Holographic storage in lithium niobate. *Applied Physics Letters*, 13(7):223–225, oct 1968.

[25] A. E. Chiou. Photorefractive phase-conjugate optics for image processing, trapping, and manipulation of microscopic objects. *Proceedings of the IEEE*, 87(12):2074–2085, Dec. 1999.

[26] N. Collings, R. Sumi, K. J. Weible, B. Acklin, and W. Xue, The use of optical hardware to find good solutions of the travelling salesman problem (TSP). In *Optical Computing*, Proceedings of SPIE vol. 1806, pages 637–641, 1992.

[27] S. Dolev and H. Fitoussi. The traveling beam: optical solution for bounded NP-complete problems. In P. Crescenzi, G. Prencipe, and G. Pucci, editors, *The fourth international conference on fun with algorithms (FUN)*, pages 120–134, 2007.

[28] J. Durand-Lose. Reversible conservative rational abstract geometrical computation is Turing-universal. In *Logical Approaches to Computational Barriers, Second Conference on Computability in Europe, (CiE)*, volume 3988 of *Lecture Notes in Computer Science*, pages 163–172, Swansea, UK, 2006. Springer.

[29] N. H. Farhat and D. Psaltis. New approach to optical information processing based on the Hopfield model. *Journal of the Optical Society of America A*, 1:1296, 1984.

[30] D. G. Feitelson. *Optical Computing: A survey for computer scientists*. MIT Press, Cambridge, Massachusetts, 1988.

[31] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proc. 10th Annual ACM Symposium on Theory of Computing*, pages 114–118, 1978.

[32] L. M. Goldschlager. *Synchronous parallel computation*. PhD thesis, University of Toronto, Computer Science Department, Dec. 1977.

[33] L. M. Goldschlager. A unified approach to models of synchronous parallel machines. In *Proc. 10th Annual ACM Symposium on Theory of Computing*, pages 89–94, 1978.

[34] L. M. Goldschlager. A universal interconnection pattern for parallel computers. *Journal of the ACM*, 29(4):1073–1086, Oct. 1982.

[35] J. W. Goodman. Operations achievable with coherent optical information processing systems. *Proceedings of the IEEE*, 65(1):29–38, Jan. 1977.

[36] J. W. Goodman. *Introduction to Fourier Optics*. Roberts & Company, Englewood, Colorado, third edition, 2005.

[37] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford university Press, Oxford, 1995.

[38] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proc. 28th Annual ACM Symposium on Theory of Computing*, pages 212–219, May 1996.

[39] P. S. Guilfoyle, J. M. Hessenbruch, and R. V. Stone. Free-space interconnects for high-performance optoelectronic switching. *IEEE Computer*, 31(2):69–75, Feb. 1998.

[40] T. Haist and W. Osten. An optical solution for the travelling salesman problem. *Optics Express*, 15(16):10473–10482, Aug. 2007. Erratum: Vol. 15(10), pp 12627. Aug. 2007.

[41] J. Hartmanis and J. Simon. On the power of multiplication in random access machines. In *Proceedings of the 15th annual symposium on switching and automata theory*, pages 13–23, The University of New Orleans, Oct. 1974. IEEE.

[42] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49(6):737–759, 1987.

[43] J. L. Horner, editor. *Optical Signal Processing*. Academic Press, San Diego, 1987.

[44] Y.-N. Hsu and H. H. Arsenault. Optical pattern recognition using circular harmonic expansion. *Applied Optics*, 21(22):4016–4019, Nov. 1982.

[45] A. Huang. Architectural considerations involved in the design of an optical digital computer. *Proceedings of the IEEE*, 72(7):780–786, July 1984.

[46] A. D. Jacobson, T. D. Beard, W. P. Bleha, J. D. Morgerum, and S. Y. Wong. The liquid crystal light value, an optical-to-optical interface device. In *Proceedings of the Conference on Parallel Image Processing, Goddard Space Flight Center*, pages 288–299, Mar. 1972. Document X-711-72-308.

[47] B. Javidi. Nonlinear joint power spectrum based optical correlation. *Applied Optics*, 28(12):2358–2367, June 1989.

[48] B. Javidi and J. Wang. Optimum distortion-invariant filter for detecting a noisy distorted target in nonoverlapping background noise. *Journal of the Optical Society of America A*, 12(12):2604–2614, Dec. 1995.

[49] M. A. Karim and A. A. S. Awwal. *Optical Computing: An Introduction*. Wiley, New York, 1992.

[50] R. M. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 17. Elsevier, Amsterdam, 1990.

[51] E. Knill, R. LaFlamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, 2001.

[52] J. N. Lee, editor. *Design Issues in Optical Processing*. Cambridge Studies in Modern Optics. Cambridge University Press, Cambridge, Great Britain, 1995.

[53] J. N. Lee, editor. *Design issues in optical processing*. Cambridge studies in modern optics. Cambridge University Press, 1995.
[54] Lenslet Labs. Enlight256. white paper report, Lenslet Ltd., 6 Galgalei Haplada St, Herzelia Pituach, 46733 Israel, Nov. 2003.
[55] R. J. Lipton. Using DNA to solve NP-complete problems. *Science*, 268:542–545, Apr. 1995.
[56] A. Louri and A. Post. Complexity analysis of optical-computing paradigms. *Applied Optics*, 31(26):5568–5583, Sept. 1992.
[57] P. D. MacKenzie and V. Ramachandran. Ercw prams and optical communication. *Theoretical Computer Science*, 196:153–180, 1998.
[58] A. D. McAulay. *Optical Computer Architectures: The Application of Optical Concepts to Next Generation Computers*. Wiley, New York, 1991.
[59] C. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, Reading, Massachusetts, 1989.
[60] D. A. Miller. Rationale and challenges for optical interconnects to electronic chips. *Proceedings of the IEEE*, 88(6):728–749, June 2000.
[61] C. Moore. Generalized shifts: undecidability and unpredictability in dynamical systems. *Nonlinearity*, 4:199–230, 1991.
[62] C. Moore. Majority-vote cellular automata, Ising dynamics and P-completeness. *Journal of Statistical Physics*, 88(3/4):795–805, 1997.
[63] T. Naughton, Z. Javadpour, J. Keating, M. Klíma, and J. Rott. General-purpose acousto-optic connectionist processor. *Optical Engineering*, 38(7):1170–1177, July 1999.
[64] T. J. Naughton. Continuous-space model of computation is Turing universal. In S. Bains and L. J. Irakliotis, editors, *Critical Technologies for the Future of Computing*, Proceedings of SPIE vol. 4109, pages 121–128, San Diego, California, Aug. 2000.
[65] T. J. Naughton. A model of computation for Fourier optical processors. In R. A. Lessard and T. Galstian, editors, *Optics in Computing 2000*, Proc. SPIE vol. 4089, pages 24–34, Quebec, Canada, June 2000.
[66] T. J. Naughton and D. Woods. On the computational power of a continuous-space optical model of computation. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations and Universality: Third International Conference (MCU'01)*, volume 2055 of *LNCS*, pages 288–299, Chişinău, Moldova, May 2001. Springer.
[67] M. Oltean. A light-based device for solving the Hamiltonian path problem. In *Fifth International Conference on Unconventional Computation (UC'06)*, volume 4135 of *LNCS*, pages 217–227, York, UK, 2006. Springer.
[68] E. L. O'Neill. Spatial filtering in optics. *IRE Transactions on Information Theory*, 2:56–65, June 1956.
[69] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1995.
[70] I. Parberry. *Parallel complexity theory*. Wiley, 1987.
[71] G. Păun. *Membrane computing: an introduction*. Springer, 2002.
[72] A. Pe'er, D. Wang, A. W. Lohmann, and A. A. Friesem. Optical correlation with totally incoherent light. *Optics Letters*, 24(21):1469–1471, Nov. 1999.
[73] V. R. Pratt, M. O. Rabin, and L. J. Stockmeyer. A characterisation of the power of vector machines. In *Proc. 6th annual ACM symposium on theory of computing*, pages 122–134. ACM press, 1974.
[74] V. R. Pratt and L. J. Stockmeyer. A characterisation of the power of vector machines. *Journal of Computer and Systems Sciences*, 12:198–221, 1976.
[75] A. Pu, R. F. Denkewalter, and D. Psaltis. Real-time vehicle navigation using a holographic memory. *Optical Engineering*, 36(10):2737–2746, Oct. 1997.
[76] J. Reif, D. Tygar, and A. Yoshida. The computability and complexity of optical beam tracing. In *31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 106–114, St. Louis, MO, Oct. 1990. IEEE.
[77] J. H. Reif and A. Tyagi. Energy complexity of optical computations. In *2nd IEEE Symposium on Parallel and Distributed Processing*, pages 14–21, Dallas, TX, Dec. 1990.
[78] J. H. Reif and A. Tyagi. Efficient parallel algorithms for optical computing with the discrete Fourier transform (DFT) primitive. *Applied Optics*, 36(29):7327–7340, Oct. 1997.
[79] A. A. Sawchuk and T. C. Strand. Digital optical computing. *Proceedings of the IEEE*, 72(7):758–779, July 1984.
[80] N. T. Shaked, S. Messika, S. Dolev, and J. Rosen. Optical solution for bounded NP-complete problems. *Applied Optics*, 46(5):711–724, Feb. 2007.

[81] N. T. Shaked, G. Simon, T. Tabib, S. Mesika, S. Dolev, and J. Rosen. Optical processor for solving the traveling salesman problem (TSP). In B. Javidi, D. Psaltis, and H. J. Caulfield, editors, *Proc. of SPIE, Optical Information Systems IV*, volume 63110G, Aug. 2006.

[82] N. T. Shaked, T. Tabib, G. Simon, S. Messika, S. Dolev, and J. Rosen. Optical binary-matrix synthesis for solving bounded NP-complete combinatorical problems. *Optical Engineering*, 46(10):108201–1–108201–11, Oct. 2007.

[83] P. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. *In Proceedings 35th Annual Symposium on Foundations Computer Science*, pages 124–134, 1994.

[84] P. Sosík. The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing*, 2(3):287–298, 2003.

[85] P. Sosík and A. Rodríguez-Patón. Membrane computing and complexity theory: A characterization of PSPACE. *Journal of Computer and System Sciences*, 73(1):137–152, 2007.

[86] J. Tromp and P. van Emde Boas. Associative storage modification machines. In K. Ambos-Spies, S. Homer, and U. Schöning, editors, *Complexity theory: current research*, pages 291–313. Cambridge University Press, 1993.

[87] G. L. Turin. An introduction to matched filters. *IRE Transactions on Information Theory*, 6(3):311–329, June 1960.

[88] P. van Emde Boas. Machine models and simulations. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 1. Elsevier, Amsterdam, 1990.

[89] J. van Leeuwen and J. Wiedermann. Array processing machines. *BIT*, 27:25–43, 1987.

[90] A. VanderLugt. Signal detection by complex spatial filtering. *IEEE Transactions on Information Theory*, 10(2):139–145, Apr. 1964.

[91] A. VanderLugt. *Optical Signal Processing*. Wiley, New York, 1992.

[92] P.-Y. Wang and M. Saffman. Selecting optical patterns with spatial phase modulation. *Optics Letters*, 24(16):1118–1120, Aug. 1999.

[93] C. S. Weaver and J. W. Goodman. A technique for optically convolving two functions. *Applied Optics*, 5(7):1248–1249, July 1966.

[94] D. Woods. *Computational complexity of an optical model of computation*. PhD thesis, National University of Ireland, Maynooth, 2005.

[95] D. Woods. Upper bounds on the computational power of an optical model of computation. In X. Deng and D. Du, editors, *16th International Symposium on Algorithms and Computation (ISAAC 2005)*, volume 3827 of *LNCS*, pages 777–788, Sanya, China, Dec. 2005. Springer.

[96] D. Woods. Optical computing and computational complexity. In *Fifth International Conference on Unconventional Computation (UC'06)*, volume 4135 of *LNCS*, pages 27–40, York, UK, 2006. Springer. Invited.

[97] D. Woods and J. P. Gibson. Complexity of continuous space machine operations. In S. B. Cooper, B. Löewe, and L. Torenvliet, editors, *New Computational Paradigms, First Conference on Computability in Europe (CiE 2005)*, volume 3526 of *LNCS*, pages 540–551, Amsterdam, June 2005. Springer.

[98] D. Woods and J. P. Gibson. Lower bounds on the computational power of an optical model of computation. In C. S. Calude, M. J. Dinneen, G. Păun, M. J. Pérez-Jiménez, and G. Rozenberg, editors, *Fourth International Conference on Unconventional Computation (UC'05)*, volume 3699 of *LNCS*, pages 237–250, Sevilla, Oct. 2005. Springer.

[99] D. Woods and J. P. Gibson. Lower bounds on the computational power of an optical model of computation. *Natural Computing*, 7(1):95–108, Mar. 2007.

[100] D. Woods and T. J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1-3):227–258, Apr. 2005.

[101] D. Woods and T. J. Naughton. Sequential and parallel optical computing. In S. Dolev, T. Haist, and M. Oltean, editors, *International Workshop on Optical SuperComputing*, volume 5172 of *LNCS*, pages 70–86, Vienna, Aug. 2008. Springer.

[102] T. Yokomori. Molecular computing paradigm – toward freedom from Turing's charm. *Natural computing*, 1(4):333–390, 2002.

[103] F. T. S. Yu, S. Jutamulia, and S. Yin, editors. *Introduction to information optics*. Academic Press, San Diego, 2001.

[104] F. T. S. Yu, T. Lu, X. Yang, and D. A. Gregory. Optical neural network with pocket-sized liquid-crystal televisions. *Optics Letters*, 15(15):863–865, Aug. 1990.