CEIG'09, San Sebastián, Sept. 9-11 (2009)
C. Andújar and J. LLuch (Editors)

Texturización automática en Entornos urbanos utilizando Algoritmos Genéticos

M[a] Dolores Robles Ortega, José López, Lidia Ortega, Francisco Feito

Dpto. Informática, Universidad de Jaén

Abstract

La representación virtual de entornos reales en 3D plantea ciertos problemas de texturización cuando dichos escenarios son de grandes dimensiones. Concretamente, el modelado de un entorno urbano con miles de edificios puede suponer un enorme reto si debe respetarse la información real del plano de la ciudad con manzanas, alturas de edificios reales, así como nombres de calles u otro tipo de información relevante. El objetivo final es posibilitar la navegación libre por dicho entorno virtual ofreciendo al usuario la mayor sensación de realismo posible, independientemente del lugar exacto por donde decida moverse en la ciudad, y exista o no información de interés en dicha zona.
A pesar de partir de toda la información posible disponible en un SIG urbano, éste no suele almacenar datos relativos al aspecto real de los inmuebles, lo cual resulta imprescindible para realizar un levantamiento realista de la ciudad completa. En este trabajo se propone una solución alternativa, aplicando texturas de forma automática a los edificios de los cuales no se tenga información exacta de su posible aspecto. Para ello se emplean dos algoritmos genéticos para asignar automáticamente texturas de una base de datos, uno para texturas superiores de edificios y otras para los bajos de dichos edificios. Para realizar una asignación de forma realista, esta metodología puede contemplar las características cuantitativas de las texturas, como anchura o altura de los inmuebles, o cualitativas como diversas zonas de la ciudad (casco antiguo o avenida de última construcción, etc.). Los resultados obtenidos muestran un buen ajuste de las texturas y un alto nivel de realismo.

Categories and Subject Descriptors (according to ACM CCS):
I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introducción

La definición de entornos virtuales sobre localizaciones reales posee unas características muy diferentes con respecto a los considerados ficticios. La mayoría de los sistemas de navegación virtual en entornos urbanos reales nos proporcionan, bien un recorrido aéreo de la ciudad o bien un recorrido peatonal sobre una pequeña porción de ésta. En el primer caso no se requiere alcanzar un alto nivel de detalle en los edificios o el mobiliario urbano como observamos en [CdSF03]. En el segundo suelen utilizarse texturas reales para alcanzar el mayor nivel de realismo, o gramáticas específicas para texturización [CBSF07,MWH*06,MZWVG07]. El reto surge, por tanto, cuando la navegación en modo peatonal puede realizarse libremente por todas las zonas de una ciudad manteniendo la misma sensación realista.

Para la mayoría de las personas que naveguen virtualmente por una ciudad concreta sería suficiente conseguir ubicar las zonas más emblemáticas de la ciudad, como el casco antiguo o la catedral, y les bastaría con localizar calles y conocer la ruta hacia determinados emplazamientos. Siendo así, cualquier SIG 3D contaría con texturas completamente realistas, procedentes de fotografías del lugar para las zonas más conocidas. Para el resto, sin embargo, utilizarían otras texturas similares para dar la sensación de continuidad, aunque no correspondieran exactamente con los edificios que representan.

El problema concreto que se plantea en este trabajo es

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG09/135-144

proporcionar un mecanismo de navegación libre a nivel peatonal en un sistema cliente-servidor sobre la ciudad de Jaén. La ciudad cuenta con miles de edificaciones, y este planteamiento nos llevaría a considerar igualmente miles de texturas reales, lo que no resulta operativo. En cualquier caso, el mantenimiento de un sistema así plantearía numerosos problemas a la hora de ser actualizado, por ejemplo con edificios nuevos, derruidos o simplemente repintados.

En este trabajo planteamos una solución al problema de texturización automática utilizando algoritmos genéticos. En la Sección 2 presentamos este problema con todo detalle, describimos el tratamiento de las texturas que se ha realizado, para posteriormente comentar la solución aportada utilizando un método de selección de texturas basado en algoritmos genéticos. La Sección 3 muestra los resultados obtenidos y finalmente describimos las conclusiones y posibles trabajos futuros en la Sección 4.

2. El problema de la texturización automática de edificios

Como se ha comentado en la Introducción, el proyecto en el que se encuadra este trabajo pretende conseguir la navegación libre en un entorno virtual urbano representando la ciudad de Jaén. El sistema es un modelo cliente-servidor esquematizado en la Figura 1. Un usuario accede al sistema mediante cualquier tipo de dispositivo o cliente, y se posiciona en algún lugar del plano de la ciudad. El servidor recibe la posición donde se ubica el usuario y construye el espacio virtual $3D$ desde donde iniciar un proceso de navegación a nivel peatonal. A pesar de la gran cantidad de edificios de cualquier ciudad, un peatón sólo verá una pequeña parte de ésta. La representación de la mayoría de los edificios de una ciudad son objetos $2{,}5D$, con geometría muy simple, lo que permite al módulo de oclusión trabajar en tiempo real para proporcionar el conjunto visible en cualquier posición del observador [ROOF09]. En este aspecto, el proceso de navegación está garantizado a pesar de ser implantado en un sistema cliente-servidor. Para conseguir el realismo deseado el sistema trabaja con texturas reales, aunque como detallaremos a continuación, el proceso de texturizado necesitará un procesamiento de texturas específico para completar la ciudad entera.

2.1. Descripción de los datos de entrada

Para llevar a cabo este proyecto se toma de partida la información del SIG $2D$ de la ciudad de Jaén en formato Mapinfo [DLW01], procedente de una versión parcial proporcionada por la Dirección General del Catastro. Las diversas capas de información de este SIG contienen el plano real de la ciudad, los nombres de las calles, la forma de edificios y manzanas, además de las alturas reales de dichos edificios. Sin embargo, cualquier SIG de estas características carece de la información necesaria para conocer el aspecto exterior de los edificios al realizar el levantamiento $3D$.

Formalmente, el problema que pretendemos resolver puede enunciarse como sigue: dado un edificio $E = \{P_E, H_E, A_E\}$, siendo $P_E = L_1 + L_2 + ... + L_n$ el perímetro, H_E la altura y $A_E = \{A_{1E}, A_{2E}, ...A_{nE}\}$ un conjunto de características asociadas al mismo (por ejemplo, zona de la ciudad, tipo de edificio, etc.) y un conjunto de texturas $T = \{T_1, T_2, ..., T_n\}$, con $T_n = \{h_n, a_n, t_n\}$, siendo h_n la altura del inmueble, a_n la anchura y t_n el tipo, obtener la mejor asignación posible de texturas al edificio $T_E = \{T_{1E}, T_{2E}, ..., T_{nE}\}$, de forma que la suma de las anchuras se corresponda con el perímetro, $\sum_{i=0}^{n} a_{i_E} = P_E$, las alturas de las texturas sean iguales que las del inmueble, $h_{1E} = h_{2E} = ... = h_{nE} = H_E$ y del mismo tipo, $t_{1E} = t_{2E} = ... = t_{nE} = A_E$.

A pesar de que Jaén es una ciudad de tamaño medio, existen más de mil manzanas y por tanto varios miles de edificios que texturizar. Esto nos hace suponer que la opción de realizar un "pegado a mano de texturas" es la más deseable, pero no la más factible inicialmente. Se cuenta de entrada con un millar de fotografías de ciertas zonas de la ciudad. Para estos casos, la texturización real es posible, sin embargo para el resto se ha optado por un proceso de texturización automática que permita al menos dar un aspecto semejante al que tendrían esas edificaciones.

La información más importante de la tabla de texturizado automático debe ser aquella que permita adaptar textura y polígono de la forma más idónea posible, y en este sentido la elección de la textura que mejor se ajusta a la anchura de dicho edificio es crucial. De no respetar este factor las texturas deberían estirarse hasta encajar con el perímetro o la fachada del edificio, haciendo que la visión global de la escena quede poco realista. La Figura 2 representa la estructura $2{,}5D$ de un edificio, de entre el conjunto de texturas representadas en la Figura 3 se deberían escoger aquellas que mejor se ajustaran a esos valores.

Mientras que las texturas reales deben guardar información sobre el lugar exacto donde fueron tomadas, en las texturas aleatorias será crucial identificar, para cada una de ellas, el ancho real del edificio al que corresponde. De este modo el modelo $3D$ resultante mejora la sensación de correspondencia a una posible realidad. Otro aspecto a tratar de modo diferente es la clasificación entre bajos de edificios (comerciales, portales, etc.) y las zonas de viviendas. Por último, como las alturas de los edificios reales es un dato conocido, interesaría que dichas alturas también fuesen incluidas en el prototipo virtual.

Por tanto, el algoritmo de asignación automático arista-textura debe considerar todos los aspectos anteriores. El proceso de adaptación de las alturas puede realizarse de modo simple, asignando texturas con un número determinado de plantas a edificios correspondientes. Sin embargo las anchuras de éstos son valores de tipo real y el ajuste necesita de un proceso más natural. El patrón seguido debería ser pseudo aleatorio para evitar que bajo circunstancias similares la respuesta sea siempre la misma, ya que la realidad no se

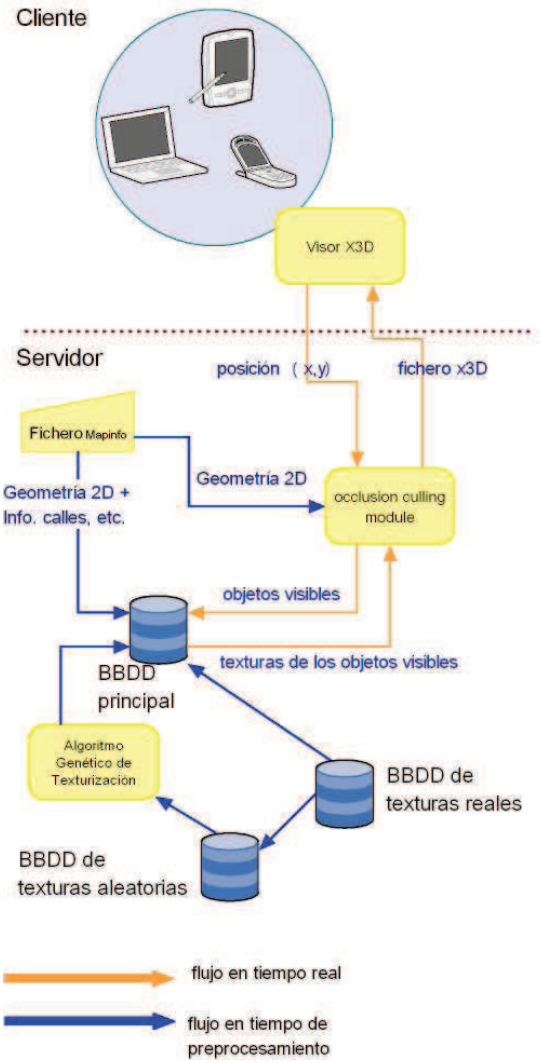


Figura 1: *Esquema general del sistema de navegación cliente-servidor.*

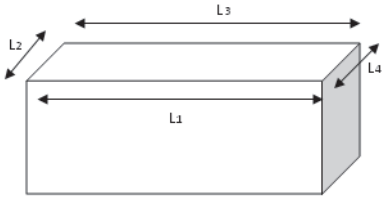


Figura 2: *El perímetro del edificio* $P = L_1 + L_2 + L_3 + L_4$

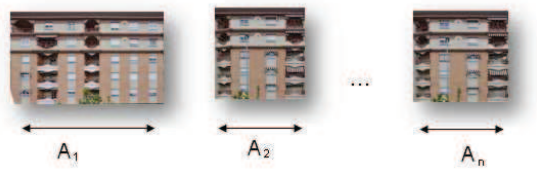


Figura 3: *Ajuste de texturas con fachadas de edificios.*

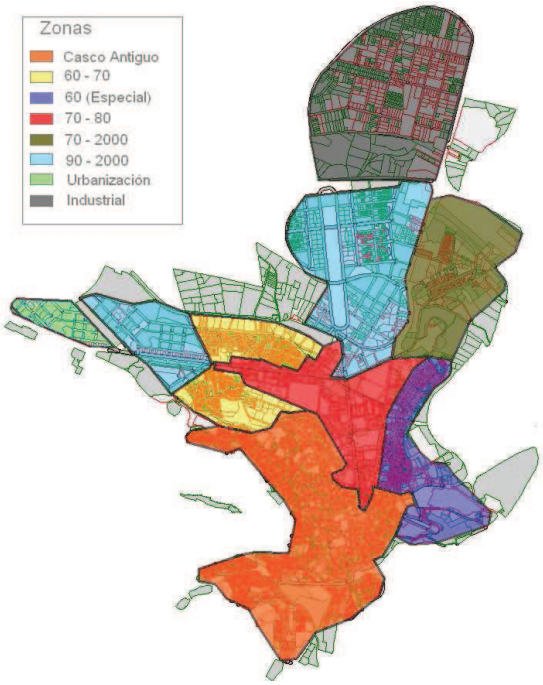


Figura 4: *Diferentes zonas de texturización en Jaén.*

comporta así. Esto nos hace descartar de entrada algoritmos voraces o Greedy porque son determinísticos en el comportamiento, otorgando por ejemplo siempre la misma textura a una fachada de siete metros. El factor de aleatoriedad junto con el de ajuste óptimo sería la opción más conveniente para dar solución a este planteamiento, por lo que optamos por implementar la asignación de texturas mediante un algoritmo genético.

2.2. Tratamiento de texturas

Según se desprende de lo anterior, el sistema de texturización se realiza en dos fases, una manual y otra automática, manteniéndose igualmente dos bases de datos diferentes de texturas como se observa en la Figura 1. Aunque exista información común en ambos casos, la base de datos de texturas realistas debe mantener información sobre la ubicación real de la textura. Entre otros campos mantendría la posición del GPS donde se toma la fotografía, hora del día para

Figura 5: *Segmentación de la fachada de un edificio.*

determinar posibles retoques fotográficos con respecto a la luminosidad de la foto, el nombre y el número de la calle donde ubicar el portal del edificio, etc.. Sin embargo, el proceso de texturizado automático requiere de una metodología y de una tabla de datos diferente. Estas texturas estarán formadas en su mayoría por porciones de otras procedentes de la base de datos realista, aunque modificando dichas texturas de modo adecuado. El tratamiento que se realiza de ellas consiste básicamente en realizar recortes basándose en los siguientes criterios:

- Troceado vertical de la textura real en diferentes porciones para ajustarse a diferentes tipos de anchos de fachadas.
- Troceado horizontal para adjudicar una textura a edificios con diferente número de plantas y para diferenciar la zona de viviendas del edificio (normalmente a partir de la primera planta) con los bajos comerciales o portales de acceso al inmueble.
- Clasificación de dichas texturas para adaptarlas a las diferentes zonas de Jaén, lo que permitiría no utilizar texturas de edificios construidos en la última década para adaptarlas a calles del casco antiguo que no sean tratadas con texturas reales. En el caso de Jaén se han clasificado varias zonas de edificios diferentes como se aprecia en la Figura 4. Esta clasificación corresponde normalmente con las zonas de expansión de la ciudad en las últimas décadas, aunque existe alguna zona mixta con edificaciones de diversas épocas (señalada con azul oscuro).

Los problemas que han ido surgiendo para realizar esta tarea han sido diversos. En primer lugar las fotografías en la base de datos de texturas aleatorias debe mantener una relación exacta entre tamaño de fotografía (ancho en píxeles) y tamaño de la fachada que representa. Si este ratio no se respeta se pierde la homogeneidad total del proceso de texturización. Otro factor importante es el propio proceso de tratamiento de imágenes sobre las fotografías reales. Las fotografías se toman normalmente entre un metro y dos a nivel del suelo, lo que requiere una corrección de la perspectiva. Existen elementos decorativos como las farolas que deben eliminarse para que tenga sentido la repetición de texturas. Otro problema encontrado es el posible formato totalmente heterogéneo de una fachada, como se observa en la Figura 5, donde existen diferencias también a nivel de altura en las mismas. En estos casos se realizaría un proceso de recortado clasificando correctamente cada porción en la base de datos.

2.3. Algoritmos Genéticos

Los algoritmos genéticos han sido utilizados con anterioridad para clasificación, identificación o generación de texturas, como se referencia en [AD97, EDA96, Wan07, QY02]. En este trabajo lo que se pretende realizar es una selección de aquellas texturas que se adapten mejor para la texturización de las fachadas de una manzana de edificios con una serie de restricciones impuestas. De entre las diversas alternativas para realizar esta asignación encontramos los algoritmos voraces o Greedy, que permiten obtener de forma rápida una solución no óptima pero sí aproximada del problema. Sin embargo estos algoritmos se consideran totalmente determinísticos, lo que iría en detrimento de la variabilidad en la asignación de texturas. Por contra, cualquier otro método totalmente aleatorio no permitiría realizar buenos ajustes. Para este caso hemos considerado conveniente utilizar la técnica de los algoritmos genéticos que, además de realizar la búsqueda, permiten incluir factores adicionales que en otro tipo de algoritmos sería muy difícil considerar. El algoritmo trabajaría una sola vez en la etapa de pre procesamiento alimentando la base de datos global del sistema, tal y como aparece en la Figura 1. Una vez que una textura es seleccionada para un edificio, esta conexión queda establecida para el futuro y es utilizada durante el proceso de navegación.

Un algoritmo genético es un algoritmo evolutivo que toma las bases del propio proceso de evolución biológica de las especies [ES03, And04]. La biología molecular ha ido evolucionando a lo largo de los milenios mejorando las especies y adaptándolas al medio al que pertenecen. Los objetivos de un algoritmo genético pueden ser el de optimización, de búsqueda o aprendizaje. Los datos de entrada se codifican como genes que forman un cromosoma. El valor que toma cada gen es lo que se denomina alelo.

El cromosoma evoluciona generación tras generación a base de cruzarse con otros cromosomas mutando a veces. Al final de cada ciclo el individuo representado por un cromosoma es evaluado y si el resultado no es satisfactorio para alcanzar un objetivo final se descarta, si por el contrario resulta más adaptado que sus ancestros seguirá formando parte de la evolución. Los ciclos de evolución se suceden hasta que se alcanza una solución suficientemente buena, siendo entonces cuando el algoritmo finaliza.

La definición de un algoritmo genético se establecen siguiendo el siguiente esquema general:

1. **Elección de una representación:** que puede ser binaria o

algorithm 1 Algoritmo Genético

Input:
Output:
Var:
BEGIN
1. $t \leftarrow \emptyset$
2. Inicializar $P(t)$
3. Evaluar $P(t)$
4. MIENTRAS (no se cumpla la condición de parada) HACER
   a) $t \leftarrow t+1$
   b) Seleccionar $P(t)$ desde $P(t-1)$
   c) Recombinar $P(t)$
   d) Mutar $P(t)$
   e) Evaluar $P(t)$
5. FIN_MIENTRAS

END

entera. Un valor binario por ejemplo, puede codificar un entero o un flotante, dependiendo de los datos de entrada.
2. **Inicialización:** se elige una representación inicial concreta si se conoce para el problema o totalmente aleatoria.
3. **Correspondencia entre genotipo y fenotipo:** el genotipo es la representación interna del cromosoma (normalmente binaria) y el fenotipo es la representación en el mundo real de dicho genotipo.
4. **Evaluación de un individuo:** un individuo después de haber pasado por una etapa de evolución debe ser evaluado para determinar su valía como solución del problema
5. **Definición del operador de mutación:** mutar implica aumentar las posibilidades de alcanzar cualquier parte en el espacio de búsqueda de la solución.
6. **Definición del operador de cruce:** permite generar nuevos individuos a partir de cromosomas padre.
7. **La selección:** permite elegir los mejores individuos de cada generación; estos individuos tendrán la posibilidad de reproducirse en el siguiente ciclo de la evolución. En cualquier caso siempre debe darse cierta oportunidad de reproducirse a los individuos menos buenos para introducir material genético útil en el proceso.
8. **Estrategia de reemplazamiento:** unas generaciones deben ser reemplazadas por los nuevos descendientes; el método para hacer este proceso puede ser a su vez determinístico o aleatorio.
9. **Criterio de parada:** llegado a un punto de la evolución, si el algoritmo ha proporcionado una solución suficientemente buena (no tiene por qué ser la óptima) el algoritmo finaliza.

El proceso evolutivo se realiza según los pasos indicados en el Algoritmo 1. Como veremos a continuación, la adaptación de este esquema general al problema de texturización automática consiste en considerar a un cromosoma como representante de todas las posibles texturas asociadas a un edificio.

2.4. Algoritmo de texturización

Para realizar un proceso de texturación automática, partimos de una base de datos con numerosas texturas con la que se pretende realizar un ajuste inteligente sobre manzanas de edificios. Este proceso debe tener en cuenta numerosos aspectos:

- El tamaño de la arista del polígono (manzana) y el tamaño real que representa la textura deben coincidir lo más posible; de otro modo una textura se estiraría demasiado en una cara del polígono perdiéndose la sensación de realismo, por ejemplo al aparecer ventanas demasiado anchas.
- Al igual que la anchura, la información catastral proporciona la altura real de cada edificio; el sistema adjudicará texturas en base a este dato.
- Este mecanismo de texturización debe asignar texturas adecuadas para cada una de las zonas de Jaén representadas en la Figura 4, aunque no se han tenido en cuenta la zona industrial y la de urbanizaciones por no corresponder con objetos $2{,}5D$.
- Las texturas de los bajos de los edificios se comportan normalmente de modo diferente al resto de plantas; un edificio tiene al menos un portal de entrada, el resto son locales comerciales o simples muros.

En base a estos requerimientos, y teniendo en cuenta que existen características muy diferentes en la adjudicación de texturas a bajos de edificios y a zonas superiores, se decide dar una solución independiente a cada caso utilizando dos algoritmos genéticos. En ambos casos se sigue el esquema general descrito en la Sección 2.3 en la que un cromosoma representa todas las texturas que podrían seleccionarse para texturizar una manzana de edificios.

Sin embargo, existen otra serie de características específicas del genético que opera con texturas superiores y el de texturas inferiores, como destacamos a continuación.

Texturas inferiores:

Como se ha comentado anteriormente, se desea obtener como solución un conjunto de texturas no repetidas cuya suma total de longitudes se corresponde con el perímetro de la manzana. Este problema de texturización puede considerarse similar al problema de la Suma de Subconjuntos [CLRS01]: dado un conjunto de enteros, ¿existe algún subconjunto no vacío cuya suma sea exactamente cero?. En nuestro caso, los componentes del conjunto serían los valores de las longitudes de cada una de las texturas que disponemos expresados en forma decimal, mientras que la suma se correspondería con el perímetro de la manzana en lugar de cero. Aunque el enunciado es simple, se trata de un problema de decisión NP-Completo [GJ79], puesto que no puede encontrarse un algoritmo que lo solucione en tiempo polinómico. Realmente saber si una solución es correcta es sencillo, pero actualmente no se conoce mejor solución que explorar los 2^n subconjuntos posibles hasta encontrar alguno que cumpla la condición.

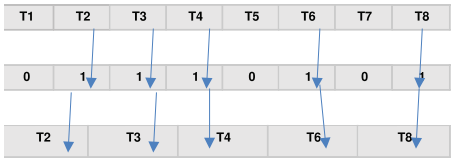


Figura 6: *Ejemplo de representación de un cromosoma de texturas.*

Por ello es necesario recurrir a otras técnicas que nos permitan obtener buenas soluciones en un tiempo razonable.

Además, sería conveniente que el número de texturas incluidas se aproxime lo más posible al número de lados. Así para dos o más posibles soluciones cuya suma se corresponda con la longitud global, se elegiría aquella con un número de texturas más parecido al número de lados del edificio, lo que evita un número excesivo de texturas para un mismo lateral.

La *representación* elegida es una codificación discreta, y en particular binaria, que hace corresponder un gen a una textura. Para ello partimos de un conjunto de las n texturas disponibles, $T = t_1, t_2, ..., t_n$. A cada una de ellas les vamos hacer corresponder un gen de un cromosoma de tamaño n, de forma que el gen de la posición j referencie a la textura en la posición j. La codificación binaria del cromosoma implica que un gen tomará un valor verdadero ó 1 si la textura correspondiente a dicho gen se ha elegido para ser utilizada en un edificio; el valor falso ó 0 significa lo contrario. Por tanto, el *genotipo* es un vector binario y el `fenotipo` la secuencia de imágenes seleccionadas para formar parte de las texturas de un edificio. El proceso de texturización de una manzana de edificios implica la ejecución de un algoritmo genético y la longitud del cromosoma será variable y dependerá del número de texturas pre-seleccionadas de la base de datos. Esta representación en sí misma garantiza que no se repitan texturas en los bajos de edificios, puesto que cada imagen tendrá asignada solamente una única posición o alelo. En la Figura 6 se representa un cromosoma de tamaño 8 en el que se seleccionan los alelos asociados a 5 texturas.

Se ha utilizado una *inicialización* de la población aleatoria, de forma que exista una componente mayor de probabilidad asociada a cada solución y no se predisponga la utilización de unas texturas frente a otras. Así, inicialmente se realiza una consulta a la base de datos que obtiene las imágenes que podrían asociarse a una determinada manzana, basándose en características como la zona donde esté localizado. Una vez determinado el tamaño del cromosoma, el valor de los alelos será 0 ó 1 con igual probabilidad.

Por las características especiales del problema, se han diseñado dos *operadores de mutación* que se aplican con distinta probabilidad. El primero de ellos es característico de la representación binaria y realiza la modificación de un gen con una determinada probabilidad o índice de mutación, que está previamente establecido. El segundo, en cambio, se ha introducido para generar una mayor variabilidad en la población y evitar así óptimos locales. Consiste en buscar, para cada cromosoma, la mayor textura disponible que podría incluirse en la longitud aún no utilizada. Por ejemplo, si el perímetro de una manzana es 150 metros y un cromosoma almacena una solución de 145 metros, se buscaría la textura mayor disponible para incluirla en el hueco de 5 metros que quedaría libre. Así, si existiese una imagen equivalente a 4,5 metros y otra a 6, se elegiría la primera, quedando finalmente un espacio libre de 0,5 metros. En algunas ocasiones, este operador de mutación podría incluso obtener una solución óptima, si dicha textura existiese. En cualquier caso, se amplía el espacio de búsqueda y se incrementa la posibilidad de obtener una mejor solución que no sea un óptimo local. Para favorecer la generación de soluciones aleatorias, la probabilidad de ocurrencia del primer operador de mutación se mantiene constante con un valor de 0,05 durante toda la ejecución, mientras que el segundo se incrementa progresivamente en un factor de 2 cada 500 iteraciones, siendo el valor inicial de 0,001. De esta forma inicialmente se da más importancia a la generación aleatoria de nuevos cromosomas y posteriormente se intenta maximizar la calidad de las soluciones obtenidas tras la ejecución del operador de cruce y de la primera mutación.

El *operador de cruce* funciona generando dos hijos a partir de dos padres, intercambiando el valor de los alelos de una forma no determinística. Así, para cada posición, se generará un número aleatorio y dependiendo del valor del mismo, el alelo se incluirá en uno u otro hijo. La elección de los padres para la reproducción se lleva a cabo mediante la estrategia de selección por torneo siendo tres el tamaño del mismo. Para incluir un mayor factor de aleatoriedad se elige el mejor o el peor de ellos con igual probabilidad. De esta forma no se favorece siempre a los mejores individuos y se permite la opción de reproducirse a cromosomas con peores valores de calidad, que sin embargo pueden incluir genes útiles para nuevas soluciones. De este modo se favorece la variabilidad y el no estancamiento en un óptimo local.

La *función de evaluación* es uno de los aspectos más importantes en el algoritmo propuesto ya que establece un orden y una medida de calidad en los cromosomas disponibles para incluirlos o no en las siguientes generaciones. Además, es una de las diferencias fundamentales con otros algoritmos que pudieran resolver el problema como los de fuerza bruta o Greedy, ya que, en cierta forma, cuantifican los requisitos cualitativos que se han descrito en la especificación del problema.

Para el caso de las texturas inferiores, la función de evaluación va a depender principalmente de que la suma de las longitudes de todas las texturas se aproxime en mayor grado al perímetro total de la manzana. Por tanto, la suma de

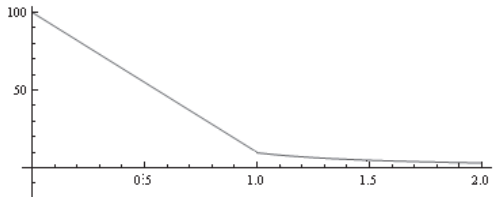


Figura 7: *Función de evaluación:* $f_1(d)$.

los lados del polígono que representa al edificio e, $P(e)$ y la suma de las anchuras del conjunto de texturas seleccionado debe ajustarse lo más posible: $P(e) \approx \quad ancho(t_i)$, para cada textura t_i, $i \in [1...m]$. La función tomará valores entre 0 y un valor máximo *Maximum* previamente establecido ($Maximum = 100$ para este caso). El valor d es la diferencia entre el perímetro del polígono y la suma de las anchuras de las texturas, $d = P(e) - \quad ancho(t_i)$.

La función de evaluación tomará el valor máximo cuando $d = 0$, es decir, cuando la suma de las texturas coincida con el perímetro del polígono (ver Figura 7).

$$f_1(d) = \begin{cases} Maximum & \text{para } d = 0 \\ -90d + Maximum & \text{para } 0 < d \leq 1 \\ (1/d) * Maximum & \text{para } d > 1 \end{cases}$$

Además de lo anterior, para que la visualización sea realista sería conveniente hacer coincidir el número de lados de un polígono con el número de texturas seleccionadas. Por esta razón se ha diseñado una segunda función de evaluación que cuantifica la calidad de la solución alcanzada en lo relativo a este aspecto. En este caso, x se define como la diferencia entre el número de texturas y el número de lados del polígono, $x = numeroTexturas - numero\_Real\_Lados\_Poligono$ y la función toma el valor máximo, *Maximum* si x es igual a cero. A partir de aquí, la función será inversamente proporcional a esta diferencia.

$$f_2(x) = \begin{cases} Maximum & para \quad x = 0 \\ (1/x) * (Maximum - 10) & para \quad x > 0 \end{cases}$$

Para obtener el resultado total es necesario ponderar ambas partes de la función. Como las dos funciones toman valores entre 0 y Maximum la ponderación debe estar también en dicho intervalo: $[f_{total} = f_{1*\mu_p} + f_2 * (1 - \mu_p)]$, siendo μ_p el factor de ponderación de la primera función. Como ésta es fundamental, se le ha dado un mayor peso que a la segunda con un valor $\mu_p = 0{,}6$.

Como *estrategia de reemplazamiento* se opta por un modelo generacional que crea una nueva población completa de individuos en cada iteración sustituyendo a la anterior. Se ha optado por utilizar el elitismo para mantener al mejor individuo de cada generación en la siguiente. Esto evita desechar una buena solución y conseguir el objetivo de encontrar la solución que mejor se ajuste a cada manzana,

Finalmente la *condición de parada* consiste en ejecutar el algoritmo un número predefinido de iteraciones, lo que garantiza obtener un resultado en un tiempo razonable.

Texturas superiores:

El problema de las texturas superiores, aunque análogo al de las inferiores, se parece más al problema de la mochila, un problema de maximización resuelto normalmente con algoritmos voraces [MT08]. El objetivo consiste en maximizar la suma del tamaño de las texturas con el perímetro de la manzana. Las peculiaridades de este problema hacen que el algoritmo genético implementado deba ser diferente al anterior. Existe una restricción clara para este caso: el número exacto de texturas que deben incluirse viene determinado por el número de texturas de portales obtenido en la solución del algoritmo anterior. Este dato proviene del SIG $2D$ y se ha tenido en cuenta en la texturización de los bajos de edificios, por lo que existe un factor importante que hay que considerar: el número de fachadas diferentes que existen debe coincidir con el de portales. No sería realista tener en la parte inferior dos entradas y en la superior cuatro fachadas diferentes. De forma excepcional podría ocurrir que no existieran texturas suficientes para rellenar el perímetro completo de la manzana, con la necesidad de repetir texturas. Esto no suele presentar problema alguno puesto son que habituales las diversas particiones verticales de fachadas en la base de datos de texturas aleatorias.

Considerando los requerimientos anteriores, la representación binaria no resultaría válida, ya que no permite la duplicidad de las texturas. Por esta razón se ha optado por una *representación entera* en la que la posición 0 indica cuántas veces aparece la textura t_0, la posición 1 el número de duplicaciones de la textura t_1 y así sucesivamente. Se ha establecido un máximo de repeticiones con el objetivo de favorecer la inclusión de texturas más amplias en lugar de clonaciones de otras mas estrechas. Se evita así además una excesiva duplicación que podría disminuir la percepción de realismo al aparecer, por ejemplo, cinco veces seguidas la misma ventana abierta o el mismo balcón con la misma decoración. Por tanto, se permite la duplicidad pero se consideran mejores las soluciones con texturas más amplias.

Tal y como se ha comentado anteriormente, para que el genotipo de un cromosoma con esta representación sea válido es necesario que el número de genes con un valor distinto de cero sea igual al número de portales obtenidos. Esta característica es fundamental y debe tenerse en cuenta en el diseño de la inicialización, así como de los operadores genéticos que deben producir siempre individuos válidos.

La *inicialización* es aleatoria, al igual que en el algoritmo anterior. Cada alelo incluye un valor entre cero y el máximo de repeticiones establecido, siendo al final el número de alelos distintos de cero igual al número de texturas que deben incluirse en la solución.

En cuanto al *operador de cruce*, a partir de dos padres se generan dos hijos de forma determinista, incluyéndose los alelos con valor distinto de cero en uno u otro hijo alternativamente. De esta forma, se consiguen nuevas combinaciones de material genético y se introduce variabilidad en la población. La elección de los padres para la reproducción se lleva a cabo, al igual que en el algoritmo para las texturas inferiores, mediante la estrategia de selección por torneo, siendo tres el tamaño del mismo. Se ha incluido un mayor factor de aleatoriedad eligiendo el mejor o el peor de ellos con igual probabilidad. De esta forma se favorece no sólo a los mejores individuos sino también a cromosomas con peores valores en su función "fitness" que pueden incluir genes útiles en nuevas soluciones. Esta estrategia disminuye la probabilidad de un estancamiento en un óptimo local y permite una mayor variabilidad.

Además se han diseñado dos *operadores de mutación* que se aplican con distinta probabilidad. El primero de ellos consiste en el intercambio de dos posiciones aleatorias del cromosoma y el segundo en incrementar el número de veces que se ha incluido una textura en la solución. Es decir, si el alelo que ocupa la posición cinco tiene un valor 3, tras la ejecución de este operador de mutación tendría 4. En el caso de que se superase el máximo establecido de duplicaciones posibles, el valor se consideraría como cero. De la misma manera que en el primer algoritmo descrito, para favorecer la generación de soluciones aleatorias, la probabilidad de ocurrencia del primer operador de mutación se mantiene constante con un valor de 0,05 durante toda la ejecución mientras que el segundo se incrementa progresivamente en un factor de 2 cada 500 iteraciones, siendo el valor inicial de 0,001.

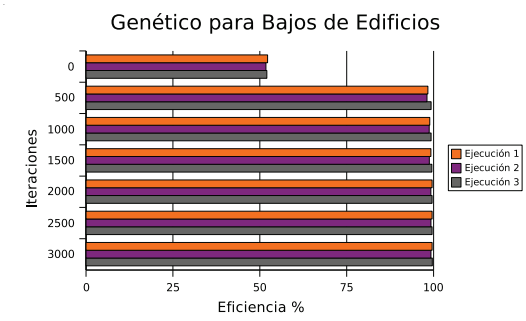
La función de evaluación en este caso depende, principalmente, de que la suma de las longitudes de todas las texturas se aproxime en mayor grado al perímetro total de la manzana. No obstante, también se consideran mejores las soluciones con un menor número de texturas, es decir, se prefieren cromosomas con texturas más anchas que se repiten menos veces. Por tanto, la función de evaluación dependerá de dos parámetros, del número de texturas y de la suma de las longitudes. El primero determina si la solución es válida o no, puesto que si el número de texturas no coincide con el valor previamente establecido por el genético de la zona inferior la función devolverá cero. Una vez comprobado este caso, tendríamos una función lineal en base a la suma de las anchuras de las texturas, que tomará el valor máximo cuando $d = 0$ y un valor inversamente proporcional a la distancia si d es mayor que 1.

$$f(nT,d) = \begin{cases} 0 & \text{si nT} \neq \text{nTN} \\ -90d + Maximum & \text{para } 0 \leq d \leq 1 \\ (1/d) * Maximum & \text{para } d > 1 \end{cases}$$

siendo, nT el número de texturas existentes y nTN el número de texturas necesarias.

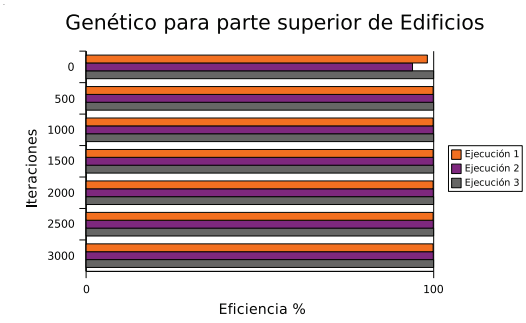
La *estrategia de reemplazamiento* es un modelo generacional que en cada iteración crea una nueva población completa de individuos que sustituye a la anterior. Como el objetivo es conseguir la solución que mejor se ajuste a cada manzana, el mejor individuo de cada generación se mantiene en la siguiente, evitando de esta forma de elitismo descartar una buena solución.

Finalmente, la *condición de parada* vuelve a ser un número predefinido de iteraciones para acotar el tiempo de ejecución. Como veremos en los resultados, el objetivo se cumple a pesar de considerar un criterio de parada cuantitativo.



	0	500	1000	1500	2000	2500	3000
1	52.24	98.37	98.91	99.23	99.59	99.59	99.59
2	51.77	98.11	98.74	98.78	99.22	99.24	99.24
3	52.02	99.29	99.29	99.5	99.57	99.57	99.63

Figura 8: *Resultados del Genético generacional de los bajos de los edificios.*



	0	500	1000	1500	2000	2500	3000
1	98.18	99,84	99,84	99,84	99,84	99,84	99,84
2	93,92	99,89	99,89	99,89	99,89	99,89	99,89
3	99,99	99,99	99,99	99,99	99,99	99,99	99,99

Figura 9: *Resultados del Genético generacional de las partes superiores de los edificios.*

3. Resultados experimentales

La implementación de los algoritmos genéticos de texturización anteriormente descritos se ha realizado en Java utilizando la biblioteca JGAP: Java Genetic Algorithms Packa-

ge [Mef], incorporando los principios descritos en el apartado 2.4.

En las Figuras 8 y 9 se representan los resultados de la función de evaluación tras la ejecución de los dos algoritmos genéticos, el correspondiente a texturas inferiores y superiores respectivamente para una manzana concreta. Aunque dos genéticos con representación y codificación diferentes no son comparables, observamos que el método aplicado a las texturas superiores tiene una convergencia más rápida que el de bajos comerciales. Esto se debe a que para las texturas inferiores cualquier combinación de alelos en un cromosoma es válida, existiendo únicamente una penalización en el valor de la función de evaluación, pero sin llegar a descartarla porque puede aportar información útil para otras generaciones. Sin embargo, en las texturas superiores el algoritmo es diferente ya que, dentro del espacio posible de búsqueda, existen muchas soluciones inválidas que no son consideradas en posteriores iteraciones. La propia función "fitness" implementada refleja la restricción del número de texturas que debe tener la solución, eliminándose los individuos que no la cumplan. Esta condición es determinante para reducir el tamaño de la población que puede ser tenida en cuenta durante la ejecución del algoritmo, lo que conlleva que sea necesario un menor número de iteraciones para obtener una solución adecuada. El valor de la función de evaluación en cualquiera de los dos casos es un valor muy próximo al 100 %.

En las Figuras 10 y 11 puede verse el resultado de la texturización automática sobre una manzana de edificios. Mientras que el algoritmo genético de los bajos de los edificios no permite repetición de texturas, el de texturización superior finalmente sí ha necesitado repetir alguna sin que sea apreciable a simple vista.

Figura 10: *Vista genérica del texturizado.*

Figura 11: *Vista a nivel peatonal del texturizado.*

4. Conclusiones y trabajos futuros

Como conclusión final de este trabajo podemos indicar que los resultados de la aplicación de algoritmos genéticos al proceso de texturización se adecuan en gran medida al problema planteado inicialmente. Los resultados de ejecución del genético se pueden considerar muy eficientes por la aproximación del ajuste a casi el 100 %, al igual que los resultados visuales. Esta metodología es extensible a cualquier otra aplicación de selección de texturas donde se necesite tener en cuenta parámetros cuantitativos (tamaños de texturas) o cualitativos (texturas más adecuadas que otras). El ciclo de vida de este proyecto en el que se irían incrementando el número de texturas reales es perfectamente compatible con el proceso de texturización automática de las zonas no conocidas.

Aunque la solución propuesta no obtenga los resultados en tiempo real, esto no supone ningún inconveniente puesto que en el proyecto completo la texturización forma parte del pre-procesamiento global de los datos. Realmente la asignación se realizaría únicamente una vez, guardándose las asignaciones obtenidas en la base de datos. Como se deduce de lo anterior, el propósito principal del algoritmo es obtener resultados realistas sin que el tiempo de cómputo sea un factor fundamental. Además, por la propia naturaleza del método implementado sería posible utilizar algún sistema distribuido para disminuir el tiempo de ejecución.

La división de las texturas en bajos comerciales y fachadas permite adaptar mejor el proceso de texturización a ciertas zonas como, por ejemplo, las comerciales. En este caso nuestro método permite incluir una amplia variabilidad de negocios para una misma manzana, concretamente tantas como texturas de comercios que se ajusten dispongamos en la base de datos. Otro ventaja adicional del algoritmo propuesto es que el posible número de combinaciones entre fachadas

superiores e inferiores es muy elevado, lo que supone que la probabilidad de que dos texturas similares aparezcan cerca en una misma zona es muy baja. Esto nos permite crear barrios heterogéneos sin un elevado coste adicional, lo que favorece el realismo de la escena (no sería normal, por ejemplo, encontrarnos el mismo bajo comercial en dos manzanas cercanas).

Una posible mejora del sistema sería un proceso de asignación de texturas mixtas, consistente en pasear por la ciudad de Jaén llevando en mano un dispositivo móvil tipo PDA conectado a una base de datos genérica de texturas. El paseante se sitúa delante de un edificio, lo localiza dentro del plano real de la ciudad y adjudica una textura de las disponibles que se parezca a la real; por ejemplo, "edificio de ladrillo rojo sin balcones". En ese instante el servidor toma nota de esta asignación, utilizando un algoritmo similar al anterior para realizar el ajuste final.

5. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia y Tecnología de España y por la Unión Europea por medio de los fondos FEDER con el proyecto TIN2004-06326-C03-03, y también por la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía con los proyectos P07-TIC-02773 y P06-TIC-01403.

References

[AD97] ASHLOCK D., DAVIDSON J.: Genetic algorithms for automated texture classification. In *Proceedings of SPIE* (1997), pp. 140–151.

[And04] ANDREW A. M.: Introduction to evolutionary computing. *Robotica 22*, 3 (2004), 349–349.

[CBSF07] COELHO A., BESSA M., SOUSA A. A., FERREIRA F.Ñ.: Expeditious modelling of virtual urban environments with geospatial l-systems. *Comput. Graph. Forum 26*, 4 (2007), 769–782.

[CdSF03] COELHO A. F. F., DE SOUSA A. A., FERREIRA F.Ñ.: 3d modelling of large urban scenes from diverse sources of information. In *ELPUB* (2003).

[CLRS01] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C.: *Introduction to Algorithms*, 2 ed. The MIT Press, September 2001.

[DLW01] DANIEL L., LOREE P., WHITENER A.: *Inside MapInfo Professional*. OnWord Press, Santa Fe, 2001.

[EDA96] ENGEBRETSON C. J., DAVIDSON J. L., ASHLOCK D.: Genetic algorithms for texture model identification and synthesis. *Statistical and Stochastic Methods for Image Processing 2823*, 1 (1996), 20–31.

[ES03] EIBEN A. E., SMITH J. E.: *Introduction to Evolutionary Computing*. Natural Computing. Springer, November 2003.

[GJ79] GAREY M. R., JOHNSON D. S.: *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd, January 1979.

[Mef] MEFFERT K. E. A.: Jgap - java genetic algorithms and genetic programming package.

[MT08] MARTELLO S., TOTH P.: *Knapsack Problems: Algorithms and Computer Implementations (Wiley Interscience Series in Discrete Mathematics and Optimization)*, revised ed. John Wiley & Sons Inc, 2008.

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., VAN GOOL L.: Procedural modeling of buildings. *ACM Trans. Graph. 25*, 3 (2006), 614–623.

[MZWVG07] MÜLLER P., ZENG G., WONKA P., VAN GOOL L.: Image-based procedural modeling of facades. *ACM Trans. Graph. 26*, 3 (2007), 85.

[QY02] QIN X., YANG Y.-H.: Estimating parameters for procedural texturing by genetic algorithms. *Graph. Models 64*, 1 (2002), 19–39.

[ROOF09] ROBLES-ORTEGA M. D., ORTEGA L., FEITO F.: An exact occlusion culling method for navigation in virtual architectural environments. In *SIACG'09, IV Simposium Iberoamericano de Computación Gráfica* (2009), pp. 23–30.

[Wan07] WANG J.-W.: Self-eigenroughness selection for texture recognition using genetic algorithms. In *ACIVS* (2007), pp. 849–854.