RESEARCH ARTICLE

Improved approximation of spatial light distribution

David Kaljun[1], **Tina Novak**[1]*, **Janez Žerovnik**[1,2]

1 Faculty of Mechanical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia, **2** Institute of Mathematics, Physics and Mechanics, 1000 Ljubljana, Slovenia

* tina.novak@fs.uni-lj.si

Citation: Kaljun D, Novak T, Žerovnik J (2017) Improved approximation of spatial light distribution. PLoS ONE 12(4): e0176252. https://doi.org/10.1371/journal.pone.0176252

Editor: J. Alberto Conejero, IUMPA - Universitat Politecnica de Valencia, SPAIN

Received: September 11, 2016

Accepted: April 8, 2017

Published: April 28, 2017

Data Availability Statement: All relevant data are within the paper and its Supporting Information files.

Funding: This work was supported by Research Agency of Slovenia (https://www.arrs.gov.si/), P1-0285, P2-0248, and ARRS-100015-0510. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Abstract

The rapid worldwide evolution of LEDs as light sources has brought new challenges, which means that new methods are needed and new algorithms have to be developed. Since the majority of LED luminaries are of the multi-source type, established methods for the design of light engines cannot be used in the design of LED light engines. This is because in the latter case what is involved is not just the design of a good reflector or projector lens, but the design of several lenses which have to work together in order to achieve satisfactory results. Since lenses can also be bought off the shelf from several manufacturers, it should be possible to combine together different off the shelf lenses in order to design a good light engine. However, with so many different lenses to choose from, it is almost impossible to find an optimal combination by hand, which means that some optimization algorithms need to be applied. In order for them to work properly, it is first necessary to describe the input data (i.e. spatial light distribution) in a functional form using as few as possible parameters. In this paper the focus is on the approximation of the input data, and the implementation of the well-known mathematical procedure for the separation of linear and nonlinear parameters, which can provide a substantial increase in performance.

Introduction

The rapid worldwide evolution of the LED (Light emitting diode) industry has resulted in the implementation of LED elements in all kind of luminaries. Their technology means that energy consumption is much reduced, while at the same time there are endless possibilities of light engine design. In the case of LEDs lighting systems it is possible to deliver the light to the environment in a controlled way, although this leads to new problems such as finding the optimal lens or lens combination, the optimal LED to use, the optimal number of LEDs and the optimal rotation of each lens. The key to discovering a successful design process is the choice of the secondary optics. Currently there are more or less just two options for designing light engines. The first option is to have the know-how and the resources to design a specific lens in order to accomplish the task. However, the cost of resources coupled with the development and production of optical elements can be enormous. For this reason a lot of manufactures now make use of the second option, which is to use ready-made off the shelf lenses. These lenses are manufactured by a number of specialized companies, which offer different types of lenses for all the major LED brands. The trick here is to choose the best combination of lenses in order to get the most efficient system. The developer frequently makes use of a trial and error method, first choosing a good combination of lenses, and then simulating the system via Monte Carlo ray-tracing methods. The success of such procedures heavily depends on the engineer's intuition and experience, sizeable computation resources are also needed to check the proposed design by means of simulations. However, analytical models and optimization tools could be used to speed up the design process, as well as to possibly improve the quality of solutions. An analytical model which could be used to describe the far field radiation pattern of a LED was recently proposed in [1, 2]. Later it was observed that the model could also be applied to LEDs with attached secondary optics [3]. Several optimization algorithms were designed and tested, showing that the model is accurate and provides an improvement in the field [4, 5]. In particular, the metaheuristics, such as the multi-start local search algorithm and genetic algorithms, proved to be excellent pre-processing methods for Newton's method [6].

The Luminous intensity pattern of LEDs is mathematically described in form Eq (1). T3his form is a linear combination of nonlinear functions $\sum a_j\varphi_j(\boldsymbol{\alpha};\theta)$ (see Eqs (7 and 28). Thus, for the calculation of the RMS error (see Eq (3)) it is more efficient to use the procedure which involves reduction of the linear parameters, which is well known from [7]. Using this procedure it has been possible to find desired solutions in a fraction of the time needed if usual discrete optimization methods are used. In turn this means that we are no longer confined to a HPC (High Performance Computing) unit but can run the approximation on a single processor desktop unit.

Materials and methods

The model

It is well-known that the luminous intensity pattern of LEDs can be represented as a sum of cosine-power functions

$$I(\theta;\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}) = I_{\max}\sum_{j=1}^{n} a_j \cos^{c_j}(\theta - b_j) \tag{1}$$

(see in [2, 6]), where θ is the polar angle, K is the number of functions to sum, and a_j, b_j and c_j are the function coefficients. For brevity, coefficients are written as vectors $\boldsymbol{a} = (a_1, a_2,\ldots,a_n)^T$, $\boldsymbol{b} = (b_1, b_2,\ldots,b_n)^T$ and $\boldsymbol{c} = (c_1, c_2,\ldots,c_n)^T$. In the paper, after the separation of the parameters $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$, vector notation $\boldsymbol{\alpha} = (b_1, b_2,\ldots,b_n, c_1, c_2,\ldots,c_n)^T$ is also used. The interval range of the coefficients is: $a_j \in [0, 1]$, $b_j \in [-90,90]$ and $c_j \in [0, 100]$ for every $j = 1,2,\ldots,n$. In [6], discrete optimization algorithms work on finite subsets where the possible values are

$$\begin{aligned} a_j &\in \{0, 0.001, 0.002, \ldots, 1\}, \\ b_j &\in \{-90, -89.9, -89.8, \ldots, 89.9, 90\}, \\ c_j &\in \{0, 1, 2, \ldots, 100\}. \end{aligned} \tag{2}$$

For given data $(\theta_i, I_M(\theta_i))$, $i = 1,2,\ldots,m$, $(N >> 39n)$ the optimal parameters minimizing the function

$$RMS(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}) = \sqrt{\frac{1}{N}\sum_{i=1}^{m}(I_M(\theta_i) - I(\theta_i;\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}))^2} \tag{3}$$

are determined. In Eq (3), N is the number of measured points in the input data, $I_M(\theta_i)$ is the measured luminous intensity value at a polar angle θ_i, and $I(\theta_i, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ the calculated luminous intensity value at the given polar angle θ, and the given triplet of vectors $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ from the finite discrete subset of $[0, 1]^n \times [-90, 90]^n \times [0, 100]^n$. The *RMS* function represents the error of the approximation named *RMSp*, and is defined by the equation

$$RMSb(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \frac{100 \cdot m \cdot RMS(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})}{\sum_{i=1}^{m} I_M(\theta_i)} [\%]. \tag{4}$$

In order to simplify the problem, it is sufficient to consider the standard least squares problem

$$G(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \sum_{i=1}^{m} (I_M(\theta_i) - I(\theta_i; \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}))^2 . \tag{5}$$

The function G can be used since to minimize the *RMS* function (3), i.e. finding parameter values $\boldsymbol{a}^*$, $\boldsymbol{b}^*$, $\boldsymbol{c}^*$ for which the value RMS s minimal, is equivalent to minimize the function G. Due to the form of the function (1), the problem (5) is one of the so-called "separable nonlinear least squares problems", which were studied already in [7, 8]. In the next subsection we present the procedure for reducing one third of the parameters in the nonlinear least square problems where the functions in the linear square problem have a special form.

Nonlinear least square problems whose variables can be separated

Separation of the linear variables. Consider the real given data as

$$(t_i, y_i) , \quad i = 1, \ldots, m. \tag{6}$$

Denote by $\boldsymbol{a}$ and $\boldsymbol{\alpha}$ two independent vectors

$$\boldsymbol{a} = (a_1, \ldots, a_n)^T \in \mathbb{R}^n \quad \text{and} \quad \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)^T \in \mathbb{R}^k$$

and let

$$\eta(\boldsymbol{a}, \boldsymbol{\alpha}; t) = \sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t) \tag{7}$$

be nonlinear models, where φ_j are functions, continuously differentiable with respect to $\boldsymbol{\alpha}$, and t is a real variable. If instead of t a vector $\boldsymbol{t} = (t_1,\ldots,t_m)^T$ is taken, one should write

$$\boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{\alpha}) = (\eta(\boldsymbol{a}, \boldsymbol{\alpha}; t_1), \ldots, \eta(\boldsymbol{a}, \boldsymbol{\alpha}; t_m))^T$$

and

$$\boldsymbol{\varphi}_j(\boldsymbol{\alpha}) = (\varphi_j(\boldsymbol{\alpha}; t_1), \varphi_j(\boldsymbol{\alpha}; t_2), \ldots, \varphi(\boldsymbol{\alpha}; t_m))^T .$$

We also write $\boldsymbol{y} = (y_1,\ldots,y_m)^T$ for the given values in Eq (6).

Problem: Find the values of parameters $\boldsymbol{a}$ and $\boldsymbol{\alpha}$ that minimize the nonlinear functional

$$r(\boldsymbol{a}, \boldsymbol{\alpha}) = ||\boldsymbol{y} - \boldsymbol{\eta}(\boldsymbol{a}, \boldsymbol{\alpha})||^2 = \sum_{i=1}^{m} (y_i - \eta(\boldsymbol{a}, \boldsymbol{\alpha}; t_i))^2 = \tag{8}$$

$$= \sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t_i) \right)^2 . \tag{9}$$

Let $\Phi(\boldsymbol{\alpha})$ be the matrix function

$$\Phi(\boldsymbol{\alpha}) = [\boldsymbol{\varphi}_1(\boldsymbol{\alpha}), \boldsymbol{\varphi}_2(\boldsymbol{\alpha}), \ldots, \boldsymbol{\varphi}_n(\boldsymbol{\alpha})] = \begin{bmatrix} \varphi_1(\boldsymbol{\alpha}; t_1) & \varphi_2(\boldsymbol{\alpha}; t_1) & \ldots & \varphi_n(\boldsymbol{\alpha} : t_1) \\ \varphi_1(\boldsymbol{\alpha}; t_2) & \varphi_2(\boldsymbol{\alpha}; t_2) & \ldots & \varphi_n(\boldsymbol{\alpha} : t_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{\alpha}; t_m) & \varphi_2(\boldsymbol{\alpha}; t_m) & \ldots & \varphi_n(\boldsymbol{\alpha} : t_m) \end{bmatrix}. \tag{10}$$

The sum $\sum_{j=1}^{n} a_j \varphi_j(\boldsymbol{\alpha}; t_i)$ (see Eq (9)) is the i-th component of the vector

$$a_1\boldsymbol{\varphi}_1(\boldsymbol{\alpha}) + a_2\boldsymbol{\varphi}_2(\boldsymbol{\alpha}) + \ldots + a_n\boldsymbol{\varphi}_n(\boldsymbol{\alpha}) = \begin{bmatrix} a_1\varphi_1(\boldsymbol{\alpha}; t_1) + a_2\varphi_2(\boldsymbol{\alpha}; t_1) + \ldots + a_n\varphi_n(\boldsymbol{\alpha}; t_1) \\ a_1\varphi_1(\boldsymbol{\alpha}; t_2) + a_2\varphi_2(\boldsymbol{\alpha}; t_2) + \ldots + a_n\varphi_n(\boldsymbol{\alpha}; t_2) \\ \vdots \\ a_1\varphi_1(\boldsymbol{\alpha}; t_m) + a_2\varphi_2(\boldsymbol{\alpha}; t_m) + \ldots + a_n\varphi_n(\boldsymbol{\alpha}; t_m) \end{bmatrix}. \tag{11}$$

Therefore, the functional r is geometrically the length of the vector

$$\boldsymbol{y} - (a_1\boldsymbol{\varphi}_1(\boldsymbol{\alpha}) + a_2\boldsymbol{\varphi}_2(\boldsymbol{\alpha}) + \ldots + a_n\boldsymbol{\varphi}_n(\boldsymbol{\alpha})) .$$

For a given vector $\boldsymbol{y}$ the value of r is minimal if and only if the sum (vector) $a_1\boldsymbol{\varphi}_1(\boldsymbol{\alpha}) + a_2\boldsymbol{\varphi}_2(\boldsymbol{\alpha}) + \ldots + a_n\boldsymbol{\varphi}_n(\boldsymbol{\alpha})$ is the orthogonal projection of $\boldsymbol{y}$ onto the subspace $\mathcal{L}\{\boldsymbol{\varphi}_1(\boldsymbol{\alpha}), \boldsymbol{\varphi}_2(\boldsymbol{\alpha}), \ldots, \boldsymbol{\varphi}_n(\boldsymbol{\alpha})\}$. For each $\boldsymbol{\alpha}$, the linear operator

$$P_{\Phi(\boldsymbol{\alpha})} = \Phi(\boldsymbol{\alpha})\Phi^{+}(\boldsymbol{\alpha})$$

is the orthogonal projection on the linear space spanned by the columns of the matrix $\Phi(\boldsymbol{\alpha})$ (see Remark 2), i.e. the linear space $\mathcal{L}\{\boldsymbol{\varphi}_1(\boldsymbol{\alpha}), \boldsymbol{\varphi}_2(\boldsymbol{\alpha}), \ldots, \boldsymbol{\varphi}_n(\boldsymbol{\alpha})\}$.

Remark 1 *The matrix $\Phi^{+}(\boldsymbol{\alpha})$ is the generalized inverse or so called Moore-Penrose pseudoinverse. For every $m \times n$ matrix A, there exists a unique $n \times m$ matrix X, such that*

$$AXA = A \tag{12}$$

$$XAX = X \tag{13}$$

$$(AX)^{\top} = AX \tag{14}$$

$$(XA)^{\top} = XA. \tag{15}$$

A^{+} is defined to be X. The proof can be found in [9].

Remark 2 *From the definition of the Moore-Penrose pseudoinverse we obtain*

$$P_{\Phi(\boldsymbol{\alpha})}\Phi(\boldsymbol{\alpha}) = \Phi(\boldsymbol{\alpha})\Phi^{+}(\boldsymbol{\alpha})\Phi(\boldsymbol{\alpha}) = \Phi(\boldsymbol{\alpha})$$

and hence

$$P_{\Phi(\boldsymbol{\alpha})}\boldsymbol{\varphi}_j(\boldsymbol{\alpha}) = \boldsymbol{\varphi}_j(\boldsymbol{\alpha})$$

for every $j = 1,\ldots,n$. Consider that v is a nonzero vector in the orthogonal complement of the subspace $\mathcal{L}\{\boldsymbol{\varphi}_1(\boldsymbol{\alpha}), \boldsymbol{\varphi}_2(\boldsymbol{\alpha}), \ldots, \boldsymbol{\varphi}_n(\boldsymbol{\alpha})\}$ according to the usual scalar product. Such a vector v is *characterized by the vanishing of the scalar product of the vectors v and $\boldsymbol{\varphi}_j(\boldsymbol{\alpha})$ for every j = 1,. . .,n. Therefore,*

$$(\Phi(\boldsymbol{\alpha}))^\top v = 0\,.$$

Hence the following is true

$$\begin{aligned}(\Phi^+(\boldsymbol{\alpha}))^\top(\Phi(\boldsymbol{\alpha}))^\top v &= 0\\ (\Phi(\boldsymbol{\alpha})\Phi^+(\boldsymbol{\alpha}))^\top v &= 0\\ \Phi(\boldsymbol{\alpha})\Phi^+(\boldsymbol{\alpha})v &= 0\\ P_{\Phi(\boldsymbol{\alpha})}v &= 0\,.\end{aligned}$$

Another linear operator needed here is

$$P^\perp_{\Phi(\boldsymbol{\alpha})} = I - P_{\Phi(\boldsymbol{\alpha})}\,. \tag{16}$$

Geometrically, this is the projection onto the orthogonal complement of the linear space spanned by the columns of the matrix $\Phi(\boldsymbol{\alpha})$. For any given $\boldsymbol{\alpha}$ we have

$$r(\hat{\boldsymbol{a}},\boldsymbol{\alpha}) = \min_{\boldsymbol{a}} r(\boldsymbol{a},\boldsymbol{\alpha}) = ||\boldsymbol{y} - \Phi(\boldsymbol{\alpha})\Phi^+(\boldsymbol{\alpha})\boldsymbol{y}||^2 = ||P^\perp_{\Phi(\boldsymbol{\alpha})}\boldsymbol{y}||^2$$

and

$$\hat{\boldsymbol{a}}(\boldsymbol{\alpha}) \equiv \Phi^+(\boldsymbol{\alpha})\boldsymbol{y}\,. \tag{17}$$

Define the modified functional as

$$r_2(\boldsymbol{\alpha}) = ||P^\perp_{\Phi(\boldsymbol{\alpha})}\boldsymbol{y}||^2. \tag{18}$$

Once a critical point (minimizer) $\hat{\boldsymbol{\alpha}}$ of r_2 is found, $\hat{\boldsymbol{a}}$ can be calculated using Eq (17) on $\hat{\boldsymbol{\alpha}}$, i.e.

$$\hat{\boldsymbol{a}} = \Phi^+(\hat{\boldsymbol{\alpha}})\boldsymbol{y}\,. \tag{19}$$

Theorem 3 (Golub, Pereyra 1973) *Let $r(\boldsymbol{a},\boldsymbol{\alpha})$ and $r_2(\boldsymbol{\alpha})$ be defined as above. We assume that in the open set $\Omega \subset \mathbb{R}^k$, the matrix $\Phi(\boldsymbol{\alpha})$ has a constant rank $r \leq \min(m, n)$.*

1. *If $\hat{\boldsymbol{\alpha}}$ is a critical point (or a global minimizer for $\boldsymbol{\alpha} \in \Omega$) of $r_2(\boldsymbol{\alpha})$, and*

$$\hat{\boldsymbol{a}} = \Phi^+(\hat{\boldsymbol{\alpha}})\boldsymbol{y} \tag{20}$$

then $(\hat{\boldsymbol{a}},\hat{\boldsymbol{\alpha}})$ is a critical point of $r(\boldsymbol{a},\boldsymbol{\alpha})$ (or a global minimizer for $\boldsymbol{\alpha} \in \Omega$) and $r(\hat{\boldsymbol{a}},\hat{\boldsymbol{\alpha}}) = r_2(\hat{\boldsymbol{\alpha}})$.

2. *If $(\hat{\boldsymbol{a}},\hat{\boldsymbol{\alpha}})$ is a global minimizer of $r(\boldsymbol{a},\boldsymbol{\alpha})$ for $\boldsymbol{\alpha} \in \Omega$, then $\hat{\boldsymbol{\alpha}}$ is a global minimizer of $r_2(\boldsymbol{\alpha})$ in Ω and $r_2(\hat{\boldsymbol{\alpha}}) = r(\hat{\boldsymbol{a}},\hat{\boldsymbol{\alpha}})$. Furthermore, if there is an unique $\hat{\boldsymbol{a}}$ among the minimizing pairs of $r(\boldsymbol{a},\boldsymbol{\alpha})$, then $\hat{\boldsymbol{a}}$ must satisfy Eq (20).*

For the proof see [8], where the method of decimation of the functional r_2 is given after the theorem. The orthogonal transformation of the matrix Φ into "trapezoidal" form is used. Since we use C++ in the computations, in the next paragraph we shall present the method of simplification of the functional r_2 in SVD manner.

Computing in C++, Singular Value Decomposition. In the Armadillo library (a high quality linear algebra library (matrix maths) for the C++ language, which aims to provide the ease of use and performance of MATLAB functions in more low level programming. "http://arma.sourceforge.net/") there is the command **svd** for calculating the Singular Value

Decomposition. For every rectangular $m \times n$ matrix Φ, where $m \geq n$, the command **svd** gives us the matrices U and V and the vector s. The matrices U and V are orthogonal matrices of dimension $m \times m$ and $n \times n$ respectively. Further, s is the vector of the singular values $s_1, s_2, \ldots, s_n$. Let S be the diagonal matrix $S = \mathrm{diag}(s_1, \ldots, s_n)$. We can assume that

$$s_1 \geq s_2 \geq \ldots \geq s_n \geq 0. \tag{21}$$

If $\mathrm{rank}(\Phi) = r$, then $s_{r+1} = \ldots = s_n = 0$. If $\mathrm{rank}(\Phi) = \mathrm{rank}(S) = r$, then denote by S_r the invertible diagonal matrix

$$S_r = \begin{bmatrix} s_1 & 0 & \ldots & 0 \\ 0 & s_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & s_r \end{bmatrix}.$$

Denote by Σ the $m \times n$ matrix defined by

1. if $r = n$, we write $\Sigma = \left[\begin{smallmatrix} S \\ 0 \end{smallmatrix}\right]$,
2. if $r < n$, then $\Sigma = \left[\begin{array}{c|c} S_r & 0 \\ 0 & 0 \end{array}\right]$.

Thus,

$$\Phi(\boldsymbol{\alpha}) = U(\boldsymbol{\alpha})\Sigma(\boldsymbol{\alpha})V^T(\boldsymbol{\alpha}) \tag{22}$$

(see [10], the alternative representation of SVD). The Moore-Penrose pseudoinverse of the matrix Σ is the following $n \times m$ matrix

1. if $r = n$, then $\Sigma^+ = [S^{-1}|0]$,
2. if $r < n$, then $\Sigma^+ = \left[\begin{array}{c|c} S_r^{-1} & 0 \\ 0 & 0 \end{array}\right]$.

Therefore, the Moore-Penrose pseudoinverse of the matrix Φ is the matrix

$$\Phi^+(\boldsymbol{\alpha}) = V(\boldsymbol{\alpha})\Sigma^+(\boldsymbol{\alpha})U^T(\boldsymbol{\alpha})\,. \tag{23}$$

The more simple form of the matrix of the projection P_Φ can be easily calculated by

$$\begin{aligned} P_{\Phi(\boldsymbol{\alpha})} &= \Phi(\boldsymbol{\alpha})\Phi^+(\boldsymbol{\alpha}) = U(\boldsymbol{\alpha})\Sigma(\boldsymbol{\alpha})V^T(\boldsymbol{\alpha})V(\boldsymbol{\alpha})\Sigma^+(\boldsymbol{\alpha})U^T(\boldsymbol{\alpha}) = \\ &= U(\boldsymbol{\alpha})\Sigma(\boldsymbol{\alpha})\Sigma^+(\boldsymbol{\alpha})U^+(\boldsymbol{\alpha}) = U(\boldsymbol{\alpha})\left[\begin{array}{c|c} I_r & 0 \\ \hline 0 & 0 \end{array}\right]U^T(\boldsymbol{\alpha}), \end{aligned}$$

where I_r denote the identity matrix of the dimension $r \times r$ for $r = 1, \ldots, n - 1$. The orthogonal projection $P^{\perp}_{\Phi(\boldsymbol{\alpha})}$ (see Eq (16)) has the form

$$P^{\perp}_{\Phi(\boldsymbol{\alpha})} = U(\boldsymbol{\alpha})\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r} \end{array}\right]U^T(\boldsymbol{\alpha}).$$

From the given vector $\boldsymbol{y} = (y_1,\ldots,y_m)^T$ another vector $\tilde{y}(\boldsymbol{\alpha})$ can be defined by

$$\tilde{\boldsymbol{y}}(\boldsymbol{\alpha}) \;=\; U^T(\boldsymbol{\alpha})\boldsymbol{y}. \tag{24}$$

The functional r_2 (see Eq (18)) can be written as

$$\begin{aligned} r_2(\boldsymbol{\alpha}) &= ||U(\boldsymbol{\alpha})\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r}\end{array}\right]U^T(\boldsymbol{\alpha})\boldsymbol{y}||^2 = \\ &= ||U(\boldsymbol{\alpha})\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r}\end{array}\right]\tilde{\boldsymbol{y}}(\boldsymbol{\alpha})||^2. \end{aligned}$$

If we write the vector $\tilde{\boldsymbol{y}}$ in a block form such as

$$\tilde{\boldsymbol{y}}(\boldsymbol{\alpha}) = \begin{bmatrix} \tilde{\boldsymbol{y}}_{[r]}(\boldsymbol{\alpha}) \\ \hline \tilde{\boldsymbol{y}}_{[m-r]}(\boldsymbol{\alpha}) \end{bmatrix} \tag{25}$$

and since the orthogonal transformation (in our case $U(\boldsymbol{\alpha})$) is isometric, we obtain

$$r_2(\boldsymbol{\alpha}) = ||\left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & I_{m-r}\end{array}\right]\tilde{\boldsymbol{y}}(\boldsymbol{\alpha})||^2 = ||\tilde{\boldsymbol{y}}_{[m-r]}(\boldsymbol{\alpha})||^2. \tag{26}$$

Model $\sum_{j=1}^{3} a_j \cos^{c_j}(\theta - b_j)$.
Let

$$\boldsymbol{a} = (a_1, a_2, a_3)^T \in \mathbb{R}^3 \quad , \qquad \boldsymbol{\alpha} = (b_1, b_2, b_3, c_1, c_2, c_3)^T \in \mathbb{R}^6 \quad \text{and} \qquad \boldsymbol{\beta} = (b, c)^T \in \mathbb{R}^2$$

be independent vectors. Denote by pr_j the projection

$$\begin{aligned} \mathrm{pr}_j : \mathbb{R}^6 &\longmapsto \mathbb{R}^2 \\ (b_1, b_2, b_3, c_1, c_2, c_3)^T &\longmapsto (b_j, c_j)^T \end{aligned}$$

for $j = 1, 2, 3$. Let ψ be the function, defined by

$$\begin{aligned} \psi : \mathbb{R}^2 \times \mathbb{R} &\longrightarrow \mathbb{R} \\ (\boldsymbol{\beta}; \theta) &\longmapsto \cos^c(\theta - b)\,. \end{aligned} \tag{27}$$

To write the function $\sum_{j=1}^{3} a_j \cos(\theta - b_j)^{c_j}$ in the form Eq (7), we take

$$\begin{aligned} \varphi_1(\boldsymbol{\alpha}; \theta_i) &= \psi(\mathrm{pr}_1(\boldsymbol{\alpha}); \theta_i) \\ \varphi_2(\boldsymbol{\alpha}; \theta_i) &= \psi(\mathrm{pr}_2(\boldsymbol{\alpha}); \theta_i) \\ \varphi_3(\boldsymbol{\alpha}; \theta_i) &= \psi(\mathrm{pr}_3(\boldsymbol{\alpha}); \theta_i)\,, \end{aligned}$$

or shortly

$$\varphi_j(\boldsymbol{\alpha}; \theta_i) = \psi(\mathrm{pr}_j(\boldsymbol{\alpha}); \theta_i) = \cos^{c_j}(\theta_i - b_j) \tag{28}$$

for $j = 1, 2, 3$ and $i = 1,\ldots,m$. The matrix $\Phi(\boldsymbol{\alpha})$ is the $m \times 3$ matrix

$$\Phi(\boldsymbol{\alpha}) = \begin{bmatrix} \cos^{c_1}(\theta_1 - b_1) & \cos^{c_2}(\theta_1 - b_2) & \cos^{c_3}(\theta_1 - b_3) \\ \cos^{c_1}(\theta_2 - b_1) & \cos^{c_2}(\theta_2 - b_2) & \cos^{c_3}(\theta_2 - b_3) \\ \vdots & \vdots & \vdots \\ \cos^{c_1}(\theta_m - b_1) & \cos^{c_2}(\theta_m - b_2) & \cos^{c_3}(\theta_m - b_3) \end{bmatrix}. \tag{29}$$

For given $\boldsymbol{\alpha}$, in C++ we calculate the matrices U, V and the vector of the singular values s by the command **svd**. The rank r of the matrix $\Phi(\boldsymbol{\alpha})$ is equal to the number of nonzero singular values in s. We calculate the vector $\tilde{\boldsymbol{y}}$ as in Eq (24) and solve the least square problems

$$r_2(\boldsymbol{\alpha}) = ||\tilde{\boldsymbol{y}}_{[m-r]}(\boldsymbol{\alpha})||^2. \tag{30}$$

As soon as the solution $\hat{\boldsymbol{\alpha}}$ of the least square problem (30) is found we can calculate $\hat{\boldsymbol{a}}$ (see Theorem 3) by $\hat{\boldsymbol{a}} = \Phi^+(\hat{\boldsymbol{\alpha}})\boldsymbol{y}$.

The algorithms

In previous works [4, 6], the model described above was applied in conjunction with several custom-built algorithms that are based on local search heuristics and some meta-heuristics. The implemented algorithms include a steepest descen T algorithm, two iterative improvement algorithms with different neighbourhoods, and two genetic algorithms, a standard one and a hybrid one in which the best individuals of every generation are optimized with an iterative improvement algorithm. For a more detailed description of the algorithms we refer to [4]. The results of the experiments showed that all of the applied algorithms are capable of providing satisfactory results in all the tested instances, and differed mainly in the computational time needed. The average RMS values obtained on real lenses were around $RMS = 2\%$. Hence, the above-mentioned results proved that the model is accurate, and that sufficiently good approximations can be found with a variety of algorithms for a sufficiently good description of lenses.

However, it should be remembered that the model can also be used for data compression tasks. A zero or very low RMS error is also essential in the foreseen application, in which premanufactured lenses need to be combined into a more complex luminaire with a prescribed light distribution.

In the model we use a sum of functions that are smooth, and hence the first and second derivatives can be calculated allowing application of continuous optimization methods in addition to the general discrete optimization meta-heuristics that were used before. We decided to use the Newton (also known as the Newton–Raphson) iterative method [11, 12] in order to find the sought-for solution. It is well known that convergence of the Newton method may significantly depend on the initial solution. We therefore applied the method in two ways. First, we used the Newton method as an optimizer which can pinpoint the local minimum of solutions found by heuristic algorithms. In a sense this implementation of the Newton method is an extension of the discrete optimization algorithm, which is used to finalize the search to end in a local minimum. (Note that the local minima may be missed by discrete optimization algorithms due to the predefined length of the discrete moves.) Secondly, we use the Newton method as a stand-alone algorithm that will, on initialization, generate a number of random (initial) solutions that are uniformly scattered over the whole search space. It then uses the Newton method on a number of the best initial solutions in order to find the local minima. Of course, for both implementations to be comparable, the iteration method has to be controlled so that the overall maximum amount of computation time is roughly the same.

Multi-start IF and IF-R. The **multi-start iterative improvement with fixed neighbourhood (IF) and IF with reduced parameters IF-R** algorithm [4–6] first initializes several initial solutions. The initial solutions are randomly chosen from the whole search space. Each of the initial solutions is then optimized using the following steps. At the beginning, the search step values (the step for numerical differentiation) $da = 0.01$, $db = 1$, and $dc = \frac{I_{max}}{10}$ are initialized, giving the 512 neighbours of the initial solution: ($a_1 \pm da$, $b_1 \pm db$, $c_1 \pm dc$, $a_2 \pm da$, $b_2 \pm db$, $c_2 \pm dc$, $a_3 \pm da$, $b_3 \pm db$, $c_3 \pm dc$). In the case of IF-R we only initialize the step values $db = 1$ and $dc = \frac{I_{max}}{10}$, giving 64 neighbours of the initial solution: ($b_1 \pm db$, $c_1 \pm dc$, $b_2 \pm db$, $c_2 \pm dc$, $b_3 \pm db$, $c_3 \pm dc$). The algorithm then randomly chooses a neighbour, and immediately moves to this neighbour if its RMS value is better than the current RMS value. If no better neighbour is found after 1000 trials, it is assumed that no better neighbour exists. In this case the algorithm morphs the neighbourhood by changing the step according to the formula $d_{i+1} = d_i + d_0$. More precisely, $db_{i+1} = db_i + db_0$ where db_0 is the initial step value and by analogy $dc_{i+1} = dc_i + dc_0$ where dc_0 is the initial step value.

This is repeated until $i = 10$. If there still is no better solution, the initial step value is multiplied by 0.9 and the search is resumed from the current solution with a finer initial step. The algorithm stops when the number of generated solutions reaches T_{max}.

Newton's method IF-N. Newton's method [11, 13, 14] is a well-known numerical optimization method, which can provide very good results under certain assumptions about the evaluation function and the initial solution. The evaluation function is indirectly minimized by looking for a solution of a system of nonlinear equations (the first derivatives of the evaluation function). Newton's method solves the system of nonlinear equations iteratively by approximating it, in each step, with a system of linear equations which produce the delta vector. The delta vector is a part of the iterative scheme $\mathbf{x}_k^{i+1} = \mathbf{x}_k^i - \mathbf{d}_k^i$. The Newton method converges when the delta vector vanishes, $d = 0$. At this point the evaluation coefficients found are the local minima. Details are given in [6]. An obvious assumption is that the evaluation function has to be a continuous non-linear function for which first and second order derivatives are defined. For Newton's method to converge, the initial solution has to be close enough to a local or global optimum. For this reason the method is very sensitive to the choice of the initial solution.

The datasets

The experimental study made use of two batches consisting of 9 and 3 instances. We selected 9 different asymmetrical lenses which are meant to be used with a CREE XT-E series LED, from one of the world's leading lens manufacturer LEDIL from Finland. We acquired the photometric data from LEDIL's on-line catalogue [15]. The data was provided in.ies format, which we then converted to a vector list which is more suitable to use in our algorithms. LEDIL measured the individual lenses with a polar precision of 1° on 25 C panels. Additionally, we also ran the approximation on three LEDIL lenses that were measured in the photometric laboratory at the Faculty of Electrical Engineering in Ljubljana. These three (Komb1, Komb2 and Komb2nr) lenses from Fig 1 were measured at higher azimuth resolution, which yielded 13032 measurements and the same number of vectors to be approximated. The total number of real instances was 12.

Experimental results and discussion

In this section the results obtained by using the three presented algorithms are discussed. All the algorithms ran for $T_{max} = 4$ million iterations. After every ten thousand iterations we saved

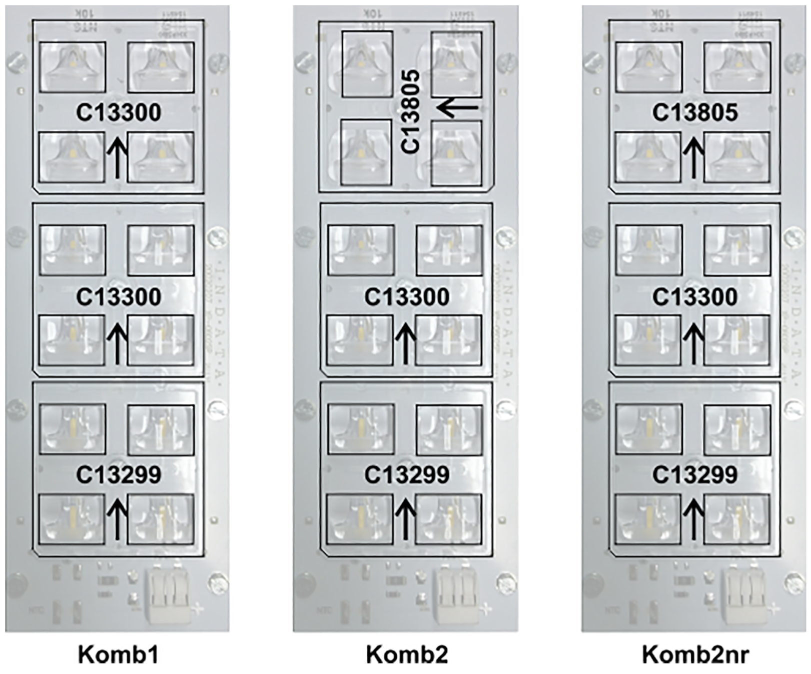

Fig 1. Graphical representation of the measured Ledil oy. lens combinations.

https://doi.org/10.1371/journal.pone.0176252.g001

the results which are presented in the section Supporting Information in S1, S2 and S3 Tables. The results are presented as the averages over 25 C-panels for each lens. The input data consisted of 12 lenses. The first 9 lenses were obtained from the manufacturers catalogue [15], and the last three were measured by us.

In S1, S2 and S3 Tables we present the average, minimum, and maximum RMS error after a varying number of iterations for the IF, IF-N and IF-R algorithms respectively. The types of lenses are given in the first columns. Experimentally, we also upgraded the IF algorithm with reduced parameters by the Newton method. Compared to S3 Table, the results of the IF-R algorithm upgraded by the Newton method are exactly the same. An obvious conclusion is that, at least in the cases tested, the results provided by the IF-R algorithm are so good that there is no room for improvement by the Newton method.

From the paper [6], we know that the IF-N algorithm improved the algorithms in [3, 4] by an average of 60% increased quality (minimized RMS). Here, S1 and S2 Tables show that the IF-N algorithm improves the classical algorithm IF slightly. However, we are more interested in a comparison of the results of the RMS errors of the IF-R algorithm with respect to the data of the IF algorithm (or the IF-N algorithm).

We can immediately see that our new algorithm (IF-R) is, on average, much better than both the IF and the IF-N algorithms. The errors after 4M iterations are much lower in our new algorithm with respect to the usual IF algorithm. On the other hand, the errors after 4M iterations of the new algorithm and the IF-N algorithm are comparable.

In order to analyze the results given in S1, S2 and S3 Tables more closely let us consider the graphs provided below. On the horizontal axis we have the number of iterations, and on the vertical axis the RMS error. The average convergence curves for the whole lenses are very similar, so that are not presented here. Instead we chose one lens, the Komb1 (all 25 C-planes) to present the results and comparisons, so that we can see the characteristic differences between the C-plane approximations.

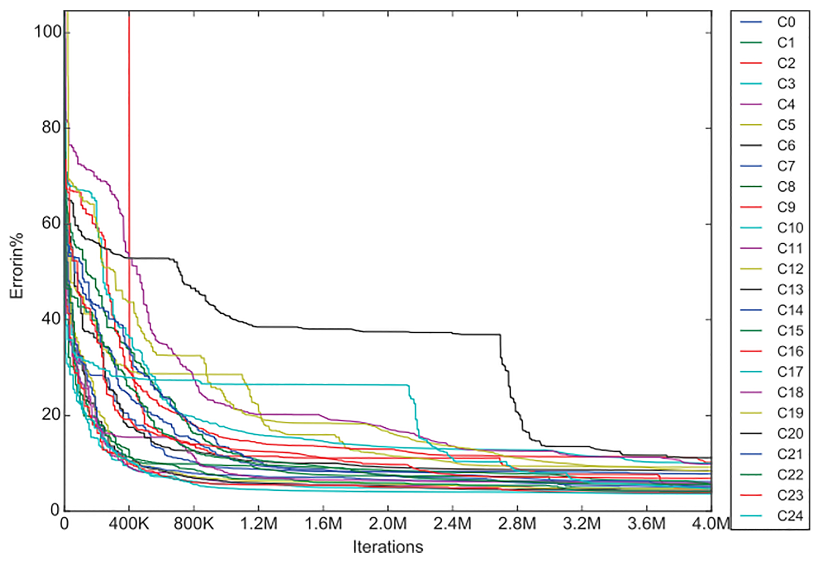

Fig 2. Convergence curves of the lens Komb1 for the *C*-planes of the IF algorithm. The curves in the graph are the best on each C-panel regardless of the multi-start. In other words C0 can be obtained from multi-start 5 and C1 from multi-start 9. The criteria for the best approximation is the RMS obtained at T_{max}.

https://doi.org/10.1371/journal.pone.0176252.g002

In Fig 2, the convergence curves of Komb1 from the IF algorithm are given for each *C*-plane. It can be seen that the improvement of the errors seems to be insignificant between 3.2M and 4M iterations. It can also be observed that, while the majority of the C-planes follow the same convergence pattern, there are some that deviate. We could attribute this deviation to poorly chosen initial solutions, or to specific input data which can cause the search algorithms to struggle (see C17 and C20 in Fig 2). As the experiment was set up in multi-start mode, which means that every approximation started 10 times with randomly chosen initial solutions, and that the presented data is the best of the best it is more likely that the convergence curve deviation is due to specific input data. However it can be seen that, in the end, no matter which input was used and how the convergence curve looks like, all the RMS at T_{max} are similar.

A comparison of the convergence of the IF-N algorithm and of the IF-R algorithm for the Komb1 lens is represented in the next figures (Figs 3, 4), where in both graphs the same scale is used.

We chose the IF-N algorithm because it exhibits almost the same convergence rate as the IF algorithm, but provides slightly better end results (see Fig 5). This makes it more appropriate for comparisons with the new algorithm.

Clearly, the convergence of the IF-R algorithm is much faster than that of the IF-N algorithm. In fact it is so fast that the scale used does not illustrate any properties of the new algorithm's convergence apart from its speed. For this reason we looked at the graph in Fig 4 on a smaller horizontal scale and got the following graph (Fig 6)

The convergence of the IF-R algorithm is 40 times faster than that of the IF and IF-N algorithms, but the curve slopes (gradients) are still similar, just on a smaller scale. Also, similarly to the previous algorithms the IF-R algorithm shows only minute errors after 100K iterations. From this it can be concluded that parameter separation has a huge impact on computation time but does not alter the algorithm behaviour when searching for the best solution, which is somewhat expected, as we did not change the algorithm workflow, just the pool of possible solutions.

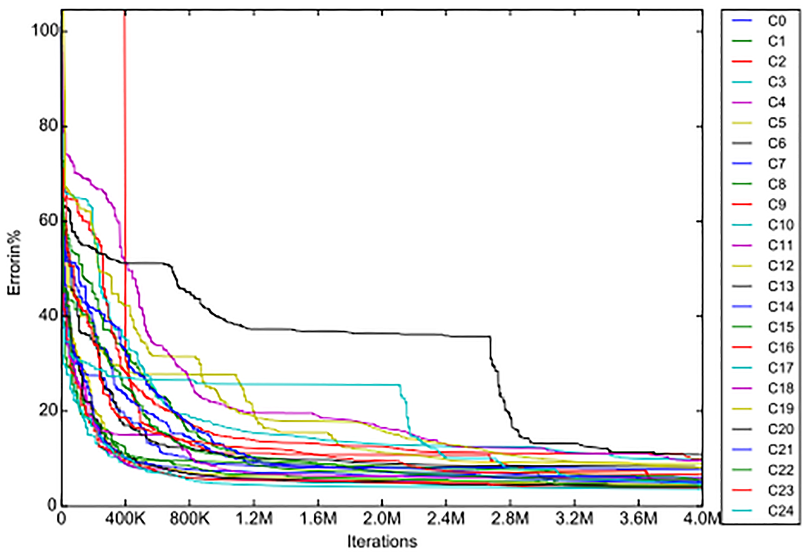

Fig 3. The convergence of the IF-N algorithm.

https://doi.org/10.1371/journal.pone.0176252.g003

Next we take out the graphs for the *C*1 plane from the figures above, and from the data of the iterations we add the convergence curve of the IF-N algorithm. The convergence curves of all three algorithms (IF, IF-N and IF-R) for the C1 plane of the Komb1 lens are displayed in Fig 5. From this it can be seen that the IF-N algorithm does not change the convergence curve as was implied above, it just improves the RMS values slightly at every step. We again observe the superiority of the algorithm with reduced parameters, which converges in a fraction of the time needed for the other two algorithms.

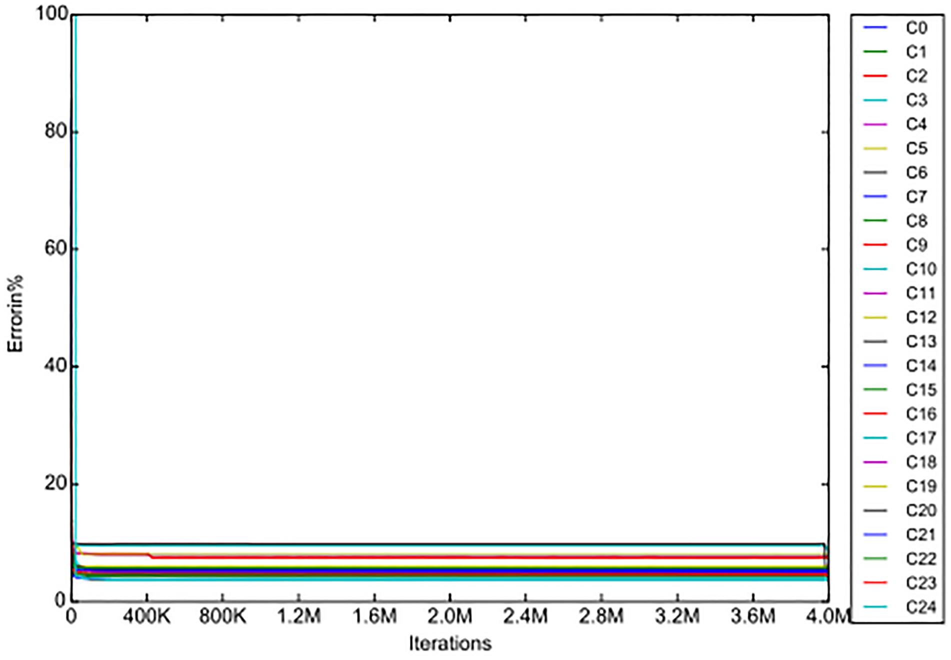

Fig 4. The convergence of the IF-R algorithm.

https://doi.org/10.1371/journal.pone.0176252.g004

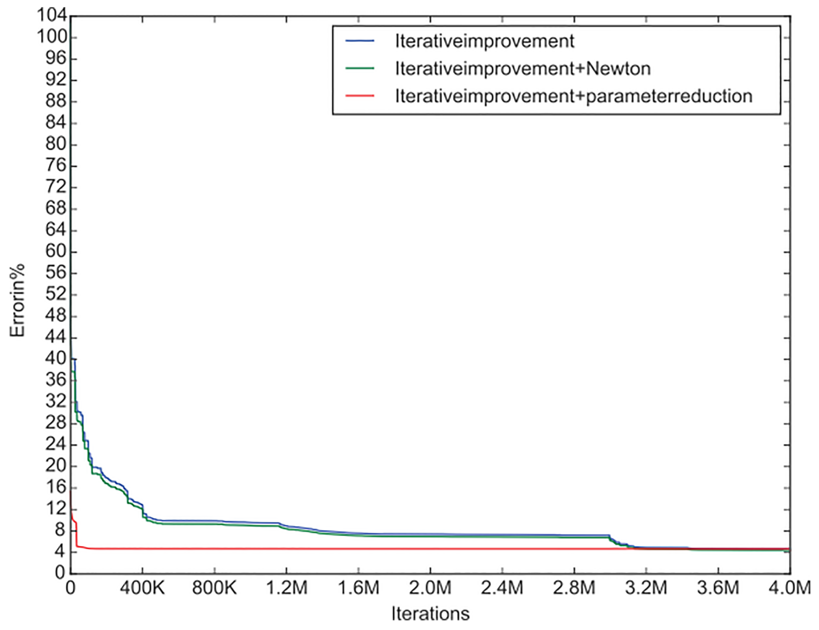

Fig 5. Convergence curves of the Komb1 lens for the *C*-plane 1.

https://doi.org/10.1371/journal.pone.0176252.g005

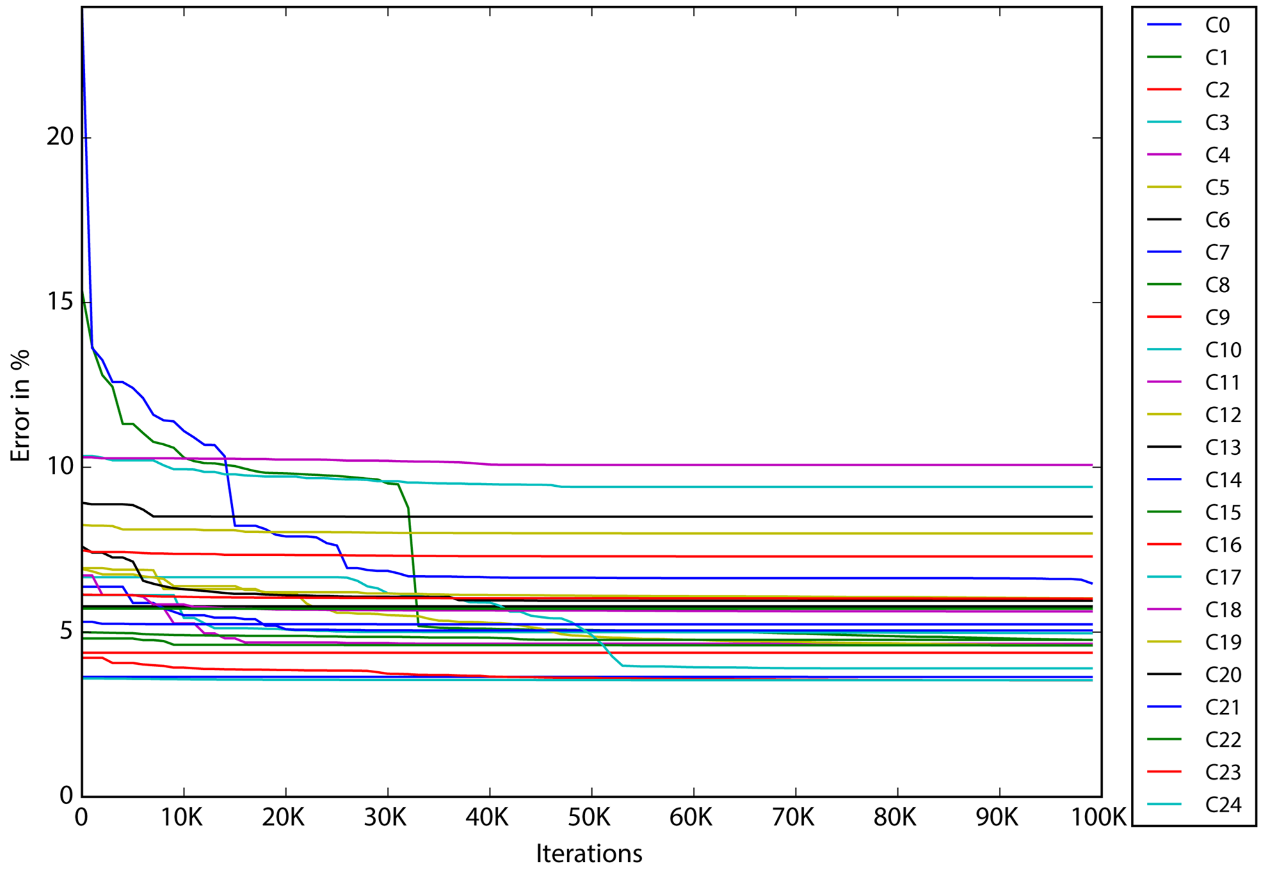

Fig 6. Convergence of the IF-R algorithm with reduced parameters in 100K iterations.

https://doi.org/10.1371/journal.pone.0176252.g006

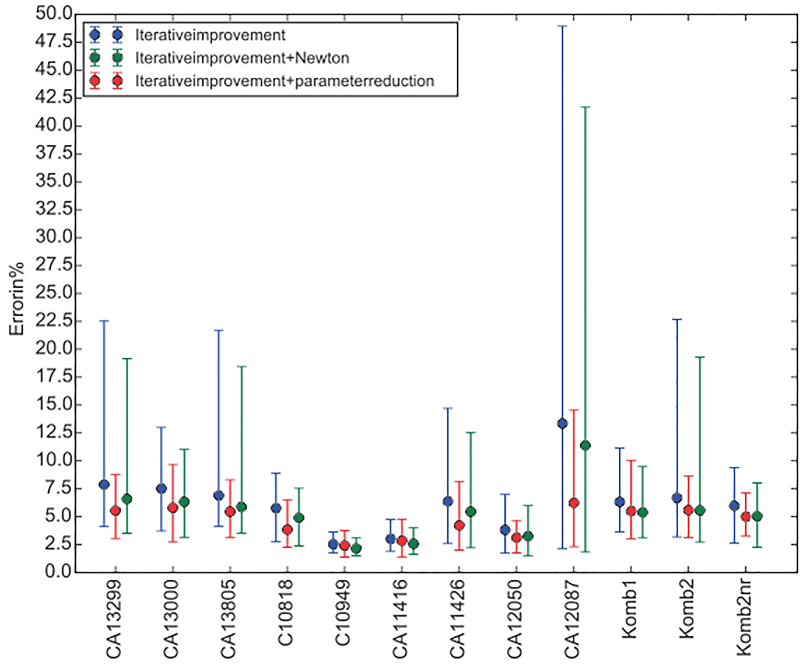

Fig 7. Min-Avg-Max scatter diagram.

https://doi.org/10.1371/journal.pone.0176252.g007

For the last comparison of the algorithms, we showcase a scatter diagram of all the lenses in Fig 7. This diagram shows the RMS values at T_{max} for the best, the worst, and the average over 25 C-panels of each lens from the experiment.

Again there is an obvious superiority of the new algorithm drawn in red, where the differences between the maximum (worst) RMS and the minimum (best) RMS are the smallest. It can be seen that the differences in the case of the new algorithm are mostly two times smaller than in the case of the old algorithms, but they can also be 4 times smaller as in case of the CA12087 lens.

It can therefore be concluded that the used method of separating the linear and non-linear parameters, provides a huge performance boost to the developed algorithms, even to the level where these algorithms could be used in commercial applications, which are meant to be used on a daily basis by users who are not programmers or mathematicians.

Remark 4 *Following the recommendation of one of the reviewers, our results were compared with the results of the standard optimization solver in Matlab. The standard function* **lsqcurvefit** *for approximation of nonlinear function given by Eq (1) was used. For the initial values, random solutions (i.e. vectors from* $[0, 1]^3 \times [-90, 90]^3 \times [0, 100]^3$*) were generated, and the measured data were the measured luminous values* $I_M(\theta_i)$ at polar angles θ_i *(see section The model). We ran* **lsqcurvefit** *for all 25 C-panels for each of the first 9 lenses. To allow approximately the same running time (Matlab was run on another computer), we aim to allow approximately the same number of feasible solutions generated by each of the solvers. Therefore, on each instance (each C panel of each lens), function* **lsqcurvefit** *was restarted 50 times with 40 000 iterations allowed and precision 1e-6. The best solutions on each instance are memorised. In S4 Table, the RMS error on best, worst and average C panel are reported (right rows). Roughly speaking, comparison with the average, minimum, and maximum RMS error obtained IF-R algorithm (left rows) shows that in most cases, the solutions are of similar quality. More precisely, on lens CA12087, Matlab* *solver provides solutions with significantly lower RMS error (two to three times better), but on the other hand, on lenses CA13299, CA13300 and CA13805, the IF-R solutions are about 6 times better on average! Note that on these three lenses, the Matlab solver was in addition run with 500 restarts, with 400 000 iterations allowed and with precision 1e-10, and in all cases, the solutions did not improve significantly. On the other hand, note that IF-R finds solutions of the reported quality in much shorter runs (see* Figs 3 and 4 *), and was allowed 4M iterations just to show the speed up in comparison to the other algorithms tested. This additional experiment thus showed that the reduction of parameters dramatically improves the convergence speed, and sometimes also the quality of approximation. An interesting avenue of further research may be to optimize the implementation and parameter tuning of IF-R, and, on the other hand, understand the properties of the instances (c.f. CA13299, CA13300 and CA13805) that allowed so diverse solution qualities.*

Supporting information

S1 Table. RMS error values for IF. The Average sub-table presents the average data over 25 C-panels, the Min table the best, and the Max table the worst C-panel.
(PDF)

S2 Table. RMS error values for IF-N. The Average sub-table presents the average data over 25 C-panels, the Min table the best, and the Max table the worst C-panel.
(PDF)

S3 Table. RMS error values for IF-R. The Average sub-table presents the average data over 25 C-panels, the Min table the best, and the Max table the worst C-panel.
(PDF)

S4 Table. RMS error values for IF-R and Matlab. The average, the best and the worst RMS error of IF-R AND Matlab solver after around 4M iterations.
(PDF)

Acknowledgments

The authors wish to thank to Bor Plestenjak for running the last experiment in which the comparison of the IF-R algorithm with the standard optimisation solver in Matlab was done. The authors also wish to sincerely thank to the anonymous reviewer for careful reading of the manuscript and for constructive suggestions that contributed to improve the presentation considerably.

Author Contributions

Conceptualization: DK TN JŽ.

Data curation: DK TN JŽ.

Formal analysis: DK TN JŽ.

Funding acquisition: DK TN JŽ.

Investigation: DK TN JŽ.

Methodology: DK TN JŽ.

Project administration: DK TN JŽ.

Resources: DK TN JŽ.

Software: DK TN JŽ.

Supervision: DK TN JŽ.

Validation: DK TN JŽ.

Visualization: DK TN JŽ.

Writing – original draft: DK TN JŽ.

Writing – review & editing: DK TN JŽ.

References

1. Molina D, Lozano M, Sánchez AM, Herrera F. Memetic algorithms based on local search chains for large scale continuous optimization problems: MA-SSW-Chains. Soft Comput. 2011; 15: 2201. https://doi.org/10.1007/s00500-010-0647-2
2. Moreno I, Sun C-C. Modeling the radiation pattering of LEDs. Opt Express. 2014; 16: 1808–1819. https://doi.org/10.1364/OE.16.001808
3. Kaljun D, Žerovnik J. Function fitting the symetric radiation pattern of a LED with attached secondary optics. Opt Express. 2014; 22: 29587–29593. https://doi.org/10.1364/OE.22.029587 PMID: 25606891
4. Kaljun D, Rupnik Poklukar D, Žerovnik J. Heurstic for optimization of led spatial light distribution model. Informatica. 2015; 39: 317–327.
5. Kaljun D, Žerovnik J. Developing led ilumination optics design. Papa G (Ed). Advances in Evolutionary Algorithms Research. Nova Science Publishers, Inc. 400 Oser Ave Suite 1600, Hauppauge NY, USA. 2015; 11788–361.
6. Kaljun D, Petrišič J, Žerovnik J. Using Newton's Method to Model the Spatial Light Distribution of an LED with Attached Secondary Optics. Stroj Vestn-J Mech E. 2016; 62: 307–317. https://doi.org/10.5545/sv-jme.2015.3234
7. Kaufman L, Pereyra V. A Method for Separable Nonlinear Least Squares Problems with Separable Nonlinear Equality Constraints. SIAM J Numer Anal. 1978; 15: 12–20. https://doi.org/10.1137/0715002
8. Golub GH, Pereyra V. The Differentiation of Pseudo-Inverses and Nonlinear Least Square Problems Whose Variables Separate. SIAM J Numer Anal. 1973; 10: 413–432. https://doi.org/10.1137/0710036
9. Penrose R. A generalized inverse for matrices. Math Proc Cambridge Philos Soc. 1955; 51: 406–413. https://doi.org/10.1017/S0305004100030401
10. Golub GH, Reinsch C. Singular Value Decomposition and Least Squares Solutions. Numer Math. 1970; 14: 403–420. https://doi.org/10.1007/BF02163027
11. Quateroni A, Saaco R, Saleri F. Nonlinear systes and numerical optimisation. Marsden J, Sirovich L, Antman S (Eds). Numerical Mathematics. Springer-Verlag GmbH, Tiergartenstrasse 17, D-69121 Heidelberg, Germany; 2015.
12. Yang H, Bergmans JM, Schenk TCW, Linnartz J-PMG, Rietman R.An analytical model for the illuminance distribution of a power LED. OSA 2008.
13. Herceg D, Herceg Dj. Means based modications of Newton'8s method for solving nonlinear equations. Appl Math Comput. 2013; 219: 6126–6133.
14. Thukral R. Introduction to a newton-type method for solving nonlinear equations. Appl Math Comput. 2008; 195: 663–668.
15. Ledil oy. from http://www.ledil.com/products/?y, accesed on 07-18-2016