Trimap Synthesis with a Conditional Adversarial Network

Mauricio Aravena Cifuentes
Estudiante de Ingeniería
Cívil Electrónica
Universidad Técnica Federico Santa María
Valparaíso, Chile
maurio.aravena@sansano.usm.cl

Diego Badillo San Juan
Estudiante de Ingeniería
Civil Electrónica
Universidad Técnica Federico Santa María
Valparaíso, Chile
diego.badillo@sansano.usm.cl

***Abstract*—In this article, we will continue with the study of *Alpha Matting*, which we started in "Study and Benchmarking on Image Matting state-of-the-art methods" [1], where we compared the performance of two methods against this problem: (S. Sengupta, V. Jayaram, B. Curless, S. Seitz, and I. Kemelmacher-Shlizerman, "Background matting: The world is your green screen") and (Y. Aksoy, T.O. Aydın, and M. Pollefeys, "Designing effective inter-pixel information flow for natural image matting").**

Particularly, we will train a cGAN (*pix2pixHD*) to get a trimap given an input image containing a person. We will then use this trimap as an input for the Information Flow Matting algorithm, thus obtaining an alpha channel for the person, so that it can be placed against other backgrounds.

Lastly we will compare our results to those obtained in our previous report.

I. Introducción

Extraer el fondo (background) de una imagen que contenga un primer plano de interés a segmentar, es una tarea útil y esencial que puede usarse en varias aplicaciones. Obtener una máscara binaria de un objeto o persona de interés en una imagen puede servirnos para saber a grandes rasgos su ubicación espacial, su forma, su área, etc.

En la realidad, es poco común que un objeto de interés tenga un límite bien definido con el fondo, al capturarlo en una imagen o vídeo. También puede darse el caso de que, incluso dentro de los contornos del objeto, haya información del fondo debido a la transparencia que pueda presentar el primer plano (objeto de interés).

Alpha Matting se refiere al problema de separar el fondo y el primer plano considerando distintos niveles de opacidad (256 para un canal alfa de una imagen codificada en 8 bits). Para esto, cada píxel de la máscara a crear representará una combinación convexa ((1), en donde $0 \leq \alpha \leq 1$) entre lo que se considera fondo (Background) lo que se considera primer plano (Foreground).

$$I = \alpha F + (1 - \alpha)B \tag{1}$$

Los algoritmos analíticos enfocados en resolver este problema usualmente necesitan un trimapa (Figura 1) como entrada: [2], [3], [4]. Sin embargo, la tarea de crear un trimapa a partir de una imagen que contenga un *foreground* y *background* no es simple. En [5] los autores proponen un método analítico para

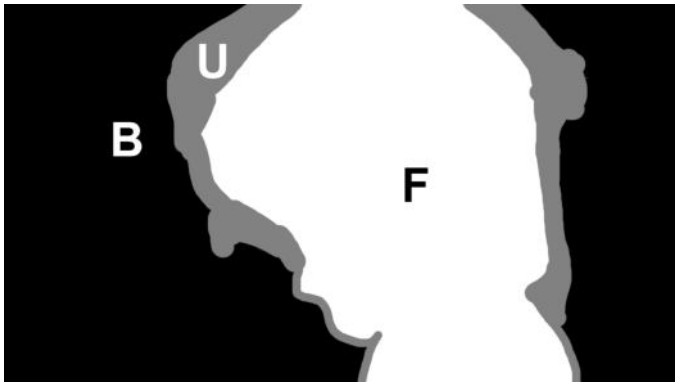

Figura 1. Trimapa. U: Unknown. B: Background. F: Foreground.

la generación de trimapas que consiste en la una segmentación binaria inicial de la imagen, de la que se obtiene la zona **U** a partir de un análisis de la distribución de colores en el borde de la segmentación. Nosotros proponemos generar los trimapas a partir del entrenamiento de una red GAN [6], en particular, una cGAN (*Conditional Generative Adversarial Network*). A grandes rasgos, la red que usamos [7] es una mejora de [8], y consiste en lograr obtener la traducción de una imagen *A* a una imagen *B*, en donde en nuestro caso, la imagen *B* corresponderá al trimapa. Su funcionamiento y arquitectura se ahondará con más detalle en la sección II.

En nuestro trabajo anterior [1] comparamos el desempeño de dos métodos [2], [9] para enfrentar el problema de *Alpha Matting*. El primero [2] usa un método analítico de afinidad entre valores de los píxeles, y recibe como entrada un trimapa y una imagen a la que se quiere encontrar el canal alfa. El segundo método estudiado [9] usa un método de aprendizaje *deep*, y recibe como entrada una imagen del *background*, y una de una persona en este *background*, a la que quiere obtenerse su canal alfa. El método analítico presentaba resultados más robustos que el método *deep*, que fallaba en segmentar correctamente a imágenes en que la piel de la persona sea de un color no natural (verde, en nuestro caso). Por la mayor robustez que presentaba el método analítico, es que nuestra propuesta es usar la red cGAN para generar el trimapa, que ocupará el algoritmo IFM para su entrada y generación del canal alfa.

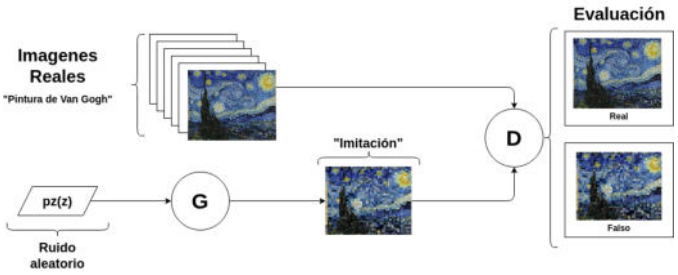

Figura 2. Arquitectura propuesta en [6], siguiendo la analogía expuesta de un falsificador y un experto en pinturas de Van Gogh. La pintura corresponde a *The Starry Night* [10]

II. Trabajo relacionado

En esta sección explicaremos el funcionamiento y arquitectura de una red GAN (Generative Adversarial Network, desde sus comienzos [6] hasta la arquitectura de la red condicional *pix2pixHD* que usamos para el entrenamiento.

II-A. Generative Adversarial Nets [6]

Contextualizando el estado del arte, para el año 2014, el mayor éxito para las aplicaciones del *deep-learning* se encontraba en la clasificación, dada por modelos de discriminación capaces de tomar información de alta dimensionalidad a una etiqueta. Estas redes discriminatorias o de clasificación, son capaces de aprender a reconocer la presencia y ubicación de un sujeto en la escena, para su posterior etiquetado, por ejemplo una red entrenada para distinguir entre perros y gatos, esta ha *aprendido* rasgos que le permiten tener una idea general de cómo distinguir el uno del otro, tiene una idea base de lo que es cada uno. Sin embargo, esta red no puede extender esta idea de distinguir entre dos etiquetas, a la generación de una imagen que pertenezca a alguna de estas, en términos simples, no se le puede pedir que genere un perro/gato *que nadie haya visto*. Este campo de modelos de redes, `Deep-Generative`, hasta el minuto no ha tenido mucho desarrollo dada la problemática de cómo utilizar la información aprendida por la red, para la generación de imágenes que correspondan a la distribución dada por el dominio de entrenamiento. El enfoque tradicional para la problemática, correspondería en entrenar una red, que a partir de un dataset de lo que se busque generar, aprenda la distribución de éste, para poder conseguir una *generalización* que le permita minimizar el error de la salida y acercarse a lo esperado. Para esta arquitectura, el resultado tenderá a ponderar a un promedio de las imágenes de entrenamiento, por lo que para un espectador estos destacarían claramente como falsos. Se necesita, cierta aleatoriedad en el resultado, no uno que converja aproximadamente a lo mismo.

Es entonces, que se propone la implementación de una nueva arquitectura [6]. Los autores proponen poner a la red a competir contra sí misma. Se consideran dos participantes o **adversarios**, una red generadora `G`, la cual a partir de ruido de entrada busca replicar la distribución del dominio y una red discriminadora `D`, la cual busca determinar si el resultado proveniente de `G`, es un elemento del dominio o un resultado generado por `G`. Una analogía útil para entender el concepto propuesto, es pensar en la red `G` como un falsificador de arte y en la red `D` como un experto en arte tratando de determinar si la pieza que le muestran es efectivamente una obra de Van Gogh o una imitación (Figura 2). A medida que la red `G`, estudia diversas obras pintadas por el autor, va entendiendo de mejor manera como replicar las pinceladas originales y con pequeños cambios intenta engañar al experto `D`, el cual a su vez estudia las replicas en busca de detalles que le permitan separar las obras reales de las copias. Este enfoque permite que la red se centre en corregir *debilidades* de la generación de resultados, dado que ambas redes están compitiendo por mejorar y *ganar* frente a la otra. El entrenamiento, entonces, se llevaría a cabo hasta que los resultados generados por `G` sean capaces de engañar de forma consistente a la red `D`. En resumen, se ha logrado realizar una replica de un Van Gogh, que para el experto, puede o no haber sido pintada por el autor, no es capaz de discernir.

II-A1. Definiendo a los adversarios: Antes de adentrar en cómo se implementa esta rivalidad entre las redes, es necesario hablar de los participantes. En primer lugar se tiene a la red generadora `G`, esta red neuronal es la encargada de tomar la información proveniente de un dominio, en este caso ruido aleatorio y llevarlo a un nuevo dominio, siguiendo la analogía acerca de las pinturas de Van Gogh, a una nueva pintura. La red generadora espera, a partir de esta entrada aleatoria, aprender la distribución que modela el dominio deseado. La distribución p_g de la red generadora se define como la distribución de muestras $G(z)$ obtenida cuando $z \sim p(z)$. Es de interés que esta distribución p_g converja a un estimador cercano a la distribución que modela el dataset p_{data}, dado suficiente tiempo y recursos, como se puede ver en la figura 3d. Esta convergencia es posible dado que la acción del segundo participante, la red `D`, permite a la red generadora fortalecer los *mapeos* de la entrada aleatoria que se corresponden con características que están dentro de la distribución del dominio del dataset. Este participante, que se comporta como el experto en arte, es una red clasificadora, la cual se entrena para asignar la etiqueta correcta a la entrada, para identificar si es una imagen del dominio o una generada por `G`. A medida que se entrena la red en su totalidad, el discriminador va mejorando su percepción de las entradas, permitiendo identificar y castigar las entradas que detecta como falsas y por ende provenientes de la red `G`. De esta forma se espera que la red evolucione hacia lo visto en la figura 3**d)**, donde la red generadora converge a la distribución que modela el dataset y la red discriminadora convergió a un punto de equilibrio con $p = 0{,}5$, indicando que no puede discernir si una imagen pertenece al dataset o fue generada, se alcanzó el punto donde el falsificador hace dudar al experto en arte.

II-A2. Objetivo GAN: Dada esta idea de implementar `G` y `D` como adversarios, se tiene que la función objetivo se convierte en un *two-player minimmax game* (Ecuación 3). Esto se refiere a que ambas redes tienen objetivos opuestos en el entrenamiento. Si la red `D` se mueve hacia su óptimo, indicando una mejora en la identificación de resultados falsos

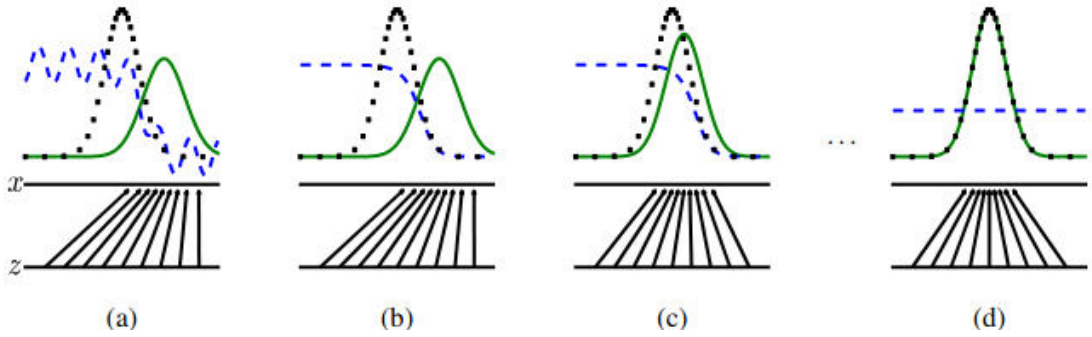

Figura 3. Evolución del aprendizaje de la red. La línea verde punteada representa la distribución de generación. La línea azul representa la distribución de discriminación (D). La línea negra representa la distribución que modela los datos de entrada (dataset). **a)** Adversarios cerca del punto de convergencia. **b)** Actualización de entrenamiento de la red D. **c)** Actualización de entrenamiento de la red G. **d)** Punto *ideal* de convergencia, dado tiempo y recursos suficientes.

en la entrada, es de interés que la red G, se mueva en dirección opuesta, de esta forma permitiendo que los resultados de G sean aún más problemáticos para el discriminador. De esta forma se puede formalizar la función *loss* de la red como:

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D(y)] + \mathbb{E}_{x\sim p_{\text{data}}(x),z\sim p_z(z)}[\log(1-D(G(x,z))] \quad (2)$$

$$\min_G \max_D \mathcal{L}_{GAN}(G,D) \quad (3)$$

II-B. Image-to-Image Translation with Conditional Adversarial Networks (pix2pix) [8]

El propósito principal de este artículo, es el uso de redes cGAN como una "solución general a problemas de traducción imagen a imagen". Para cada problema en particular, es necesario definir una función *loss* acorde al problema. En [11] se utiliza como función loss a minimizar la distancia euclidiana entre la *groundtruth* y la imagen predicha, esto genera resultados borrosos puesto que minimizar la distancia euclidiana implica promediar las posibles salidas. Elegir la función loss a utilizar para un problema en concreto es un problema abierto que generalmente requiere un análisis arduo del problema en particular. Aquí es donde es conveniente el uso de una red GAN, que *aprenden* la función loss del problema al presentar en su arquitectura dos redes que se entrenan mutuamente, como es explicado en II-A. El problema entonces se reduciría a entrenar la red con un par de datasets, *A* y *B* relacionados entre sí, con el único propósito de, dada una imagen de A, queremos convertirla a su par en B. El discriminador de la red GAN, se entrena entonces para "hacer que la salida generada por el generador sea indistinguible de la *realidad*". [1]

La diferencia de *pix2pix* del trabajo pasado con cGANs, es la arquitectura usada para el generador y el discriminador. En particular, para el generador se usa una arquitectura *U-Net* [12], que tiene una estructura simétrica de dos fases, de contracción y expansión; y para el discriminador un clasificador convolucional que los autores llaman "Patch GAN", que clasifica en parches que se corren convolucionalmente por la imagen. Una ventaja de la última arquitectura mencionada, es que al aplicarse en parches, puede correr más rápido, y ser más liviana, al tener menos parámetros que entrenar.

En una GAN condicional [13], la función objetivo puede expresarse como

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y\sim p_{\text{data}}(x,y)}[\log D(x,y)] + \mathbb{E}_{x\sim p_{\text{data}}(x),z\sim p_z(z)}[\log(1-D(x,G(x,z))] \quad (4)$$

en donde G intenta *engañar* al discriminador (i.e. minimizar (4)), y D intenta identificar imágenes falsas (i.e. maximizar (4)): $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D)$. Al comparar la función objetivo de una cGAN (4) con la de una GAN (2), la diferencia está en que el discriminador está *condicionado* a la entrada x (es una entrada más).

Algunas aplicaciones de *pix2pix* se muestran en la figura 4.

II-C. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs (pix2pixHD) [7]

Este artículo se presenta como una mejora a la estructura de *pix2pix*, con el propósito principal de mejorar el desempeño de las imágenes generadas para mayores resoluciones, es decir, logrará crear imágenes más realistas y con mayor nivel de detalles. Para esto se propone una nueva arquitectura para el generador G, que vaya de lo general a lo particular en la imagen (*coarse-to-fine*), como así también un discriminador D de multiescala. También se propone una mejora en la función objetivo, haciéndola más robusta.

II-C1. Generador G "coarse-to-fine": Consiste en una separación en dos subredes G_1: generador global y G_2 generador local *mejorador*. La idea principal, es que el generador global G_1 trabaje a menores resoluciones (mediante un downsampling a la imagen original), para así obtener un entrenamiento de las características generales que se quieren aprender del dataset. El generador local *mejorador* G_2 es conectado en cascada como se muestra en la figura 5. Esta misma idea se puede replicar con más generadores locales, conectados como un envoltorio del generador G' anterior, para así lograr

[1] Realidad en el sentido de la *groundtruth*. En nuestro caso, la "realidad" corresponderá a los trimapas segmentados manualmente usados en el dataset para entrenar.

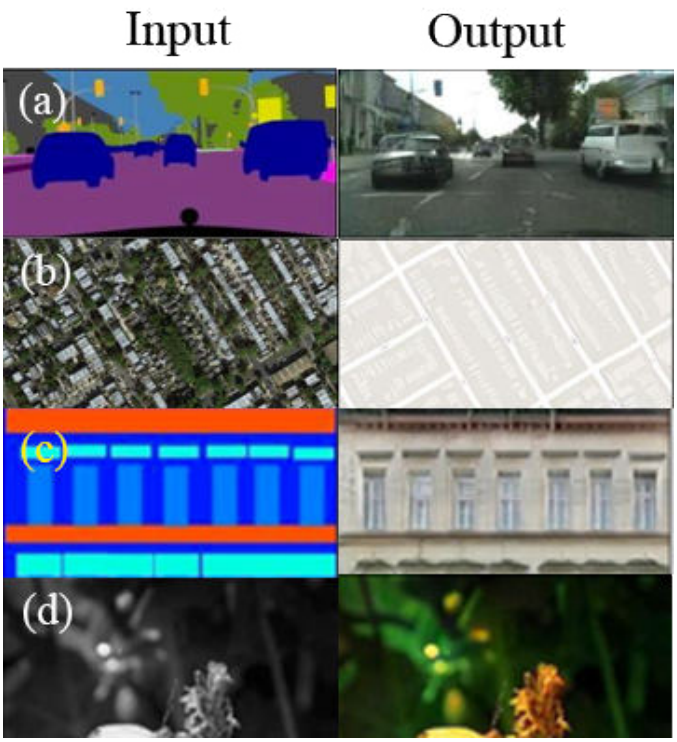

Figura 4. Algunas aplicaciones de *pix2pix*: Labels to street (a), aerial to map (b), labels to facade (c), greyscale to colour (d)

entrenamientos cada vez con imágenes de mayor resolución. La idea general es, entonces, entrenar primero a G_1 luego G_2 por separado, y finalmente unirlas para un entrenamiento final.

II-C2. Discriminador D multiescala: Implementar un discriminador que sea capaz de detectar entre imágenes reales e imágenes sintetizadas es un problema complejo a altas resoluciones, puesto que se necesitaría un mayor número de parámetros, lo que causaría directamente una red más pesada y propensa a un overfitting del dataset de entrenamiento. Para atacar este problema, se usa un enfoque similar que para el generador, y se proponen 3 sub-discriminadores que trabajen a distintas escalas, en particular, con factor de downsampling x2 y x4 respectivamente. Esta arquitectura permite entonces un entrenamiento para imágenes de mayor resolución más práctico, puesto que, al entrenarse primero con una versión de menor resolución de la imagen, logra obtener características generales, que si luego permanecen en imágenes de mayor resolución, bastaría con agregar más discriminadores en cascada a mayores escalas, en vez de tener que reentrenar la red desde cero si es que se implementara solamente un discriminador general para toda la imagen.

II-C3. Mejora de función loss adversaria (Ecuación 4): Aprovechando que se ha diseñado un generador G y un discriminador D bajo la misma idea de trabajar a multiescalas, se propone que el discriminador *discrimine* en varias escalas, lo que obliga al generador a producir imágenes que sean "reales" (para el discriminador) en todas las etapas de G. Esto es representado matemáticamente en la ecuación 5, T se refiere al número total de capas en cada discriminador, y N_i el número de parámetros en cada capa.

$$\mathcal{L}_{\mathrm{FM}}\left(G, D_k\right)=\mathbb{E}_{(\mathbf{s}, \mathbf{x})} \sum_{i=1}^{T} \frac{1}{N_i}\left[\left\|D_k^{(i)}(\mathbf{s}, \mathbf{x})-D_k^{(i)}(\mathrm{s}, G(\mathrm{~s}))\right\|_1\right] \tag{5}$$

III. Nuestro enfoque

III-A. Importancia de un buen trimapa

Para el correcto funcionamiento de algoritmos de afinidad que usen de entrada un trimapa como es el caso de Information Flow Matting [2], es necesaria una buena segmentación del trimapa, en las respectivas zonas **B**, **U** y **F** (Figura 1). No basta para la creación del trimapa, por ejemplo, obtener una máscara del foreground, y aplicar operaciones morfológicas. Esto entregaría un resultado parecido al mostrado en la figura 6e. Decidir qué secciones corresponden a la zona gris, es una tarea que usualmente se debe realizar manualmente para buenos resultados. Por esto nosotros proponemos utilizar un método de aprendizaje para una generación de trimapas, utilizando la red cGAN Pix2PixHD [7].

III-B. Implementación de la red

Como se puede ver en la figura 7, proponemos dividir el problema en dos partes principales. En primer lugar el problema de la obtención del trimapa, para ello consideramos el entrenamiento de una red `Pix2PixHD` [7], la cual pueda tomar como entrada una imagen y generar el trimapa correspondiente. En segundo lugar, es la obtención del canal alfa

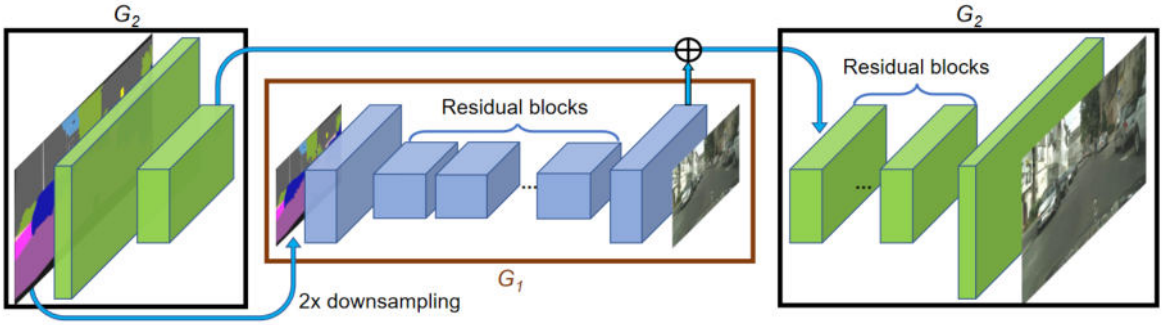

Figura 5. Diagrama de arquitectura para el generador propuesto. (Referenciado directamente de [7])

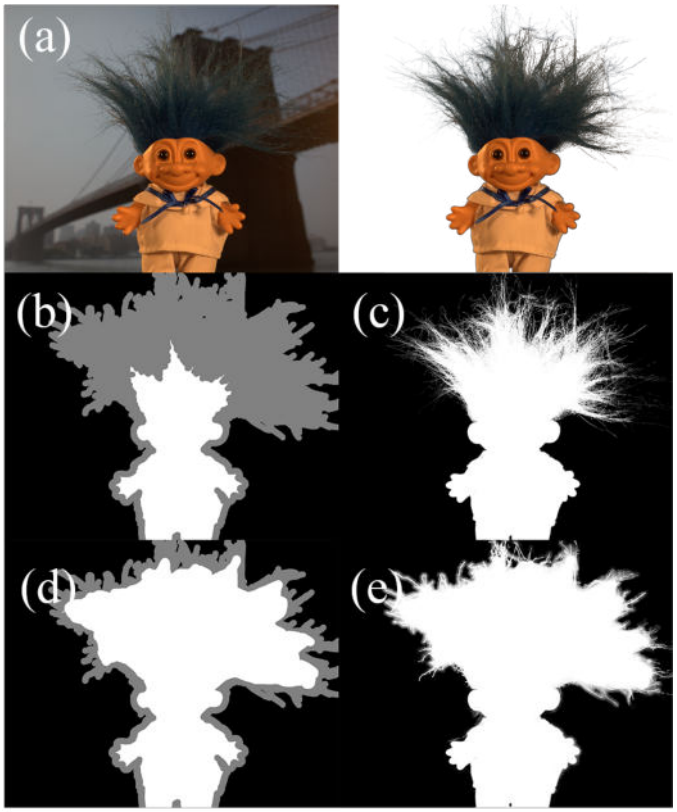

Figura 6. Importancia de un buen trimapa. Entrada (a), trimapa ideal (b), trimapa no ideal (d) y resultados usando IFM para cada trimapa respectivamente (c) y (e).

necesario para el *matting*. Para nuestro enfoque, consideramos los resultados obtenidos en nuestro benchmarking [1]. A partir de estos, determinamos que los mejores resultados se obtendrían utilizando la solución analítica dada por el algoritmo de *Information-Flow-Matting* [2]. De esta forma, el problema se puede concentrar en el entrenamiento de esta red generadora de trimapas.

III-C. Entrenamiento de la red

La red Pix2PixHD [7], se encuentra definida para la recepción de un dataset de entrada, con imágenes de tamaño definido en 1024×512. Si bien este puede ser aumentado o disminuido dependiendo de las necesidades de la aplicación, se tiene que tener en mente las limitaciones en términos de hardware que esto implica. Los autores de [7] recomiendan que para entrenar con el tamaño por defecto, se recomienda tener una *GPU* con al menos 11 GB de memoria *VRAM*. Esta cota inferior limitó las posibilidades de resolución con la que podíamos trabajar dado que al momento del diseño de esta red, contábamos con dos *GPUs*, una `GTX 1050 4 GB` y una `GTX 1060 6 GB`, por lo que la resolución de nuestro dataset tendría que al menos ser disminuida a 512×256, para poder ser entrenada en los equipos que teníamos a disposición. Sin embargo, pese a esta disminución la arquitectura más antigua presente en los `CUDACores`, implicaría que los tiempos de entrenamiento seguirían siendo altos, pese a esta disminución en la resolución, y que únicamente se podría trabajar con batches singulares a la entrada de la red. Frente a esto, se tomó la decisión de que se trabajaría con un dataset de imágenes de resolución 256×128. Debido a que nuestro problema consiste en generar un trimapa, y no un canal alfa con la red *pix2pixHD*, es que entrenar con imágenes de baja resolución no supone un problema: el trimapa no debe contener detalles, solo debe ser capaz de identificar zonas seguras de *foreground*, zonas seguras de *background* y zonas desconocidas. Una vez obtenido el trimapa con baja resolución, basta con reescalarlo al tamaño original mediante interpolación bilineal o bicúbica, y luego aplicar un *threshold* para asegurarnos que los píxeles intermedios grises queden a un valor de 128 (de 0 a 255).

Uno de los puntos a destacar de nuestra implementación de solución consiste en qué a diferencia de la red implementada para [9], la cual recibía dos entradas, siendo la primera la imagen y la segunda una donde el sujeto *fg* no estuviese presente (i.e. solo el *background*), nosotros esperamos que la red implementada sea capaz de trabajar únicamente con una entrada, que corresponde a la imagen completa a procesar. El resultado obtenido, finalmente se entrega al algoritmo `IFM` [2] para la obtención del canal alfa. Con esto se tendría todo lo necesario para responder a un problema de *image-matting*. Dada la extensión del dominio de este problema, se considerará una simplificación inspirada en el enfoque tomado en [9], donde para términos de funcionamiento se considera que la red sólo trabajará con imágenes donde exista una persona presente, la cual está mayormente de frente a la cámara, ya sea de cuerpo completo o únicamente un cuadro centrado en el rostro.

III-D. Métricas de evaluación

Como se discutió anteriormente, nos concentraremos en el problema de la generación del trimapa. Dado que los resultados para la evaluación del canal alfa generado mediante IFM fueron evaluados en [1], nos centraremos en los resultados directos de la generación del trimapa mediante la red `pix2pixHD`, por lo que utilizaremos métricas similares a las utilizadas en [1]: MAE y MSE, aplicadas a los trimapas del dataset de pruebas.

III-E. Dataset

III-F. Entrenamiento

III-F1. Formación de dataset base: Para definir nuestro dataset se utilizó una selección de imágenes del dataset de entrenamiento de Adobe [14] donde exista un sujeto humano, que esté con el rostro en su mayoría visible para la cámara, ya sea de cuerpo completo o con enfoque en el rostro. A partir de estas guías se obtuvo 133 imágenes de *foreground* y su respectivo canal alfa. Dado que se busca evaluar el resultado de generación de trimapa, se realizó manualmente la construcción de los trimapas para las imágenes seleccionadas. Se añadieron también 3 imágenes del dataset de `DAVID` [1], para un total de 136 imágenes con su respectivo canal alfa y trimapa. Dado que buscamos entregarle una imagen compuesta a la red (*foreground* y *background*), se utilizó un *pool* de

Figura 7. Diagrama de bloques de la solución propuesta.

423 imágenes de fondos tomadas por nosotros mismos. Como se mencionó anteriormente, dadas las restricciones a nivel de recursos (`GPU`), el dataset deberá ser construido con una resolución objetivo de 256×128.

III-F2. Extensión del dataset (data augmentation): Dado que un dataset de 136 imágenes, resultaría en un claro *overfitting* de la red generadora, es necesario añadir variaciones y perturbaciones al dataset, que además de evitar overfitting permitan a la red aprender de las posibles variaciones de los sujetos. Para esto se proponen los siguientes procedimientos a realizar para cada miembro del dataset (foreground, canal alfa y trimapa), en primer lugar **rotación**. Las imágenes serán rotadas en un rango de $\pm 15°$ de forma aleatoria, figura 9-b). Dado que esta operación puede causar que los sujetos queden *flotando* dentro del cuadro, se realiza una segunda operación, que consiste en **re-encuadre** de la imagen, es decir, un recorte para evitar el problema de los sujetos *flotantes* en la extensión del dataset, figura 9b. Considerando los resultados obtenidos en [1], una de las mejoras sugeridas, era la modificación del `HUE` de las imágenes del dataset de entrenamiento, dado que la red evaluada tenía un sesgo hacia tonos *naturales* de piel, dejando fuera la posibilidad de que si una persona fuese iluminada con una luz con algún color predominante, como por ejemplo verde, la piel al ser reflectante se tendería a ver de este color y la red evaluada no sería capaz de procesarla. Para solucionar este problema, se introdujo una operación de **cambio aleatorio de HUE**, la cual hace un movimiento de éste parámetro del espacio `HSV` de la imagen de *foreground*, figura 9c. Finalmente, se realiza una operación de **composición** donde del *pool* de *backgrounds* se selecciona uno de estos de forma aleatoria y se realiza la operación definida en la ecuación 1, utilizando el canal alfa asociado al *foreground*. Estas operaciones se realizan ocho veces para cada tripleta integrante del dataset. Además se añade una tripleta a la cual sólo se le realiza composición, de esta forma el dataset original es expandido nueve veces. Esto nos da un dataset de entrenamiento conformado por 1224 tripletas.

III-G. Pruebas

Para la evaluación de la red, generamos un dataset consistente de cuatro imágenes (Figura 10) para las cuales segmentamos de forma manual su trimapa. Estas imágenes cumplen los mismos criterios y limitaciones que el de entrenamiento, con la salvedad de que estas son imágenes sin rotaciones, ni cambios de `HUE` artificiales. Estas imágenes no son compuestas artificialmente por un *background* o *foreground*.

IV. Resultados

Adjuntamos algunos resultados para imágenes fuera del dataset de entrenamiento en la figura 10. En nuestro `github` está disponible además el mejor resultado para cada epoch para el dataset de entrenamiento. Esto se encuentra en `https://github.com/Wauro21/gan_trimap`.

Además, en la figura 11 adjuntamos un gráfico que muestra la evolución del MSE para una imagen fuera del dataset de entrenamiento. Es importante señalar que debido a lo pequeño del dataset, [2] la red no es capaz de generalizar correctamente, pero al menos es capaz de identificar, de manera gruesa, a las personas de la figura 10. Se espera que con un dataset de mayor tamaño, la red debería presentar un mejor desempeño. A modo de ejemplo, en el apéndice de [7], se señala que para las aplicaciones que usaron el modelo *pix2pixHD*, usaron 1200 imágenes de entrenamiento en el caso menor (NYU Indoor

[2]Sin considerar el *data augmentation*, el dataset consiste de 136 foregrounds con sus respectivos trimapas segmentados manualmente.

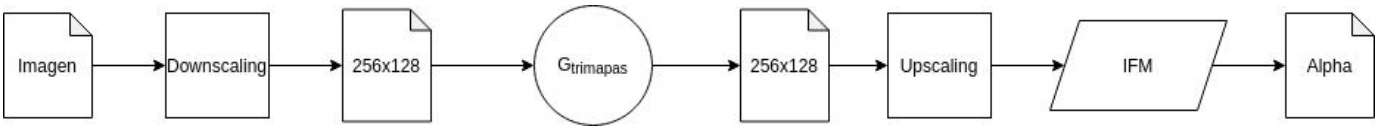

Figura 8. Solución considerando etapas de down/upscaling

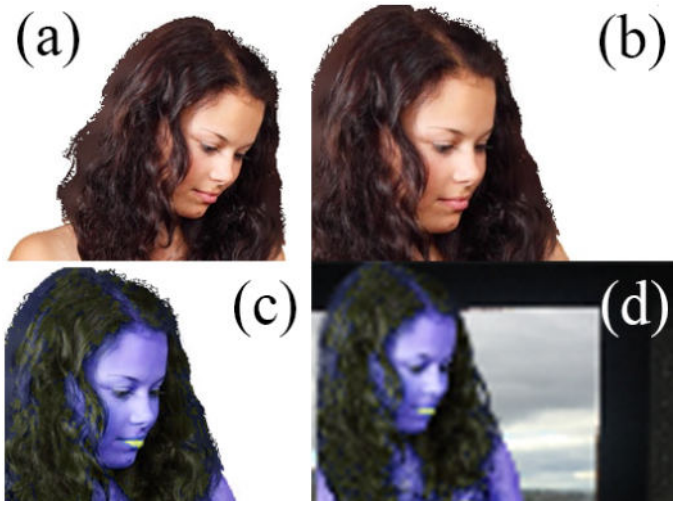

Figura 9. Pasos para el *data augmentation*: Foreground original (a), rotación aleatoria de -15º a 15º (b), cambio aleatorio del *Hue* (c), *downsampling* y *composite* con background aleatorio de un total de 423 (d).

Figura 10. Resultados para cuatro imágenes fuera del dataset de entrenamiento.

RGBD dataset), y 20210 imágenes de entrenamiento para el caso mayor (ADE20K dataset).

IV-A. Problema de sombras

Previamente, en [1] habíamos evaluado el desempeño del método en [9] para la generación del canal alfa dada una

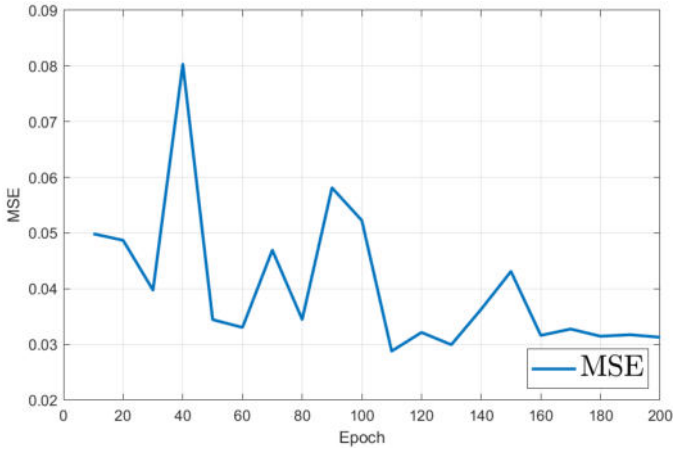

Figura 11. MSE para una imagen fuera del dataset de entrenamiento en función de la época.

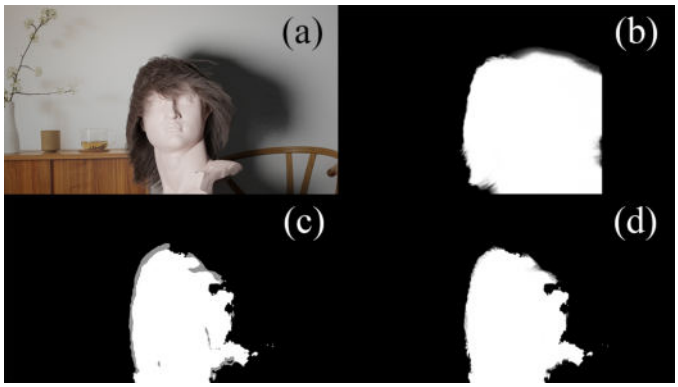

Figura 12. Problema de sombras: Imagen de entrada (a), salida mediante método de [9] (b), trimapa generado por *pix2pixHD* (c), salida de IFM usando nuestro trimapa (d).

persona (en nuestro caso usamos un modelo tridimensional del torso de una estatua) que proyectaba sombra sobre su entorno, de manera que al capturar una foto del *background* (paso necesario para la entrada de este método), en la segmentación se capturaba la sombra en el canal alfa (ver Figura 12). Debido a que nuestro enfoque no necesita una captura adicional del fondo para funcionar, entrega mejores resultados para este caso particular.

V. Conclusión

La intención de este trabajo fue comprobar la factibilidad de generación de trimapas usando una arquitectura cGAN, en particular la propuesta en [7]: *pix2pixHD*.

Hemos implementado un esquema que a diferencia de la solución propuesta en [9], no necesita de una entrada adicional del *background* para funcionar. Debido a que el dataset usado no es lo suficientemente extenso (136 foregrounds, y el resto obtenido de *data augmentation*), la red no logra generalizar para ciertos casos en los que, por ejemplo, se vea una persona de espaldas. Sin embargo, a pesar de esto, la arquitectura propuesta por [7] resulta generar resultados bastante robustos

con un dataset pequeño para imágenes similares a las usadas para el entrenamiento (torsos de personas de vista frontal, mayormente). Además debemos considerar que la red está haciendo un doble trabajo: detección de personas y generación de trimapa sobre ellas.

Considerando lo anterior, es esperable que la red no entregue resultados competitivos a [9], pero que, sin embargo, son prometedores si se generara un dataset más extenso, lo que requiere segmentación de trimapas manual. Así la red sería capaz de generalizar mejor.

Por otro lado, hemos atacado el problema de detección de pieles de colores poco naturales al considerar, en el *data augmentation*, variaciones del *hue* solamente de las personas antes de mezclarlas con los *backgrounds.*

Por último, al no usar imágenes de fondo como entrada adicional, como se hacía en [9], el problema de sombras ya no persiste, como se muestra en la figura 12.

AGRADECIMIENTOS

Nos gustaría agradecer a Brian Price [14] por su facilitación del dataset de Adobe, y a *STbanani* y *JamesBonk* por el préstamo de poder de procesamiento de sus GPUs para el entrenamiento.

REFERENCIAS

[1] M. A. C. and D. B. S., "Study and benchmarking on image mattingstate-of-the-art methods," 2021.
[2] Y. Aksoy, T. O. Aydın, and M. Pollefeys, "Designing effective inter-pixel information flow for natural image matting," in *Proc. CVPR*, 2017.
[3] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 228–242, 2008.
[4] L. Karacan, A. Erdem, and E. Erdem, "Image matting with kl-divergence based sparse sampling," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 424–432.
[5] D. Cho, S. Kim, Y.-W. Tai, and I. S. Kweon, "Automatic trimap generation and consistent matting for light-field images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 8, pp. 1504–1517, 2017.
[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.
[7] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8798–8807.
[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2018.
[9] S. Sengupta, V. Jayaram, B. Curless, S. Seitz, and I. Kemelmacher-Shlizerman, "Background matting: The world is your green screen," in *Computer Vision and Pattern Regognition (CVPR)*, 2020.
[10] V. V. Gogh, "The starry night."
[11] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," 2016.
[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.
[13] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.
[14] N. Xu, B. Price, S. Cohen, and T. Huang, "Deep image matting," 2017.