Technical Documentation

Preface

Background Introduction

Diffuse lighting reconstruction is an important problem in computer graphics. Its purpose is to reconstruct a lighting vector from a set of lighting samples for subsequent rendering. In practical applications, due to the noise in the lighting samples, the reconstructed lighting vector is often affected by noise, so it is necessary to denoise the lighting vector.

Purpose

This document aims to introduce the algorithm for diffuse lighting reconstruction, including the encoding, denoising, and reconstruction of the lighting vector, as well as the variance estimation of the denoised lighting vector.

Note:

This algorithm was inspired by the first-order spherical harmonics lighting approximation. It is unclear whether there is a similar algorithm, and it may be original. If it is not original, please point it out in the issue.

Diffuse Lighting Denoising and Reconstruction Algorithm

Model

Definition

Assume a set of $N+1$ dimensional vectors V_i (the vector set itself is denoted as V, the $(N+1)$-th dimension is non-negative) and an energy measurement function $w(v): \mathbb{R}^{N+1} \to \mathbb{R}$, w is defined as

$$w(v) := \sqrt{\sum_{i=1}^{N} v_i^2} + v_{N+1}$$

Define the mapping

$$T : \mathrm{List}(\mathbb{R}^{N+1}) \to \mathbb{R}^{N+1}$$

where List is an ordered set, $T(V)$ is the mapping of all vectors in V to $\mathbb{R}^{N+1}$, and T satisfies:

1. Energy Conservation

   For any $V \in \mathrm{List}(\mathbb{R}^{N+1})$,

$$w(T(V)) = \sum_{i=1}^{n} w(V_i)$$

2. Associativity

   For any $x \in \mathbb{R}^{N+1}$ and $V \in \mathrm{List}(\mathbb{R}^{N+1})$,

$$T(\{x, V\}) = T(\{x, T(V)\})$$

3. Linearity

   For any $x \in \mathbb{R}^{N+1}$, $n \in \mathbb{N}_+$,

$$T(\{x, x, ..., x\}) = nx$$

   where $\{x, x, ..., x\}$ is an ordered set of n x

Let

$$V_i = (\vec{v}_i, I_i)$$

$$\Lambda(V_i) := v_i, I(V_i) := I_i$$

where

$$\vec{v}_i \in \mathbb{R}^N, I_i \in \mathbb{R}_+$$

Then the above mapping T exists, and its form is:

$$T(V) = \left(\sum_{i=1}^{n} v_i, \sum_{i=1}^{n} \omega(V_i) - |\sum_{i=1}^{n} v_i|\right)$$

where

$$\omega(V_i) := |v_i| + I_i$$

Physical Meaning

Select $N = 3$, take V as the set of lighting samples, each sample V_i is a lighting vector, the first three dimensions of V_i are directional lighting, the fourth dimension is ambient lighting, $w(V_i)$ is the energy of the lighting, T is the lighting synthesis operator, $T(V)$ is the synthesized lighting, and T satisfies energy conservation, associativity, and linearity.

Algorithm

Take the input of the algorithm as a set of lighting vectors V input into the BRDF model, and the output as a lighting vector $T(V)$. The goal of the algorithm is to reduce the noise of the lighting vector and reconstruct the lighting vector.

Algorithm Process

1. Encode V_i as $V_i = \left(I * \vec{d}, 0\right)$, where I is the sampled lighting intensity, $\vec{d}$ is the sampled incident lighting direction, and 0 is the ambient lighting intensity
2. Filter V_i in the time domain to obtain $V_{\text{out}} = \frac{1}{n}T(V)$, where n is the number of sampled lighting samples
3. Apply the SVGF algorithm to V_{out} for denoising to obtain V'_{out}
4. Input V'_{out} into the BRDF model to obtain the denoised lighting

Algorithm Implementation

1. Actually encode V_i as $V_i = \left(I * \vec{d}, I\right)$, where I is the sampled lighting intensity, $\vec{d}$ is the sampled incident lighting direction
2. Directly mix V_i during weighted mixing

3. During reconstruction, according to the expression of the T operator, the mixing operation in step 2 is equivalent to the weighted sum of the first three dimensions of V_i and the weighted sum of the fourth dimension, and can correctly construct the result
4. Only apply the T operator during decoding, and in other cases, treat V_i similarly to normal lighting processing

The key code is as follows:

```
struct T
{
    mediump vec4 v_I; // (I * arrow(d), I)
    mediump vec2 CoCg; // (Co, Cg)
};

vec3 project_T_irradiance(T L, vec3 N)
{
    float Y = L.v_I.w;
    float T = Y - L.CoCg.y * 0.5;
    float G = L.CoCg.y + T;
    float B = T - L.CoCg.x * 0.5;
    float R = B + L.CoCg.x;

    vec3 irradiance = vec3(R,G,B) * (max(dot(L.v_I.xyz, N),0) + (Y -
length(L.v_I.xyz))) / (Y+1e-3);
    return max(irradiance, vec3(0.0));
}

T irradiance_to_T(vec3 irradiance, vec3 dir)
{
    T result;
    float Co = irradiance.r - irradiance.b;
    float t = irradiance.b + Co * 0.5;
    float Cg = irradiance.g - t;
    float Y = max(t + Cg * 0.5, 0.0);

    result.CoCg = vec2(Co, Cg);
    result.v_I = vec4(dir * Y,Y);
}

T mix_T(T a, T b, float s)
{
    T result;
    result.v_I = mix(a.v_I, b.v_I, s);
    result.CoCg = mix(a.CoCg, b.CoCg, s);
    return result;
}

T init_T()
{
    T result;
    result.v_I = vec4(0);
    result.CoCg = vec2(0);
    return result;
}
```

```
T scaleT(T A, float x) {
    T tmp;
    tmp.CoCg = A.CoCg * x;
    tmp.v_I = A.v_I * x;
    return tmp;
}

void accumulate_T(inout T accum, T b, float scale)
{
    accum.v_I += b.v_I * scale;
    accum.CoCg += b.CoCg * scale;
}
```

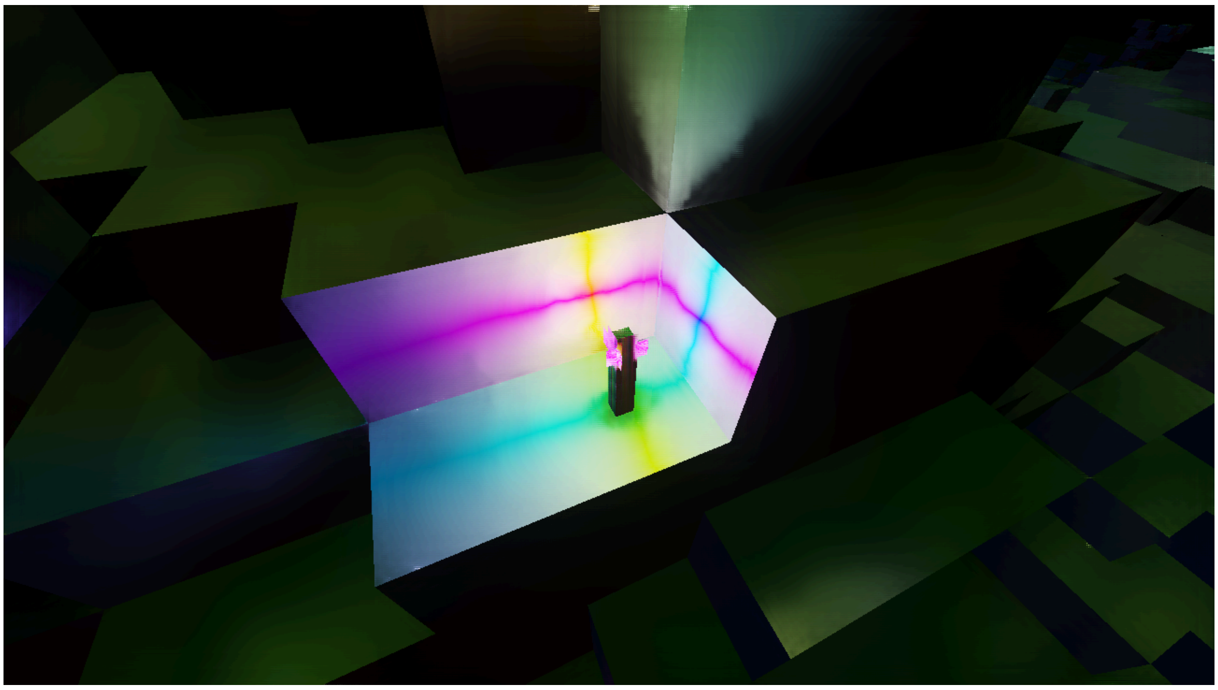
Lighting Reconstruction

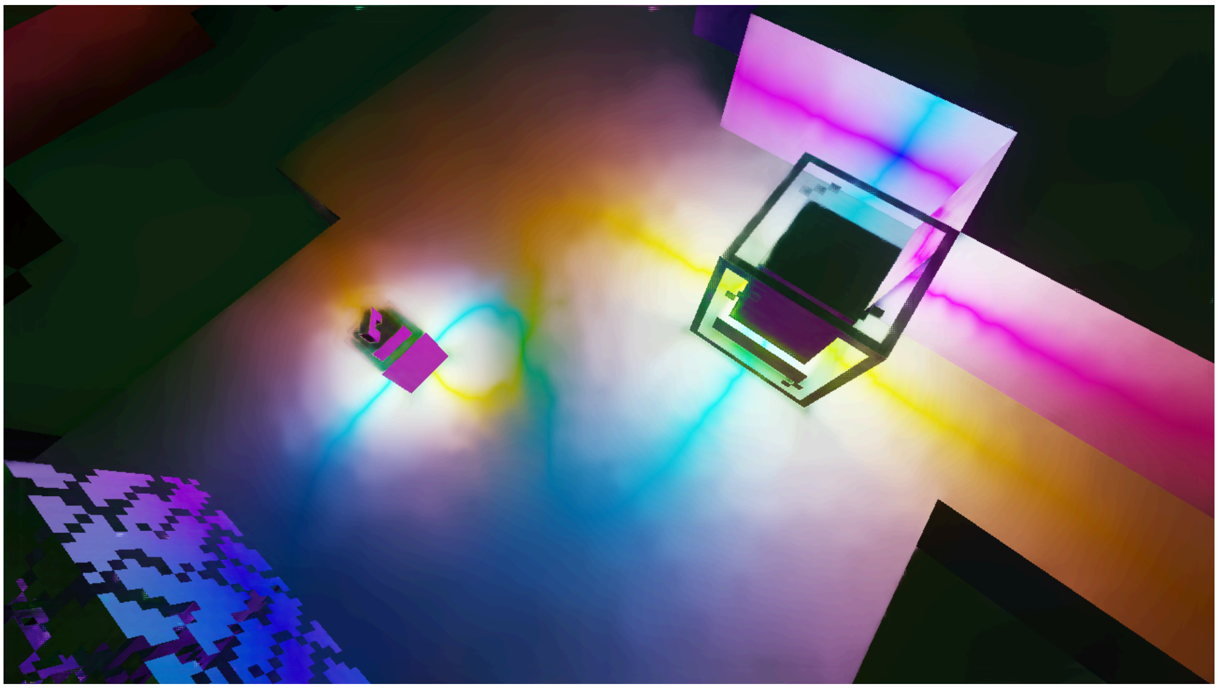
Assuming the denoised lighting $V'_{\text{out}} = \left(\vec{D}, E\right)$ has been obtained, where $\vec{D}$ is the directional lighting vector and E is the ambient lighting component, for a given BRDF model, the reconstructed lighting can be obtained by the following method:

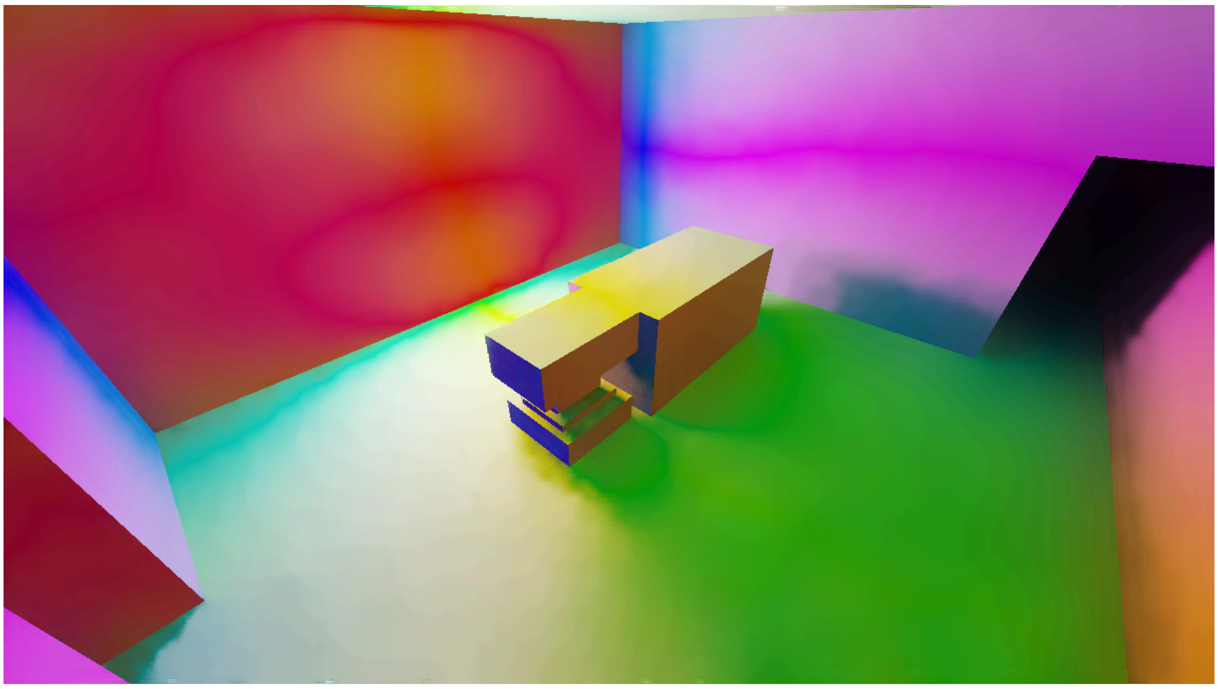
$$I_o = f_{\text{brdf}}\left(\vec{e}, \vec{D}, \vec{N}\right) + \int_{\omega} f_{\text{brdf}}\left(\vec{e}, E\vec{\omega}, \vec{N}\right) \mathrm{d}\omega$$

where I_o is the reconstructed lighting intensity, f_{brdf} is the BRDF function, $\vec{e}$ is the viewing direction, $\vec{N}$ is the normal vector, $\vec{\omega}$ is the incident lighting direction, and $\mathrm{d}\omega$ is the differential solid angle.

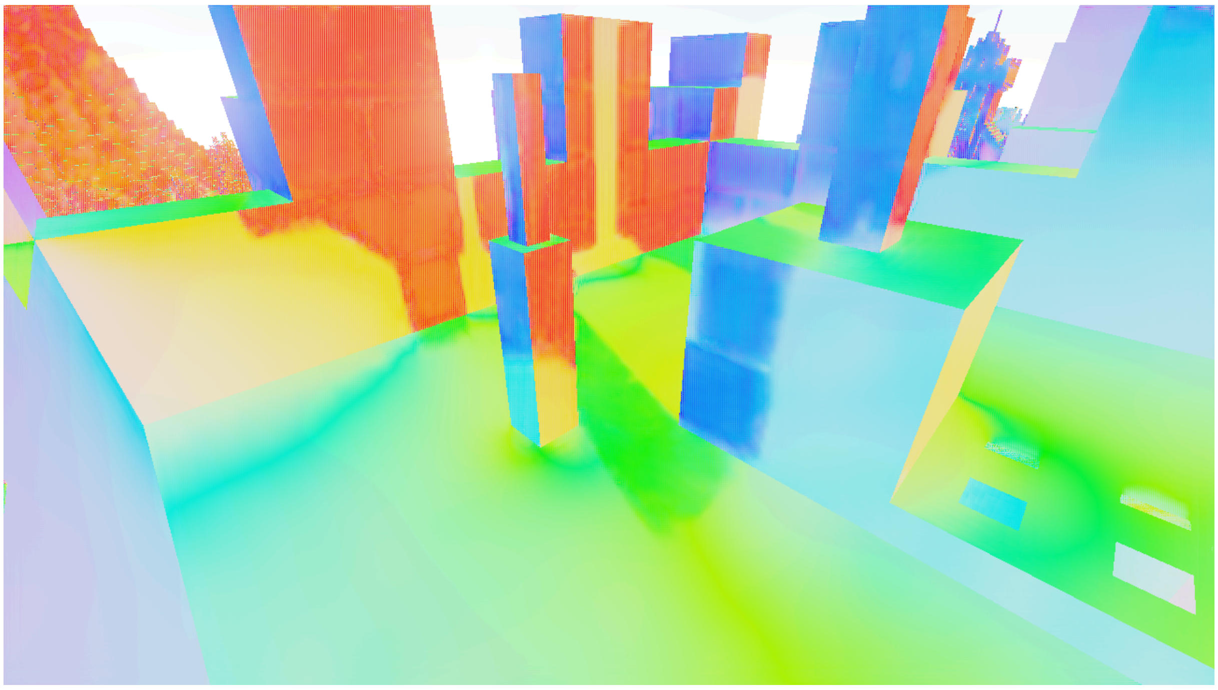
The following images show the directional components

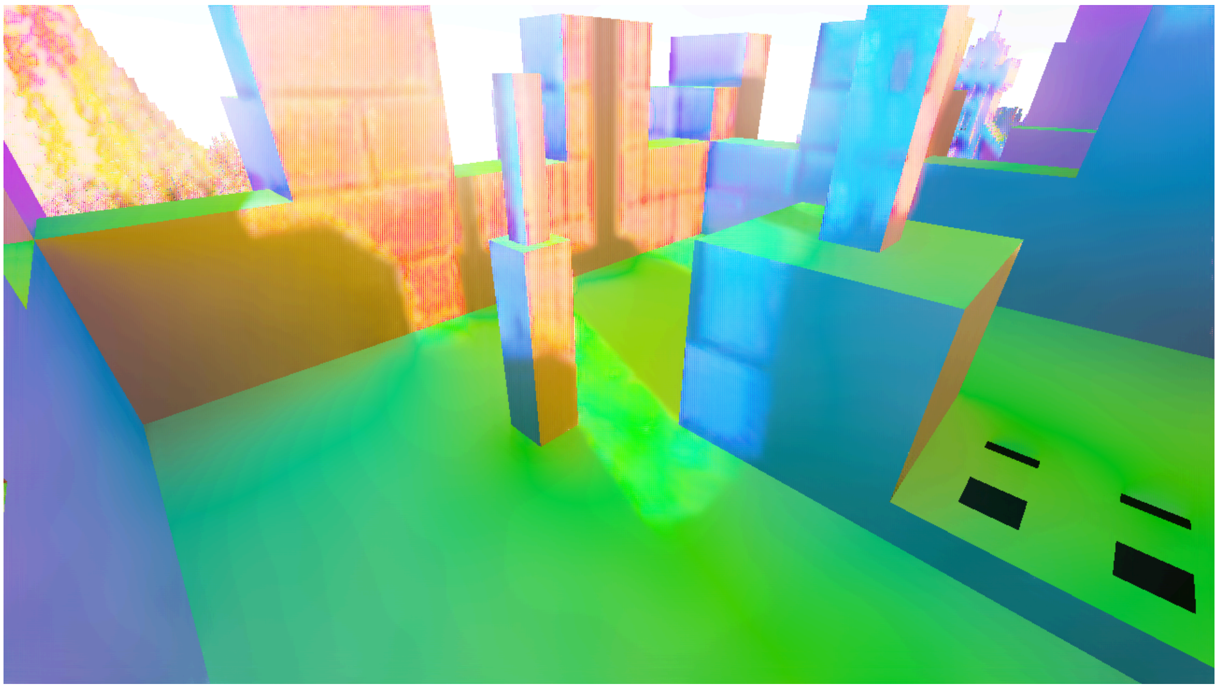


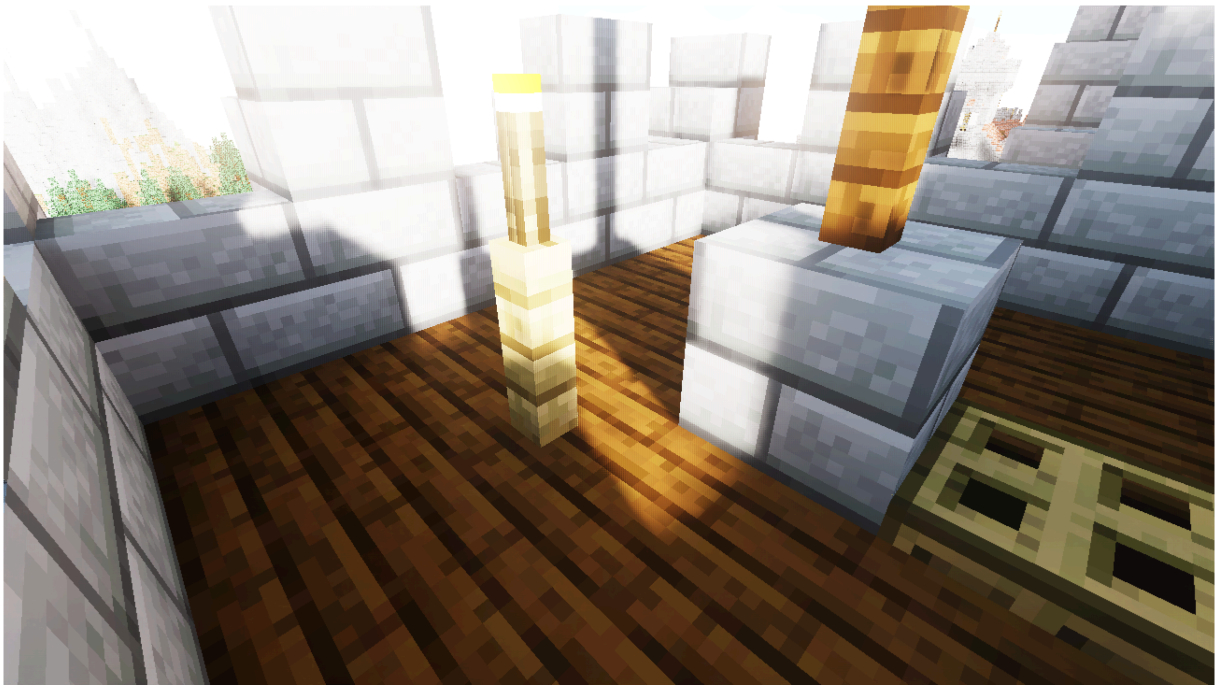




The following images show the directional components and their diffuse lighting reconstruction results







It is clear that the directional components of the denoised lighting roughly point to the direction with significant lighting contribution. Therefore, we can reuse this directional component, input it into the sampler for importance sampling, and obtain more accurate lighting reconstruction results.

It is expected that future path tracers will reuse the lighting directional components from the previous frame for importance sampling to obtain more accurate lighting reconstruction results, rather than simply using the BRDF' s pdf for aimless sampling.

Variance Estimation (Possibly Useful?)

Consider the following optimization problem (temporarily ignoring the $(N+1)$-dimensional component):

$$E\left(\vec{V}^2\right) = \min \frac{1}{n}\sum_{i=1}^{n}|\vec{V}_i|^2$$

$$s.t. \frac{1}{n}\sum_{i=1}^{n}\vec{V}_i = \vec{\mu}$$

$$s.t. \frac{1}{n}\sum_{i=1}^{n}|\vec{V}_i| = I$$

where $\vec{\mu}$ is the expectation of $\vec{V}$, and I is the expectation of $|\vec{V}|$

We need to solve this problem to obtain $E\left(\vec{V}^2\right)$

Considering symmetry, let $\vec{V}_i = \frac{1}{n}\vec{\mu} + \vec{e}_i$, satisfying $\vec{\mu}\cdot\vec{e}_i = 0$ and $\sum \vec{e}_j = 0$, $|\vec{e}_i| = |\vec{e}_j|$

Obviously, the first constraint is satisfied, and the second constraint becomes

$$\frac{1}{n}\sum_{i=1}^{n}|\frac{1}{n}\vec{\mu} + \vec{e}_i| = I$$

That is

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{\left(\frac{1}{n}\vec{\mu} + \vec{e}_i\right)^2} = I$$

That is

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{1}{n^2}\ |\vec{\mu}|^2 + \frac{2}{n}\vec{\mu}\cdot\vec{e}_i + |\vec{e}_i|^2} = I$$

Considering $\vec{\mu}\cdot\vec{e}_i = 0$, then

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{1}{n^2}\ |\vec{\mu}|^2 + |\vec{e}_i|^2} = I$$

Since $|\vec{e}_i| = |\vec{e}_j|$, then

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{1}{n^2}\ |\vec{\mu}|^2 + |\vec{e}_i|^2} = \sqrt{\frac{1}{n^2}\ |\vec{\mu}|^2 + |\vec{e}_i|^2}$$

Thus

$$\vec{V}_i^{\,2} = I^2$$

That is

$$E\left(\vec{V}^2\right) = I^2$$

Considering the original problem, we have

$$E(V^2) = \frac{1}{n}\omega(T(V))^2$$

Then the variance estimation is

$$D(T(V)) = \frac{1}{n}\omega(T(V))^2 - \frac{1}{n^2}\Lambda(T(V))^2$$

Simplified as $D(T(V)) = \sigma^2(V)$

Properties of Variance

$$\sigma^2(\lambda V) = \lambda^2\sigma^2(V)$$