UNIVERSIDADE FEDERAL DE JUIZ DE FORA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Aleksander Yacovenco

Directional Light Vector Estimation From a Virtual AR Object and its 2D Shadow Mask

Juiz de Fora

2023

Aleksander Yacovenco

Directional Light Vector Estimation From a Virtual AR Object and its 2D Shadow Mask

Dissertação apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Computação Gráfica

Orientador: Prof. Dr. Marcelo Bernardes Vieira

Coorientador: Prof. Dr. Rodrigo Luis de Souza da Silva

Juiz de Fora

2023

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Yacovenco, Aleksander.
Directional Light Vector Estimation From a Virtual AR Object and its 2D Shadow Mask / Aleksander Yacovenco. – 2023.
66 f. : il.

Orientador: Marcelo Bernardes Vieira
Coorientador: Rodrigo Luis de Souza da Silva
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Departamento de Ciência da Computação. Programa de Pós Graduação em Ciência da Computação, 2023.

1. Realidade Aumentada. 2. Estimativa de iluminação. 3. Reconstrução 3D. I. Vieira, Marcelo Bernardes, orient. II. Silva, Rodrigo Luis de Souza da, coorient. III. Título.

Aleksander Yacovenco

“Directional Light Vector Estimation From a Virtual AR Object and its 2D Shadow Mask

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do título de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação.

Aprovada em 19 de dezembro de 2023.

BANCA EXAMINADORA

Prof. Dr. Marcelo Bernardes Vieira - Orientador

Universidade Federal de Juiz de Fora

Prof. Dr. Rodrigo Luís de Souza da Silva - Coorientador

Universidade Federal de Juiz de Fora

Prof. Dr. Luiz Maurílio da Silva Maciel

Universidade Federal de Juiz de Fora

Prof. Dr. Alex Fernandes da Veiga Machado

Instituto Federal do Sudeste de Minas Gerais

Juiz de Fora, 11/12/2023.

Documento assinado eletronicamente por **Marcelo Bernardes Vieira**, **Professor(a)**, em 21/12/2023, às 12:33, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Rodrigo Luis de Souza da Silva**, **Professor(a)**, em 21/12/2023, às 13:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alex Fernandes da Veiga Machado**, **Usuário Externo**, em 21/12/2023, às 14:43, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Maurílio da Silva Maciel**, **Professor(a)**, em 21/12/2023, às 14:45, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Criado por 1140997, versão 6 por 1140997 em 11/12/2023 10:28:07.

Dedico este trabalho ao meu falecido avô, Jobel Pinto Teixeira

ACKNOWLEDGEMENTS

First and foremost, I'd like to thank the faculty of the post-graduate program in Computer Science of the Federal University of Juiz de Fora (UFJF), as well as every professional from the campus, from cleaners to administrative officers. It would not be possible to conclude this work if not for the hard work of every employee in the campus.

I would also like to thank CAPES for funding my research, and the National Association of Postgraduate Students (ANPG) for standing against the pay cuts of postgraduate scholarships in December 2022.

I thank my advisors Marcelo Bernardes and Rodrigo Luis for their tremendous patience and for sharing invaluable knowledge. Now I'd like to say some special thanks to friends and family in my native language, Portuguese.

À minha noiva, Laura, por seu suporte incondicional, pelo carinho e pelo companheirismo durante essa jornada de pós graduação, que trilhamos juntos, apesar de pertencermos a departamentos distintos.

Ao meu falecido avô, Jobel, por todo o apoio que me deu em vida, pelas boas memórias que compartilhamos e pelas lições de vida que eu aprendi até mesmo depois de seu falecimento.

À minha terapeuta, Sonia, por ter me ajudado a entender os meus próprios sentimentos e como lidar com eles durante um período tão conturbado, marcado não só pelas minhas experiências pessoais, mas também por uma pandemia global.

Aos meus amigos e colegas de mestrado, Arthur e Karla, pelo apoio mútuo e pela inspiração durante todo o curso.

Por fim, gostaria de agradecer à comunidade nacional de Esgrima Histórica, bem como todas as comunidades de artes marciais das quais já participei. Para além da vida acadêmica, as artes marciais sempre me deram um norte na vida, e por isso eu sou eternamente grato. Dentre todos os envolvidos, gostaria de agradecer especialmente a: Gabriel Neto, Leonardo Simões, Caíque Soares, Natália Teles, Eduarda Coppo, Otávio Nikolaus, Matheus Acha, Pedro Duarte e Hallysson Mendes.

"You cannot change fate. However, you can rise to meet it."
Mononoke Hime, 1997

RESUMO

Realidade Aumentada (RA) é uma tecnologia que permite mesclar ambientes virtuais e reais. Como RA trabalha com quadros de imagem 2D, é necessário encontrar um sistema de coordenadas ligado à perspectiva do observador, o que permite adicionar elementos 3D à cena 2D corretamente. Isso é comumente feito usando um marcador fiducial, um objeto plano conhecido que é marcado como a origem do sistema de coordenadas. Entretanto, encontrar esse sistema de coordandas não é o suficiente para criar um ambiente em que objetos virtuais são adicionados ao mundo real de forma totalmente imersiva. Existem problemas como lidar corretamente com oclusão, extensão do campo de visão, sombreamento, iluminação, dentre outros. Neste trabalho, o objetivo é lidar com o problema de iluminação.

Este trabalho apresenta um novo método para inferir a principal fonte de luz em uma cena 3D, dadas apenas entradas 2D: uma imagem de câmera e uma estimativa grosseira de sombra. É proposto um algoritmo de duas etapas, em que a primeira etapa lida com as entradas e realiza uma estimativa inicial do vetor de luz, e a segunda etapa encontra o vetor que corresponde mais adequadamente à luz direcional real da cena, dadas as condições de contorno do sistema. Analisa-se a eficácia do método proposto para encontrar o vetor associado ao melhor máximo local dentro do espaço de busca definido.

Os experimentos são realizados em ambientes virtuais e reais, e com diferentes níveis de controle sobre a cena e de informações conhecidas pelo sistema. Os resultados mostram que o método é capaz de encontrar o vetor 3D de luz direcional, bem como aprimorar a estimativa inicial grosseira da sombra.

Palavras-chave: Realidade Aumentada. Estimativa de iluminação. Reconstrução 3D.

ABSTRACT

Augmented Reality (AR) is a technology that enables merging real and virtual environments. Since AR works with 2D image frames, a coordinate system from the viewer's perspective must be set, therefore allowing the 3D virtual elements to be merged to the 2D picture correctly. This is commonly achieved by using a fiducial marker, a known planar object that is set as the origin of the new coordinate system. However, simply finding that coordinate system is not enough to create a scene where virtual objects are seamlessly merged to the real world. There are issues such as correctly handling occlusion, field of view, shading, lighting, and others. In this work, we aim to solve the lighting problem.

This work presents a novel method for inferring the main directional light in a 3D scene, given only 2D inputs, namely a camera image and a rough shadow mask. A two stage algorithm is proposed, in which the first stage handles the inputs and makes an initial estimation for the light source, and the second and main stage finds the most suitable vector corresponding to the directional light, given the system constraints. We sample a search space around the initial estimation, to which the proposed algorithm is constrained, to measure the accuracy of our method in finding the vector associated with the global maximum value of that space.

The experiments are made both in virtual and real environments, in scenes with different levels of control and known data. Results show that our method is capable of finding the 3D light vector from the 2D scene and enhancing the initial rough shadow input.

Keywords: Augmented Reality. Light estimation. 3D reconstruction.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AR	Augmented Reality
BRDF	Bidirectional Reflection Distribution Function
CNN	Convolutional Neural Network
DR	Differentiable Rendering
GAN	Generative Adversarial Network
HDR	High Dynamic Range
HMD	Head Mounted Display
IoU	Intersection over union
IRT	Inverse Ray Tracing
RGB	Red, green and blue
RGBD	Red, green, blue and depth
SH	Sphere Harmonics
SLAM	Simultaneous Localization and Mapping

SUMMARY

1 INTRODUCTION

Mobile technology has advanced at an astounding rate in the past decades. Among those advancements, Augmented Reality (AR) had a particularly significant growth, together with related technologies on which AR can rely on, such as computer vision and artificial intelligence (1). AR can be defined as a variation of Virtual Reality (VR) where the user can still see and interact with the real world environment, thus merging virtual objects and real ones in real time (2). This merge, however, can be difficult to implement with the desired graphic quality. The human eye may not be so easy to deceive, and AR scenes must have proper field of view (FoV), eyebox, angular resolution, dynamic range, depth cue, among others aspects (3).

Another challenge when dealing with AR is to correctly light the virtual objects in the scene. The light is an element of the real environment, which means the AR application must be capable of extracting the light properties of the real world and correctly place them in the virtual scene. Usually, researchers employ more specialized hardware, such as a multiple camera setup (5), depth cameras (4), or more specialized methods, such as Simultaneous Localization and Mapping (SLAM) (6), or a combination of multiple methods (7). Overall, this can be seen as a costly problem. In contrast to these approaches, we aim to achieve the same precision in light estimation by using a single camera and fiducial markers.

Fiducial markers are useful for problems where camera-object pose is desired or necessary, which is common in AR (8). Once the AR application finds the marker and sets the coordinate system for the virtual scene, it can place virtual objects in it. The marker is the referential that binds together the real and virtual environments. However, a common 2D camera and a marker alone can't extract useful data such as real world depth of lighting, hence the need for specialized hardware in most works. But instead of working with specialized hardware, the problem could be approached differently if we assume we have another input. By using a neural network capable of estimating the shadow contour of a known object, the task of retrieving the light vector becomes feasible.

In the literature, it is relatively common to find computer vision papers which aim to find shadow contours (10, 11, 12, 13), with and without the usage of neural networks. Neural networks are usually faster and less costly than previous works. However, finding the shadow contour is only the first part of the problem. We also need information about the 3D object that is projecting that shadow. By using ARShadowGAN (9), a Generative Adversarial Network (GAN) capable of estimating the shadow of an object in 2D space from its mask, we can retrieve the estimated shadow of the known 3D virtual object.

The main reason why we have chosen to work with the light vector instead of the shadow contour, as most works do, is because we envision that our approach fits a marker

based AR context, specifically. Therefore, a method that outputs the 3D light vector is more suitable than a method that estimates the shadow contour of every object in every few key frames. When we claim our method is suited to AR contexts, we need to make sure that it runs even with a moving camera and that it doesn't need to run constantly. Given that the light vector is an object within the scene, which is bound to the AR marker, we know that the directional light orientation would match that of the marker. Once the method outputs the vector, it would only need to be run again if the lighting conditions of the scene change significantly.

In order to output the 3D vector, the method must go through a number of steps. First, it retrieves the 3D points from the estimated shadow of our scene. We combine those points with points from the virtual object's surface points, creating a set of vectors. A vector can be defined from two points, so each combination of shadow and object points is used to cast a different shadow. The vector from this set that casts the shadow closest to the input shadow is assigned as the initial light vector estimation. Finally, we run an heuristic to enhance the light vector found in the last step by searching its vicinity. This search space is integrated, forming an integral image. We aim to find the best local maximum of the integral image, which corresponds to the best light vector within it.

Our main contribution in this work is the development of a method to enhance a rough shadow input and acquire the main 3D light vector of the real environment using a single 2D camera and a fiducial marker. The acquisition of the main light vector allows us to correctly set the scene lighting and project shadows. Thus, our contribution allows for a better 3D light estimation of AR scenes in less specialized mobile hardware.

1.1 PROBLEM DEFINITION

Our problem can be described as a sequence of problems that need to be solved sequentially. First, given a basic AR scene with a single marker, we need to get a pair of object and shadow. For that we use a shadow estimator, because it is able to give us the shadow of the virtual object, which we already know in 3D space. The estimator gives us a 2D shadow contour, which we then need to pass on to 3D space. These points are combined to form a set of candidate vectors for the directional light.

Finally, starting from the best candidate, the main problem is to find the light vector that projects the shadow closest to the estimated one, within a search space around the candidate. The search space is a sample of the continuous space, so it can be made into an integral image. Furthermore, the whole space is scanned so the results can be more accurately evaluated and compared. By scanning the whole space, all the results can be compared to the global maximum, and the heuristic's behavior can be evaluated in different scenarios.

1.2 OBJECTIVES

The main objective is to estimate the main light vector of the real world environment. This enables proper lighting and shadow casting of the virtual elements. Since we only employ the bare minimum hardware for this task, we believe the methods presented in this study could be fitted into the majority of lighting estimation problems in AR, especially marker-based AR. Furthermore, we demonstrate that this study has the potential to be used on standalone AR applications to produce quality lighting estimation.

2 FUNDAMENTALS

In this chapter, the fundamentals of this work are presented. The main techniques on which our method depends are inverse ray tracing (IRT) and integral imaging. Shadow estimation is not contemplated in this chapter because, even though it is present in our method, it is understood that the shadow mask is an input and not intrinsically present in the method itself.

2.1 INVERSE RENDERING

According to Yu (35), "Inverse rendering is the problem of estimating one or more of illumination, reflectance properties and shape from observed appearance (i.e. one or more images)".

Inverse Rendering is an area that has been studied for over 20 years (38). It is related to transport theory, and essentially the objective is to extract relevant illumination data by solving the transport equation:

$$L(\mathbf{r}, \omega) = L_e(\mathbf{r}, \omega) + \int_{S_i} f_r(\mathbf{r}, \omega, \omega_i) \cdot L(\mathbf{r}, \omega_i) \cdot cos(\theta)\, d\omega_i. \tag{2.1}$$

$L(\mathbf{r}, \omega)$ is the radiance at a given point $\mathbf{r}$ in direction ω, and L_e is the emittances of the light sources. f_r is the bidirectional reflection distribution function (BRDF), θ is the angle between the surface normal at $\mathbf{r}$ and ω, S_i is the hemisphere of incoming directions with respect to $\mathbf{r}$ and ω_i is an incoming direction. However, this equation can be rewritten to highlight the most important terms (38):

$$L = L_e + \hat{K} \cdot \hat{G} \cdot L. \tag{2.2}$$

The local reflection operator $\hat{K}$ maps the incident light distribution onto the corresponding exiting light distribution that results from one local reflection. The field radiance operator $\hat{G}$ transforms an exiting light distribution into the incident light distribution that results from surfaces illuminating one another.

Patow (38) states that there are four categories of problems, one for each variable of Equation 2.2. If L is unknown, we have a direct problem. But if we have some knowledge of L, there can be other types of inverse lighting problems.

If we have a complete model of the scene ($\hat{K}$ and $\hat{G}$) and some knowledge of L, for example a photograph, we can find the emittances L_e. This is an inverse lighting problem.

If we must solve the equation for $\hat{K}$, we have an inverse reflectometry problem. Depending on the imposed constraints, the problem can be subdivided into two categories:

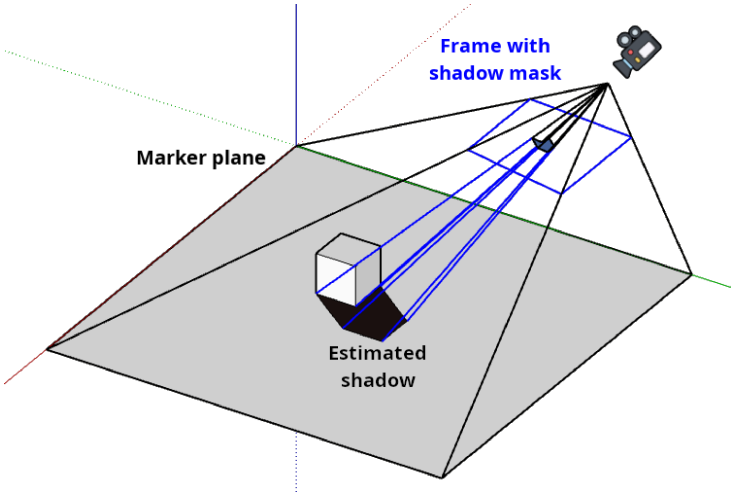


Figure 1: Visualization of how the shadow points on the plane are placed by ray casting the shadow pixels on the estimated shadow picture.

inverse texture measurement (constraints on the directional variation), or inverse BRDF measurement (spatial uniformity is assumed).

Finally, if $\hat{G}$ is unknown, the problem is classified as an inverse geometry problem. It can also be referred to as reflector design problem.

Given the constraints of our method, it can be understood as an inverse lighting problem. The scene geometry $\hat{G}$ and some of the lighting conditions L are known, and the objective is to find the position of the light source L_e. To achieve that, we employ inverse ray tracing, which is a differentiable rendering technique.

2.1.1 Differentiable rendering

According to Kato (39), in recent years, it is clear that neural networks have proven to be effective for estimating 3D data from 2D inputs. However, the training process of 3D estimation methods can be costly and the data needed for the training process is not as easily obtained as 2D data. Hence, recent approaches have leveraged 2D data and devised more fitting training processes for 3D estimation. One approach is to integrate the rendering process to the neural network pipeline.

Kato (39) defines differentiable rendering (DR) as a family of techniques that aim to obtain useful gradients of the rendering process by optimizing the process with this integration. DR allows the neural network to optimize 3D entities while dealing with 2D data.

However, as implied by (40), when the 3D geometry of the scene is known, other techniques, such as inverse path tracing or inverse ray tracing, may prove more fitting for acquiring other illumination information on the scene.

2.1.1.1 Inverse ray tracing

Inverse ray tracing (IRT) is the process of tracing the light rays from a certain region in the scene back to the point of observation (37).

IRT enables to compute the distance of 3D points in the scene relative to each other. Given a shadow mask M and a known plane P centered at an AR marker, and considering that all shadows are cast on this plane with no occlusions, each shadow pixel of M can be mapped to a point on P by ray tracing that pixel back to the camera position and intersecting it with P. This intersection marks the 3D position of that shadow pixel.

2.2 INTEGRAL IMAGE

The concept of integral image was first introduced by Crow (36) in 1984 under the name "summed area table". The original goal of this method was to enable computing the sum of pixel intensity values of a texture map in $O(1)$. However, this concept can be applied to any map from which it is intended to compute the sum of its elements.

This technique consists of taking a map and making a table in which each element is the sum of the map's elements to the left and to the top of said element. Take a 2D map M and a table T, for example. Each element $t_{x,y}$ of T is a sum of M elements:

$$t_{x,y} = \sum_{i=1}^{x} \sum_{j=1}^{y} m_{i,j}. \tag{2.3}$$

If the goal is to retrieve the sum s of an arbitrary section of M, ranging from x_1 to x_2 and from y_1 to y_2, the equation is:

$$s = \sum_{i=x_1}^{x_2} \sum_{j=y_1}^{y_2} m_{i,j}. \tag{2.4}$$

It is clear that, when implementing that equation, each sum operation will compute in $O(n^2)$ time. If the sum operation is called multiple times, this can become costly. However, with the table T, which is the integral image of M, this operation becomes $O(1)$, since each element of T is a pre-computed sum of M. So retrieving s as seen in Equation 2.4 becomes:

$$s = t_{x2,y2} - t_{x1,y2} - t_{x2,y1} + t_{x1,y1}. \tag{2.5}$$

Each element t of T is the sum of M elements in a rectangle from the first element (top left) up to the position of t. To retrieve the sum of an arbitrary range, represented by a rectangle, it is possible to subtract the undesired area from the larger rectangle. This method is illustrated in Figure 2, where the goal is to retrieve a cut of D. B and C are

subtracted from D. But B and C intersect at A, so A is summed to the final value, since it was subtracted twice.

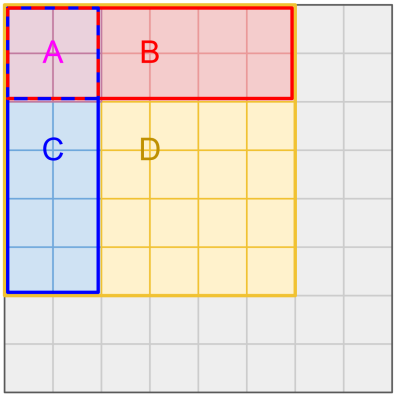
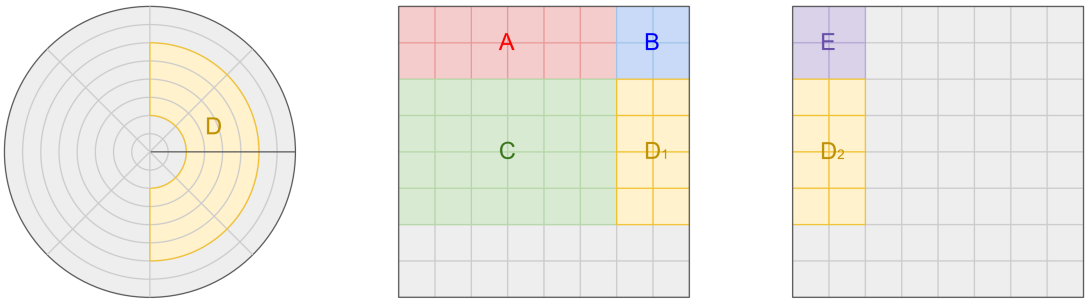


Figure 2: Visual representation of the integral image.

2.2.1 Integral image in polar coordinates

There is one more step to the integral image when working in polar coordinates. The integral image is a table and it can be represented by rectangular sections. Mapping a circular shape to a rectangle is relatively simple, except for the continuity. The integral image has a beginning and an end, but the circle doesn't.

The discontinuity is inevitable, but it is possible to overcome this problem by separating the sum in two steps: if the sum encompasses the discontinuity, the result will be the sum of the subsection before the discontinuity plus the sum of the subsection after the discontinuity. The discontinuity is the imaginary line between the end of the table and the beginning as if it was a circle. This can be seen more clearly in Figure 3.



(a) (b) (c)

Figure 3: Integral image section when dealing with the discontinuity. In (a) the discontinuity is highlighted in the polar representation. The sum regarding area section D is divided into D_1 and D_2, respectively in (b) and (c), which are square representations of the integral image.

3 RELATED WORKS

Light estimation has been a recurrent challenge in the literature for over 20 years (18). The approaches may greatly diverge, but the overall goal is still the same in most papers. The methods retrieve either the directional light vector, point light positions in 3D space or the spherical harmonics (SH) representation of the whole illumination conditions of the scene.

Historically, the first approaches to retrieve illumination information of the scene are mostly based on the scene geometry (26, 27, 28, 30, 31). They rely on known points of a controlled scene as well as one or more 2D images to be able to estimate the main light vector using triangulation or similar methods. Early approaches also make use of image processing techniques, such as image filtering and thresholding, to extract useful features from the image.

Since some geometry information is needed for these methods to work, RGBD cameras were also employed to some extent (29, 32, 33). RGBD cameras are a good solution for estimating the scene geometry because they provide depth information for the whole scene. The main issue with RGBD cameras is that, even though they replace the need to work on a tightly controlled environment, they are still a prerequisite and the method still needs a specialized form of input.

Over the past few years, however, there was an exponential increase in popularity of deep learning approaches in computer vision. Deep learning is now the current paradigm, and most recent works employ some form of neural network in their methods (4, 19, 20, 21, 22, 23, 24, 25). Deep learning methods provide better experimental results than previous approaches and can work in less constrained environments. For example, the neural networks can usually estimate some information of the scene geometry, which means this information doesn't need to be present in the initial input. Most deep learning approaches use a single monocular image as input.

3.1 GEOMETRY BASED LIGHT SOURCE ESTIMATION

When extracting 3D illumination information from a scene, there must be either some information or estimation of a known 3D geometry or a certain set of initial assumptions and conditions. Even the deep learning approaches that can estimate 3D points from 2D data can only do so because they are given labeled 3D data during the training step. The approaches that don't rely on deep learning, however, need a different strategy to ensure this data is present. Usually they have a constrained scenario with known geometry to work with.

Cao (26), for example, relies on two known 3D points to extract the directional light from the scene. Along with the projected shadow of these points and two 2D pictures

taken from different angles, they can estimate not only the light source, but also the camera parameters. They match at least four points between the two input pictures. These points are used to triangulate the sun's position. The sun's position is then further refined using linear optimization, and the output is the 3D light vector.

Koc's approach (27) also works with a known geometry, but only shading information and no projected shadows. In their approach, they chose to use face points as their known geometry. Using OpenCV's face detection algorithm (34) and knowing the camera and face orientation relative to each other, they estimate the light source based on the shading of detected faces in the scene. They assume the face is a Lambertian surface and that the only light present is the sunlight. The azimuth and zenith angles of the sunlight can be estimated with the shading data extracted with image filtering. Finally, the light coordinates are mapped from spherical coordinates to real world coordinates by using gyroscope information taken from the same mobile device that took the input picture.

Wang (28) works with an arbitrary object of known geometry. His method searches for critical points in the geometry and for shadows cast from those points, excluding occluded and partially occluded shadows. Wang defines critical points as follows: "Given an image, let $\mathbf{L}_i$, $i = 1, 2, ...$, be the light sources of the image. A point in the image is called a critical point if the surface normal at the corresponding point on the surface of the object is perpendicular to some light source $\mathbf{L}_i$.". These points are used to calculate the critical boundaries, which are a group of critical points corresponding to one real light. These boundaries are dynamically adjusted in small steps so that spurious boundaries are excluded. After those adjustments, the real lights, defined by the author as a subtraction of two virtual light vectors separated by a critical boundary, are calculated, and the output is a set of unit 3D light vectors in spherical space.

The works of Nguyen (30) and Lopez (31) don't require any known 3D geometry nor do they rely on deep learning. They do, however, need a partially known object present on the scene. Nguyen and Lopez assume this object is convex and acts as a light probe. Nguyen states that light probes, also known as environment maps, were the most common approach to light estimation in the literature at the time their article was published. Their method, however, does not estimate the object's silhouette. This input is given by the user. With the picture and the object's silhouette available, their method finds the number of light sources, their azimuth and zenith angles and their intensity. This is done by exploring the object in 2D space for local maximums and minimums in illuminance. It is important to note, however, that Nguyen admits that the average error of this method may be greater than that of other approaches of its time. The most significant advantage of this method over others is its concise input and a relatively low number of initial constraints. Lopez, however, states that their method surpasses the state of the art of the time. Even though their methods are very similar, almost identical in some aspects, there are key differences in what image filtering techniques they employ and how they search for local maximums

and minimums in illumination.

Similarly, Gruber (33) also works with virtual light probes. However, his method needs an additional input compared to Nguyen and Lopez: the depth map. Gruber uses the Microsoft Kinect's RGBD camera to acquire the necessary inputs. The depth map allows for a completely different approach. Instead of studying the silhouette of the object selected as the light probe, as Nguyen and Lopez do, Gruber starts his method by computing the SH coefficients and mapping them to the points acquired by the depth sensor. He also excludes the samples that would be occluded in the 2D picture. This is done by ray casting the light at that position in space. The method outputs the SH lighting instead of a set of light vectors.

Another work that relies on RGBD data is Chotikakamthorn's (32). Their approach consists of detecting the cast shadows of the object retrieved with the depth sensor, segmentating and clustering the shadows, and solving the equation that returns the shadow centroids to determine the direction of the light sources that match the clustered shadows. They assume the shadows are all cast on the same plane, and use a cost function to decide whether the cast shadow belongs to the known object. This cost function evaluates the set of points of the object silhouette and the set of points belonging to the shadow. However, they do not further describe the function, they only state that, ideally, all points in the sets should match, but in practice, that is not the case. Their output is a set of 3D light vectors. Each shadow cluster is affected by a single light vector, so the number of clusters and vectors must be the same.

It is also possible to work with RGBD probe-less data, which is Boom's case (29). In their work, they assume all surfaces are Lambertian and the albedo is uniform in contiguous space. They also chose to ignore cast shadows and work with attached shadows, or shading, alone. They state that cast shadows may be difficult to handle because there may be a shadow cast by an object outside of the observed scene. Their goal is to estimate the main directional light of the scene. They initially assume that the light source matches the camera position, and they move the light source over a 3D grid searching for a minimum error value. The error value is the difference between the original image and the reconstructed image intensity.

3.2 DEEP LEARNING BASED LIGHT SOURCE ESTIMATION

In the past few years, deep learning has been employed in many computer graphics challenges besides light estimation, such as 3D reconstruction, image segmentation and simultaneous localization and mapping (SLAM). Today, deep learning is one of the most common approaches for light estimation in the literature.

Kan's method (4) retrieves the main light source present in the scene. Their method relies on RGBD input and employs a residual convolutional neural network (ResNet) to

extrapolate the light position. The training step was done with a synthetic dataset composed of 23,111 images. They also tried using a real dataset composed of 5,650 RGBD images, but the network converged better with the synthetic dataset. The images in the synthetic dataset are composed of a basic scene with an object of known geometry that casts a shadow on a plane. The images used in the experiments are relatively similar to the synthetic ones: there is a large table illuminated by a single light, and on top of it there is an AR marker and a real object side by side. This means the network was tailored to work on indoor single light scenes.

Marques' method (19) is similar to Kan's in the sense that it uses a network to extrapolate the light position of the scene. Marques, however, focuses on the usage of a head mounted display (HMD). They assume the user's hand will be visible for most of the time in this scenario. First, the hands mask is extracted from the input image, and then a ResNet is used to estimate the light source based on the segmented hands. More specifically, the most significant data for the CNN to work with is the shading information present in the hands. They assume there is a main light source in the scene and output a single light vector representing that light source estimation.

Liu (20) also retrieves the main light vector of the scene, but their method is composed of more steps in comparison to Marques and Kan. First, a network is employed to extract the foreground objects and another to extract the visible cast shadows. The objects and shadows are divided in two sets, which need to be correctly paired, and only then the light direction is estimated given the object and shadow pairs. From the pairing step onwards, the networks are not needed anymore. The pairing is done by finding the maximum value of a similarity function and the light estimation is done by triangulating key points of the objects and their matched shadows. Also, when calculating the light elevation angle, the authors assume the light source is a parallel light, which means the shadow and object width should be the same. Therefore, it is possible to estimate the object height given its aspect ratio. The height is essential when triangulating the light source in 3D space.

The approaches of LeGendre (21), Sun (22), Wang (23), Liu (24) and Marques (25) are very similar: they retrieve the SH light from a single monocular image using one or more CNNs. All except Sun's work retrieve the HDR illumination. What differs the most in these studies is the architecture of their neural networks, and the dataset construction. In all cases, the dataset construction is a key part of the method, and it must be robust enough to enable the network to handle a significant amount of data. Another thing that differs these methods from one another is how their problems are modeled. Wang's work, for example, lists how each important feature in the training data is segmented and how this impacts the architecture of the network. It is important to note that LeGendre's method is fundamental to the others, since it is cited in most of the other works.

Our method differs from others in the literature because it does not compute the 3D light vector with a neural network. However, a neural network is used to generate the method's input shadow mask, but any other method can be used for this task. We improve an initial directional light vector estimation by using a recursive approach to subdivide the space around it.

3.3 COMPARISON

A table was devised to make it easier to note the advantages and disadvantages of the methods seen in this chapter. Our method is listed at the end. Note that, in the last column, our method is listed among the ones that do not use deep learning. It does, however, use a pre-trained model at the very beginning to acquire the object and shadow pair. This is considered the method's input, and the method itself is not dependant on deep learning.

Table 1 – Comparison of light source estimation methods

Method	Input	Output	Approach	Implementation details
Kan (4)	RGBD image	3D light vector	Deep learning	Network extrapolates light data from RGBD 3D points.
Marques (19)	2D image	3D light vector	Deep learning	Network extrapolates light data from user hands in front of HMD device.
Liu (20)	2D image	3D light vector	Deep learning	Shadow and object shapes are estimated and paired.
LeGendre (21)	2D image	SH lighting	Deep learning	HDR estimation from LDR data.
Sun (22)	2D image	SH lighting	Deep learning	Network predicts SH coefficients.
Wang (23)	2D image	SH lighting	Deep learning	Network receives a panorama and outputs an HDR skybox with SH data.
Liu (24)	2D image	SH lighting	Deep learning	Network decomposes input and estimates SH data.
Marques (25)	2D image	SH lighting	Deep learning	Network is trained with HDR data to extrapolate SH data.
Cao (26)	Two 2D images	3D light vector	Geometry	Two pictures of the same scene are taken from different angles for triangulation.
Koc (27)	2D image	3D light vector	Geometry	Shading on a 2D face present is detected with OpenCV filters.
Wang (28)	2D image; known object	Set of 3D light vectors	Geometry	Light vectors in the scene are estimated from surface lighting of a of known object.
Boom (29)	RGBD image	3D light vector	Geometry	Scene is rendered multiple times and best light vector found is outputted.
Nguyen (30)	2D image; object mask	Set of 3D light vectors	Geometry	Reflectance points are extrapolated from their corresponding normals.
Lopez (31)	2D image; object mask	Set of 3D light vectors	Geometry	Silhouette is transformed into a light probe by removing texture and noise. Reflectance points are extrapolated from corresponding 3D normals.
Chotika-kamthorn (32)	RGBD image	Set of 3D light vectors	Geometry	Shadow silhouette is acquired from RGBD data and directional light is traced from 3D object to centroid of shadow.
Gruber (33)	RGBD image	SH lighting	Geometry	Depth map acquired with Kinect's RGBD camera is used to compute the SH coefficients of the image.
Ours	2D image; shadow mask	3D light vector	Deep learning*	Network extrapolates 2D shadow mask. Scene is rendered multiple times and best available vector in search space is chosen.

*our method does not use deep learning directly. Currently, the object's shadow is estimated by a neural network.

4 PROPOSED METHOD

In this chapter, we discuss our proposal to solve the problem of light vector estimation. To ensure our method works properly, we need to enforce the following conditions: we assume there is a single directional light acting on a known virtual object; the shadow cast by that directional light is projected on a single plane; the plane is coplanar with the fiducial marker's plane; both the shadow and the virtual object are within the visible frame and not occluded by any other objects; finally, we assume the object is placed exactly on top of the fiducial marker.

To begin the process, our method needs two inputs: a 2D frame of the scene with the fiducial marker present; and the 3D virtual object. These inputs are then passed onto a shadow estimator and an AR system. The AR system is responsible for the 3D scene reconstruction and it provides important data such as the marker's 3D position and orientation, as well as the camera position relative to the marker. The shadow estimator is responsible for estimating the shadow in 2D space. It retrieves the initial shadow estimation M, which is a shadow mask. Since we know the shadow is cast on the same plane where the object is placed, which is the marker's plane, and will be denoted as plane P, we are able to retrieve that shadow's 3D points from that 2D estimation. Those 3D points of the shadow compose a set that will be denoted as B.

The virtual object and its cast shadow in 3D space are used to make the initial light vector estimation. Essentially, we separate some points over the object's surface and all shadow points in two sets, respectively A and B, and those points are combined to make vectors that start somewhere on the object's surface and end somewhere on the cast shadow. This combination set C can be expressed as $C = \{\mathbf{c_0}, \mathbf{c_1}, ..., \mathbf{c_k}\}, k = m \cdot n$ where $\mathbf{c_i} = \mathbf{a}_{\lceil \mathbf{i/n} \rceil} - \mathbf{b_i} \mod \mathbf{n}$. m, n and k are respectively the number of elements in sets A, B and C. Each vector in this new set C is a valid candidate for an initial directional light vector $\mathbf{v}$, and $\mathbf{v}$ is updated to match the vector $\mathbf{c_i}$ which casts the shadow that covers the most area compared to the initial shadow estimation M.

We cannot guarantee this estimation is the best available since we have worked with discrete points to find $\mathbf{v}$. Therefore, to further improve the directional light estimation, the preliminary vector $\mathbf{v}$ corresponding to that initial estimation is set as the center of a radial search space S in which we search for the best available directional light vector $\mathbf{v}'$. We repeat this process recursively by diminishing radiuses until we reach the limit of repetitions or until the directional light can no longer be improved.

Figure 4 provides an overview of our method's steps and data flow. The colored rectangles and all processing steps are explained in the following sections.

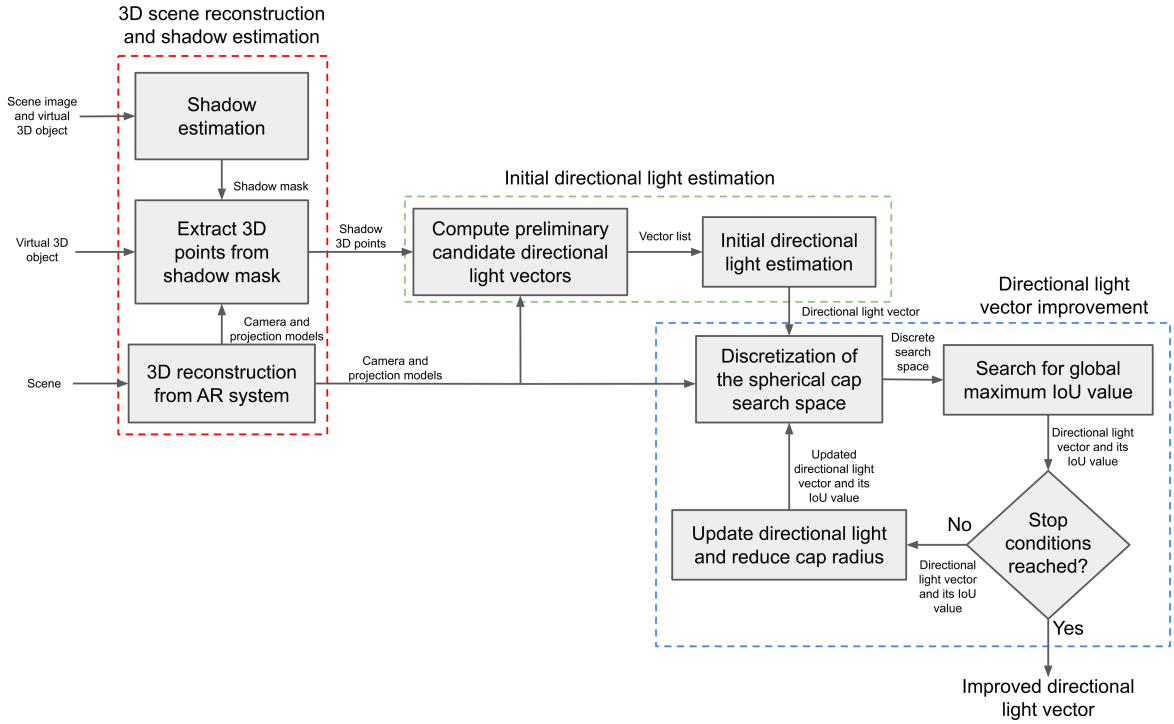


Figure 4: Diagram indicating the steps of the method and data flow.

4.1 SCENE RECONSTRUCTION AND SHADOW ESTIMATION

This is the first step before the main method can take place. Our method was designed to work with a shadow and the object projecting it in 3D space, but initially only the scene with the fiducial marker and a 3D object are available. These inputs are used to acquire an estimation of the shadow cast by the known object in 3D space. As previously mentioned, this object is exactly on top of the marker, which must be present in the scene, and cannot be occluded. Also, both the object and its estimated shadow must fit the camera frame. We also assume the shadow is cast exactly over the marker's plane.

First, the 2D frame of the scene with the marker present is passed onto the AR system. The AR system is responsible for retrieving the marker's position and orientation relative to the camera and for mapping virtual objects from 3D to 2D space. We place the 3D virtual object on top of the marker. This enables us to retrieve both the 3D scene with the virtual object and the marker's plane and the 2D frame of the scene with the virtual object present.

With that, a 2D mask of the virtual object is produced and passed onto the shadow estimator, alongside the 2D frame of the scene with the object already present. The shadow estimator then outputs a 2D shadow mask, which is mapped back to 3D space. This mapping becomes trivial because we know beforehand the plane where the shadow is cast.

This data pair, the virtual object and its shadow in 3D space, is then used to compute the preliminary candidate directional light vectors, which are discussed in Section 4.2.

4.1.1 3D reconstruction from AR system

The most essential feature in any marker based AR system is to identify a marker and retrieve its position and orientation relative to the camera. Any AR system could be employed in this step, as long as it is marker based. In our method, we assume the marker to be at ground level and parallel to the ground. In other words, the marker's center defines the origin of the plane where the shadow is projected. We need the camera, the plane position, and the orientation relative to each other to reconstruct the 3D scene. The camera and the plane are used to map points from 3D to 2D and vice versa.

When the virtual object is placed on top of the marker, we can compute: 1) the 3D virtual object in camera's coordinates; 2) the 2D image with the virtual object inserted in the scene from the estimated camera's perspective; 3) the virtual object's 2D mask by coloring it white on a black background. The object's mask is needed for the shadow estimation.

4.1.2 Shadow estimation

Shadow estimation is not a new problem in the literature. Other authors have already covered this problem or similar ones (4, 10, 11, 12). Shadow estimation is not the focus of our method either. In this step we employ a readily available shadow estimator to retrieve the 2D shadow mask of an object, given the picture of the scene with the virtual object inserted and the 2D virtual object mask.

The particular shadow estimator chosen in our study, ARShadowGAN (9), outputs the estimated shadow mask and only needs the image with the virtual object present and the object's mask as input. It was chosen because its inputs are easy to handle and because it has a readily available pre-trained model. But other shadow estimators could be used instead.

For any shadow estimator, the shadow will inevitably have some level of noise. ARShadowGAN provides a gray scale image where black pixels are background and white pixels form the estimated shadow regions. The closest the pixel is to white, the closest is the confidence that it is indeed a shadow point. We employ Otsu's thresholding method (14) to get a binary mask. Next, we employ Suzuki's algorithm (15) to find the shadow contours. We then compute which contour of the multiple contours found has the highest average intensity in the original shadow estimation. A new shadow mask is computed with the selected contour and its inner pixels marked as white, and the rest of the pixels marked as black.

4.1.3 Extract 3D points from shadow mask

For the final step before the main part of the method, the shadow needs to be passed from 2D space to 3D space. This is done by mapping the white pixels of the shadow mask to the corresponding points on the marker's plane, given the camera and the plane's orientation and position.

4.2 INITIAL DIRECTIONAL LIGHT ESTIMATION

In this step, the 3D object and its 3D estimated shadow are already available. From here, we start our main contribution, which is composed of multiple steps. First, we need a set of candidate directional light vectors. We compare the shadows produced by each vector with the initial shadow estimation mentioned in Section **4.1.2**. Then, the best directional light vector is chosen and the next and final step of the method can take place.

In principle, if there is an object projecting a shadow on a plane, the directional light vector can be computed by finding the correct correspondence of a point on the object's surface and a point on the shadow. This is valid, of course, if we assume a continuous space and that the shadow is perfectly cast. With discrete objects and imperfect shadow estimation, the directional light vector cannot be easily retrieved by looking for correspondences. However, it is still a reasonable starting point.

After acquiring the candidate directional light set, we need a criterion to compare each vector in the set and decide objectively which one is a reasonable initial guess. Every candidate vector provides a shadow of the virtual object over the marker's plane. We propose to evaluate them based on how they intersect the estimated shadow. A suitable directional light vector must maximize the area of intersection but also minimize the area of union. In other words, the best candidate vector generates a shadow that best fits the estimated shadow mask.

4.2.1 Compute preliminary candidates

We have discussed in the beginning of Chapter 4 how we acquire the 3D shadow points set B. We propose to combine these points with a set of points A sampled from the virtual object's surface. More specifically, We connect all the points sampled from the virtual object's surface to all the 3D points from the shadow mask, forming set C. The list of candidates are the unit vectors from this list of pairwise connections.

To form the set of points from the virtual object's surface, we cannot possibly acquire every point on the object's surface. A sparse set of points can result in a coarse initial directional light vector. A dense set would generate redundant candidates and increase time cost. We propose that the set of points over the virtual object's surface be composed of its vertices, edges midpoints and faces midpoints.

4.2.2 Initial directional light estimation

Each vector in the candidate set is used to cast a shadow of the virtual object over the marker's plane. These shadows are then compared to the initial estimated shadow mask. The vector that casts the shadow that is closest to the estimated shadow mask is selected as the initial directional light vector. Thus, we need to objectively measure the similarity of two shadow masks.

If we define the estimated shadow mask (Sec. **4.1.2**) as the correct estimation, we can label the pixels of the shadow mask obtained from a candidate vector as: false positive (FP), false negative (FN), true positive (TP) and true negative (TN). True positive is when the two shadows masks overlap. True negative is when there is no shadow pixels in either mask. False positive is a pixel in the candidate shadow mask marked as shadow but not marked as shadow in the estimated shadow mask. False negative is a pixel not marked as shadow in the candidate shadow mask but marked as shadow in the estimated shadow mask. These categories can be seen more clearly in Figure 5.

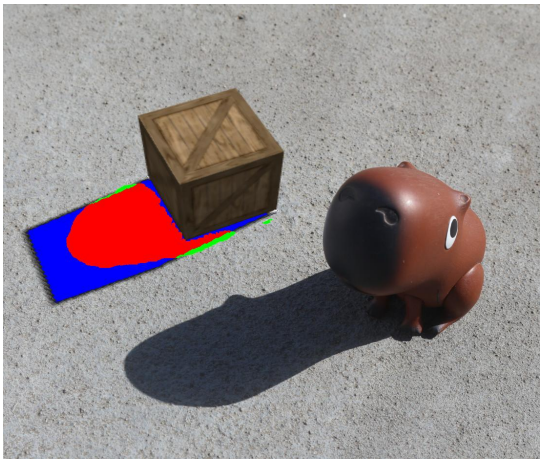


Figure 5: Visualization of a candidate shadow projected by the method compared to the estimated shadow. The red portion is TP and represents the intersection. The green portion is FN and represents the estimated shadow. The blue portion is FP and represents the candidate shadow. The rest of the picture, which wasn't highlighted, is TN.

Assume the shadows are sets of points in the two masks. Given that true negatives are essentially background, they can be disregarded in this context. True positives, false positives and false negatives represent the union of the two shadow sets. True positives are the intersection. Thus, we use the intersection over union (IoU) as comparison criterion:

$$IoU = \frac{TP}{TP + FP + FN}. \tag{4.1}$$

4.3 IMPROVING THE DIRECTIONAL LIGHT VECTOR

It is possible to search the vicinity of the initial directional light vector estimation $\mathbf{v}$ to find a vector that improves the IoU criterion. Consider a unit sphere S containing

the unit vector $\mathbf{v}$. We define the search space as the sphere cap $S'_{\mathbf{v}}$, over the unit sphere S, centered at $\mathbf{v}$ with azimuthal angle $\phi = \alpha$, where α is a parameter that defines the size of the cap. A sample of this cap is seen in Figure 6, inspired by Ureña's work (17). We parametrize $S'_{\mathbf{v}}$ using a local coordinate system $F = (\mathbf{i}, \mathbf{j}, \mathbf{k})$:

$$\mathbf{i} = \frac{\mathbf{z} \times \mathbf{v}}{||\mathbf{z} \times \mathbf{v}||}, \quad \mathbf{j} = \mathbf{i} \times \mathbf{v}, \quad \mathbf{k} = \mathbf{v}, \tag{4.2}$$

where vector $\mathbf{z}$ is a unit vector in the positive z axis. Note that when $\mathbf{v}$ is equal to either $\mathbf{z}$ or $-\mathbf{z}$, we have to use canonical coordinates, because $\mathbf{z} \times \mathbf{v} = 0$. Having defined the coordinate system, we can proceed to the parametric equation of $S'_{\mathbf{v}}$:

$$S'_{\mathbf{v}} = \{(i, j, k) \mid i = \sin\phi \cdot \cos\theta,\ j = \sin\phi \cdot \sin\theta,\ k = \cos\phi,\ 0 \leq \phi \leq \alpha, 0 \leq \theta \leq 2\pi\}. \tag{4.3}$$

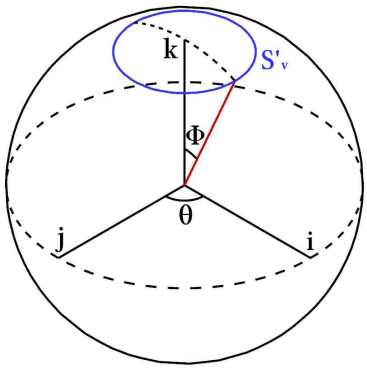


Figure 6: Sphere cap $S'_{\mathbf{v}}$ in F coordinates.

Having defined the search space $S'_{\mathbf{v}}$, we need a function to retrieve the IoU value of a given point within $S'_{\mathbf{v}}$. Consider we have a smooth function, $\mu_{\mathbf{v}}$, to compute IoU in continuous space. $\mu_{\mathbf{v}}$ can be integrated to acquire the total IoU value w of any area region over the cap as:

$$w = \int_{\theta_1}^{\theta_2} \int_{\phi_1}^{\phi_2} \mu_{\mathbf{v}}(\sin\phi \cdot \cos\theta, \sin\phi \cdot sin\theta, \cos\phi)\, d\theta\, d\phi, \tag{4.4}$$

where θ_1, θ_2, ϕ_1, ϕ_2 define a sector where we want to find a better directional light vector. The IoU value w represents the density of IoU in the sector. Higher the w, higher the chances to find a better directional light vector. Notice that the IoU function $\mu_{\mathbf{v}}$ is not convex and there may be multiple local maxima over the sphere cap. Unfortunately, function $\mu_{\mathbf{v}}$ cannot be solved in continuous space. We need to proceed with a numerical solution for the integration (Eq. 4.4) and a method to search for a IoU maximum corresponding to a better directional light vector.

4.3.1 Discretization of search space

Firstly, the sphere cap must be sampled. Consider a $m \times n$ matrix C. m and n are the sample resolution parameters. To make C a sampling of $S'_{\mathbf{v}}$, we use Rodrigues' rotation formula (16) to rotate twice around $\mathbf{v}$, first on ϕ from 0 to α and then on θ from 0 to 2π. The ranges $[0, \alpha]$ and $[0, 2\pi]$ are mapped to $A = \{\frac{1}{m} \cdot \alpha, \frac{2}{m} \cdot \alpha, ..., \frac{m-1}{m} \cdot \alpha, \alpha\}$ and $B = \{\frac{1}{n} \cdot 2\pi, \frac{2}{n} \cdot 2\pi, ..., \frac{n-1}{n} \cdot 2\pi, 2\pi\}$, respectively. This sampling method is illustrated by Algorithm 1.

Algorithm 1 Sphere cap sampling

Input: center vector $\mathbf{v}$, opening angle α, matrix dimensions m and n
Output: discrete search space C
Require: $\mathbf{v}$ is a unit vector; $\alpha, m, n > 0; m, n \in \mathbb{N}$

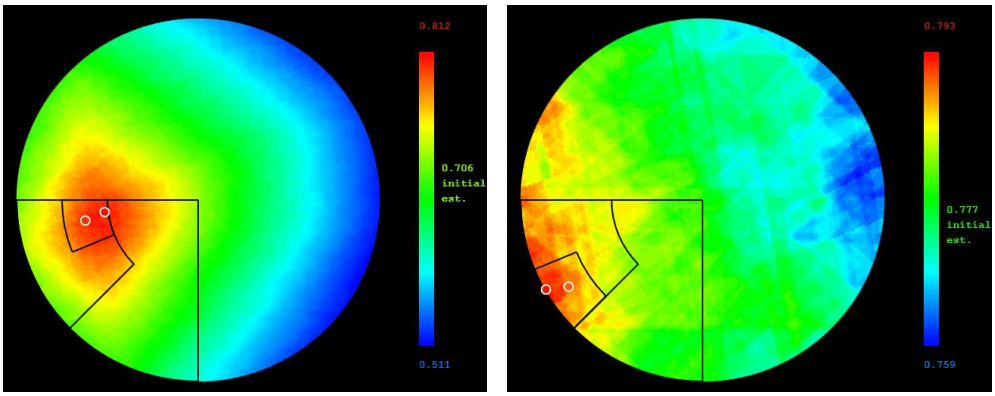
```
function sphereCapSampling(v, α, m, n)
    let C be a m × n matrix
    let u be a unit vector in ℝ³ perpendicular to v
    for i ← 0 to m do
        c_{i,0} ← v · cos(i·α/m) + (u × v) · sin(i·α/m) + u · (u · v) · (1 − cos(i·α/m))
        for j ← 1 to n do
            c_{i,j} ← c_{i,0} · cos(j·2π/n) + (v × c_{i,0}) · sin(j·2π/n) + v · (v · c_{i,0}) · (1 − cos(j·2π/n))
        end for
    end for
    return C
end function
```

4.3.2 Search for a maximum IoU value

Since the search space $S'_{\mathbf{v}}$ is sampled, pursuing the individual vector with highest IoU within it does not necessarily mean it is the best vector in continuous space. We propose to evaluate the area around the highest IoU values and not just the sampled points individually. We partition the cap in sectors and pick the one whose IoU integration per unit area is the highest. The sector is subdivided a number of times equal to parameter s. After the final subdivision, we sample a smaller cap around the section, and repeat the process recursively. This subdivision is demonstrated in Figure 7.

The number of recursions r is also a parameter. For each recursion, the new cap's center point $\mathbf{v}'$ is equal to the previous section center and its radius α' is equal to the distance from its center to its furthest vertex. This ensures the new cap encompasses the previous section and provides some leniency in contrast to using the smallest enclosing circle.

Algorithm 2 provides an overview of how the method computes the search space $S'_{\mathbf{v}}$ and searches for the maximum value g. Each point of the sphere cap $S'_{\mathbf{v}}$, when connected to the origin, represents a vector, and each vector is associated with an IoU value, as mentioned in Section **4.2.2**. Function computeIoU$(M, \mathbf{v})$ is the discrete implementation



(a) (b)

Figure 7: Top view of the subdivision of the sphere cap $S'_{\mathbf{v}}$. Parameters in (a) are: $m = 257, n = 513, \alpha = 9.0, r = 1, s = 3$ and in (b) are: $m = 257, n = 513, \alpha = 3.0, r = 2, s = 3$

Algorithm 2 Enhance directional light vector

Input: directional light estimation vector $\mathbf{v}$; opening angle α; number of resamples r; number of subdivisions s; matrix dimensions m and n; shadow mask M
Output: improved directional light estimation vector $\mathbf{v}'$
Require: $r, m, n > 0; r, s, m, n \in \mathbb{N}$

function ENHANCEDIRECTIONALLIGHT($\mathbf{v}$, α, r, s, m, n, M)
$\mathbf{v}' \leftarrow \mathbf{v}$

// Compute sampling C and its quantization Q
let C and Q be $m \times n$ matrices
$C \leftarrow$ SPHERECAPSAMPLING($\mathbf{v}, \alpha, m, n$)
for each vector $c_{i,j}$ in C **do**
$q_{i,j} \leftarrow$ COMPUTEIOU($M, c_{i,j}$)
end for

// Compute integral image T
let T be a $m \times n$ matrix
for each element $t_{i,j}$ in T **do**
$t_{i,j} \leftarrow q_{i,j} - q_{i-1,j} - q_{i,j-1} + q_{i-1,j-1}$
end for

// Search for vector with best IoU within the most dense IoU section
$\mathbf{v}', \alpha' \leftarrow$ RETRIEVEBESTSECTOR($c_{0,0}, s, C, T, 0, m, 0, n$)

// If reached recursion limit, return this vector
if $r > 0$ **then**
return $\mathbf{v}'$
else
// Reduce radius and repeat until conditions are met
return ENHANCEDIRECTIONALLIGHT($\mathbf{v}'$, α', $r - 1$, s, m, n, M)
end if
end function

of the $\mu_{\mathbf{v}}$ function to acquire the IoU value of a vector by comparing the shadow mask produced by it to the first shadow mask estimation M. This implementation is seen in Algorithm 3.

Algorithm 3 Compute IoU of a given vector

Input: vector $\mathbf{v}$; shadow mask M
Output: IoU value q
Require: Assume M and R have the same width and height

```
function computeIoU(M, v)
    let d be a directional light source
    let d_pos ← v
    let R be the shadow mask of the rendered scene
    let p be a shadow pixel
    let TP, TN, FP, FN ← 0
    for i ← 0 to M_width do
        for j ← 0 to M_height do
            if m_{i,j} = p and r_{i,j} = p then
                TP = TP + 1
            end if
            if m_{i,j} ≠ p and r_{i,j} = p then
                FP = FP + 1
            end if
            if m_{i,j} = p and r_{i,j} ≠ p then
                FN = FN + 1
            end if
            if m_{i,j} ≠ p and r_{i,j} ≠ p then
                TN = TN + 1
            end if
        end for
    end for
    return TP / (TP+FP+FN)
end function
```

We search for the maximum IoU by using the IoU integral computed for a set of sectors. Each sector comprises a area region inside the cap. In this work, the cap is partitioned in sectors regularly distributed in $\phi, \theta \in [0, \alpha] \times [0, 2\pi]$. By using an integral image of the sampled cap $S'_{\mathbf{v}}$, we can easily compute the IoU integral for each sector in O(1). We look for the sector with highest integrated IoU integral per unit area, i.e. the sector with higher average IoU. We assume that this sector is likely to contain the local maximum IoU g.

Algorithm 4 recursively searches for an improved local maximum IoU value g. It subdivides the area into 4 sectors and retrieves the one with highest average IoU. It loops until the IoU value g' does not improve or until the maximum number of recursions is reached.

Algorithm 4 Retrieve center vector of the most dense subsection from integral image

Input: center of section vector $\mathbf{v}$; number of subdivisions s; discrete search space C; integral image T; boundaries $\phi_1, \phi_2, \theta_1, \theta_2$

Output: center vector of most dense subsection $\mathbf{v}'$, opening radius α'

Require: $\mathbf{v}$ is a unit vector; $\delta, \phi_1, \phi_2, \theta_1, \theta_2 \in \mathbb{N}; \phi_2 > \phi_1; \theta_2 > \theta_1$

function RETRIEVEBESTSECTOR($\mathbf{v}, s, C, T, \phi_1, \phi_2, \theta_1, \theta_2$)

// Divide current section in 4, retrieve subsection with highest average IoU

$\overline{\phi} \leftarrow \lfloor \frac{\phi_1+\phi_2}{2} \rfloor$

$\overline{\theta} \leftarrow \lfloor \frac{\theta_1+\theta_2}{2} \rfloor$

$b_0 \leftarrow t_{\overline{\phi},\overline{\theta}} - t_{\phi_1,\overline{\theta}} - t_{\overline{\phi},\theta_1} + t_{\phi_1,\theta_1}$

$b_1 \leftarrow t_{\phi_2,\overline{\theta}} - t_{\overline{\phi},\overline{\theta}} - t_{\phi_2,\theta_1} + t_{\overline{\phi},\theta_1}$

$b_2 \leftarrow t_{\overline{\phi},\theta_2} - t_{\phi_1,\theta_2} - t_{\overline{\phi},\overline{\theta}} + t_{\phi_1,\overline{\theta}}$

$b_3 \leftarrow t_{\phi_2,\theta_2} - t_{\overline{\phi},\theta_2} - t_{\phi_2,\overline{\theta}} + t_{\overline{\phi},\overline{\theta}}$

$b_{max} \leftarrow \frac{\max(b_0,b_1,b_2,b_3)}{(\overline{\phi}-\phi_1)\cdot(\overline{\theta}-\theta_1)}$

let $\phi'_1, \phi'_2, \theta'_1, \theta'_2$ be the boundaries of b_{max}

$\overline{\phi}' \leftarrow \lfloor \frac{\phi'_1+\phi'_2}{2} \rfloor$

$\overline{\theta}' \leftarrow \lfloor \frac{\theta'_1+\theta'_2}{2} \rfloor$

$\mathbf{v}' \leftarrow c_{\overline{\phi}',\overline{\theta}'}$

// If reached subdivision limit, return this vector and the new opening radius

if $s > 0$ **then**

$\alpha'_1 \leftarrow \arccos\left(\frac{\mathbf{v}' \cdot c_{\phi'_1,\theta'_1}}{|\mathbf{v}'| \cdot |c_{\phi'_1,\theta'_1}|}\right)$

$\alpha'_2 \leftarrow \arccos\left(\frac{\mathbf{v}' \cdot c_{\phi'_1,\theta'_2}}{|\mathbf{v}'| \cdot |c_{\phi'_1,\theta'_2}|}\right)$

$\alpha'_3 \leftarrow \arccos\left(\frac{\mathbf{v}' \cdot c_{\phi'_2,\theta'_1}}{|\mathbf{v}'| \cdot |c_{\phi'_2,\theta'_1}|}\right)$

$\alpha'_4 \leftarrow \arccos\left(\frac{\mathbf{v}' \cdot c_{\phi'_2,\theta'_2}}{|\mathbf{v}'| \cdot |c_{\phi'_2,\theta'_2}|}\right)$

$\alpha'_{max} \leftarrow \max(\alpha'_1, \alpha'_2, \alpha'_3, \alpha'_4)$

return $\mathbf{v}', \alpha'_{max}$

else

// Otherwise, continue search

return RETRIEVEBESTSECTOR($\mathbf{v}', s-1, C, T, \phi'_1, \phi'_2, \theta'_1, \theta'_2$)

end if

end function

4.4 Summary

Our method is based on the IoU of the estimated shadow mask and the candidate shadow mask as an objective comparison criterion. We fist define an initial direction light vector $\mathbf{v}$ with associated IoU value g' from the estimated shadow mask in 3D space and points sampled from the surface of the virtual object.

This initial solution is improved in a recursive process. A cap surface $S_{\mathbf{v}}$ over a unit sphere is defined around $\mathbf{v}$ with opening α. This parameter defines how far we can search for a better directional light vector. The sampling resolution, which is the number of points sampled in the axes ϕ and θ, is defined by parameters m and n. The cap is point-wise uniformly sampled and IoU is computed for each sample. An integral image of the sampled cap is built. The cap is partitioned in sector regions comprising several samples. The integral of IoU per unit area is computed for each sector. The sector with highest average IoU is selected. The sector is subdivided and this step is repeated a number of times equal to parameter s. If the newfound maximum IoU value g'' is higher than the current g', set its corresponding vector $\mathbf{v}'$ as the improved directional light vector.

Finally, knowing that the directional light vector found in a discretized space can be further improved, we shorten the opening angle α and use the newly found vector $\mathbf{v}'$ as center of another search space $S''_{\mathbf{v}'}$, with the same properties as the previous one, but a shortened angle α' and centered at $\mathbf{v}'$. This new space is again sampled and integrated from 0 to α' and from 0 to 2π.

This step is repeated a number of times equal to parameter r, the recursion limit, and finally returns vector $\mathbf{v}'$ with highest value g' from the sample $S'_{\mathbf{v}}$. Assuming we cannot improve this result any further in our method, we output the vector $\mathbf{v}'$ as the best directional light estimation.

Our heuristic is effective and easy to implement. There are multiple metaheuristics present in the literature that can be used in our method. However, despite that the search for the global maximum could be greatly improved, the time cost is prohibitive. Our method provides a fair result with a low time complexity. It is important to notice, however, that the cost for computing IoU for each sample dominates the time complexity.

5 EXPERIMENTAL RESULTS

In this chapter, the results obtained with our method are presented. Our goal is to evaluate the proposed method's feasibility and performance when finding the highest IoU value within the search space $S'_{\mathbf{v}}$.

5.1 SCENE SETS

To be able to evaluate the method in multiple environments, we assembled real world scenes, gathered from pictures, and synthetic scenes, assembled virtually. The synthetic scenes are divided into two subsets: synthetic scenes with solid color backgrounds; and synthetic scenes with textured background, as explained in Section **5.1.2**.

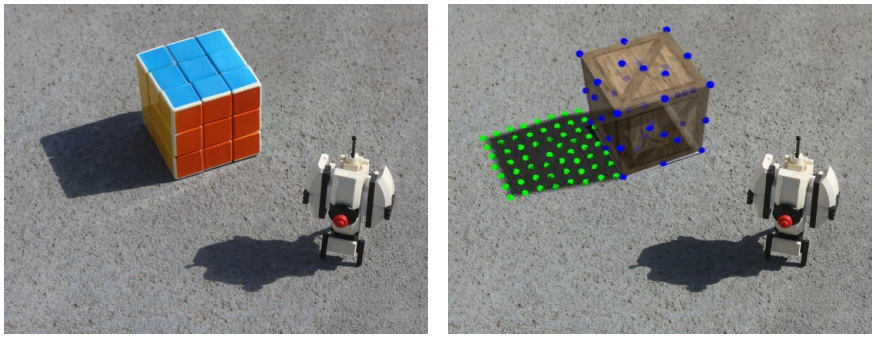
Our method does not compute textures, so the criteria for acquiring a broader range of experiment data was to get scenes with different light source positions. In synthetic scenes, the directional light was repositioned for each scene to generate shadows in different angles.

For real world scenes, the photos were taken around 3p.m. on a sunny day. The camera used to take the photos was a Canon T5i with 18-55mm lenses. For synthetic scenes, we developed a simple ThreeJS application to control the directional light and place meshes of basic shapes on the scene, such as cubes, spheres and cylinders.

5.1.1 Real world scenes

We evaluate the method's performance with real scenes, in which the ground truth (GT) directional light vector is not known. To overcome this problem, we visually and empirically estimated the directional light vector. The Estimated Ground Truth (EGT) is defined by the vector that best matches the shadow of the real objects in the scene. The environment must allow for two pictures to be taken in sequence, one with a fiducial marker present, and one with a real cube of same size as the virtual cube placed exactly on top of the marker. The cube was chosen as the virtual object because it is an easy shape to handle and because of the availability of real cubes, namely a rubrik cube and a larger wooden cube. Everything in the scene, except for the cube and its shadow, must remain the same in both pictures. This is achieved by placing the camera on a tripod and taking the pictures without touching it, to prevent it from rotating or moving slightly. Additionally, the shadow must be cast on a single plane. The EGT is the vector that best replicates the real cube shadow virtually, by using the fiducial marker. In the following experiments, the EGT is compared to our method's output to evaluate the estimation error. This process is depicted in Figure 8.

After acquiring the EGT, the experiments are executed by passing a picture with a fiducial marker to the method. There are 16 pictures in total, labeled from r01 to r16.



(a) (b)

Figure 8: Pictures from the same scene taken with rubrik cube (a) and a virtual cube (b). The blue dots on the cube's surface and the green dots on the shadow are a subset of the ones which are paired to compute the preliminary candidates, mentioned in Section **4.2.1**. To aid the visualization, not all shadow points are shown, as they would be overly dense.

Each picture must also have an associated initial shadow estimation, which is generated by ARShadowGAN (9). The pictures corresponding to the real world dataset can be seen in Figure 9.

5.1.2 Synthetic scenes

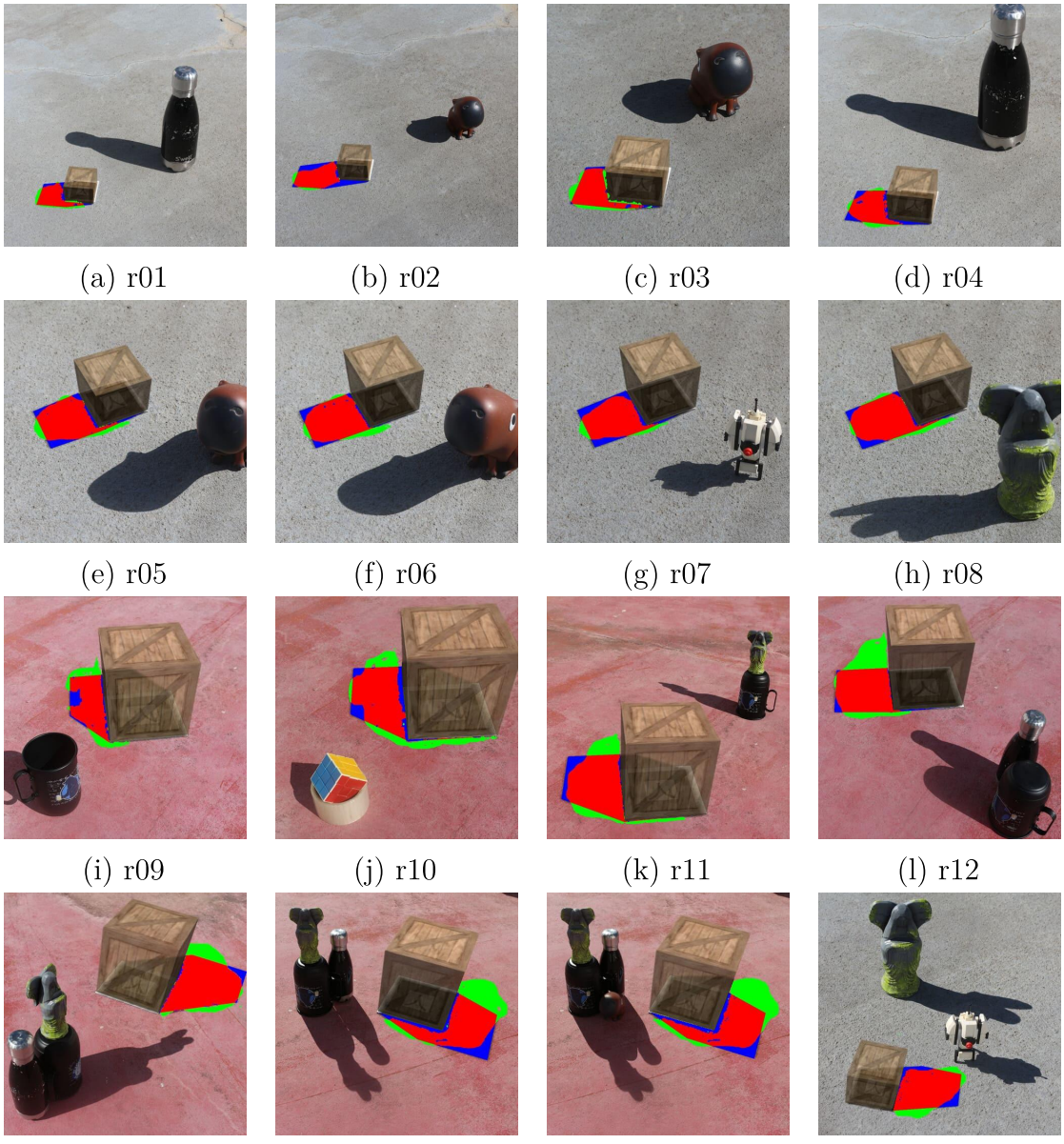
The synthetic scenes are complementary to evaluate the method's performance. It consists of 32 virtual scenes modeled to mimic the real ones. In this set the ground truth directional light vector (GT) is exactly known and the object's shadow cast by the GT can be used as input. Its purpose it to evaluate the method's performance under nearly ideal conditions. The synthetic scenes were divided into two sets: scenes with solid color background; and scenes with textured background. The purpose is to assess of the impact of the background of the scene on the estimated shadow from the ARShadowGAN. The two sets are identical except for their background (Figures 10 and 11).

5.2 QUANTITATIVE RESULTS

In this section, we present and discuss the quantitative aspects of our method. As presented, the virtual object virtually placed in the scene is a cube. Its geometry is interesting since the its shadow tends to have straight lines in its borders.

Regarding the evaluation criteria, two values are relevant when analyzing the results: the angle between the ground truth light vector and the output vector, and the overlap between the shadow cast by the GT and the estimated shadow. This overlap is measured by the IoU formula shown in Equation 4.1.

Our method has 4 parameters that can impact the final result: cap opening angle α, number of IoU resamplings r, number of subdivisions s, and IoU sampling resolution $m \times n$. In order to evaluate which set of parameters provides the best and most consistent



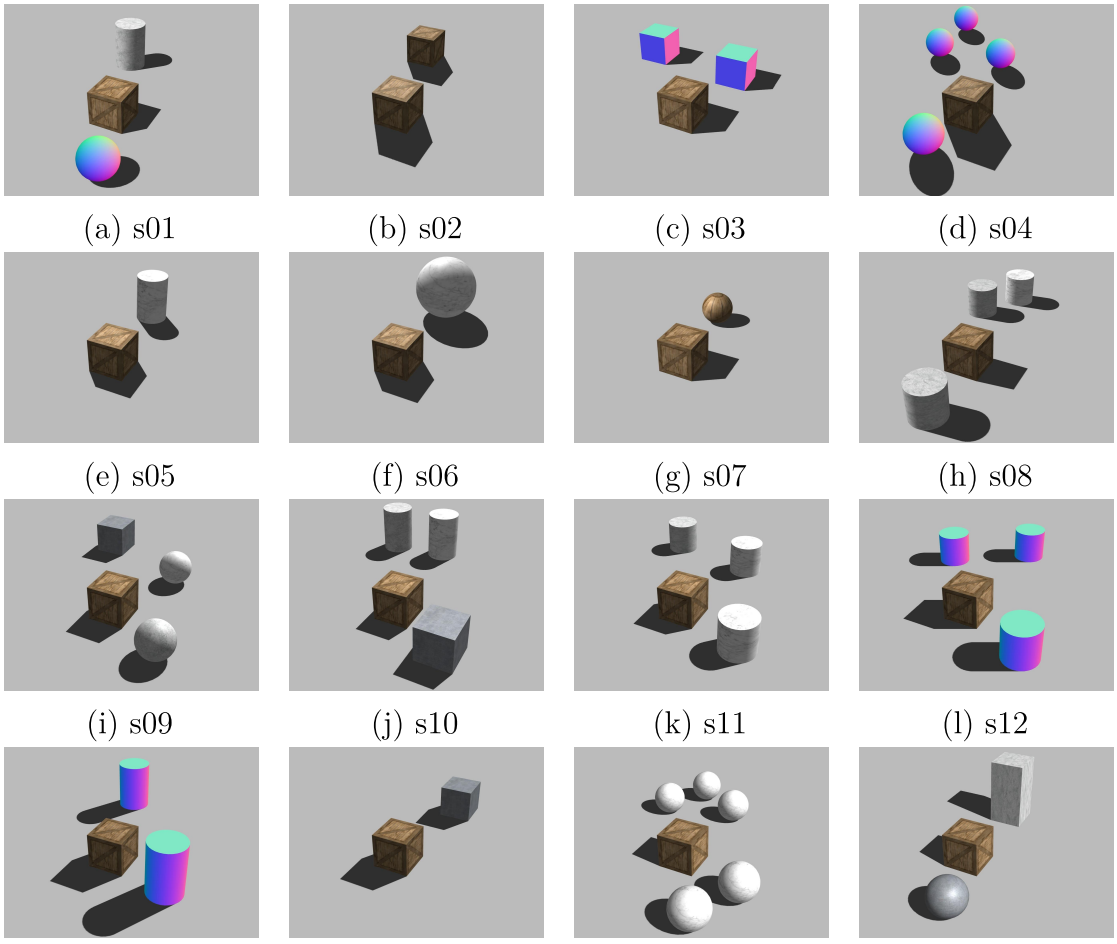
(a) r01 (b) r02 (c) r03 (d) r04

(e) r05 (f) r06 (g) r07 (h) r08

(i) r09 (j) r10 (k) r11 (l) r12

(m) r13 (n) r14 (o) r15 (p) r16

Figure 9: Real world dataset. The green area is the input shadow mask computed by the ARShadowGAN, the blue area is the shadow cast by the initial directional light vector $\mathbf{v}$ and the red area is the intersection between them.



(a) s01 (b) s02 (c) s03 (d) s04

(e) s05 (f) s06 (g) s07 (h) s08

(i) s09 (j) s10 (k) s11 (l) s12

(m) s13 (n) s14 (o) s15 (p) s16

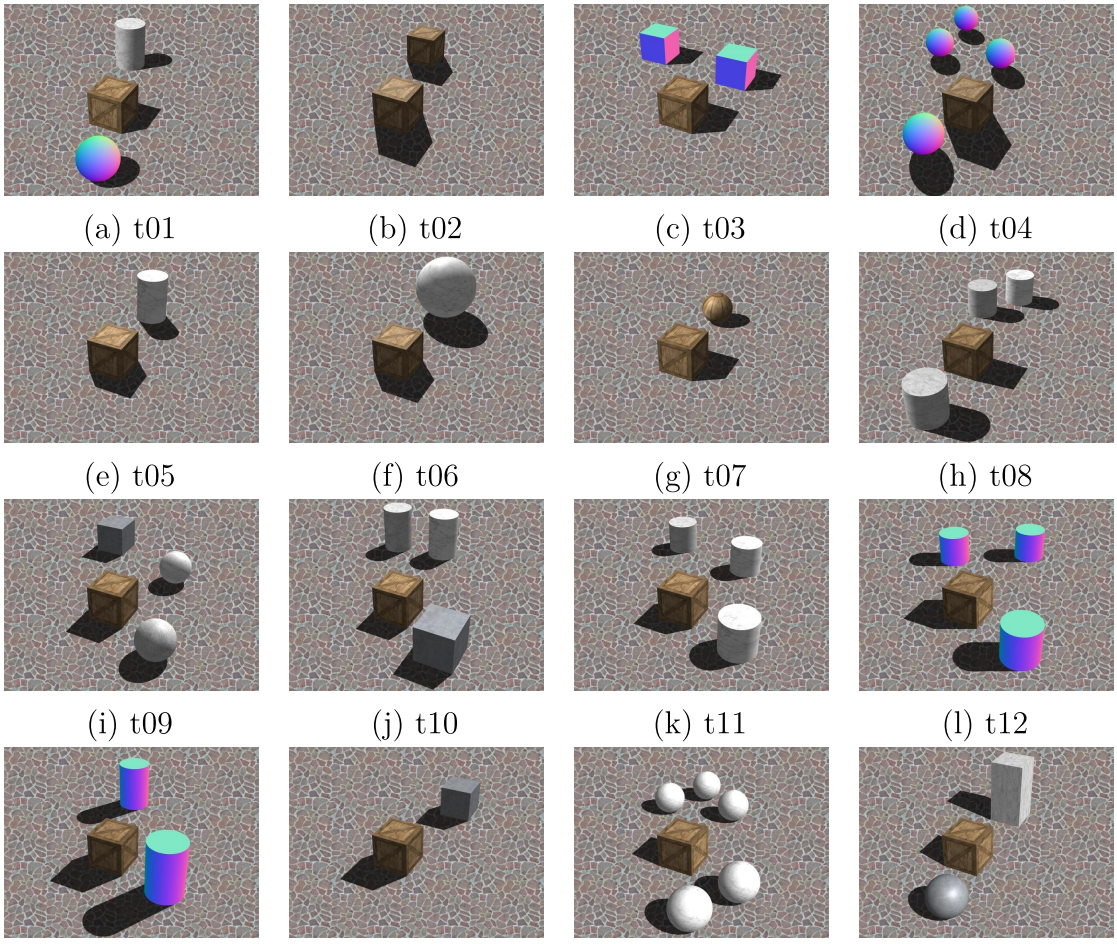
Figure 10: Synthetic dataset with solid color background. The object's shadow is cast by the GT.

results on average, an ablation study was done by combining different values for each parameter, as explained in Section **5.2.1**.

5.2.1 Ablation

The first ablation seeks to understand how various sets of parameters α, r and s impact the output directional light estimation. We have empirically chosen the angles $\alpha \in \{5, 15, 30, 45\}$. Both r and s assume values in the set $\{0, 1, 2, 3\}$. The results are shown in Table 2 for 3 specific real scenes: one considered an easy example (r13), one considered of medium difficulty (r09), and one considered a hard case for our method (r16). The tables for all sixteen scenes are in Appendix A. In all tables, the IoU and the angle difference to the EGT (angle error) is presented for each parameter set. The resolution was empirically fixed as 129×513 for all scenes.

In r13 on the left, the input is considered easier since the shadow generated by ARShadowGAN has is geometrically close to the expected one, pointed to the correct direction. The best result yielded a light direction only 2.28° from the EGT vector. In r09



(a) t01 (b) t02 (c) t03 (d) t04

(e) t05 (f) t06 (g) t07 (h) t08

(i) t09 (j) t10 (k) t11 (l) t12

(m) t13 (n) t14 (o) t15 (p) t16

Figure 11: Synthetic dataset with textured background. The object's shadow is cast by the GT.

in the middle, the input has a concave shape that extends beyond the expected shadow. This complicates the search for the directional light. However, the error decreases as the IoU increases because the EGT shadow is roughly a subset of the input shadow. vector. With $\alpha = 30°$, the cap is large enough to include the best IoU result, with an angle error of 13.18° . In r16 in the right, the input shadow (green) is misaligned with the EGT shadow (blue). It is clearly distributed towards to the top right direction whereas whereas the real objects' shadow points to the bottom right direction. Our method was designed to find a maximum, as all results in Appendix A show. Its performance thus depends entirely on the input shadow estimation. This example shows that the final direction light vector diverges from the EGT as IoU increases. The angle error worsened from 14.63° to 20.85°. Only two scenes ($r02$ and $r16$) had angle error worsened as the IoU increases, from the 16 considered (Appendix A).

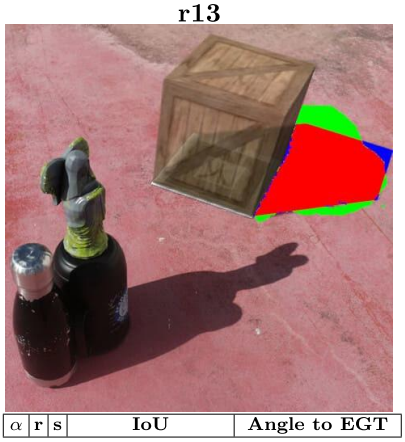
The IoU tends to increase as α, r and s increase. Regarding the α parameter, the best overall value is around 30°. Even though $\alpha = 45°$ does provide a broader sphere cap $S'_{\mathbf{v}}$, since α is too broad, the method could make a large step when looking for the most

dense section of $S'_{\mathbf{v}}$ and miss the maximum value. Even if s is increased to acquire a better subsection of $S'_{\mathbf{v}}$, it consequentially diminishes α' for the next loop, and the new cap $S''_{\mathbf{v}'}$ might be even further from the EGT. $\alpha = 5°$ is the worst value in most cases because the EGT is mostly further away from the initial vector $\mathbf{v}$ than that.

When fixing α and varying r and s, the combination that provided the most consistent results was $r = 3, s = 3$, especially when paired with $\alpha = 30°$. Overall, $\alpha = 45°$ provided surprisingly good results on average, but it was also more prone to fluctuation. This is partly due to resolution, but the difference in the results for each resolution was mostly below the order of 10^{-3}, which indicates that the resolution of the experiments does not need to be higher.

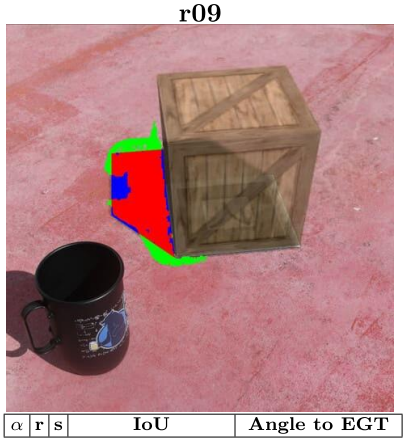
Table 2 – Ablation study for the α, r and s parameters for experiments r09, r13 and r16, respectively. The first row is the initial estimation, and the percentage values represent how much each value improved upon the initial estimation.

r13



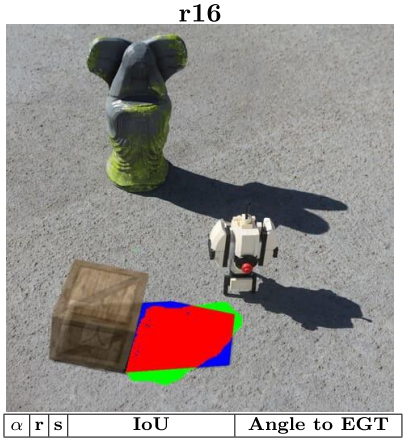
α	r	s	IoU	Angle to EGT
-	0	0	0.519 (0.00%)	12.71° (0.00%)
5	1	0	0.595 (14.75%)	9.79° (22.91%)
5	2	1	0.614 (18.27%)	9.92° (21.90%)
5	2	2	0.635 (22.32%)	8.54° (32.76%)
5	2	3	0.635 (22.47%)	9.19° (27.70%)
5	3	1	0.620 (19.51%)	9.85° (22.50%)
5	3	2	0.647 (24.59%)	8.16° (35.77%)
5	3	3	0.642 (23.80%)	8.94° (29.61%)
15	1	0	0.701 (35.10%)	4.77° (62.49%)
15	2	1	0.704 (35.62%)	4.19° (67.02%)
15	2	2	0.706 (36.12%)	3.32° (73.84%)
15	2	3	0.682 (31.35%)	5.71° (55.03%)
15	3	1	0.706 (36.01%)	3.94° (69.02%)
15	3	2	**0.707 (36.27%)**	2.48° (80.52%)
15	3	3	0.691 (33.11%)	5.13° (59.66%)
30	1	0	0.705 (35.93%)	3.30° (74.05%)
30	2	1	0.704 (35.66%)	3.67° (71.11%)
30	2	2	0.706 (36.05%)	2.87° (77.44%)
30	2	3	0.697 (34.28%)	3.02° (76.26%)
30	3	1	0.706 (36.13%)	2.70° (78.72%)
30	3	2	0.707 (36.17%)	2.95° (76.76%)
30	3	3	0.703 (35.48%)	2.38° (81.24%)
45	1	0	0.704 (35.72%)	3.25° (74.46%)
45	2	1	0.703 (35.50%)	3.48° (72.59%)
45	2	2	0.706 (36.07%)	2.91° (77.08%)
45	2	3	0.706 (36.13%)	2.53° (80.12%)
45	3	1	0.707 (36.16%)	2.55° (79.93%)
45	3	2	0.703 (35.44%)	**2.28° (82.05%)**
45	3	3	0.707 (36.24%)	2.51° (80.25%)

r09



α	r	s	IoU	Angle to EGT
-	0	0	0.301 (0.00%)	23.24° (0.00%)
5	1	0	0.328 (8.88%)	22.64° (2.60%)
5	2	1	0.340 (12.65%)	22.40° (3.64%)
5	2	2	0.344 (14.29%)	24.45° (-5.19%)
5	2	3	0.349 (15.68%)	23.40° (-0.70%)
5	3	1	0.345 (14.40%)	22.56° (2.94%)
5	3	2	0.350 (16.24%)	24.56° (-5.68%)
5	3	3	0.354 (17.56%)	23.24° (0.00%)
15	1	0	0.398 (32.03%)	20.79° (10.56%)
15	2	1	0.428 (41.96%)	21.17° (8.93%)
15	2	2	0.471 (56.10%)	18.29° (21.31%)
15	2	3	0.460 (52.57%)	19.83° (14.68%)
15	3	1	0.431 (42.87%)	22.80° (1.90%)
15	3	2	0.477 (58.24%)	19.29° (16.98%)
15	3	3	0.472 (56.48%)	19.67° (15.37%)
30	1	0	0.508 (68.70%)	15.10° (35.05%)
30	2	1	0.490 (62.62%)	18.58° (20.05%)
30	2	2	0.567 (88.26%)	15.48° (33.38%)
30	2	3	0.603 (99.92%)	14.28° (38.55%)
30	3	1	0.544 (80.44%)	16.11° (30.70%)
30	3	2	0.605 (100.88%)	13.32° (42.69%)
30	3	3	**0.618 (105.14%)**	13.18° (43.29%)
45	1	0	0.606 (101.02%)	**12.53° (46.10%)**
45	2	1	0.617 (104.61%)	13.32° (42.71%)
45	2	2	0.532 (76.47%)	17.71° (23.79%)
45	2	3	0.462 (53.16%)	20.52° (11.70%)
45	3	1	0.617 (104.59%)	13.36° (42.50%)
45	3	2	0.618 (104.88%)	13.17° (43.33%)
45	3	3	0.509 (68.78%)	18.15° (21.90%)

r16



α	r	s	IoU	Angle to EGT
-	0	0	0.623 (0.00%)	**14.63° (0.00%)**
5	1	0	0.682 (9.35%)	16.00° (-9.37%)
5	2	1	0.690 (10.76%)	16.26° (-11.17%)
5	2	2	0.698 (12.04%)	16.27° (-11.22%)
5	2	3	0.698 (11.94%)	16.53° (-13.05%)
5	3	1	0.693 (11.16%)	16.22° (-10.91%)
5	3	2	0.699 (12.22%)	16.26° (-11.15%)
5	3	3	0.700 (12.29%)	16.59° (-13.46%)
15	1	0	0.721 (15.75%)	20.91° (-42.99%)
15	2	1	0.723 (16.07%)	20.18° (-37.98%)
15	2	2	0.722 (15.85%)	20.83° (-42.41%)
15	2	3	0.723 (16.02%)	20.18° (-37.98%)
15	3	1	0.723 (15.93%)	20.92° (-43.01%)
15	3	2	0.724 (16.14%)	20.14° (-37.68%)
15	3	3	0.724 (16.13%)	20.19° (-38.03%)
30	1	0	0.720 (15.50%)	20.03° (-36.97%)
30	2	1	0.722 (15.77%)	20.10° (-37.44%)
30	2	2	0.723 (16.07%)	20.18° (-37.97%)
30	2	3	0.723 (15.93%)	19.96° (-36.44%)
30	3	1	0.723 (16.02%)	20.16° (-37.86%)
30	3	2	0.723 (16.08%)	20.67° (-41.33%)
30	3	3	0.724 (16.11%)	20.18° (-37.98%)
45	1	0	0.721 (15.64%)	20.39° (-39.41%)
45	2	1	0.721 (15.61%)	21.00° (-43.62%)
45	2	2	0.723 (15.97%)	20.18° (-37.99%)
45	2	3	0.722 (15.91%)	21.82° (-49.20%)
45	3	1	0.722 (15.82%)	20.57° (-40.68%)
45	3	2	0.723 (15.99%)	20.24° (-38.40%)
45	3	3	**0.724 (16.14%)**	20.85° (-42.56%)

To study how the resolution impacts the method's performance, the 16 real world

Table 3 – IoU absolute value and percentual increase for $\alpha = 5°, 15°, 30°, 45°$.

(a) Resolution 97×449.

Scene	5°	15°	30°	45°
r01	0.716 (9.59%)	**0.716 (9.70%)**	0.715 (9.49%)	0.714 (9.25%)
r02	0.606 (16.70%)	**0.610 (17.58%)**	0.610 (17.46%)	0.608 (17.10%)
r03	0.620 (25.72%)	0.736 (49.12%)	**0.737 (49.43%)**	0.732 (48.33%)
r04	**0.604 (3.03%)**	0.603 (2.92%)	0.602 (2.82%)	0.598 (2.10%)
r05	0.617 (19.44%)	0.687 (32.84%)	0.679 (31.45%)	**0.688 (33.03%)**
r06	**0.703 (5.42%)**	0.703 (5.35%)	0.702 (5.27%)	0.702 (5.21%)
r07	0.626 (10.58%)	0.639 (12.99%)	**0.639 (13.01%)**	0.638 (12.70%)
r08	0.612 (19.73%)	0.699 (36.75%)	**0.703 (37.52%)**	0.701 (37.11%)
r09	0.344 (14.21%)	0.422 (39.94%)	0.512 (69.91%)	**0.564 (87.18%)**
r10	0.404 (2.01%)	0.452 (14.11%)	0.511 (28.84%)	**0.542 (36.89%)**
r11	0.541 (15.01%)	0.659 (40.07%)	**0.682 (44.97%)**	0.679 (44.45%)
r12	0.577 (0.47%)	0.586 (2.06%)	**0.590 (2.81%)**	0.569 (-0.81%)
r13	0.627 (20.75%)	0.699 (34.73%)	0.703 (35.50%)	**0.705 (35.93%)**
r14	0.525 (15.27%)	0.533 (17.00%)	**0.533 (17.00%)**	0.529 (16.14%)
r15	0.506 (14.93%)	0.508 (15.56%)	0.512 (16.28%)	**0.520 (18.19%)**
r16	0.694 (11.33%)	**0.722 (15.94%)**	0.722 (15.87%)	0.722 (15.87%)
Avg.	0.583 (12.76%)	0.623 (21.67%)	0.635 (24.85%)	**0.638 (26.17%)**

(b) Resolution 129×513.

Scene	5°	15°	30°	45°
r01	0.717 (9.75%)	**0.717 (9.82%)**	0.716 (9.60%)	0.713 (9.24%)
r02	0.606 (16.72%)	**0.611 (17.70%)**	0.610 (17.59%)	0.608 (17.21%)
r03	0.620 (25.64%)	0.736 (49.24%)	**0.738 (49.54%)**	0.732 (48.44%)
r04	**0.604 (3.11%)**	0.603 (3.01%)	0.603 (2.90%)	0.599 (2.21%)
r05	0.618 (19.47%)	0.687 (32.89%)	0.680 (31.60%)	**0.688 (33.16%)**
r06	0.703 (5.45%)	**0.704 (5.48%)**	0.703 (5.37%)	0.703 (5.35%)
r07	0.626 (10.59%)	0.639 (12.89%)	**0.640 (13.09%)**	0.640 (13.07%)
r08	0.613 (19.75%)	0.699 (36.62%)	**0.704 (37.67%)**	0.702 (37.24%)
r09	0.344 (14.24%)	0.448 (48.61%)	0.562 (86.57%)	**0.566 (87.64%)**
r10	0.402 (1.56%)	0.453 (14.29%)	0.511 (28.97%)	**0.542 (36.87%)**
r11	0.540 (14.91%)	0.655 (39.25%)	**0.682 (45.07%)**	0.680 (44.69%)
r12	0.577 (0.51%)	0.587 (2.22%)	**0.593 (3.26%)**	0.571 (-0.56%)
r13	0.627 (20.81%)	0.699 (34.80%)	0.704 (35.67%)	**0.705 (35.89%)**
r14	0.525 (15.30%)	0.533 (17.05%)	**0.534 (17.16%)**	0.530 (16.42%)
r15	0.506 (15.02%)	**0.513 (16.64%)**	0.512 (16.35%)	0.510 (15.85%)
r16	0.694 (11.39%)	**0.723 (15.98%)**	0.723 (15.93%)	0.722 (15.87%)
Avg.	0.583 (12.76%)	0.625 (22.28%)	**0.638 (26.02%)**	0.638 (26.16%)

(c) Resolution 161×577.

Scene	5°	15°	30°	45°
r01	0.717 (9.78%)	**0.718 (9.86%)**	0.716 (9.64%)	0.713 (9.20%)
r02	0.606 (16.73%)	**0.611 (17.70%)**	0.611 (17.63%)	0.610 (17.49%)
r03	0.620 (25.66%)	**0.736 (49.30%)**	0.709 (43.77%)	0.733 (48.68%)
r04	**0.604 (3.11%)**	0.604 (3.05%)	0.603 (2.87%)	0.598 (2.17%)
r05	0.617 (19.46%)	0.687 (32.87%)	0.680 (31.58%)	**0.688 (33.11%)**
r06	0.703 (5.45%)	**0.704 (5.48%)**	0.703 (5.43%)	0.703 (5.40%)
r07	0.626 (10.62%)	0.639 (12.92%)	**0.640 (13.18%)**	0.640 (13.05%)
r08	0.613 (19.77%)	0.700 (36.80%)	**0.704 (37.64%)**	0.702 (37.34%)
r09	0.344 (14.28%)	0.448 (48.64%)	0.562 (86.53%)	**0.566 (87.70%)**
r10	0.403 (1.59%)	0.453 (14.28%)	0.529 (33.49%)	**0.542 (36.87%)**
r11	0.540 (14.93%)	0.656 (39.59%)	**0.682 (45.15%)**	0.671 (42.65%)
r12	0.577 (0.52%)	0.587 (2.20%)	**0.593 (3.28%)**	0.570 (-0.70%)
r13	0.627 (20.81%)	0.700 (34.82%)	0.704 (35.76%)	**0.706 (36.00%)**
r14	0.525 (15.34%)	0.533 (17.07%)	**0.533 (17.14%)**	0.530 (16.48%)
r15	0.506 (15.05%)	**0.513 (16.66%)**	0.512 (16.38%)	0.510 (15.85%)
r16	0.694 (11.41%)	0.722 (15.96%)	**0.723 (16.00%)**	0.722 (15.86%)
Avg.	0.583 (12.78%)	0.626 (22.33%)	**0.638 (25.97%)**	0.638 (26.07%)

scenes were passed to the method with the same parameters and 3 different resolutions: 97×449, 129×513 and 161×577, for a total of 48 experiments. The default value is 129×513, in which m and n are a power of 2 plus 1, namely $2^7 + 1$ and $2^9 + 1$. This is so the subdivisions are favorable for the method and always produce subsections of odd resolution, so that the center point of each section is always at the center and is not an approximation. The higher and the lower resolutions, respectively 97×449 and 161×577, have $m \pm 32$ and $n \pm 64$ compared to the default resolution. The default value 129×513 was achieved empirically, by passing different resolutions to the method that correspond to $2^x + 1 \times 2^y + 1$, where $y > x$. It was chosen because it provides consistent results for both lower and higher α values and because it's time cost is not prohibitive. To generate a single table of Appendix A, the method runs a total of 64 loops. Table 6 can be consulted to visualize the time cost for different resolutions.

The IoU difference for each resolution in Table 3 is mostly marginal, but it is indicative that the method performs as intended. The method always finds the same region regardless of the resolution change, the best IoU value only increasing or decreasing slightly due to sampling. It also indicates that the resolution 129×513 is best, since it has a somewhat significant average increase in IoU of 1.17% compared to resolution 97×449, for $\alpha = 30°$, which we stated that is the most consistent value from the ablation study. Therefore, it was decided to fix the resolution at the initial value of 129×513, and fix the other parameters at $\alpha = 30°$, $r = 3$ and $s = 3$ for the remainder of the experiments.

To be able to make an objective analysis of our method's accuracy regardless of the input shadow quality, the only available criterion is the IoU value. A significant increase in the IoU value indicates that the method can perform well independently of the precision of the input. However, the angle between the output vector $\mathbf{v}'$ and the EGT is also indicative of our method's accuracy. We compare the highest IoU value found on each scene to the input of that scene. The angle is compared similarly, but the minimum angular difference to the EGT is selected instead, since a lower angular difference indicates a better result.

5.2.2 Results for synthetic scenes

The experiments on synthetic scenes were executed in two scenarios: with the GT cast shadow as input; and with the estimated shadow as input. This is to evaluate the behavior of the method under ideal conditions and to assess the impact of ARShadowGAN on the results compared to the ideal scenario.

5.2.2.1 Results for ground truth shadow input

By using the mask of the shadow cast by each scene's GT as input, it is possible to assess the method's performance under ideal conditions. As Section **5.2.1** shows, the best average results are recorded for $\alpha = 30°, r = 3, s = 3$ and resolution 129×513. Therefore,

those are the parameters used in the experiments of this section, shown in Table 4.

The experiments were executed on all synthetic scenes, both with solid color and textured background. However, since the cast shadow is identical for the corresponding scenes of each set, as can be verified comparing Figures 10 and 11, the results were identical for both sets.

Table 4 – Results for experiments on the synthetic dataset with GT shadow input and parameters $\alpha = 30°, r = 3, s = 3$ and resolution 129×513.

Scene	**IoU**	**Angle to GT**
s01, t01	0.86061	0.278°
s02, t02	0.94523	0.325°
s03, t03	0.88010	0.203°
s04, t04	0.95534	0.086°
s05, t05	0.91028	0.078°
s06, t06	0.94956	0.415°
s07, t07	0.88586	0.067°
s08, t08	0.87851	0.255°
s09, t09	0.88232	0.666°
s10, t10	0.88258	0.035°
s11, t11	0.87726	0.643°
s12, t12	0.91847	1.404°
s13, t13	0.91801	0.104°
s14, t14	0.91391	0.255°
s15, t15	0.78571	0.783°
s16, t16	0.76767	0.918°

Table 4 indicates that, if there is a shadow estimator capable of estimating a shadow identical to the one cast by the GT, our method can output the light vector with an angular error below 1° in most cases. Furthermore, our method is still suitable even if the shadow can be estimated perfectly, because it retrieves the light vector, not the 2D shadow. The light vector is extremely useful for AR contexts, whereas the estimated shadow of a single frame is not.

5.2.2.2 Results for estimated shadow input

By using the shadows computed by ARShadowGAN as the input for the synthetic sets, it is possible to evaluate ARShadowGAN's performance for each set, and what characteristics of each scene impact the estimated shadow. It is also possible to compare the results for estimated shadow input to the results for GT shadow input, to evaluate how ARShadowGAN impacts the final output.

The results from Table 5a and 5b do not greatly vary from each other. This indicates that the textured background does not have a big impact on the ARShadowGAN shadow. An input of poor quality tends to result in an output far from the GT.

When compared to the real world set, the synthetic sets have a lower Angle to GT but lower IoU as well. This is because the shadows of synthetic scenes assume concave

Table 5 – Results for experiments on the synthetic dataset with estimated shadow input and parameters $\alpha = 30°, r = 3, s = 3$ and resolution 129×513.

Scene	IoU	Angle to GT
s01	0.69084	15.551°
s02	0.36949	20.626°
s03	0.63324	10.365°
s04	0.51553	17.028°
s05	0.50114	8.163°
s06	0.59682	13.087°
s07	0.57809	21.708°
s08	0.65337	0.417°
s09	0.59545	11.142°
s10	0.50749	9.664°
s11	0.54816	9.022°
s12	0.57036	2.994°
s13	0.63626	13.662°
s14	0.57873	1.751°
s15	0.53267	14.872°
s16	0.48286	16.750°

(a) Solid color background set.

Scene	IoU	Angle to GT
t01	0.67672	13.821°
t02	0.47989	12.658°
t03	0.62685	7.497°
t04	0.57663	14.023°
t05	0.58123	9.887°
t06	0.61991	11.121°
t07	0.59201	26.348°
t08	0.64447	0.097°
t09	0.58027	23.086°
t10	0.55320	13.509°
t11	0.57655	13.190°
t12	0.60568	0.775°
t13	0.62613	18.357°
t14	0.60080	1.594°
t15	0.57824	17.919°
t16	0.54579	11.294°

(b) Textured background set.

and more complex shapes more frequently than the shadows of the real world set, as seen on Figures 12 and 13. These shapes complicate the method's search, since they don't fit the cube's shadow shape.

5.2.3 Execution times

The runtime was also evaluated in the experiments. The experiments were run on Google Chrome on a computer with a GeForce GTX 1050Ti Nvidia GPU, i7-11700K Intel processor and 48GB RAM. The average runtime for each loop of the method given a certain $m \times n$ resolution is recorded on Table 6.

Table 6 – Average execution time for each loop of the method's implementation.

$m \times n$ resolution	Rendered frames	Avg. exec. time
10×36	360	00:02
97×449	43553	03:48
129×513	66177	05:37
161×577	92897	08:07
257×1025	263425	24:07

When dividing the average execution time, in seconds, by the number of rendered frames for each resolution, we get the time taken to render each frame, which is approximately 5ms. This is true for every resolution in Table 6, which indicates that the time cost increases linearly with the resolution. This is expected, since the rendering step dominates the algorithm's complexity.

The execution time was measured by running the method on real world scenes with fixed parameters $\alpha = 30°$, $r = 1$, $s = 0$ and resolution according to the left column of

Table 6. The only parameters that impact the execution time, however, are the resolution $m \times n$ and the number of resamples r. A higher resolution means more frames to render, and more resamplings mean more loops to the method.

5.3 QUALITATIVE RESULTS

The criterion for evaluating the performance of our method is the angular difference between the estimated ground truth (EGT) and the output light vector $\mathbf{v}'$. However, our method is guided by the IoU value of the cast shadows within our search space, as mentioned in Section 4.3. That is because a higher IoU value corresponds to a lower angular difference when the shadow perfectly corresponds to the one cast by the GT, if available.

To qualitatively evaluate our method's graphic output, we compare the shadow cast by the output vector $\mathbf{v}'$ to the input. The visual output is a reasonable assessment of how our method can improve a rough shadow estimation.

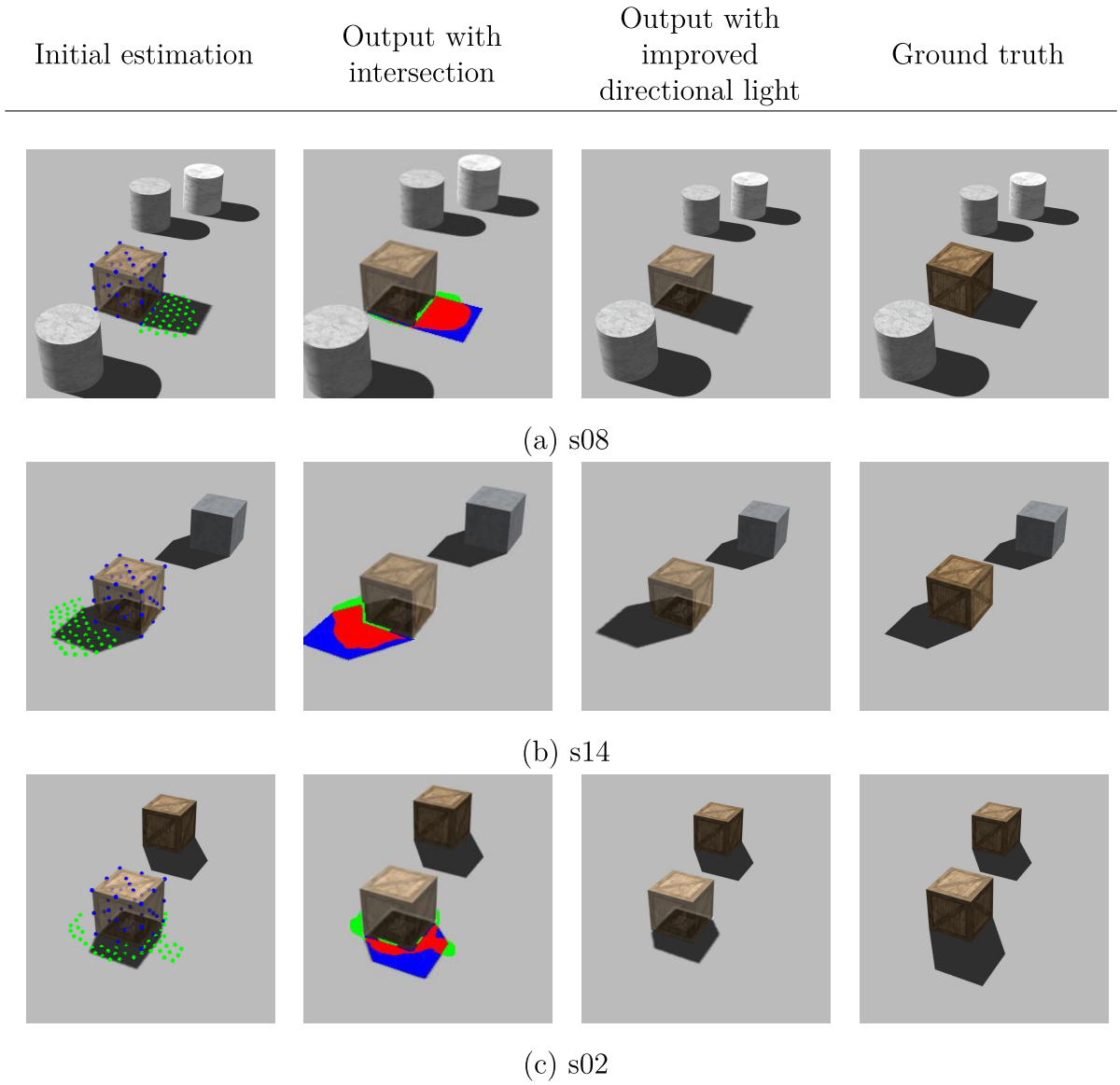
The results are divided in sections for each dataset: synthetic with solid color background, synthetic with textured background, and real world pictures. Most of our results were visually similar to the GT, for synthetic scenes, or EGT, for real world scenes. These are considered success cases, and are discussed in Section **5.3.1**, Section **5.3.2** and Section **5.3.3**. The cases in which the output shadow is visibly misaligned with the GT or EGT are discussed in Section 5.4.

5.3.1 Synthetic pictures with solid color background

The objective of the qualitative experiments is to evaluate how our method improves upon a rough shadow shape. Therefore, the estimated input shadow, and not the GT shadow, is compared to the output shadow. In Figure 12, the results are divided in four columns: the initial estimation, the intersection between input and output shadows, the clean output scene, and the GT shadow. In the leftmost column, only a subset of the shadow points are shown. That is because otherwise they would be too dense to be seen as individual points, and would look like a single blob instead.

In all experiments, the wooden cube is the known virtual object, and the rest of the scene is treated as the background picture.

In Figure 12a, the output is closely aligned with the GT. The input shadow is of convex shape and easy complexity. In Figure 12b, the input is of medium complexity and the output doesn't fit the estimated shadow as well as in Figure 12a. However, it is still reasonably aligned with the GT. The input shadow in Figure 12c is the most complex of Figure 12. It is concave and the output shadow fits it poorly. Nonetheless, the output is visually acceptable and seems natural if not compared to the GT.



(a) s08

(b) s14

(c) s02

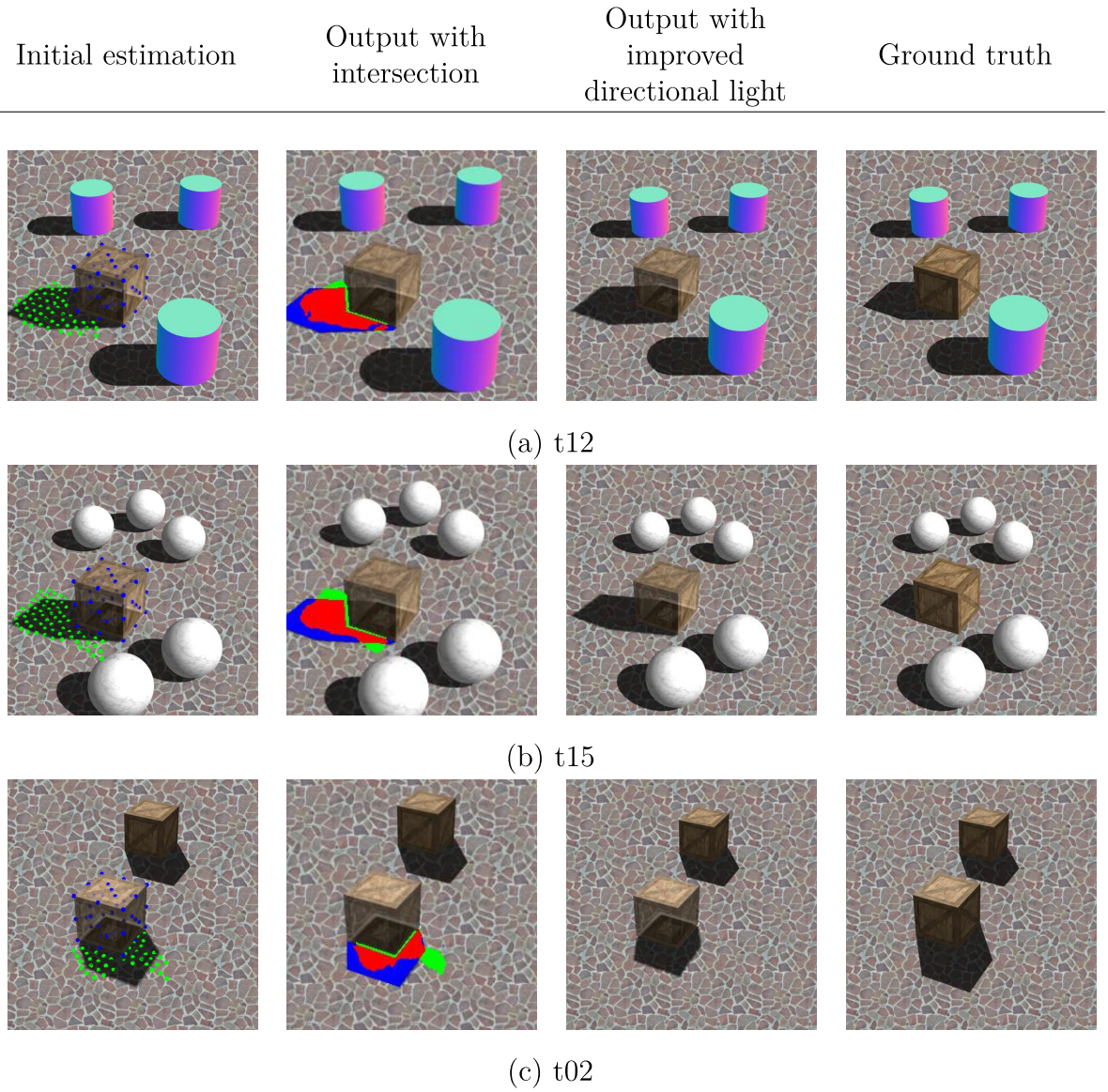
Figure 12: Subset of the synthetic scenes with solid color background. (*a*) is a scene with an easy shadow input, (*b*) is a scene with an input of medium complexity and (*c*) is a scene with a complex input. The complexity is defined by the shadow shape. The leftmost column represents the object points (blue) and estimated shadow points (green). The next column from left to right is the intersection between the estimated shadow (green) and the final output shadow (blue). The last two columns are respectively the final output shadow and the ground truth.

5.3.2 Synthetic pictures with textured background

The results in Figure 13 are assembled exactly as in Figure 12. The first row shows the case with input of easiest complexity, and the last shows the case with input of hardest complexity. The textured background was placed to evaluate if it makes a difference when using a shadow estimator, compared to the plain background.

The input and output shadows in Figure 13a are closely matched, and the output image are visually fine. Figure 13b depicts an interesting case, in which the output seems

to fit the scene, but when compared to the GT, its misalignment becomes clear. Their shadows appear to be cast to the left, but in truth they are cast diagonally up in the picture. This example shows that sometimes high angles or lengths errors are still enough to perceive the scene as acceptable. Due to the camera angle, the shadows seem to be to the left of the objects. Finally, in Figure 13c, the input is more complex and of concave shape, which makes the search for a local maximum harder. The output is misaligned with the GT, but retains acceptable visual quality.



(a) t12

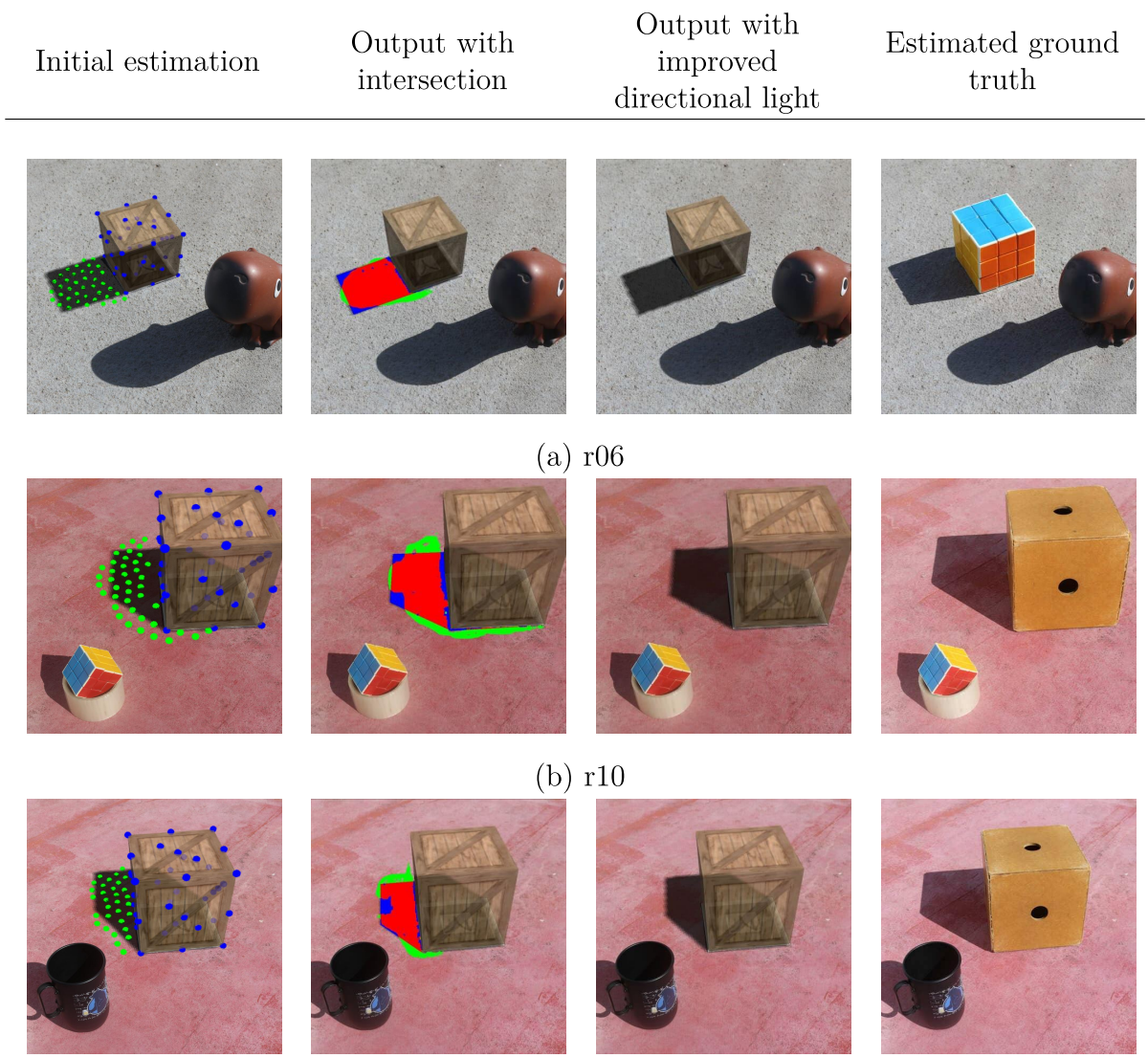
(b) t15

(c) t02

Figure 13: Subset of the synthetic scenes with textured background. (a) is a scene with an easy shadow input, (b) is a scene with an input of medium complexity and (c) is a scene with a complex input. The complexity is defined by the shadow shape. The leftmost column represents the object points (blue) and estimated shadow points (green). The next column from left to right is the intersection between the estimated shadow (green) and the output shadow (blue). The last two columns are respectively the final output shadow and the ground truth.

5.3.3 Real world pictures

Compared to the synthetic experiments, the real world ones have worse results on average, as seen on Section 5.2. However, the visual output is not perceived as misaligned to the GT or EGT as the quantitative results indicate. The estimated shadows for the real world scenes have more consistent shapes on average. Convex shapes are usually more suitable for the method, but they can still be ambiguous and result in a misaligned output. This is discussed in Section 5.4.



(a) r06

(b) r10

(c) r09

Figure 14: Subset of the real world scenes. (a) is a scene with an easy shadow input, (b) is a scene with an input of medium complexity and (c) is a scene with a complex input. The complexity is defined by the shadow shape. The leftmost column represents the object points (blue) and estimated shadow points (green). The next column from left to right is the intersection between the estimated shadow (green) and the output shadow (blue). The last two columns are respectively the final output shadow and the ground truth.

The input shadow in Figure 14a is almost aligned with the EGT, and the output

fits it very closely. It has good visual quality. Figures 14b and 14c have shadows of somewhat similar complexity. Figure 14c has a slightly worse result because it is concave. In both cases, the outputs have good visual quality, albeit the shadows are slightly shorter than the EGT.

The results show that our method is capable of enhancing the visual quality of the shadow. When examining the shape of the input and output shadows, our method significantly improves the quality of the ARShadowGAN estimation. The correct shape of the output shadow is due to the estimated light vector. In Figure 14, it can be seen that the shape of the shadows is enhanced, but the output maintains the direction and length of the input.

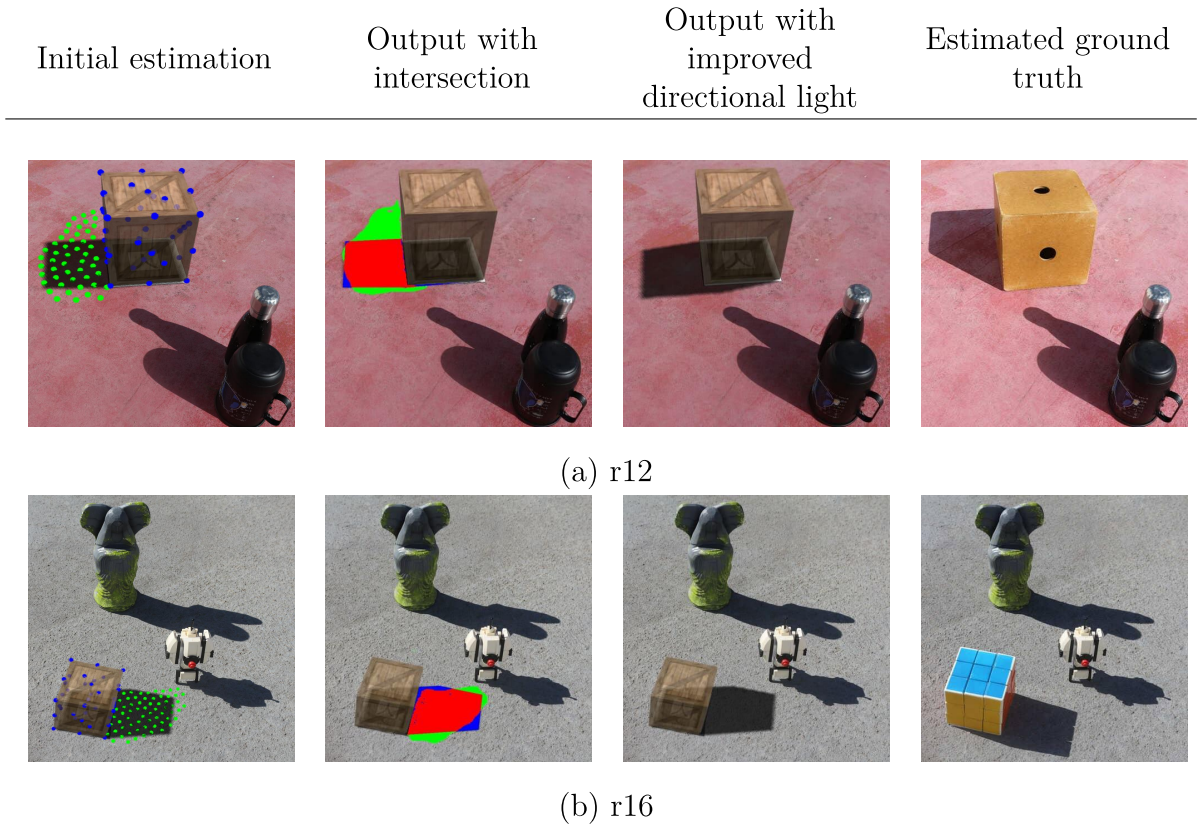
5.4 LIMITATIONS

As stated in Section **5.3.3**, the shape of the shadow is consistent with the virtual object because the shadow is rendered, not estimated. This means the shape is always consistent with the virtual object. The main problem arises when the length and direction of the input shadow are very distinct from the ground truth.

The length and direction of the cast shadow were the only features that our method failed to output correctly in some cases. The results are then perceived as incoherent when visually compared to the ground truth. Those cases are shown in Figure 15 and Figure 16. Our method finds the directional light vector that casts the shadow closest to the input. This can be confirmed by comparing the input and output columns of Figure 15 and 16, and even of Figure 14. Essentially, the output shadow mimics the input shadow. When the input provides a shadow with incorrect lenght or direction, the output will perpetuate or even worsen the error.

Figure 16 depicts two scenes in which the output shadow has a visibly shorter length than expected, especially when compared side by side with the EGT. However, the error in length is not as drastic as the error in direction. In Figure 15, the visual mismatch of the shadows is easily perceived. This is more clear in Figure 15b, in which the shadow is of correct length but has misaligned direction. Figure 15a shows an example where the input shadow has good geometry and length, but the slightly wrong direction hinders the method to find a visually coherent shadow.

In conclusion, our method depends on the estimator. It cannot detect the imprecision of the input, as the GT is unknown to it. An input shadow that matches the length and direction of the GT will result in an output of good visual quality. The input shadow shape is irrelevant, since our method always delivers shadows of correct shape.



(a) r12

(b) r16

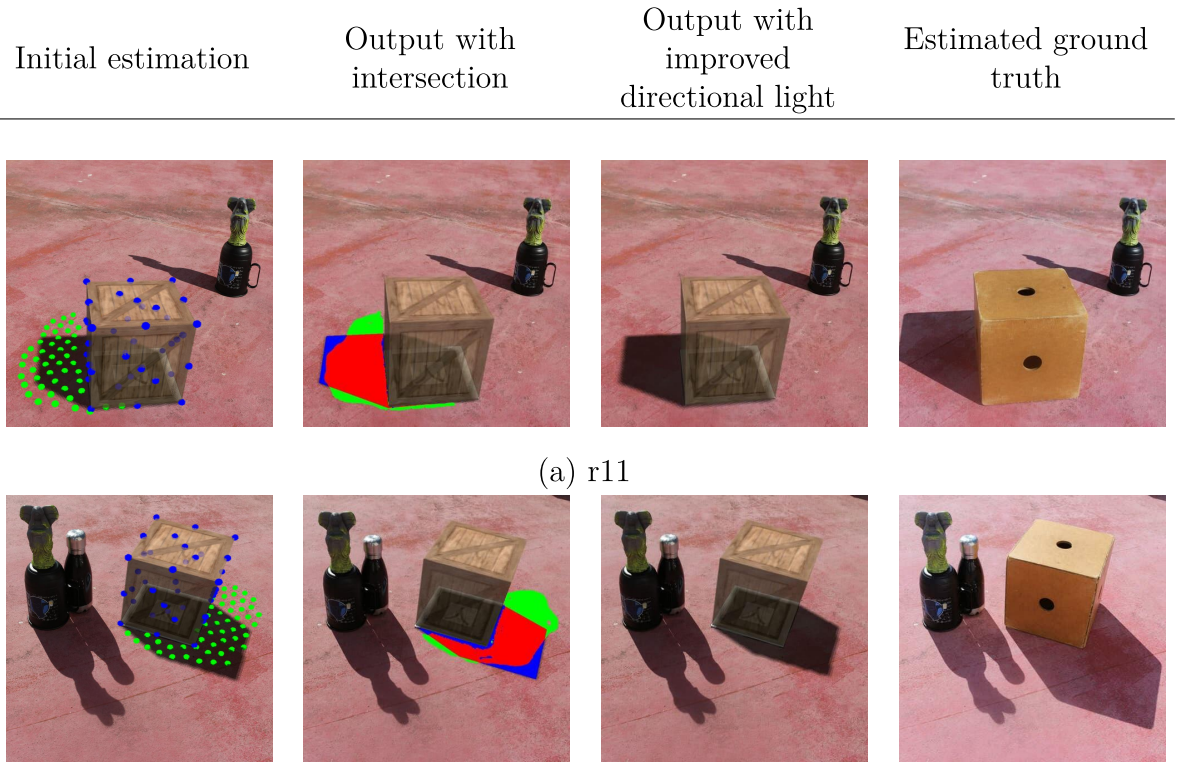
Figure 15: Results with incorrect direction. The leftmost column represents the object points (blue) and estimated shadow points (green). The next column from left to right is the intersection between the estimated shadow (green) and the output shadow (blue). The last two columns are respectively the final output shadow and the ground truth.

5.5 SUMMARY OF THE EXPERIMENTAL RESULTS

Section 5.2 and Section 5.3 indicate that our method performs well in finding the directional light of a scene, given the correct input. When giving to the method an input shadow that doesn't match the GT shadow length nor its direction, the output will also fail to match those characteristics, since the method's only available criteria for comparison is the input.

In Section 5.2, when analyzing the numeric results, it is shown that the method always enhances the initial shadow estimation by finding a vector with higher IoU value associated to it in the search space $S'_{\mathbf{v}}$. This is true for both synthetic and real world scenes. The angle to the GT or EGT, however, does not decrease in all cases. It was observed that, when the input is of good visual quality, the angle to the GT or EGT tend to decrease. When the input is of bad quality, the opposite happens: the output deviates further from the GT, as the method tries to match it with the input.

However, when comparing the output to the GT, it is not possible to precisely know if the GT exists within the search space $S'_{\mathbf{v}}$, which is sampled and not continuous. Therefore, this is compared qualitatively in Section 5.3. The results indicate that when a



(a) r11

(b) r14

Figure 16: Results with mismatched length. The leftmost column represents the object points (blue) and estimated shadow points (green). The next column from left to right is the intersection between the estimated shadow (green) and the output shadow (blue). The last two columns are respectively the final output shadow and the ground truth.

reasonably good quality input is provided to the method, the output is also visually good. When given a bad input, however, the method is not able to fix it, and the output is also of bad quality. The length and, most importantly, the direction of the input shadow must be visually close to the GT to make the output believable to the human eye.

To summarize, we understand that our method provides good quantitative results when looking for the best vector within its search space $S'_{\mathbf{v}}$. However, since the search space $S'_{\mathbf{v}}$ is generated according to the input, that is, an estimated shadow mask for the virtual object, the final output still depends on the input quality. The cast shadow's shape is consistently improved in the final output, but the length and direction of the shadow are still highly dependent on the input.

6 CONCLUSION

In this work, we presented a method to retrieve the main directional light 3D vector of a scene given a 2D shadow mask input. We propose an heuristic to find the best available light vector in the search space, namely a local maximum of IoU. Our hypothesis is that the heuristic can enhance the directional light vector by searching the vicinity for vectors associated with cast shadows that fit best the input.

The method needs an image with a fiducial marker present and a 2D shadow mask as input. Assuming the shadow is cast on the marker's plane, it acquires the 3D contour of the shadow. The shadow points are combined with the points of a virtual object that is assumed to cast that shadow, generating a set of candidates for the initial directional light estimation. From this set, the vector which casts the shadow closest to the input shadow mask with maximum IoU is chosen as the initial estimation.

From this first estimation, our method looks for the best directional light vector within a search space. The search space is represented as a sphere cap $S'_{\mathbf{v}}$ of the unit sphere S centered at the origin. $S'_{\mathbf{v}}$ is centered at vector $\mathbf{v}$, which represents the initial directional light estimation. The method then recursively enhances the directional light by finding the maximum by recursively subdividing and resampling the sphere cap.

The results in Section 5.2 show that the IoU value always increases compared to the input IoU, and in most cases by a significant margin. The angle difference to the GT or EGT also decreases in most cases, although not in all, since it depends on the quality of the input. Section 5.3 shows that, even if the resulting error (angle to GT or EGT) is high, the visual shadow result seems coherent.

It is visually noticeable that our method depends on the shadow input. This is shown by qualitative analysis in Section 5.3. When an input shadow, that is somewhat on the same direction, is provided and has the same length as the ground truth, the method outputs a visually coherent directional light vector and enhances the shadow by rendering it with correct shape.

When unreliable input is provided, however, the method carries over its imprecise features, namely the incorrect length or direction of the shadow. As far as our experiments show, this is the most significant limitation, and it can lead to a visually incorrect output. Nonetheless, even when given a misleading input, the method can slightly improve it by delivering a shadow of correct shape.

Furthermore, because the output is the directional light vector and not an estimated shadow, it can be employed in AR applications. Since the vector is relative to the fiducial marker, the method wouldn't need to be run again if the camera changes position, or even if the fiducial marker disappears and reappears in the scene.

6.1 FUTURE WORKS

The method developed in this work has a major limitation, which is its dependence on the shadow input. This problem can be approached by future works by experimenting with multiple shadow estimators. The method would extract from the ensemble which shadow and vector are best. Another possibility is to integrate a new shadow estimation method to the pipeline. An integrated shadow estimator can target only high quality shadow direction and length.

Another interesting approach to future works would be to develop a new heuristic. Our method decides the next step recursively by choosing the section of search space $S'_{\mathbf{v}}$ which has the highest integral. By refining this criterion or even the subdivision scheme, the method could have a higher accuracy retrieving a local IoU maximum.

REFERENCES

1 CHEN, Yunqiang et al. An overview of augmented reality technology. In: **Journal of Physics: Conference Series**. IOP Publishing, 2019. p. 022082.

2 AZUMA, Ronald T. A survey of augmented reality. **Presence: teleoperators & virtual environments**, v. 6, n. 4, p. 355-385, 1997.

3 XIONG, Jianghao et al. Augmented reality and virtual reality displays: emerging technologies and future perspectives. **Light: Science & Applications**, v. 10, n. 1, p. 216, 2021.

4 KÁN, Peter; KAFUMANN, Hannes. Deeplight: light source estimation for augmented reality using deep learning. **The Visual Computer**, v. 35, p. 873-883, 2019.

5 FRAHM, Jan-Michael et al. Markerless augmented reality with light source estimation for direct illumination. In: **Conference on Visual Media Production CVMP, London**. Stevenage: IET, 2005. p. 211-220.

6 WHELAN, Thomas et al. ElasticFusion: Real-time dense SLAM and light source estimation. **The International Journal of Robotics Research**, v. 35, n. 14, p. 1697-1716, 2016.

7 MEILLAND, Maxime; BARAT, Christian; COMPORT, Andrew. 3d high dynamic range dense visual slam and its application to real-time object re-lighting. In: **2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)**. IEEE, 2013. p. 143-152.

8 FIALA, Mark. ARTag, a fiducial marker system using digital techniques. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. IEEE, 2005. p. 590-596.

9 LIU, Daquan et al. Arshadowgan: Shadow generative adversarial network for augmented reality in single light scenes. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2020. p. 8139-8148.

10 KHAN, Salman Hameed et al. Automatic feature learning for robust shadow detection. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. IEEE, 2014. p. 1939-1946.

11 KHAN, Salman H. et al. Automatic shadow detection and removal from a single image. **IEEE transactions on pattern analysis and machine intelligence**, v. 38, n. 3, p. 431-446, 2015.

12 HOU, Le et al. Large scale shadow annotation and detection using lazy annotation and stacked CNNs. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 43, n. 4, p. 1337-1351, 2019.

13 KURBATOVA, E. E.; LYALINA, V. A. Shadow detection on color images. In: **Journal of Physics: Conference Series**. IOP Publishing, 2019. p. 032018.

14 Otsu N. A threshold selection method from gray-level histograms. **IEEE transactions on systems, man, and cybernetics**. 1979 Jan;9(1):62-6.

15 SUZUKI, Satoshi et al. Topological structural analysis of digitized binary images by border following. **Computer vision, graphics, and image processing**, v. 30, n. 1, p. 32-46, 1985.

16 O. Rodrigues, **Journal de Mathématiques** 5, p. 380, 1840.

17 UREÑA, Carlos; GEORGIEV, Iliyan. Stratified sampling of projected spherical caps. In: **Computer Graphics Forum**. 2018. p. 13-20.

18 STAUDER, Jürgen. Point light source estimation from two images and its limits. **International Journal of Computer Vision**, v. 36, p. 195-220, 2000.

19 MARQUES, Bruno AD et al. Deep Light Source Estimation for Mixed Reality. In: **Visigrapp (1: grapp)**. 2018. p. 303-311.

20 LIU, Damon Shing-Min; WU, Shao-Jun. Light direction estimation and hand touchable interaction for augmented reality. **Virtual Reality**, v. 26, n. 3, p. 1155-1172, 2022.

21 LEGENDRE, Chloe et al. Deeplight: Learning illumination for unconstrained mobile mixed reality. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2019. p. 5918-5928.

22 SUN, Yu-ke et al. Learning Illumination from a Limited Field-of-View Image. In: **2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)**. IEEE, 2020. p. 1-6.

23 WANG, Zian et al. Neural Light Field Estimation for Street Scenes with Differentiable Virtual Object Insertion. In: **Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II**. Cham: Springer Nature Switzerland, 2022. p. 380-397.

24 LIU, Celong et al. Real-time lighting estimation for augmented reality via differentiable screen-space rendering. **IEEE Transactions on Visualization and Computer Graphics**, v. 29, n. 4, p. 2132-2145, 2022.

25 MARQUES, Bruno Augusto Dorta et al. Spatially and color consistent environment lighting estimation using deep neural networks for mixed reality. **Computers & Graphics**, v. 102, p. 257-268, 2022.

26 CAO, Xiaochun; FOROOSH, Hassan. Camera calibration and light source orientation from solar shadows. **Computer Vision and Image Understanding**, v. 105, n. 1, p. 60-72, 2007.

27 KOC, Emre; BALCISOY, Selim. Estimation of environmental lighting from known geometries for mobile augmented reality. In: **2013 International Conference on Cyberworlds**. IEEE, 2013. p. 132-139.

28 WANG, Yang; SAMARAS, Dimitris. Estimation of multiple directional light sources for synthesis of mixed reality images. In: **10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings**. IEEE, 2002. p. 38-47.

29 BOOM, Bastiaan J. et al. Interactive light source position estimation for augmented reality with an RGB-D camera. **Computer Animation and Virtual Worlds**, v. 28, n. 1, p. e1686, 2017.

30 NGUYEN, Rang MH; LE, Minh Ngoc. Light source estimation from a single image. In: **2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)**. IEEE, 2012. p. 1358-1363.

31 LOPEZ-MORENO, Jorge et al. Multiple light source estimation in a single image. In: **Computer Graphics Forum**. 2013. p. 170-182.

32 CHOTIKAKAMTHORN, Nopporn. Near point light source location estimation from shadow edge correspondence. In: **2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)**. IEEE, 2015. p. 30-35.

33 GRUBER, Lukas; RICHTER-TRUMMER, Thomas; SCHMALSTIEG, Dieter. Real-time photometric registration from arbitrary geometry. In: **2012 IEEE international symposium on mixed and augmented reality (ISMAR)**. IEEE, 2012. p. 119-128.

34 AbhinayE, "Face detection using opencv," June 2012.

35 YU, Ye; SMITH, William AP. Inverserendernet: Learning single image inverse rendering. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.** 2019. p. 3155-3164.

36 CROW, Franklin C. Summed-area tables for texture mapping. In: **Proceedings of the 11th annual conference on Computer graphics and interactive techniques.** 1984. p. 207-212.

37 COLAÏTIS, Arnaud et al. Adaptive inverse ray-tracing for accurate and efficient modeling of cross beam energy transfer in hydrodynamics simulations. **Physics of Plasmas**, v. 26, n. 7, p. 072706, 2019.

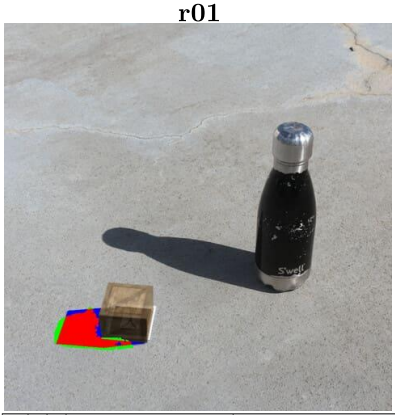
38 PATOW, Gustavo; PUEYO, Xavier. A survey of inverse rendering problems. In: **Computer graphics forum.** Oxford, UK and Boston, USA: Blackwell Publishing, Inc, 2003. p. 663-687.

39 KATO, Hiroharu et al. Differentiable rendering: A survey. **arXiv preprint arXiv:2006.12057**, 2020.

40 AZINOVIC, Dejan et al. Inverse path tracing for joint material and lighting estimation. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2019. p. 2447-2456.

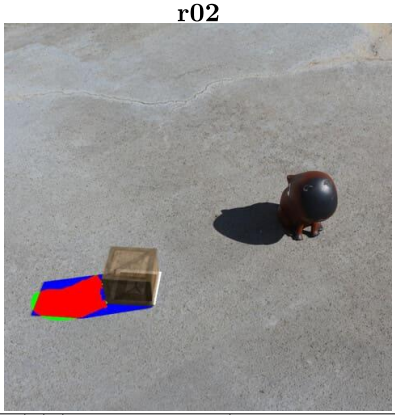
A Ablation study results with fixed resolution 129 x 513

r01



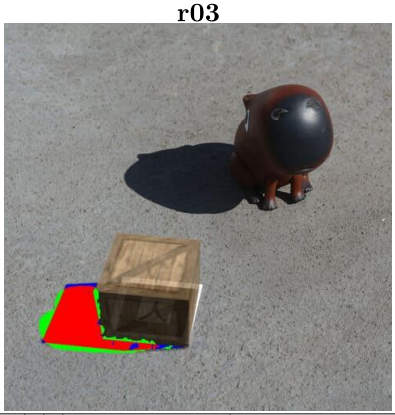
α	r	s	IoU	Angle to EGT
-	0	0	0.653 (0.00%)	28.26° (0.00%)
5	1	0	0.708 (8.35%)	25.23° (10.71%)
5	2	1	0.718 (9.95%)	25.00° (11.51%)
5	2	2	0.718 (9.95%)	24.52° (13.22%)
5	2	3	0.719 (10.02%)	24.26° (14.16%)
5	3	1	0.718 (10.00%)	25.01° (11.51%)
5	3	2	0.718 (10.00%)	25.01° (11.48%)
5	3	3	0.718 (9.97%)	24.42° (13.58%)
15	1	0	0.716 (9.59%)	25.07° (11.28%)
15	2	1	0.716 (9.67%)	25.01° (11.51%)
15	2	2	0.718 (9.92%)	25.26° (10.59%)
15	2	3	0.718 (9.92%)	24.37° (13.76%)
15	3	1	0.717 (9.85%)	25.03° (11.43%)
15	3	2	0.719 (10.02%)	**24.25° (14.17%)**
15	3	3	0.717 (9.79%)	25.01° (11.51%)
30	1	0	0.713 (9.22%)	25.31° (10.44%)
30	2	1	0.713 (9.09%)	24.82° (12.18%)
30	2	2	0.716 (9.64%)	25.57° (9.51%)
30	2	3	0.717 (9.80%)	24.33° (13.90%)
30	3	1	0.716 (9.59%)	24.54° (13.17%)
30	3	2	0.717 (9.81%)	25.56° (9.53%)
30	3	3	0.719 (10.02%)	24.26° (14.13%)
45	1	0	0.714 (9.39%)	24.88° (11.96%)
45	2	1	0.712 (9.07%)	25.18° (10.90%)
45	2	2	0.716 (9.69%)	24.59° (12.98%)
45	2	3	0.716 (9.67%)	24.33° (13.90%)
45	3	1	0.712 (8.98%)	25.32° (10.41%)
45	3	2	0.704 (7.82%)	26.27° (7.02%)
45	3	3	**0.719 (10.04%)**	25.25° (10.65%)

r02



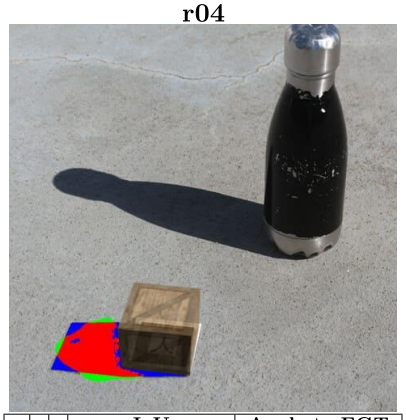
α	r	s	IoU	Angle to EGT
-	0	0	0.519 (0.00%)	**27.21° (0.00%)**
5	1	0	0.595 (14.55%)	28.51° (-4.77%)
5	2	1	0.603 (16.21%)	28.35° (-4.18%)
5	2	2	0.607 (17.01%)	28.93° (-6.30%)
5	2	3	0.609 (17.35%)	29.05° (-6.74%)
5	3	1	0.608 (17.18%)	28.97° (-6.45%)
5	3	2	0.609 (17.27%)	29.04° (-6.72%)
5	3	3	0.610 (17.43%)	29.04° (-6.70%)
15	1	0	0.611 (17.62%)	28.87° (-6.08%)
15	2	1	0.610 (17.59%)	28.97° (-6.46%)
15	2	2	0.610 (17.54%)	28.77° (-5.72%)
15	2	3	0.611 (17.62%)	28.92° (-6.27%)
15	3	1	0.611 (17.76%)	28.53° (-4.82%)
15	3	2	0.612 (17.95%)	29.04° (-6.70%)
15	3	3	0.612 (17.79%)	28.72° (-5.54%)
30	1	0	0.609 (17.36%)	28.74° (-5.61%)
30	2	1	0.608 (17.14%)	28.97° (-6.47%)
30	2	2	0.609 (17.37%)	28.82° (-5.88%)
30	2	3	0.611 (17.76%)	28.81° (-5.86%)
30	3	1	0.610 (17.59%)	28.97° (-6.46%)
30	3	2	0.612 (17.90%)	29.04° (-6.72%)
30	3	3	**0.613 (18.01%)**	28.77° (-5.70%)
45	1	0	0.605 (16.64%)	28.62° (-5.16%)
45	2	1	0.600 (15.66%)	29.65° (-8.93%)
45	2	2	0.610 (17.48%)	28.95° (-6.36%)
45	2	3	0.612 (17.82%)	29.05° (-6.74%)
45	3	1	0.609 (17.40%)	28.95° (-6.37%)
45	3	2	0.610 (17.59%)	29.07° (-6.83%)
45	3	3	0.612 (17.90%)	28.71° (-5.51%)

r03



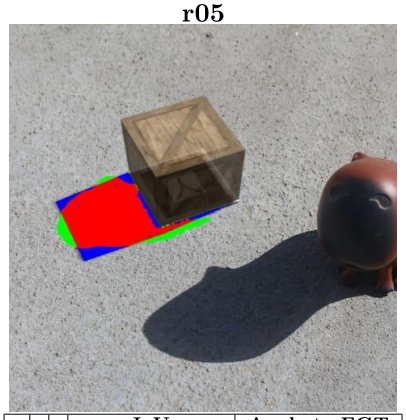
α	r	s	IoU	Angle to EGT
-	0	0	0.493 (0.00%)	36.75° (0.00%)
5	1	0	0.572 (15.92%)	33.94° (7.66%)
5	2	1	0.604 (22.44%)	32.96° (10.33%)
5	2	2	0.623 (26.34%)	32.16° (12.50%)
5	2	3	0.635 (28.74%)	31.80° (13.48%)
5	3	1	0.622 (26.09%)	32.39° (11.87%)
5	3	2	0.638 (29.39%)	31.71° (13.72%)
5	3	3	0.644 (30.55%)	31.40° (14.56%)
15	1	0	0.720 (45.92%)	28.90° (21.37%)
15	2	1	0.747 (51.50%)	26.65° (27.47%)
15	2	2	0.734 (48.83%)	27.79° (24.38%)
15	2	3	0.718 (45.52%)	25.93° (29.45%)
15	3	1	0.747 (51.42%)	27.29° (25.74%)
15	3	2	**0.748 (51.61%)**	26.36° (28.27%)
15	3	3	0.739 (49.88%)	26.19° (28.74%)
30	1	0	0.743 (50.59%)	26.10° (28.98%)
30	2	1	0.746 (51.31%)	27.59° (24.94%)
30	2	2	0.736 (49.24%)	26.87° (26.89%)
30	2	3	0.699 (41.74%)	25.25° (31.31%)
30	3	1	0.747 (51.48%)	27.34° (25.61%)
30	3	2	0.747 (51.42%)	27.61° (24.89%)
30	3	3	0.745 (51.02%)	26.49° (27.92%)
45	1	0	0.743 (50.59%)	26.10° (28.98%)
45	2	1	0.744 (50.73%)	27.45° (25.31%)
45	2	2	0.747 (51.45%)	27.40° (25.43%)
45	2	3	0.671 (35.96%)	**24.41° (33.57%)**
45	3	1	0.743 (50.66%)	26.79° (27.11%)
45	3	2	0.746 (51.29%)	27.62° (24.84%)
45	3	3	0.732 (48.42%)	25.43° (30.80%)

r04



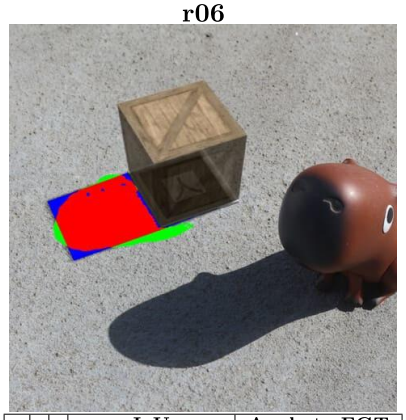
α	r	s	IoU	Angle to EGT
-	0	0	0.586 (0.00%)	21.88° (0.00%)
5	1	0	0.604 (3.06%)	19.79° (9.57%)
5	2	1	0.604 (3.14%)	20.59° (5.92%)
5	2	2	0.604 (3.10%)	18.68° (14.65%)
5	2	3	0.603 (2.98%)	19.77° (9.63%)
5	3	1	0.604 (3.12%)	19.55° (10.67%)
5	3	2	**0.605 (3.22%)**	19.55° (10.68%)
5	3	3	0.604 (3.15%)	19.74° (9.79%)
15	1	0	0.603 (2.95%)	20.65° (5.62%)
15	2	1	0.602 (2.77%)	18.62° (14.92%)
15	2	2	0.604 (3.16%)	20.61° (5.81%)
15	2	3	0.603 (2.95%)	19.67° (10.11%)
15	3	1	0.603 (3.01%)	19.57° (10.58%)
15	3	2	0.604 (3.18%)	19.73° (9.83%)
15	3	3	0.604 (3.08%)	19.36° (11.54%)
30	1	0	0.603 (2.95%)	20.65° (5.62%)
30	2	1	0.599 (2.31%)	18.28° (16.48%)
30	2	2	0.602 (2.83%)	20.78° (5.02%)
30	2	3	0.604 (3.05%)	18.23° (16.69%)
30	3	1	0.603 (2.96%)	18.93° (13.48%)
30	3	2	0.604 (3.10%)	19.85° (9.27%)
30	3	3	0.604 (3.08%)	19.36° (11.54%)
45	1	0	0.603 (2.95%)	20.65° (5.62%)
45	2	1	0.577 (-1.48%)	**16.66° (23.85%)**
45	2	2	0.601 (2.66%)	20.21° (7.65%)
45	2	3	0.604 (3.06%)	19.62° (10.32%)
45	3	1	0.598 (2.15%)	17.95° (17.98%)
45	3	2	0.604 (3.05%)	20.60° (5.85%)
45	3	3	0.604 (3.12%)	19.73° (9.81%)

r05



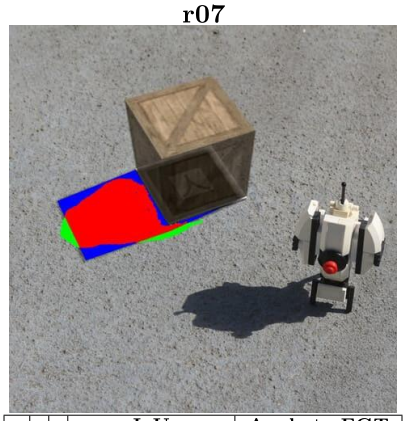
α	r	s	IoU	Angle to EGT
-	0	0	0.517 (0.00%)	25.41° (0.00%)
5	1	0	0.589 (14.01%)	22.60° (11.03%)
5	2	1	0.605 (17.08%)	21.74° (14.42%)
5	2	2	0.631 (21.99%)	20.82° (18.04%)
5	2	3	0.620 (19.90%)	20.70° (18.53%)
5	3	1	0.616 (19.14%)	21.39° (15.82%)
5	3	2	0.637 (23.27%)	20.35° (19.90%)
5	3	3	0.625 (20.89%)	20.45° (19.50%)
15	1	0	0.678 (31.17%)	17.87° (29.67%)
15	2	1	0.689 (33.35%)	16.55° (34.88%)
15	2	2	0.686 (32.76%)	14.58° (42.61%)
15	2	3	0.687 (32.92%)	14.42° (43.25%)
15	3	1	**0.691 (33.61%)**	16.75° (34.07%)
15	3	2	0.689 (33.21%)	14.47° (43.04%)
15	3	3	0.689 (33.22%)	14.80° (41.76%)
30	1	0	0.688 (33.15%)	17.36° (31.67%)
30	2	1	0.690 (33.41%)	16.71° (34.22%)
30	2	2	0.690 (33.40%)	15.79° (37.87%)
30	2	3	0.651 (26.04%)	**10.85° (57.30%)**
30	3	1	0.689 (33.35%)	16.57° (34.80%)
30	3	2	0.690 (33.47%)	15.72° (38.12%)
30	3	3	0.664 (28.41%)	11.97° (52.88%)
45	1	0	0.688 (33.15%)	16.61° (34.61%)
45	2	1	0.690 (33.45%)	17.04° (32.93%)
45	2	2	0.690 (33.51%)	15.72° (38.14%)
45	2	3	0.682 (32.02%)	13.48° (46.93%)
45	3	1	0.689 (33.34%)	16.67° (34.39%)
45	3	2	0.690 (33.47%)	16.75° (34.06%)
45	3	3	0.689 (33.21%)	14.47° (43.03%)

r06



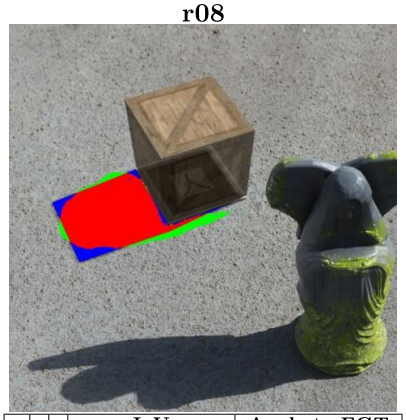
α	r	s	IoU	Angle to EGT
-	0	0	0.667 (0.00%)	16.25° (0.00%)
5	1	0	0.703 (5.33%)	13.20° (18.73%)
5	2	1	0.703 (5.43%)	12.68° (21.93%)
5	2	2	0.703 (5.44%)	12.26° (24.54%)
5	2	3	0.703 (5.44%)	12.26° (24.56%)
5	3	1	0.704 (5.53%)	13.22° (18.64%)
5	3	2	0.704 (5.47%)	12.22° (24.80%)
5	3	3	0.704 (5.48%)	12.26° (24.53%)
15	1	0	0.703 (5.33%)	13.20° (18.73%)
15	2	1	**0.704 (5.59%)**	13.19° (18.83%)
15	2	2	0.704 (5.51%)	12.73° (21.68%)
15	2	3	0.703 (5.45%)	12.12° (25.41%)
15	3	1	0.704 (5.52%)	12.73° (21.67%)
15	3	2	0.703 (5.43%)	12.12° (25.42%)
15	3	3	0.704 (5.56%)	12.73° (21.67%)
30	1	0	0.702 (5.30%)	12.94° (20.39%)
30	2	1	0.703 (5.31%)	12.98° (20.13%)
30	2	2	0.703 (5.36%)	12.73° (21.67%)
30	2	3	0.703 (5.38%)	12.64° (22.21%)
30	3	1	0.702 (5.22%)	12.62° (22.35%)
30	3	2	0.703 (5.40%)	12.68° (21.95%)
30	3	3	**0.704 (5.59%)**	13.19° (18.79%)
45	1	0	0.702 (5.30%)	12.94° (20.39%)
45	2	1	0.702 (5.20%)	13.42° (17.41%)
45	2	2	0.702 (5.28%)	12.15° (25.23%)
45	2	3	0.702 (5.30%)	**12.10° (25.50%)**
45	3	1	0.703 (5.38%)	12.71° (21.76%)
45	3	2	0.704 (5.46%)	12.69° (21.89%)
45	3	3	0.704 (5.51%)	13.24° (18.54%)

r07



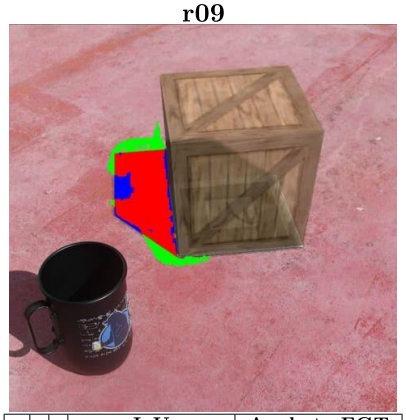
α	r	s	IoU	Angle to EGT
-	0	0	0.566 (0.00%)	20.12° (0.00%)
5	1	0	0.617 (9.11%)	16.88° (16.08%)
5	2	1	0.620 (9.57%)	16.74° (16.79%)
5	2	2	0.633 (11.80%)	15.49° (23.01%)
5	2	3	0.625 (10.42%)	15.52° (22.84%)
5	3	1	0.625 (10.46%)	16.02° (20.39%)
5	3	2	0.635 (12.15%)	15.33° (23.80%)
5	3	3	0.626 (10.64%)	15.29° (23.99%)
15	1	0	0.639 (12.99%)	13.94° (30.72%)
15	2	1	0.640 (13.09%)	14.30° (28.91%)
15	2	2	0.634 (12.11%)	**12.54° (37.69%)**
15	2	3	0.638 (12.78%)	13.34° (33.70%)
15	3	1	0.639 (13.02%)	14.71° (26.88%)
15	3	2	0.640 (13.03%)	13.74° (31.70%)
15	3	3	0.641 (13.23%)	13.56° (32.61%)
30	1	0	0.638 (12.76%)	14.20° (29.44%)
30	2	1	0.639 (12.93%)	14.09° (29.97%)
30	2	2	0.640 (13.14%)	13.53° (32.77%)
30	2	3	0.640 (13.11%)	14.33° (28.77%)
30	3	1	0.640 (13.13%)	14.54° (27.74%)
30	3	2	0.641 (13.21%)	14.56° (27.64%)
30	3	3	0.641 (13.32%)	14.37° (28.59%)
45	1	0	0.638 (12.79%)	14.46° (28.15%)
45	2	1	0.640 (13.15%)	14.30° (28.93%)
45	2	2	0.639 (13.02%)	14.12° (29.81%)
45	2	3	0.640 (13.06%)	14.06° (30.10%)
45	3	1	0.639 (13.00%)	14.14° (29.73%)
45	3	2	0.640 (13.12%)	14.61° (27.36%)
45	3	3	**0.641 (13.34%)**	14.38° (28.54%)

r08



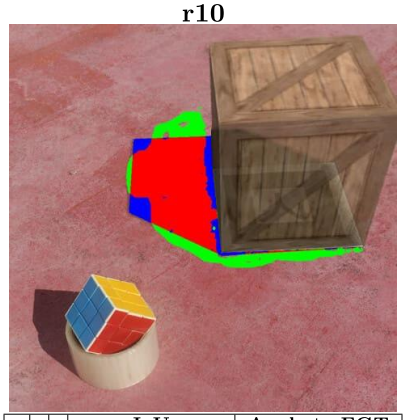
α	r	s	IoU	Angle to EGT
-	0	0	0.512 (0.00%)	23.02° (0.00%)
5	1	0	0.587 (14.85%)	19.97° (13.25%)
5	2	1	0.598 (16.90%)	19.34° (15.98%)
5	2	2	0.624 (21.97%)	18.27° (20.63%)
5	2	3	0.614 (19.96%)	18.20° (20.94%)
5	3	1	0.610 (19.35%)	18.90° (17.90%)
5	3	2	0.632 (23.55%)	17.99° (21.82%)
5	3	3	0.622 (21.67%)	17.90° (22.23%)
15	1	0	0.696 (36.16%)	15.47° (32.80%)
15	2	1	0.706 (38.12%)	13.99° (39.22%)
15	2	2	0.691 (35.15%)	13.33° (42.07%)
15	2	3	0.692 (35.35%)	12.18° (47.09%)
15	3	1	0.707 (38.15%)	14.02° (39.09%)
15	3	2	0.698 (36.38%)	12.68° (44.91%)
15	3	3	0.701 (37.08%)	12.88° (44.03%)
30	1	0	0.706 (38.08%)	14.10° (38.73%)
30	2	1	0.706 (37.95%)	14.02° (39.07%)
30	2	2	0.706 (38.02%)	14.02° (39.09%)
30	2	3	0.693 (35.55%)	12.27° (46.67%)
30	3	1	0.706 (38.03%)	14.06° (38.93%)
30	3	2	**0.707 (38.16%)**	14.10° (38.73%)
30	3	3	0.705 (37.89%)	13.59° (40.97%)
45	1	0	0.705 (37.82%)	14.29° (37.92%)
45	2	1	0.705 (37.86%)	13.67° (40.60%)
45	2	2	0.706 (38.07%)	14.30° (37.87%)
45	2	3	0.684 (33.67%)	**11.68° (49.27%)**
45	3	1	0.707 (38.14%)	14.24° (38.11%)
45	3	2	0.706 (38.02%)	14.48° (37.08%)
45	3	3	0.701 (37.11%)	13.29° (42.28%)

r09



α	r	s	IoU	Angle to EGT
-	0	0	0.301 (0.00%)	23.24° (0.00%)
5	1	0	0.328 (8.88%)	22.64° (2.60%)
5	2	1	0.340 (12.65%)	22.40° (3.64%)
5	2	2	0.344 (14.29%)	24.45° (-5.19%)
5	2	3	0.349 (15.68%)	23.40° (-0.70%)
5	3	1	0.345 (14.40%)	22.56° (2.94%)
5	3	2	0.350 (16.24%)	24.56° (-5.68%)
5	3	3	0.354 (17.56%)	23.24° (0.00%)
15	1	0	0.398 (32.03%)	20.79° (10.56%)
15	2	1	0.428 (41.96%)	21.17° (8.93%)
15	2	2	0.471 (56.10%)	18.29° (21.31%)
15	2	3	0.460 (52.57%)	19.83° (14.68%)
15	3	1	0.431 (42.87%)	22.80° (1.90%)
15	3	2	0.477 (58.24%)	19.29° (16.98%)
15	3	3	0.472 (56.48%)	19.67° (15.37%)
30	1	0	0.508 (68.70%)	15.10° (35.05%)
30	2	1	0.490 (62.62%)	18.58° (20.05%)
30	2	2	0.567 (88.26%)	15.48° (33.38%)
30	2	3	0.603 (99.92%)	14.28° (38.55%)
30	3	1	0.544 (80.44%)	16.11° (30.70%)
30	3	2	0.605 (100.88%)	13.32° (42.69%)
30	3	3	**0.618 (105.14%)**	13.18° (43.29%)
45	1	0	0.606 (101.02%)	**12.53° (46.10%)**
45	2	1	0.617 (104.61%)	13.32° (42.71%)
45	2	2	0.532 (76.47%)	17.71° (23.79%)
45	2	3	0.462 (53.16%)	20.52° (11.70%)
45	3	1	0.617 (104.59%)	13.36° (42.50%)
45	3	2	0.618 (104.88%)	13.17° (43.33%)
45	3	3	0.509 (68.78%)	18.15° (21.90%)

r10



α	r	s	IoU	Angle to EGT
-	0	0	0.396 (0.00%)	35.07° (0.00%)
5	1	0	0.404 (1.84%)	36.43° (-3.86%)
5	2	1	0.405 (2.11%)	36.72° (-4.69%)
5	2	2	0.404 (2.03%)	37.05° (-5.63%)
5	2	3	0.396 (-0.07%)	38.15° (-8.77%)
5	3	1	0.405 (2.30%)	36.53° (-4.17%)
5	3	2	0.404 (1.94%)	36.60° (-4.34%)
5	3	3	0.399 (0.79%)	37.74° (-7.59%)
15	1	0	0.417 (5.25%)	28.30° (19.32%)
15	2	1	0.449 (13.24%)	24.20° (31.00%)
15	2	2	0.445 (12.18%)	25.33° (27.79%)
15	2	3	0.462 (16.63%)	22.85° (34.86%)
15	3	1	0.460 (16.14%)	23.17° (33.93%)
15	3	2	0.463 (16.96%)	22.75° (35.13%)
15	3	3	0.474 (19.65%)	21.42° (38.93%)
30	1	0	0.485 (22.41%)	20.31° (42.10%)
30	2	1	0.528 (33.30%)	13.57° (61.31%)
30	2	2	0.472 (19.10%)	18.56° (47.07%)
30	2	3	0.502 (26.70%)	14.20° (59.50%)
30	3	1	0.538 (35.82%)	10.41° (70.32%)
30	3	2	0.523 (32.03%)	13.42° (61.73%)
30	3	3	0.529 (33.42%)	11.61° (66.89%)
45	1	0	0.528 (33.21%)	14.35° (59.08%)
45	2	1	0.545 (37.61%)	8.66° (75.32%)
45	2	2	0.544 (37.37%)	8.55° (75.64%)
45	2	3	0.540 (36.31%)	9.03° (74.24%)
45	3	1	0.546 (37.81%)	**8.33° (76.26%)**
45	3	2	0.546 (37.87%)	8.34° (76.23%)
45	3	3	**0.547 (37.93%)**	8.38° (76.11%)

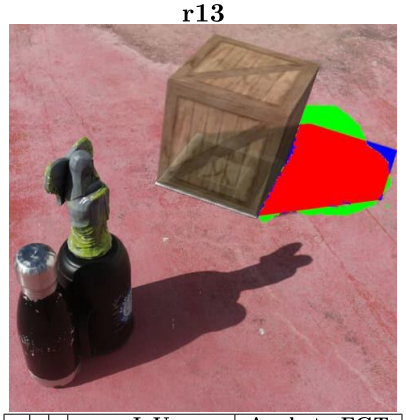
r11

α	r	s	IoU	Angle to EGT
-	0	0	0.470 (0.00%)	34.42° (0.00%)
5	1	0	0.512 (8.80%)	32.02° (6.98%)
5	2	1	0.531 (12.95%)	30.83° (10.42%)
5	2	2	0.542 (15.22%)	29.83° (13.34%)
5	2	3	0.547 (16.41%)	29.61° (13.97%)
5	3	1	0.542 (15.23%)	30.32° (11.90%)
5	3	2	0.552 (17.48%)	29.33° (14.79%)
5	3	3	0.556 (18.26%)	29.15° (15.32%)
15	1	0	0.591 (25.63%)	26.93° (21.75%)
15	2	1	0.636 (35.36%)	23.45° (31.87%)
15	2	2	0.680 (44.66%)	20.21° (41.30%)
15	2	3	0.662 (40.71%)	21.21° (38.38%)
15	3	1	0.658 (39.99%)	22.15° (35.66%)
15	3	2	0.686 (46.01%)	18.47° (46.34%)
15	3	3	0.669 (42.38%)	21.35° (37.97%)
30	1	0	0.681 (44.77%)	19.51° (43.32%)
30	2	1	0.686 (45.99%)	17.49° (49.19%)
30	2	2	0.687 (46.08%)	17.48° (49.21%)
30	2	3	0.670 (42.41%)	13.06° (62.04%)
30	3	1	**0.687 (46.16%)**	17.93° (47.92%)
30	3	2	0.687 (46.12%)	18.26° (46.94%)
30	3	3	0.677 (43.92%)	14.78° (57.05%)
45	1	0	0.687 (46.04%)	17.49° (49.17%)
45	2	1	0.687 (46.10%)	17.37° (49.55%)
45	2	2	0.669 (42.28%)	**12.96° (62.34%)**
45	2	3	0.669 (42.40%)	13.09° (61.97%)
45	3	1	0.687 (46.09%)	18.03° (47.61%)
45	3	2	0.682 (44.99%)	15.51° (54.94%)
45	3	3	0.681 (44.92%)	15.29° (55.57%)

r12

α	r	s	IoU	Angle to EGT
-	0	0	0.574 (0.00%)	24.89° (0.00%)
5	1	0	0.576 (0.41%)	25.65° (-3.03%)
5	2	1	0.578 (0.74%)	28.70° (-15.27%)
5	2	2	0.576 (0.44%)	25.66° (-3.09%)
5	2	3	0.577 (0.56%)	25.55° (-2.64%)
5	3	1	0.576 (0.44%)	26.30° (-5.66%)
5	3	2	0.577 (0.51%)	26.29° (-5.62%)
5	3	3	0.577 (0.46%)	25.67° (-3.13%)
15	1	0	0.594 (3.41%)	34.20° (-37.39%)
15	2	1	0.594 (3.44%)	34.54° (-38.76%)
15	2	2	0.584 (1.83%)	31.10° (-24.93%)
15	2	3	0.576 (0.43%)	26.31° (-5.68%)
15	3	1	**0.594 (3.56%)**	34.10° (-36.97%)
15	3	2	0.589 (2.60%)	32.82° (-31.85%)
15	3	3	0.576 (0.28%)	26.20° (-5.26%)
30	1	0	0.593 (3.41%)	34.48° (-38.52%)
30	2	1	0.593 (3.39%)	33.71° (-35.41%)
30	2	2	0.593 (3.40%)	34.22° (-37.47%)
30	2	3	0.594 (3.47%)	34.27° (-37.68%)
30	3	1	0.594 (3.48%)	34.17° (-37.26%)
30	3	2	0.587 (2.28%)	32.35° (-29.96%)
30	3	3	0.593 (3.39%)	33.57° (-34.86%)
45	1	0	0.594 (3.41%)	34.20° (-37.39%)
45	2	1	0.556 (-3.19%)	**18.21° (26.85%)**
45	2	2	0.570 (-0.63%)	23.55° (5.38%)
45	2	3	0.572 (-0.41%)	22.65° (9.02%)
45	3	1	0.561 (-2.30%)	19.17° (23.01%)
45	3	2	0.568 (-0.98%)	20.64° (17.10%)
45	3	3	0.575 (0.19%)	24.96° (-0.25%)

r13

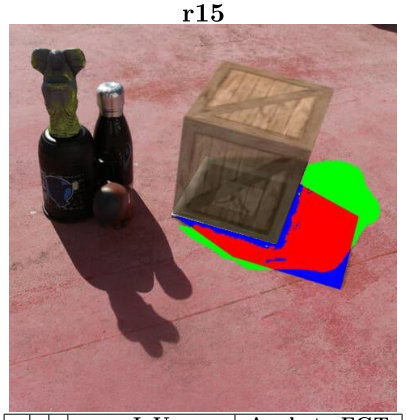


α	r	s	IoU	Angle to EGT
-	0	0	0.519 (0.00%)	12.71° (0.00%)
5	1	0	0.595 (14.75%)	9.79° (22.91%)
5	2	1	0.614 (18.27%)	9.92° (21.90%)
5	2	2	0.635 (22.32%)	8.54° (32.76%)
5	2	3	0.635 (22.47%)	9.19° (27.70%)
5	3	1	0.620 (19.51%)	9.85° (22.50%)
5	3	2	0.647 (24.59%)	8.16° (35.77%)
5	3	3	0.642 (23.80%)	8.94° (29.61%)
15	1	0	0.701 (35.10%)	4.77° (62.49%)
15	2	1	0.704 (35.62%)	4.19° (67.02%)
15	2	2	0.706 (36.12%)	3.32° (73.84%)
15	2	3	0.682 (31.35%)	5.71° (55.03%)
15	3	1	0.706 (36.01%)	3.94° (69.02%)
15	3	2	**0.707 (36.27%)**	2.48° (80.52%)
15	3	3	0.691 (33.11%)	5.13° (59.66%)
30	1	0	0.705 (35.93%)	3.30° (74.05%)
30	2	1	0.704 (35.66%)	3.67° (71.11%)
30	2	2	0.706 (36.05%)	2.87° (77.44%)
30	2	3	0.697 (34.28%)	3.02° (76.26%)
30	3	1	0.706 (36.13%)	2.70° (78.72%)
30	3	2	0.707 (36.17%)	2.95° (76.76%)
30	3	3	0.703 (35.48%)	2.38° (81.24%)
45	1	0	0.704 (35.72%)	3.25° (74.46%)
45	2	1	0.703 (35.50%)	3.48° (72.59%)
45	2	2	0.706 (36.07%)	2.91° (77.08%)
45	2	3	0.706 (36.13%)	2.53° (80.12%)
45	3	1	0.707 (36.16%)	2.55° (79.93%)
45	3	2	0.703 (35.44%)	**2.28° (82.05%)**
45	3	3	0.707 (36.24%)	2.51° (80.25%)

r14

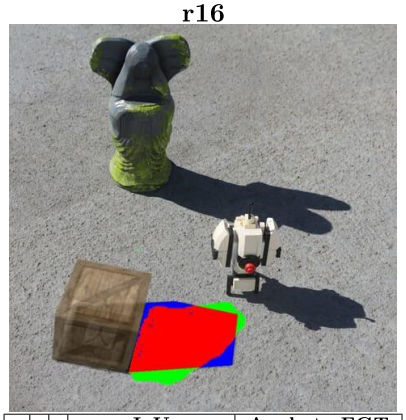
α	r	s	IoU	Angle to EGT
-	0	0	0.455 (0.00%)	30.01° (0.00%)
5	1	0	0.498 (9.45%)	28.04° (6.58%)
5	2	1	0.519 (13.91%)	27.01° (10.01%)
5	2	2	0.528 (15.87%)	25.56° (14.83%)
5	2	3	0.532 (16.89%)	25.63° (14.62%)
5	3	1	0.526 (15.54%)	26.56° (11.52%)
5	3	2	0.534 (17.27%)	24.98° (16.76%)
5	3	3	0.538 (18.16%)	25.44° (15.25%)
15	1	0	0.543 (19.23%)	24.69° (17.75%)
15	2	1	0.546 (19.86%)	24.41° (18.67%)
15	2	2	0.533 (16.96%)	22.25° (25.86%)
15	2	3	0.504 (10.56%)	20.49° (31.73%)
15	3	1	0.542 (18.92%)	25.29° (15.75%)
15	3	2	0.544 (19.35%)	23.45° (21.86%)
15	3	3	0.521 (14.44%)	21.66° (27.83%)
30	1	0	0.544 (19.51%)	24.22° (19.30%)
30	2	1	0.545 (19.78%)	24.35° (18.88%)
30	2	2	0.535 (17.41%)	22.28° (25.77%)
30	2	3	0.496 (8.95%)	20.38° (32.11%)
30	3	1	0.546 (19.86%)	24.41° (18.67%)
30	3	2	**0.546 (19.90%)**	24.40° (18.69%)
30	3	3	0.522 (14.69%)	21.95° (26.85%)
45	1	0	0.544 (19.42%)	24.08° (19.77%)
45	2	1	0.543 (19.17%)	24.57° (18.12%)
45	2	2	0.545 (19.64%)	24.41° (18.66%)
45	2	3	0.484 (6.20%)	**19.37° (35.47%)**
45	3	1	0.545 (19.65%)	24.36° (18.84%)
45	3	2	0.539 (18.33%)	26.00° (13.36%)
45	3	3	0.513 (12.56%)	21.78° (27.45%)

r15



α	r	s	IoU	Angle to EGT
-	0	0	0.440 (0.00%)	29.97° (0.00%)
5	1	0	0.481 (9.30%)	27.76° (7.37%)
5	2	1	0.500 (13.55%)	26.92° (10.17%)
5	2	2	0.510 (16.01%)	25.79° (13.97%)
5	2	3	0.513 (16.59%)	25.73° (14.15%)
5	3	1	0.507 (15.30%)	26.65° (11.08%)
5	3	2	0.516 (17.21%)	25.04° (16.45%)
5	3	3	0.515 (17.17%)	25.26° (15.71%)
15	1	0	0.521 (18.33%)	24.17° (19.36%)
15	2	1	0.521 (18.38%)	23.21° (22.58%)
15	2	2	0.518 (17.70%)	22.50° (24.94%)
15	2	3	0.489 (11.04%)	20.75° (30.76%)
15	3	1	0.521 (18.49%)	23.63° (21.15%)
15	3	2	**0.521 (18.53%)**	23.62° (21.18%)
15	3	3	0.501 (13.98%)	21.61° (27.91%)
30	1	0	0.520 (18.15%)	23.37° (22.01%)
30	2	1	0.521 (18.45%)	23.89° (20.30%)
30	2	2	0.517 (17.55%)	22.33° (25.50%)
30	2	3	0.481 (9.26%)	20.80° (30.60%)
30	3	1	0.521 (18.38%)	23.21° (22.58%)
30	3	2	**0.521 (18.53%)**	23.63° (21.16%)
30	3	3	0.502 (14.09%)	21.63° (27.84%)
45	1	0	0.521 (18.33%)	24.17° (19.36%)
45	2	1	0.520 (18.20%)	23.47° (21.70%)
45	2	2	0.520 (18.30%)	24.39° (18.61%)
45	2	3	0.472 (7.27%)	**19.91° (33.57%)**
45	3	1	0.521 (18.33%)	23.48° (21.66%)
45	3	2	0.520 (18.21%)	24.15° (19.42%)
45	3	3	0.494 (12.32%)	21.17° (29.38%)

r16



α	r	s	IoU	Angle to EGT
-	0	0	0.623 (0.00%)	**14.63° (0.00%)**
5	1	0	0.682 (9.35%)	16.00° (-9.37%)
5	2	1	0.690 (10.76%)	16.26° (-11.17%)
5	2	2	0.698 (12.04%)	16.27° (-11.22%)
5	2	3	0.698 (11.94%)	16.53° (-13.05%)
5	3	1	0.693 (11.16%)	16.22° (-10.91%)
5	3	2	0.699 (12.22%)	16.26° (-11.15%)
5	3	3	0.700 (12.29%)	16.59° (-13.46%)
15	1	0	0.721 (15.75%)	20.91° (-42.99%)
15	2	1	0.723 (16.07%)	20.18° (-37.98%)
15	2	2	0.722 (15.85%)	20.83° (-42.41%)
15	2	3	0.723 (16.02%)	20.18° (-37.98%)
15	3	1	0.723 (15.93%)	20.92° (-43.01%)
15	3	2	0.724 (16.14%)	20.14° (-37.68%)
15	3	3	0.724 (16.13%)	20.19° (-38.03%)
30	1	0	0.720 (15.50%)	20.03° (-36.97%)
30	2	1	0.722 (15.77%)	20.10° (-37.44%)
30	2	2	0.723 (16.07%)	20.18° (-37.97%)
30	2	3	0.723 (15.93%)	19.96° (-36.44%)
30	3	1	0.723 (16.02%)	20.16° (-37.86%)
30	3	2	0.723 (16.08%)	20.67° (-41.33%)
30	3	3	0.724 (16.11%)	**20.18° (-37.98%)**
45	1	0	0.721 (15.64%)	20.39° (-39.41%)
45	2	1	0.721 (15.61%)	21.00° (-43.62%)
45	2	2	0.723 (15.97%)	20.18° (-37.99%)
45	2	3	0.722 (15.91%)	21.82° (-49.20%)
45	3	1	0.722 (15.82%)	20.57° (-40.68%)
45	3	2	0.723 (15.99%)	20.24° (-38.40%)
45	3	3	**0.724 (16.14%)**	20.85° (-42.56%)