En gen\_aleat gen\_aleat\_cuda() no tiene sentido usar memoria compartida, pues

sólo se accede a la matriz una vez (para escribir el valor generado).

En realidad no se optimiza nada usando memoria compartida, dado que los núcleos

o bien no realizan suficientes lecturas o escrituras como para que compense el uso de

memoria compartida, o directamente no usan la memoria global más que para escribir el

resultado (como en gen\_aleat\_cuda). Sin embargo, somo se pide en la práctica, se ha

implementado en los siguientes núcleos:

-> eliminar\_fila\_cuda

-> eliminar\_columna\_cuda

-> girar\_matriz\_cuda

-> buscar\_coinc\_cuda\_fila (aquí se usan dos matrices en memoria compartida)

-> buscar\_coinc\_cuda\_col (se usan dos matrices en memoria compartida)

-> llenar\_vacios\_cuda (es el único núcleo en el que se disminuye realmente el

número de accesos a memoria global)

El único sitio donde sí puede llevar alguna ventaja el uso de memoria

compartida, debido al gran número de accesos, es buscar\_lleno()

Además, se ha usado el desenrrollamiento en el bucle para copiar el cuadrante de

3x3 en la memoria compartida, dentro del núcleo 'girar\_matriz\_cuda'