Chapter 6

Roads/Runway Extraction from Edges

In Chapter 3 and Chapter 4 several methods to extract an accurate edge map from challenging SAR images (equivalent number of looks equal to one) are presented, together with a method to refine and reconstruct missed edges (see Chapter 5). The problem faced in this section is about the linear landmarks extraction (e.g. roads/runways) from a SAR image, given its accurate edge map.

6.1 Line Extraction

The problem presented in this section concerns the pixel grouping in order to recognize linear parts of the detected edges. A common mathematical tool to solve this type of problem is the Hough Transform [52]. The peculiarity of the Hough Transform is that it is able to transform a complex problem, such as finding pixels belonging to a straight line, in an easier task, such as finding local maxima in Hough Transform space. Actually, the Hough Transform is a tool to detect the slope and offset parameters of the lines presented in an image. This is why we need another procedure to transform the parameters detected into vectors which lie over the image pixels, i.e. we need a grouping procedure. In this section both of the previously mentioned procedures are described, but before starting it is necessary to make some preliminary remarks for the sake of clarity.

6.1.1 Hough and Radon Transform: Definition

Given an image $g(x,y)$, the Hough transformed version $G(p,\tau)$ is defined as follows, with p and τ respectively the slope and the offset of a generic line:

$$G(p,\tau) = \int_{-\infty}^{\infty} g(x, px+\tau)dx \tag{6.1}$$

From Eq. (6.1) it is clear how the Hough Transform value at a point (p^*,τ^*) is simply the value of the image integral along the line $y = p^*x + \tau^*$. A way to practically implement this mathematical operator is to discretize the involved variables by quantizing them. In this manner the Eq. (6.1) becomes:

$$G(p_k,\tau_h) = \sum_{m=0}^{M-1} g(x_m, p_k x_m + \tau_h)\Delta x \tag{6.2}$$

where a linear quantization is considered:

$$\begin{cases} x_m = x_0 + m\Delta x & ,m = 0,\cdots,M-1 \\ y_n = y_0 + n\Delta y & ,n = 0,\cdots,N-1 \\ p_k = p_0 + k\Delta p & ,k = 0,\cdots,K-1 \\ \tau_h = \tau_0 + h\Delta\tau & ,n = 0,\cdots,H-1 \end{cases} \tag{6.3}$$

with x_0, y_0, p_0 and τ_0 the initial variable values. Usually, because the Hough Transform version in Cartesian coordinates suffers from the potentially infinite value of the variable p, the version in Polar coordinates shown in Eq. (6.5) is used in literature:

$$G([\rho_r],\theta_t) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g(x_m, y_n)\Delta x\Delta y \tag{6.4}$$

where the following well-known equality is utilized:

$$\rho_r = x_m \cos\theta_t + y_n \sin\theta_t \tag{6.5}$$

with $\theta \in [0,\pi)$ and the square parenthesis $[\cdot]$ indicates the interpolation operator (e.g. nearest neighbour).

In the literature, the terminology related on Hough Transform [52] and Radon Transform [54] can be sometimes misleading. Actually, the difference mainly reside in the domain where the variables interpolation is performed. However, since Radon Transform has got a continue definition and considering the historical point of view, it can be said that the Hough Transform is a particular implementation of the Radon Transform. To explain this concept better, let us introduce the discrete Radon transform:

$$\begin{aligned} G(\rho_r,\theta_t) &= \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g([x_m],[y_n])\Delta y\Delta x \\ &= \sum_{j=0}^{J-1} g\big([\rho_r\cos\theta_t - s_j\sin\theta_t],[\rho_r\sin\theta_t + s_j\cos\theta_t]\big)\Delta s \end{aligned} \tag{6.6}$$

where the following equalities are utilized:

$$\begin{cases} x_m = \rho_r\cos\theta_t - s_j\sin\theta_t \\ y_m = \rho_r\sin\theta_t + s_j\cos\theta_t \end{cases} \tag{6.7}$$

with *s* the axis taken along the line of coordinates (ρ,θ), see Fig. 6.1, and:

$$\begin{cases} x_m = x_0 + m\Delta x & ,m = 0,\cdots,M-1 \\ y_n = y_0 + n\Delta y & ,n = 0,\cdots,N-1 \\ \theta_t = \theta_0 + t\Delta\theta & ,t = 0,\cdots,T-1 \\ \rho_r = \rho_0 + r\Delta\rho & ,r = 0,\cdots,R-1 \end{cases} \tag{6.8}$$

It is totally clear from the comparison between Eq. (6.4) and (6.6) that the main different resides in the domain where the interpolation is performed. In the Hough Transform the variable ρ is interpolated and this means that the interpolation is performed on transformed domain $\Gamma x\Theta$. Instead, in the Radon Transform the interpolation is performed in the image domain XxY because the variables *x* and *y* are discretized.

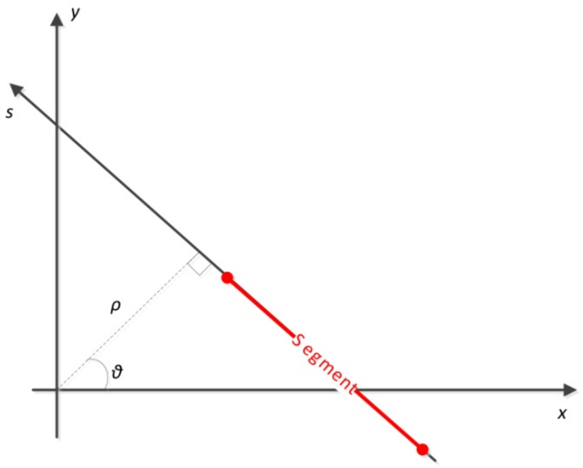


Fig. 6.1 - Reference system used by the Radon Transform

The advantage in rounding on image domain (Radon Transform) is that the integration line can pass through every point inside the pixel (here thought as a square with values in the interval $[x_m - \Delta x/2\,, x_m + \Delta x/2] x [y_n - \Delta y/2\,, y_n + \Delta y/2]$). On the contrary, rounding variables on transformed domain (Hough Transform) forces the line to pass through the centre (x_m, y_n) of the pixels. However, in order to reduce the quantization error only one variable is usually rounded so that Radon Transform can be written as:

$$G(\rho_r, \theta_t) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(x_m, [y_n]) \Delta y \Delta x \tag{6.9}$$

forcing the integration line to pass through the points in $x_m\ x\ [y_n - \Delta y/2\,, y_n + \Delta y/2]$. However, rounding variables on transformed domain (Hough Transform) is expressly suggested with binary images because it allows to avoid summing zero valued pixels and so to save a lot of computation load.

For the sake of completeness, the generalized versions of the Hough and of the Radon Transform are introduced hereafter.

6.1.1.1 Generalized Radon Transform

Let us denote a generic curve as $\phi(x, y; \boldsymbol{v}) = 0$, with $\boldsymbol{v} = (v_1, \cdots, v_\zeta)$ the parameter vector which describe the curve totally (e.g. for the line case $\phi(x, y; \boldsymbol{v}) = y - px + \tau$ with $\boldsymbol{v} = (p, \tau)$). Therefore, the Generalized Radon Transform (GRT) can be written as:

$$G(\boldsymbol{v}) = \int_{-\infty}^{\infty} g\big(x, \phi(x; \boldsymbol{v})\big) dx \tag{6.10}$$

Then, involved variables can be discretized by quantizing them

$$\boldsymbol{v} = f(\boldsymbol{J}) \tag{6.11}$$

where f is the quantization function and $\boldsymbol{J} = (j_1, \cdots, j_\zeta)$ is the index vector (e.g. the linear quantization function is $v_i = v_{i_0} + j_i \Delta v_i$ with $v_i \in \boldsymbol{v}$, $j_i = 0, \cdots, J_1 - 1$ and v_{i_0} the initial value of v_i). The quantized version of the GRT can be written as:

$$G(\boldsymbol{J}) = \sum_{m=0}^{M-1} g(x_m, [\phi(x_m; \boldsymbol{J})]) \Delta x \tag{6.12}$$

6.1.1.2 Generalized Hough Transform

Despite the Radon Transform, the Hough Transform presupposes the function ϕ invertible.

In fact, only in that case, it is possible to retrieve a parameter $v_r \in \boldsymbol{v}$ through the variables x, y, and the remaining parameters $\boldsymbol{v}' \in \boldsymbol{v}/v_r$:

$$v_r = \phi_{v_r}^{-1}(x, y; \boldsymbol{v}') \tag{6.13}$$

Indicating with j_r the index of the parameter v_r and with $\boldsymbol{J}'$ the index vector of the remaining parameters, the quantized version of the Generalized Hough Transform (GHT) can be written as:

$$G([j_r], \boldsymbol{J}') = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} g(x_m, y_n) \Delta x \Delta y \tag{6.14}$$

with:

$$j_r = \left[f_{v_r}^{-1}(v_r)\right] = \left[f_{v_r}^{-1}\left(\phi_{v_r}^{-1}(x, y; \boldsymbol{v}')\right)\right] \tag{6.15}$$

6.2 Hough and Radon Transform: Implementation Problems

In order to not lose any information about image lines, some constraints about the variables involved in both transforms have to be respected. It is worth nothing that the choice of the approximation domain changes the variable constraints heavily. Usually, there are two manners to face this issue in literature. One manner involves the signal processing theory, in which variable constraints are set so that it is possible to recover the original image from the transformed one. The second way is more practical and its main goal is to compute the transformed image avoiding any type of computation error. Surprisingly, even though the second way is much more intuitive than the first one, the constraints are nearly the same. For this reason in this document the practical treating is followed and only the final signal processing results are reported for a clear comparison.

6.2.1 Generalized Radon Transform

As shown in Eq. (6.12), in order to avoid skipping some image points in the GRT computation, for each variation Δv_i of parameters $v_i \in \boldsymbol{v}$, and for each variation Δx of x, keeping constant the other variables each time, the function $\phi(x_m;\boldsymbol{J})$ has to vary less than 1 point. This consideration is translated in the following constraints:

$$\begin{cases}\max|\phi(x_m; v_i+\Delta v_i)-\phi(x_m;\, v_i)| \le \Delta y \quad ,\forall v_i \\ \max|\phi(x_{m+1};\boldsymbol{v})-\phi(x_m;\,\boldsymbol{v})| \le \Delta y\end{cases} \tag{6.16}$$

Moreover, if the function ϕ is differentiable, in order to retrieve the variable constraints more easily, it is convenient to exploit the linear approximation as follows:

$$\begin{cases}\max\left\{\left|\dfrac{\partial\phi(x;v_i)}{\partial v_i}\right|\Delta v_i\right\} \le \Delta y \quad ,\forall v_i \\ \max\left\{\left|\dfrac{\partial\phi(x;v_i)}{\partial x}\right|\Delta x\right\} \le \Delta y\end{cases} \tag{6.17}$$

For the sake of clearness, sometimes the function ϕ is indicated as function of the vector parameter (e.g. $\phi(x_{m+1};\boldsymbol{v})$) and other times with the index parameter (e.g. $\phi(x_m;\boldsymbol{J})$), but in both cases the meaning is the same.

Before showing the precedent constraint results in the line case, another effort needs doing to solve the problem concerning the infinite value within the domain of parameter p (line slope). This problem arises when only one variable is rounded and to solve that in an easy way the following equality can be exploited:

$$G(\rho,\theta)=\frac{1}{|\sin\theta|}\int_{-\infty}^{+\infty} g\left(x,-x\cot\theta+\frac{\rho}{\sin\theta}\right)dx \tag{6.18}$$

and also

$$G(\rho,\theta)=\frac{1}{|\cos\theta|}\int_{-\infty}^{+\infty} g\left(-y\,\mathrm{tg}\theta+\frac{\rho}{\cos\theta},y\right)dy \tag{6.19}$$

Hence, in order to avoid infinite values and divisions by zero, the Eq. (6.18) can be used in $\theta \in (\frac{\pi}{4}, \frac{3\pi}{4}]$ and Eq. (6.19) in $\theta \in \left(\frac{\pi}{4}, \pi\right) \cup [0, \frac{\pi}{4})$. Therefore, comparing the previous equations (Eq. (6.18) and (6.19)) with the following definitions of the Radon Transform in Cartesian coordinates:

$$G(p,\tau)=\int_{-\infty}^{+\infty} g(x,px+\tau)dx \tag{6.20}$$

and

$$G(r,\eta)=\int_{-\infty}^{+\infty} g(ry+\eta,y)dy \tag{6.21}$$

it becomes clear how the Radon Transform can be divided in two parts:

1) $\mathbf{sin\theta > 1/\sqrt{2}}$

$$G(\rho,\theta) = \frac{1}{|\sin\theta|} G(p,\tau)|_{\substack{p=-\cot\theta \\ \tau=\rho/\sin\theta}} \tag{6.22}$$

2) $\mathbf{sin\theta \le 1/\sqrt{2}}$

$$G(\rho,\theta) = \frac{1}{|\cos\theta|} G(r,\eta)|_{\substack{p=-\mathrm{tg}\theta \\ \tau=\rho/\cos\theta}} \tag{6.23}$$

It is worth noting as this way is the same as dividing the piano in two parts: in the first one are accounted lines of equation $y = px + \tau$ with $-1 < p \le 1$, and in the second one are considered lines of equation $x = ry + \eta$ with $-1 \le r < 1$, where:

$$\begin{cases} r = \frac{1}{p} \\ \eta = -\frac{\tau}{p} \end{cases} \tag{6.24}$$

Quantizing variables involved in Eq. (6.22) and (6.23) we have the following implementable Radon Transform version:

1) $\mathbf{sin\boldsymbol{\theta} > 1/\sqrt{2}}$

$$\begin{aligned} G(\rho_r,\theta_t) &= \frac{\Delta x}{|\sin\theta_t|} \sum_{m=0}^{M-1} g(x_m, p_k x_m + \tau_h)|_{\substack{p_k=-\cot\theta_t \\ \tau_h=\rho_r/\sin\theta_t}} \\ &= \frac{\Delta x}{|\sin\theta_t|} \sum_{m=0}^{M-1} g(x_m, -\cot\theta_t x_m + \rho_r/\sin\theta_t) \end{aligned} \tag{6.25}$$

2) $\mathbf{sin\boldsymbol{\theta} \le 1/\sqrt{2}}$

$$\begin{aligned} G(\rho_r,\theta_t) &= \frac{\Delta y}{|\cos\theta_t|} \sum_{n=0}^{N-1} g(r_k y_n + \eta_h, y_n)|_{\substack{r_k=-\mathrm{tg}\theta_t \\ \eta_h=\rho_r/\cos\theta_t}} \\ &= \frac{\Delta y}{|\cos\theta_t|} \sum_{n=0}^{N-1} g(-\mathrm{tg}\theta_t y_n + \rho_r/\cos\theta_t\,, y_n) \end{aligned} \tag{6.26}$$

Now, it is possible to derive the constraints applying Eq. (6.17) to each variable involved in Eq. (6.22) and (6.23). For the sake of clarity all the mathematical passages are given:

- **Variation of $\boldsymbol{\rho}$ and $\mathbf{sin\boldsymbol{\theta} > 1/\sqrt{2}}$**

$$\begin{aligned} &\max\left\{\left|\frac{\partial}{\partial\rho}\left[-x\cot\theta + \frac{\rho}{\sin\theta}\right]\right| \Delta\rho\right\} \le \Delta y \\ &\to \max\left\{\frac{\Delta\rho}{|\sin\theta|}\right\} \le \Delta y \\ &\to \sqrt{2}\Delta\rho \le \Delta y \end{aligned} \tag{6.27}$$

- **Variation of $\boldsymbol{\theta}$ and $\mathbf{sin}\boldsymbol{\theta} > \mathbf{1}/\sqrt{\mathbf{2}}$**

$$\begin{aligned}
&\max\left\{\left|\frac{\partial}{\partial\theta}\left[-x\cot\theta+\frac{\rho}{\sin\theta}\right]\right|\Delta\theta\right\}\le\Delta y\\
&\rightarrow\max\left\{\left|\frac{x-\rho\cos\theta}{\sin^2\theta}\right|\Delta\theta\right\}\le\Delta y\\
&\rightarrow\max\left\{\left|\frac{x-\rho\cos\theta}{\sin^2\theta}\right|\Delta\theta\right\}\le\max\left\{\left|\frac{x}{\sin^2\theta}\right|\right\}\Delta\theta+\max\left\{\left|\frac{\rho\cos\theta}{\sin^2\theta}\right|\right\}\Delta\theta\le\Delta y\\
&\rightarrow\left(2x_{M-1}+\sqrt{2}\rho_{R-1}\right)\Delta\theta\le\Delta y
\end{aligned}\tag{6.28}$$

- **Variation of x and $\mathbf{sin}\boldsymbol{\theta} > \mathbf{1}/\sqrt{\mathbf{2}}$**

$$\begin{aligned}
&\max\left\{\left|\frac{\partial}{\partial x}\left[-x\cot\theta+\frac{\rho}{\sin\theta}\right]\right|\Delta x\right\}\le\Delta y\\
&\rightarrow\max\{|-\cot\theta|\Delta x\}\le\Delta y\\
&\rightarrow\Delta x\le\Delta y
\end{aligned}\tag{6.29}$$

Considering that the case $\sin\theta \le 1/\sqrt{2}$, is obtainable from the case $\sin\theta > 1/\sqrt{2}$ replacing *x* with *y* we have the final constraints:

$$\begin{cases}
\Delta\rho\le\min\left\{\dfrac{\Delta y}{\sqrt{2}},\dfrac{\Delta x}{\sqrt{2}}\right\}\\
\Delta\theta\le\min\left\{\dfrac{\Delta y}{\left(2x_{M-1}+\sqrt{2}\rho_{R-1}\right)},\dfrac{\Delta x}{\left(2y_{N-1}+\sqrt{2}\rho_{R-1}\right)}\right\}\\
\Delta x=\Delta y
\end{cases}\tag{6.30}$$

However, if the constraints for Cartesian coordinates are solved first and they are replaced with the corresponding Polar variables next, less tight constraints on $\Delta\theta$ are found:

$$\begin{cases}
\Delta\rho\le\min\left\{\dfrac{\Delta y}{\sqrt{2}},\dfrac{\Delta x}{\sqrt{2}}\right\}\\
\Delta\theta\le\min\left\{\dfrac{\Delta y}{\sqrt{2}\rho_{R-1}},\dfrac{\Delta x}{\sqrt{2}\rho_{R-1}},\dfrac{\pi\Delta y}{4x_{M-1}},\dfrac{\pi\Delta x}{4y_{N-1}}\right\}\\
\Delta x=\Delta y
\end{cases}\tag{6.31}$$

Usually to save memory and computation time the reference system origin is taken in the centre of the image so that $\rho_{R-1}=\sqrt{x_{M-1}^2+y_{N-1}^2}$ reducing the previous equalities into:

$$\begin{cases}
\Delta\rho\le\min\left\{\dfrac{\Delta y}{\sqrt{2}},\dfrac{\Delta x}{\sqrt{2}}\right\}\\
\Delta\theta\le\min\left\{\dfrac{\Delta y}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}},\dfrac{\Delta x}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}}\right\}\\
\Delta x=\Delta y
\end{cases}\tag{6.32}$$

6.2.2 Generalized Hough Transform

For the GHT the computations are similar to the GRT ones. Rewriting Eq. (6.14) as:

$$G([v_r],\boldsymbol{v}')=\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}g(x_m,y_n)\Delta x\Delta y\tag{6.33}$$

and applying the same previous reasoning, the following constraints are found:

$$\begin{cases} \max\left\{\left|\frac{\partial v_r}{\partial v_i}\right| \Delta v_i\right\} \leq \Delta v_r \quad, \forall v_i \\ \max\left\{\left|\frac{\partial v_r}{\partial x}\right| \Delta x\right\} \leq \Delta v_r \\ \max\left\{\left|\frac{\partial v_r}{\partial y}\right| \Delta y\right\} \leq \Delta v_r \end{cases} \tag{6.34}$$

For what concerns the line case we have:

$$v_r = \rho = x\cos\theta + y\sin\theta \tag{6.35}$$

hence, replacing Eq. (6.35) in Eq. (6.34) the following results are obtained:

- **Variation of $\boldsymbol{\theta}$**

$$\begin{aligned} &\max\left\{\left|\frac{\partial}{\partial\theta}[x\cos\theta + y\sin\theta]\right| \Delta\theta\right\} \leq \Delta\rho \\ &\to \max\{|-x\sin\theta + y\cos\theta|\Delta\theta\} \leq \Delta\rho \\ &\to \Delta\theta\sqrt{x_{M-1}^2 + y_{N-1}^2} \leq \Delta\rho \end{aligned} \tag{6.36}$$

- **Variation of *x***

$$\begin{aligned} &\max\left\{\left|\frac{\partial}{\partial x}[x\cos\theta + y\sin\theta]\right| \Delta x\right\} \leq \Delta\rho \\ &\to \max\{|\cos\theta|\Delta x\} \leq \Delta\rho \\ &\to \Delta x \leq \Delta\rho \end{aligned} \tag{6.37}$$

- **Variation of *y***

$$\begin{aligned} &\max\left\{\left|\frac{\partial}{\partial y}[x\cos\theta + y\sin\theta]\right| \Delta y\right\} \leq \Delta\rho \\ &\to \max\{|\sin\theta|\Delta y\} \leq \Delta\rho \\ &\to \Delta y \leq \Delta\rho \end{aligned} \tag{6.38}$$

and, to summarize:

$$\begin{cases} \Delta\theta \leq \frac{\Delta\rho}{\sqrt{x_{M-1}^2 + y_{N-1}^2}} \\ \Delta x \leq \Delta\rho \\ \Delta y \leq \Delta\rho \end{cases} \tag{6.39}$$

It is very important to note that results in Eq. (6.39) are tighter than those suggested in [55] where "**peak extension**" and "**peak spreading**" issues are studied (see Fig. 6.2 for an example). These two issues cannot be avoided together but, as still suggested in [55], "peaks extension" are easier to remove whereas "peaks spreading" should be treated with a sliding window filter. It is worth noting that only "peaks extension" in θ direction are allowed if constraints in Eq. (6.39) are respected. Finally, as it is going to be shown subsequently, this type of vote spreading is accounted and easily removed by the final line extraction algorithm.

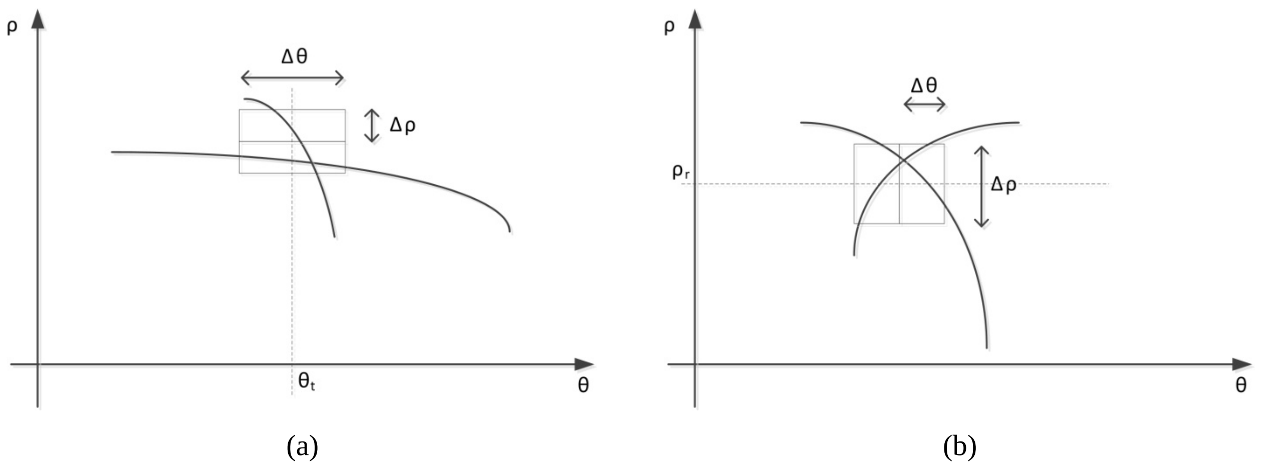


Fig. 6.2 - Problems due to the quantization of the transformed space. (a) "peak spreading" in ρ direction. (b) "peak extension" in θ direction

6.2.3 Constraint comparison with Signal Theory

As previously said, the signal theory puts some limits on signal sampling in order to not lose any information about the original signal. From this point of view, thanks to the linearity property of the Radon Transform, every Radon transformed image $R_g(\rho,\theta)$ can be viewed as the convolution between the original images $g(\rho,\theta)$ and the Radon transform of the impulse $R_\delta(\rho,\theta)$. It is worth noting that the continue definition of the Radon Transform is equivalent to the Hough Transform one, and for this reason these two transforms can be thought as equivalent in this section. Indicating with $F_{R_g}(\omega_\rho,\omega_\theta)$ the Fourier Transform of the Radon transformed image $R_g(\rho,\theta)$, we have:

$$F_{R_g}(\omega_\rho,\omega_\theta) = F_g(\omega_\rho,\omega_\theta)\, F_{R_\delta}(\omega_\rho,\omega_\theta) \tag{6.40}$$

with $F_g(\omega_\rho,\omega_\theta)$ and $F_{R_\delta}(\omega_\rho,\omega_\theta)$ respectively the Fourier Transform of the original image and of the Radon Transformed impulse. Carrying on some computations it can be proved that the support of$F_{R_\delta}(\omega_\rho,\omega_\theta)$ is a "finite-length bowtie" [56] under the condition of finite space-band product of $g(\rho,\theta)$. Indicating respective-ly with ρ_{Max} and ω_{Max} the maximum extension radius of $g(\rho,\theta)$ and the maximum angular frequency of $F_g(\omega_\rho,\omega_\theta)$, the Nyquist limits are [56]:

$$\begin{cases} f_{\rho_c} \geq 2\dfrac{\omega_{\text{Max}}}{2\pi} \\ f_{\theta_c} \geq 2\dfrac{\lceil 1+\omega_{\text{Max}}\rho_{\text{Max}}\rceil}{2\pi} \end{cases} \tag{6.41}$$

where $f_\rho = 1/\Delta\rho$ and $f_\theta = 1/\Delta\theta$ are the sampling frequency of the variable ρ and θ (in case of rectangular grid sampling). Hence, doing some computations:

$$\begin{cases} \Delta\rho \leq \dfrac{\pi}{\omega_{\text{Max}}} \\ \Delta\theta \leq \dfrac{\pi}{\lceil 1+\omega_{\text{Max}}\rho_{\text{Max}}\rceil} \end{cases} \tag{6.42}$$

Now, knowing the maximum variation frequency allowed in a digital image ($f_{\text{Max}} = max\left\{\frac{1}{2\Delta x},\frac{1}{2\Delta y}\right\}$) and presupposing to take the reference system origin in the image centre ($\rho_{\text{Max}} = \sqrt{x_{\text{Max}}^2 + y_{\text{Max}}^2}$) we have:

$$\begin{cases} \Delta\rho \leq \min\{\Delta x, \Delta y\} \\ \Delta\theta \leq \dfrac{\pi}{\left\lceil 1 + \dfrac{\pi}{\max\{\Delta x, \Delta y\}} \sqrt{x_{\text{Max}}^2 + y_{\text{Max}}^2} \right\rceil} \end{cases} \tag{6.43}$$

We can clearly see how close these results are to the ones of Eq. (6.32).

6.3 Road/Runway Extraction Algorithm

As said before, in the line extraction issue the final goal is to transform the binary image, which comes from the edge detector output (or the double thresholding output), in an image which has vectors at the place of binary pixel segments. As can be caught from the Section 6.1 and 6.2, the Radon (Hough) Transform is a tool useful to estimate the parameters of the lines in which an image can be decomposed. Once the parameters have been estimated, the problem moves to what pixels of the image belong to the line with certain parameters. After recognizing these pixels, we have to group them in a structure (e.g. vector) in order to manipulate them easily. This issue is faced by the grouping algorithm and in this section the proposed grouping procedure is going to be described in details.

The Fig. 6.3 summarizes the main blocks which the line extraction algorithm consists of. In that figure the non-maxima suppression and the double thresholding procedures are included in the Edge Detection block. As we can see from the same figure, some information from the edge detection block is passed to the input of the iterative grouping algorithm. Specifically, in addition to the binary image, the gradient sign map and the edge direction map are supplied. These maps have the same size of the binary image, and for each pixel the gradient sign map provides the information related on the gradient sign (positive or negative) whereas the edge direction map let us know which edge direction (among the 4 or 8 considered for the edge detection) is the most probable. The gradient sign map and the edge direction map hold nothing but the information of the gradient direction, so that, in principle, it could be substituted by the phase of it. The choice of using the gradient sign and the edge direction is due to the choice of using the RoA as edge detector. In fact, such statistical filter (Section 3.1.2) does not allow a straightforward gradient computation to be implemented. In particular, the RoA computes the ratio between sample means $\bar{I}_1$ and $\bar{I}_2$ on the two sides of an edge by using a windows oriented as θ, and then it calculates r_θ as:

$$r_\theta = \min\left\{\frac{\bar{I}_1}{\bar{I}_2}, \frac{\bar{I}_2}{\bar{I}_1}\right\} \tag{6.44}$$

It is clear that, by comparing the r_θ obtained by the 4 or 8 direction considered (as happens in the edge direction map), we can only know the edge direction with an ambiguity of π. For this reason, in the gradient sign map the knowledge about the two possibilities $\bar{I}_1 > \bar{I}_2$ and $\bar{I}_2 > \bar{I}_1$ are considered as a binary decision, which is 1 on pixels where $\bar{I}_1 > \bar{I}_2$ and 0 otherwise. The other information provided to the Hough Transform is the step angle $\Delta\theta$[10] between two consecutive directions (e.g. with 4 directions $\Delta\theta = \pi/4$, and with 8 directions $\Delta\theta = \pi/8$). This information together with the edge direction map is very useful in order to speed up the computation time and to heavily improve the estimation performance. In fact, given a point (x_m, y_m) in the image with an edge direction map value $\theta_{\text{Edge}}^{(x_m,y_m)}$, the computation in Eq. (6.5) can be done only with $\theta_t \in \left[\theta_{\text{Edge}}^{(x_m,y_m)} - \Delta\theta, \theta_{\text{Edge}}^{(x_m,y_m)} + \Delta\theta\right]$ to avoid generating noise in the remaining transformed domain interval

[10] This $\Delta\theta$ is not the quantization step also provided as input to Hough Transform, but is related to the orientation of the filtering window.

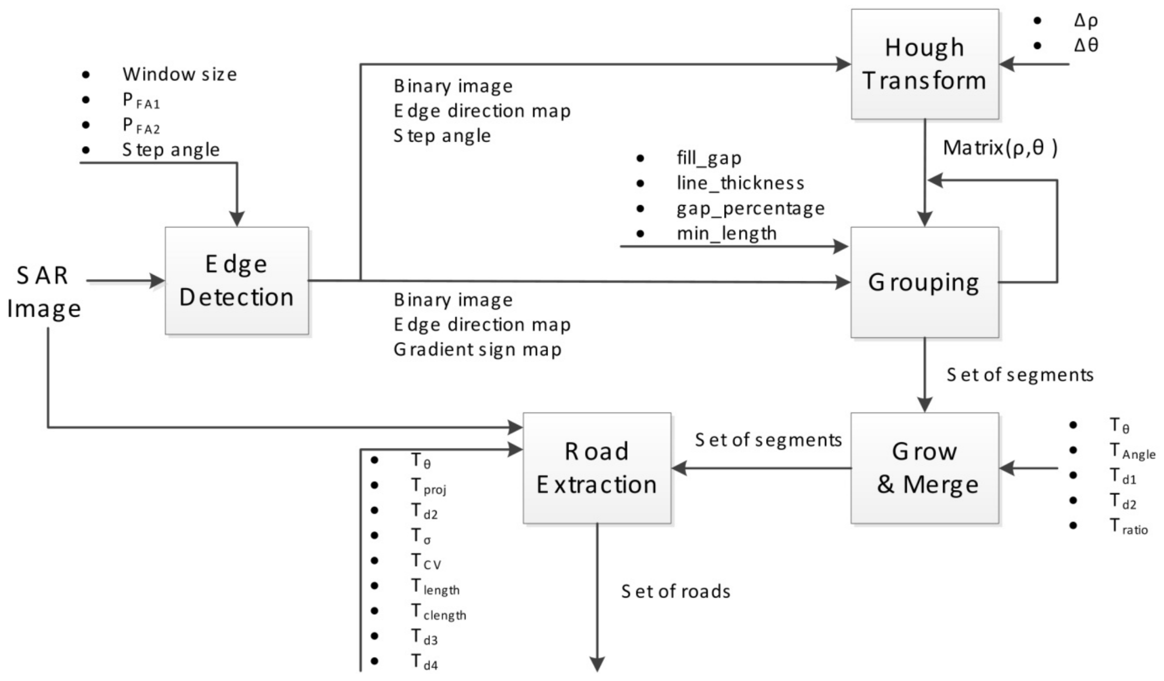


Fig. 6.3 - Main blocks of processing chain

and to permit to speed up the whole computation time. Then, the Hough Transform of the binary image provides in output the Matrix(ρ, θ) used by the **grouping procedure**.

6.3.1 Grouping Step

The grouping algorithm can be summarized in Tab. 6.1. It is worth noting that at the end of the previous procedure the binary image is completely vectorized. As said in Section 6.2, the "peak extension" is completely removed by the step 1.1 of the pseudo code in Tab. 6.1. In fact, the 8-connected neighbourhood of the point (ρ_i, θ_i) is extracted in this step, next every values of Matrix(ρ, θ) in this neighbourhood are evaluated, and finally all values equal to Matrix(ρ_i, θ_i) are deleted. This procedure allows solving completely the problem of "peak extension" defined in [55].

```
1 For each maximum point $(\rho_i,\theta_i)$ such that $Matrix(\rho_i,\theta_i) > 0$
    1.1 Delete $Matrix(\rho,\theta)$ values in the point $(\rho_i,\theta_i)$ and in a controlled neighbourhood of it;
    1.2 Get pixels $\{(x_m, y_n)\}$ in the image domain which lie on the line $(\rho_i,\theta_i)$;
    1.3 Add to the set $\{(x_m, y_n)\}$ pixels close to line $(\rho_i,\theta_i)$ less than line_thickness;
    1.4 For each gradient sign $s \in \{+,-\}$
        1.4.1 Round $\theta_i$ to the nearest edge direction map value $\theta^i_{Edge}$;
        1.4.2 Delete from the set $\{(x_m, y_n)\}$ pixels with edge direction map values $\{\theta^{(x_m,y_n)}_{Edge}\} \neq \theta^i_{Edge}$;
        1.4.3 Delete from the set $\{(x_m, y_n)\}$ pixels with gradient sign map values $\{s^{(x_m,y_n)}\} \neq s$;
        1.4.4 Sort pixels along the line $(\rho_i,\theta_i)$, compute the distance between consecutive pixels and divide the line
            in different segments every time a distance major than fill_gap is encountered;
        1.4.5 For each found segment $L$
            1.4.5.1 Compute the ratio $R_L$ between the pixel number of the segment and the Euclidean segment
                length $Length(L)$;
            1.4.5.2 If $R_L \geq$ gap_percentage and $Length(L) \geq$ min_length
                a Store $L$ in the line vector;
                b Delete the response of the pixels $\{(x_m, y_n)\}$ in $Matrix(\rho,\theta)$;
        1.4.6 end For
    1.5 end For
2 end For
```

Tab. 6.1 - Grouping algorithm pseudo-code.

6.3.2 Grow & Merge Step

The vectorized image could have some segments belonging to the same linear contour separated. For this reason, in order to merge segments together a join procedure is needed. Actually, before applying the join procedure, the properties (i.e. ρ, θ and length values) of the vectors are corrected. In fact, for each group of pixels belonging to the same vector the coefficients ρ and θ of the last square line are computed.

The join algorithm is summarized in Tab. 6.2. In the previous pseudo code the function $Angle(L_i, L_j)$ computes the maximum angle between segment L_i and the segment which links L_i midpoint to one of the two L_j endpoints (see Fig. 6.4), the function $d_1(L_i, L_j)$ is the minimum distance among endpoints of segment L_i and L_j, and the function $d_2(L_i, L_j)$ is the mean distance point-line between L_j endpoints and the line passing through the segment L_i computed as:

$$d_2(L_i, L_j) = \frac{|ax_k + by_k + c|}{\sqrt{a^2 + b^2}} \tag{6.45}$$

```
1 For each segment $L_i$ starting from the longest
    1.1 For each of the two semipianos $sem_k^i$, $k = \{1,2\}$ which divides the segment $L_i$ in equal parts
        1.1.1 Find segments $\{L_j^{(i,k)}\}$ which lay totally on semipiano $sem_k^i$;
        1.1.2 For each segment $L_j \in \{L_j^{(i,k)}\}$
            1.1.2.1 if $Angle(L_i, L_j) > T_{Angle}$ skip this j;
            1.1.2.2 if $\Delta\theta = |\theta_i - \theta_j| > T_\theta$ skip this j;
            1.1.2.3 if $d_1(L_i, L_j) > T_{d_1}$ skip this j;
            1.1.2.4 if $length(L_j)/[length(L_j) + d_1(L_i, L_j)] > T_{ratio}$ skip this j;
            1.1.2.5 if $d_2(L_i, L_j) > T_{d_2}$ skip this j;
            1.1.2.6 if not $isOk_1(L_i, L_j)$ skip this j;
            1.1.2.7 Append $L_j$ in the candidate list $\{L_j^{(i)}\}$;
        1.1.3 end For
        1.1.4 For each segment $L_j \in \{L_j^{(i)}\}$ compute a "matching" factor using $\{\Delta\theta, d_1(L_i, L_j), d_2(L_i, L_j)\}$ and se-
              lect $L_j^{best} \in \{L_j^{(i)}\}$ with the lowest factor;
        1.1.5 Apply recursively this algorithm on $L_j^{best}$ looking for candidates on the same semipiano $sem_k^i$;
        1.1.6 Merge all found segments in $L_i$;
    1.2 end For
    1.3 Compute the least square line among segments merged in $L_i$;
2 end For
```

Tab. 6.2 - Grow & Merge algorithm pseudo-code.

with:

$$\begin{cases} a = \cos\theta_i \\ b = \sin\theta_i \\ c = -\rho_i \end{cases} \tag{6.46}$$

and (x_k, y_k) one endpoints of segment L_j. The function $isOk_1(L_i, L_j)$ computes the projections of the L_j endpoints on the line passing through L_i and returns "true" only if zero or one of the two endpoint projections belongs to the segment L_i (see Fig. 6.5). Practically speaking, the previous check avoids joining to L_i segments L_k which are just at one side of $L_i$and do not form a possible continuation of this latter segment.

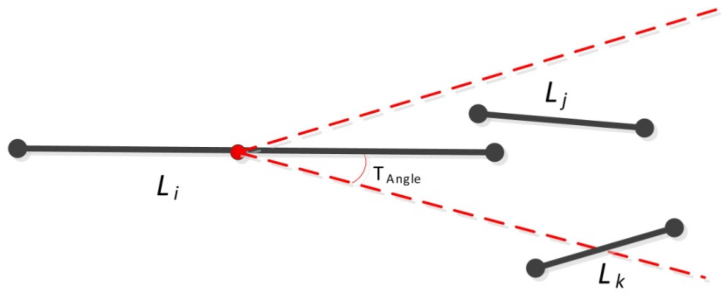


Fig. 6.4 - Meaning of T_{Angle} variable. In this example only segment L_j pass the $Angle(L_i, L_{\{j,k\}})$ check.

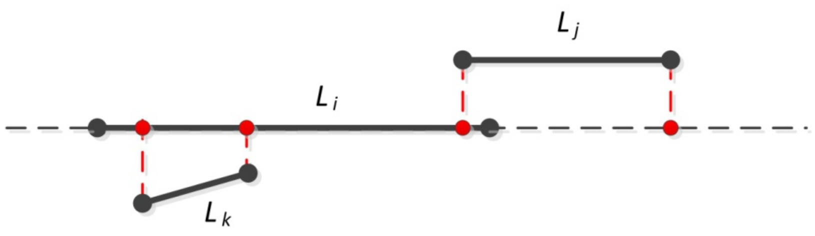


Fig. 6.5 - Meaning of $\boldsymbol{isOk_1}(\boldsymbol{L_i}, \boldsymbol{L_{\{j,k\}}})$ function. In this example only segment $\boldsymbol{L_j}$ pass the $\boldsymbol{isOk_1}(\boldsymbol{L_i}, \boldsymbol{L_{\{j,k\}}})$ check.

Finally, for each segment L_i, the "matching" factor $\boldsymbol{m}$, which is a vector of length n equal to the number of candidate segments $\left\{L_j^{(i)}\right\}$ to be merged with L_i summarizes a measure of "distance" between its n candidates in the following manner:

$$\boldsymbol{m} = \frac{\Delta\boldsymbol{\theta} + \boldsymbol{d_1} + \boldsymbol{d_2}}{3} \tag{6.47}$$

with:

$$\begin{cases} \Delta\boldsymbol{\theta} = \dfrac{\{\Delta\theta_1, \cdots, \Delta\theta_n\}}{\max\{\Delta\theta_1, \cdots, \Delta\theta_n\}} \\ \boldsymbol{d_1} = \dfrac{\{d_1(L_i, L_j), \cdots, d_1(L_i, L_j)\}}{\max\{d_1(L_i, L_j), \cdots, d_1(L_i, L_j)\}} \\ \boldsymbol{d_2} = \dfrac{\{d_2(L_i, L_j), \cdots, d_2(L_i, L_j)\}}{\max\{d_2(L_i, L_j), \cdots, d_2(L_i, L_j)\}} \end{cases} \tag{6.48}$$

From the previous pseudo code it is clear that only geometric information is used to join different segments together. For this reason the join procedure is only a sub optimal solution because it is known that human visual system uses much more information to solve this problem [57]. However, even exploiting this information alone, it can be said that this code is pretty effective for the prefixed scope.

6.3.3 Coupling Step (Road Detection)

This step is used in order to extract roads as two parallel segments which have a low radar cross section (RCS) and a good homogeneity (e.g. low coefficient of variation CV) of the pixels in the middle. Moreover, even single segments can have been extracted as possible road if their length is over a certain threshold. The algorithm can be summarized in Tab. 6.3.

```
1 For each segment $L_i$ starting from the longest
    1.1 For each segment $L_j$, $i \neq j$ starting from the longest
        1.1.1 if $\frac{projectionLength(L_i, L_j)}{\max\{Length(L_i), Length(L_j)\}} < T_{proj}$ skip this j;
        1.1.2 if $\Delta\theta = |\theta_i - \theta_j| > T_\theta$ skip this j;
        1.1.3 if $d_2(L_i, L_j) < T_{d_2}^{low}$ OR $d_2(L_i, L_j) > T_{d_2}^{high}$ skip this j;
        1.1.4 if not $isOk_2(L_i, L_j)$ skip this j;
        1.1.5 Append $L_j$ in the candidate list $\{L_j^{(i)}\}$;
    1.2 end For
    1.3 For each segment $L_j \in \{L_j^{(i)}\}$ compute a "matching" factor using $\{\Delta\theta, d_2(L_i, L_j), Length(L_j), Length(L_i)\}$;
    1.4 Divide the set of segments $L_j \in \{L_j^{(i)}\}$ relying on their position p (left and right semipiano) respect to segment $L_i$;
    1.5 For each of the two possible position p (left and right)
        1.5.1 For each of the segment $L_j^p$ on position p starting from $L_j^p$ with the lowest "matching" factor
            1.5.1.1 Extract the image regions $R_{i,j}^p$ between $L_i$ and $L_j^p$;
            1.5.1.2 Compute the RCS $\sigma_{i,j}^p$ and the coefficients of variation (CV) $CV_{i,j}^p$;
            1.5.1.3 If ( $\sigma_{i,j}^p < T_\sigma$ AND $CV_{i,j}^p < T_{CV}$ )
                a Memorize $L_j^p$ as the candidate segment for the position p, break this cycle and skip this p;
        1.5.2 end For
    1.6 end For
    1.7 If one segment for each positions p (left and right) was found
        1.7.1 Select $L_j \in \{L_j^{left}, L_j^{right}\}$ which has the lowest "matching" factor and couple it with $L_i$;
    1.8 else if only a segment on position p=left was found
        1.8.1 Couple $L_j^{left}$ with $L_i$;
    1.9 else if only a segment on position p=right was found
        1.9.1 Couple $L_j^{right}$ with $L_i$;
    1.10 else if $Length(L_i) > T_{length}$ memorize $L_i$ in single_line list;
2 end For
```

Tab. 6.3 - Coupling algorithm pseudo-code.

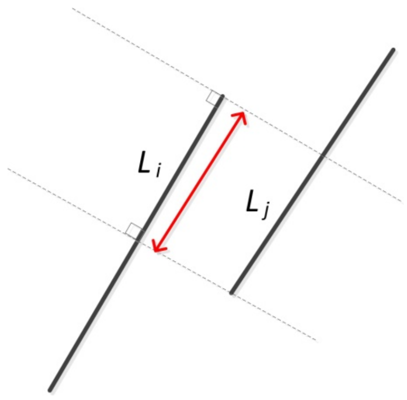


Fig. 6.6 - Example of the projection length between segments L_i and L_j.

In the previous pseudo code the function $ProjectionLength(L_i, L_j)$ computes the projection length of the segment L_j on the segment L_i as in Fig. 6.6. The function $isOk_2(L_i, L_j)$ computes the projections of the L_j endpoints on the line passing through L_i in the same manner as shown in Fig. 6.6, but it does return "true" only if one or both the endpoint projections belong to the segment L_i.

Furthermore, in a very similar manner respect to the join procedure, the "matching" factor vector $\underline{m}$ between the n candidate segments $\left\{L_j^{(i)}\right\}$ and L_i is computed as follows:

$$\boldsymbol{m} = \frac{\Delta\boldsymbol{\theta} + \Delta\boldsymbol{Length} + \boldsymbol{d_2}}{3} \tag{6.49}$$

with:

$$\begin{cases} \Delta\boldsymbol{\theta} = \dfrac{\{\Delta\theta_1, \cdots, \Delta\theta_n\}}{\max\{\Delta\theta_1, \cdots, \Delta\theta_n\}} \\ \Delta\boldsymbol{Length} = \dfrac{\{\Delta Length(L_1), \cdots, \Delta Length(L_n)\}}{\max\{\Delta Length(L_1), \cdots, \Delta Length(L_n)\}} \\ \boldsymbol{d_2} = \dfrac{\{d_2(L_i, L_j), \cdots, d_2(L_i, L_j)\}}{\max\{d_2(L_i, L_j), \cdots, d_2(L_i, L_j)\}} \end{cases} \tag{6.50}$$

where the equality $\Delta Length(L_j) = |Length(L_i) - Length(L_j)|$ is used.

Anyway, after applying the previous algorithm, many false roads can be extracted. The main problems on MSTAR images arise from the extended radar shadows which have low RCS and high homogeneity (low CV) as the roads researched. In order to remove the most false alarms a simple algorithm can be applied. In fact, following the consideration that a road is always linked to another road or it extends throughout the whole image or, eventually, it has a high length, the algorithm in Tab. 6.4 has been used. In the pseudo code in Tab. 6.4, the function $d_3(Image, C_i)$ is the maximum distance between the endpoints of the median segment (segment in the middle of the segment couple C_i) and the two image borders which are crossed by the line passing through the median segment, as shown in Fig. 6.7. Finally, the function $d_4(C_i, C_j)$ computes the minimum distance between median segment endpoints of C_j and the coupled segments C_i.

```
1    For each segment couple $C_i$
     1.1 if $Length(C_i) > T_{C_{length}}$ label $C_i$ as "road" and skip this i;
     1.2 if $d_3(image, C_i) < T_{d_3}$ label $C_i$ as "road" and skip this i;
2    end For
3    For each segment couple $C_i$ labelled as "road"
     3.1 For each segment couple $C_j$ not labelled as "road"
         3.1.1   if $d_4(C_i, C_j) < T_{d_4}$ label $C_j$ as "road";
         3.1.2   Apply recursively the algorithm which starts from step 3 on $C_j$;
     3.2 end For
4    end For
```

Tab. 6.4 - False alarm removal pseudo-code.

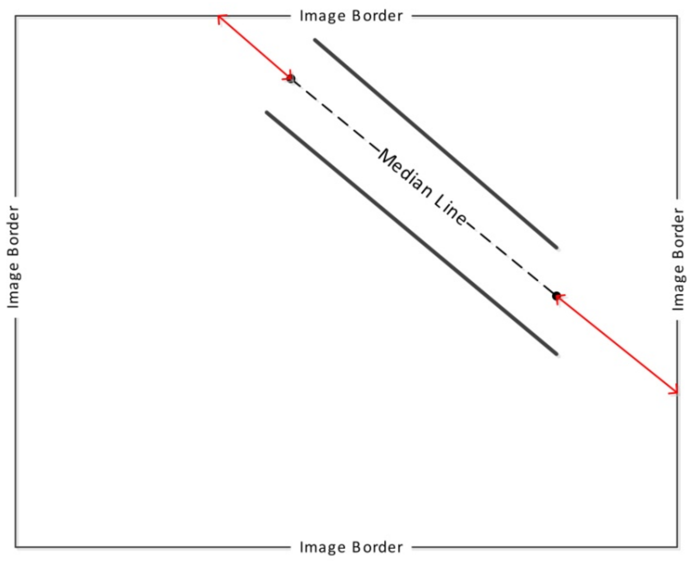


Fig. 6.7 - Example of the function $d_3(Image, C_i)$

6.4 Results

6.4.1 Hough and Radon Transform Comparison

Before showing the results of the road/runway extraction algorithm, a comparison between the performance of the Hough and Radon Transform has been carried out. Using the same notation of the Section 6.2, and comparing Eq. (6.32) and (6.39) when the origin of the reference system is in the centre of the image ($\rho_{R-1} = \sqrt{x_{M-1}^2 + y_{N-1}^2}$), the inequalities in Tab. 6.5 are obtained.

Radon	**Hough**
$\begin{cases}\Delta\rho \le \min\left\{\frac{\Delta y}{\sqrt{2}},\frac{\Delta x}{\sqrt{2}}\right\}\\ \Delta\theta \le \min\left\{\frac{\Delta y}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}},\frac{\Delta x}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}}\right\}\\ \Delta x = \Delta y\end{cases}$	$\begin{cases}\Delta\theta \le \frac{\Delta\rho}{\sqrt{x_{M-1}^2+y_{N-1}^2}}\\ \Delta x \le \Delta\rho\\ \Delta y \le \Delta\rho\end{cases}$

Tab. 6.5 - Sampling limits of the transformed domain parameters

Clearly, because a discrete input image is used, we always have $\Delta x = \Delta y = 1$. Replacing the previous equality in the inequalities shown in Tab. 6.5 we have $\Delta\rho \le 1/\sqrt{2}$ for Radon Transform, and $\Delta\rho \ge 1$for Hough Transform. Hence, no common sampling point exists between them. However, in order to obtain some useful considerations, a performance comparison can be done choosing the nearest sampling point between them, which is:

$$\begin{cases}\Delta\rho_1 = \frac{1}{\sqrt{2}}\\ \Delta\theta^* = \frac{1}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}}\end{cases} \tag{6.51}$$

for the Radon Transform, and:

$$\begin{cases}\Delta\rho_2 = 1\\ \Delta\theta^* = \frac{1}{\sqrt{2}\sqrt{x_{M-1}^2+y_{N-1}^2}}\end{cases} \tag{6.52}$$

for the Hough Transform.

In order to quantitatively measure the estimation error of each transform, a “Monte Carlo” simulation was carried on applying the grouping algorithm on 10 binary images 50x50 where there were 10 lines with random length (at least 5 pixels) and random position in each image. We have made $\Delta\theta$ vary of the amount from $\Delta\theta^*/5$ to $\Delta\theta^*$ and the other parameters are listed in Tab. 6.6. It is worth noting that to speed up the waiting time the binary image is simulated directly and therefore the information arising from edge direction map and gradient sign map are not used. The results are reported in Fig. 6.8.

Parameter	Value
$x_{M-1} = y_{N-1}$	24
$\Delta\theta$	from 0.24° to 1.19° with step 0.24°
$\Delta\rho_1$	$1/\sqrt{2}$
$\Delta\rho_2$	1
`line_thickness`	1
`fill_gap`	3
`gap_percentage`	0.6
`min_length`	4

Tab. 6.6 - Hough and Radon Transform comparison parameters

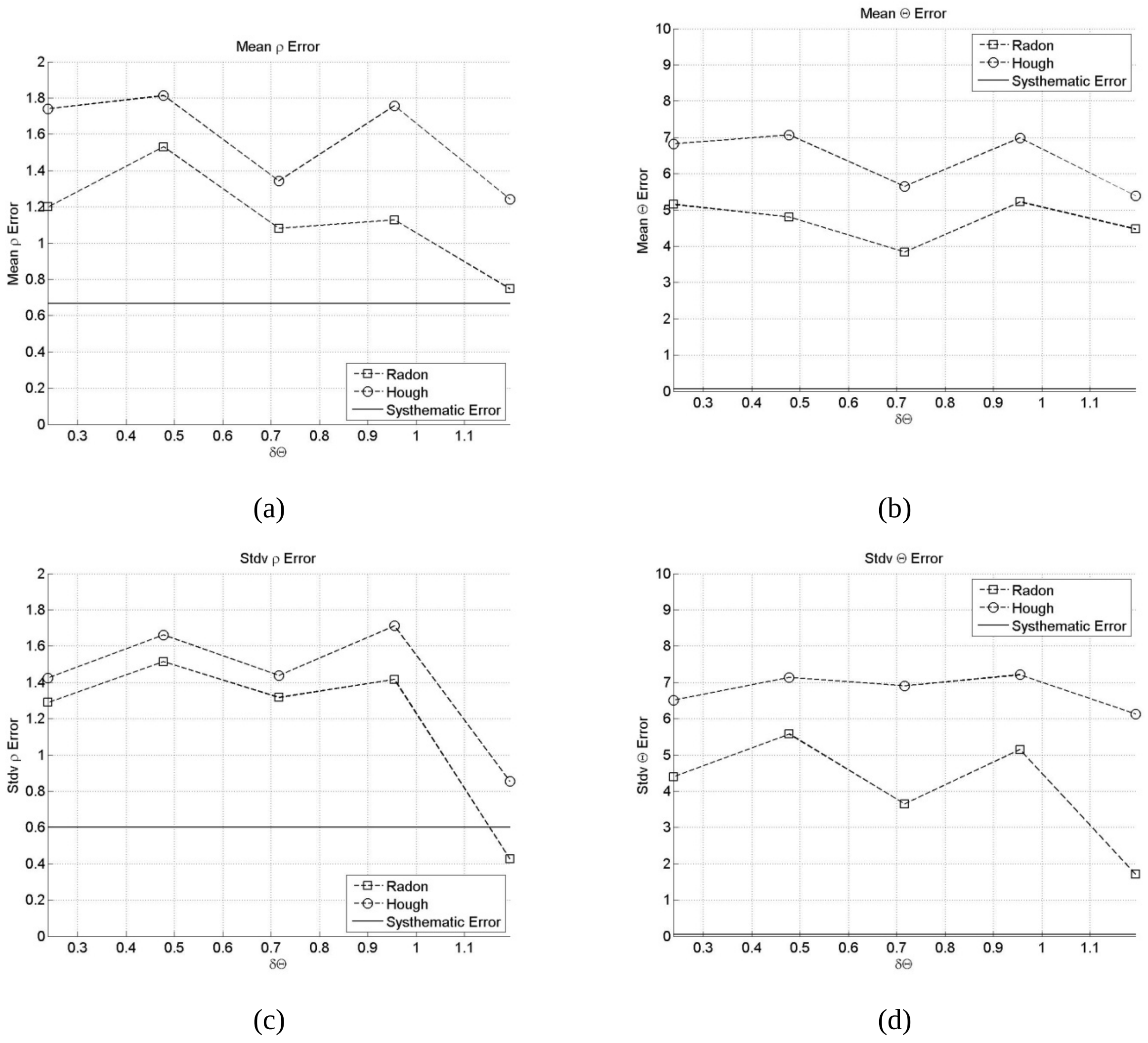


Fig. 6.8 - Results “Monte Carlo” simulation when $\Delta\theta$ varies. (a) Mean ρ error. (b) Mean θ error in degree. (c) Standard deviation ρ error. (d) Standard deviation θ error in degree

As can be seen from these graphs, the Radon Transform yields a lower estimation error both in θ and ρ parameter. In those figures the systematic error is the error between the original parameters and the ones estimated from the corresponding image pixels by a least square error estimate.

Anyway, if we consider the number of correct detections, shown in Fig. 6.9, i.e. the number of times the transformation operator manages to include in a vector the correct original pixels, we see that the Hough

Transform obtains slightly better results. In fact, because the integration line of the Radon Transform have one more degree of freedom than that of the Hough Transform (it can pass in $x_m x\ [y_n - \Delta y/2\,, y_n + \Delta y/2]$ whereas the Hough integration line has to pass in (x_m, y_n)) , the Radon Transform has more probability to fail in the detection (even though when it detects correctly it makes a lower estimation error). This fact can be seen in Fig. 6.10 where the results of Radon and Hough Transform on a simulated image are shown. It can be clearly seen how the Radon Transform usually include in the same vector a major or equal number of pixels than the Hough Transform. Hence, when there are two very close lines, the Radon Transform operates as if had to include all those pixels in the same vector. It is worth noting that applying a least square estimate of the parameters after the Grouping block, for each correct detection we obtain the same estimation error as the systematic one. For the previous reasons, in addiction to accounting the very huge amount of saved time in using the Hough Transform despite of the Radon Transform, the former transform is preferred to be used in the grouping algorithm.

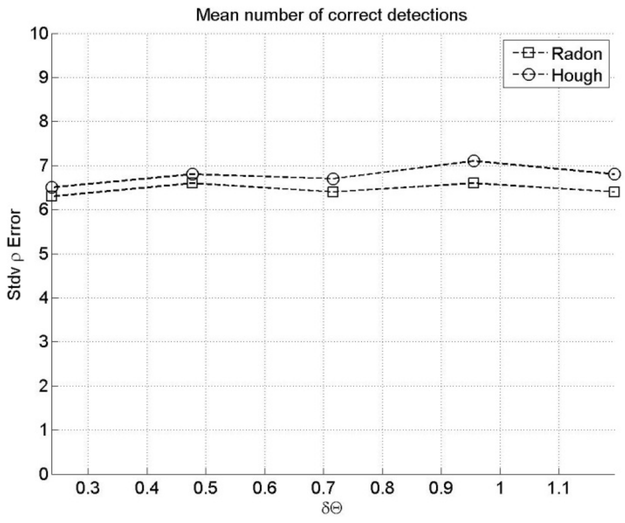
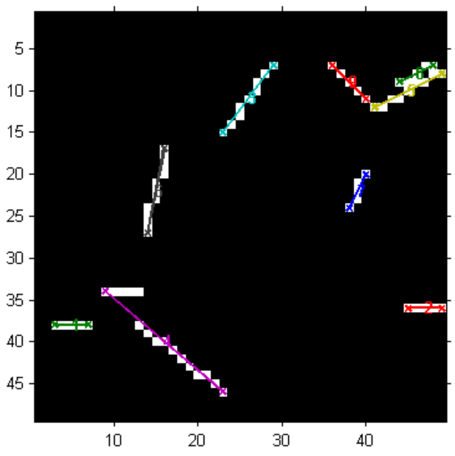
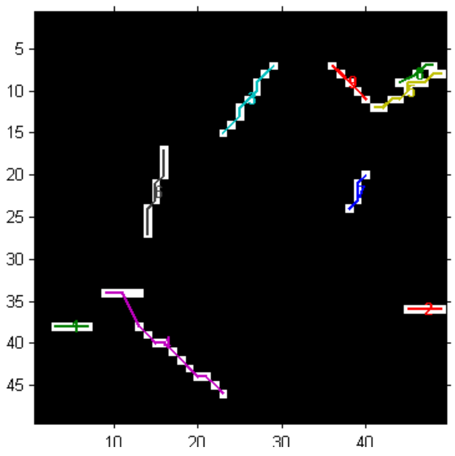


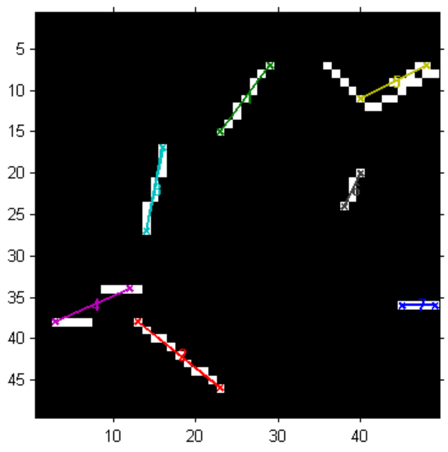
Fig. 6.9 - Mean correct detections



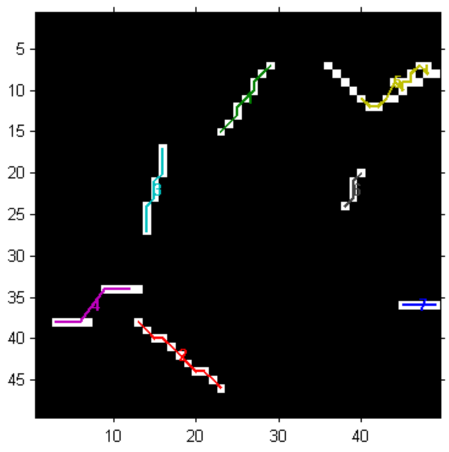
(a)



(b)



(c)



(d)

Fig. 6.10 - Results of the Hough and Radon Transform on a simulated image. (a) Hough Transform segments. (b) Hough Transform segments where the pixels included in a segment are linked each other. (c) Radon Transform segments. (d) Radon Transform segments where the pixels included in a segment are linked each other.

6.4.2 Simulated Image (SIM1)

In this section the line extraction algorithm is going to be explained step by step showing the results on the simple simulated image shown in Fig. 6.11. In that figure there is a classical cross-shaped crossroads, two isolated pieces of road and a square placed just beside a road with intensity lower than the surrounding background. The image intensity consists of independent samples with Gamma pdf whose shape parameter (number of looks) is set to 1.

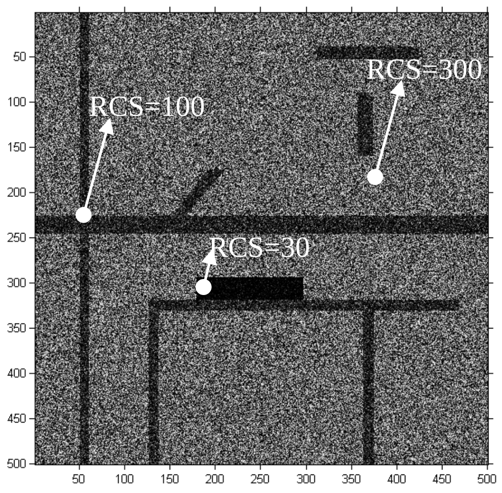


Fig. 6.11 - Simulated image with independent Gamma samples. The mean RCS value is indicated at the top of the arrows

6.4.2.1 Edge Detection Step

In Fig. 6.12 (a) the output of the Edge Detection block is shown when the RoA filter is used with the parameters in Tab. 6.7. Moreover, in Fig. 6.12 (b)-(c) the edge direction map and the gradient sign map are provided. It is worth noticing that the background pixels (RCS=300) are set to a null value for each of the previous maps to allow an easier understanding.

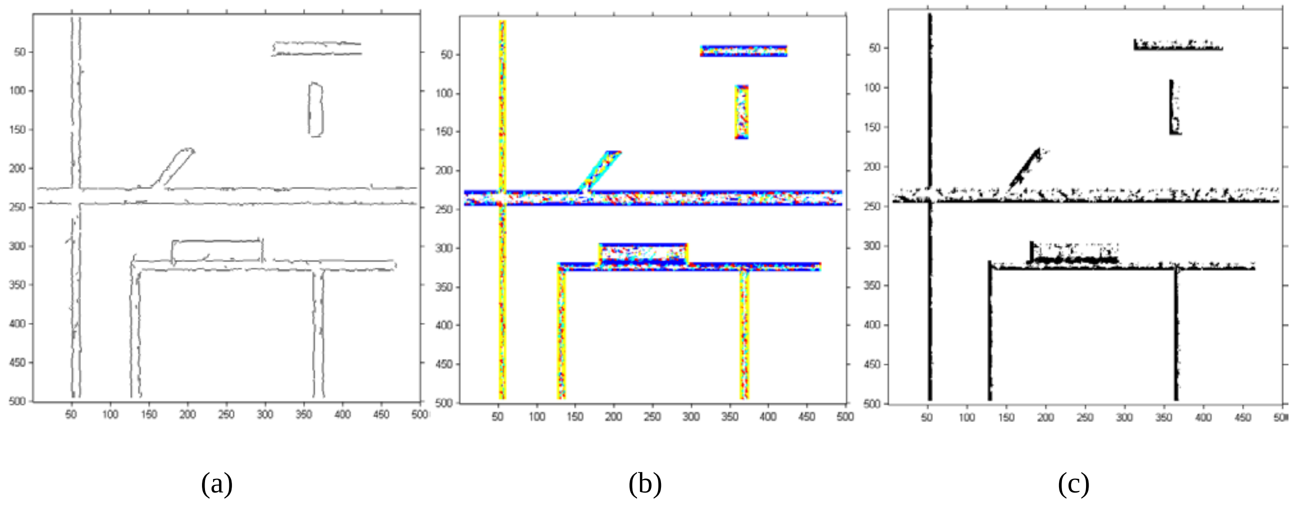


Fig. 6.12 - Results of the Edge Detection block. (a) Edge Detection block output . (b) Edge direction map where background pixels are set to zero. (c) Gradient sign map where background pixels are set to one

Parameter	Value
Window Size	11
P_{FA1}	10^{-9}
P_{FA2}	10^{-1}
Step angle	45°(i.e. 4 different windows)

Tab. 6.7 - Edge Detection block parameters

6.4.2.2 Grouping Step

In Fig. 6.14 (a) the Hough Transform Matrix(ρ, θ) of the binary image in Fig. 6.14 (a) it is shown. The parameter used are listed in Tab. 6.8. As can be seen from Fig. 6.14 (a), thanks to the smart use of edge map information, a lot of noise can be avoided in the transformed domain part which does not involve the peaks. Then, as can be seen in Fig. 6.14 (b)-(c), removing the response of vectorized pixels at each iteration can reduce the noise left drastically.

The result of the Grouping block can be seen in Fig. 6.13 and the used parameters are reported in Tab. 6.9.

Parameter	Value
$\Delta x = \Delta y$	1
$\Delta\rho$	1
$\Delta\theta$	0.0028 (0.16°)

Tab. 6.8 - Hough Transform block Parameters.

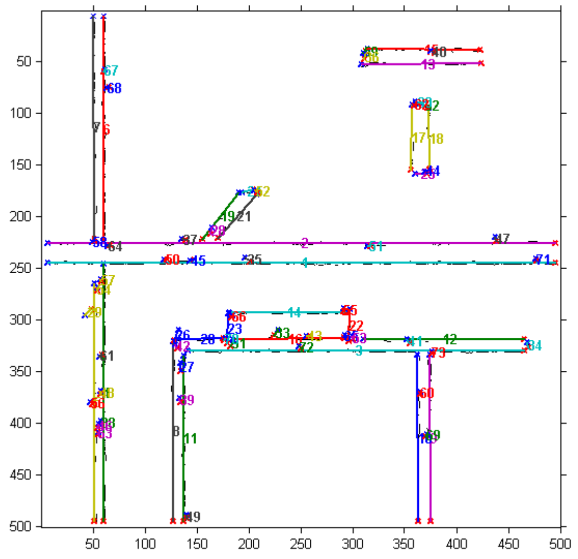
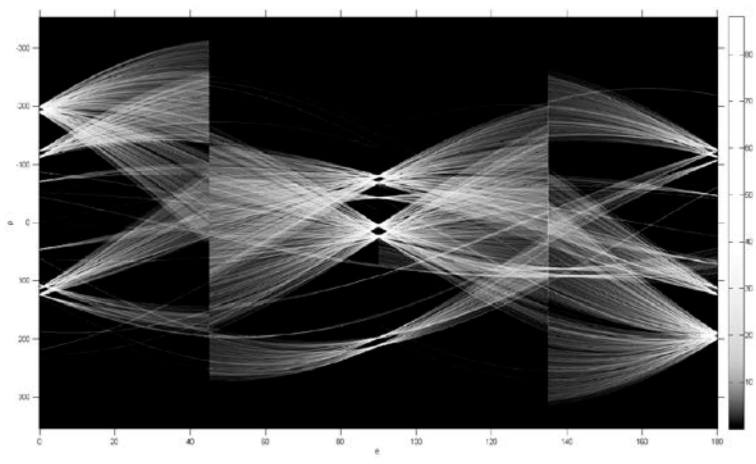
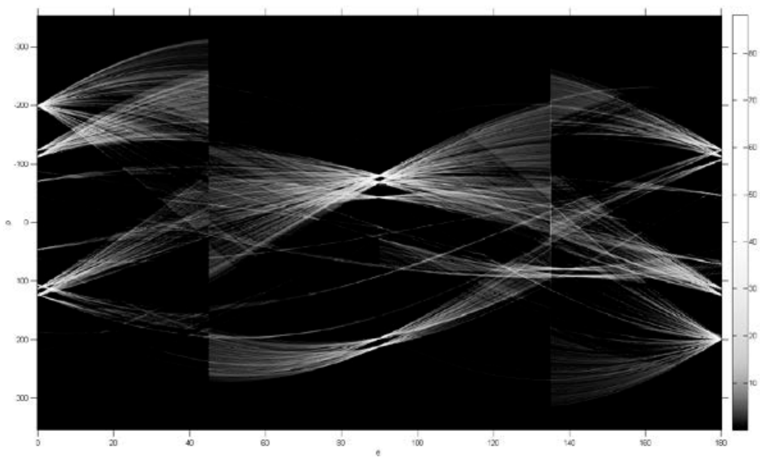


Fig. 6.13 - Results of the Grouping block where an identification number is printed beside each vector.

Parameter	Value
`line_thickness`	2
`fill_gap`	20
`gap_percentage`	0.6
`min_length`	1

Tab. 6.9 - Grouping block parameters.


(a)


(b)

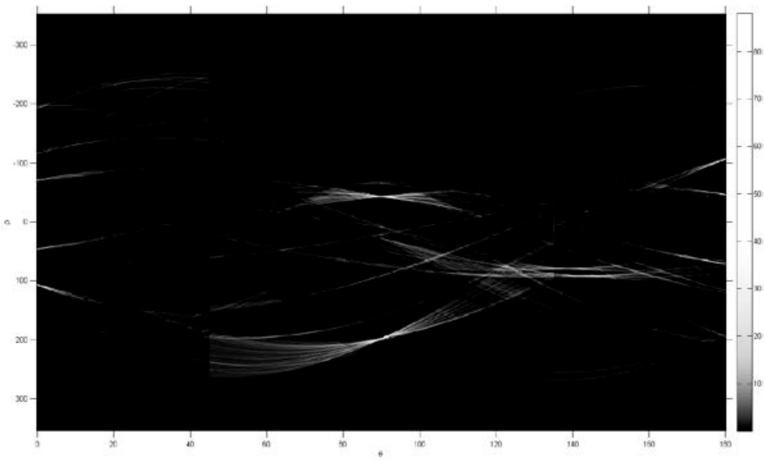

(c)

Fig. 6.14 - Hough Transform Matrix(ρ, θ), with θ (in degree) shown along abscissa axes and ρ shown along the ordinate one. (a) Initial transform. (b) After 4 iterations. (c) After 12 iterations.

6.4.2.3 Grow & Merge Step

In Fig. 6.15 the output of the Grow & Merge block is shown and input parameters are listed in Tab. 6.10. As we can notice comparing Fig. 6.13 and , the yellow segment inside the dashed ellipse of Fig. 6.15 has been obtained merging three different segments of Fig. 6.13. In fact, since the segment in the middle had a different gradient sign value than the segments around it (see Fig. 6.12 (c)), the Grouping algorithm had divided the same line in three different segments correctly.

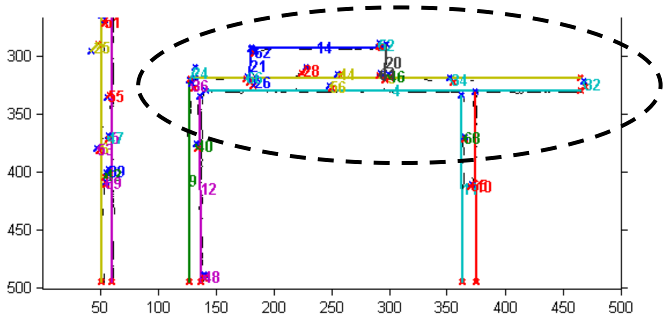


Fig. 6.15 - Results of the Grow & Merge block.

Parameter	Value
T_{Angle}	5°
T_{θ}	15°
T_{d_1}	20 (`fill_gap`)
T_{ratio}	0.6
T_{d_2}	3

Tab. 6.10 - Grow & Merge block parameters.

6.4.2.4 Coupling Step (Road Detection)

As has been previously said in Section 6.3.3, the Road Extraction block is divided in two steps. Firstly, it tries to recognize a road as two parallel lines whose pixels in the middle have a low RCS (less than T_{σ}) and a low CV (less than T_{CV}). Moreover, even single segments are extracted as possible road if their length is over the threshold T_{lenght}. Secondly, it tries to remove false road detections following the consideration that a road is always linked to another road or it extends throughout the whole image or, eventually, it has a length over $T_{C_{lenght}}$. In Fig. 6.16 (a) the result of the first step (parameters are reported in Tab. 6.11) can be seen and the only single segment extracted is shown as a dashed line. In Fig. 6.16 (b) we can see how the second step has been managed to remove the isolated pieces of roads which are linked with no other roads.

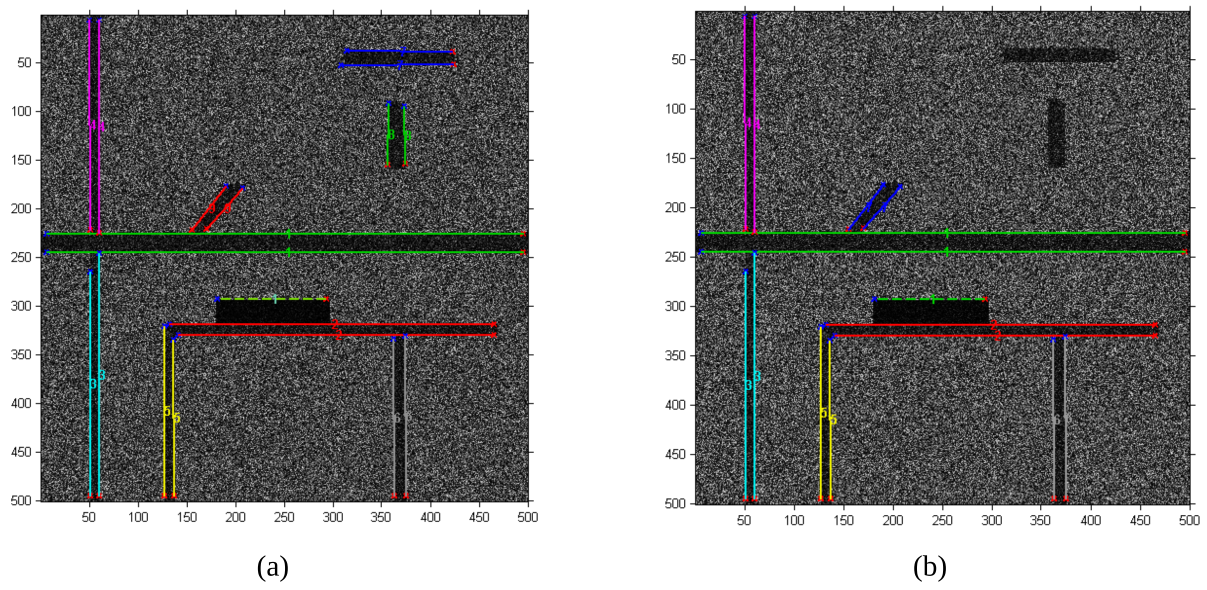


Fig. 6.16 - Results of the Road Extraction block. The dashed line indicate the segments extracted as "single". (a) First step. (b) Second step where the single segment is added.

Parameter	Value
T_{proj}	0.6
T_{θ}	15°
$\left[T_{d_2}^{low}, T_{d_2}^{high}\right]$	[3, 25]
T_{σ}	150
T_{CV}	1.5
T_{lenght}	100
$T_{C_{lenght}}$	150
T_{d_3}	20
T_{d_4}	10

Tab. 6.11 - Road Extraction block parameters.

In order to see how the “matching” factor works let’s see what occur when this factor is computed on candidate segments of the yellow vector inside the dashed ellipse in Fig. 6.15. At the step 1.2 of the first road extraction algorithm, leaving the thresholds T_{proj} out and setting $T_{d_2}^{high} = 30$, the candidate segments found are shown in the Fig. 6.17 and their matching factor value is shown in Tab. 6.12.

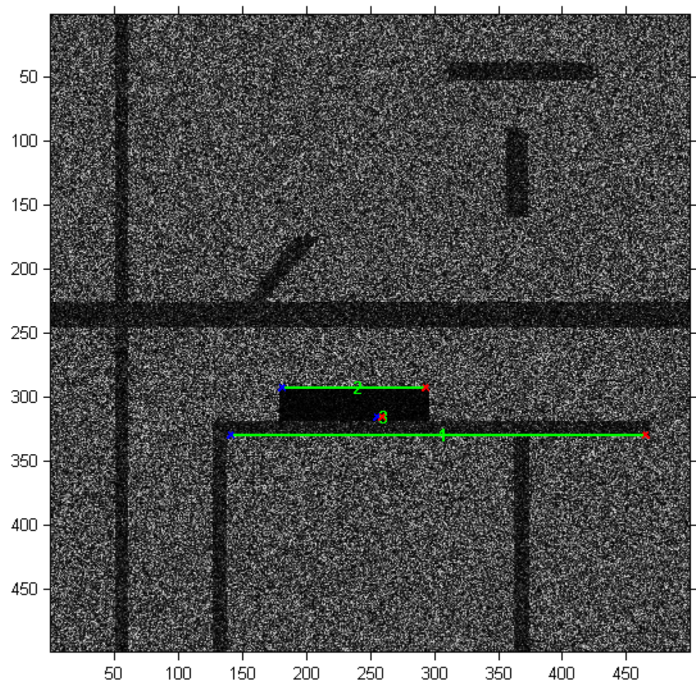


Fig. 6.17 - Candidate segments of the yellow vector inside the dashed ellipse in Fig. 6.15.

ID	p	$\Delta\theta$	$\Delta Length$	d_2	m
1	“left”	0	0.02	0.42	0.15
2	“right”	0	0.67	1	0.55
3	“right”	0	1	0.11	0.37

Tab. 6.12- Matching factor values of the candidate segments.

As we can see from Tab. 6.12 the algorithm has found one segment on the “left” semipiano and two segments on the “right” semipiano. Anyway, after applying the cycle which starts from step 1.5 the segment with ID=3 is chosen as correct candidate among the segments on the right semipiano and the segment with ID=1 is chosen as candidate of the left semipiano. In fact, their respective regions with the yellow vector pass the test on RCS and CV as shown in Tab. 6.13. Hence, now there is one candidate for each position and since the segment with ID=1 has a lower $\boldsymbol{m}$ value it is chosen as the final candidate segment. The final result for the Coupling Step is shown in .Fig. 6.18, where the crossroads are identified by the respective four yellow circles.

ID	RCS	CV
1	101.1	1
3	105.5	1.4

Tab. 6.13- Matching factor values of the candidate segments.

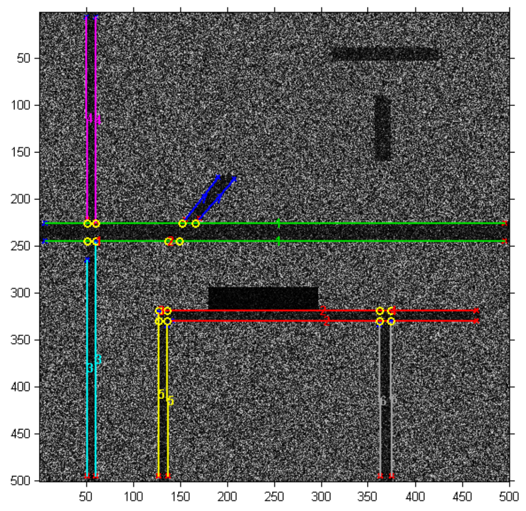
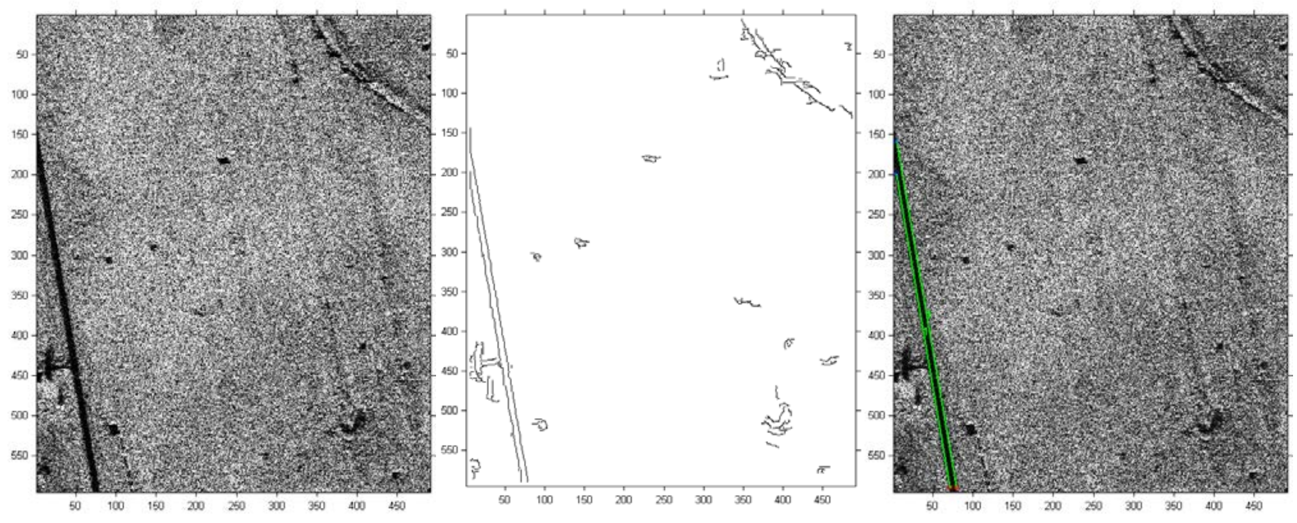


Fig. 6.18 - Results of the Information Extraction block where an identification number is printed beside each vector and each crossroads.

6.4.3 MSTAR Images

In this section results of the whole chain on some MSTAR images are given. The parameters used are the same as those set for the previous simulated image. The only parameter changed is $T_\sigma = 30$.



(a) (b) (c)

Fig. 6.19- Results on MSTAR "HB06173". (a) Original image. (b) Edge detection output. (c) Final result.

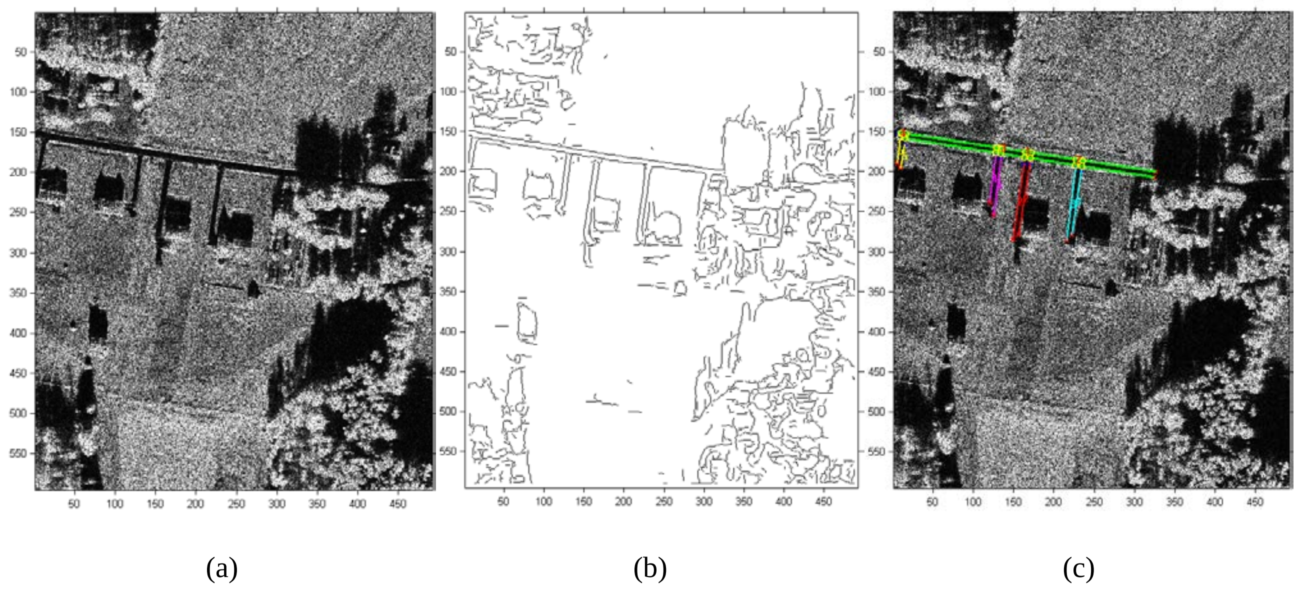


Fig. 6.20 - Results on MSTAR "HB06210". (a) Original image. (b) Edge detection output. (c) Final result.

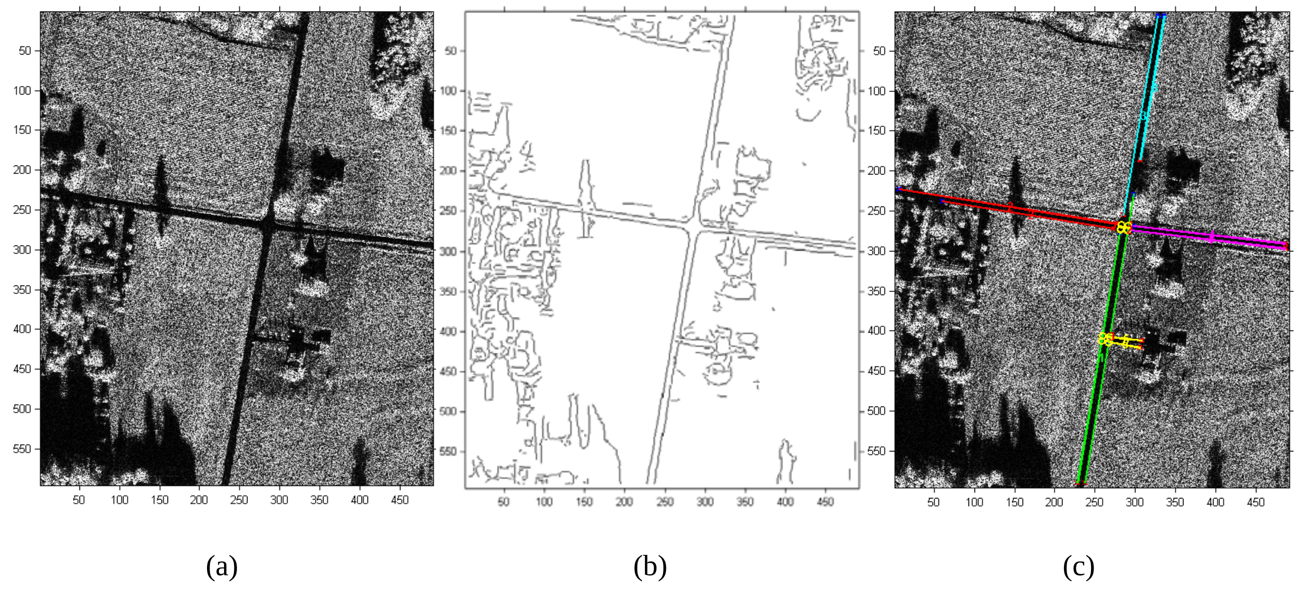


Fig. 6.21- Results on MSTAR "HB06211". (a) Original image. (b) Edge detection output. (c) Final result.