# Seguimiento de óptimos en movimiento a través de predicciones basadas en filtros de Kalman

Claudio Rossi and Antonio Barrientos

*Resumen*— **Los problemas de optimización dinámica tratan con entornos variables. En este artículo se presentan varias técnicas para integrar información de movimiento en un algoritmo genético, cuando la función de fitness cambia con el tiempo y los cambios en el óptimo siguen una ley no basada en el azar. Se propone un mecanismo para estimar dicha ley con el fin de mejorar la capacidad de seguimiento del optimo, que finalmente se prueba con una aplicación de seguimiento de objetos en movimiento para un sistema de visión artificial.**

*Palabras clave*— **Algoritmos Evolutivos, seguimiento basado en visión, problemas de optimización dinámica.**

## I. Introducción

Las aplicaciones en el mundo real a menudo tratan con un entorno variable y en presencia de ruido.

Los Algoritmos Evolutivos (EAs) son conocidos por funcionar bien en este tipos entornos. Las nuevas soluciones candidatas, generadas en cada paso, pueden ser evaluadas y seleccionadas a través de funciones de objetivo (*fitness* en adelante) que cambian en el tiempo. Además, ya que las soluciones tienen tendencia a estar aglomeradas alrededor del optimo, hay una buena probabilidad de que cuando el óptimo se mueva, uno de los individuos de la población se encontrará cerca del nuevo óptimo y constituirá una buena solución del problema.

En los últimos años ha habido un interés creciente en los problemas de optimización dinámica, la cual estudia los problemas que varian en el tiempo. En este campo, la mayoría de los trabajos se centran en estudiar técnicas que mantienen un cierto grado de diversidad en la población, con el fin de garantizar un nivel de exploración adecuado, evitando que ésta se concentre demasiado. Una convergencia excesiva podría hacer que los cambios en el entorno pasen desapercibidos para el algoritmo, mientras que una población más esparcida puede adaptarse a los cambios con más facilidad. Otras líneas que se encuentran en la literatura sobre este tema son: el control dinámico de los parámetros [1], [2]; operadores adaptables especializados [3]; control de la población con migraciones [4]; múltiples poblaciones y nichos [6]; enfoques basado en la memoria [7].

Una revisión muy completa de la bibliografía sobre técnicas para abordar problemas de optimización no-estática se puede encontrar en [8].

Departamento de Automática, Ingeniería Electrónica e Informática Industrial, Universidad Politécnica de Madrid, Madrid, 28006, Spain. E-mail: claudio.rossi@upm.es .

Nuestro enfoque es diferente, y se basa en la consideración de que en muchos problemas del mundo real los cambios no son casuales y se pueden predecir. La ley que gobierna dichos cambios es estimada por un mecanismo ad-hoc, que es usado para predecir los estados futuros del sistema dinámico. Esta información se proporciona al EA con el fin de mejorar su capacidad de seguimiento.

Dividimos la búsqueda de un optimo en movimiento en dos fases. En la primera fase, el algoritmo no tiene información sobre el óptimo, y tiene que explorar el espacio para encontrarlo. Cuando el óptimo (o sub-óptimo) es encontrado, empieza la fase de seguimiento, donde el algoritmo tiene que ser capaz de seguir el optimo. Durante esta fase, bajo la hipótesis que los cambios no están basados en el azar, la nueva posición del optimo se encontrará en una vecindad de la posición actual, cuyo radio depende de la velocidad del movimiento. La transición entre búsqueda y seguimiento no está bien definida, y es un aspecto que hay que considerar.

Durante la fase de seguimiento, la ley subyacente del movimiento puede ser estimada mirando a la secuencia de las soluciones que se van produciendo. Esa estimación puede ser utilizada para predecir donde se encontrará el optimo en los pasos futuros, y así ayudar al proceso de seguimiento, focalizando la búsqueda en la región donde se prevé que éste estará.

Con el fin de dirigir la búsqueda, hemos diseñado diferente técnicas. La idea común a todas es generar o privilegiar individuos que están cerca de la posición que se estima más probable. De esta forma, el algoritmo intentará anticiparse al movimiento del optimo y mejorará sus prestaciones.

En concreto, hemos aplicado los algoritmos evolutivos en aplicaciones de visión artificial para el seguimiento de objetivos en movimiento y la navegación relativa en sistemas robóticos. En este tipo de aplicaciones, los datos de entrada contienen mucho ruido, debido a la adquisición de la imagen y a su pre-procesamiento. Los datos pueden ser afectados por la baja calidad de la imagen, condiciones de luz, vibraciones, etc., y deben ser procesados en tiempo real con el fin de estimas la posición y trayectoria del objetivo. Claramente, los objetos no se mueven al azar, si no que siguen leyes físicas bien definidas.

En el resto del articulo se usará el término *movimiento* a nivel de genes, o sea referido al genotipo de la solución

## II. Estimación del Movimiento

En analogía con las leyes de Newton, usamos los términos *posición* y *velocidad*. La *posición* de un óptimo es el cromosoma de la solución candidata correspondiente, o sea un punto $n$-dimensional en el espacio de búsqueda. La *velocidad* de un optimo es un vector $n$-dimensional que contiene el gradiente del cambio de cada gen. En este trabajo consideramos leyes de primer orden:

$$\overline{p}_t = \overline{p}_{t-1} + \overline{v}_{t-1}\Delta t \tag{1}$$

Dada esta definición, el *estado* $S$ de un optimo en el instante $t$ puede ser definido como el conjunto de su posición y su velocidad.

$$S_t = \begin{bmatrix} \overline{p}_t \\ \overline{v}_t \end{bmatrix} \tag{2}$$

En general, el estado de un sistema puede ser observado solo parcialmente. En nuestro caso, lo que se observa es la posición del optimo, proporcionada por el EA. El estado completo puede ser estimado considerando la parte observable y la secuencia de los estados previos, teniendo en cuenta que pueden estar afectados por un cierto error.

Un método muy conocido para llevar a cabo dicha estimación es el *filtro de Kalman* [9]. El filtro de Kalman es un conjunto de ecuaciones matemáticas que proporcionan una forma eficiente para la estimación de un sistema dinámico, dada una secuencia de observaciones y un modelo del sistema. Las observaciones puede contener ruido, y el modelo puede ser una aproximación del modelo real.

Una característica interesante de los filtros de Kalman es que también proporcionan una estimación del estado *futuro* del sistema. Esta estimación es proporcionada por el modelo del sistema, y después es corregida con la observación real, con el fin de dar una estimación más precisa en el siguiente paso. De este modo, el modelo es actualizado de forma continua con el fin de ajustarse a las observaciones. Además, las diferencias entre las predicciones y las observaciones son usadas para calcular la incertidumbre asociada a las predicciones, en forma de (co-)varianzas de error, proporcionado así una medida de la confianza sobre la estimación.

En un instante dado $t$, observada una secuencia de óptimos y sus posiciones

$$\overline{p}_0, \ldots, \overline{p}_{t-2}, \overline{p}_{t-1},$$

(correspondientes al mejor individuo de la generación $0 \ldots t-1$), y con un modelo como (1), podemos tener una estimación de los vectores de posición y velocidad

$$\hat{\overline{p}}_t, \hat{\overline{v}}_t$$

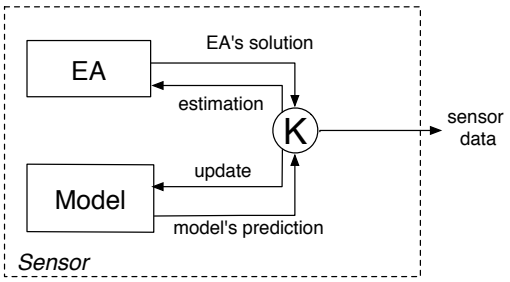


Fig. 1. Mejorar el EA

o sea, una estimación del nuevo estado $\hat{S}_t$, junto a una estimación de su precisión, en términos de varianza:

$$(\hat{\overline{\sigma}}_t^p)^2, (\hat{\overline{\sigma}}_t^v)^2.$$

Para nuestros fines, las estimaciones más importantes son los valores $\hat{\overline{p}}_t$ y $\hat{\overline{\sigma}}_t^p$ que nos proporcionan una indicación de donde es probable que se encuentre el próximo optimo. En el resto del artículo, nos referiremos a ellos como a la *predicción* y su *precisión*. Para mayor claridad, omitiremos los índices y adornos cuando no haya riesgo de confusión.

La Figura 1 ilustra la idea básica de nuestro enfoque: el EA es parte de un sensor, ya que es el EA quien va a proporcionar la medida, y el filtrado es usado con el fin de mejorar la precisión del sensor mejorando el comportamiento del EA.

El coste computacional del filtro de Kalman es esencialmente debido a álgebra de matrices y una inversión de matrices, que tienen coste polinomial, y por lo tanto tienen un bajo impacto en la prestaciones del algoritmo, si asumimos que el coste computacional global está dominado por el coste de evaluar la función de fitness.

En este trabajo, los parámetros del filtro usado han sido calculados durante la puesta a punto del algoritmo y permanecen constantes a lo largo de la ejecución.

## III. Incorporar la Información de Movimiento

A la hora de usar las predicciones hay que considerar ciertas cuestiones.

En primer lugar, *cuando empieza el seguimiento*. Hay que establecer un criterio para decidir en que momento el EA ha encontrado un óptimo satisfactorio para seguirlo (por ejemplo, un umbral prefijado). Además, el estimador necesita tiempo para ajustarse, así que hay que decidir a partir de que momento sus predicciones son fiables. El mecanismo de selección del EA puede ser usado para tratar con ambas cuestiones. Al principio, el estimador proporciona predicciones imperfectas, y los individuos generados usando estas predicciones serán con toda probabilidad sub-optimos, y serán descartados por el mecanismo de selección. Según el EA evolucione, la secuencia de óptimos se tornará más coherente,

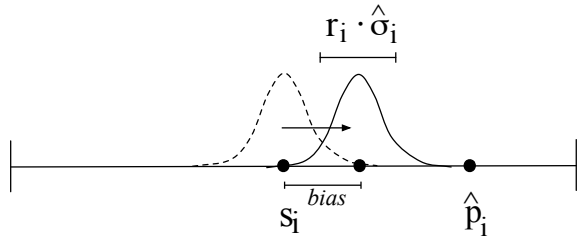


Fig. 2. Perturbación

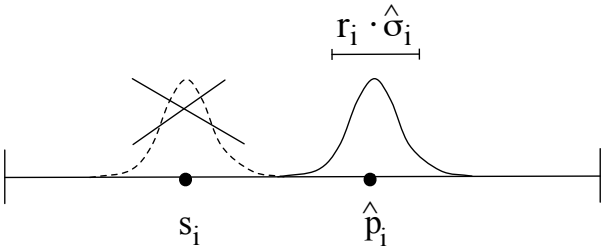


Fig. 3. Nuevo gen random

reflejando el hecho que el algoritmo se acerca al optimo y empieza a seguirlo. Entonces, las predicciones se harán más precisas y los individuos que las incorporan serán beneficiados.

*El ruido puede afectar las predicciones.* En este caso, la salida del EA a su vez afecta la entrada del estimador. Aunque los estimadores como el filtro de Kalman están diseñados para tratar con ello, no queremos fiarnos completamente de ellos.

*El modelo es aproximado.* Por esta razón, el estimador necesita constantemente nuevas observaciones para adaptarse a la ley del movimiento real. Además, sin actualizaciones constantes, el estimador no podría darse cuenta de cambios en la trayectoria del optimo.

*Hay que buscar mejores soluciones.* El algoritmo podría estar persiguiendo una solución no optima. Incluso durante el seguimiento, podrían existir soluciones mejores en un entorno cercano o en regiones inexploradas. Por lo tanto, hay que proporcionar al algoritmo la capacidad de alcanzar estas soluciones, y no solo focalizarse en el optimo actual. En otras palabras, mientras el estimador ayuda la explotación, un cierto grado de exploración tiene que ser garantizado.

Con el fin de proveer la información del estimador al algoritmo evolutivo, se pueden diseñar tres técnicas.

### *A. Operadores genéticos modificados*

#### A.1 Mutación

El operador de mutación modifica uno o más genes del individuo seleccionado. Esta modificación puede ignorar el valor anterior (*nuevo gen random*), o modificar el valor original (*perturbación*). La operación más sencilla es añadir o restar una pequeña cantidad. En lugar de usar cantidades aleatorias ("random"), se puede usar la predicción para dirigir la exploración hacia la región donde se espera encontrar el nuevo óptimo, según el estimador (Fig. 2).

Si $s$ es el cromosoma seleccionado para la mutación, $s'$ el cromosoma después de la mutación, e $i \in 1 \dots n$ el gen seleccionado para ser modificado (donde $n$ es el numero de genes, o sea la longitud del cromosoma). Sean $\hat{p}$ y $\hat{\sigma}$ la predicción y su precisión.

En nuestro caso, la perturbación debería generar valores de $s'_i$ cerca de la estimación $\hat{p}_i$, dependiendo de la precisión $\hat{\sigma}_i$, y en un rango que también dependa de ella. Una estrategia de perturbación que tiene en cuenta estos factores tiene la forma siguiente:

$$s'_i = s_i + pert(r_i, \hat{p}_i, \hat{\sigma}_i). \tag{3}$$

Proponemos la siguiente función heurística:

$$pert(r_i, \hat{p}_i, \hat{\sigma}_i) = b_i \cdot \tfrac{1}{1+\hat{\sigma}_i} \cdot (\hat{p}_i - s_i) + N(0, r_i\hat{\sigma}_i),$$
$$0 < b_i < 1 + \hat{\sigma}_i, \quad r_i \geq 0 \tag{4}$$

El primer término es una *corrección de dirección* (Fig. 2), que determina el desplazamiento deseado hacia la predicción. Una mala predicción produce una baja corrección: si la predicción no es precisa, no tiene que ser tenida en cuenta. El segundo termino determina el rango de la perturbación, haciéndolo mayor o menor según la precisión de la predicción.

Los parámetros $b_i$ y $r_i$ ajustan los valores. Cuando $b_i = 1 + \hat{\sigma}_i$ se obtiene el desplazamiento máximo, mientras $b_i = 0$ significa ningún desplazamiento. Nótese que si $\hat{\sigma}_i = 0$ y $b_i = 1$ el nuevo gen será exactamente $\hat{p}_i$ (máxima confianza en la predicción).

Un caso especial de este operador se da poniendo el máximo desplazamiento. En este caso, el centro de la perturbación coincide con la predicción, y el valor anterior del gen será ignorado (Fig. 3):

$$s'_i = N(\hat{p}_i, r_i\hat{\sigma}_i). \tag{5}$$

Nos referiremos a este caso como al operador *nuevo gen random*.

#### A.2 Cruce

El operador de cruce busca en el espacio recombinando información de dos individuos para generar nuevos individuos. Este operador no crea nuevo material genético. La única forma de incorporar las predicciones en esta búsqueda es cruzar un individuo de la población con uno nuevo, generado teniendo en cuenta las predicciones (véase la Sección III-C). de este modo, los nuevo individuos podrán incorporar información de la predicción, y la búsqueda será dirigida hacia la región indicada como más probable por el estimador.

### *B. Función de fitness modificada*

Añadir términos a la función de fitness es una práctica común para incorporar información

heurística al algoritmo evolutivo. Aquí, la idea es privilegiar los individuos que estén cerca de la futura posición estimada, con el fin de ayudar al algoritmo a mantenerse cerca del óptimo en movimiento.

Sea $s$ el cromosoma por evaluar y $f$ la función de fitness de base. La función modificada será $h$:

$$h(s) = (1-a)\cdot f(s) + a \cdot r(s) \qquad 0 < a < 1 \quad (6)$$

donde una heurística sencilla es

$$r(s) = \frac{1}{1+\hat{\sigma}_{max}} \cdot |\hat{p} - s|$$

donde $\hat{\sigma}_{max} = ||\overline{\hat{\sigma}}||_\infty$ es el máximo de los $\hat{\sigma}_i$. El termino de refinamiento $r(s)$ tiene en cuenta la distancia desde la futura posición estimada, y lo arregla con la confianza sobre la predicción El parámetro $a$ es usado para ajustar la importancia relativa de los dos términos.

### C. Individuos dotados

Esta técnica consiste en generar un cierto número de individuos nuevos usando la información del estimador. En principio, si el estimador fuese perfecto, se podría generar un solo individuo $s = \hat{p}$, y este sería el nuevo optimo. En general, el nuevo optimo probablemente se encontrará en una región alrededor de la predicción $\hat{p}$, cuyo tamaño es estimado por $\hat{\sigma}$. Por lo tanto, la idea es generar unos cuantos individuos en dicha región.

Sea $P$ el tamaño de la población, y $0 < g < 1$ la proporción de población que queremos que sea compuesta por nuevos individuos "dotados". Un numero de $\lfloor gP \rfloor$ individuos serán generados alrededor de la predicción ($s_i = N(\hat{p}_i, \hat{\sigma}_i),\ i = 1 \ldots n$) y los otros $P - \lfloor gP \rfloor$ de la forma habitual. El caso $g = 0$ significa rechazar la predicción, el caso $g = 1$ significa rechazar la vieja población y generar una enteramente nueva alrededor de $\hat{p}$.

Una sencilla heurística es tal que $g$ es inversamente proporcional a $\hat{\sigma}_{max}$: cuanto menos confianza tenemos en la predicción, menos individuos generamos a su alrededor.

$$g = \frac{c}{1+\hat{\sigma}_{max}}, \qquad 0 < c < 1 + \hat{\sigma}_{max} \qquad (7)$$

## IV. El Problema del *Pose*

Con el fin de probar las técnicas descritas en la sección anterior, se han hecho experimentos con un problema conocido como *pose*.

Se considera una escena de un mundo en tres dimensiones, en donde un objeto es visto por una cámara proyectado en un plano bidimensional (Fig. 4). $T_x$, $T_y$ y $T_z$ son los valore de traslación con respecto a la cámara (posicionada en el origen de los ejes), y $\alpha$, $\beta$, $\gamma$ son los ángulos de rotación respecto a los ejes.

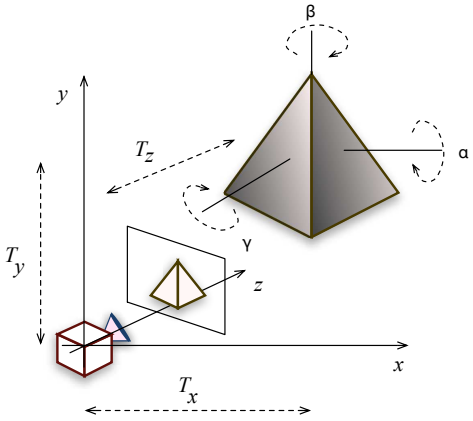


Fig. 4. Geometría del problema

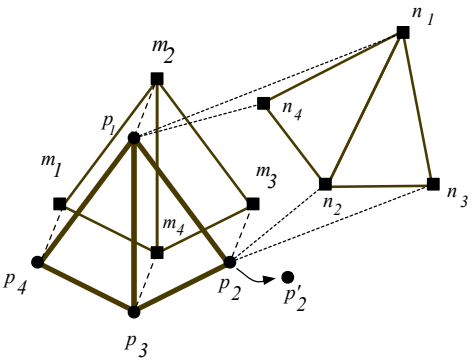


Fig. 5. Distancia y puntos coincidentes

Dada una imagen bidimensional de un objeto, el problema de la estimación del pose consiste en encontrar la posición y orientación del objeto en el espacio. Los métodos basados en modelos usan un modelo del objeto buscado y buscan una correspondencia entre algunas de sus características como esquina o marcas al contenido de la imagen. Un enfoque común es dividir la búsqueda en dos pasos: primero, las características son extraídas de la imagen, y después se busca una correspondencia.

### A. Estimación Evolutiva del Pose

El problema del pose puede ser formulado como un problema de optimización de la siguiente forma [10].

Se tienen que encontrar los valores de los seis parámetros $(T_x, T_y, T_z)$ y $(\alpha, \beta, \gamma)$ (Fig. 4). Si estos parámetros fuesen conocidos, pongamos que sean $s^o = (T_x^o, T_y^o, T_z^o, \alpha^o, \beta^o, \gamma^o)$, rotando y trasladando el modelo del objeto y proyectándolo en el plano de la cámara, obtendríamos una correspondencia perfecta entre las características del objeto y de la imagen. En este caso, la suma $d$ de las distancias entre cada pareja de puntos correspondientes sería nula.

Si un parámetro fuese distinto del valor correcto, por ejemplo $T_x = T_x^o + \epsilon$, cada punto $m_i$ del modelo estaría desplazado en una cantidad $f_i(\epsilon)$ con respecto a los puntos de la imagen, siendo $f_i$ funciones que dependen de la geometría y de las característi-

cas de la cámara. La suma de las distancias sería en este caso $d = \sum_i |f_i(\epsilon)|$. Cuanto más cerca esté la proyección $s = (T_x, T_y, T_z, \alpha, \beta, \gamma)$ a la imagen, más pequeña será $d$. Siendo $s^o$ desconocida, una forma de calcular $d$ es medir la distancia entre los puntos de la imagen y los puntos del modelo proyectado, y buscar una solución $s$ tal que $d$ sea mínima. El problema de la correspondencia es por lo tanto transformado en un problema de optimización:

$$min(d = \sum_i |p_i - m_i(s)|),$$
$$p_i, m_i \in \mathbb{R}^2 \quad (8)$$
$$s = (T_x, T_y, T_z, \alpha, \beta, \gamma)$$

donde $p_i$ es un punto en la imagen, $m_i(s)$ es el punto del modelo proyectado según los parámetros $s = (T_x, T_y, T_z, \alpha, \beta, \gamma)$ y $||$ es la distancia Euclidea. Un punto de la imagen se considera correspondiente a un punto del modelo si es el más cercano. Así la distancia finalmente será:

$$d = \sum_i min_j(|p_i - m_j(s)|) \quad (9)$$

La Figura 5 muestra un ejemplo donde dos conjuntos de parámetros $s^M$ y $s^N$ producen dos conjuntos de puntos proyectados $M = \{m_1, m_2, m_3, m_4\}$ y $N = \{n_1, n_2, n_3, n_4\}$. Las correspondencias entre puntos es representada por líneas discontinuas. La distancia entre los puntos del conjunto $M$ y los puntos de $P = \{p_1, p_2, p_3, p_4\}$ es claramente inferior de la distancia total de los puntos $N$ de $P$. Por lo tanto, el conjunto $M$ se corresponde mejor, y el conjunto de parámetros $s^M$ es la solución mejor.

Es fácil ver como en el caso de que los puntos extraídos de la imagen presenten alguna imprecisión, la distancia mínima $d$ no será nula. Esto puede afectar a la solución óptima, dependiendo del nivel de ruido presente. Sin embargo, si el ruido es limitado, la distancia mínima se corresponde a un buena solución, ya que el ruido afecta a ambos $s^M$ and $s^N$. Este efecto se muestra en la Figura 5, donde el punto $p_2$ es desplazado a $p_2'$: si bien la distancia total cambia, el conjunto $M$ sigue siendo mejor que el conjunto $N$. Lo mismo ocurre cuando falta algún punto de la imagen, o si algún punto es extraído por error (falso positivo).

De todas formas, la función de distancia puede presentar mínimos locales, sobre todo cuanto hay un elevado número de falsos positivos en la imagen. Por esta razón es importante recurrir a métodos de búsqueda como los algoritmos evolutivos, que pueden evitar los mínimos locales muestreando el espacio de búsqueda.

En el algoritmo *EvoPose* el vector $s$ es una solución candidata, y una buena solución es buscada a través un algoritmo evolutivo.

La complejidad computacional para el calculo de una solución es polinomial con el número de puntos de la imagen $|P|$ y puntos del modelo $|M|$. Por lo tanto, el algoritmo escala bien con la complejidad de las imágenes reales, donde se consideran decenas o incluso cientos de puntos.

### B. Objetos en Movimiento

El problema del seguimiento de un objeto en movimiento consiste en determinar el pose de un objeto en una secuencia de imágenes, donde la posición y/o orientación del objeto pueden cambiar. Una vez que el pose correcto ha sido encontrado, se puede usar esta información para encontrar la posición del objeto en las imágenes que siguen, sin tener que resolver el problema desde el principio otra vez para la nueva imagen

Cuando se trata con imágenes en movimiento, el conjunto de puntos extraídos $P = \{p_i\}$ cambia con el tiempo, o sea $P(t) = \{p_i(t)\}$. Por lo tanto, la función distancia (9) también será función del tiempo:

$$d(t) = \sum_i min_j(|p_i(t) - m_j(s)|) \quad (10)$$

En nuestro problema de búsqueda, esto significa que tenemos una función de fitness que cambia con el tiempo.

Nótese que ya que las nueva soluciones candidatas se generan a partir de la información de las mejores soluciones ya encontradas, la información sobre la posición anterior es efectivamente usada para encontrar la posición en las imágenes que siguen.

## V. Resultados Experimentales

Hemos probado los métodos descritos en la Sección III, usando el algoritmo *EvoPose* y considerando un cubo moviéndose y rotando a distintas velocidades. (Fig. 6, izquierda). El término *velocidad* usado aquí se refiere al incremento $\Delta$ aplicado a cada gen cada generación.

En cada generación, el pose del objeto ha sido actualizado de la forma descrita más abajo, y una nueva imagen ha sido generada. Luego, la imagen ha sido procesada para extraer los puntos de interés (esquinas del cubo), y los puntos usados para calcular la distancia (10). Durante este proceso, puede aparecer ruido, como por ejemplo falsos positivos ( Fig. 6, derecha).

El rango de los genes es $[0, 1]$, mientras el rango del fenotipo es $[0, 500]$ para las posiciones $x$ e $y$ (tamaño de la imagen de $500 \times 500$ pixels), $[0, 1]$ para la distancia $z$, y $[0, 2\pi]$ para las rotaciones. El tamaño del cubo es de 35 pixels. Por ejemplo, un valor de $\Delta = 0{,}001$, significa una velocidad de traslación del cubo de 0,5 pixels/generación y rotación de 0,36 grados/generación.

Cada una de las técnicas ha sido probada con diferentes ajustes de sus parámetros. Por cada ajuste se han ejecutado 10 pruebas. En total, se han probado 204 combinaciones diferentes.

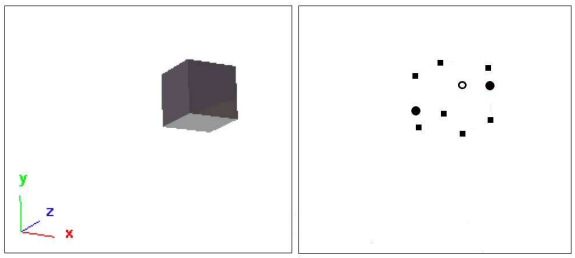


Fig. 6. Una imagen del objetivo (izquierda) y los puntos extraídos por el proceso de extracción de puntos (derecha). En este ejemplo, siete vértices I han sido detectados correctamente (cuadrados), un punto está ausente ya que está escondido (circulo vacío) y han sido detectados dos falsos positivos (círculos llenos)

### A. El Algoritmo Evolutivo

El algoritmo evolutivo usado tenía las siguientes características:

- **Tipo**: generacional. Población: 100 individuos.
- **Selección**: standard roulette.
- **Representación**: vector de 6 genes, números reales en el rango $[0, 1]$.
- **Cruce**: standard crossover de un punto (con excepción del caso SINGLE-XOVER). Probabilidad: 0.5.
- **Mutación**: Perturbación Gaussiana (menos cuando se han usado los operadores (3) y (5)). Probabilidad = 0.9, Valor de $r_i = 0{,}1$ (10 % del rango).

La discusión se centra en la segunda mitad de cada ejecución, que es cuando se supone que el algoritmo ha encontrado el objeto y lo está siguiendo. Para comparar la regularidad del seguimiento, se ha tenido en cuenta la desviación estándar de la distancia a lo largo de la ejecución.

### B. Discusión

En general, las versiones del algoritmo que incorporan las predicciones ha obtenido mejores resultados que el algoritmo original. Esto indica que la información proporcionada por las predicciones ayuda efectivamente el seguimiento, y la posición calculada está más cerca a la posición real del objeto.

En lo que concierne a los parámetros de ajuste, el comportamiento del algoritmo no ha mostrado diferencia significativas. Los peores resultados se han obtenido con valores altos de lo ajustes. Esto parece indicar que mientras las predicciones ayudan el seguimiento, éstas no tienen que ser sobreestimadas. Este comportamiento puede ser explicado por el hecho que las predicciones ayudan en la fase de explotación, pero pueden llevar a una excesiva convergencia de la población.

La comparación entre el mejor ajuste de las diversas técnicas en el caso de movimiento de traslación se enseña en la Figura 7 [1]. Las mejores prestaciones se obtienen con el operador de cruce de un solo padre, que además posee la ventaja de no tener la necesidad de parámetros de ajuste. De todas formas, no hay una clara superioridad de ninguna de las técnicas, ya que las diferencias entre ellas no son significativas. Con velocidades bajas, las diferencias con el algoritmo original son pequeñas, pero las diferencias van apareciendo según la velocidad aumenta.

La Figura 8 muestra la comparación entre la desviación estándar media de la diferencia entre el mejor individuo y la posición real durante una ejecución. Una desviación baja indica menos oscilaciones, o sea un seguimiento más constante.

Los resultados obtenidos para la rotación son parecidos a los de la traslación, ya que otra vez las versiones modificadas del algoritmo mejoran los resultados. Los resultados están resumidos el la Figura 9, que muestra la comparación entre el mejor ajuste de las diversas técnicas y el algoritmo evolutivo sin ellas (indicado como *Plain EA* en las Figuras 7-10).

Encontrar la orientación correcta es un problema más difícil que encontrar la posición. Esto es porqué mientras que en las traslaciones la posición y la distancia relativa entre los puntos no varía, en las rotaciones los puntos pueden cambiar de posición relativa, y la relación entre ellos es de origen trigonométrica. Esto se refleja en las Figuras 9 y 10.

Se puede ver como las mejoras con respecto al algoritmo original son inferiores al caso de los movimientos de traslación, y como la regularidad del seguimiento no mejora si no es en casos puntuales.

## VI. Conclusiones

En este artículo se han presentado diferentes técnicas para dirigir la búsqueda y el seguimiento de un optimo en movimiento en problemas de optimización dinámica, usando información proporcionada por un mecanismo de predicción. El mecanismo de predicción se basa en la hipótesis de que en las aplicaciones del mundo real los cambios no son casuales y se pueden aprender. La idea común en todas las técnicas es la de generar individuos que se encontrarán más cerca a la futura posición estimada, con el fin de ayudar al algoritmo a mantener el paso del optimo en movimiento, anticipándolo.

El mecanismo de predicción que adoptamos está basado en los filtros de Kalman. Aunque el filtrado es una práctica común en las aplicaciones de seguimiento, la forma en que se ha utilizado es diferente. En nuestro enfoque, el filtro es usado con el fin de mejorar el comportamiento del algoritmo, y no como un post-procesado como se hace normalmente.

Los experimentos que se han llevado a cabo usando un conocido problema de visión artificial, indican que las predicciones ayudan al seguimiento del optimo, tanto en términos de distancia media desde la solución real como en términos de fluidez. El meca-

[1] En las figuras, las diversas variantes del algoritmos se indican con: New random gene (mutación con nuevo gen aleatorio), Biased perturbation (mutacióon a través de perturbación), Single-parent crossover (operador de cruce), Refining function (función de objetivo modificada).

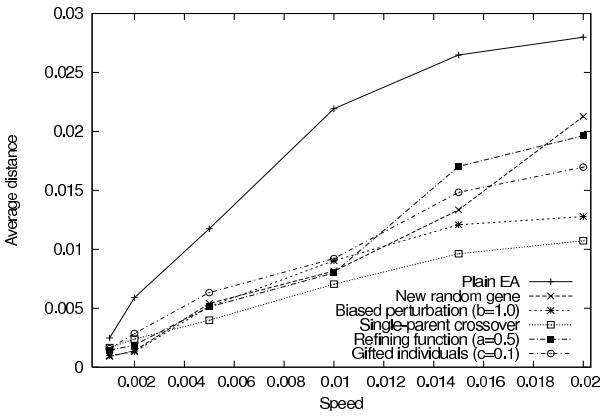


Fig. 7. Translación: comparación de todas las estrategias

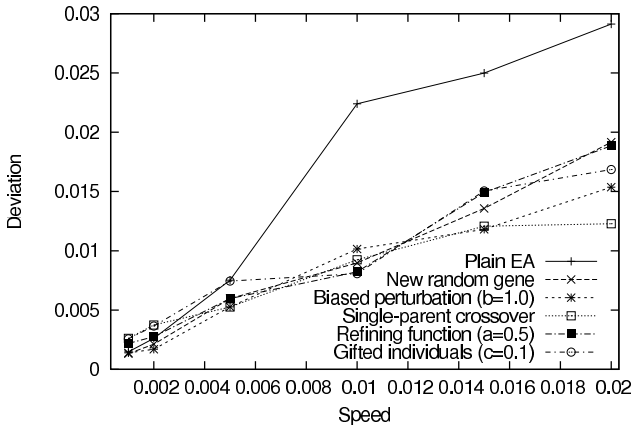


Fig. 8. Translación: desviación estándar media de la distancia

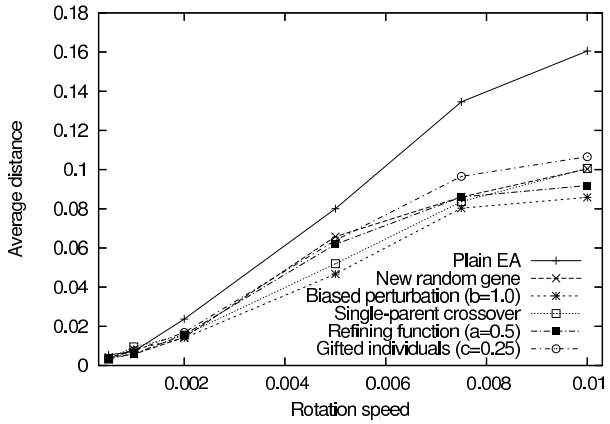


Fig. 9. Rotación: comparación de todas las estrategias

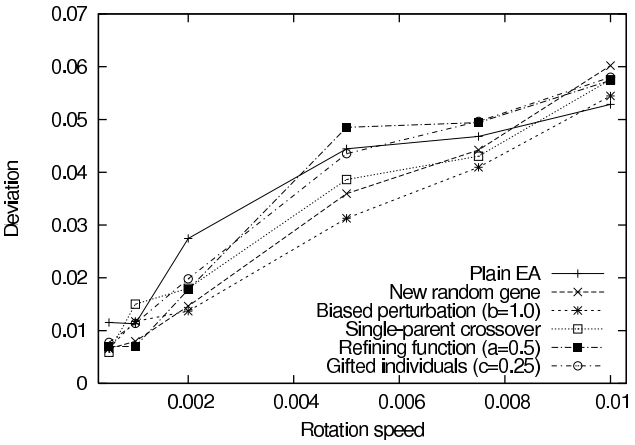


Fig. 10. Rotación: desviación estándar media de la distancia

nismo de aprendizaje y predicción tiene un impacto no significativo en las prestaciones, en cuanto se refiere al coste computacional.

Los experimentos también indican que mientras las predicciones ayudan el seguimiento, no tiene que ser sobreestimadas, ya que pueden llevar a una excesiva convergencia, lo que tiene un impacto negativo sobre el algoritmo.

La principal desventaja de nuestro enfoque es que en algunas de las técnicas propuestas se han introducido unos parámetros de ajuste, lo que puede incrementar la complejidad de ajustar el algoritmo para obtener los mejores resultados. Además, el filtro de Kalman necesita un periodo de ajuste inicial (offline) para obtener prestaciones óptimas.

## Agradecimientos

El primer autor es investigador "Ramon y Cajal". Este trabajo ha sido financiado parcialmente por el Ministerio de Ciencia y Tecnología a través de los proyectos DPI2005-04302 y DPI2006-03444.

## Referencias

[1] Peter J. Angeline, "Tracking extrema in dynamic environments," in *Int. Conf. on Evolutionary Programming*, P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, Eds. 1997, pp. 335–345, Springer Verlag.

[2] T. Bäck, "On the behavior of evolutionary algorithms in dynamic environments," in *IEEE Int. Conf. on Evolutionary Computation*, D. B. Fogel, H.-P. Schwefel, T. Back, and X. Yao, Eds. 1998, pp. 446–451, IEEE Press.

[3] Helen G. Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," Tech. Rep. AIC-90-001, Navy Center for App. Research in Artificial Intelligence, Washington, DC, 1990.

[4] J.J. Grefenstette, "Genetic algorithms for changing environments," in *Parallel Problem Solving from Nature*, R.Männer and B.Manderick, Eds. 1992, pp. 137–144, Elsevier Science Publisher.

[5] J. Branke, T. Kaußler, C. Schmidt, and H. Schmeck, "A multi-population approach to dynamic optimization problems," in *Adaptive Computing in Design and Manufacturing 2000*, 2000, pp. 299–308.

[6] W. Cedeno and V. R. Vemuri, "On the use of niching for dynamic landscapes," in *Congress on Evolutionary Computation*. 1997, pp. 361–366, IEEE Press.

[7] Jürgen Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Congress on Evolutionary Computation*, Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, Eds. 6-9 1999, vol. 3, pp. 1875–1882, IEEE Press.

[8] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments - a survey," *IEEE Trans. on evolutionary computation*, vol. 9, no. 3, pp. 303–317, 2005.

[9] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[10] Rossi, C., Abderrahim, M., and Diaz, J. (2005b). Evopose: A new model-based pose estimation algorithm with correspondences determination. In *IEEE International Conference on Mechatronics and Automation 2005*, volume 3, pages 1551–1556.