High-Quality Automatic Foreground Extraction Using Consensus Equilibrium

Xiran Wang, Student Member, IEEE, Jason Juang and Stanley H. Chan, Senior Member, IEEE

Abstract—Extracting accurate foreground objects from a scene is a fundamental step in creating virtual reality content. However, majority of the professional softwares today still require human interventions, e.g., providing trimaps or labeling key frames. This is not only time consuming, but is also error prone. In this paper, we present a fully automatic algorithm to extract foreground objects. Our solution is based on a newly developed concept called the Multi-Agent Consensus Equilibrium (MACE), a framework which allows us to integrate multiple sources of expertise to produce an overall superior result. Our MACE framework consists of three agents: (1) A new dual layer closed-form matting agent to estimate the foreground mask using the color image and a background image; (2) A background probability estimator using color difference and object segmentation; (3) A total variation minimization agent to control the smoothness of the foreground masks. We show how these agents are constructed, and how their interactions lead to better performance. The algorithm is evaluated by comparing to several state-of-the-art methods. Extensive experimental study shows that the proposed method has less error compared to the competing methods.

Index Terms—Foreground extraction, alpha matting, video matting, Multi-Agent Consensus Equilibrium, background subtraction

I. INTRODUCTION

A. Motivation and Scope

The proliferation of virtual reality displays and rendering technology over the past decade is rapidly changing the landscape of cinematography [1]–[4]. From the traditional motion pictures to the recent 3D animation, it is safe to argue that the next wave in the film-making industry is immersive experience, e.g., head-mounted virtual reality displays. In order to offer sufficient content to these displays, data has to be acquired in special ways, e.g., using 360-degree volumetric imagers [5]. Typically, such videos are high-definition 8192 × 4320, full frame rate at 60 fps, and are captured using as many as 100 cameras simultaneously. This is an enormous amount of data: A five-minute video sequence using the above configuration is already equal to more than one million images. Efficient image processing of these images is therefore critical.

The subject of this paper is the alpha matting / background subtraction problem arising from the virtual reality content creation process. The goal of alpha matting / background subtraction is to segment the foreground object from an image so that the virtual background can be substituted. However, compared to the classical alpha matting problem [6]–[10], and the classical background subtraction problem [11]–[13], our work lies in the middle ground. Classical alpha matting requires human to provide trimaps, hence it is not fully automatic. Background subtraction is fully automatic, but it produces significantly lower quality results than alpha matting. Our proposed solution achieves comparable performance with alpha matting while maintaining the level of autonomy as background subtraction. Figure 1 and Table I outline the key differences between this paper and the previous work in the literature. The major assumption we make is that a static background image, called the plate image, is available. This plate image can be easily obtained in a filming setting by using the first few frames before an object enters the scene.

B. Challenges of the Plate Images

Readers at this point may argue that the plate assumption is strong: If the plate is available, it seems that a standard frame difference with morphographic operations (e.g., erosion / dilation) would be enough to provide a trimap, and thus a sufficiently powerful alpha matting algorithm would work. However, except for synthetic videos, plate images are never perfect. Below are three typical problems:

1. Plate images are not perfect: The plate image may not be perfect, e.g., due to occlusion or motion.
2. Plate images are rare: It is typically difficult to capture a plate image, especially for complex scenes.
3. Plate images are not always available: In some cases, a plate image may not be available due to technical issues or copyright concerns.
Fig. 2. Three common issues of automatic foreground extraction. Case I: Vibrating background. Notice the small vibration of the leaves in the background. Case II. Similar foreground / background color. Notice the missing parts of the body of the man, and the excessive large uncertainty region of the trimap. Case III. Auto-exposure. Notice the false alarm in the background of the frame difference map. We compare our method with DCNN [10], a semi-supervised alpha matting method using the generated trimaps. The video data of Case III is from [14].

- **Background vibration.** While we assume that the plate does not contain large moving objects, small vibration of the background generally exists. Figure 2 Case I shows an example where the background tree vibrates.
- **Color similarity.** When foreground color is very similar to the background color, the trimap generated will have false alarms and misses. Figure 2 Case II shows an example where the cloth of the man has a similar color to the wall.
- **Auto-exposure.** If auto-exposure is used, the background intensity will change over time. Figure 2 Case III shows an example where the background cabinet becomes dimmer when the man leaves the room.

As shown in the examples, error in frame difference can be easily translated to false alarms and misses in the trimap. While we can increase the uncertainty region of the trimap to rely more on the color constancy model of the alpha matting, in general the alpha matting performs worse when the uncertainty region grows. We have also tested more advanced background estimation algorithms, e.g., [15] in OpenCV. However, the results are similar or sometimes even worse.

C. Related Work

We briefly describe the available methods in the literature.

- **Alpha Matting.** The classical alpha matting formulates the problem as minimizing an energy function associated with the foreground and background color, for example, Poisson matting [17], closed-form matting [16], shared matting [18], Bayesian matting [19], and robust matting [20]. More recently, a number of deep neural network based approaches are proposed, e.g., [21] and [10]. One big drawback of the classical alpha matting algorithms is that they require error-free trimaps. Figure 3 illustrates the performance of various alpha matting algorithms applied to a multi-object-moving scene. In this example, we use frame difference to construct a trimap and then apply various alpha matting algorithms to estimate the alpha matte. Although the performance is reasonable, we observe that small error in the trimap causes major issues in the resulting alpha matte.

- **Video Matting.** Video matting algorithms extend alpha matting by considering the temporal dimension [22]–[24]. The temporal consistency is handled by introducing constraints or regularization functions. Trimaps are still needed for these algorithms, which could become intractable when the video is long. Alternative methods such as [25]–[27] identify key frames of the video and propagate the user labeled alpha mattes to adjacent frames. However, the propagation is often error-prone especially in the case of occlusion.

- **Trimap Generation.** In the absence of trimaps, there are multi-stage methods to first create the trimap and then perform alpha matting [28]. However, these methods still require initial segmentations using, e.g., GrabCut [29]. Other methods [30], [31] require additional sensor data, e.g., depth measurements, which are not always available.

- **Background Subtraction.** Background subtraction methods range from the simple frame difference method to the more sophisticated mixture models [11]. These methods also include classical video segmentation methods, e.g., using saliency [12], [13]. Most background subtraction methods are used to track objects instead of extracting the alpha mattes. They are fully-automated and are real time, but the foreground masks generated are usually of low quality.

D. Contributions

This paper contributes to the alpha matting and background subtraction literature in two ways.

- **Automatic Alpha Matting.** We provide a fully automatic solution to the alpha matting problem assuming that a plate image is available. To the best of our knowledge, this is the first of the kind in the literature.

- **Multi-Agent Consensus Equilibrium.** Our solution leverages a newly developed framework called the Multi-agent consensus equilibrium (MACE) [32] which is a generalization of the Plug-and-Play ADMM [32]–[36]. We present three customized agents for the alpha matting problem. We
demonstrate, for the first time, how MACE can be used to
tackle non-traditional image restoration problems.

In order to explain all the essential concepts, we organize
the paper in a way that the general framework and the specific
components are addressed in two different sections. In Section
2 we describe the MACE framework. We will discuss its
derivation, the intuition, and the algorithm. Then in Section
3 we go deep into the details of each component of the
MACE framework that are specific to foreground extraction.
Experimental results are presented in Section 4.

II. MULTI-AGENT CONSENSUS EQUILIBRIUM

Our proposed method is based on the Multi-Agent Consen-
sus Equilibrium (MACE), recently developed by Buzzard et
al. [32]. In this section, we describe the key components of
MACE and briefly discuss why it works.

A. ADMM

The starting point of MACE is the alternating direction
method of multipliers (ADMM) algorithm [27]. The ADMM
algorithm aims at solving a constrained minimization:

$$\min_{x_1, x_2} f_1(x_1) + f_2(x_2), \quad \text{subject to} \quad x_1 = x_2, \quad (1)$$

where $x_i \in \mathbb{R}^n$, and $f_i : \mathbb{R}^n \to \mathbb{R}$ are mappings, typically
a forward model describing the image formation process and
a prior distribution of the latent image. ADMM solves the
problem by solving a sequence of subproblems as follows:

$$x_1^{(k+1)} = \arg\min_{v \in \mathbb{R}^n} f_1(v) + \frac{\rho}{2} \|v - (x_2^{(k)} - u^{(k)})\|^2, \quad (2a)$$

$$x_2^{(k+1)} = \arg\min_{v \in \mathbb{R}^n} f_2(v) + \frac{\rho}{2} \|v - (x_1^{(k+1)} + u^{(k)})\|^2, \quad (2b)$$

$$u^{(k+1)} = u^{(k)} + (x_1^{(k+1)} - x_2^{(k+1)}). \quad (2c)$$

In the last equation (2c), the vector $u^{(k)} \in \mathbb{R}^n$ is the
Lagrange multiplier associated with the constraint. Under mild
conditions, e.g., when $f_1$ and $f_2$ are convex, close, and proper,
global convergence of the algorithm can be proved [33].
Recent studies show that ADMM converges even for some
non-convex functions [34].

When $f_1$ and $f_2$ are convex, the minimizations in (2a) and
(2b) are known as the proximal maps of $f_1$ and $f_2$, respectively
[38]. If we define the proximal maps as

$$F_i(z) = \arg\min_{v \in \mathbb{R}^n} f_i(v) + \frac{\rho}{2} \|v - z\|^2, \quad (3)$$

then it is not difficult to see that at the optimal point, (2a) and
(2b) become

$$F_1(x^* - u^*) = x^*, \quad (4a)$$

$$F_2(x^* + u^*) = x^*, \quad (4b)$$

where $(x^*, u^*)$ are the solutions to the original constrained
optimization in (1). (4a) and (4b) shows that the solution
$(x^*, u^*)$ can now be considered as a fixed point of the system
of equations.

Rewriting (2a) (2c) in terms of (4a) and (4b) allows us to
consider agents $F_i$ that are not necessarily proximal maps, i.e.,
$f_i$ is not convex or $F_i$ may not be expressible as optimizations.
One example is to use an off-the-shelf image denoiser for $F_i$,
e.g., BM3D, non-local means, or neural network denoisers.
Such algorithm is known as the Plug-and-Play ADMM [33–
[35] (and variations therefrom [32, 36]).

B. MACE and Intuition

MACE generalizes the above ADMM formulation. Instead of
minimizing a sum of two functions, MACE minimizes a sum of
$N$ functions $f_1, \ldots, f_N$:

$$\min_{x_1, \ldots, x_N} \sum_{i=1}^N f_i(x_i), \quad x_1 = \ldots = x_N. \quad (5)$$

In this case, the equations in (4a) (4b) are generalized to

$$F_i(x^* + u^*_i) = x^*, \quad \text{for} \ i = 1, \ldots, N, \quad (6)$$

$$\sum_{i=1}^N u^*_i = 0.$$
What does $G$ buy us? Intuitively, $G$ suggests that in a system containing $N$ agents, each agent will create a tension $u_i^* \in \mathbb{R}^n$. For example, if $F_1$ is an inversion step whereas $F_2$ is a denoising step, then $F_1$ will not agree with $F_2$ because $F_1$ tends to recover details but $F_2$ tends to smooth out details. The agents $F_1, \ldots, F_N$ will reach an equilibrium state where the sum of the tension is zero. This explains the name “consensus equilibrium”, as the algorithm is seeking a consensus among all the agents.

How does the equilibrium solution look like? The following theorem, shown in [32], provides a way to connect the equilibrium condition to a fixed point of an iterative algorithm.

**Theorem 1 (MACE solution [32]).** Let $u^* \overset{\text{def}}{=} [u_1^*; \ldots; u_N^*]$. The consensus equilibrium $(\mathbf{x}^*, \mathbf{u}^*)$ is a solution to the MACE equation (6) if and only if the points $v_i^* \overset{\text{def}}{=} \mathbf{x}^* + u_i^*$ satisfy

$$
\frac{1}{N} \sum_{i=1}^{N} v_i^* = \mathbf{x}^* \tag{7}
$$

$$(2G - I)(2F - I)\mathbf{u}^* = \mathbf{u}^*, \tag{8}
$$

where $v_i^* \overset{\text{def}}{=} [v_1^*; \ldots; v_N^*] \in \mathbb{R}^{nN}$, and $F, G : \mathbb{R}^{nN} \to \mathbb{R}^{nN}$ are mappings defined as

$$
F(z) = \begin{bmatrix} F_1(z_1) \\ \vdots \\ F_N(z_N) \end{bmatrix}, \quad \text{and} \quad G(z) = \begin{bmatrix} \langle z \rangle \\ \vdots \\ \langle z \rangle \end{bmatrix}, \tag{9}
$$

where $\langle z \rangle \overset{\text{def}}{=} \frac{1}{N} \sum_{i=1}^{N} z_i$ is the average of $z$.

**Algorithm 1 MACE Algorithm**

1: Initialize $v^t = [v_1^t; \ldots; v_N^t]$.
2: for $t = 1, \ldots, T$ do
3:  \% Perform agent updates, $(2F - I)(v^t)$
4:  \[ \begin{bmatrix} z_1^t \\ \vdots \\ z_N^t \end{bmatrix} = \begin{bmatrix} 2F_1(v_1^t) - v_1^t \\ \vdots \\ 2F_N(v_N^t) - v_N^t \end{bmatrix} \tag{10} \]
5:  \% Perform the data aggregation $(2G - I)(z^t)$
6:  \[ \begin{bmatrix} v_1^{t+1} \\ \vdots \\ v_N^{t+1} \end{bmatrix} = \begin{bmatrix} 2(z_1^t) - z_1^t \\ \vdots \\ 2(z_N^t) - z_N^t \end{bmatrix} \tag{11} \]
7: end for
9: Output $(v^T)$.

The algorithm of the MACE is illustrated in Algorithm 1. According to (8), $\mathbf{u}^*$ is a fixed point of the set of equilibrium equations. Finding the fixed point can be done by iteratively updating $\mathbf{v}^{(t)}$ through the procedure

$$
\mathbf{v}^{(t+1)} = (2G - I)(2F - I)\mathbf{v}^{(t)} \tag{12}
$$

Therefore, the algorithm steps are no more complicated than updating the individual agents $(2F - I)$ in parallel, and then aggregating the results through $(2G - I)$.

The convergence of MACE is guaranteed when $T$ is non-expansive [32] summarized in the proposition below.

**Proposition 1.** Let $F$ and $G$ be defined as (9), and let $T \overset{\text{def}}{=} (2G - I)(2F - I)$. Then the following results hold:

(i) $F$ is firmly non-expansive if all $F_i$’s are firmly non-expansive.
(ii) $G$ is firmly non-expansive.
(iii) $T$ is non-expansive if $F$ and $G$ are firmly non-expansive.

Proof. See Appendix. \(\square\)

### III. Designing MACE Agents

After describing the MACE framework, in this section we discuss how each agent is designed for our problem.

**A. Agent 1: Dual-Layer Closed-Form Matting**

The first agent we use in MACE is a modified version of the color line model

$$
J(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m} \left( \sum_{j\in w_i} \left( \alpha_i - \sum_{c} a^c_i I^c_i - b_j \right)^2 + \epsilon \sum_{c} (a^c_i)^2 \right), \tag{14}
$$

Here, $(a^c, a^g, a^b, b)$ are the linear combination coefficients of the color line model $\alpha_i \approx \sum_{c\in\{r,g,b\}} a^c I^c_i + b$, and $\alpha_i$ is the alpha matte value of the $i$th pixel [16]. The weight $w_j$ is a $3 \times 3$ window of pixel $j$. With some algebra, we can show that the marginalized energy function $J(\alpha, \mathbf{a}, \mathbf{b})$ is equivalent to

$$
J(\alpha, \mathbf{a}, \mathbf{b}) = \min_{\alpha, \mathbf{a}, \mathbf{b}} J(\alpha, \mathbf{a}, \mathbf{b}) = \alpha^T \mathbf{L} \alpha, \tag{15}
$$

where $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the so-called matting Laplacian matrix. When trimap is given, we can regularize $J(\alpha)$ by minimizing the overall energy function:

$$
\tilde{\alpha} = \arg\min_{\alpha} \alpha^T \mathbf{L} \alpha + \lambda(\alpha - z)^T \mathbf{D}(\alpha - z), \tag{16}
$$

where $\mathbf{D}$ is the diagonal matrix with diagonal entry $D_{ii} = \sum_{j\in w_i} (a^c_i)^2$. Then we can obtain $\tilde{\alpha}$ through $\mathbf{D}^{-1} \mathbf{L} \alpha$, and $\mathbf{D}^{-1} \mathbf{D}(\alpha - z)$.

**Review of Closed-Form Matting** To understand the meaning of (13), we recall that the classical closed-form matting is an algorithm that tries to solve

$$
J(\alpha, \mathbf{a}, \mathbf{b}) = \sum_{i=1}^{m} \left( \sum_{j\in w_i} \left( \alpha_i - \sum_{c} a^c_i I^c_i - b_j \right)^2 + \epsilon \sum_{c} (a^c_i)^2 \right), \tag{14}
$$

where $\mathbf{L}$ and $\mathbf{D}$ are matrices, and will be explained below. The constant $\lambda_1$ is a parameter.
where \( D \) is a binary diagonal matrix with entries one for pixels that are labeled in the trimap, and zero otherwise. The vector \( z \in \mathbb{R}^n \) contains specified alpha values given by the trimap. Thus, for large \( \lambda \), the minimization in (16) will force the solution to satisfy the constraints given by the trimap.

**Dual-Layer Matting Laplacian \( \tilde{L} \).** In the presence of the plate image, we have two pieces of complementary information: \( I \in \mathbb{R}^{n \times 3} \) the color image containing the foreground object, and \( P \in \mathbb{R}^{n \times 3} \) the plate image. Correspondingly, we have alpha matte \( \alpha_i^f \) for \( I \), and the alpha matte \( \alpha_i^p \) for \( P \). When \( P \) is given, we can redefine the color line model as

\[
[\alpha_i^f] \approx \sum_{c \in \{r,g,b\}} a_c^f [I_i^c] + b.
\]

In other words, we ask the coefficients \((a_r^f, a_g^f, a_b^f, b)\) to fit simultaneously the actual image \( I \) and the plate image \( P \). When (17) is assumed, the energy function \( J(\alpha, a, b) \) becomes

\[
\tilde{J}(\alpha^f, \alpha^p, a, b) = \sum_{j \in \mathcal{E}} \left( \sum_{i \in w_j} \left( \alpha_i^f - \frac{1}{|w_j|} \sum_c a_c^f I_i^c \right)^2 + \eta \sum_{i \in w_j} \left( \alpha_i^p - \frac{1}{|w_j|} \sum_c a_c^p P_i^c \right)^2 + \epsilon \sum_c \left( a_c^f \right)^2 \right),
\]

where we added a constant \( \eta \) to regulate the relative emphasis between \( I \) and \( P \).

**Theorem 2.** The marginal energy function

\[
\tilde{J}(\alpha) \overset{\text{def}}{=} \min_{a,b} \tilde{J}(\alpha, 0, a, b)
\]

can be equivalently expressed as \( \tilde{J}(\alpha) = \alpha^T \tilde{L} \alpha \), where \( \tilde{L} \in \mathbb{R}^{n \times n} \) is the modified matting Laplacian, with the \( (i,j) \)th element

\[
\tilde{L}_{i,j} = \sum_{k \mid (i,j) \in w_k} \left\{ \delta_{ij} - \frac{1}{|w_k|} \left( 1 + (I_i - \mu_k)^T \Sigma_k - n(1 + \eta)\mu_k\mu_k^T \right)^{-1} (I_j - \mu_k) \right\}.
\]

Here, \( \delta_{ij} \) is the Kronecker delta, \( I_i \in \mathbb{R}^3 \) is the color vector at the \( i \)th pixel. The vector \( \mu_k \in \mathbb{R}^3 \) is defined as

\[
\mu_k = \frac{1}{|w_k|} \sum_{j \in w_k} (I_j + P_j),
\]

and the matrix \( \Sigma_k \in \mathbb{R}^{3 \times 3} \) is

\[
\Sigma_k = \frac{1}{2} \left\{ \frac{1}{|w_k|} \sum_{j \in w_k} (I_j - \mu_k)(I_j - \mu_k)^T + \frac{1}{|w_k|} \sum_{j \in w_k} (P_j - \mu_k)(P_j - \mu_k)^T \right\}.
\]

**Proof.** See Appendix.

Because of the plate term in (18), the modified matting Laplacian \( \tilde{L} \) is positive definite. See Appendix for proof. The original \( L \) in (15) is only positive semi-definite.

**Dual-Layer Regularization \( \tilde{D} \).** The diagonal regularization matrix \( \tilde{D} \) in (13) is reminiscent to the binary matrix \( D \) in (16), but \( \tilde{D} \) is defined through a sigmoid function applied to the input \( z \). To be more precise, we define \( \tilde{D} \overset{\text{def}}{=} \text{diag}(\tilde{d}_i) \), where

\[
\tilde{d}_i = \text{diag} \left\{ \frac{1}{1 + \exp(-\kappa(z_i - \theta))} \right\} \in \mathbb{R}^{n \times n},
\]

and \( z_i \) is the \( i \)-th element of the vector \( z \in \mathbb{R}^n \), which is the argument of \( F_1 \). The scalar constant \( \kappa > 0 \) is a user defined parameter specifying the stiffness of the sigmoid function, and \( 0 < \theta < 1 \) is another user defined parameter specifying the center of the transient. Typical values of \( (\kappa, \theta) \) for our MACE framework are \( \kappa = 30 \) and \( \theta = 0.8 \).

A closer inspection of \( D \) and \( \tilde{D} \) reveals that \( D \) is performing a hard-threshold whereas \( \tilde{D} \) is performing a soft-threshold. In fact, the matrix \( D \overset{\text{def}}{=} \text{diag}(d_i) \) has diagonal entries

\[
d_i = \begin{cases} 0, & \theta_1 < z_i < \theta_2, \\ 1, & \text{otherwise.} \end{cases}
\]

for two cutoff values \( \theta_1 \) and \( \theta_2 \). This hard-threshold is equivalent to the soft-threshold in (23) when \( \kappa \rightarrow \infty \).

There are a few reasons why (23) is preferred over (24), especially when we have the plate image. First, the soft-threshold in (23) tolerates more error present in \( z \), because the values of \( \tilde{D} \) represent the probability of having foreground pixels. Second, the one-sided threshold in (23) ensures that the background portion of the image is handled by the plate image rather than the input \( z \). This is usually beneficial when the plate is reasonably accurate.

**B. Agent 2: Background Estimator**

Our second agent is a background estimator, defined as

\[
F_2(z) = \text{argmin}_\alpha \| \alpha - r_0 \|^2 + \lambda_2 \| \alpha - z \|^2 + \gamma \alpha^T (1 - \alpha).
\]

The reason of introducing \( F_2 \) is that in \( F_1 \), the matrix \( \tilde{D} \) is determined by the current estimate \( z \). While \( \tilde{D} \) handles part of the error in \( z \), large missing pixels and false alarms can still cause problems especially in the interior regions. The goal of \( F_2 \) is to complement \( F_1 \) for these interior regions.

**Initial Background Estimate \( r_0 \).** Let us take a look at (25).

The first two terms are quadratic. The interpretation is that given some fixed initial estimate \( r_0 \) and the current input \( z \), \( F_2(z) \) returns a linearly combined estimate between \( r_0 \) and \( z \). The initial estimate \( r_0 \) consists of two parts:

\[
r_0 = r_c \odot r_e,
\]

where \( \odot \) means elementwise multiplication. The first term \( r_c \) is the color term, measuring the similarity between foreground and background colors. The second term \( r_e \) is the edge term, measuring the likelihood of foreground edges relative background edges. In the followings we will discuss these two terms one by one.

**Defining the Color Term \( r_c \).** We define \( r_c \) by measuring the distance \( \| I_j - P_j \|^2 = \sum_{c \in \{r,g,b\}} (I_j^c - P_j^c)^2 \) between a color
pixel $I_j \in \mathbb{R}^3$ and a plate pixel $P_j \in \mathbb{R}^3$. Ideally, we would like $r_c$ to be small when $\|I_j - P_j\|^2$ is large.

In order to improve the robustness of $\|I_j - P_j\|^2$ against noise and illumination fluctuation, we modify $\|I_j - P_j\|^2$ by using the bilateral weighted average over a small neighborhood:

$$
\Delta_i = \sum_{j \in \Omega_i} w_{ij} \|I_j - P_j\|^2,
$$

(27)

where $\Omega_i$ specifies a small window around the pixel $i$. The bilateral weight $w_{ij}$ is defined as

$$
w_{ij} = \frac{\tilde{w}_{ij}}{\sum_j \tilde{w}_{ij}},
$$

(28)

where

$$
\tilde{w}_{ij} = \exp \left\{ - \frac{\|x_i - x_j\|^2}{2\tilde{h}_s^2} \right\} \exp \left\{ - \frac{\|I_i - I_j\|^2}{2\tilde{h}_r^2} \right\}.
$$

Here, $x_i$ denotes the spatial coordinate of pixel $i$, $I_i \in \mathbb{R}^3$ denotes the $i$th color pixel of the color image $I$, and $(\tilde{h}_s, \tilde{h}_r)$ are the parameters controlling the bilateral weight strength. The typical values of $\tilde{h}_s$ and $\tilde{h}_r$ are both 5.

We now need a mapping which maps the distance $\Delta \overset{\text{def}}{=} [\Delta_1, \ldots, \Delta_n]^T$ to a vector of numbers $r_c$ in $[0, 1]^n$ so that the term $\|\alpha - r_0\|^2$ makes sense. To this end, we choose a simple Gaussian function:

$$
r_c = 1 - \exp \left\{ - \frac{\Delta_2}{2\sigma_\Delta^2} \right\},
$$

(30)

where $\sigma_\Delta$ is a user tunable parameter. We tested other possible mappings such as the sigmoid function and the cumulative distribution function of a Gaussian. However, we do not see significant difference compared to (30). The typical value for $\sigma_\Delta$ is 10.

### Defining the Edge Term $r_e$

The color term $r_c$ is able to capture most of the difference between the image and the plate. However, it also generates false alarms if there is illumination change. For example, shadow due to the foreground object is often falsely labeled as foreground. See the shadow near the foot in Figure 4.

In order to reduce the false alarm due to minor illumination change, we first create a "super-pixel" mask by grouping similar colors. Our super-pixels are generated by applying a standard flood-fill algorithm [39] to the image $I$. This gives us a partition of the image $I$ as

$$
I \rightarrow \{I^{S_1}, I^{S_2}, \ldots, I^{S_m}\},
$$

(31)

where $S_1, \ldots, S_m$ are the $m$ super-pixel index sets. The plate image is partition using the same super-pixel indices, i.e., $P \rightarrow \{P^{S_1}, P^{S_2}, \ldots, P^{S_m}\}$.

While we are generating the super-pixels, we also compute the gradients of $I$ and $P$ for every pixel $i = 1, \ldots, n$. Specifically, we define $\nabla I_i = [\nabla_x I_i, \nabla_y I_i]^T$ and $\nabla P_i = [\nabla_x P_i, \nabla_y P_i]^T$, where $\nabla_x I_i \in \mathbb{R}^3$ and $\nabla_y I_i \in \mathbb{R}^3$ are the two-tap horizontal (and vertical) finite difference at the $i$-th pixel. To measure how far $I_i$ is from $P_i$, we compute

$$
\theta_i = \|\nabla I_i - \nabla P_i\|_2.
$$

(32)

Thus, $\theta_i$ is small for background regions because $I_i \approx P_i$, but is large when there is a foreground pixel in $I_i$. If we set a threshold operation after $\theta_i$, i.e., set $\theta_i = 1$ if $\theta_i > \tau_\theta$ for some threshold $\tau_\theta$, then shadows can be removed as their gradients are weak.

Now that we have computed $\theta_i$, we still need to map it back to a quantity similar to the alpha matte. To this end, we compute a normalization term

$$
A_i = \max (\|\nabla I_i\|_2, \|\nabla P_i\|_2),
$$

(33)

and normalize $1\{\theta_i > \tau_\theta\}$ by

$$
(r_e)_i \overset{\text{def}}{=} \frac{\sum_{j \in S_i} 1\{A_i > \tau_A\} 1\{\theta_i > \tau_\theta\}}{\sum_{j \in S_i} 1\{A_i > \tau_A\}},
$$

(34)

where $1\{\cdot\}$ denotes the indicator function, and $\tau_A$ and $\tau_\theta$ are thresholds. In essence, (34) says in the $i$-th super-pixel $S_i$, we count the number of edges $1\{\theta_i > \tau_\theta\}$ that have strong difference between $I_i$ and $P_i$. However, we do not want to count every pixel but only pixels that already contains strong edges, either in $I$ or $P$. Thus, we take the weighted average using $1\{A_i > \tau_A\}$ as the weight. This defines $r_e$, as the weighted average $(r_e)_i$ is shared among all pixels in the super-pixel $S_i$. Figure 5 shows a pictorial illustration.
Why is \( r_e \) helpful? If we look at \( r_e \) and \( r_e \) in Figure 6 we see that the foreground pixels of \( r_e \) and \( r_e \) coincide but background pixels roughly cancel each other. The reason is that while \( r_e \) creates weak holes in the foreground, \( r_e \) fills the gap by ensuring the foreground is marked.

**Regularization** \( \alpha^T(1 - \alpha) \). The last term \( \alpha^T(1 - \alpha) \) in (23) is a regularization to force the solution to either 0 or 1. The effect of this term can be seen from the fact that \( \alpha^T(1 - \alpha) \) is a symmetric concave quadratic function with a value zero for \( \alpha = 1 \) or \( \alpha = 0 \). Therefore, it introduces penalty for solutions that are away from 0 or 1. For \( \gamma \leq 1 \), one can show that the Hessian matrix of the function \( f_2(\alpha) = \|\alpha - r_0\|^2 + \gamma \alpha^T(1 - \alpha) \) is positive semidefinite. Thus, \( f_2 \) is strongly convex with parameter \( \gamma \).

### C. Agent 3: Total Variation Denoising

The third agent we use in this paper is the total variation denoising agent. A denoising agent is used in the MACE framework because denoising algorithms rely on the patch matching procedure, and so they tend to under-smooth repeated patterns of false alarm / misses. Neural network denoisers are better candidates but they need to be trained with the specifically distorted alpha mattes. From our experience, we do not see any particular advantage of using CNN-based denoisers. Figure 7 shows some comparison.

### D. Parameters and Runtime

The typical values for parameters of the proposed method are presented in Table III-D. \( \lambda_1 \) and \( \lambda_2 \) are rarely changed, while \( \lambda_3 \) determines the denoising strength of Agent 3. \( \gamma \) has a default value of 0.05. Increasing \( \gamma \) causes more binary results with clearer boundaries. \( \tau_A \) and \( \tau_B \) determine the edge term \( r_e \) in Agent 2 and are fixed. \( \sigma_B \) determines the color term \( r_e \) in Agent 2. Large \( \sigma_B \) produces less false negative but more false positive. Overall, the performance is reasonably stable to these parameters.

| Parameter | \( \lambda_1 \) | \( \lambda_2 \) | \( \lambda_3 \) | \( \gamma \) | \( \tau_A \) | \( \tau_B \) | \( \sigma_B \) |
|-----------|----------------|----------------|----------------|---------|------------|------------|----------|
| Value     | 0.01           | 2              | 0.05           | 0.01    | 0.02       | 10         |

In terms of runtime, the most time-consuming part is Agent 1 because we need to solve a large-scale sparse least squares problem. Its runtime is determined by the number of foreground pixels. Table III shows the runtime of the sequences we tested. In generating these results, we used an un-optimized MATLAB code on a Intel i7-4770k. The typical runtime is about 1-3 minutes per frame, which can further be improved by porting the algorithms to C++ or GPUs. From our experience working with professional artists, even with professional film production software, e.g., NUKE, it takes 15 minutes to label a ground truth label using the plate and temporal cues. Therefore, the runtime benefit offered by our algorithm is substantial.

### IV. Experimental Results

#### A. Dataset

To evaluate the proposed method, we create a Purdue dataset containing 8 video sequences using the HypeVR Inc. 360 degree camera as shown in Figure 8. The original image resolution is 8192 x 4320 at a frame rate of 48fps, and these images are then downsamples and cropped to speed up the matting process. In addition to these videos, we also include 6 videos sequences from a public dataset [14], making a total of 14 video sequences. Snapshots of the sequences are shown in Figure 7. All video sequences are captured without camera motion. Plate images are available, either during the first or the last few frames of the video. To enable objective evaluation, for each video sequence we randomly select 10 frames and manually generate the ground truths. Thus totally there are 140 frames with ground truths.

The characteristics of the dataset is summarized in Table III. The Purdue dataset has various resolution, and the Public
dataset has one resolution 480 × 640. The foreground percentage for the Purdue dataset videos ranges from 1.03% to 55.10%, whereas that public dataset has similar foreground percentage around 10%. The runtime of the algorithm (per frame) is determined by the resolution and the foreground percentage. In terms of content, the Purdue dataset focuses on outdoor scenes whereas the public dataset are only indoor. The shadow column indicates the presence of shadow. Lighting issues include illumination change due to auto-exposure and auto-white-balance. The background vibration only applies to outdoor scenes where the background objects have minor movements, e.g., moving grass or tree branches. The camouflage column indicates the similarity in color between the foreground and background, which is a common problem for most sequences. The green screen column shows which of the sequences have green screens to mimic the common chroma-keying environment.

### TABLE III
DESCRIPTION OF THE VIDEO SEQUENCES USED IN OUR EXPERIMENTS.

| Dataset | Resolution | FGD % | time/Frame (sec) | outdoor | indoor | shadow | lighting issues | Backgnd vibration | camouflage | green screen | ground truth |
|---------|------------|-------|------------------|---------|--------|--------|----------------|-------------------|-------------|-------------|--------------|
| Purdue  | Book       | 632x1012 | 4.03%            | 170.8   | 231    | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Building   | 540x1900 | 19.75%           | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Coach      | 790x1264 | 4.68%            | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Studio     | 480x270  | 55.10%           | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Road       | 675x1175 | 103%             | 232.9   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Tackle     | 501x1676 | 4.80%            | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Gravel     | 790x1536 | 2.53%            | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Office     | 623x1229 | 3.47%            | 396.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
| Public  | Bootstrap  | 480x640  | 13.28%           | 109.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Cespatx    | 480x640  | 10.31%           | 109.1   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | DCam       | 480x640  | 12.23%           | 123.6   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Gen        | 480x640  | 10.23%           | 123.6   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Multipeople| 480x640  | 9.04%            | 99.5    | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |
|         | Shadow     | 480x640  | 11.97%           | 115.2   | 70.8   | ✓      | ✓              | ✓                 | ✓           | ✓           | ✓            |

### TABLE IV
DESCRIPTION OF THE COMPETING METHODS.

| Methods | Automated? | Key idea |
|---------|------------|----------|
| BSVS    | semi-auto  | bilateral space video segmentation |
| Matting | trimap     | Trimap generation + alpha matting |
| Grabcut | full-auto  | iterative graph cuts video segmentation |
| ViBe    | full-auto  | pixel model based background subtraction |
| NLVS    | full-auto  | non-local voting video segmentation |
| Pbas    | full-auto  | non-parametric background subtraction |

### B. Competing methods
We categorize the competing methods into three different categories. The key ideas of the competing methods are summarized in Table [LV]

- **Full-automatic methods**: We consider two background subtraction algorithms Pixel-based adaptive segmenter (Pbas) [15], Visual background extractor (ViBe) [44], one video segmentation method Non-local consensus voting (NLVS) [45] and a modified version of Grabcut [29]. Pbas [15] and ViBe [44] are real-time. However, the generated masks have low quality. NLVS [45] uses optical flow as the main cue and exploits the similarities among super-pixels. Thus it suffers from errors caused by the super-pixel step. The modified version of Grabcut [29] is made to use the plate image and no longer requires user input. It performs reasonably well when background and foreground have different colors, but suffers when the foreground and background colors are similar.

- **Semi-automatic method**: We consider the Bilateral space video segmentation (BSVS) [25], a semi-supervised video segmentation algorithm using the bilateral space. BSVS is semi-supervised as it requires the user to provide ground truth labels for key frames.

- **Alpha matting using trimap**: We consider one of the state-of-the-art alpha matting algorithm using CNN [43]. The trimaps are generated by applying frame difference between the plate and color images, followed by morphological and thresholding operations.
We emphasize that while these competing methods are possibly the closest methods to our proposed method, they are considerably different. For example, BSVS [25] does not require a plate image, but it requires key frames. NLVS [45] also does not require a plate image, but it cannot handle videos with relatively stable objects as it relies on optical flow.

C. Results

We use intersection over union (IoU) as an evaluation metric. The IoU contains two parts, intersection and union:

$$\cap = \sum \min(\hat{x}_i, x_i), \quad \cup = \sum \max(\hat{x}_i, x_i),$$

and $IoU = \frac{\cap}{\cup}$. In this equation, $\hat{x}_i$ is the $i$-pixel of the estimated alpha matte, and $x_i$ is that of the ground truth.

- **Comparison with full-automatic methods:** The results are shown in Figure 10, where we plot the overall IoU for different methods. In this chart, it is clear that the proposed method performs the best. Among the video sequences, the foreground object is stationary or it only has rotational movements so that it is treated as background.

- **Comparison with semi-automatic method:** We compare our method with BSVS [25]. Since BSVS requires ground truth key frames, in this experiment we use 10 key frames for each video sequence. When testing, we feed the video sequence and the key frames to the algorithm. The output is a processed sequence, and the key frames are modified because the algorithm propagates the error.

The results of this experiment show that the proposed algorithm is on average better than BSVS even though BSVS has ground truth key frames. For the Office sequence, BSVS scores a IoU of 0.77 while the proposed method scores 0.95. BSVS performs badly because the background is complex and sometimes it is similar to the foreground color. Even with key frame ground truth, BSVS is still confused. Similar observation can be found for the case of Gravel. BSVS performs better in the Shadow sequence scoring 0.96 in comparison with the proposed method scoring 0.90. This is because the key frame ground truth helps preventing mislabeling the shadows as foreground. On average, the proposed method scores 0.93 which is higher than 0.86 scored by BSVS.

- **Comparison with trimap + alpha-matting methods:** In this experiment we compare with several state-of-the-art alpha matting algorithms. The visual comparison is shown in Figure 3 and the IoU value of DCNN [10] is shown in Figure 10. In order to make this method work, careful tuning during the trimap generation stage is required.

Figure 3 and Figure 10 show that most alpha matting algorithms suffer from false alarms near the boundary, e.g., spectral matting [6], closed-form mating [16], learning-based matting [7] and comprehensive matting [9]. The more recent methods such as K-nearest neighbors matting [8] and DCNN [10] have equal amount of false alarm and miss. Yet, the overall performance is still worse than the proposed method.

D. Ablation test and MACE with various denoisers

Since the proposed framework contains three different agents, we conduct an ablation study to verify the relative importance of the individual agents. To do so, we remove one of the three agents while keeping the other two fixed. The result is shown in Figure 12. Based on the result, the matting agent, not surprisingly, has the most impact on the performance, followed by background estimator and denoiser. The drop in performance is most significant for hard sequences such as Book as it contains moving background, and Road as it contains strong color similarity between foreground and background. The proposed method scores 0.98 for Book while in the absence of Agent 1 and 3 it only scores 0.38 and 0.48 respectively. The performance drop is even more for Road with the proposed method scoring 0.91, 0.06 without Agent 1 and 0.11 without Agent 3.

In this ablation study, we also observe spikes of error for some scenes when background estimator is absent in MACE. This is because, without the $\alpha^T(1 - \alpha)$ term from the background estimator, the result will look grayish instead of close-to-binary. This behavior leads to the error spikes. For example, the performance drops from 0.94 for the proposed method to 0.72 when Agent 2 becomes absent. For green screen scenes like Tackle and Coach, since the initial estimate $r_0$ is already close to the ground truth the performance does not suffer when the matting or denoiser agents are absent.

For Agent 3 the denoiser, we observe that the total variation denoiser leads to the best performance for MACE. In a visual comparison shown in Figure 7 we observe that IRCNN [42] produces more detailed boundaries but fails to remove false alarms near the feet. BM3D [41] removes false alarms better but produces less detailed boundaries. TV on the other hand produces a more balanced result. As shown in Figure 12 BM3D performs similarly as IRCNN scoring 0.85 on average, while TV performs the best with an average score of 0.93.
BM3D and IRCNN score the worst for the Road sequence, as they fail to remove false alarms caused by moving tree branches in the background. This is because the false alarms in the background form a repetitive pattern which could confuse BM3D who denoises by block matching.

V. LIMITATIONS AND DISCUSSION

While the proposed method demonstrates superior performance than the state-of-the-art methods, it also has several limitations.

- **Quality of Plate Image.** The plate assumption may not hold when the background is moving substantially. When this happens, a more complex background model that includes dynamic information is needed. However, if the background is non-stationary, additional designs are needed to handle the local error and temporal consistency.
- **Loss of Fine Details.** In our proposed method, fine details such as hairs are compromised for robustness. Figure 13 illustrates an example. In some videos, the color difference between foreground and background is similar. This creates holes in the initial estimate $r_0$, can be filled by a strong denoiser such as total variation. However, total variation is known to oversmooth fine details. To mitigate this issue, an additional post-processing step using alpha matting could bring back the details around the boundary.
- **Strong Shadows.** Strong shadows are sometimes treated as foreground, as shown in Figure 14. This is caused by the lack of shadow modeling in the problem formulation. The
edge based initial estimate $r_e$ can resolve the shadow issue to some extent, but not when the shadow is very strong. We tested a few off-the-shelf shadow removal algorithms [16]–[18], but generally they do not help because the shadow in our dataset can cast on the foreground object which should not be removed.

![Fig. 13. Limitation 1: Loss of fine details. (a) Color input. (b) Our result. (c) Improving our result by generating a trimap from (b). (d) post-processed result by alpha matting using (b).](image)

![Fig. 14. Limitation 2: Strong shadows. When shadows are strong, they are easily misclassified as foreground.](image)

VI. CONCLUSION

We presented a new foreground extraction algorithm based on the multi-agent consensus equilibrium (MACE) framework. MACE is an information fusion framework which integrates multiple weak experts to produce a strong estimator. Equipped with three customized agents: a dual-layer closed form matting agent, a background estimation agent and a total variation denoising agent, MACE offers substantially better foreground masks than state-of-the-art algorithms. MACE is a fully automatic algorithm, meaning that human interventions are not required. This provides significant advantage over semi-supervised methods which require trimaps or scribbles. In the current form, MACE is able to handle minor variations in the background plate image, illumination changes and weak shadows. Extreme cases can still cause MACE to fail, e.g., background movement or strong shadows. However, these could potentially be overcome by improving the background and shadow models.

VII. ACKNOWLEDGEMENT

The authors thank the anonymous reviewers for valuable feedback. This work is supported, in part, by the National Science Foundation under grants CCF-1763896 and CCF1718007.

VIII. APPENDIX

A. Proof of Theorem 2

Proof. We start by writing (18) in the matrix form

$$\tilde{J}(\alpha^I, \alpha^P, a, b) = \sum_{k \in I} \left[ \frac{H_k}{\sqrt{\eta G_k}} \frac{1}{\sqrt{\eta I}} \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \begin{bmatrix} \alpha^I_k \\ \alpha^P_k \end{bmatrix} \right]^2$$

where

$$H_k = \begin{bmatrix} I^i_k \\ I^g_k \\ I^b_k \end{bmatrix}, \quad G_k = \begin{bmatrix} P_i \\ P_g \\ P_b \end{bmatrix},$$

$$a_k = \begin{bmatrix} a^I_k \\ a^g_k \\ a^b_k \end{bmatrix}, \quad \alpha^I_k = \begin{bmatrix} \alpha^I_k \\ \vdots \\ \alpha^I_k \end{bmatrix}, \quad \alpha^P_k = \begin{bmatrix} \alpha^P_k \\ \vdots \\ \alpha^P_k \end{bmatrix},$$

and $i$ denotes the index of the $i$-th pixel in the neighborhood $w_i$. The difference with the classic closed-form matting [16] is the new terms $G_k, 1$ and $\alpha^P_k$ (i.e., the second row of the quadratic function above.)

Denote

$$B_k \overset{\text{def}}{=} \begin{bmatrix} H_k \\ \sqrt{\eta G_k} \\ \sqrt{\eta I} \end{bmatrix},$$

and use the fact that $\alpha^P = 0$, we can find out the solution of the least-squares optimization:

$$\begin{bmatrix} a_k \\ b_k \end{bmatrix} = (B_k^T B_k)^{-1} B_k^T \begin{bmatrix} \alpha^I_k \\ 0 \\ 0 \end{bmatrix}$$

We now need to simplify the term $B_k^T B_k$. First, observe that

$$B_k^T B_k = \begin{bmatrix} H_k^T H_k + \eta G_k^T G_k + \epsilon I_{3 \times 3} \quad H_k^T 1 + \eta G_k^T 1 \\ (H_k^T 1 + \eta G_k^T 1)^T \quad (H_k 1 + \eta G_k 1) \end{bmatrix}$$

$$= \Sigma_k \begin{bmatrix} \mu_k \\ \mu_k^T \end{bmatrix}$$
where we define the terms \( \Sigma_k \equiv H_k^T H_k + \eta G_k^T G_k + c I \), 
\( \mu_k \equiv H_k^T 1 + \eta G_k^T 1 \) and \( c = n(1 + \eta) \). Then, by applying the block inverse identity, we have
\[
(B_k^TV_kB_k)^{-1} = \begin{bmatrix}
T_k^{-1} & -T_k^{-1}\tilde{\mu}_k \\
-(T_k^{-1}\tilde{\mu}_k)^T & -\tilde{\mu}_k^T + \frac{1}{c} \mu_k T_k^{-1}\mu_k
\end{bmatrix}^{-1} 
\] (39)
where we further define \( T_k = \Sigma_k - \frac{\mu_k T_k^T}{c} \) and \( \tilde{\mu}_k = \frac{\mu_k}{c} \).
Substituting (38) back to \( \tilde{J} \), and using (39), we have
\[
\tilde{J}(\alpha^T) = \sum_k \left| (I_{3x3} - B_k (B_k^T B_k)^{-1} B_k^T) \begin{bmatrix} \alpha^T_0 \\ 0 \\ 0 \end{bmatrix} \right|^2 
= (\alpha^T L_k \alpha_k^T, 
where
\[
L_k = I_{3x3} - \left( H_k T_k^{-1} H_k^T - H_k T_k^{-1} \tilde{\mu}_k 1^T \right) 
- T_k^{-1}(\tilde{\mu}_k^T H_k + \frac{1}{c} T_k^{-1}\tilde{\mu}_k 1^T) \right) 
\] (40)
The \((i, j)\)-th element of \( L_k \) is therefore
\[
L_{k(i, j)} = \delta_{ij} - \left( I_{3x3}^{-1} I_{k(i, j) - I_{k(i, j)}^{-1} T_k^{-1} \tilde{\mu}_k} 
- \tilde{\mu}_k T_k^{-1} I_{k(i, j)} + \frac{1}{c} + \tilde{\mu}_k T_k^{-1} \tilde{\mu}_k \right) 
= \delta_{ij} - \left( \frac{1}{c} + (I_{k(i, j)} - \tilde{\mu}_k)^T T_k^{-1} (I_{k(i, j)} - \tilde{\mu}_k) \right) \] (41)
Adding terms in each \( w_k \), we finally obtain
\[
\tilde{L}_{i,j} = \sum_{k(i, j) \in w_k} \left| \delta_{ij} - \left( \frac{1}{c} + (I_{k(i, j)} - \tilde{\mu}_k)^T T_k^{-1} (I_{k(i, j)} - \tilde{\mu}_k) \right) \right| \]
\[ \blacksquare \]

**C. Proof of Proposition 1**

**Proof.** Let \( \alpha \in \mathbb{R}^{nN} \) and \( \gamma \in \mathbb{R}^{nN} \) be two super-vectors.

(i). If the \( F_i \)'s are non-expansive, then
\[
\| F(\alpha) - F(\gamma) \|^2 + \| \alpha - \gamma - (F(\alpha) - F(\gamma)) \|^2 
= \sum_{i=1}^{N} \left( \| F_i(x_i) - F_i(y_i) \|^2 + \| x_i - y_i - (F_i(x_i) - F_i(y_i)) \|^2 \right) 
\] (c)
\[
\leq \sum_{i=1}^{N} \| x_i - y_i \|^2 = \| x - y \|^2 
\]
where (c) holds because each \( F_i \) is firmly non-expansive. As a result, \( F \) is also firmly non-expansive.

(ii). To prove that \( G \) is firmly non-expansive, we recall from Theorem 1 that \( 2G - I \) is self-inverse. Since \( G \) is linear, it has a matrix representation. Thus, \( \| (2G - I)x \|^2 = x^T (2G - I)^T (2G - I) x \). Because \( G \) is an averaging operator, it has to be symmetric, and hence \( G^T = G \). As a result, we have \( \| (2G - I)x \|^2 = \| x \|^2 \) for any \( x \), which implies non-expansiveness.

(iii). If \( F \) and \( G \) are both firmly non-expansive, we have
\[
\| (2G - I)(2F - I)(x) \| \leq \| (2G - I)(2F - I)(y) \| \leq \| x - y \|^2 
\]
where (a) is true due to the firmly non-expansiveness of \( G \) and (b) is true due to the non-expansiveness of \( F \). Thus, \( T \equiv (2G - I)(2F - I) \) is non-expansive. This result also implies convergence of the MACE algorithm, due to (a).

\[ \blacksquare \]

**REFERENCES**

[1] M. Zyda, “From visual simulation to virtual reality to games,” *Computer*, vol. 38, no. 9, pp. 25–32, Sep. 2005.

[2] F. Biocca and M. R. Levy, Eds., *Communication in the age of virtual reality*, Routledge, Feb. 2013.

[3] G. C. Burdea and P. Coiffet, *Virtual reality technology*, vol. 1, John Wiley & Sons, Jun. 2003.

[4] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, “Evaluating display fidelity and interaction fidelity in a virtual reality game,” *IEEE Trans. on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 626–633, Apr. 2012.

[5] J. Chen, D. Bautemachb, and S. Izadi, “Scalable real-time volumetric surface reconstruction,” *ACM Trans. on Graphics (ToG)*, vol. 32, no. 4, Jul 2013, Article 113.

[6] A. Levin, A. Rav-Acha, and D. Lischinski, “Spectral matting,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1699–1712, Oct. 2008.

[7] Y. Zheng and C. Kambhamettu, “Learning based digital matting,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Sep. 2009, pp. 889–896.

[8] Q. Chen, D. Li, and C. Tang, “KNN matting,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 9, pp. 2175–2188, Sep. 2013.

[9] E. Shahrian, D. Rajan, B. Price, and S. Cohen, “Improving image matting using comprehensive sampling sets,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013, pp. 636–643.
[10] D. Cho, Y. Tai, and I. Kweon, “Natural image matting using deep convolutional neural networks,” in Proc. European Conference on Computer Vision (ECCV), Oct. 2016, pp. 626–643, Springer.

[11] Z. Zivkovic, “Improved adaptive gaussian mixture model for background subtraction,” in Proc. IEEE International Conference on Pattern Recognition, Aug. 2004, vol. 2, pp. 28–31.

[12] J. W. Davis and V. Sharma, “Fusion-based background-subtraction using contour saliency,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2005, pp. 11–19.

[13] V. Mahadevan and N. Vasconcelos, “Background subtraction in highly dynamic scenes,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2008, pp. 1–6.

[14] M. Campilani, L. Maddalena, G. M. Alcover, A. Petrosino, and L. Salgado, “A benchmarking framework for background subtraction in RGBD videos,” in New Trends in Image Analysis and Processing-ICIAP 2017 Workshops. Sep. 2017, pp. 219–229, Springer.

[15] M. Hofmann, P. Tiefenbacher, and G. Rigoll, “Background segmentation with feedback: The pixel-based adaptive segmenter,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2012, pp. 38–43.

[16] A. Levin, D. Lischinski, and Y. Weiss, “A closed-form solution to natural image matting,” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 228–242, Feb. 2008.

[17] J. Sun, J. Jia, C. K. Tang, and H. Shum, “Poisson matting,” ACM Trans. on Graphics (ToG), vol. 28, no. 3, pp. 315–321, Aug. 2009.

[18] E. Gastal and M. Oliveira, “Shared sampling for real-time alpha matting,” Euro Graphics, vol. 29, no. 2, pp. 575–584, 2010.

[19] Y. Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, “A Bayesian approach to digital matting,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Dec 2001, vol. 2, pp. 11–18.

[20] J. Wang and M. F. Cohen, “Optimized color sampling for robust matting,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2007, pp. 1–8.

[21] N. Xu, B. Price, and T. Huang, “Deep image matting,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 311–320.

[22] Y. Chuang, A. Agharwa, B. Curless, and D. H. Salesin, “Video matting of complex scenes,” ACM Trans. on Graphics (ToG), vol. 21, no. 3, pp. 243–248, Jul. 2002.

[23] N. Apostoloff and A. Fitzgibbon, “Bayesian video matting using learnt image priors,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2004, vol. 1, pp. 1–8.

[24] S. Lee, J. Yoon, and I. Lee, “Temporally coherent video matting,” Graphical Models, vol. 72, no. 3, pp. 25–33, May 2010.

[25] N. Marki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, “Bilateral space video segmentation,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 743–751.

[26] S. A. Ramakanth and R. V Babu, “Seamseg: Video segmentation using patch seams,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2014, vol. 2, pp. 376–383.

[27] S. D. Jain and K. Grauman, “Supervoxel-consistent foreground propagation in video,” in Proc. European Conference on Computer Vision (ECCV), Sep. 2014, pp. 656–671, Springer.

[28] C. Hsieh and M. Lee, “Automatic trimap generation for digital image matting,” in Proc. IEEE Signal and Information Processing Association Annual Summit and Conference (APSIPA), Oct. 2013, pp. 1–5.

[29] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Intertactive foreground extraction using iterated graph cuts,” ACM Trans. on Graphics (ToG), vol. 23, no. 3, pp. 309–314, Aug. 2004.

[30] J. Cho, T. Yamasaki, K. Aizawa, and K. H. Lee, “Depth video camera based temporal alpha matting for natural 3d scene generation,” in 3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), May 2011, pp. 1–4.

[31] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang, “Automatic natural video matting with depth,” in 15th Pacific Conference On Computer Graphics and Applications, Oct. 2007, pp. 469–472.

[32] G. Buzzard, S. H. Chan, S. Sreehari, and C. A. Bouman, “Plug-and-play unplugged: optimization free reconstruction using consensus equilibrium,” SIAM Journal on Imaging Science, vol. 11, no. 3, pp. 2001–2020, Sep. 2018.

[33] S. Sreehari, S. V. Venkataraman, B. Wohlfeg, G. T. Buzzard, L. F. Drummm, J. P. Simmons, and C. A. Bouman, “Plug-and-play priors for bright field electron tomography and sparse interpolation,” IEEE Trans. on Computational Imaging, vol. 2, no. 4, pp. 408–423, Dec. 2016.

[34] S. H. Chan, X. Wang, and O. A. Elgendy, “Plug-and-play ADMM for image restoration,” IEEE Trans. on Computational Imaging, vol. 3, no. 1, pp. 84–98, Mar. 2017.

[35] S. Venkataraman, C. Bouman, and B. Wohlfeg, “Plug-and-play priors for model based reconstruction,” in Proc. IEEE Global Conference on Signal and Information Processing, Dec. 2013, pp. 945–948.

[36] X. Wang and S. H. Chan, “Parameter-free-plug-and-play ADMM for image restoration,” in Proc. IEEE International Conference on Acoustic, Speech, Signal Process. (ICASSP), Mar. 2017, pp. 1323–1327.

[37] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Foundations and Trends in Machine learning, vol. 3, no. 1, pp. 1–22, Jul. 2011.

[38] N. Parikh and S. Boyd, “Proximal algorithms,” Foundations and Trends in Optimization, vol. 1, no. 3, pp. 127–239, Jan. 2014.

[39] S. V. Burtsev and Y. P. Kuzmin, “An efficient flood-fill algorithm,” Computers & Graphics, vol. 17, no. 5, pp. 549–561, Sept 1993.

[40] S. H. Chan, R. Khoshabeh, K. B. Gibson, P. E. Gill, and T. Q. Nguyen, “An augmented Lagrangian method for total variation video restoration,” IEEE Trans. on Image Processing, vol. 20, no. 11, pp. 3097–3111, May 2011.

[41] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, “Image denoising by sparse 3D transform-domain collaborative filtering,” IEEE Trans. on Image Processing, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[42] K. Zhang, W. Zuo, S. Gu, and L. Zhang, “Learning deep cnn denoiser for image restoration,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jul. 2017, pp. 2808–2817.

[43] K. Cho, S. Kim, and Y. W. Tai, “Automatic trimap generation and consistent matting for light-field images,” IEEE Trans. on pattern Analysis and Machine Intelligence, vol. 39, no. 8, pp. 1504–1517, Aug. 2017.

[44] O. Barnich and V. D. Marc, “ViBe: A universal background subtraction algorithm for video sequences,” IEEE Trans. on Image Processing, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[45] A. Faktor and M. Irani, “Video segmentation by non-local consensus voting,” in Proc. British Machine Vision Association (BMVC), Jun. 2014, vol. 2, p. 8.

[46] R. Guo, Q. Dai, and D. Hoiem, “Single-image shadow detection and removal using paired regions,” in proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2011, pp. 2033–2040.

[47] E. Arbel and H. Hel-Or, “Shadow removal using intensity surfaces and texture anchor points,” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 33, no. 6, pp. 1202–1216, Jun. 2011.

[48] F. Liu and M. Gleicher, “Texture-consistent shadow removal,” in Proc. European Conference on Computer Vision (ECCV), Oct. 2008, pp. 437–450, Springer.