Parallel and sequential optical computing*

Damien Woods[1] and Thomas J. Naughton[2,3]

[1] Department of Computer Science and Artificial Intelligence,
University of Seville, Spain
d.woods@cs.ucc.ie
[2] Department of Computer Science
National University of Ireland, Maynooth, County Kildare, Ireland
[3] University of Oulu, RFMedia Laboratory, Oulu Southern Institute, Vierimaantie 5,
84100 Ylivieska, Finland
tomn@cs.nuim.ie

Abstract. We present a number of computational complexity results for an optical model of computation called the continuous space machine. We also describe an implementation for an optical computing algorithm that can be easily defined within the model. Our optical model is designed to model a wide class of optical computers, such as matrix vector multipliers and pattern recognition architectures. It is known that the model solves intractable PSPACE problems in polynomial time, and NC problems in polylogarithmic time. Both of these results use large spatial resolution (number of pixels). Here we look at what happens when we have constant spatial resolution. It turns out that we obtain similar results by exploiting other resources, such as dynamic range and amplitude resolution. However, with certain other restrictions we essentially have a sequential device. Thus we are exploring the border between parallel and sequential computation in optical computing. We describe an optical architecture for the unordered search problem of finding a one in a list of zeros. We argue that our algorithm scales well, and is relatively straightforward to implement. This problem is easily parallelisable and is from the class NC. We go on to argue that the optical computing community should focus their attention on problems within P (and especially NC), rather than developing systems for tackling intractable problems.

1 Introduction

Over the years, optical computers were designed and built to emulate conventional microprocessors (digital optical computing), and for image processing over continuous wavefronts (analog optical computing). Here we are interested in the latter class: optical computers that store data as images. Numerous physical implementations exist and example applications include fast pattern recognition and matrix-vector algebra. There have been much resources devoted to designs,

* DW acknowledges support from Junta de Andalucía grant TIC-581. TN acknowledges support from a Marie Curie Fellowship through the European Commission Framework Programme 6.

implementations and algorithms for such optical information processing architectures (for example see [1, 8, 11, 19, 29] and their references).

We investigate the computational complexity of a model of computation that is inspired by such optical computers. The model is relatively new and is called the continuous space machine (CSM). The model was originally proposed by Naughton [17, 18]. 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. We analyse the model in terms of seven complexity measures inspired by real-world resources.

For the original [18] CSM definition, it was shown [17] that the CSM can simulate Turing machines (this was a sequential simulation). A less restricted CSM definition [20, 35] was shown to be too general for proving reasonable upper bounds on its computational power [33], so in this paper we mostly focus on computational complexity results for a restricted CSM called the $\mathcal{C}_2$-CSM.

In Section 2 we recall the definition of the model, including a number of optically-inspired complexity measures [35]. In Section 2.5 we describe a number of known computational complexity results for the model, including characterisations of PSPACE and NC. These results were shown a few years ago [30, 34] (later improved [32]), and were the first to prove that optical computers were capable of solving NP-complete (and other intractable) problems in polynomial time. Of course, these results make use of exponential space-like resources. In particular, these algorithms used exponential *spatial resolution* (number of pixels). Since we have a clear model definition, including definitions of relevant optical resources, it is relatively easy to analyse CSM algorithms to determine their resource usage. Recently, Shaked et al. [25–27] have designed an optical system for solving the NP-hard travelling salesman problem in polynomial time. Their algorithm can be seen as a special case of our results. Interestingly, they give both implementations and simulations. As we argue below, we believe that tackling intractable problems is probably not going to really highlight any advantages of optics over digital electronic systems. As a step in another direction, we have shown that if we restrict ourselves to using polylogarithmic time, and polynomial space-like resources, then parallel optical systems can solve exactly those problems that lie in the (parallel) class NC.

In Section 3 we present a number of new results for our model. In particular we look at what happens when spatial resolution is constant. Parallel optical algorithms and experimental setups usually exploit the fact that we can operate over many pixels in constant time. However, we show that even with a constant number of pixels we can solve problems in (and characterise) presumed intractable classes such as PSPACE, in polynomial time. In this case we make exponential usage of other resources, namely amplitude resolution and dynamic range. We argue that this is an even more unrealistic method of optical computing than using exponential numbers of pixels. We go on to show that if we disallow image multiplication, restrict to polynomial numbers of pixels and/or

images, but put no restrictions on the other resources, then in polynomial time the model characterises P.

This results lead us to suggest of a new direction for optical algorithm designers. Rather than trying to solve intractable problems, perhaps the community should focus its attention on problems that are known to be easily parallelisable, for example those in NC. Of course, these problems are polynomial time solvable on sequential machines. However, using our NC characterisations one can see that optics has the potential to solve such problems exponentially faster than sequential computers. Also, due to relatively low communication costs and high fan-in, optics has the potential to out-perform parallel digital electronic architectures. Perhaps such benefits of optics will be seen where very large datasets and input instances are concerned. We give evidence for this opinion by citing existing optical algorithms, as well as the following result in this paper.

We design an optoelectronic implementation for the unordered search problem of finding a single one in a list of zeros. Of course, this problem can be sequentially solved in $n-1$ steps. Our algorithm works in $O(\log n)$ time but, most importantly, we get this low time overhead on an optical set-up that scales well (uses at most n pixels), and is relatively straightforward to build. As we discuss in Section 4.1, this problem is contained in some of the lowest classes within NC.

2 CSM and $\mathcal{C}_2$-CSM

We begin by describing the model in its most general sense, this brief overview is not intended to be complete and more details are to be found in [30].

2.1 CSM

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 $\mathbb{N} \times \mathbb{N}$.

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 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

$\mathrm{h}(i_1;i_2)$	: 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 3.

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.

may grow in size as the computation progresses. Address *sta* is the start location for the program so the programmer should write the first program instruction (beginning) at *sta*. Addresses a and b define special images that are frequently used by some program instructions.

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 [30, 35]. We refrain from giving a description of this programming language and instead describe a less cumbersome high-level language [30]. Figure 1 gives the basic instructions of this high-level language. The copy instruction is illustrated in Figure 3. There are also **if/else** and **while** control flow instructions with conditions of the form $(f_\psi == f_\phi)$ where f_ψ and f_ϕ are *binary symbol images* (see Figures 2(a) and 2(b)).

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. Thus we insist that 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. For example, we put a restriction of logspace computability on $\mathfrak{E}$ in Definition 7 below. Configurations are 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.

2.2 Complexity measures

Next we define some CSM complexity measures. All resource bounding functions map from $\mathbb{N}$ into $\mathbb{N}$ and are assumed to have the usual properties [2].

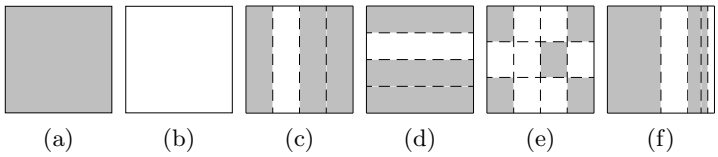

Fig. 2. Representing binary data. The shaded areas denote value 1 and the white areas denote value 0. (a) Binary symbol image representation of 1 and (b) of 0, (c) list (or row) image representation of the word 1011, (d) column image representation of 1011, (e) 3×4 matrix image, (f) binary stack image representation of 1101. Dashed lines are for illustration purposes only.

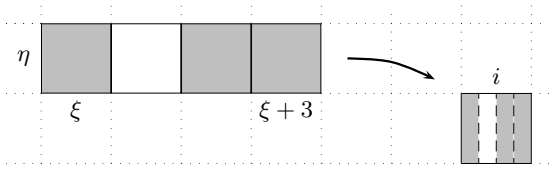

Fig. 3. Illustration of the instruction $i \leftarrow [\xi, \xi + 3, \eta, \eta]$ that copies four images to a single image that is denoted i.

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.*

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 $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \rightarrow \mathcal{I}$, 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 spatial light modulator.

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 [30, 35]. 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 the range of M's images. The phaseRes of M is the total number (per 2π) of discrete evenly spaced phase values in the range of M's images. freq is a measure of the optical frequency of M's images [35].

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 all of M's complexity measures except* time.

2.3 Representing data as images

There are many ways to represent data as images. Here we mention some data representations that we have used in previous results. Figures 2(a) and 2(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 2(c) and 2(d). These row and column images both have spatialRes of 4. In the matrix image representation in Figure 2(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 2(f).

Figure 3 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.

2.4 $\mathcal{C}_2$-CSM

Motivated by a desire to apply standard complexity theory tools to the model, we defined [30, 33] the $\mathcal{C}_2$-CSM, a restricted 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 $O(2^t)$ 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 (DFT) 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 $f : [0,1)\times[0,1) \rightarrow \{0, \pm\frac{1}{2}, \pm 1, \pm\frac{3}{2}, \ldots\}$. We have replaced the Fourier transform with the DFT [5], 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 [30, 33]. Restricting the growth of SPACE is not unique to our model, such restrictions are to be found elsewhere [10, 21, 22]. The condition on the address encoding function $\mathfrak{E}$ amounts to enforcing uniformity (we do not wish to use $\mathfrak{E}$ as a powerful oracle).

In this paper we prove results for variants (generalisations and restrictions) on the $\mathcal{C}_2$-CSM model. If we are not stating results for the $\mathcal{C}_2$-CSM itself, then we always specify the exact model that we are using.

2.5 Some existing $\mathcal{C}_2$-CSM complexity results

We have given 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 [30, 32, 34]. This thesis [7, 9] states that parallel time corresponds, within a polynomial, to sequential space for reasonable parallel models. See, for example, [12, 14, 21, 28] for details. Let $S(n)$ be a space bound that is $\Omega(\log n)$. The languages accepted by nondeterministic Turing machines in $S(n)$ space are accepted by $\mathcal{C}_2$-CSMs computing in polynomial TIME $O(S^2(n))$ (see [32] for this result, which improves on the version in [30, 34]):

Theorem 1. $\text{NSPACE}(S(n)) \subseteq \mathcal{C}_2\text{-CSM–TIME}(O(S^2(n)))$

For example, polynomial TIME $\mathcal{C}_2$-CSMs accept the PSPACE languages[4]. Of course any polynomial TIME $\mathcal{C}_2$-CSM algorithm that we could presently write to solve PSPACE-complete, or NP-complete, problems would require exponential SPACE. Theorem 1 is established using an implementation of a well-known transitive closure algorithm on the $\mathcal{C}_2$-CSM. Using this result, we also find that $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME accept the class NC [30, 34].

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

We have also given the other of the two inclusions that are necessary in order to verify the parallel computation thesis: $\mathcal{C}_2$-CSMs computing in TIME $T(n)$ are no more powerful than $O(T^2(n))$ space bounded deterministic Turing machines [30, 31].

Theorem 2. $\mathcal{C}_2\text{-CSM-TIME}(T(n)) \subseteq \text{DSPACE}(O(T^2(n)))$

Via the proof of Theorem 2, we get another result. $\mathcal{C}_2$-CSMs that simultaneously use polynomial SPACE and polylogarithmic TIME accept at most NC [30, 31].

[4] PSPACE is a well-known class of problems that are solvable by Turing machines that use space polynomial in input length n. This class contains NP, since a polynomial space bounded Turing machine can simulate, in turn, each of the exponentially many possible computation paths of a nondeterministic polynomial time Turing machine.

Corollary 2. $\mathcal{C}_2$-CSM-SPACE, TIME$(n^{O(1)}, \log^{O(1)} n) \subseteq$ NC

The latter two inclusions are established via $\mathcal{C}_2$-CSM simulation by logspace uniform circuits of size and depth polynomial in SPACE and TIME respectively. Thus $\mathcal{C}_2$-CSMs that simultaneously use both polynomial SPACE and polylogarithmic TIME characterise NC.

3 Parallel and sequential $\mathcal{C}_2$-CSM computation

As we have seen in the previous section, a number of 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. To date, these results strongly depended 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 that such machines characterise PSPACE.

Theorem 3. 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 $O(2^{2^T})$*.*

Proof. The PSPACE upper bound comes directly from a minor extension to the proof of Theorem 2, sketched as follows. The proof of Theorem 2 showed that $\mathcal{C}_2$-CSMs that run in polynomial TIME $T = O(n^k)$, are simulated by circuits of polynomial depth $O(T^2)$ and size exponential in T, and it remains to be shown that our AMPLRES and DYRANGE generalisations do not affect these circuit bounds by more than a polynomial factor. In the previous proof [30, 31] DYRANGE was $O(2^T)$, in accordance with the usual $\mathcal{C}_2$-CSM definition. Thus, in the circuit simulation, images values $x \in \{1, \ldots, O(2^T)\} \subseteq \mathbb{N}$ were represented by binary words $\hat{x}$ of length $|\hat{x}| = O(T)$. We directly apply the previous construction to represent values $x \in \{1, \ldots, O(2^{2^T})\}$ as words of length $|\hat{x}| = O(2^T)$. Since the circuits are already of size exponential in T, this modification only increases circuit size by a polynomial factor in the existing size. Also, the circuit simulation algorithms experience at most a polynomial factor increase in their depth. A similar argument works for AMPLRES (even though in the previous proof AMPLRES was $O(1)$). Here we are using constant SPATIALRES and GRID (as opposed to $O(2^T)$ for the previous proof), so circuit size and depth are each decreased by a polynomial factor in their respective previous values. We omit the details.

For the lower bound we use the results of Schönhage [24] and Bertoni et al. [4] which show that PSPACE is characterised by RAMs augmented with integer addition, multiplication, and left shift instructions, that run in time that is polynomial in input length n. We show how to simulate such an augmented RAM with a $\mathcal{C}_2$-CSM that has time overhead that is polynomial in RAM time.

The numerical value $x \in \mathbb{N}$ of the binary word in a RAM register is stored as an image, with a single pixel, of value x. The RAM uses a constant (independent of input length n) number of registers, and therefore the $\mathcal{C}_2$-CSM uses a constant number of images. The addition and multiplication RAM operations are trivially simulated in constant time by $\mathcal{C}_2$-CSM addition and multiplication instructions.

The RAM shift instruction $x \leftarrow y$ takes a register x containing a binary value and shifts it to the right by an amount stored in another binary register y (Schönhage defines the shift instruction as $\lfloor x/2^y \rfloor$). In the $\mathcal{C}_2$-CSM this can be simulated (using multiplication and addition) by $(x \cdot 1/2^y) + (-1 \cdot x')$ where x' is the result of (thresholding and multiplication) $\rho(x \cdot 1/2^y, 1/2^y, 1; x')$, and the value $1/2^y$ is computed by repeated multiplication in $O(\log y)$ steps.

The $\mathcal{C}_2$-CSM algorithm uses AMPLRES and DYRANGE that are exponential in the space used by the RAM and TIME polynomial in the time of the RAM. All other resources are constant. □

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 in optics is difficult to implement. Some systems have up to a few hundred distinct amplitude levels [8] (for example 8 bits when we have 256×256 pixels [15], although higher accuracy is possible when we have a single pixel[5]). Therefore, one could argue that this kind of multiplication is quite unrealistic. 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 4. *$\mathcal{C}_2$-CSMs without multiplication, that compute in polynomial* TIME*, polynomial* GRID $O(n^k)$*, and* SPATIALRES $O(1)$*, characterise P.*

Proof (Sketch). For the P lower bound assume that we wish to simulate a deterministic Turing machine with one-way infinite tapes. Each tape is represented as a row of images, one for each tape cell. We store a pointer to the current tape head position as an image address. Then it is a straightforward matter to convert the Turing machine program to CSM program, where a left (right) move corresponds to decrementing (incrementing) the address pointer. Reading and writing to the tape is simulated by copying images. CSM branching instructions simulate branching in the Turing machine program. The CSM runs in TIME that is linear in Turing machine time.

For the P upper bound we assume some representation of images as binary words (such as the representation given above, or in [30, 31]), and apply a simple inductive argument. The initial configuration of our restricted $\mathcal{C}_2$-CSM is encoded by a binary word of length polynomial in the input length n. Assume at

[5] Inexpensive off-the-shelf single point intensity detectors have intensity resolutions of at least 24 bits (see, for example, the specifications for silicon-based point detectors and optical power meters from popular manufacturers such as www.mellesgriot.com, www.newport.com, and www.thorlabs.com).

$\mathcal{C}_2$-CSM TIME t that the binary word encoding the configuration is of polynomial length. For each pixel, addition of two images leads to an increase of at most one bit per pixel under our representation, and can be simulated in polynomial time on a Turing machine. The DFT over a finite field is computable in polynomial time and is defined in such a way that it does not increase the number of pixels in an image. Also, its input and output values are from the same set, therefore the upper bounds on the other space-like resources are unaffected by the DFT. Copying (up to) a polynomial number of encoded images can be computed in polynomial time. It is straightforward to simulate complex conjugation and thresholding in linear time. □

The first proof of universality for the CSM was a simulation of Turing machines that used SPACE that is exponential in Turing machine space [17]. Specifically, it used constant GRID and exponential SPATIALRES. The previous theorem improves the SPACE bound to linear, by using linear GRID and only constant SPATIALRES.

If we take the previous restricted $\mathcal{C}_2$-CSM, and restrict further to allow only constant GRID, but allow ourselves polynomial SPATIALRES, then we also characterise P.

Theorem 5. *CSMs without multiplication, that compute in polynomial* TIME*, polynomial* SPATIALRES $O(n^k)$*, and* GRID $O(1)$*, characterise P.*

Proof (Sketch). Here we are considering a $\mathcal{C}_2$-CSM model that is similar to the model in Theorem 4; we are swapping GRID for SPATIALRES. Hence a very similar technique can be used to show the P upper bound, so we omit the details.

For the lower bound, we store each one-way, polynomial $p(n)$ length, binary, Turing machine tape as a binary list image. We store the current tape head position $i \in \{1, \ldots, p(n)\}$ as a binary list image that represents i in binary. Then, to extract the bit stored at position i, we can use a $O(\log p(n))$ TIME binary search algorithm (split tape image in two, if $i \leqslant p(n)/2$ then take the left image, otherwise take the right, let $p(n) := p(n)/2$ and repeat). This technique, along with suitable masks, can also be applied to write to the tape. The Turing machine program is simulated using $\mathcal{C}_2$-CSM branching instructions. □

Theorems 4 and 5 give conditions under which our optical model essentially looses its parallel abilities and acts like a standard sequential Turing machine.

4 Implementation of an unordered search algorithm

We provide a design for an optoelectronic implementation of a binary search algorithm that can be applied to unordered lists. Consider an unordered list of n elements. For a given property P, the list could be represented by an n-tuple of bits, where the bit key for each element denotes whether or not that element satisfies P. If, for a particular P, only one element in the list satisfies P, the problem of finding its index becomes one of searching an unordered binary list for a single 1. The problem is defined formally as follows.

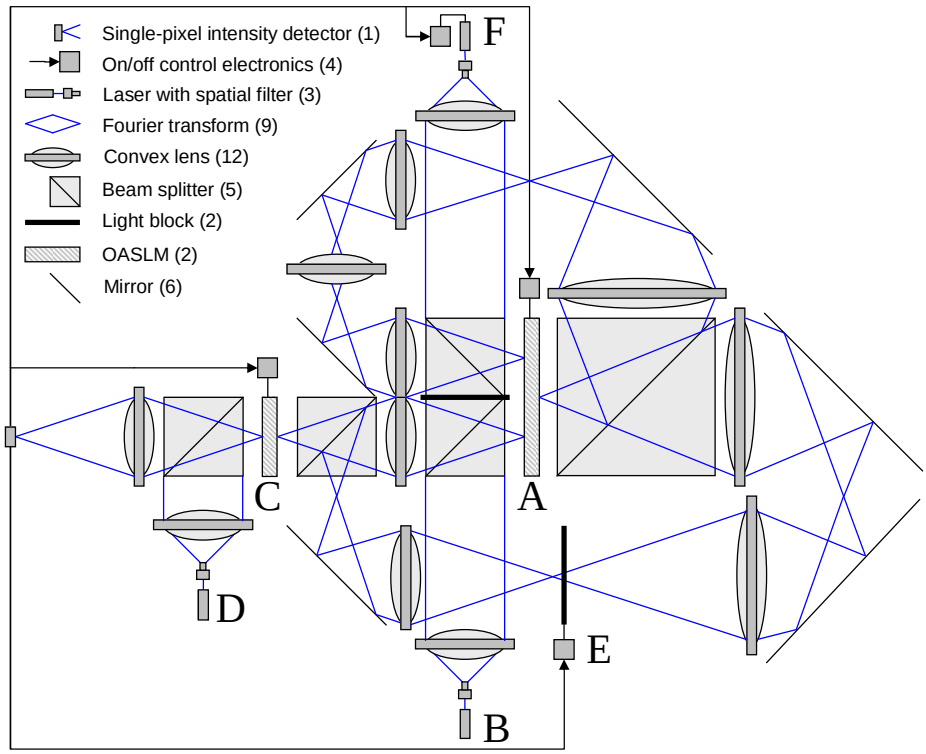

Fig. 4. Optical apparatus to perform a binary search for a single 1 in a bitstream of 0s. The legend explains the optical components and the number of them required. The labels A-F are explained in the text. OASLM: optically-addressed spatial light modulator.

Definition 8 (Needle in haystack problem). *Let* $L = \{w : w \in 0^*10^*\}$. *Let* $w \in L$ *be written as* $w = w_0 w_1 \dots w_{n-1}$ *where* $w_i \in \{0, 1\}$. *Given such a* w, *the needle in haystack problem asks what is the index of the symbol* 1 *in* w. *The solution to the needle in haystack problem for a given* w *is the index* i, *expressed in binary, where* $w_i = 1$.

This problem was posed by Grover in [13]. His quantum computer algorithm requires $O(\sqrt{n})$ comparison operations on average. Bennett *et al.* [3] have shown the work of Grover is optimal up to a multiplicative constant, and that in fact any quantum mechanical system will require $\Omega(\sqrt{n})$ comparisons. It is not difficult to see that algorithms for sequential models of computation require $\Theta(n)$ comparisons in the worst case to solve the problem. We present an algorithm that requires $O(\log_2 n)$ comparisons, with a model of computation that has promising future implementation prospects.

Our search algorithm is quite simple. A single bright point is somewhere in an otherwise dark image. If we block one half of the image we can tell in a single

step if the other half contains the bright point or not. This forms the basis of a binary search algorithm to determine the precise location of the bright point.

Theorem 6 ([35]). *There exists a CSM that solves the needle in haystack problem in* $\Theta(\log_2 n)$ *comparisons for a list of length* n*, where* $n = 2^k, k \in \mathbb{N}, k \geq 1$.

The CSM instance that performs this computation is given elsewhere [35], as are details of a counter to determine when the computation has finished and details of how the next most significant bit of the address is built up in an image at each step. We explain the main loop of the CSM algorithm. The binary input list w can be represented as a binary list image as illustrated in Fig. 2(c), or more simply by using a single point of light instead of a vertical stripe to denote value 1. Therefore, w would be represented by a small bright spot (a high amplitude peak) in an otherwise black image, where the position of the peak denotes the location of the 1 in w. During the first iteration of the loop, w is divided equally into two images (a left-hand image and a right-hand image). The nonzero peak will be either in the left-hand image or the right-hand image. In order to determine which image contains the peak in a constant number of steps, the left-hand image is Fourier transformed, squared, and Fourier transformed again. This effectively moves the nonzero peak (wherever it was originally) to the centre of the image, where it can be easily compared to a template (using a single conditional branching instruction in the CSM). If the left-hand image contains a nonzero amplitude at its centre, then the left-hand image contained the peak. In this case, the right-hand image is discarded, and the most significant bit of the address of 1 in w is 0. Otherwise, the right-hand image contained the peak, the left-hand image is discarded, and the most significant bit of the address is 1. For the next iteration, the remainder of the list is divided equally into two images and the computation proceeds according to this binary search procedure.

A schematic for an optoelectronic implementation is shown in Fig. 4. At the start of the computation, optically addressed spatial light modulator (OASLM) A is initialized to display the input list. One half of the input is read out of the OASLM using illumination B and a beam splitter in standard configuration, and is transformed by a Fourier lens so that its Fourier transform falls on OASLM C. The act of detection within C squares the image, and it is read out and Fourier transformed again using illumination D. A single point detector is centred on the optical axis – exactly where light would fall if there was a bright spot anywhere in the left half of the list on OASLM A. If light is detected, light block E is opened to allow the left half of the list to be copied to the full extent of A, otherwise illumination F is switched on to allow the right half of the list to be copied to the full extent of A. The act of detection within A will itself square the image but this is no concern because it is intended that the list would have constant phase everywhere. In fact, the response of A could be configured to nonlinearly transform the intensities in the image, to suppress any background noise and enhancing the bright spot, thus avoiding the propagation of errors which is the overriding problem with numerical calculations implemented in analog optics [16]. The left half of the remainder of the list is now ready to be Fourier transformed itself onto C, for the next step in the computation. For

simplicity, at each step we let the next bit of the address to be recorded at the detector electronics.

To more rigorously verify the operation of the apparatus in Fig. 4 from an optical engineering standpoint, the following steps in the process are stated explicitly.
Step 1: Ensure F is off, E closed, B on, D on, C cleared, input displayed on A.
Step 2: (Detector will sense the presence or absence of light.) Put A into write mode. If light is sensed, record address bit of 0 and open E for an instant, otherwise record an address bit of 1 and switch F on for an instant. Take A out of write mode. Clear C.
Step 3: Go to step 2.

All that is required to augment the design is a counter to determine when to halt (which can also be performed electronically) and a method of initialising OASLM A with the input (which can be performed by replacing the upper-right mirror with a beam splitter). This is by no means a definitive implementation of the operation, but conceptually it is very simple, and as a design it is straightforward to implement. The most difficult implementation issues concern ensuring that light close to the centre of A is appropriately partitioned by the pair of side-by-side Fourier lenses, and ensuring that the feedback paths (the two paths from A to itself) are not unduly affected by lens abberations. Ultimately, the spatial resolution of the input images (and so the size of the list inputs) is limited by the finite aperture size of the lenses. Practically, it would be desirable to configure both B and D to be pulsed in the same way as F, although this adds to the control burden. It would be possible to replace the two 4-f feedback paths with 2-f feedback paths (thereby removing two lenses from each path) if one took note that the list would be reversed at each alternate step. Further, each pair of Fourier lenses in the upper feedback arm could be replaced by a single lens in imaging configuration if one ignores the phase errors – Fourier transforming lenses are used exclusively in this design to ease detailed verification of the apparatus by the reader. Imaging lenses would also allow reduction in size of the largest of the mirrors and beamsplitter in the design. Furthermore, passive beamsplitters and planar mirrors are specified here to maintain the quality of the optical wavefronts at reasonable notional financial cost; instead employing active beam splitter technology and curved mirrors would reduce the number of components further while admitting their own disadvantages. Finally, cylindrical lenses could be used rather than spherical lenses because Fourier transformation in one dimension only is required.

4.1 Complexity of the unordered search problem

It is possible to give an AC^0 circuit family to solve the Needle in haystack problem. In fact, is is possible to give constant time CSM, or $\mathcal{C}_2$-CSM, algorithms to solve the problem. However, although fast, we felt than any such CSM algorithm that we could think of was more difficult to implement optically than the above algorithm. For example, one can consider a CSM algorithm that encodes the values $1, \ldots n$ at addresses $a_1, \ldots, a_n$, respectively. Next we assume an ordering on the n possible inputs that corresponds to the ordering in the

addressing scheme. Using such an input, the machine would simply branch to the address at the input's value, and output the image at the i^{th} address. The algorithm runs in constant TIME, linear GRID, and all other resources are constant. Although simple to describe, the use of addressing would complicate the algorithm's implementation.

5 A new direction for optical algorithm designers?

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 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 (by characterise we mean that the model solves exactly those problems in P), for a wide range of reasonable parallel and sequential optical models (see Section 3). 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. Thus, 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 [12].

How does this relate to optics? As discussed in Section 2.5, a wide range of optical computers that run for at most polylogarithmic time, and use at most polynomial space-like resources, solve exactly NC [30–32, 34]. In effect this means that we have an algorithmic method (in other words, a compiler) to convert existing NC algorithms into optical algorithms that use similar amounts of resources.

From the practical point of view, 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).[6] Another example is the NC^1 unordered search problem given in Section 4. Another

[6] On a technical note, NC can be defined as $\cup_{k=0}^{\infty}\text{NC}^k$, 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.

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 of Reif and Tyagi [23], and Caulfield's observation on the benefits of unbounded fan-in [6], can be interpreted as exploiting this important and efficient aspect of optics.

There is scope for further work here, on the continuous space machine in particular, in order to find exact characterisations, or as close as possible for NC^k for given k. Or even better, to find exact characterisations of the AC^k or TC^k classes of problems.

References

1. 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.
2. J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity II*, volume 22 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1988.
3. C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.
4. A. Bertoni, G. Mauri, and N. Sabadini. A characterization of the class of functions computable in polynomial time on random access machines. In *STOC*, pages 168–176, Milwaukee, Wisconsin, May 1981. ACM.
5. R. N. Bracewell. *The Fourier transform and its applications*. Electrical and electronic engineering series. McGraw-Hill, second edition, 1978.
6. 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.
7. 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.
8. D. G. Feitelson. *Optical Computing: A survey for computer scientists*. MIT Press, Cambridge, Massachusetts, 1988.
9. L. M. Goldschlager. *Synchronous parallel computation*. PhD thesis, University of Toronto, Computer Science Department, Dec. 1977.
10. L. M. Goldschlager. A universal interconnection pattern for parallel computers. *Journal of the ACM*, 29(4):1073–1086, Oct. 1982.
11. J. W. Goodman. *Introduction to Fourier optics*. McGraw-Hill, New York, second edition, 1996.
12. R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford university Press, Oxford, 1995.
13. 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.
14. 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, pages 869–941. Elsevier, Amsterdam, 1990.

AC^k, and TC^k are defined analogously, except we use unbounded fan-in gates, and threshold gates, respectively.

15. Lenslet Labs. Enlight256. White paper report, Lenslet Ltd., 6 Galgalei Haplada St, Herzelia Pituach, 46733 Israel, Nov. 2003.
16. 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.
17. 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.
18. 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.
19. T. J. Naughton and D. Woods. Optical computing. In Encyclopedia of Complexity and System Science. Edited by Robert A. Meyers. Springer. To appear.
20. 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.
21. I. Parberry. *Parallel complexity theory*. Wiley, 1987.
22. 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.
23. 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.
24. A. Schönhage. On the power of random access machines. In H. A. Maurer, editor, *ICALP*, volume 71 of *Lecture Notes in Computer Science*, pages 520–529, Graz, Austria, July 1979. Springer.
25. 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.
26. 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.
27. 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.
28. 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.
29. A. VanderLugt. *Optical Signal Processing*. Wiley, New York, 1992.
30. D. Woods. *Computational complexity of an optical model of computation*. PhD thesis, National University of Ireland, Maynooth, 2005.
31. 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.
32. 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.
33. 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.

34. 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.
35. D. Woods and T. J. Naughton. An optical model of computation. *Theoretical Computer Science*, 334(1-3):227–258, Apr. 2005.